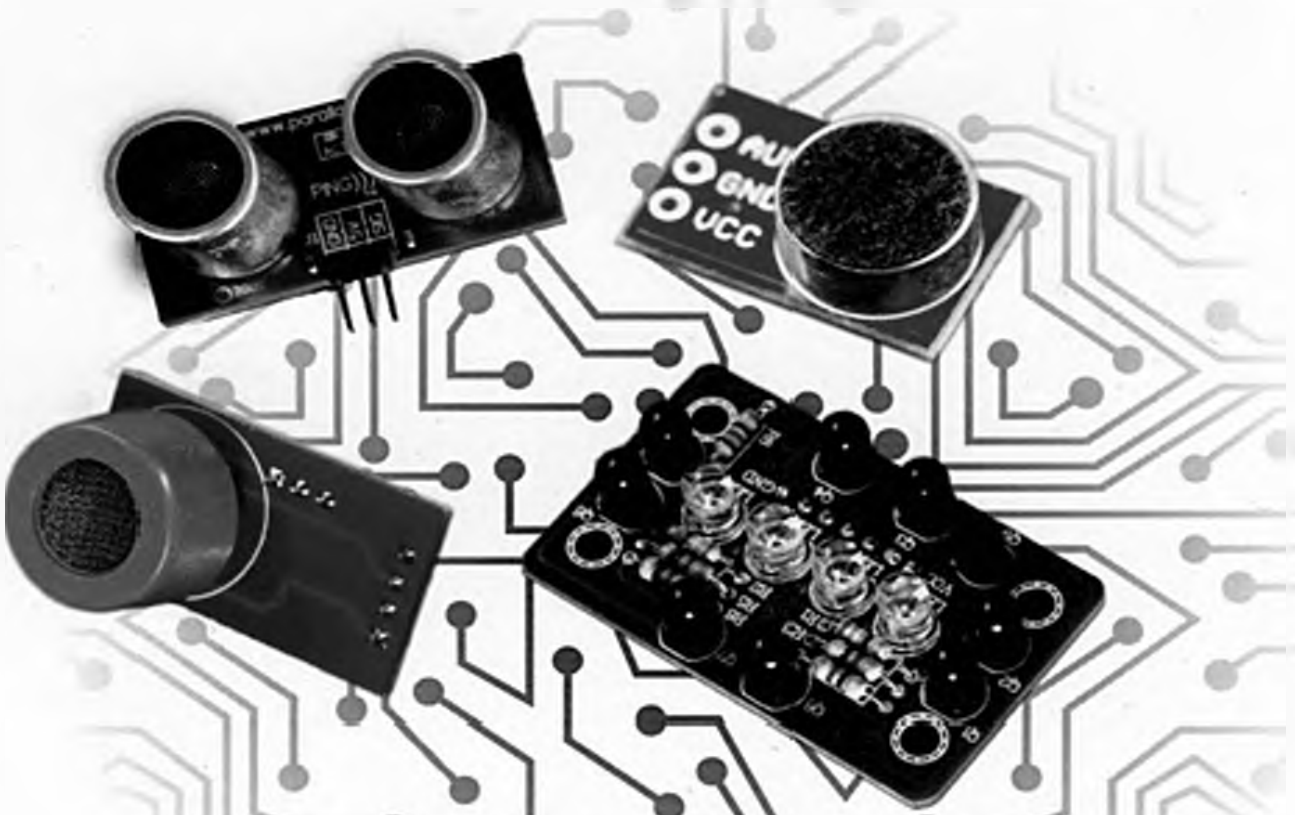
Tero **Karvinen**
Kimmo **Karvinen**
Ville **Valtokari**

Make:

Les capteurs pour Arduino et Raspberry Pi

Tutoriels et projets



DUNOD

Tero Karvinen
Kimmo Karvinen
Ville Valtokari

Make:

Les capteurs pour Arduino et Raspberry Pi

Tutoriels et projets

Traduit de l'américain
par Dominique Maniez

DUNOD

Authorized French translation of material from the English edition of ***Make:Sensors*** ISBN 978-1-4493-6810-4

Adaptation française de la couverture : WIP

5 rue Laromiguière, 75005 Paris
www.dunod.com

ISBN 978-2-10-071793-4

Table des matières

Avant-propos

Grâce à ce livre, vous allez bientôt pouvoir réaliser des objets capables d'analyser leur environnement. Vous allez utiliser des capteurs pour mesurer le monde physique, représenter le résultat sous la forme d'une valeur numérique et agir en fonction de cette valeur.

Les **capteurs** peuvent mesurer la chaleur, la pression, la lumière ou l'accélération et afficher une valeur (22°C, 1 015 millibars ou accélération de 2,3 g ; vous noterez que dans le cas de la lumière on représente la valeur sous la forme d'un booléen, oui/non, au lieu d'une valeur numérique).

Le **microcontrôleur** constitue le cerveau du robot, du système ou du gadget que vous allez construire. Vous allez écrire votre propre logiciel qui sera exécuté sur ce microcontrôleur. Dans ce livre, vous allez travailler avec deux cartes très populaires : Arduino et Raspberry Pi. Ces deux microcontrôleurs facilitent l'écriture de programmes exploitant des composants électroniques.

LAISSEZ S'EXPRIMER VOTRE CRÉATIVITÉ

Si votre passion pour l'électronique a débuté avec la volonté d'apprendre rapidement les bases pour concevoir ensuite vos propres robots, gadgets ou projets, vous avez choisi le bon ouvrage. Ce livre va vous apprendre comment mettre rapidement vos idées en pratique.

La théorie, les compétences et les bases sont utiles pour autant qu'elles sont au service de votre créativité. Sentez-vous libre d'expérimenter vos idées et ayez le courage de publier vos résultats sur le Web.

Chaque chapitre présente un mini-projet qui illustre la manière de combiner différentes technologies. Par exemple, vous allez construire un dôme lumineux qui change de couleur en fonction de son environnement ou bien un détecteur de fumée qui vous avertit par courriel. Ce sont des projets amusants qui constituent aussi de bons points de départ pour vos futures réalisations.

Les compétences que vous allez acquérir avec Arduino sont facilement applicables à des projets professionnels. En ce qui nous concerne, nous avons utilisé un Arduino pour créer un prototype de capteur solaire pour le premier satellite finlandais (Figure 1).

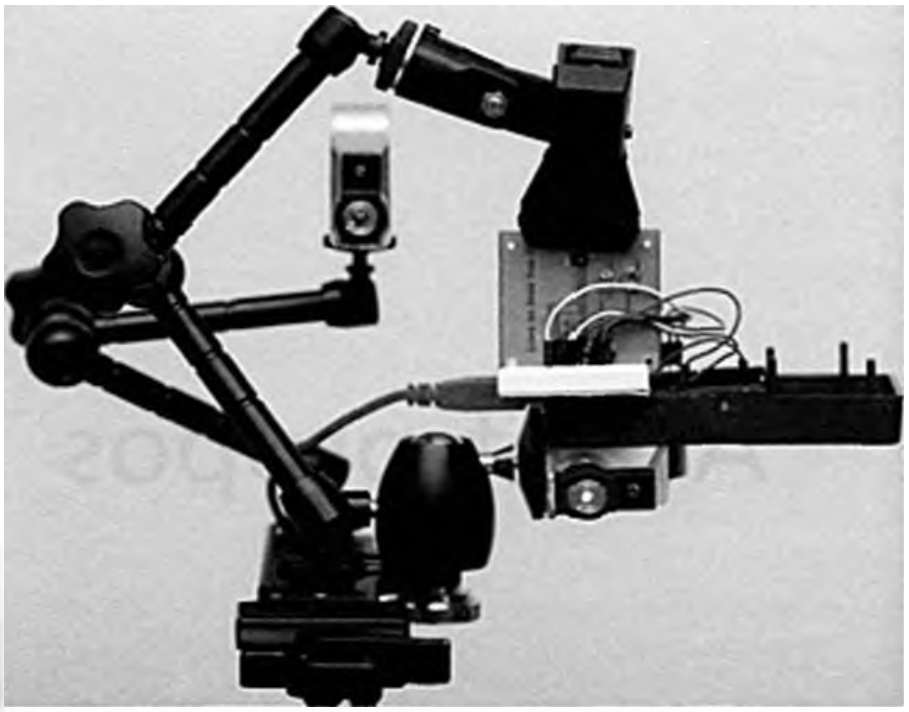


Figure 1 La Finlande lance son premier satellite en 2014.
Nous avons conçu et réalisé le prototype du capteur solaire avec un Arduino

COMMENT LIRE CE LIVRE

À partir d'une idée, vous allez pouvoir rapidement créer un prototype à l'aide de ce livre. Au lieu de passer des heures avec les notices techniques des composants, vous pouvez simplement choisir un capteur, puis utiliser un schéma et un programme prêts à l'emploi. Vous pouvez utiliser les capteurs comme des briques de construction pour votre projet, mais à la différence du Meccano ou du Lego, les possibilités avec Arduino et Raspberry Pi sont presque illimitées.

Si vous savez ce que vous voulez mesurer, vous trouverez facilement le capteur qui convient. L'ouvrage est organisé en chapitres qui couvrent les phénomènes physiques que l'on peut mesurer :

- Distance (Chapitre 3)
- Fumée et gaz (Chapitre 4)
- Toucher (Chapitre 5)
- Mouvement (Chapitre 6)
- Lumière (Chapitre 7)
- Accélération et mouvement angulaire (Chapitre 8)
- Électricité (Chapitre 9)
- Son (Chapitre 10)
- Météo (Chapitre 11)

Vous pouvez aussi utiliser ce livre comme source d'inspiration et le feuilleter pour avoir des idées des technologies disponibles et ainsi penser à de nouveaux projets.

Si vous voulez comprendre comment les capteurs sont connectés à un Arduino et à un Raspberry Pi, vous apprécierez les explications détaillées. Tous les exemples de code sont totalement

indépendants et illustrent parfaitement l'interaction avec le capteur. La compréhension du fonctionnement des capteurs de cet ouvrage vous aidera à appliquer vos compétences à de nouveaux capteurs, même s'ils ne sont pas encore sur le marché.

Les capteurs que nous avons choisis constituent un échantillon aussi varié qu'utile, et leur facilité ou leur difficulté d'emploi n'a pas été un critère de sélection. Cela signifie que vous disposez d'une grande palette de solutions qui relèvent des défis techniques à l'aide de capteurs connectés à un Arduino et un Raspberry Pi.

Dans chaque chapitre, vous trouverez des **expériences**, des **tests d'environnement** et un projet final :

1. Les **expériences** fournissent de brèves instructions sur la manière d'utiliser un capteur avec un Arduino et un Raspberry Pi. Vous pouvez facilement les utiliser comme briques de construction pour vos propres projets ou bien simplement voir comment le capteur fonctionne.
2. Les **tests d'environnement** permettent de tester les capteurs et d'analyser les modifications de l'environnement. Cela vous donne une idée de la manière dont les capteurs perçoivent le monde et fonctionnent réellement.
3. Les capteurs sont plus intéressants quand on réalise quelque chose avec les résultats qu'ils fournissent. Vous allez **réaliser un projet**, construire un appareil basé sur un capteur et apprendre ainsi à utiliser différents périphériques de sortie.

ENTRÉE, TRAITEMENT ET SORTIE

Tous les robots ou gadgets que vous allez construire doivent gérer trois choses : **entrée, traitement et sortie**.

1. Comme la plupart des appareils que vous allez monter ne possèdent ni clavier ni souris, les capteurs constituent les entrées.
2. Le traitement est réalisé par le programme qui s'exécute sur l'Arduino ou le Raspberry Pi. C'est votre programme qui décide de ce qui va se passer.
3. La sortie affecte le monde qui entoure l'appareil. Vous pouvez allumer une LED, activer un servo, ou bien émettre un son. Il s'agit là des principaux types de sorties, mais il y en a d'autres (par exemple, une rétroaction haptique, comme une vibration, l'affichage d'un message sur un écran à encre électronique, ou bien encore l'activation d'un appareil électroménager).

PROTOCOLES

Un **protocole** définit la manière dont un capteur communique avec le microcontrôleur. Le protocole indique le branchement des câbles et la façon dont le code doit traiter les mesures.

Même s'il y a une quantité impressionnante de capteurs différents, il existe un nombre limité de protocoles. Vous apprendrez chacun de ces protocoles au fur et à mesure que vous réaliserez les expériences et les projets, mais voici un aperçu des protocoles que vous rencontrerez (voir aussi le Tableau 1).

Protocole	Ex. de valeur	Arduino	Raspberry Pi (Python)	Ex. de capteurs
Résistance numérique	1 ou 0	digitalRead()	botbook_gpio.read()	Bouton, capteur IR, capteur d'inclinaison, mouvement IR passif.
Résistance analogique	5%, 10%, 23 C	analogRead()	botbook_mcp3002.readAnalog(),chip	Potentiomètre, photorésistance, détecteur d'alcool MQ-3, détecteur de gaz MQ X (fumée, hydrocarbone, CO...), pression FlexiForce, flamme KY-026, couleur HDJD-S822-QR999, température LM35, humidité
Durée d'impulsion	20 millisecondes	pulseIn()	gpio.pulseInHigh()	Ping et distance ultrasons. HC-SR04, accélération MX2125.
Port série	A9B3C5B3C5	Serial.read()	pySerial.read()	Scanner GT-511C3
I2C	(2,11 g, 0,0 g, 0,1 g), valeurs très précises	Wire.h	smbus	Wii Nunchuk, accéléromètre MPU 6050, pression atmosphérique GY65
SPI	57 °C, valeurs très précises	Bit-banging	spidev	Convertisseur analogique-numérique MCP3002
Bits encodés en impulsions très courtes	53 %	Bit-banging	Bit-banging	Humidité DHT11

Tableau 1 Protocoles des capteurs, du plus simple au plus complexe.

Résistance numérique

Certains capteurs fonctionnent comme un bouton et possèdent deux états : *on* ou *off*. Ces capteurs sont faciles à lire. L'état *on* est représenté quand une tension appelée HIGH est appliquée à la broche d'entrée du microcontrôleur. Il s'agit habituellement d'une tension de 3,3 volts ou 5 volts en fonction de la carte que vous utilisez.

Résistance analogique

Les capteurs de résistance analogiques modifient leur résistance en réponse à un changement physique (comme quand on tourne le bouton d'un cadran). Arduino et Raspberry Pi indiquent les changements de résistance en mesurant la tension qui passe au travers du capteur. Par exemple, vous pouvez tourner un potentiomètre pour diminuer ou augmenter la résistance. Ces capteurs de résistance analogique sont très faciles à réaliser avec un Arduino. Le Raspberry Pi a besoin d'une puce externe pour mesurer les valeurs analogiques. Vous apprendrez à utiliser le convertisseur analogique-numérique pour mesurer la résistance avec un Raspberry Pi dans l'expérience sur le *Suivi de mouvement à l'aide d'infrarouge (œil composite IR)* au chapitre 3. La plupart des capteurs d'entrée analogiques indiquent leur valeur en utilisant la résistance, si bien qu'il s'agit de capteurs de résistance analogique.

Durée d'impulsion

Certains capteurs indiquent leur valeur par une durée d'impulsion, ou bien la période pendant laquelle la broche est maintenue à la valeur HIGH. On utilise des fonctions comme pulseIn() ou gpio.pulseInHigh() pour lire la longueur de l'impulsion. Comme ceci est géré par une fonction,

vous n'avez pas à vous embarrasser des opérations de bas niveau du microcontrôleur comme les interruptions puisque tout est pris en charge par une bibliothèque.

Port série

Un port série fait circuler des caractères de texte entre deux appareils. C'est la même technique que l'ordinateur emploie quand il communique avec l'Arduino *via* le port USB. Vous vous familiariserez avec le port série quand vous afficherez des messages sur le Moniteur série de l'Arduino dans plusieurs projets.

I2C

I2C est un protocole industriel standard populaire. On le trouve couramment dans des ordinateurs et des joysticks bien connus comme le Nunchuk de la console Wii. I2C autorise la connexion de 128 appareils sur les mêmes câbles. Dans cet ouvrage, nous vous fournissons du code prêt à l'emploi et des circuits pour deux capteurs utilisant I2C.

SPI

SPI est un autre protocole industriel standard. Le code livré dans cet ouvrage vous permettra d'utiliser facilement un convertisseur analogique-numérique sur le Raspberry Pi. La création de votre propre code à partir de zéro pour de nouveaux appareils utilisant SPI nécessitera cependant plus de travail.

Bit-banging

Parfois, un capteur est suffisamment rare pour qu'il ne fonctionne pas avec un protocole standard. Dans ce cas-là, vous devez élaborer votre propre code pour communiquer avec ce capteur. On appelle souvent cette opération bit-banging, car on manipule le signal du capteur la plupart du temps au niveau du bit. Vous trouverez un exemple de cette technique dans l'expérience *Est-ce que c'est humide ?*.

Au fur et à mesure que vous manipulerez les capteurs, vous vous familiariserez avec ces protocoles, mais si vous êtes pressé de placer de nouveaux capteurs dans vos robots et vos appareils innovants, vous pouvez vous contenter d'utiliser le code fourni dans ce livre puis aborder les détails ultérieurement.

APPROPRIEZ-VOUS LES CHOSES

Il est difficile de trouver des circuits et des composants bruts qui conviennent parfaitement à nos envies et vous devrez souvent faire preuve d'imagination pour rassembler tous les composants d'un projet dans un packaging élégant.

Cet ouvrage donne un exemple de packaging pour chaque projet, mais il n'est pas nécessaire de suivre nos instructions aveuglément. Vous devez tester différents matériaux et employer différents outils.

Pourquoi ne pas essayer du carton, du tissu ou une impression 3D (Figure 2) ?

L'expérimentation et l'apprentissage de nouvelles techniques, comme le soudage ou l'emploi du caoutchouc, rend le processus de création plus intéressant (voir la Figure 3).

Nous utilisons aussi beaucoup de matériel recyclé dans nos projets. Non seulement ils sont bon marché (voire gratuits), mais ils donnent aussi un aspect unique à un projet.

Figure 2 Bender 3D. Photo extraite de la collection Ars Electronica.

Figure 3 Moule d'une tête de gorille animée et peau en latex réalisée à partir du moule.

ACHAT DES COMPOSANTS

Si vous avez besoin de composants de bonne qualité sans prise de tête, choisissez une enseigne réputée, de préférence en Europe ou aux États-Unis. Si vous voulez des composants peu onéreux, tournez-vous vers l'Asie.

Parmi les enseignes de qualité qui vendent aux passionnés d'électronique, on peut lister Maker Shed, SparkFun, Parallax et Adafruit. Maker Shed est l'enseigne de l'éditeur de la version américaine de ce livre. SparkFun commercialise de nombreuses platines d'évaluation qui nécessitent des soudures. Parallax a créé Basic Stamp, la génération précédente de microcontrôleurs pour les *makers*[1]. Adafruit possède un grand catalogue et produit de nombreux articles. Les sites web

1. « Ceux qui fabriquent », les nouveaux adeptes de la mouvance DIY (*Do it Yourself*, c'est-à-dire ceux qui cherchent à construire par eux-mêmes).

de SparkFun et d'Adafruit contiennent beaucoup d'information sur leurs composants, notamment des tutoriels.

À l'heure actuelle, de grands distributeurs comme Element14 et RS electronics ont également investi le marché des *makers*. Il est devenu plus facile de s'y retrouver dans leur immense catalogue car ils ont créé des rubriques pour Arduino et Raspberry Pi.

Si vous voulez des articles spéciaux ou très bon marché, il faut aller sur le continent asiatique. DealExtreme (http://dx.com) est en ce moment très populaire. La livraison est lente et la qualité variable, mais les prix sont très bas et il y a beaucoup de choix. AliExpress (http://www.aliexpress.com) qui est aussi une boutique asiatique vaut également le détour.

NOTE DE L'ÉDITEUR

Vous trouverez ci-joint une liste non exhaustive de revendeurs de composants présents en France :

- **GO TRONIC**
  35 ter Route nationale, BP 45, 08110 Blagny.
  Tél. : 03 24 27 93 42
  http://www.gotronic.fr/
- **HOLDELEC** (ventes exclusivement réservées aux professionnels)
  Avenue de la Victoire, 59117 Wervicq-Sud
  contact@holdelec.fr
  http://holdelec.net/
- **Kubii.fr – Distributeur officiel de Raspberry Pi pour le compte de Farnell en France**
  69110 Sainte-Foy-les-Lyon
  Tél. : 09 63 25 11 39
  http://www.kubii.fr/
- **LEXTRONIC**
  36/40 rue du Général de Gaulle, 94510 La-Queue-en-Brie
  Tél. : 01 45 76 83 88
  www.lextronic.fr
- **SELECTRONIC**
  16 rue Jules Verne, ZAC Orée Golf, 59790 Ronchin.
  http://www.selectronic.fr/

CONVENTIONS UTILISÉES DANS CE LIVRE

Les conventions typographiques suivantes sont utilisées dans cet ouvrage :

- Le bleu indique de nouveaux termes, les URL, les adresses électroniques, les titres des figures et des exemples.
- *Le bleu italique* indique les noms de fichiers et de répertoires.
- La couleur orange est utilisée pour faire référence à des éléments de programmation, comme les noms des variables ou des fonctions, les bases de données, les types de données, les variables d'environnement, les instructions et les mots-clés.
- La couleur orange en gras indique les commandes ou le texte qui doit être saisi littéralement par l'utilisateur.

UTILISATION DES EXEMPLES DE CODE

Vous pouvez télécharger le code source de cet ouvrage sur le site des éditions Dunod (www.dunod.com/contenus-complémentaires/9782100717934).

Dans le texte de ce livre, il est souvent fait référence au site web des auteurs où vous pouvez également télécharger le code des exemples et projets : http://makesensors.botbook.com/

Nous attirons cependant votre attention sur le fait que le site des auteurs classe les exemples par plateforme (Arduino ou Raspberry Pi) alors que nous avons opté pour un classement par chapitre. De plus, cette traduction de l'ouvrage américain *Make: Sensors*, publié chez O'Reilly ne reprend pas le chapitre 9 de l'édition originale. Si vous téléchargez les exemples de code sur le site des auteurs, vous trouverez donc des programmes qui ne figurent pas dans la traduction française (tous les exemples du chapitre 9 consacré au contrôle de l'identité). Pour faciliter le repérage des exemples de code, nous avons conservé les noms originaux des programmes, mais nous avons traduit en français tous les commentaires.

> Si vous n'avez aucune expérience de la programmation avec Arduino et Raspberrry Pi, nous vous conseillons de lire les deux ouvrages d'initiation de Christian Tavernier, parus aux éditions Dunod, *Arduino – Maîtrisez sa programmation et ses cartes d'interface (shields)* et *Raspberry Pi – Prise en main et premières réalisations*.

RÉUTILISATION DES PROGRAMMES

Ce livre est censé faciliter la réalisation de vos projets. En général, vous pouvez utiliser le code de cet ouvrage dans vos programmes et votre documentation. Vous n'avez pas besoin de nous contacter pour nous demander la permission, à moins que vous ne reproduisiez une quantité importante de code. Par exemple, l'écriture d'un programme qui utilise plusieurs portions de code de ce livre ne nécessite aucune autorisation. En revanche, la vente ou la distribution d'un CD-ROM d'exemples extraits de ce livre nécessite une autorisation. Si vous répondez à une question en citant cet ouvrage et en reprenant un exemple de code, vous n'avez pas besoin de demander la permission, mais si vous incorporez une grande quantité d'exemples de code dans votre produit ou votre documentation, il vous faut demander une autorisation.

Nous apprécions, sans que cela ne revête un caractère obligatoire, que lorsque vous citez notre ouvrage vous mentionniez ses références exactes qui comprennent le titre, l'auteur, l'éditeur et l'ISBN. Pour cette édition de l'ouvrage, voici comment le citer correctement :

Make: Sensors by Tero Karvinen, Kimmo Karvinen, and Ville Valtokari.
Copyright 2014 Tero Karvinen, Kimmo Karvinen, and Ville Valtokari, 978-1-449-36810-4.

Si vous pensez que votre utilisation des exemples de code de ce livre n'est pas couverte par les principes que nous venons d'énoncer, veuillez nous contacter à bookpermissions@makermedia.com.

REMERCIEMENTS

Les auteurs souhaitent remercier Hipsu, Marianna, Nina, Paavo Leinonen et Valtteri.

1 Raspberry Pi

Nous vous recommandons de commencer avec le Raspberry Pi Model B, qui comprend une prise Ethernet et deux ports USB pour une souris et un clavier. Cela facilitera votre initiation.

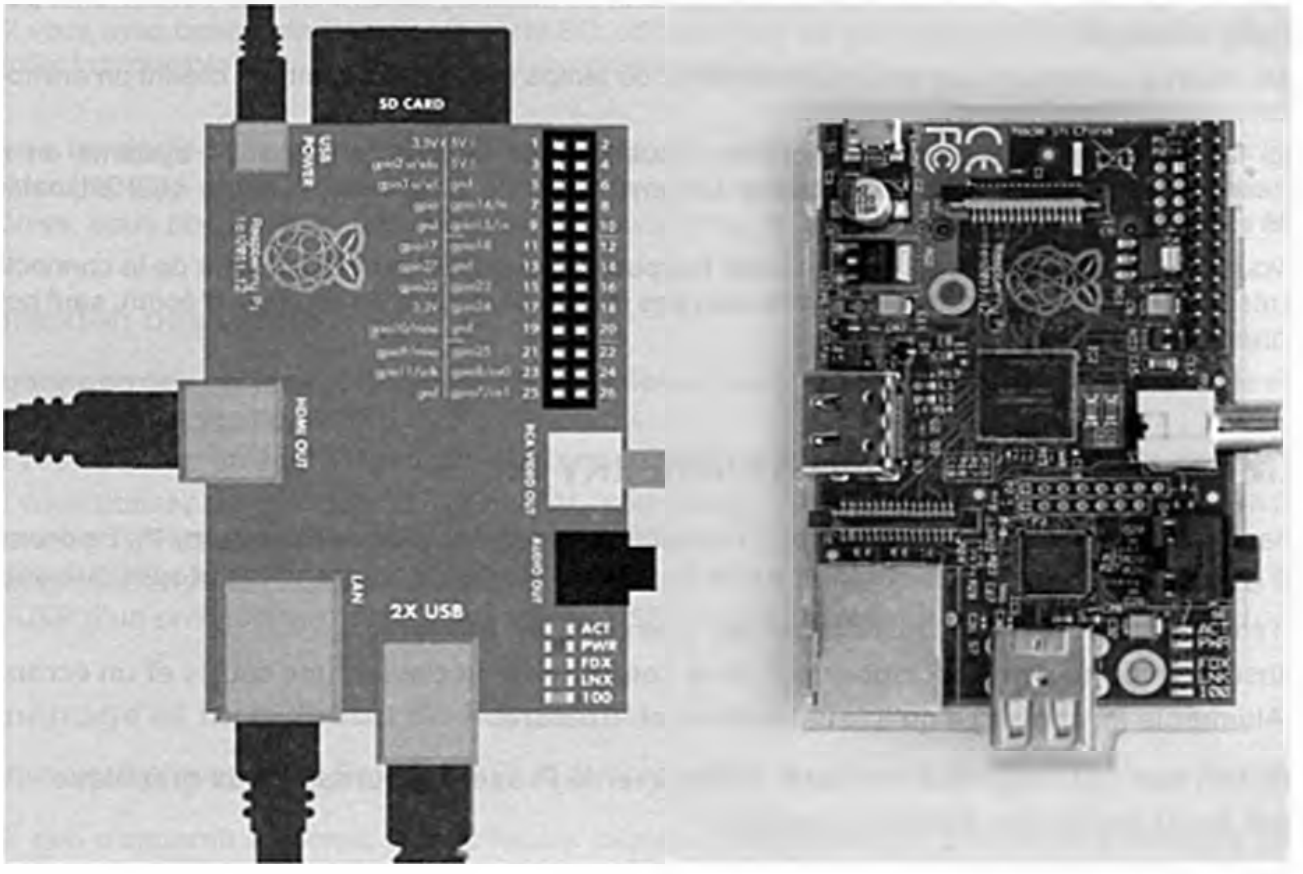


Figure 1.1 Connexions du Raspberry Pi

À moins d'acheter votre Raspberry Pi dans un kit, il n'est probablement pas livré avec un boîtier, mais en le posant directement sur la table, vous passerez pour un geek. Si vous avez accès à une imprimante 3D, vous trouverez de nombreux modèles de boîtiers à confectionner sur http://www.thingiverse.com.

Une carte mémoire SD de 4 Go est suffisante pour stocker le système d'exploitation, mais une carte plus grande s'usera moins vite car il y aura plus d'espace à allouer au *wear-leveling* (technique qui répartit équitablement les écritures sur l'espace de stockage disponible) ; il est donc préférable d'avoir une carte d'au moins 8 Go.

Le Raspberry Pi peut piloter un écran HD et même envoyer le son par la prise HDMI. La plupart du temps, un téléviseur HD conviendra parfaitement à votre Pi.

Pour démarrer, il vous faut un clavier et une souris qui se brancheront justement sur les deux ports USB du Raspberry Pi.

> Si vous voulez ajouter un adaptateur WiFi USB, il vous faudra un hub USB alimenté. Consultez http://elinux.org/RPi_USB_Wi-Fi_Adapters pour une liste des adaptateurs WiFi compatibles avec le Raspberry Pi. On configure le WiFi sur le Pi en faisant un double-clic sur l'icône « Configuration WiFi du Bureau » après avoir installé le système d'exploitation et démarré dans l'environnement graphique.

L'ORDINATEUR À 35 EUROS LE PLUS CHER DU MONDE ?

Si vous devez acheter les câbles, le clavier, la souris et l'écran, cela risque de vous coûter le prix de plusieurs Raspberry Pi. Si vous n'avez pas déjà tout ce matériel, cela va finir par revenir cher pour un petit ordinateur.

Vous noterez cependant que vous économiserez du temps (donc de l'argent) en créant un environnement de développement confortable.

Plus tard, lorsque votre projet fonctionnera, vous pourrez facilement alléger le système en ne conservant que les éléments nécessaires. Comme on dit, le Raspberry Pi est le seul ordinateur à 35 euros qui en coûte une centaine.

Si vous décidez de communiquer avec votre Raspberry Pi *via* SSH ou VNC, il suffit de le connecter au réseau et de l'alimenter ; vous n'aurez ainsi pas besoin de clavier, de souris ni d'écran, sauf pour le premier démarrage.

PREMIER DÉMARRAGE DU RASPBERRY PI

Ce chapitre vous permet de commencer à travailler rapidement avec le Raspberry Pi. La première chose à faire consiste à installer Linux sur le Raspberry Pi. Cela comprend les étapes suivantes :

- Télécharger et extraire l'installateur sur une carte SD formatée.
- Insérer la carte dans le Raspberry Pi et le connecter à un clavier, une souris et un écran.
- Allumer le Pi, choisir ce qu'il faut installer, et attendre.

Une fois que c'est fait, vous êtes prêt à démarrer le Pi avec un bureau Linux graphique.

Vous aurez besoin des éléments suivants :

- Raspberry Pi Model B
- Câble micro USB et chargeur USB (ou ordinateur)
- Carte SD de 4 Go
- Écran avec prise HDMI
- Câble HDMI
- Souris USB
- Clavier USB

Extraction de NOOBS*.zip

Téléchargez *_NOOBS_vX_Y_Z.zip_* (au moment de la rédaction, il s'agissait de la version _NOOBS_v1_3_7.zip, mais le nom du fichier sera peut-être différent quand vous lirez ce livre) à partir de http://raspberrypi.org/downloads.

> Vous trouverez aussi tous les liens importants mentionnés dans cet ouvrage sur http://bolbook.com, ainsi que des copies miroir de certains fichiers.

Insérez la carte SD dans votre ordinateur. La plupart des cartes SD sont formatées en FAT32 en usine et il suffit d'extraire l'archive NOOBS sur la carte SD, à moins que vous n'utilisiez une carte SD que vous avez vous-même formatée.

Après avoir décompressé le fichier, assurez-vous que le fichier *_bootcode.bin_* est dans le répertoire racine de la carte SD.

> Si vous avez besoin de formater la carte SD, utilisez l'outil de formatage de la SD Card Association qui est disponible à https://www.sdcard.org/downloads/formatter_4/.

Dans les versions modernes de Linux, Windows et Mac, il suffit de faire un double-clic sur l'archive NOOBS ou bien un clic droit pour la décompresser. Avec les versions anciennes de Windows, vous pouvez installer 7-Zip (http://www.7-zip.org) pour décompresser les fichiers ZIP.

Connexion des câbles

La connexion des câbles est facile car chaque câble a une prise bien spécifique. Branchez la souris et le clavier dans les ports USB du Raspberry Pi.

Si vous utilisez un écran HDMI, connectez un câble HDMI entre l'écran et le Raspberry Pi.

Si vous utilisez un moniteur NTSC ou PAL, connectez un câble vidéo composite à la prise jaune du Raspberry Pi pour le raccorder à l'écran.

Connectez ensuite le câble micro USB au Raspberry pour l'alimenter. Branchez ce câble sur le port USB d'un ordinateur ou sur un chargeur USB 5 volts qui fournit au moins 700 mA.

Démarrage et installation de Raspbian

Dès que vous alimentez le Raspberry Pi, il démarre. Il n'y a donc pas besoin d'interrupteur.

> Si rien n'apparaît à l'écran, il vous faudra peut-être sélectionner le bon mode d'affichage pour le Raspberry Pi.
>
> Le mode d'affichage par défaut est HDMI, mais si vous êtes connecté en HDMI et que rien n'apparaît, essayez d'appuyer sur la touche **2** du clavier connecté au Raspberry Pi pour sélectionner le mode HDMI Safe.
>
> Si vous êtes connecté *via* le connecteur jaune composite, appuyez sur la touche **3** pour un moniteur ou un téléviseur PAL et **4** pour un écran NTSC.

Vous êtes accueilli par un menu graphique des différents systèmes d'exploitation.

Sélectionnez « Raspbian [RECOMMENDED] » (Figure 1.2) et choisissez la langue et le type de clavier que vous utilisez.

Figure 1.2 Choix d'un système d'exploitation

Si vous connaissez n'importe quelle distribution Debian, Mint ou Ubuntu, vous serez en terrain connu ; si ce n'est pas le cas, continuez à lire et tout ira bien !

L'installation de Raspbian ne prend que quelques minutes (Figure 1.3). Quand le processus est terminé, un message vous invite à cliquer sur OK pour redémarrer.

Figure 1.3 Installation de Raspbian

L'utilitaire de configuration du Raspberry Pi s'ouvre. Utilisez les flèches et la touche Tabulation pour vous déplacer, appuyez sur la touche Entrée pour sélectionner une option (Figure 1.4).

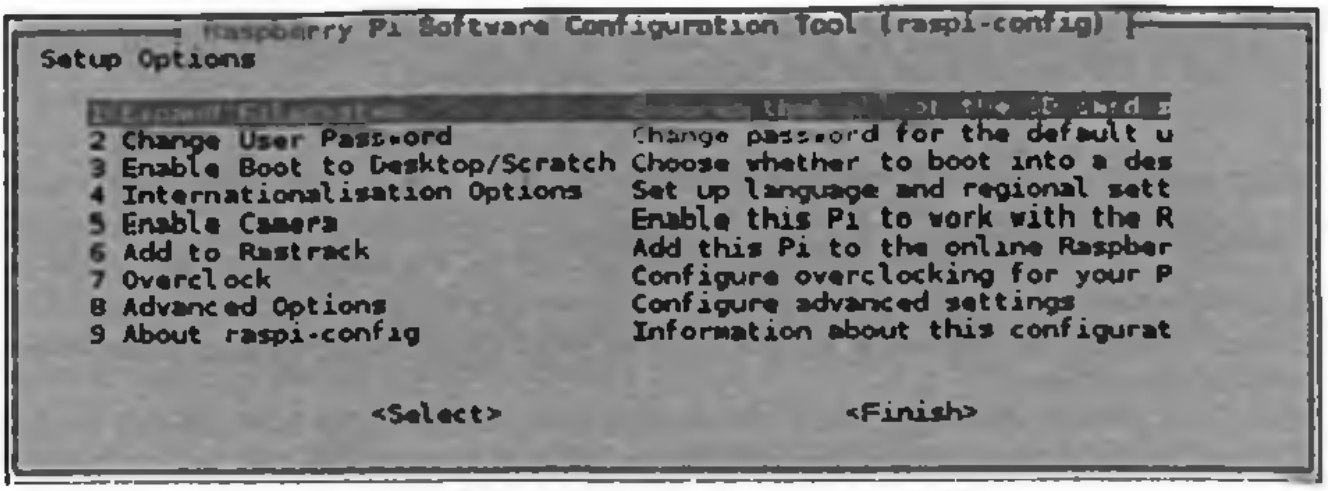


Figure 1.4 Utilitaire de configuration raspi-config

Il est souhaitable d'activer l'option **Enable Boot to Desktop/Scratch**. Quand la modification des paramètres est achevée, utilisez la touche Tabulation pour sélectionner **Finish** et redémarrer.

Une fois que le Raspberry Pi a redémarré, vous vous retrouvez dans un environnement graphique et une session sera ouverte automatiquement.

Si vous n'avez pas activé l'option **Boot to Desktop**, vous démarrerez toujours avec une interface en ligne de commande. Il faut alors vous connecter sous le nom « **raspberry** » avec le mot de passe « **pi** » (sauf si vous avez modifié le mot de passe). Après vous être identifié, saisissez `startx` pour démarrer X Window System, qui est l'interface graphique.

Bienvenue chez Linux !

Vous avez à présent installé Raspbian sur un Raspberry Pi (Figure 1.5).

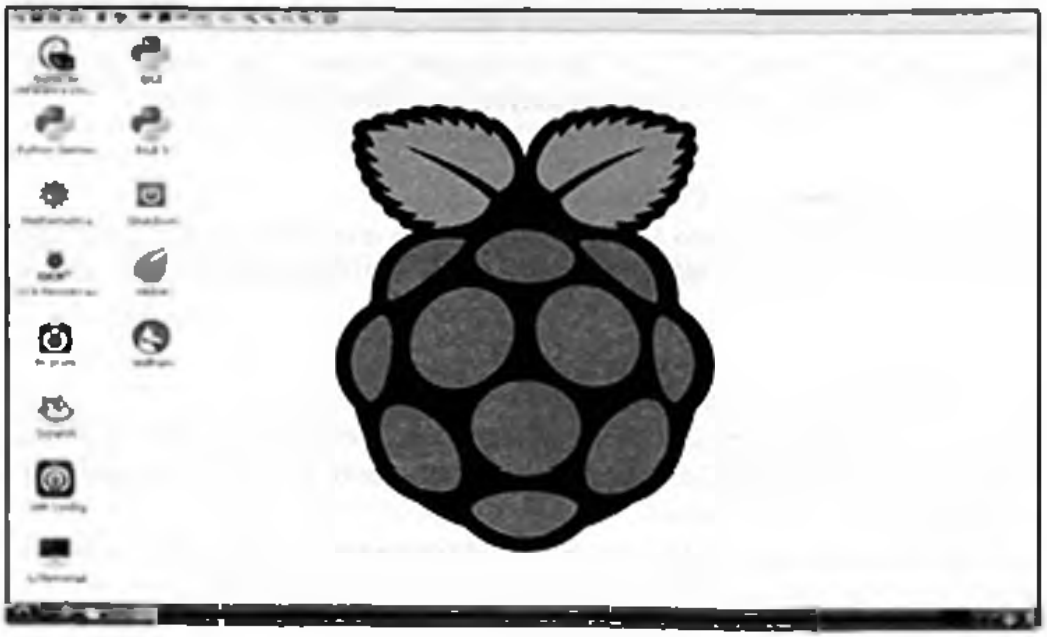


Figure 1.5 Bureau Linux

Pour éteindre votre Raspberry Pi, faites un double-clic sur l'icône Shutdown du bureau. Une fois le processus d'arrêt terminé, vous pouvez débrancher l'alimentation.

Dépannage de votre installation Raspberry Pi

Voici quelques solutions à quelques problèmes courants :

Votre carte est-elle formatée en FAT32 ?

Si vous n'arrivez pas à démarrer à partir de la carte SD, c'est peut-être parce qu'elle n'est pas correctement formatée.

Sous Linux, utilisez l'éditeur graphique intégré de partition (saisissez `sudo gparted` pour l'exécuter). Formatez la totalité de la carte en FAT. Vous pouvez exécuter un autre outil avec la commande `sudo palimpsest` (ou `sudo gnome-disks`) et, pour les utilisateurs avancés, `sudo parted` fournit un outil classique de gestion des partitions en ligne de commande.

Sous Windows et Mac, utilisez l'outil de la SD Association (https://www.sdcard.org/downloads/formatter_4/). Sous Windows, choisissez l'option « format size adjustment » de l'utilitaire et sous Mac, l'option « Overwrite Format ».

La LED rouge d'alimentation (PWR) ne s'allume pas.

La LED d'alimentation brille faiblement ou clignote ? Elle s'allume puis elle s'éteint ? Le Raspberry Pi n'a pas suffisamment de courant. Connectez un chargeur USB qui fournit 5 volts et au moins 1 ampère. Si vous alimentez le Pi *via* le port USB d'un ordinateur portable, connectez-le à la prise USB d'un ordinateur fixe ou utilisez un chargeur puissant de téléphone mobile ou de tablette.

L'écran est noir, mais la LED rouge est allumée.

Il est possible que le Raspberry Pi ne puisse pas lire l'amorce de la carte SD. Éteignez-le, enlevez la carte SD et réinsérez-la franchement. Vérifiez que le premier fichier utilisé dans la séquence de démarrage, *bootcode.bin*, est dans le répertoire racine de la carte SD.

Si le problème persiste, formatez la carte SD et décompressez à nouveau l'archive NOOBS sur la carte SD. Si cela ne règle pas le problème, essayez une carte SD différente.

Il y a quatre carrés de couleur à l'écran.

L'amorce a été lue à partir de la carte SD, mais le système d'exploitation *kernel.img* n'a pas réussi à démarrer. Formatez la carte SD et décompressez à nouveau l'archive NOOBS ou essayez une autre carte SD.

Le démarrage échoue et il y a des messages d'erreur.

Déconnectez tous les périphériques USB, comme le clavier, la souris et l'adaptateur WiFi (s'il est présent). Ne laissez que la carte SD, l'écran et l'alimentation. Enlevez et insérez la carte SD pour vous assurer que le contact est correct.

Si le problème persiste, reformatez la carte SD et décompressez à nouveau l'archive NOOBS.

Vous avez mis la pagaille dans le système d'exploitation.

Si les commandes normales ne fonctionnent plus et que votre écran est rempli de messages d'erreur ou que le Raspberry Pi cesse de fonctionner soudainement, ne vous inquiétez pas. Maintenez enfoncée au démarrage la touche Majuscule et choisissez l'option de réinstallation de Raspbian. Cela est

assez rapide et facile, mais toutes les données de la carte SD sont effacées. Si le problème persiste, reformatez la carte SD et décompressez à nouveau l'archive **NOOBS**, puis recommencez le processus d'installation.

Vous n'avez pas d'Internet.

Si vous avez connecté un câble Ethernet avant de démarrer, cela doit fonctionner parfaitement sur un réseau classique. Vérifiez que la LED LNK est allumée sur le Raspberry Pi.

Si ce n'est pas le cas, cela signifie que le Raspberry Pi pense qu'une extrémité du câble Ethernet n'est pas connectée. En général, vous devez voir 100 (ce qui indique une connexion de 100 Mbit/s), et la LED FDX (*full duplex*) doit être allumée.

Si LNK est allumée et que vous avez encore des problèmes, vous pouvez essayer les commandes suivantes : `ifconfig` (affiche la configuration de l'adaptateur réseau), `route -n` (affiche la table de routage de la connexion réseau), `cat /etc/resolv.conf` (affiche le serveur de noms utilisé), et `ping -c 1 dunod.com` (indique si vous pouvez accéder au Web).

Si vous rencontrez des messages d'erreur qui ne sont pas listés ici, essayez de saisir entre guillemets le message d'erreur dans un moteur de recherche et assurez-vous d'avoir bien saisi le même message d'erreur. Si le message apparaît lors du démarrage, prenez une photo avec votre APN ou votre téléphone mobile et saisissez-le dans un moteur de recherche.

INTRODUCTION À LINUX

Raspberry Pi est Linux, ce qui signifie qu'il est bâti sur Linux. Le nom « Linux » est employé pour décrire le noyau du système d'exploitation ainsi que le système d'exploitation lui-même.

Le système d'exploitation Linux est composé du noyau et de milliers d'utilitaires et d'applications émanant de différentes sources.

Le Raspberry Pi n'est pas une station de travail, mais s'apparente plus en termes de puissance à une tablette d'entrée de gamme ou à un téléphone mobile.

Dans ces conditions, même si vous utilisez le bureau graphique, ne comptez pas abandonner tout de suite votre ordinateur de bureau ou votre portable. Une puissance minimale combinée à une mémoire réduite empêche les applications comme LibreOffice et Mozilla Firefox d'être utilisables.

Une interface en ligne de commande omniprésente

Êtes-vous prêt à ressentir la puissance de l'invite `$` ?

La ligne de commande a résisté à l'épreuve du temps et c'est peut-être quelque chose que vous apprendrez à vos petits-enfants. Les commandes que vous allez utiliser tout le temps, comme `pwd`, `ls`, ou `cat`, existaient bien avant l'invention de Linux (Linux a été créé par un étudiant finlandais à Helsinki, à cinq kilomètres du jardin botanique où j'écris ces lignes). Les utilisateurs chevronnés de Mac OS X et de Windows utilisent aussi la ligne de commande quand ils ont besoin de faire quelque chose impossible à réaliser avec une souris.

La plupart des commandes que vous allez utiliser avec un Raspberry Pi sont identiques à celles que l'on emploie sur un Mac ou un ordinateur sous Linux, et elles sont similaires aux outils en ligne de commande sous Windows.

Comme vous le savez sans doute, la plupart des serveurs tournent sous Linux (Google, Facebook, Amazon et la majorité des superordinateurs). Comme les serveurs web n'ont pas d'interface

utilisateur graphique, les programmeurs et les administrateurs système doivent apprendre à utiliser l'interface en ligne de commande.

OMNIPRÉSENCE DE L'INTERFACE EN LIGNE DE COMMANDE

Si votre smartphone tourne sous Android, il s'exécute en fait sous Linux, et vous pouvez exécuter certaines commandes sur votre téléphone. Android prend en charge un sous-ensemble limité de commandes, même sans *jailbreaking* (vous aurez alors besoin d'une application d'environnement de shell comme **ZShaolin** disponible à http://www.dyno.org/software/zshaolin/).

Comme les ordinateurs Apple, l'iPhone et l'iPad sont basés sur un cousin lointain de Linux, BSD, et vous pouvez accéder à la ligne de commande sur un iPhone jailbreaké.

L'interface en ligne de commande est facile à automatiser. Toute commande saisie dans l'invite (que l'on appelle aussi le **shell**), peut être transformée en programme. Il suffit de placer chaque commande dans un fichier texte, une commande par ligne (on vous montre comment faire dans la prochaine section).

Vous pouvez ensuite exécuter le programme avec `dash nom_du_fichier` (sur Raspberry Pi, le shell s'appelle dash). Quand vous vous rendrez compte que vous n'arrêtez pas de ressaisir la même dizaine de commandes, il sera temps de les transformer en script.

Même si les scripts de commandes sont si faciles à écrire qu'on les emploie pour des scripts à usage unique, ils conviennent aussi parfaitement pour des applications critiques. Par exemple, Linux utilise des scripts pour démarrer et contrôler les démons (applications serveur qui s'exécutent en arrière-plan).

Première visite

Après avoir démarré votre Raspberry Pi en mode graphique, vous pouvez démarrer l'interface en ligne de commande en faisant un double-clic sur l'icône LXTerminal du bureau.

L'invite `$` signifie que Linux attend votre commande. Saisissez `pwd` pour afficher le répertoire actuel de travail (l'emplacement où vous vous trouvez dans le système de fichiers). Linux répond en indiquant un chemin, comme */home/pi/*.

Pour lister les fichiers du répertoire de travail, saisissez `ls` et appuyez sur Entrée. Voilà des fichiers que vous pouvez facilement manipuler. Lorsque vous obtenez un message d'erreur comme « No such file or directory », utilisez simplement `pwd` pour voir où vous êtes puis `ls` pour lister quels fichiers se trouvent dans le répertoire de travail.

> Vous devez toujours appuyer sur la touche Entrée après avoir saisi une commande shell comme `ls`. Avant d'avoir validé par Entrée, vous pouvez utiliser les flèches de curseur pour modifier la commande.

Pour modifier (ou créer) un fichier appelé *test.txt*, saisissez `nano test.txt`. Entrez du texte puis appuyez sur Ctrl+X pour l'enregistrer (confirmez en saisissant y et appuyez sur Entrée quand on vous demande le nom du fichier). Pour modifier le fichier, saisissez à nouveau la commande `nano test.txt`.

Fichiers texte de configuration

Sous Linux, la plupart des choses sont représentées par des fichiers texte. Avec seulement les commandes listées dans la section précédente et la commande sudo, un hacker compétent peut modifier de nombreux paramètres système.

Tous les paramètres de configuration Linux sont stockés dans des fichiers texte. Les fichiers de configuration système se trouvent dans le répertoire */etc/* et la configuration de chaque utilisateur se trouve dans le répertoire */home/pi/*.

Même les broches d'entrée et de sortie du Pi peuvent être manipulées en modifiant des fichiers texte du répertoire */sys/*. Vous apprendrez plus loin dans ce livre à connecter des capteurs et des LED au Raspberry grâce aux broches GPIO.

Si vous avez connecté le Pi à un réseau avec un câble Ethernet avant de le démarrer, votre Raspberry Pi doit déjà être connecté à Internet.

Testez cela en saisissant ping www.dunod.com. Vous verrez un résultat toutes les secondes.

Interrompez la commande au bout d'un certain temps en saisissant Ctrl+C. Si le réseau fonctionne bien, il doit indiquer une perte de paquets égale à 0 %.

Vous pouvez aussi télécharger des pages web complètes grâce à des commandes comme curl botbook.com et wget botbook.com.

sudo

Linux est réputé pour son modèle de sécurité robuste, la « *séparation des permissions des utilisateurs* » étant l'une de ses fonctionnalités caractéristiques.

Les utilisateurs normaux ne peuvent modifier que les fichiers qui affectent leur environnement de travail. Cela signifie qu'ils ne peuvent modifier que les fichiers de leur répertoire de travail (*/home/pi/*) et les répertoires de travail temporaires comme */tmp/*.

Le superutilisateur, que l'on appelle root, a tous les pouvoirs et peut modifier n'importe quel fichier du système. Pour utiliser les privilèges de root, il suffit de placer sudo devant la commande que l'on souhaite exécuter.

Par exemple, le gestionnaire de paquets de Raspbian facilite l'installation de programmes supplémentaires, mais vous avez besoin d'utiliser les privilèges de root pour installer quoi que ce soit. Avant d'installer un nouveau logiciel, vous devez mettre à jour la liste des programmes disponibles avec la commande sudo apt-get update. Cela nécessite une connexion réseau car tous les paquets logiciels se trouvent sur un serveur de fichiers.

> De nombreux systèmes Linux et Unix (comme Mac OS X) sont configurés de telle sorte qu'il faut saisir le mot de passe de l'utilisateur quand on utilise sudo, ce qui constitue une sécurité supplémentaire. Par défaut, le système d'exploitation Raspbian ne vous demandera pas cela.
>
> Faites attention dans l'utilisation de sudo car vous pouvez facilement commettre des erreurs qui empêcheront le système de démarrer.

Vous pouvez installer n'importe quel programme en spécifiant le nom de son paquet. Pour installer ipython (un outil interactif qui permet de tester le langage Python et la visualisation des données), saisissez sudo apt-get -y install ipython. Le paramètre `-y` indique au gestionnaire de paquets (apt) de présupposer une réponse positive à toutes les questions qu'il pose.

Après son installation, vous pouvez exécuter le nouveau paquet en saisissant ipython. Toute commande python fonctionnera, mais exit() vous renverra à l'invite de commande ($) où vous pouvez saisir des commandes du shell.

> Des invites différentes sont un moyen de vous indiquer à quel programme vous parlez :
> – Quand vous voyez l'invite du shell (`$`), vous pouvez saisir le nom d'une des commandes intégrées du shell ou d'un programme installé sur votre Raspberry Pi.
> – Quand vous voyez une autre invite, comme celle d'ipython (`In [1]:`), vous pouvez saisir des commandes Python.

L'installation de démons (que l'on appelle aussi serveurs) est également facile.

Testez l'installation du serveur web le plus populaire de la planète, Apache (http://www.apache.org), avec cette commande : `sudo apt-get -y install apache2`.

Quand l'installation est terminée, vous pouvez aller sur la page d'accueil de votre serveur web avec `curl localhost`.

Pour accéder à votre serveur web Apache à partir d'un autre ordinateur, déterminez votre adresse IP à l'aide de la sortie de la commande `ifconfig` et saisissez cette adresse dans un navigateur web sur un autre ordinateur connecté au réseau.

> Vous verrez au moins deux adaptateurs listés dans la sortie de la commande `ifconfig`.
>
> Utilisez l'adresse de l'adaptateur Ethernet (`eth0`) ou bien celle de l'adaptateur WiFi en fonction de la connexion réseau que vous employez.

Cela fait beaucoup de lignes de commandes si vous débutez avec Linux ! Vous avez bien mérité une pause et vous allez éteindre le Raspberry Pi en saisissant la commande `sudo shutdown -P now`.

Tout comme vous le faites avec votre ordinateur de bureau, vous devez arrêter proprement votre machine de telle sorte que les données soient écrites sur le disque avant de l'éteindre. Une fois que le Pi est arrêté, vous pouvez débrancher le cordon d'alimentation USB. Quand vous voudrez le réutiliser, il suffira de le rebrancher.

Reportez-vous à l'*Annexe A* pour une liste des commandes Linux.

UN MANCHOT PRODIGE

Je demande parfois à mes étudiants de deviner combien de temps il me faut pour saisir « **supercalifragilisticexpialidocious.foo.bar.txt.botbook.com.pdf.cc** » en me servant uniquement de ma main gauche. Quand ils ont donné quelques réponses, je fais le test.

Au préalable, j'ai créé un fichier grâce à cette commande :

```
$ nano
supercalifragilisticexpialidocious.foo.bar.txt.botbook.com.pdf.cc
```

Je compte jusqu'à trois et je saisis le tout en moins de cinq secondes :

```
$ ls
supercalifragilisticexpialidocious.foo.bar.txt.botbook.com.pdf.cc
```

En réalité, j'ai simplement saisi `ls s` puis j'ai appuyé sur la touche Tabulation.

Le shell complète le mot que vous saisissez après avoir pressé la touche Tab et vous devez tout le temps utiliser cette fonctionnalité.

C'est surprenant la manière dont cela fonctionne : non seulement les répertoires et les fichiers sont devinés, mais aussi les serveurs réseau après des commandes comme `ssh` et `ping`. Comme la touche Tab ne complète que les noms de fichiers corrects, vous ne ferez plus d'erreur de saisie.

CONNEXIONS DES BROCHES DU RASPBERRY PI

Les broches du port GPIO (*General-Purpose Input and Output*, c'est-à-dire entrées et sorties à usage général) permettent de connecter des composants électroniques directement au Raspberry Pi.

On parle de broches à usage général car on peut décider de leur fonctionnalité et il est en outre possible de configurer la même broche pour qu'elle soit une entrée et une sortie à des moments différents.

Tout au long de cet ouvrage, vous apprendrez à maîtriser :

- La sortie numérique (allumer et éteindre une LED).
- La résistance numérique (détecter si on a appuyé sur un bouton ou si un capteur est actif).
- L'entrée numérique pour de très courtes impulsions (utilisées par des capteurs comme un capteur de distance).
- La résistance analogique (utilisée par des capteurs de pression, de lumière, de température).
- Les protocoles standard industriels comme I2C et SPI (utilisés par le Nunchuk de la Wii et les convertisseurs analogique-numérique).

À la différence de la plupart des autres tutoriels, nous vous apprendrons à utiliser l'entrée et la sortie numérique sans avoir à invoquer les privilèges de root tout le temps, ce qui améliore la sécurité et la stabilité.

Pour l'entrée numérique, vous apprendrez à utiliser la résistance pull-up interne du Pi, de telle sorte que votre circuit utilise un nombre minimal de composants.

Pour mesurer la résistance analogique, on utilisera une puce de conversion analogique-numérique.

De nombreux composants que l'on trouve dans des objets courants communiquent à l'aide des protocoles I2C et SPI dont vous trouverez des exemples dans ce livre.

Nous allons pour l'instant commencer par utiliser la forme la plus simple de sortie numérique.

Hello GPIO, clignotement d'une LED

Dans cet exemple de programme « Hello GPIO World », vous allez connecter une LED au Raspberry Pi et la faire clignoter.

Nous commençons tous nos projets par un programme « Hello World » quelle que soit la plateforme et quel que soit le langage. Alors quand on s'apprête à créer quelque chose de plus compliqué, c'est une bonne idée de commencer par un « Hello GPIO World » car cela confirmera que le matériel et le logiciel fonctionnent à leur niveau le plus élémentaire.

Si votre programme « Hello World » ne fonctionne pas, vous devez alors résoudre les problèmes avant d'entreprendre quelque chose de plus complexe.

Voici la liste des éléments nécessaires :

- Raspberry Pi.
- Câbles de liaison femelle-mâle, noir et vert ou bien jaune.
- Platine de test sans soudure (appelée également plaque d'essai ; en anglais *breadboard*).
- Résistance de 470 ohms (bandes jaune-violet-marron).
- Une LED.

Si vous n'avez pas tous ces éléments sous la main, consultez la section *Dépannage* de ce chapitre pour obtenir des suggestions.

Pour les câbles, on abrège souvent femelle par F et mâle par M quand on indique leur longueur. Le type de câble femelle-mâle dont vous avez besoin est en général vendu sous la dénomination de câbles mâle vers femelle pour platine d'expérimentation.

Les broches du port GPIO ne sont pas protégées contre les surtensions (Figure 1.6). À la différence de l'Arduino, le Raspberry Pi ne pardonne pas les erreurs. Les broches de données n'acceptent qu'une tension de 3,3 V.

La connexion d'une alimentation de +5 V sur une broche de données peut facilement endommager votre Raspberry Pi, ou au minimum, rendre cette broche inutilisable.

Vérifiez bien deux fois tout montage sur votre platine de test et faites bien attention à l'endroit où vous placez votre sonde si vous utilisez un multimètre avec les broches GPIO.

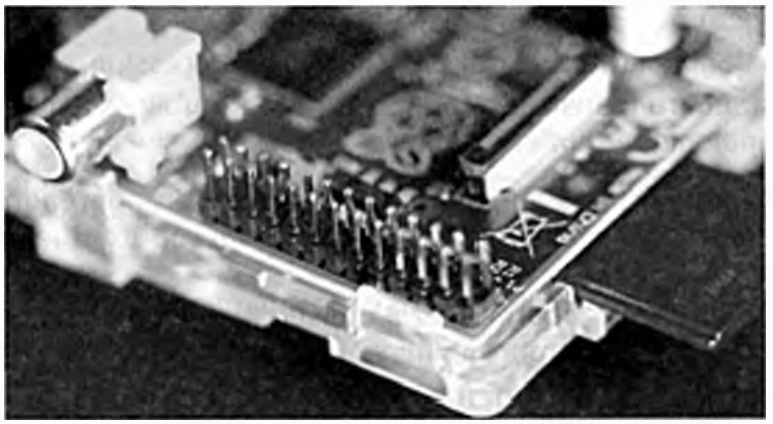


Figure 1.6 Broches GPIO

Montage du circuit

Le circuit consiste simplement en une LED avec une résistance limitant le courant, connectées en série entre la broche GPIO 27 et la masse (Figure 1.7).

Connectez le fil court (négatif) de la LED au câble noir, et le fil long (positif) à la résistance. Vous devez faire ces connexions quand le Raspberry Pi est éteint et débranché.

Montez le circuit sur la platine de test (Figure 1.8).

Vérifiez deux fois toutes les connexions pour éviter d'endommager votre Raspberry Pi.

Quand vous êtes certain d'avoir connecté les câbles correctement, vous pouvez allumer votre Pi.

Pour tout savoir sur la numérotation des broches du Raspberry Pi, poursuivez votre lecture.

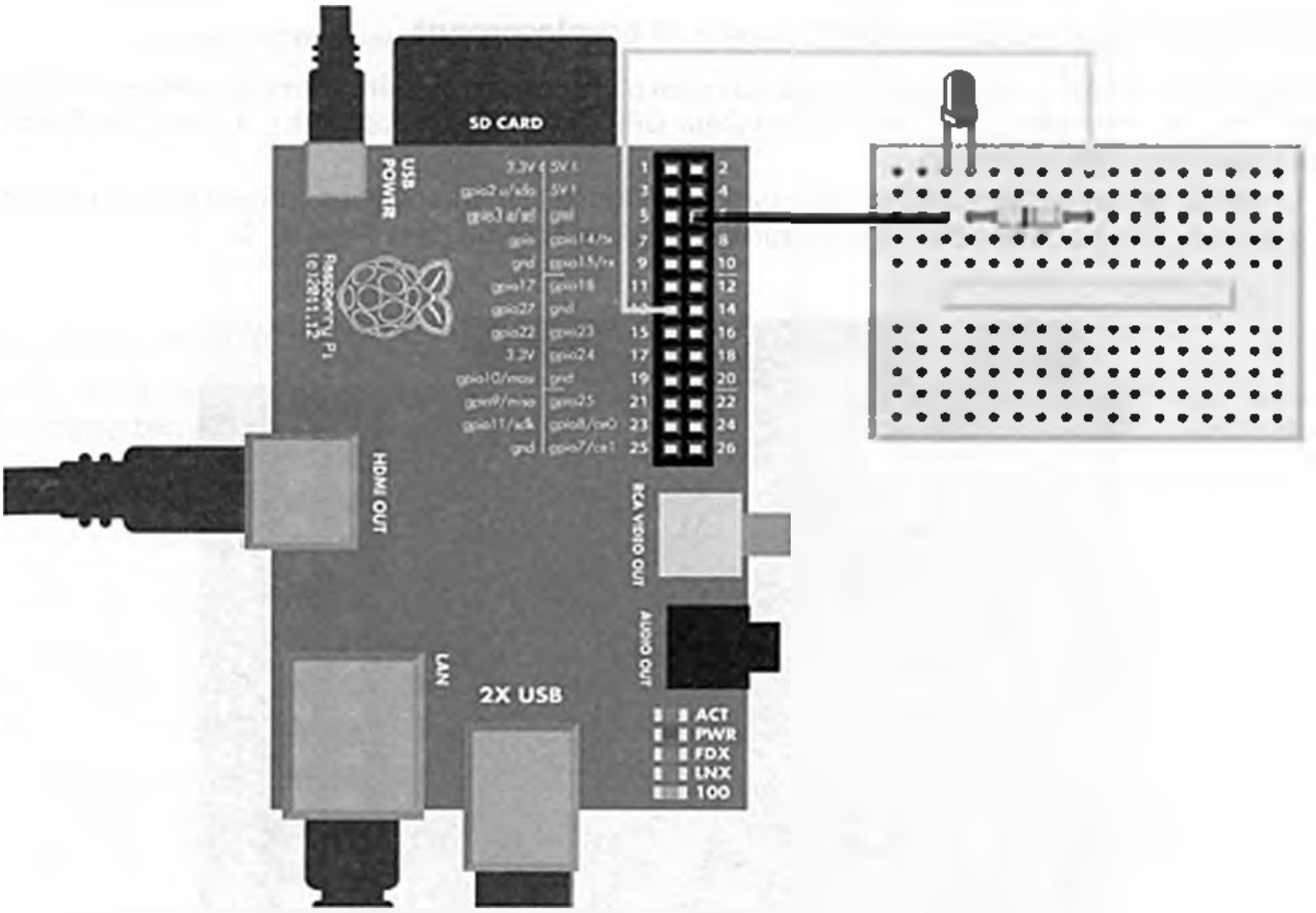


Figure 1.7 Montage de la LED sur la platine de test

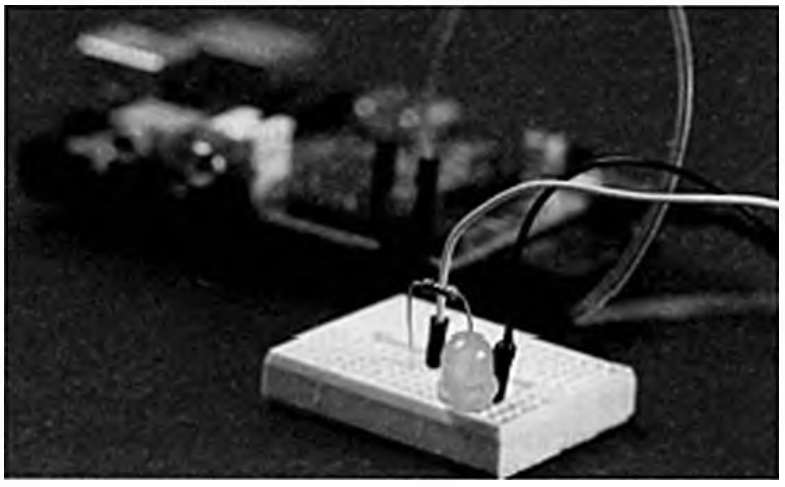


Figure 1.8 Hello GPIO !

Deux systèmes de désignation : usage et emplacement

Chaque broche GPIO a deux désignations : un nom d'usage et un numéro d'emplacement physique. Pour trouver la broche correcte du connecteur GPIO, vous devez apprendre à jongler entre ces deux systèmes de désignation.

Localisez le connecteur GPIO du Raspberry Pi (Figure 1.6). Quand vous convertissez une dénomination de broche d'un système à un autre, utilisez le schéma de la Figure 1.9.

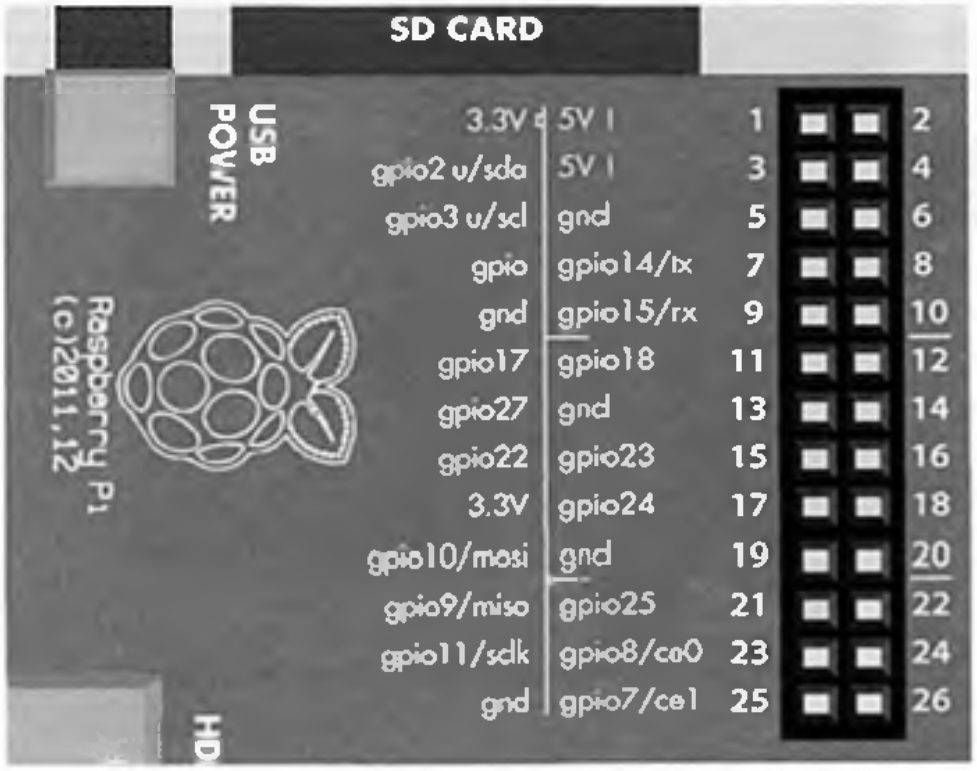


Figure 1.9 Numérotation des broches GPIO

Le côté gauche du diagramme de numérotation indique l'usage des broches (GND, GPIO 27) alors que le côté droit indique l'emplacement physique (de 1 à 26).

Il y a un petit carré blanc à côté de la première broche qui comporte aussi la mention **P1**. Ce numéro de broche physique indique où insérer le câble de liaison. Ces numéros physiques sont également appelés **numéros de la carte**.

Dans l'autre système de désignation basé sur l'usage, le code que l'on écrit utilise les numéros GPIO, comme GPIO 27, qui correspond à la broche physique 13 (qui est la broche que l'on connecte à la résistance à l'aide d'un câble).

En général, les schémas des circuits font référence aux désignations d'usage comme la masse (GND), +5 V et +3.3 V, et la Figure 1.9 vous aidera à faire la correspondance avec l'emplacement physique. Certaines broches peuvent avoir plusieurs usages. Par exemple, la broche GPIO 10 peut être utilisée pour le bus SPI. Cette désignation par l'usage est également appelée **codage BCM**.

Pour vous aider à démarrer, les deux broches utilisées sont listées dans le Tableau 1.1. Pour vos futurs projets, apprenez à localiser les numéros physiques dans le schéma de la Figure 1.9.

Broche GPIO (Codage BCM, utilisé dans le code)	Emplacement physique (sur la carte)
GPIO 27	13
GND	6

Tableau 1.1 Broches utilisées dans le programme Hello GPIO

Contrôle des broches GPIO en ligne de commande

Nous allons voir comment contrôler en ligne de commande les broches GPIO que nous venons de connecter. Nous allons d'abord tenter l'opération en tant que root, puis sans avoir besoin d'invoquer les privilèges de sudo à chaque fois.

Comme nous l'avons déjà écrit, les fichiers texte contrôlent tout en Linux. Le pilote GPIO au niveau du noyau (un logiciel qui contrôle la manière dont Linux communique avec les broches GPIO) rend les broches GPIO disponibles via le système de fichiers virtuel */sys/*. Pour contrôler le GPIO, il suffit de modifier ces fichiers texte.

Nous n'avons pas besoin d'interface graphique pour le moment si bien qu'il est possible de tout faire en ligne de commande à partir de LXTerminal. Faites un double-clic sur son icône pour lancer cette application.

> Si vous avez configuré votre Raspberry Pi pour qu'il ne démarre pas avec une interface graphique ou si vous avez utilisé SSH pour vous connecter à distance à partir d'un autre ordinateur, vous n'avez pas besoin de démarrer le bureau graphique pour réaliser cette expérience.

Pour activer une broche, on commence par l'**exporter**, puis on la configure en mode « **out** » et on écrit le nombre « **1** ». Tout ceci est réalisé en modifiant des fichiers texte.

Écriture de fichiers sans éditeur

Précédemment, nous avons vu comment modifier le contenu d'un fichier avec l'éditeur de texte nano. Voici comment modifier des fichiers sans avoir besoin d'un éditeur.

Voyons tout d'abord comment afficher un texte. Vous pouvez afficher un texte sur le terminal avec la commande suivante (ne saisissez pas le caractère $ qui indique l'invite du shell) :

```
$ echo "Hello BotBook"
```

Tout texte affiché de cette manière peut aussi être redirigé pour écraser un fichier (faites attention avec cette technique et n'écrasez pas de fichier important) :

```
$ echo "Hello BotBook" > test.txt
```

Si le fichier *test.txt* n'existe pas, il sera créé. S'il existe, il sera écrasé sans avertissement. Vous pouvez utiliser l'opération > (redirection) pour envoyer la sortie de pratiquement n'importe quelle commande dans un fichier texte. Vous pouvez visualiser le contenu de ce fichier avec la commande cat :

```
$ cat test.txt
Hello BotBook
```

N'était-il pas possible d'utiliser simplement `nano test.txt` ? Bien entendu, cela est possible, mais nécessite plus de saisie et ce n'est pas aussi facilement automatisable.

Allumage de la LED

Nous allons utiliser `sudo` la première fois pour allumer une LED. Est-ce que c'est mal d'agir en tant que root pour des tâches qui ne relèvent pas de l'administration système ? En effet, si vous faites une erreur de saisie dans une commande `sudo`, vous pouvez rendre votre système d'exploitation inutilisable et vous devrez alors le réinstaller.

Ne vous inquiétez pas pour autant car nous allons utiliser `sudo` simplement à titre de test. Plus tard, nous réglerons le problème des permissions des fichiers Linux et nous modifierons les fichiers qui contrôlent le GPIO en tant qu'utilisateur normal.

Saisissez `sudo -i` pour obtenir un shell root. N'utilisez le shell root que pendant le temps nécessaire à cette tâche et saisissez `exit` quand vous avez terminé. Vous noterez que votre invite se transforme en un signe dièse, `#`. Faites bien attention à ce que vous saisissez en tant que root car les erreurs peuvent être lourdes de conséquences pour votre système d'exploitation.

Après avoir démarré le shell root, exportez la broche GPIO 27 afin de pouvoir la manipuler (n'oubliez pas qu'il faut se contenter de saisir le texte à droite de l'invite du shell `#`) :

```
# echo "27">/sys/class/gpio/export
```

Cela crée le nouveau fichier virtuel que vous allez utiliser pour faire clignoter la LED. Ensuite, on initialise la broche 27 en mode out, de telle sorte que l'on puisse l'activer et la désactiver.

```
# echo "out" > /sys/class/gpio/gpio27/direction
```

À présent, on active la broche :

```
# echo "1" > /sys/class/gpio/gpio27/value
```

Votre LED doit maintenant s'allumer. Pour l'éteindre, il suffit de la désactiver :

```
# echo "0" > /sys/class/gpio/gpio27/value
```

Est-ce que votre LED s'est allumée ? Hello, GPIO world !

Une fois que vous avez terminé d'agir en tant que root, n'oubliez pas de saisir `exit`. Votre invite reprend la forme du signe dollar, `$`, ce qui indique que vous utilisez le shell en tant qu'utilisateur normal.

Dépannage

Si vous avez des problèmes dans l'exécution de ce projet, vérifiez les choses suivantes :

Il vous manque une résistance de 470 ohms.

Utilisez n'importe quelle résistance de plusieurs centaines d'ohms pour ne prendre qu'un risque minime avec votre carte. La résistance est simplement utilisée pour limiter le courant qui passe à travers la LED afin d'éviter de griller la LED ou les broches du Raspberry Pi.

Si cela ne vous gêne pas que la LED soit trop brillante ou surchauffe légèrement, vous pouvez utiliser n'importe quelle résistance comprise entre 100 ohms et 1 kΩ (de telles résistances ont une troisième bande marron ; voir l'encadré *Règle de la troisième bande* de ce chapitre).

Je n'ai pas de câble de liaison femelle-mâle.

Vous pouvez utiliser une ancienne nappe de disque dur IDE de 40 brins (pas 80).

Placez le côté femelle du câble dans le connecteur GPIO. Coupez l'autre extrémité. Faites attention aux numéros du câble. Si vous n'avez pas besoin de tous les câbles, ne les dénudez pas. Si vous les dénudez tous, utilisez au moins un ruban adhésif pour protéger les extrémités du fil du +5 V.

Vous pouvez vous reporter à http://bit.ly/1f0GWfV pour un tutoriel sur l'utilisation d'une ancienne nappe IDE. Il existe aussi de nombreux connecteurs prêts à l'emploi comme celui qui est disponible chez Adafruit (http://bit.ly/POg1rj). Le Starter Kit pour Raspberry Pi de chez MAKE (http://bit.ly/1hILDew), qui comprend un Raspberry Pi, inclut le câble, une platine de test et une platine d'évaluation.

Ma LED ne s'allume pas.

L'avez-vous insérée correctement ? Les LED ont une polarité et si vous les insérez dans le mauvais sens, elles ne s'allument pas.

La longue patte positive de la LED va sur la broche GPIO 27 (*via* la résistance). Le côté négatif de la LED a un méplat et une patte moins longue que le fil positif. Le côté négatif de la LED va à la masse. La flèche de la Figure 1.10 pointe vers la cathode (côté négatif). Il faut regarder attentivement pour remarquer le méplat à la base de la LED.

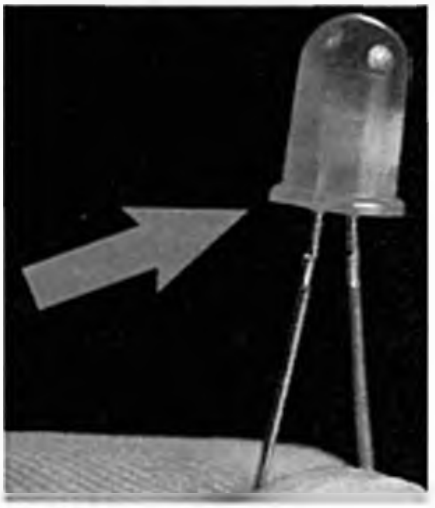


Figure 1.10 Polarité d'une LED

Rien ne se passe quand je saisis des commandes root.

Ne saisissez pas l'invite (#) ; elle est indiquée dans les exemples précédents pour vous montrer ce que vous verrez à l'écran de votre Raspberry Pi. Le shell indique cette invite quand il attend la saisie d'une de vos commandes quand vous êtes en shell root. Il ne vous viendrait pas non plus à l'idée de saisir l'invite de l'utilisateur normal « $ ».

Le symbole dièse a une signification spéciale pour le shell : il signifie que le reste de la ligne est un commentaire (une remarque intelligible par un être humain qu'on laisse dans un programme) qui est par conséquent ignoré par le shell.

Je n'arrive pas à accéder au shell.

Relisez les instructions de la section *Première visite* de ce chapitre.

J'obtiens un message d'erreur.

Copiez exactement le message d'erreur et collez-le dans un moteur de recherche comme Google. Essayez de mettre votre message d'erreur entre guillemets. Vous pouvez inclure les termes de recherche supplémentaires « Raspbian » ou « Raspberry Pi ». Quand vous avez résolu votre problème, c'est une bonne idée de le signaler sur un forum et d'inclure le message d'erreur exact dans votre message de manière à ce que d'autres personnes qui ont le même problème que vous puissent trouver la réponse.

RÈGLE DE LA TROISIÈME BANDE

La troisième bande d'une résistance permet d'identifier facilement sa valeur.

Pour cela, il faut regarder le multiplicateur qui est en général la troisième bande (sur une résistance à cinq bandes, le multiplicateur est la quatrième bande). Dans de nombreuses connexions, la valeur exacte de la résistance n'est pas importante tant qu'elle est suffisamment proche.

Par exemple, une résistance de 10 kΩ a un multiplicateur kilo (soit mille, 10^3), dont la valeur 3 est indiquée en orange. Une résistance de 10 MΩ à 50 MΩ a un multiplicateur méga (soit un million, 10^6), si bien que la troisième bande a une valeur de 6 qui est de couleur bleue.

Pour calculer toutes les bandes de couleur, recherchez sur le Web la chaîne « *calculateur code couleur résistance* ». Pour vérifier l'identification de votre résistance, mesurez sa résistance à l'aide d'un multimètre.

GPIO SANS ROOT

En évitant les privilèges de root, on rend le système plus sécurisé et plus stable. Si l'on prend l'exemple d'un programme qui publie des données de capteurs sur le Web, serait-il prudent d'exécuter un programme auquel des hackers pourraient se connecter en tant que root ?

Avant de commencer, assurez-vous que vous pouvez allumer et éteindre la LED (voir *Allumage de la LED*).

Dans les versions modernes de Linux, les périphériques reliés au système sont contrôlés par udev qui est un système basé sur des règles et pouvant exécuter des scripts quand les périphériques sont branchés. Si vous avez développé des applis Android sous Linux, vous avez sans doute créé une règle udev pour modifier les permissions quand votre téléphone mobile est branché à votre ordinateur. Si vous avez développé avec Arduino sous Linux, vous vous êtes probablement ajouté au groupe dialout pour avoir accès au port série *via* le port USB.

Normalement, les fichiers GPIO se trouvant dans */sys/class/gpio/* sont détenus par l'utilisateur root et le groupe root. Nous verrons comment écrire une règle udev pour changer le groupe en « dialout ».

Il suffit ensuite d'autoriser ce groupe à lire et à modifier les fichiers du répertoire */sys/class/gpio/*.

Pour finir, il faut changer le paramètre sticky du dossier de telle sorte que tous les nouveaux fichiers et dossiers créés soient détenus par le groupe *dialout*.

Tous les paramètres système se trouvant dans */etc/*, il n'est pas surprenant que les fichiers de configuration d'udev soient dans */etc/udev/*.

Commencez par ouvrir un éditeur avec sudoedit de manière à pouvoir créer un nouveau fichier de règle :

```
$ sudoedit /etc/udev/rules.d/88-gpio-without-root.rules
```

Ajoutez le texte de l'Exemple 1.1 au fichier. Assurez-vous de saisir chaque ligne correctement (ne saisissez pas les numéros en fin de ligne qui sont là pour expliquer le programme) car les règles udev sont très sensibles aux erreurs de saisie.

Exemple 1.1 88-gpio-without-root.rules

```
# /etc/udev/rules.d/88-gpio-without-root.rules - GPIO sans utilisateur root
pour Raspberry Pi # 1
# Copyright 2013 http://BotBook.com
# 2
SUBSYSTEM=="gpio", RUN+="/bin/chown -R root.dialout /sys/class/gpio/"
SUBSYSTEM=="gpio", RUN+="/bin/chown -R root.dialout
/sys/devices/virtual/gpio/"
# 3
SUBSYSTEM=="gpio", RUN+="/bin/chmod g+s /sys/class/gpio/"
SUBSYSTEM=="gpio", RUN+="/bin/chmod g+s /sys/devices/virtual/gpio/"
# 4
SUBSYSTEM=="gpio", RUN+="/bin/chmod -R ug+rw /sys/class/gpio/"
SUBSYSTEM=="gpio", RUN+="/bin/chmod -R ug+rw /sys/devices/virtual/gpio/"
```

1 Ce commentaire explique l'objet de ce fichier.
2 Définit le propriétaire des deux répertoires pour être root, et le groupe pour être *dialout*.
3 Initialise le drapeau sticky bit sur ces deux répertoires.
4 Configure les permissions sur les répertoires pour donner aux membres du groupe *dialout* des autorisations en lecture et en écriture.

Les règles sont traitées dans l'ordre chronologique, mais c'est probablement la seule règle affectant les répertoires GPIO, si bien que le numéro n'a pas d'importance. En code Morse, 88 est une abréviation pour les embrassades[1] et nous préférons cela au nombre 99 qui est souvent sélectionné et qui signifie que l'on est perdu.

Afin de minimiser les erreurs de saisie qui sont inévitables, vous pouvez télécharger une copie du fichier *88-gpio-without-root.rules* à partir de http://botbook.com.

Enregistrez le fichier (Ctrl+X, appuyez sur y, puis sur Entrée).

Pour utiliser vos nouvelles règles, redémarrez le démon udev et déclenchez la nouvelle règle avec ces commandes :

```
$ sudo service udev restart
$ sudo udevadm trigger -subsystem-match=gpio
```

Dans les versions modernes de Linux, tous les démons (« serveurs ») sont contrôlés par des scripts. Il est donc tout aussi facile de dire au serveur Web Apache de lire à nouveau son fichier de configuration avec sudo service apache2 reload. Vous pouvez aussi redémarrer le serveur SSH (on arrête et on démarre) avec sudo service ssh restart.

Vérifiez ensuite si l'appartenance est correcte :

1. « Love and kisses »

```
$ ls -lR /sys/class/gpio/
```

Le listing doit indiquer le groupe *dialout* plusieurs fois. Le paramètre -l permet d'afficher un listing long (avec le propriétaire, le groupe et les permissions), et -R signifie qu'on liste de manière récursive le contenu du répertoire.

Testons à présent le GPIO sans root. Vous noterez que le symbole d'invite est un dollar ($), ce qui indique que l'on est un utilisateur normal.

```
$ echo "27" > /sys/class/gpio/unexport
$ echo "27" > /sys/class/gpio/export
$ echo "out" > /sys/class/gpio/gpio27/direction
$ echo "1" > /sys/class/gpio/gpio27/value
```

La première commande unexport permet que la commande export ne génère pas d'erreur.

La LED doit s'allumer à présent. Vous pouvez l'éteindre avec :

```
$ echo "0" > /sys/class/gpio/gpio27/value
```

Nous avons réussi à contrôler la LED en tant qu'utilisateur normal ! Cela ouvre la voie à l'utilisation de GPIO à partir de presque n'importe quel langage de programmation.

Dépannage de GPIO

Permissions de fichiers non modifiées.

Si ls -lR /sys/class/gpio/ n'indique pas « dialout », la cause la plus probable est une erreur de saisie dans le fichier de règle */etc/udev/rules.d/88-gpio-without-root.rules*.

Téléchargez le fichier à partir de http://botbook.com et placez-le au bon endroit (par exemple, *sudo mv 88-gpio-without-root.rules /etc/udev/rules.d/*).

Pour que Linux utilise les nouvelles règles, exécutez sudo service udev restart ou bien redémarrez avec sudo shutdown -r now.

LED éteinte.

Essayez d'abord en tant que root (voir *Allumage de la LED* dans ce chapitre) pour vous assurer que cela fonctionne.

Vous avez un message d'erreur.

Copiez-le et faites une recherche sur le Web.

GPIO EN PYTHON

Vous pouvez utiliser le GPIO en Python simplement en écrivant et en lisant les fichiers se trouvant dans /sys/. Il s'agit de la même méthode que vous avez employée précédemment avec le shell.

Hello Python

Comme toujours, on commence par un programme « Hello World » pour tester l'environnement. Avec un éditeur de texte, créez le fichier :

```
$ nano hello.py
```

> Si vous ne pouvez pas enregistrer le fichier, c'est peut-être que vous avez utilisé la commande cd pour aller dans une partie du système de fichiers dans laquelle vous n'avez pas les permissions. Saisissez cd ~ pour retourner à votre répertoire d'accueil et essayez à nouveau d'enregistrer.

Le fichier ne contient qu'une seule ligne :

```
print "Hello world!"
```

Enregistrez le fichier (avec nano, on fait Ctrl+X, y puis Entrée). Exécutez maintenant le programme :

```
$ python hello.py
Hello world!
```

> L'exécution de la commande python sans paramètre démarre une console interactive (voir la section *Console Python* du chapitre 8).

GPIO avec Python

Faisons clignoter la LED connectée à la broche GPIO 27.

Connectez la LED si vous ne l'avez pas déjà fait dans l'expérience précédente (Figure 1.7). Enregistrez le code de l'Exemple 1.2 dans un fichier nommé led_hello.py sur votre Raspberry Pi et exécutez-le :

```
$ python led_hello.py

La LED du GPIO 27 a clignoté une fois...
```

Est-ce que votre LED s'est allumée pendant deux secondes ? Hello GPIO !

Pas de problèmes ? Lisez les explications du code et la section sur le dépannage. « Hello World » est le bon endroit pour résoudre ces problèmes. Quand vous aurez des problèmes avec du code plus compliqué, assurez-vous que vous êtes toujours en mesure d'exécuter ce « Hello World ».

Exemple 1.2 led_hello.py

```
# led_hello.py - allume une LED en utilisant le GPIO du Raspberry Pi
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time      # 1
import os
def writeFile(filename, contents):      # 2
        with open(filename, 'w') as f:  # 3
                f.write(contents)
# main
print "La LED du GPIO 27 a clignoté une fois..."          # 4
if not os.path.isfile("/sys/class/gpio/gpio27/direction"):      # 5
        writeFile("/sys/class/gpio/export", "27")       # 6
time.sleep(0.1)
writeFile("/sys/class/gpio/gpio27/direction", "out")    # 7
writeFile("/sys/class/gpio/gpio27/value", "1")  # 8
time.sleep(2)   # seconds           # 9
writeFile("/sys/class/gpio/gpio27/value", "0") # 10
```

1 Importe les bibliothèques dont vous avez besoin. Chaque bibliothèque a un **espace de noms** du même nom que la bibliothèque si bien que toutes les commandes que vous utilisez à partir de la bibliothèque **time** commencent par ce mot, comme time.sleep(2).

2 Définit une nouvelle fonction pour l'écriture des fichiers. La fonction ne s'exécute qu'ultérieurement lorsqu'elle est appelée.

3 En Python moderne, on accède aux fichiers avec la syntaxe with. Cela gère automatiquement la fermeture du fichier en cas d'exceptions. Dans la fonction open(), le nom de fichier peut être */sys/class/gpio/gpio27/direction* et w signifie qu'on ouvre le fichier en écriture. Cela crée un nouvel *handle* de fichier f que l'on utilise pour les opérations de fichiers.

4 Même si voulez faire clignoter une LED, c'est une bonne idée d'afficher quelque chose à l'écran afin de confirmer que le programme fonctionne. En Python, cela n'est pas critique car Python peut afficher des messages plutôt pertinents sur l'origine de l'erreur.

5 Vérifie que la broche n'est pas déjà exportée. Sinon la seconde exécution du code provoquera une erreur « IOError: [Errno 16] Device or resource busy ».
Ce style de contrôle de la condition au préalable est parfois appelé « demande de permission ». Il existe un autre style qui n'est pas employé ici : cela consiste à « demander pardon » et à utiliser try...except. On a choisi ici de « demander la permission » de manière à pouvoir utiliser une clause if et à garder le programme simple.

6 Exporte la broche. Cela crée tous les fichiers de contrôle de la broche, comme direction et value.

7 Comme on veut écrire sur la broche pour l'activer ou la désactiver, on définit la direction à « out ». Si l'on veut lire la valeur, il faut définir la direction à « in ». Si vous êtes familier avec Arduino, cela devrait vous rappeler la commande pinMode().

8 Définit la valeur à « 1 » pour allumer la LED. Cela signifie que la broche est initialisée à 3,3 V, le niveau HIGH d'une broche GPIO du Raspberry Pi.

9 Attend deux secondes. C'est une bonne idée de placer les unités dans les commentaires au moins une fois pour chaque fonction ou variable de telle sorte qu'un autre programmeur lisant votre code sache où vous voulez en venir. Pendant cette période, les broches sont laissées dans leur état en cours. Dans notre exemple, la broche GPIO 27 est à « 1 » et allume la LED.

10 Définit la valeur à « 0 » pour éteindre la LED.

Dépannage

Vous recevez un message d'erreur indiquant qu'une permission a été refusée.

Si vous obtenez une erreur du genre « IOError: [Errno 13] Permission denied ou IOError: [Errno 2] No such file », vous pouvez vivre dangereusement et tester le programme en tant que root :

```
$ sudo python led_hello.py # uniquement à des fins de test
```

Si cela fonctionne correctement en tant que root, c'est parfait ! Vous pouvez à présent régler les permissions des fichiers virtuels GPIO (voir *GPIO sans root* dans ce chapitre).

Si le problème persiste, redémarrez le Raspberry Pi en l'arrêtant, en débranchant le cordon d'alimentation et en le rebranchant.

Si tout fonctionne, vous pouvez exécuter le programme en tant qu'utilisateur normal :

```
$ python led_hello.py
```

La LED ne s'allume pas, mais le programme ne génère pas d'erreur.

Vérifiez la polarité de la LED et les connexions. Vérifiez que vous avez connecté les câbles aux bonnes broches du connecteur GPIO (Figure 1.9). Si cela ne résout pas le problème, vous pouvez utiliser un multimètre pour vérifier que vous avez utilisé la bonne résistance (par exemple, 470 ohms). Vous pouvez tester la LED avec un circuit qui n'a qu'une pile, la résistance et la LED en série. Cela vous dira si la LED fonctionne.

POUR ALLER PLUS LOIN

Vous avez à présent installé votre propre environnement Linux à 35 euros et vous pouvez même le câbler directement à des composants. Cela signifie que vous avez combiné la puissance de Linux et de l'électronique.

Les techniques d'administration système que vous avez testées vont dans la bonne direction car il est toujours préférable de ne pas abuser des privilèges de root.

Pour continuer à jouer avec votre Raspberry Pi, vous pouvez maintenant aborder les projets des capteurs de ce livre. Les exemples avec des capteurs individuels vous apprennent à utiliser des entrées numériques, des entrées analogiques, des protocoles standard industriels et à mesurer la durée d'une impulsion avec une interruption.

Vous pouvez appliquer vos nouvelles compétences en matière d'administration système et de GPIO avec tous les capteurs de cet ouvrage.

Le Raspberry Pi brille habituellement avec les capteurs qui utilisent des protocoles plus avancés. Cet avantage se manifeste notamment avec les capteurs I2C comme le Nunchuk de la Wii, le gyro-accéléromètre MPU 6050 et le capteur de pression atmosphérique GY65.

Bienvenue dans Linux embarqué !

2 Arduino

Arduino est une carte de développement simple et robuste (Figure 2.1). C'est le produit idéal pour apprendre l'électronique programmable, et il est de plus extrêmement fiable.

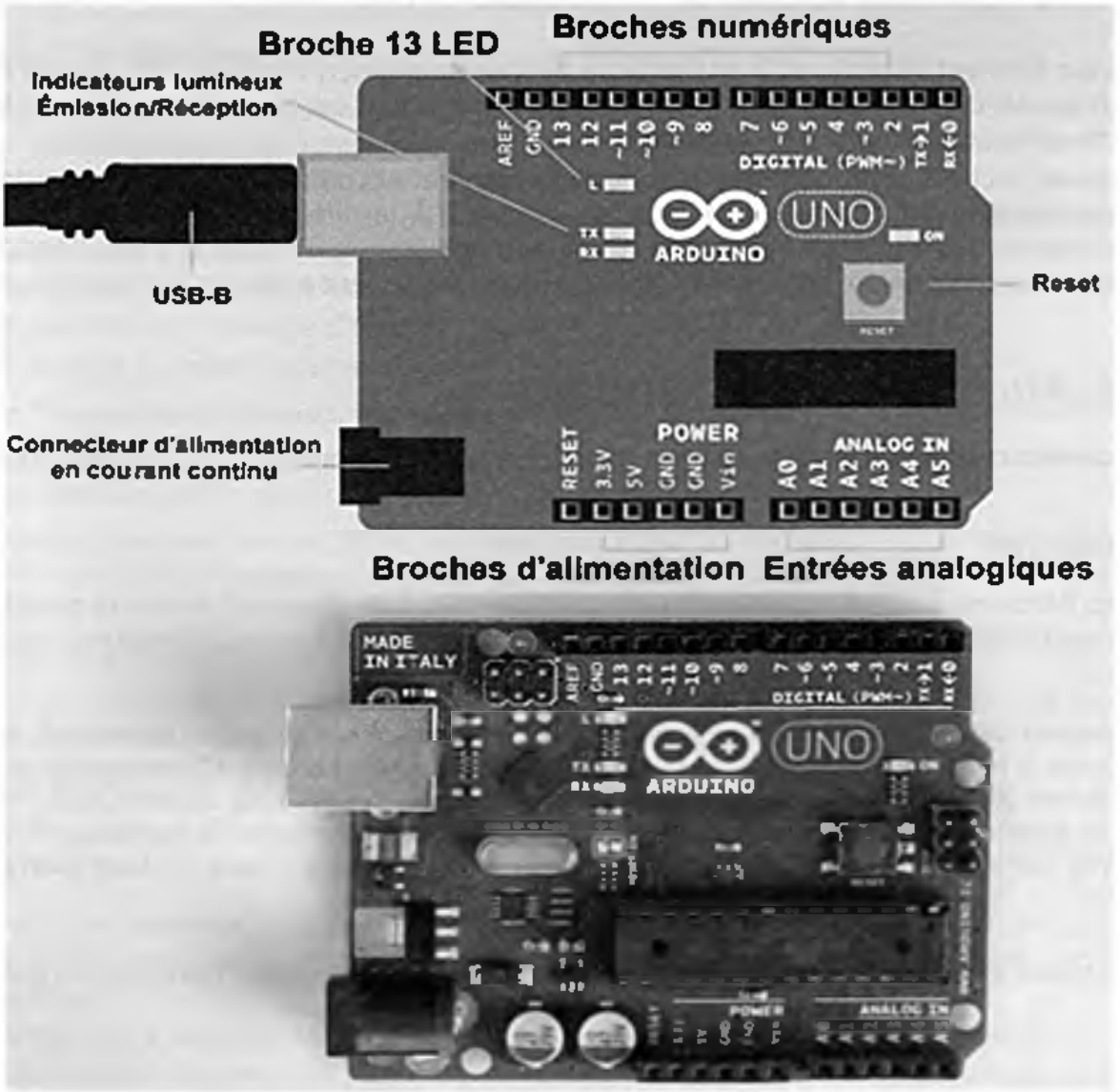


Figure 2.1 Connexions de l'Arduino

Il ne faut pas un gros investissement pour se mettre à Arduino : il suffit d'une carte Arduino Uno et d'un câble USB qui ne devraient pas vous coûter plus d'une trentaine d'euros. Le logiciel est libre (le code source est disponible pour que l'on puisse l'utiliser, l'étudier, le modifier et le partager).

Nous allons d'abord vous montrer comment installer l'environnement de développement Arduino (souvent appelé EDI pour environnement de développement intégré) sur votre ordinateur. Après cela, nous brancherons le câble USB et chargerons notre premier programme (appelé sketch dans la terminologie Arduino). On ne peut installer qu'un seul programme sur un Arduino, le sketch que l'on exécute. Il n'y a rien d'autre à faire car à la différence de Raspberry Pi, Arduino n'a pas de système d'exploitation. Il n'y a que vous, votre programme et la quincaillerie.

> En fait, il y a quand même quelque chose en plus. Arduino contient un programme d'amorçage qui occupe une petite portion de l'espace de stockage de la puce. Le programme d'amorçage est un petit programme qui s'exécute brièvement quand vous allumez ou réinitialisez la carte, et qui permet de charger le programme via le câble USB sans nécessiter un programmateur matériel séparé.

L'Arduino Uno est robuste et il est difficile de l'endommager, même quand on connecte un câble du mauvais côté (ne soyez pas cependant trop imprudent car si vous abusez il est toutefois possible de griller une broche de l'Arduino).

Arduino est facile à apprendre et les débutants peuvent accomplir toute une série de choses, simplement en activant et en désactivant des broches. À la différence du Raspberry Pi, vous pouvez brancher des capteurs ayant une résistance analogique directement sur l'Arduino sans avoir besoin d'un matériel externe, car l'Arduino a un convertisseur analogique-numérique intégré.

INSTALLATION DE BASE D'ARDUINO

Vous trouverez ci-dessous les instructions d'installation d'Arduino sous Linux, Windows et Mac.

Linux (Ubuntu)

Connectez l'Arduino à votre ordinateur avec un câble USB. L'Arduino est alimenté directement à partir du port USB si bien qu'aucune alimentation externe n'est requise. Démarrez l'application de terminal.

> Vous pouvez démarrer le terminal en ligne de commande de plusieurs façons différentes. Vous pouvez l'ouvrir à partir du menu principal avec **Applications→Accessoires→Terminal** (sous Xubuntu et les autres distributions basées sur XFCE, comme Debian, avec XFCE). Super-T, que l'on appelle aussi la *touche horrible* ou *touche Windows*, fonctionne sur de nombreux bureaux. Si vous utilisez Unity dans la distribution standard Ubuntu, cherchez « Terminal » dans le Dash (coin supérieur gauche).

Pour installer l'EDI Arduino, installez le paquet *arduino*. Voici comment faire avec la distribution Ubuntu :

```
$ sudo apt-get update
$ sudo apt-get -y install arduino
```

Accordez-vous la permission d'accéder au port série via le port USB (c'est nécessaire pour que l'environnement de développement d'Arduino fonctionne). La première commande vous ajoute

au groupe dialout et la seconde vous bascule dans ce groupe sans que vous ayez besoin de vous déconnecter puis de vous reconnecter :

```
$ sudo adduser $(whoami) dialout
$ newgrp dialout
```

Démarrez Arduino :

```
$ arduino
```

L'EDI Arduino s'ouvre.

Après vous être déconnecté, puis reconnecté, vous pouvez aussi démarrer l'EDI Arduino à partir des menus.

Vous êtes maintenant prêt à tester votre installation avec le programme Hello World.

Windows 7 et Windows 8

Téléchargez la dernière version du logiciel Arduino à partir de http://arduino.cc/en/Main/Software. Décompressez le fichier que vous avez téléchargé dans un emplacement qui vous convient (le *Bureau* ou le dossier *Téléchargements* par exemple).

Connectez votre Arduino Uno à votre ordinateur avec un câble USB. L'Arduino est alimenté directement à partir du port USB si bien qu'aucune alimentation externe n'est requise. Windows démarre un processus d'installation automatique des pilotes Arduino. La procédure peut échouer et faire apparaître un message d'erreur.

Si l'installation du pilote échoue :

1. Ouvrez l'Explorateur Windows, faites un clic droit sur Ordinateur et sélectionner Gérer.
2. À partir de la Gestion d'ordinateur, choisissez Gestionnaire de périphériques. Localisez Arduino Uno dans la liste, faites un clic droit et sélectionnez Mettre à jour le pilote.
3. Choisissez « Rechercher un pilote sur mon ordinateur ». Naviguez jusqu'au dossier Arduino que vous avez décompressé, ouvrez le répertoire *drivers*, choisissez *arduino.inf*, puis cliquez sur Suivant.
4. Windows va à présent installer le pilote.

Lancez l'EDI Arduino en faisant un double-clic sur l'icône Arduino située dans le dossier que vous avez décompressé.

Il est temps de tester votre installation avec le programme Hello World.

Mac OS X

Téléchargez la dernière version du logiciel Arduino à partir de http://arduino.cc/en/Main/Software. Décompressez le fichier que vous avez téléchargé et copiez-le dans votre dossier */Applications*.

Connectez votre Arduino Uno à votre ordinateur avec un câble USB. L'Arduino est alimenté directement à partir du port USB si bien qu'aucune alimentation externe n'est requise. Vous n'avez pas besoin d'installer de pilote sous OS X.

Lancez l'EDI Arduino en faisant un double-clic sur l'icône Arduino située dans le dossier */Applications*.

Il est temps de tester votre installation avec le programme Hello World.

Hello World

Maintenant que l'EDI est ouvert, vous pouvez exécuter un équivalent Arduino du programme « Hello World ».

Commencez par confirmer que vous avez sélectionné la bonne carte. Arduino Uno est la carte par défaut. Si vous avez une autre carte, comme un Mega ou un Leonardo, sélectionnez-la dans le menu **Outils→Type de carte**.

Il faut maintenant charger le programme de test Blink. Choisissez **Fichier→Exemples→1.Basics→Blink**. Cliquez sur le bouton **Téléverser** (représenté par une flèche vers la droite dans la barre d'outils ou bien choisissez dans le menu **Fichier→Téléverser**) pour compiler et charger le programme dans l'Arduino.

La première fois que vous faites cela, il est possible que l'Arduino affiche un message d'erreur : « Port série COM1 introuvable ». Cela est dû au fait que vous n'avez pas sélectionné le port série à utiliser (la connexion entre votre ordinateur et l'Arduino se réalise par le biais d'un port série USB). Sélectionnez votre port série à partir du menu déroulant. Sous Linux, c'est en général /dev/ttyACM0. Sous Mac, cela peut ressembler à /dev/usbmodem1234, et sous Windows, c'est l'un des ports COM.

> Si vous voyez un message d'erreur différent, choisissez votre port série à partir du menu **Outils→Port série**. Si vous n'arrivez pas à déterminer à quel port série votre Arduino est connecté, mémorisez la liste des ports, débranchez l'Arduino, et notez le port qui a disparu. C'est le port de l'Arduino. OS X liste chaque port deux fois, par exemple sous la forme /dev/cu.usbmodem1234 et /dev/tty.usbmodem1234. Les deux ports fonctionnent.

Quand le programme se charge, les diodes TX et RX (transmission et réception) de l'Arduino clignotent rapidement. Pour finir, quand le programme s'exécute, la petite lumière indiquée par un « L » clignote.

Le clignotement de la LED L signifie que l'installation s'est correctement déroulée et que vous venez d'exécuter votre premier sketch.

Félicitations ! Souvenez-vous de cette procédure simple : si vous êtes coincé et que vous ne savez même pas si votre Arduino exécute du code, exécutez à nouveau cet exemple « Hello World ». Quand vous démarrez un nouveau programme, commencez par un « Hello World » pour vous assurer que tout fonctionne.

Anatomie d'un programme Arduino

Un programme Arduino démarre en exécutant une fois le code à l'intérieur de la fonction setup(). Après cela, le code à l'intérieur de loop() est répété à l'infini (ou jusqu'à ce que vous coupiez l'alimentation). Reportez-vous à l'Exemple 2.1.

Exemple 2.1. blink.ino

```
// blink.ino - fait clignoter la LED L pour tester l'environnement de
développement
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
void setup() {  // 1
  pinMode(13, OUTPUT);  // 2
}
void loop() {   // 3
  digitalWrite(13, HIGH);       // 4
```

```
    delay(1000);  // ms    // 5
    digitalWrite(13, LOW);
    delay(1000);
}
```

1 Quand l'Arduino démarre, il exécute setup() une seule fois.
2 Configure la broche numérique D13 en mode OUTPUT, de manière à pouvoir la contrôler par programmation.
3 Quand setup() est terminée, Arduino appelle loop(). À la fin de loop(), la fonction est répétée à l'infini.
4 Définit la broche D13 à HIGH, ce qui ordonne à l'Arduino de fournir le +5 V à la broche.
5 Pendant ce délai, les broches restent en l'état. Ici la broche D13 reste à HIGH, de telle sorte que la LED intégrée de l'Arduino reste allumée. Au cours du délai suivant, elle passe à LOW, si bien qu'elle s'éteint. Elle reste allumée une seconde (1000 millisecondes), et s'éteint pendant une seconde, et ainsi de suite.

Intérêt des shields

Les shields sont des cartes qui se fixent au sommet de l'Arduino afin d'étendre ses possibilités ou de le rendre plus facile à utiliser (Figure 2.2). Il existe de nombreux shields différents, des simples shields de prototypage à des shields plus complexes, comme les shields Ethernet ou WiFi. L'un des gros avantages des shields est qu'ils réduisent le besoin de câbles supplémentaires ; cela est dû au fait qu'ils s'empilent au-dessus de l'Arduino et utilisent des connexions de broche à broche au lieu de liaisons par câble. Bien entendu, il n'existe pas toujours de shield pour tous vos besoins, mais c'est une option qu'il ne faut pas négliger.

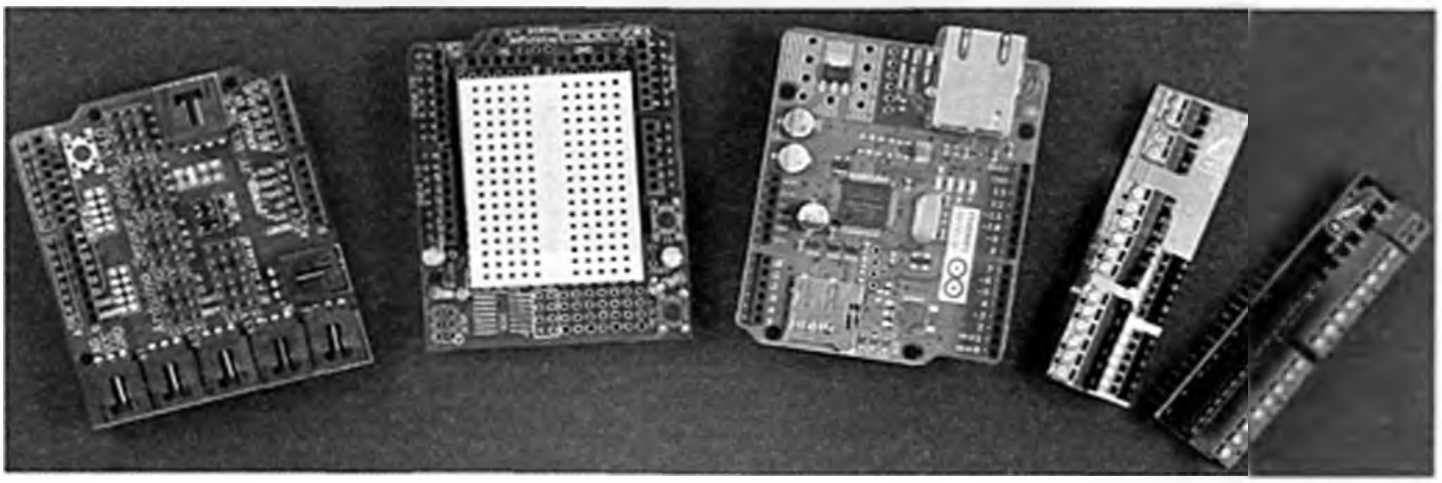


Figure 2.2 Shields

Certains shields n'embarquent pas d'électronique, mais sont simplement conçus pour faciliter le prototypage : ils mettent en général en évidence les connecteurs de l'Arduino de manière à ce qu'ils puissent être adjacents à une platine de test sans soudure, ce qui permet de connecter facilement les câbles. Notre shield favori est le ScrewShield, qui ajoute des « ailes » avec des borniers de chaque côté de l'Arduino. Cela élimine les câbles mal fixés, ce qui est sans doute la chose la plus ennuyeuse dans la création de prototypes.

Vous pouvez aussi envisager la conception de vos propres shields afin de créer des add-on Arduino faciles d'emploi et robustes, comme ceux qui sont illustrés à la Figure 2.3. Il suffit de souder les connecteurs des broches à un circuit imprimé de telle sorte qu'ils correspondent au brochage de l'Arduino.

Figure 2.3 Shields fabriqués par Andreas Zingerle

3 Mesurer les distances

Les deux façons les plus courantes de mesurer la distance sont les échos du son et la réflexion de la lumière. Pour éviter d'ennuyer les gens avec des signaux sonores et des clignotements constants, la fréquence sonore est en général si élevée que les êtres humains ne peuvent pas l'entendre, et la fréquence lumineuse est si basse que les humains ne peuvent pas la voir. On appelle ultrason le son à haute fréquence et infrarouge la lumière à basse fréquence.

> Même si l'infrarouge est invisible, nous allons vous montrer comment l'observer avec des objets courants.

Un capteur à ultrasons peut fournir des mesures exactes de la distance. Par exemple, il peut déterminer que la distance par rapport à un objet est de 36 cm.

Pour détecter la proximité des humains et des autres êtres vivants, les capteurs peuvent détecter la chaleur qu'ils dégagent. Cela permet de détecter la présence des choses chaudes dans la zone mesurée, mais pas leur distance exacte. La chaleur peut se déplacer de plusieurs manières différentes : la conduction, la convection et la radiation. Un capteur infrarouge passif mesure la chaleur dégagée sous la forme d'une lumière infrarouge.

À la différence d'un capteur infrarouge passif, un capteur de distance infrarouge actif envoie une lumière invisible et teste sa réflexion. Il peut dire si quelque chose est proche d'une distance donnée. Par exemple, un capteur infrarouge actif peut dire s'il y a un objet à moins de 30 cm, mais il ne sait pas s'il est à 5 cm ou 29 cm. Il existe cependant quelques rares capteurs qui estiment la distance à partir d'une lumière infrarouge réfléchie.

On utilise couramment l'infrarouge actif pour activer automatiquement un robinet ou un sèche-mains dans les toilettes publiques. Certaines poubelles automatiques ouvrent aussi leur couvercle quand on s'en approche. La lumière infrarouge rend les choses plus hygiéniques car vous n'avez pas à toucher les objets manipulés par d'autres personnes.

Les télémètres longue distance peuvent utiliser un rayon laser pour mesurer une distance. La plupart d'entre eux sont basés sur la factorisation de la vitesse de la lumière et du temps qu'elle met à réfléchir un rayon. Comme la lumière est très rapide, le circuit doit être capable de mesurer le temps très précisément. Cela rend les télémètres assez chers (les prix démarrent à une centaine d'euros) et ils sont donc beaucoup moins utilisés pour le prototypage avec Arduino ou Raspberry Pi que le son ou l'infrarouge.

EXPÉRIENCE : MESURE DE LA DISTANCE AVEC UN ULTRASON (PING)

Ping, 1, 2, 3... pong. Un capteur à ultrasons envoie un son puis mesure le temps que l'écho met à revenir. Comme vous savez que le son se déplace à environ 330 mètres par seconde, votre programme peut calculer la distance.

De nos jours, il existe de nombreux capteurs à ultrasons bon marché inspirés par le capteur Ping de chez Parallax (Figure 3.1). Plus loin, dans ce chapitre, nous allons étudier le code d'un de ces capteurs, comme le HC-SR04 (voir *Capteur à ultrasons HC-SR04* dans ce chapitre). Pour mieux comprendre tous les autres capteurs à ultrasons similaires à Ping, il est utile de se familiariser avec l'original et nous allons examiner dans un moment un peu de code. De plus, il s'agit d'un capteur extrêmement répandu et il est donc intéressant de le connaître.

> Pour comprendre comment un capteur à ultrasons mesure la distance, reportez-vous à la section *Calcul de l'écho* de ce chapitre.

Ping est un capteur populaire et ancien commercialisé par Parallax. Comparé à ses concurrents, il est un peu cher (une trentaine d'euros). Si vous avez besoin d'un grand nombre de capteurs de distance, vous préférerez sans doute quelque chose de moins cher, mais si vous n'en achetez qu'un seul, Ping est un excellent choix. Le capteur HC-SR04 ne coûte que deux euros et la seule différence entre Ping et le HC-SR04 tient à une broche (le HC-SR04 utilise une broche pour déclencher l'envoi d'une impulsion et une autre broche pour lire l'écho), mais les capteurs ont presque un code identique.

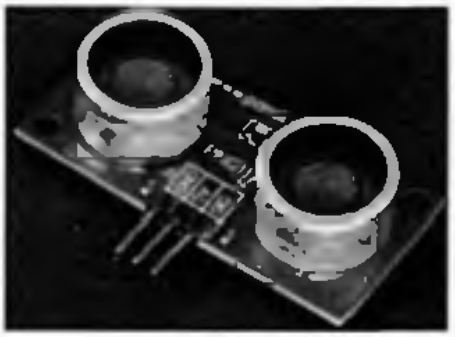


Figure 3.1 Capteur Ping

Ping pour Arduino

La Figure 3.2 illustre le schéma de câblage du capteur Ping et de l'Arduino. Montez le circuit, puis compilez et chargez le code à l'aide de l'IDE Arduino.

> Vous pouvez télécharger l'exemple de code à partir de http://makesensors.botbook.com.

Pour voir la mesure, utilisez le Moniteur série (dans l'IDE Arduino **Outils→Moniteur série**). Si vous obtenez des caractères incompréhensibles à la place du texte, assurez-vous d'avoir spécifié la même vitesse (bits/s ou « bauds ») dans votre code (Serial.begin) et dans le Moniteur série.

Même s'il y a beaucoup de code dans cet exemple, il est facile à réutiliser si bien que vous pouvez mesurer la distance dans vos propres projets : il suffit de copier la partie du code qui convient (la fonction distanceCm() et les variables globales) et de la coller dans votre propre code. Vous pouvez ensuite mesurer la distance avec une seule ligne de code :

```
int d=distanceCm();
```

Comme Ping fonctionne en écoutant l'écho d'un son, son placement est assez important. Si vous obtenez toujours la même mesure (par exemple 2 cm), assurez-vous que le faisceau large

du son ne rebondit pas sur quelque chose comme le bord de votre platine de test ou une table. Si vous placez Ping sur le bord d'une platine de test, vous n'obtiendrez pas de réflexion du son.

Vous pouvez facilement placer Ping loin de votre Arduino en utilisant un câble d'extension de servo, de type male-femelle. Ping n'a que trois broches si bien qu'il s'ajuste parfaitement à ce type de câble.

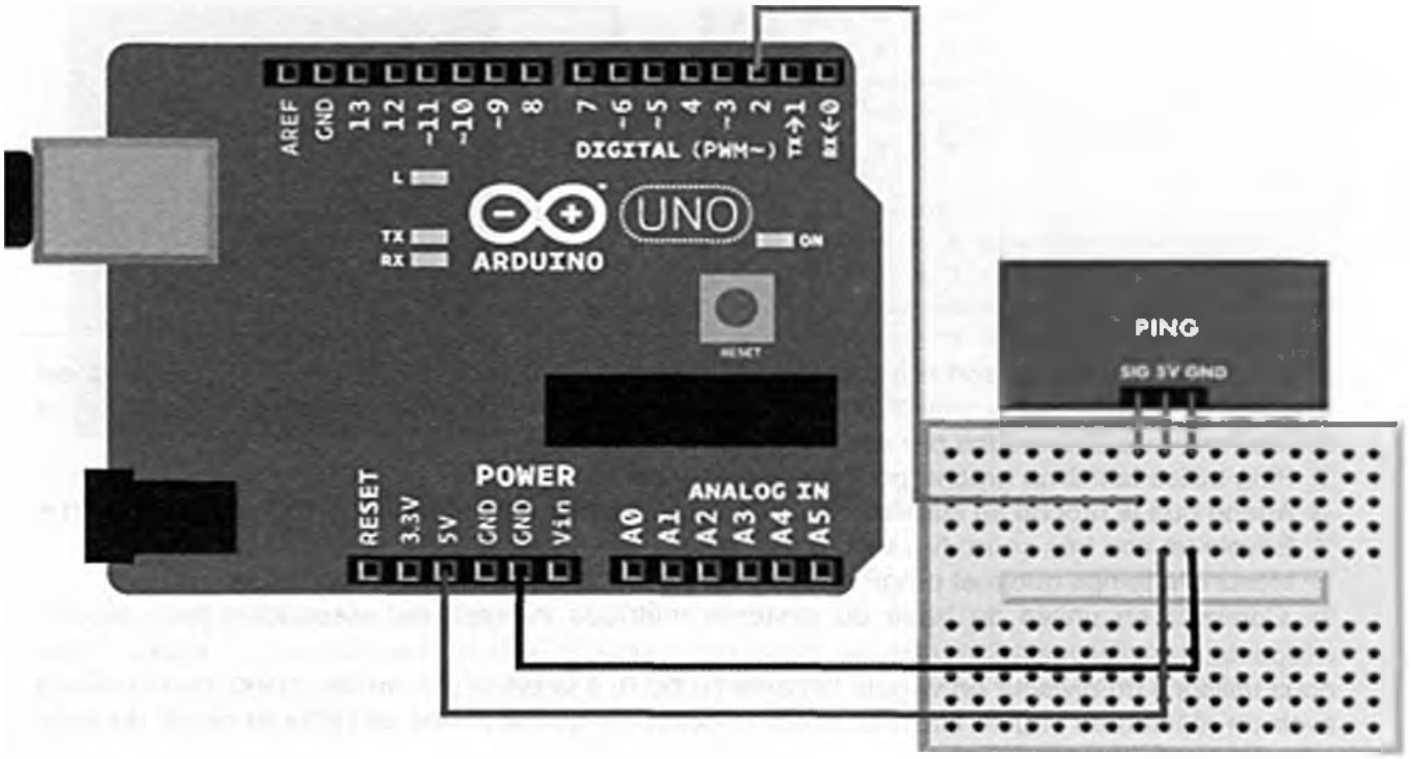


Figure 3.2 Circuit du capteur Ping pour Arduino

L'Exemple 3.1 liste le code complet de la mesure d'un capteur de distance à ultrasons Ping.

Exemple 3.1. distance_ping.ino

```
// distance_ping.ino - distance avec un capteur Ping à ultrasons
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int pingPin = 2;
float v=331.5+0.6*20; // m/s  // 1
void setup()
{
  Serial.begin(115200);
}
float distanceCm(){
  // envoie une impulsion sonore
  pinMode(pingPin, OUTPUT); // 2
  digitalWrite(pingPin, LOW);
  delayMicroseconds(3); // 3
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5); // 4
  digitalWrite(pingPin, LOW);
```

```
  // écoute l'écho
  pinMode(pingPin, INPUT);
  float tUs = pulseIn(pingPin, HIGH); // microsecondes  // 5
  float t = tUs / 1000.0 / 1000.0 / 2; // s  // 6
  float d = t*v; // m  // 7
  return d*100; // cm
}
void loop()
{
  int d=distanceCm();   // 8
  Serial.println(d, DEC);       // 9
  delay(200); // ms     // 10
}
```

1 Calcule la vitesse du son v à une température de 20° C (si votre température ambiante est très différente, modifiez la valeur 20 avec votre température ambiante en degrés C). La vitesse est de l'ordre de 340 mètres par seconde, soit 1200 km/h.
2 Ping utilise la même broche pour l'entrée et la sortie.
3 Attend que la broche se stabilise. 1 µs == 1 millionième de seconde, ou 10^{-6} s == 0,000001 s
4 Envoie un son très court. 5 µs, ou $5*10^{-6}$ s
5 Mesure le temps que met pingPin (D2) pour passer à LOW, en microsecondes.
6 Convertit en unités de base du système métrique international, secondes (voir http://fr.wikipedia.org/wiki/Unit%C3%A9s_de_base_du_syst%C3%A8me_international). Notez que nous utilisons un diviseur en virgule flottante (1000,0) à la place d'un entier, 1000, de manière à avoir un résultat en virgule flottante. Ceci ne constitue que la moitié de l'aller et retour du trajet effectué par l'ultrason.
7 La distance est le temps multiplié par la vitesse.
8 Mesure la distance et l'enregistre dans une nouvelle variable, d. C'est comme cela qu'il faut l'utiliser dans votre code.
9 Affiche la valeur de d sur le Moniteur série.
10 Placez toujours des délais dans vos boucles. Si vous exécutez votre sketch sans pause, vous allez monopoliser le CPU de l'Arduino et gaspiller de la puissance de calcul (le fait de faire quelque chose aussi vite que possible peut accaparer 100% de la puissance sur un CPU ne possédant qu'un seul cœur).

Ping pour Raspberry Pi

Montez le circuit pour Ping dans un Raspberry Pi (Figure 3.3) puis exécutez l'Exemple 3.2.

> Soyez prudent quand vous connectez quoi que ce soit au connecteur GPIO. Une mauvaise connexion peut facilement (au mieux) endommager une broche ou (au pire) griller le Raspberry Pi.
>
> Vous pouvez éviter les problèmes en déconnectant l'alimentation quand vous effectuez les connexions et bien vérifier le brochage avant de rallumer.

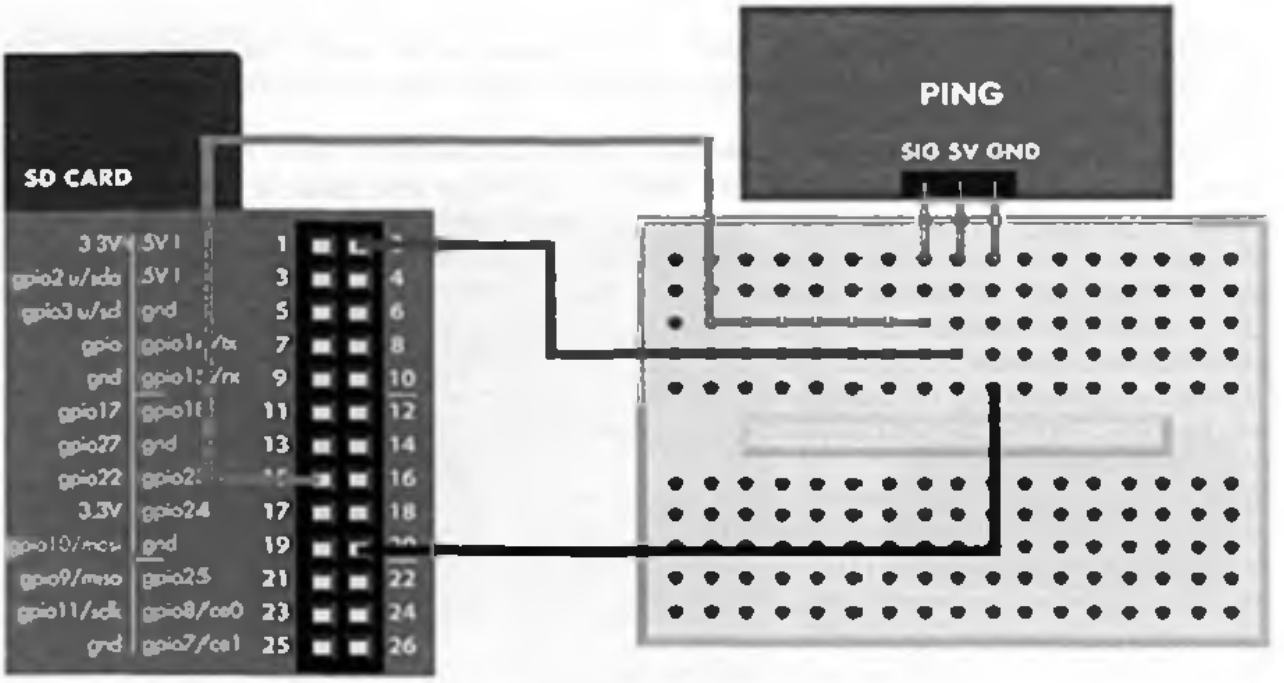


Figure 3.3 Circuit du capteur Ping pour Raspberry Pi

Exemple 3.2. distance_ping.py

```
# distance_ping.py - affiche la distance
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time # 1
import botbook_gpio as gpio # 2
def readDistanceCm():
        sigPin=22
        v=(331.5+0.6*20)
        gpio.interruptMode(sigPin, "both")      # 3
        gpio.mode(sigPin, "out")        # 4
        gpio.write(sigPin, gpio.LOW)    # 5
        time.sleep(0.5) # s
        gpio.write(sigPin, gpio.HIGH)   # 6
        time.sleep(1/1000.0/1000.0)     # 7
        gpio.mode(sigPin, "in") # 8
        #Read high pulse width
        t = gpio.pulseInHigh(sigPin) # s   # 9
        d = t*v
        d = d/2 # 10
        return d*100    # cm
def main():
        d = readDistanceCm()    # 11
        print "La distance est %.2f cm" % d # 12
        time.sleep(0.5)
if __name__ == "__main__":
        main()
```

1 L'importation de la bibliothèque *time* crée un espace de noms du même nom (time) qui contient les fonctions de la bibliothèque de telle sorte que cette ligne permet d'invoquer time.sleep(1) plus loin dans le code.
2 Pour importer vos propres bibliothèques, elles doivent être dans le même répertoire. Dans ces conditions, assurez-vous que ***botbook_gpio.py*** soit dans le même dossier que ***distance_ping.py***. Vous trouverez ce dossier dans les exemples de code disponibles à http://makesensors.botbook.com (voir *GPIO sans root* dans le chapitre 1 pour des informations sur la configuration de l'accès au GPIO du Raspberry Pi).
3 Le mode d'interruption both signifie que pulseInHigh() va mesurer une impulsion entière à partir du front montant du signal (de 0 à 1) jusqu'au front descendant (de 1 à 0).
4 Avec le capteur Ping, on bascule la même broche de mode « out » en mode « in » en fonction des besoins. Pour les autres capteurs, comme le HC-SR04, on utilise des broches séparées pour chaque fonction.
5 Désactive la broche et attend qu'elle se stabilise. Une demi-seconde est un délai convenable.
6 Démarre l'impulsion (front montant). C'est ici que le code critique pour le temps démarre.
7 Attend une microseconde (10^{-6} s), c'est-à-dire un millionième de seconde.
8 Définit la broche en mode « in ». Cela a pour conséquence de désactiver l'impulsion, ce qui crée le front descendant de l'impulsion courte.
9 Lit la durée de l'impulsion en secondes. gpio.pulseInHigh() mesure la durée de la totalité de l'impulsion, du début (front montant) jusqu'à la fin (front descendant). Raspbian exécute un système d'exploitation complet si bien que le minutage n'est pas aussi précis qu'avec Arduino car d'autres programmes s'exécutant sur le système peuvent affecter la mesure du temps.
10 L'aller n'est que la moitié de l'aller et retour.
11 C'est la ligne dont vous avez besoin pour mesurer la distance dans vos programmes.
12 Affiche la distance dans la fenêtre de terminal que ce programme exécute. La clause « %.2f » fait partie de la chaîne de formatage. Elle marque l'emplacement de la variable d. « %f » est en virgule flottante (décimal), et « .2 » signifie qu'il faut afficher deux décimales. Si vous utilisez simplement « print d », vous obtenez un nombre décimal très long.

CAPTEUR À ULTRASONS HC-SR04

Le HC-SR04 est similaire au Ping, mais il coûte dix fois moins cher. Le code de ce capteur est presque identique à celui de Ping, sauf que le HC-SR04 utilise des broches séparées pour déclencher le son et écouter l'écho. Pour des explications du code détaillées, reportez-vous à *Ping pour Arduino* et *Ping pour Raspberry Pi* ; les explications de cette section se concentrent sur les différences.

> Pour comprendre comment un capteur à ultrasons mesure la distance, allez jusqu'à la section *Calcul de l'écho* dans ce chapitre.

Figure 3.4 Capteur à ultrasons HC-SR04

HC-SR04 pour Arduino

Montez le circuit illustré à la Figure 3.5 et chargez le code.

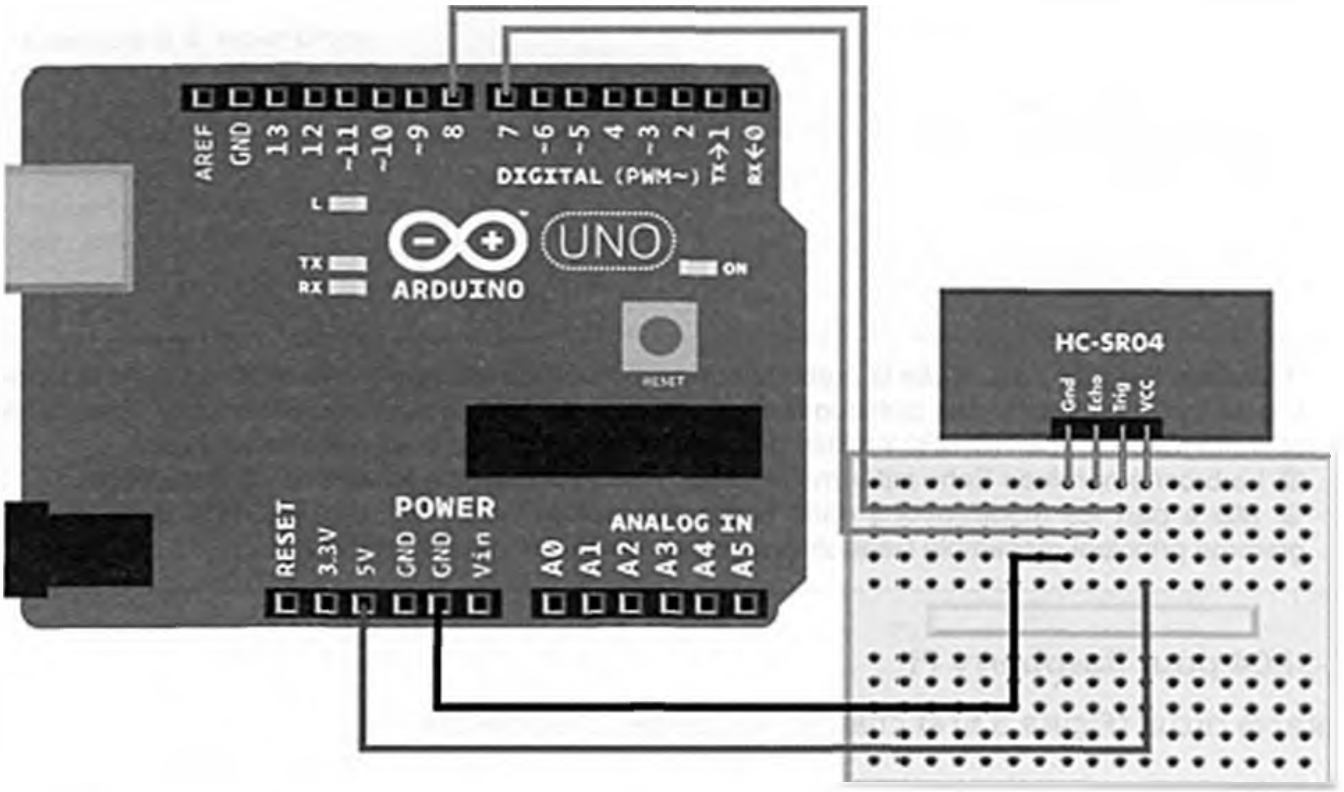


Figure 3.5 Circuit du capteur HC-SR04 pour Arduino

Exemple 3.3. hc-sr04.ino

```
// hc_sr04.ino - affiche la distance sur le port série
// (c) BotBook.com - Karvinon, Karvinon, Valtokari
int trigPin = 8;
int echoPin = 7;
float v=331.5+0.6*20; // m/s
void setup()
{
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);      // 1
  pinMode(echoPin, INPUT);       // 2
}
float distanceM(){
  // envoie une impulsion sonore
  digitalWrite(trigPin, LOW);
  delayMicroseconds(3);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(trigPin, LOW);
  // écoute l'écho
  float tUs = pulseIn(echoPin, HIGH); // microsecondes
```

```
  float t = tUs / 1000.0 / 1000.0 / 2; // s
  float d = t*v; // m
  return d*100; // cm
}
void loop()      // 3
{
  int d=distanceM();
  Serial.println(d, DEC);
  delay(200); // ms
}
```

1 Avec le Ping, on ne suit pas la pratique normale de définir les modes des broches dans la fonction setup() ; on les modifie puisque le Ping utilise la même broche pour déclencher l'impulsion et lire la réflexion. Le HC-SR04 utilise une broche nommée Trig pour déclencher le son.
2 La broche nommée Echo retourne le temps écoulé pendant la lecture de l'écho réfléchi.
3 Mis à part les modifications dans la fonction setup, l'utilisation du HC-SR04 dans le programme principal ressemble beaucoup au code utilisé par le capteur Ping.

HC-SR04 pour Raspberry Pi

Montez le circuit (Figure 3.6) et chargez le code de l'Exemple 3.4.

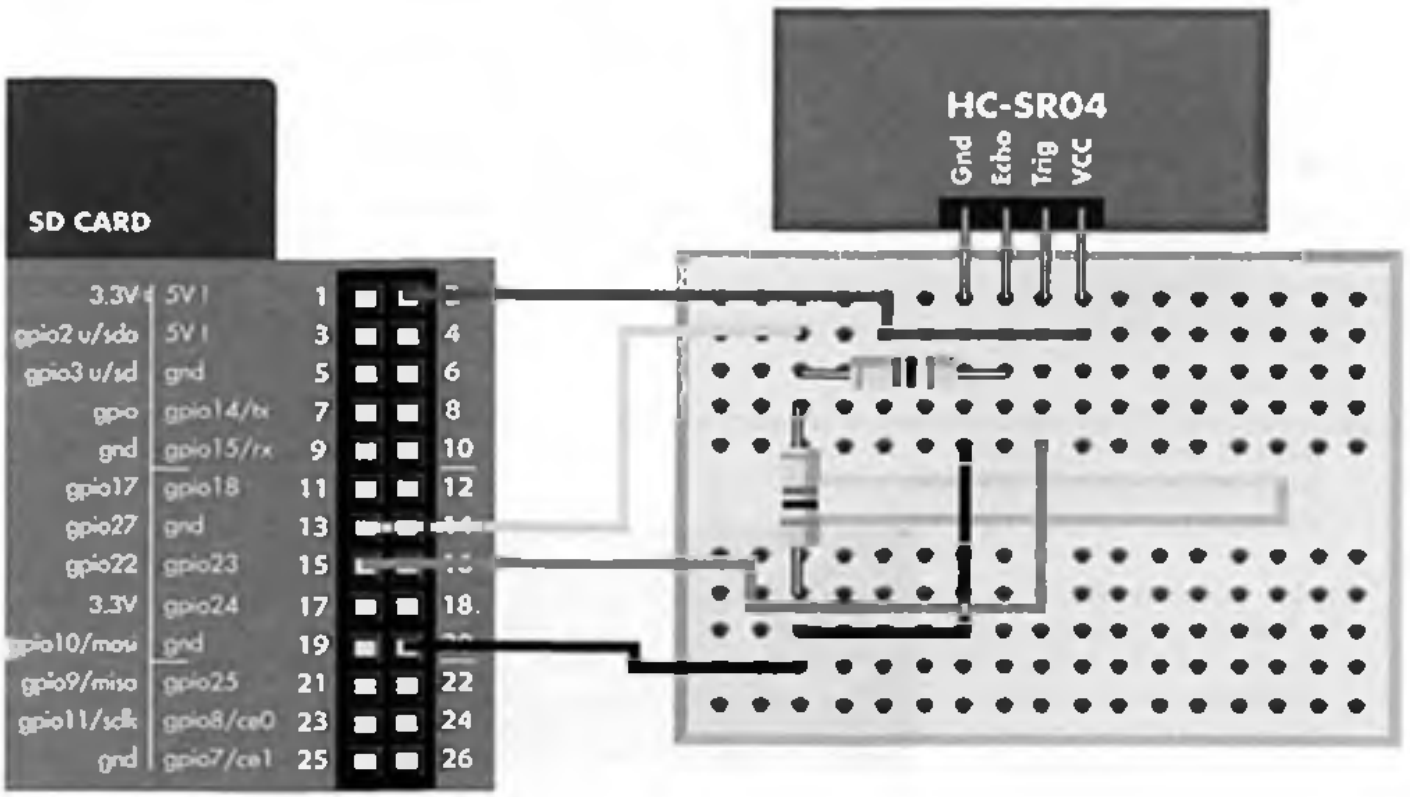


Figure 3.6 Circuit du capteur HC-SR04 pour Raspberry Pi

Vous noterez qu'en plus des câbles vous devez aussi ajouter deux résistances de 10 kΩ (pour identifier les résistances, vous pouvez utiliser la méthode décrite dans la section *Règle de la troisième bande* du chapitre 1). Le code est très similaire à celui de Ping.

Exemple 3.4. hc-sr04.py

```
# hc-sr04.py - affiche la distance par rapport à un objet en cm
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio
def readDistanceCm():
        triggerPin = 22 # 1
        echoPin = 27
        v=(331.5+0.6*20) # m/s
        gpio.mode(triggerPin, "out")
        gpio.mode(echoPin, "in")
        gpio.interruptMode(echoPin, "both")
        gpio.write(triggerPin, gpio.LOW)
        time.sleep(0.5)
        gpio.write(triggerPin, gpio.HIGH)
        time.sleep(1/1000.0/1000.0)
        gpio.write(triggerPin, gpio.LOW)
        t = gpio.pulseInHigh(echoPin) # s
        d = t*v
        d = d/2
        return d*100    # cm
def main():
        d = readDistanceCm()    # 2
        print "Distance is %.2f cm" % d
        time.sleep(0.5)
if __name__ == "__main__":
        main()
```

1 La seule différence par rapport à l'exemple Ping est que le HC-SR04 utilise deux broches (Trig et Echo).
2 On peut lire le résultat du HC-SR04 comme celui du Ping.

Pourquoi le HC-SR04 a-t-il besoin d'une résistance alors que ce n'est pas le cas du Ping ?

Selon la notice technique du HC-SR04, la sortie est de niveau TTL, ce qui correspond au +5 V.

La notice technique du Ping promet la compatibilité avec le +3,3 V.

Nous avons vérifié ces valeurs maximales en mesurant la sortie. La tension maximale des broches du connecteur GPIO du Raspberry Pi est de +3,3 V, et l'envoi du +5 V endommagerait les broches du Raspberry Pi.

Calcul de l'écho

Un orage gronde dans les parages ? Vous pouvez estimer la distance de l'endroit où frappe la foudre en comparant le moment où vous voyez l'éclair et celui où vous entendez le tonnerre.

Comptez le nombre de secondes après l'apparition de l'éclair. Chaque seconde correspond à 330 mètres à partir de l'emplacement de l'impact de foudre (ce nombre dépend en fait de la température de l'air et nous y reviendrons dans un instant).

330 mètres par seconde, cela signifie que le son se déplace approximativement d'un kilomètre en trois secondes. On voit la lumière presque instantanément, mais le son met plus de temps à nous atteindre.

Un capteur à ultrasons mesure en général les distances sur une plage variant de 3 cm à 6 m. Si vous achetez un instrument de mesure (télémètre à ultrason) dans un magasin de bricolage, il pourra mesurer une distance d'une vingtaine de mètres (il utilise un cône pour projeter le son et un thermomètre pour calibrer la vitesse du son de la température ambiante).

Le temps mis par le son pour faire un centimètre est très court, de l'ordre de 30 millionièmes de seconde. Comment arriver à gérer un nombre pareil ?

Pour des problèmes difficiles, particulièrement ceux qui impliquent des nombres très petits ou très grands, il est utile de les modéliser comme des problèmes analogiques en se servant de quantités que l'on rencontre dans la vie quotidienne.

Par exemple, si je conduis pendant deux heures (*t*) à la vitesse (*v*) de 50 km par heure, je peux avec *t* et *v* calculer la distance (*d*) :

```
t = 2 h
v = 50 km/h
d = t•v = 2 h • 50 km / h =
    2•50 km • h/h = 100 km
```

Maintenant que l'on sait que deux heures équivalent à 100 km dans ce système, essayons la même formule avec une durée très courte (3,33 millisecondes), une vitesse beaucoup plus rapide et des unités de base en mètres et secondes (le préfixe milli signifie un millième si bien qu'une milliseconde représente un millième de seconde).

```
t = 3,33 ms = 0,00333 s
v = 330 m/s
d = t•v = 0,00333 s • 330 m/s = 1,10 m
```

C'est la manière dont vous pouvez mesure la distance dans votre programme avec un capteur à ultrasons : si le son met 3,33 millisecondes à revenir, il a parcouru 1,1 m.

Si vous lisez du code écrit par d'autres, vous constaterez que certains mesurent l'inverse des mètres par seconde, c'est-à-dire les secondes par mètre, exprimées sous la forme de millisecondes/mètre :

```
1/v = 1/(330 m/s) = 0,00303 s/m = 3,03 ms/m
```

Il faut à peu près 3 millisecondes au son pour parcourir un mètre.

> Le son se déplace plus vite quand il fait chaud. Le son est la vibration de l'air, et les vibrations se déplacent mieux si les molécules d'air vibrent déjà à cause de la chaleur. Si vous vivez dans un endroit chaud, nous vous envions car vous aurez sans doute besoin de moins de calibrage. Dans le nord de la Finlande, il peut faire +22° C à l'intérieur et –40° C à l'extérieur, ce qui fait une différence

de température de 60° C. Une telle différence modifie les mesures. La température (*T*) affecte la vitesse du son (*r*) selon la formule suivante :

v = (331.3+0.606*T) m/s

Cette formule donne la vitesse du son en pratique (343 m/s à 20° C). Si vous commencez à finasser avec ça, vous allez devoir calibrer de nombreux facteurs. Si vous montez au sommet d'une montagne ou si vous vivez dans un sous-marin, vous devez aussi prendre en compte le changement de pression atmosphérique. Si vous passez du Sahara à une buanderie, il faudra également calibrer l'humidité de l'air. Cela étant, la plupart des télémètres à ultrasons commercialisés ne calibrent que la température.

Quand on utilise ces calculs dans le code, il faut les placer au début du code. Arduino ou Raspberry Pi peuvent les calculer en un instant, et les calculs effectués en dehors de `loop()` ne sont accomplis qu'une seule fois. Assurez-vous de commenter ces calculs et vous vous remercierez de pouvoir comprendre votre code quand vous le relirez une semaine plus tard...

Expérience : des objets invisibles

Vous pouvez facilement leurrer un capteur à ultrasons de telle sorte qu'il pense qu'il n'y a rien devant lui.

Fixez le capteur à un outil et pointez-le sur un objet solide et plat.

Chargez le code et ouvrez le Moniteur série comme vous l'avez fait précédemment dans ce chapitre. Vous devriez voir s'afficher une distance normale.

Ensuite, placez un oreiller moelleux ou un objet en peluche entre le capteur et l'objet solide (Figure 3.7).

Vérifiez à nouveau le Moniteur série. Est-ce que l'objet solide est toujours là ?

Les plans inclinés sont un autre point faible des capteurs à ultrasons. Enlevez l'objet en peluche et commencez à incliner l'objet solide qui fait face au capteur. Continuez à vérifier le Moniteur série au fur et à mesure que vous inclinez l'objet.

Pourquoi cela se produit-il ? Les objets mous (comme notre lapin illustré à la Figure 3.7) absorbent tant les sons qu'il n'y a pas suffisamment d'écho. Et pour les plans inclinés, ils renvoient le son, mais dans la mauvaise direction (par vers le capteur). C'est la manière dont les avions furtifs arrivent à tromper les radars.

Figure 3.7 Test du capteur Ping avec un objet mou

EXPÉRIENCE : DÉTECTION DES OBSTACLES AVEC L'INFRAROUGE (CAPTEUR DE DISTANCE INFRAROUGE)

Un détecteur infrarouge (Figure 3.8) est plus fiable qu'un capteur à ultrasons, mais il est moins polyvalent.

Vous ne pouvez le leurrer aussi facilement qu'un capteur à ultrasons, comme dans l'expérience précédente.

Mais un détecteur infrarouge ne peut vous indiquer que la présence de quelqu'un et il ne sait pas mesurer la distance.

Et comme le Soleil est une source énorme de lumière infrarouge, il est suffisamment fort pour aveugler un détecteur infrarouge.

Figure 3.8 Détecteur infrarouge

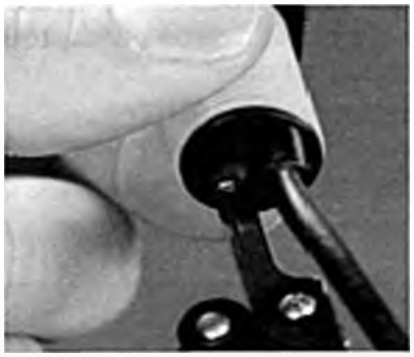


Figure 3.9 Vous pouvez ajuster la distance à laquelle le capteur détecte les obstacles

Détecteur infrarouge pour Arduino

La Figure 3.10 montre comment connecter un Arduino à un capteur infrarouge. Le sketch est listé dans l'Exemple 3.5.

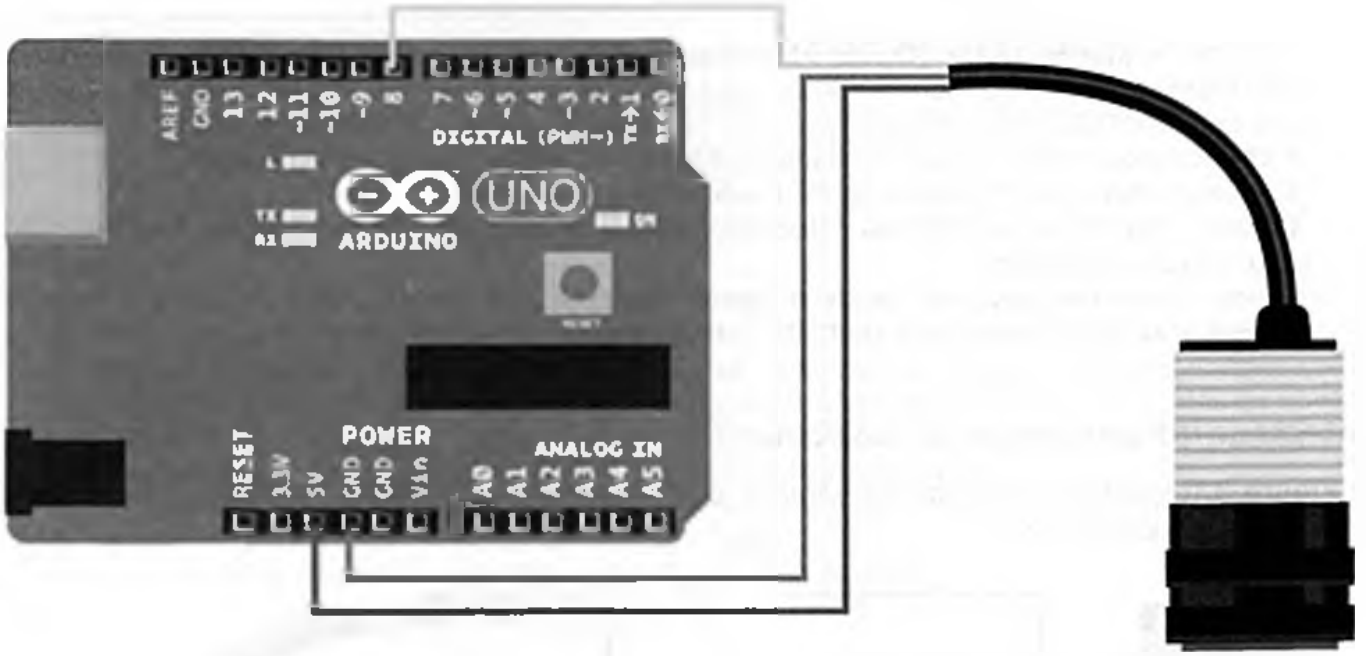


Figure 3.10 Connexions d'un capteur infrarouge à l'Arduino

Exemple 3.5. adjustable_infrared_sensor_switch.ino

```
// adjustable_infrared_sensor_switch.ino - affiche la détection sur le port
série et allume une LED.
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int sensorPin = 8;
const int ledPin = 13;
//Valeur du capteur
int switchState = 0;
void setup() {
  Serial.begin(115200); // 1
  pinMode(sensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
}
void loop() {
  switchState = digitalRead(sensorPin); // 2
  Serial.println(switchState);  // 3
  if(switchState == 0) {
    digitalWrite(ledPin, HIGH);
    Serial.println("Objet détecté !"); // 4
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(10); // ms       // 5
}
```

1 Ouvrez le Moniteur série (**Outils→Moniteur série**). Vous devez définir la même vitesse dans votre code et dans le Moniteur série. La vitesse la plus rapide est 115 200 bits/seconde. Si vous avez un câble USB peu fiable (ou très long), modifiez la vitesse à 9 600 bits/seconde.
2 Un détecteur infrarouge est similaire à un bouton. C'est la ligne qui lit le capteur.
3 Affiche l'état de la broche du capteur à des fins de débogage.
4 L'état 0 signifie qu'un objet est détecté. On allume la LED intégrée de l'Arduino pour indiquer qu'un objet a été détecté.
5 Vous devez toujours avoir un délai, même très court, dans loop(). Cela empêche le sketch d'utiliser 100% en permanence du CPU de l'Arduino.

Détecteur infrarouge pour Raspberry Pi

La Figure 3.11 illustre le schéma de câblage pour Raspberry Pi. Le code Python correspondant figure dans l'Exemple 3.6.

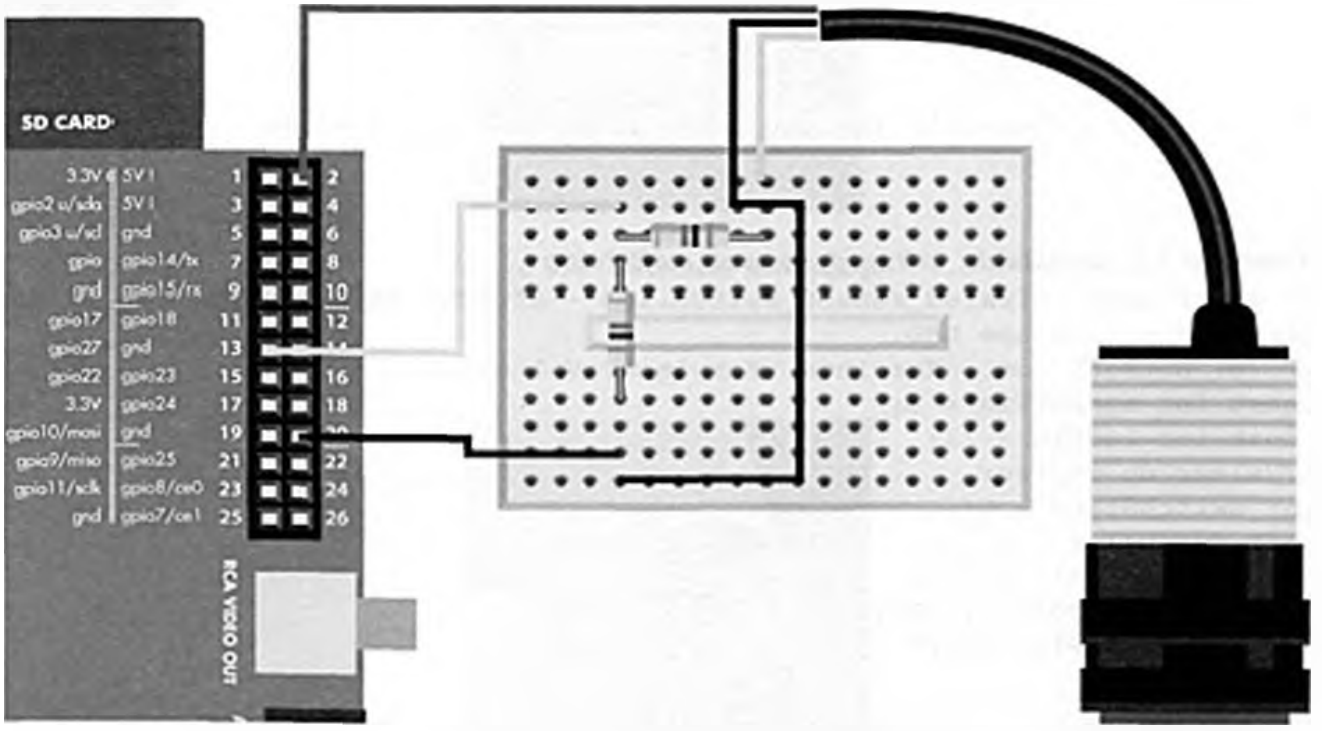


Figure 3.11 Connexions du capteur infrarouge à un Raspberry Pi

Exemple 3.6. adjustable-infrared-sensor-switch.py

```
# adjustable-infrared-sensor-switch.py - lit un détecteur infrarouge
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio      # 1
def main():
        switchPin = 27
        gpio.mode(switchPin, "in")      # 2
        x = gpio.read(switchPin)        # 3
        if( x == gpio.LOW ):    # 4
```

```
                print "Il y a quelque chose dans la plage de détection"
        else:
                print "Il n'y a rien dans la plage de détection"
        time.sleep(0.1)
if __name__ == "__main__":
        main()
```

1 Importe la bibliothèque *botbook_gpio*. Elle doit être dans le même répertoire que ce code, si bien que vous devez vous assurer que *botbook_gpio.py* figure dans le même dossier que *adjustable-infrared-sensor-switch.py*. Vous trouverez ce dossier dans les exemples de code disponibles à http://makesensors.botbook.com (voir *GPIO sans root* dans le chapitre 1 pour des informations sur la configuration de l'accès au GPIO du Raspberry Pi).
2 Configure la broche à laquelle le détecteur est connecté et la met en mode d'entrée.
3 Lit l'état de la broche et le stocke dans la variable x.
4 Si la broche est à LOW, cela signifie qu'un objet a été détecté.

EXPÉRIENCE : COMMENT VOIR L'INFRAROUGE

Comme nous l'avons déjà mentionné, l'infrarouge est en dehors de la plage de la lumière visible. Comment arriver à voir l'invisible ? Vous pouvez utiliser des jumelles de vision nocturne ou bien n'importe quel appareil de photo numérique bon marché.

Essayez de regarder un capteur infrarouge par le biais de l'appareil photo de votre smartphone. Vous devriez être capable de voir une lueur violette sur l'émetteur infrarouge illustré à la Figure 3.12. Baissez les lumières et tirez les rideaux pour le rendre encore plus visible. Ce n'est pas aussi excitant que la vision nocturne, mais c'est un moyen simple et rapide de voir que l'émetteur fonctionne.

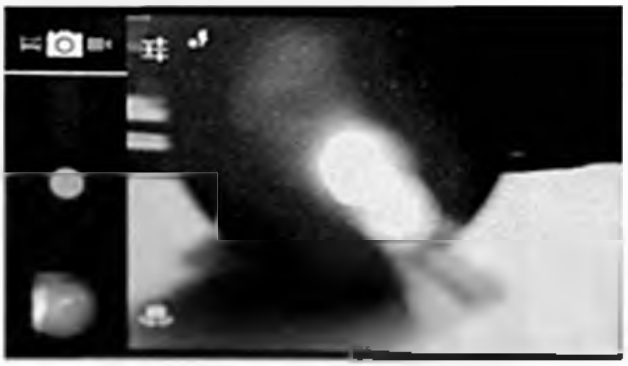


Figure 3.12 L'appareil photo d'un téléphone voit les capteurs infrarouges

Vous pouvez aussi faire l'expérience avec un reflex numérique haut de gamme. Vous devez cependant être conscient que certains APN ont de puissants filtres infrarouges qui empêchent les longueurs d'ondes indésirables d'apparaître sur vos photos (cela fonctionne avec les APN moins coûteux car les filtres infrarouges ne sont pas aussi bons si bien que la lumière infrarouge parvient à exciter certains capteurs de votre appareil photo). Si c'est le cas, vous pouvez aller dans une pièce sombre et régler votre APN sur une vitesse d'obturation très lente (par exemple, plusieurs secondes) et prendre une photo en utilisant un trépied. Comme l'infrarouge est la seule lumière de la pièce, elle finira par apparaître sur la photo.

Si vous avez la chance de disposer de jumelles de vision nocturne, c'est l'instrument idéal pour observer la manière dont les capteurs infrarouges fonctionnent car elles amplifient la lumière visible, mais sont particulièrement avides de spectre infrarouge.

Les modèles les moins chers sont basés sur un rayon infrarouge qu'ils produisent eux-mêmes. Si le modèle que vous utilisez a un émetteur infrarouge, assurez-vous de le désactiver (ou obturez-le) afin de voir la lumière plus faible de votre capteur de distance infrarouge.

Ce qui est amusant avec les jumelles de vision nocturne, ce n'est pas seulement de constater que votre capteur fonctionne, mais aussi de voir la réflexion que la lumière infrarouge crée autour des autres objets (Figure 3.13).

Figure 3.13 Capteur infrarouge vu par des jumelles de vision nocturne

EXPÉRIENCE : SUIVEZ LE MOUVEMENT GRÂCE À L'INFRAROUGE (ŒIL COMPOSÉ INFRAROUGE)

Un œil composé comporte plusieurs LED et transistors sensibles à l'infrarouge ; il peut suivre le mouvement dans les limites d'une vingtaine de centimètres.

Même s'il n'a qu'un capteur, chacun des transistors sensibles à la lumière infrarouge peut être lu séparément. La correction de la lumière ambiante est réalisée en désactivant les LED infrarouges et en comparant les valeurs.

Figure 3.14 Œil composé

Le nom exact de ce capteur est « œil composé IR » (en anglais *IR compound eye*). Si vous vous mettez à inventer et à vendre des capteurs, essayez de leur donner un nom de code unique en plus d'un nom générique. Un nom de code unique facilitera la recherche du composant.

Si vous voulez améliorer les résultats de votre œil composé, vous devez le calibrer. Attendez la nuit qu'il n'y ait plus de lumière infrarouge ambiante. Si vous ne pouvez pas attendre qu'il fasse nuit, allez dans votre cave ou bien dans une pièce sans fenêtre.

Figure 3.15 Véritable œil composé

Montez le circuit illustré à la Figure 3.16 de manière à pouvoir mesurer les valeurs.

Placez du papier devant le capteur (à peu près à 20 cm) et regardez la variation des valeurs pour chaque broche. Les valeurs doivent être presque identiques (+/- 100) avec le papier. Si l'une des valeurs est trop élevée, vous pouvez bloquer une partie de la lumière infrarouge avec un adhésif opaque ou un emballage. Si la valeur est trop basse, bloquez une partie de la lumière infrarouge allant vers d'autres capteurs.

> Quand vous utilisez ce capteur, vous mesurez une résistance analogique. Pour un exemple très simple de lecture de résistance analogique, consultez l'*Expérience : potentiomètre (résistance variable)* au chapitre 5.

Œil composé pour Arduino

La Figure 3.16 illustre le schéma de câblage de l'Arduino et de l'œil composé. Le sketch est listé dans l'Exemple 3.7.

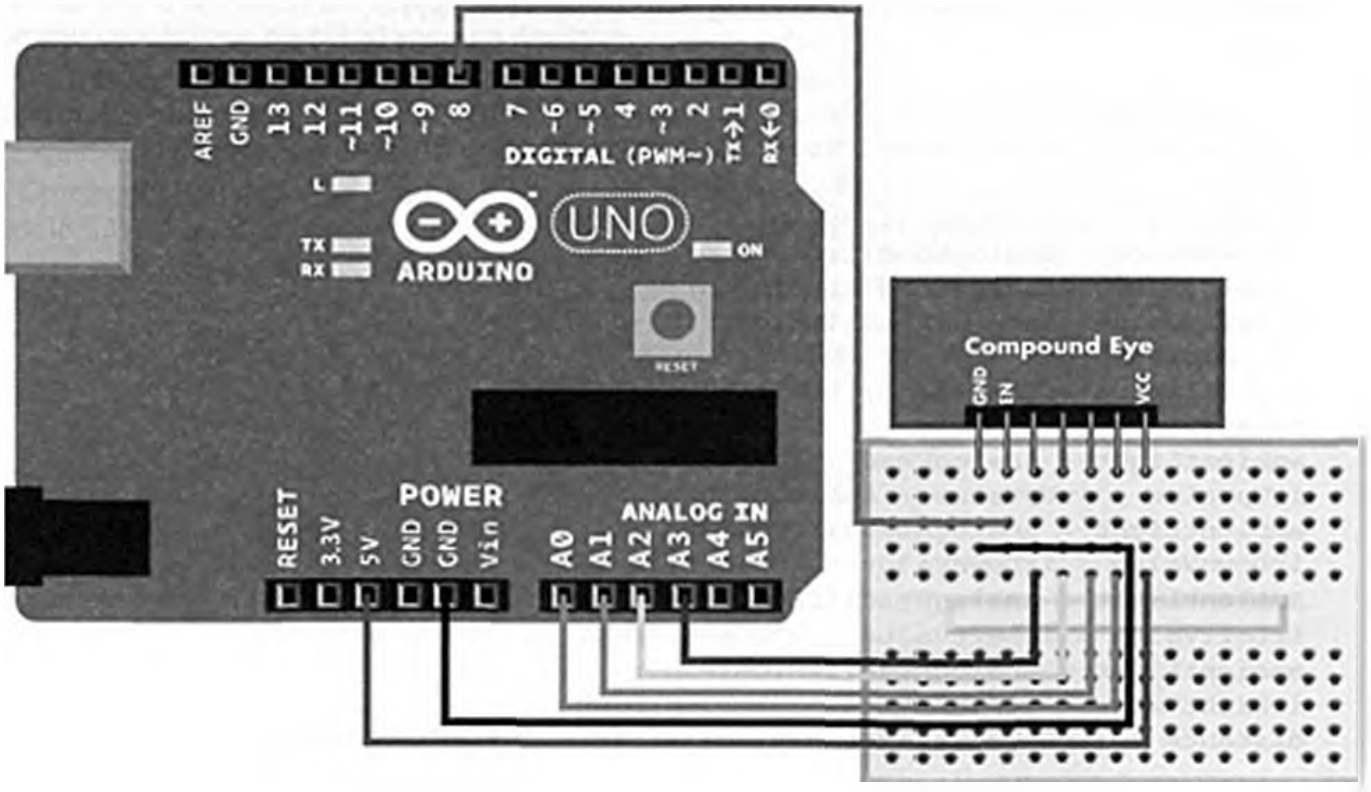


Figure 3.16 Connexions d'un œil composé sur un Arduino

Exemple 3.7. compound_eye.ino

```
// compound_eye.ino - affiche la distance et la direction sur le port série
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int irEnablePin = 8;      // 1
const int irUpPin = 0;
const int irDownPin = 2;
const int irLeftPin = 1;
const int irRightPin = 3;
int distance = 0;       // 2
int irUpValue = 0;
int irDownValue = 0;
int irLeftValue = 0;
int irRightValue = 0;
void setup() {
  Serial.begin(115200);
  pinMode(irEnablePin, OUTPUT);
}
void loop() {
  readSensor(); // 3
  Serial.print("Valeurs : "); // 4
  Serial.print("irUpValue"); Serial.print(irUpValue); Serial.print(",");
  Serial.print("irDownValue"); Serial.print(irDownValue); Serial.print(",");
  Serial.print("irLeftValue"); Serial.print(irLeftValue); Serial.print(",");
  Serial.print("irRightValue"); Serial.print(irRightValue);
Serial.print(",");
  Serial.print("distance"); Serial.println(distance);
  delay(100);
}
void readSensor() {
  digitalWrite(irEnablePin, HIGH);      // 5
  delay(5); // ms       // 6
  irUpValue = analogRead(irUpPin);
  irDownValue = analogRead(irDownPin);
  irLeftValue = analogRead(irLeftPin);
  irRightValue = analogRead(irRightPin);
  int ambientLight = 0; // 7
  digitalWrite(irEnablePin, LOW);       // 8
  delay(5);
  ambientLight = analogRead(irUpPin);   // 9
  irUpValue = irUpValue - ambientLight; // 10
  ambientLight = analogRead(irDownPin);
  irDownValue = irDownValue - ambientLight;
  ambientLight = analogRead(irLeftPin);
  irLeftValue = irLeftValue - ambientLight;
  ambientLight = analogRead(irRightPin);
  irRightValue = irRightValue - ambientLight;
  distance = (irUpValue+irDownValue+irLeftValue+irRightValue) / 4; // 11
}
```

1 Les numéros des broches sont déclarés sous la forme de constantes (const), si bien que vous ne pourrez pas les modifier ailleurs dans le sketch (si vous tentez d'écrire du code qui leur réassigne une valeur, même accidentellement, vous obtiendrez une erreur quand vous voudrez vérifier ou charger le sketch).
2 Les valeurs du capteur sont stockées dans des variables globales. Les variables globales sont disponibles dans toutes les fonctions. En C et C++ (sur lequel Arduino est basé), c'est une bonne pratique d'initialiser les variables en même temps qu'on les déclare (par exemple, int foo = 0;).
3 readSensor() ne retournera pas de valeur. Au lieu de cela, elle modifie certaines variables globales. De cette manière, elle peut modifier plusieurs valeurs quand vous l'exécutez.
4 Affiche les résultats. Serial.print() n'ajoute pas de nouvelle ligne (à la différence de Serial.println()).
5 Active la LED infrarouge pour illuminer la cible à mesurer.
6 Attend que les entrées se stabilisent.
7 Commence à mesurer la lumière ambiante (par exemple, la lumière invisible infrarouge du soleil).
8 Désactive la LED infrarouge. Toutes les lumières détectées proviennent maintenant des sources ambiantes.
9 Utilise chacun des transistors sensibles à l'infrarouge pour mesurer la lumière ambiante.
10 Supprime la valeur de la lumière ambiante de chacun des capteurs.
11 Calcule la distance moyenne à partir des quatre transistors sensibles à l'infrarouge.

Œil composé pour Raspberry Pi

L'œil composé contient huit capteurs sensibles à l'infrarouge, qui sont connectés par paires, si bien qu'il y a un total de quatre capteurs que l'on peut lire. Chaque capteur infrarouge est lu comme un capteur de résistance analogique.

Le Raspberry Pi nécessite un convertisseur analogique-numérique externe pour lire les capteurs infrarouges. Une puce MCP3002 peut lire deux entrées analogiques. Comme nous devons lire quatre capteurs, on utilisera deux puces MCP3002.

Ce circuit (illustré à la Figure 3.17) comporte de nombreux éléments, mais son principe est simple : il y a quatre capteurs de résistance analogique, et on les lit un par un. Montez le circuit illustré puis exécutez le code listé dans l'Exemple 3.8.

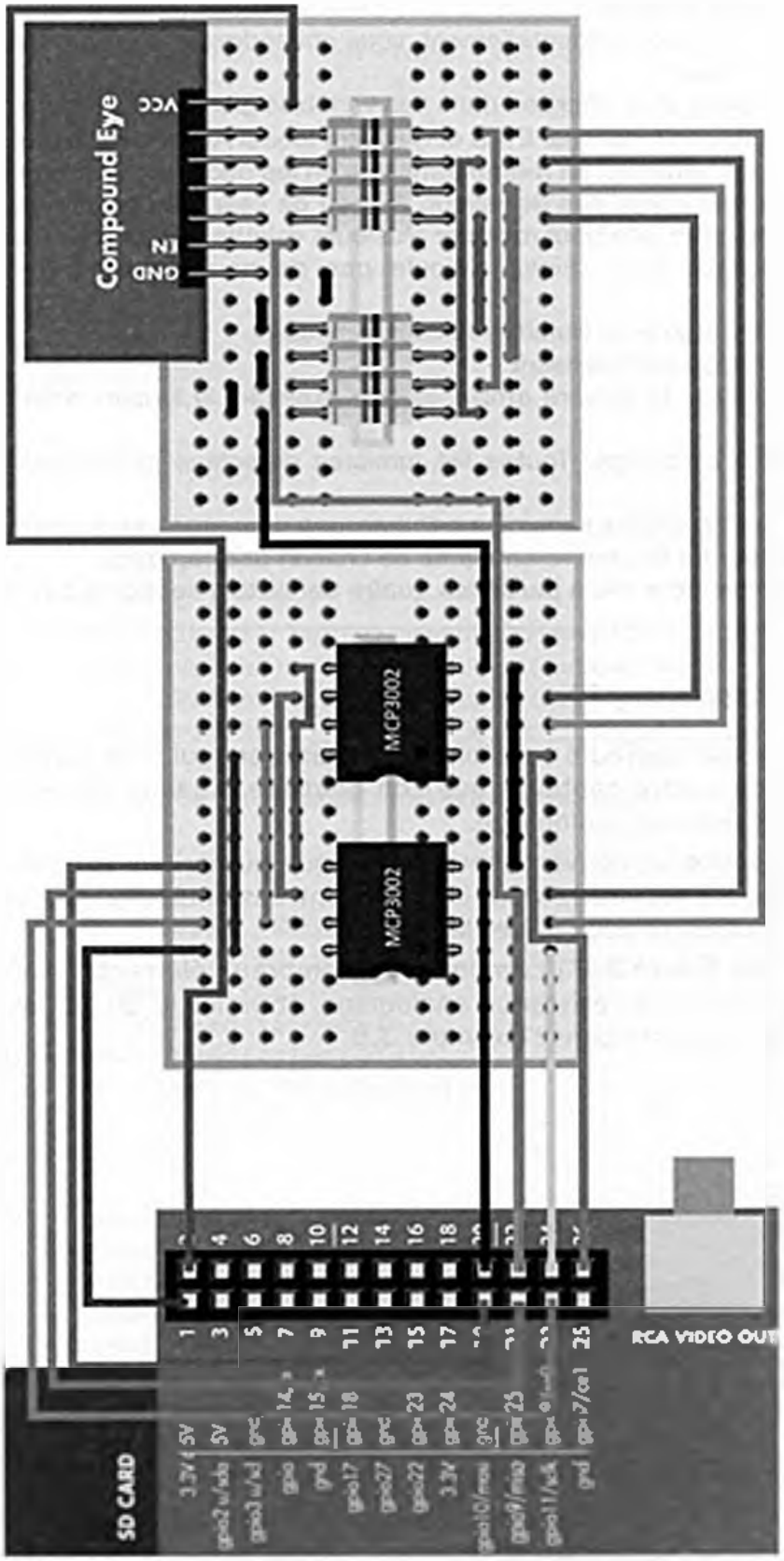


Figure 3.17 Connexions d'un œil composé sur un Raspberry Pi

Exemple 3.8. compound_eye.py

```
# compound_eye.py - lit la distance et la direction.
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio      # 1
import botbook_mcp3002 as mcp    # 2
irUpValue = 0    # 3
irDownValue = 0
irLeftValue = 0
irRightValue = 0
distance = 0
def readCompoundEye():
        global irUpValue,irDownValue,irLeftValue,irRightValue,distance  # 4
        ledPin = 25
        gpio.mode(ledPin, "out")         # 5
        gpio.write(ledPin, gpio.HIGH)
        # attend que les capteurs soient prêts
        time.sleep(0.05)         # 6
        irUpValue = mcp.readAnalog(0, 0)         # 7
        irDownValue = mcp.readAnalog(1, 0)
        irLeftValue = mcp.readAnalog(0, 1)
        irRightValue = mcp.readAnalog(1, 1)
        ambientLight = 0
        gpio.write(ledPin, gpio.LOW)     # 8
        time.sleep(0.05)
        ambientLight = mcp.readAnalog(0, 0)      # 9
        irUpValue = irUpValue - ambientLight     # 10
        ambientLight = mcp.readAnalog(1, 0)      # 11
        irDownValue = irDownValue - ambientLight
        ambientLight = mcp.readAnalog(0, 1)
        irLeftValue = irLeftValue - ambientLight
        ambientLight = mcp.readAnalog(1, 1)
        irRightValue = irRightValue - ambientLight
        distance = (irUpValue+irDownValue+irLeftValue+irRightValue)/4 # 12
def main():
        global irUpValue,irDownValue,irLeftValue,irRightValue,distance
        while True:      # 13
                readCompoundEye()        # 14
                print "Values:"
                print "Up: %f" % irUpValue
                print "Down: %f" % irDownValue
                print "Left: %f" % irLeftValue
                print "Right: %f" % irRightValue
                print "Distance: %f" % distance
                time.sleep(0.5) # s      # 15
if __name__ == "__main__":
        main()
```

1 Importe `gpio` pour activer et désactiver les broches numériques (dans cet exemple, gpio25). Le fichier *botbook_gpio.py* qui figure dans les exemples de code doit se trouver dans le même répertoire que ce programme (*compound_eye.py*).
2 Importe la bibliothèque `mcp3002` pour lire les valeurs du capteur analogique avec la puce MCP3002 du convertisseur analogique-numérique. On l'utilise pour lire les valeurs de chaque transistor sensible à l'infrarouge. Le fichier de la bibliothèque *botbook_mcp3002.py* doit se trouver dans le même répertoire que *compound_eye.py*. Vous devez aussi installer la bibliothèque *spidev*, qui est importée par *botbook_mcp3002.py*. Lisez les commentaires au début du fichier *botbook_mcp3002/botbook_mcp3002.py* ou bien *Installation de SpiDev* un peu plus loin dans ce chapitre.
3 Déclare des variables globales.
4 Pour utiliser des variables globales à l'intérieur d'une fonction, elles doivent être listées au début de la fonction.
5 Active la broche gpio25 connectée aux LED infrarouges. Cela illumine la zone ciblée pour la mesure.
6 Attend que les broches se stabilisent.
7 Lit chacune des valeurs des transistors sensibles à l'infrarouge. Comme le Raspberry Pi n'a pas de conversion analogique-numérique intégrée, on utilise la puce externe MCP3002.
8 Pour supprimer l'effet de la lumière ambiante, on désactive la LED infrarouge.
9 et 11 La valeur de chaque transistor sensible à l'infrarouge est lue à nouveau.
10 La valeur de la lumière ambiante est supprimée de la mesure réelle (qui a été réalisée par l'illumination infrarouge).
12 La distance est la moyenne des mesures des quatre transistors.
13 Dans les applications embarquées, `while(True)` est une méthode courante pour accomplir la même action à l'infini. La plupart des appareils embarqués sont supposés fonctionner en permanence et ne sont pas censés quitter le programme et s'arrêter de marcher au bout d'un certain temps. Pour arrêter un programme s'exécutant dans une boucle `while(True)`, appuyez sur Ctrl+C dans la même session de terminal à partir de laquelle vous avez exécuté le programme.
14 La fonction `readSensor()` n'a pas besoin de retourner de valeur car elle modifie des variables globales. En Python, on peut aussi retourner plusieurs valeurs, comme dans l'exemple `a,b,c=foo()`.
15 Quand on exécute une boucle, un petit délai garantit que la boucle n'accapare pas 100% du temps CPU.

L'œil composé comporte lui-même quelques connexions, ce qui finit par faire un bon nombre de câbles avec le convertisseur analogique-numérique.

Installation de SpiDev

Le convertisseur analogique-numérique MCP3002 utilise le protocole SPI. SPI est un protocole assez complexe, mais vous pouvez installer la bibliothèque SpiDev pour gérer les détails.

La bibliothèque SpiDev est nécessaire pour tout le code qui utilise `import spidev`. Cela comprend notamment le code du potentiomètre pour Raspberry Pi, et tous les capteurs de résistance analogique utilisés dans cet ouvrage car SpiDev est importée par *botbook_mcp3002*.

Sur votre Raspberry Pi, ouvrez un terminal. Commencez par installer les prérequis nécessaires :

```
$ sudo apt-get update
$ sudo apt-get -y install git python-dev
```

Téléchargez la dernière version de SpiDev à partir du site de développement :

```
$ git clone https://github.com/doceme/py-spidev.git
$ cd py-spidev/
```

Puis installez-la sur votre système :

```
$ sudo python setup.py install
```

Vous devez ensuite activer le module SPI sur le Raspberry Pi. Assurez-vous d'abord qu'il n'est pas désactivé. Modifiez */etc/modprobe.d/raspi-blacklist.conf* avec la commande sudoedit /etc/modprobe.d/raspi-blacklist.conf et supprimez cette ligne :

```
blacklist spi-bcm2708
```

Enregistrez le fichier : appuyez sur Ctrl+X, saisissez y, puis appuyez sur Entrée.

Pour autoriser l'accès à SPI sans les privilèges de root, copiez le fichier *udev* de l'Exemple 3.9 (ou de l'exemple de code du répertoire *botbook_mcp3002*) :

```
$ sudo cp 99-spi.rules /etc/udev/rules.d/99-spi.rules
```

Exemple 3.9. 99-spi.rules

```
# /etc/udev/rules.d/99-spi.rules - SPI sans root sur Raspberry Pi
# Copyright 2013 http://BotBook.com
SUBSYSTEM=="spidev", MODE="0666"
```

Redémarrez votre Raspberry Pi, ouvrez LXTerminal, vérifiez que vous pouvez voir les périphériques SPI et que l'appartenance est correcte :

```
$ ls -l /dev/spi*
```

Le listing doit afficher deux fichiers qui doivent avoir les permissions crw-rw-rwT. Si ce n'est pas le cas, recommencez les étapes précédentes.

Vous pouvez maintenant utiliser la puce MCP3002 et les autres appareils SPI avec le Pi.

Circuits alternatifs pour Raspberry Pi

Le circuit Raspberry Pi de l'œil composé est assez complexe. Même s'il n'est pas difficile à comprendre, il comporte de nombreux câbles à connecter. Pour créer un système plus simple, vous pouvez utiliser un autre convertisseur analogique-numérique ou un Arduino (voir *Pi + Arduino* ci-après).

Pour ne gérer qu'un seul convertisseur, vous pouvez employer la puce MCP3008 qui comporte huit entrées. Cela nécessite cependant la modification de la bibliothèque botbook_mcp3002. Vous pouvez aussi étudier le code du MCP3008 de chez Adafruit (https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code).

Pi + Arduino

Si vous trouvez le nombre de câbles excessif, vous pouvez aussi lire le capteur à partir d'un Arduino, puis envoyer les résultats au Raspberry Pi via une connexion USB-série. Une autre option consiste à utiliser le PI a la Mode (http://bit.ly/1icPd0z), une carte compatible Arduino qui se greffe au-dessus d'un Raspberry Pi pour créer un hybride parfait.

Voir *Communiquer avec Arduino à partir d'un Raspberry Pi* au chapitre 11 pour de plus amples informations.

PROJET : ALARME POSTURALE (ARDUINO)

Tout geek connaît bien ce problème : plus on travaille longtemps, plus la tête se rapproche dangereusement de l'écran, ce qui n'est guère conforme aux prescriptions de Dame Nature. En combinant un capteur de distance infrarouge et un buzzer piézo, on peut facilement créer un gadget qui avertit quand on est trop proche de l'écran (Figure 3.18).

Figure 3.18 Alarme posturale

Ce que vous allez apprendre

Dans ce projet d'alarme posturale, vous allez apprendre à :

- Combiner une entrée, un traitement et une sortie.
- Jouer des notes avec un buzzer piézo.
- Inclure votre projet dans un boîtier.

Buzzer piézo

Un cristal piézoélectrique change de forme quand on lui applique une tension. En utilisant du courant alternatif ou une simple onde carrée, on peut faire vibrer le cristal piézo. Cela fait vibrer l'air, ce qui produit un son, en l'occurrence un son désagréable.

Figure 3.19 Buzzer piézo

Le buzzer piézo le plus courant émet un bip quand on lui envoie une onde carrée (Figure 3.19). Une onde carrée varie entre HIGH (5 V pour Arduino) et LOW (0 V).

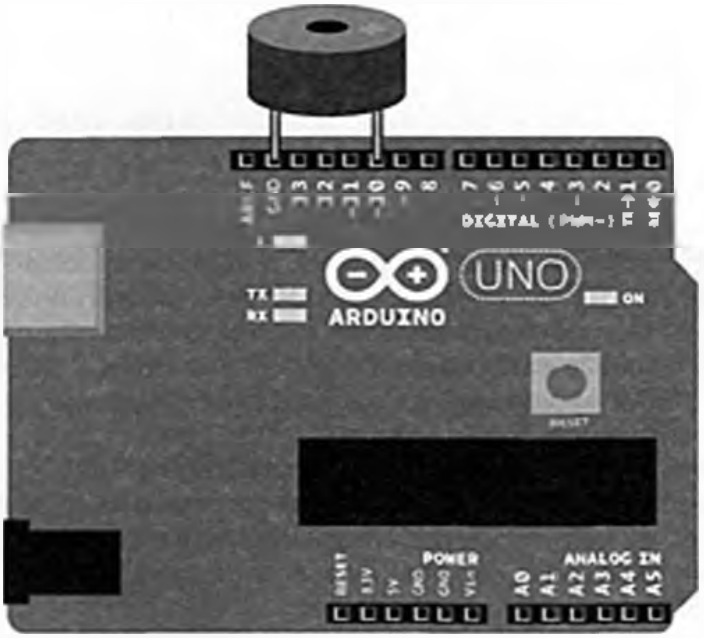


Figure 3.20 Buzzer piézo connecté

Vous pouvez facilement créer une onde carrée en activant et en désactivant de manière répétée une broche de données avec `digitalWrite`. Vous pouvez aussi utiliser la fonction intégrée `tone()` qui implémente un processus plus complexe pour produire la même onde.

> Le phénomène piézoélectrique fonctionne aussi d'une autre manière : vous pouvez générer de l'électricité en pressant un cristal piézoélectrique. Les allume-gaz électriques utilisent souvent l'effet piézoélectrique pour créer une étincelle.

Exemple 3.10. piezo_beep.ino

```
// piezo_beep.ino - sonne à une fréquence donnée avec un buzzer piézo
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int speakerPin = 10;
void wave(int pin, float frequency, int duration)   // 1
{
  float period=1/frequency*1000*1000; // microsecondes (µs) // 2
  long int startTime=millis();  // 3
  while(millis()-startTime < duration) {   // 4
    digitalWrite(pin, HIGH);    // 5
    delayMicroseconds(period/2);
    digitalWrite(pin, LOW);
    delayMicroseconds(period/2);
  }
}
void setup()
{
 pinMode(speakerPin, OUTPUT);
}
void loop()
```

```
{
  wave(speakerPin, 440, 500);    // 6
  delay(500);
}
```

1 Pour utiliser la fonction wave dans vos projets, il suffit de spécifier la broche, la fréquence de la note et sa durée en millisecondes. Le contenu de la fonction ne concerne que des détails d'implémentation : c'est bien pédagogiquement, mais pas nécessaire pour utiliser la fonction.
2 Calcule la période T : quelle est la durée d'une onde (un HIGH + un LOW) ? C'est l'inverse de la fréquence f : T = 1/f. Par exemple, si vous avez 2 ondes d'une seconde (2 hertz, c'est-à-dire 2 * 1/s), une onde durera une demi-seconde (1 / 2 Hz, ou 1/(2 * 1/s), ce qui donne 1/2 s). Notez que hertz (nombre de cycles par seconde) est abrégé en Hz.
3 C'est un modèle courant utilisé en programmation pour accomplir quelque chose pendant une durée déterminée : on commence par enregistrer l'heure de départ dans une variable (millis fournit le nombre de millisecondes écoulées depuis le démarrage de l'Arduino)...
4 ...et on attend jusqu'à ce que la durée spécifiée soit écoulée.
5 Crée une onde complète. On crée d'abord la partie HIGH, puis la partie LOW.
6 Cet appel suffit à créer un bip dans le programme principal.

Alarme, alarme !

On peut à présent passer d'un simple bip à un nouveau monde de notes : l'alarme.

Exemple 3.11. piezo_alarmtone.ino

```
// piezo_alarmtone.ino - utilise un buzzer piézo pour jouer une alarme
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int speakerPin = 10;
void wave(int pin, float frequency, int duration)       // 1
{
  float period=1 / frequency * 1000 * 1000; // microsecondes (µs)
  long int startTime=millis();
  while(millis()-startTime < duration) {
    digitalWrite(pin, HIGH);
    delayMicroseconds(period/2);
    digitalWrite(pin, LOW);
    delayMicroseconds(period/2);
  }
}
void setup()
{
 pinMode(speakerPin, OUTPUT);
}
void loop()
{
  wave(speakerPin, 440, 40);     // 2
  delay(25);
  wave(speakerPin, 300, 20);
  wave(speakerPin, 540, 40);
  delay(25);
```

```
    wave(speakerPin, 440, 20);
    wave(speakerPin, 660, 40);
    delay(25);
    wave(speakerPin, 540, 20);
}
```

1 Utilise la même fonction wave() que l'on a employée pour générer un simple bip.
2 Joue une note, attend pendant un temps très court, joue une autre note, et ainsi de suite.

Combinaison d'un buzzer avec un capteur infrarouge

Montez le circuit d'alarme posturale (Figure 3.21). Chargez le code et préparez-vous à améliorer votre posture face à l'écran, sinon vous entendrez une alarme sans fin...

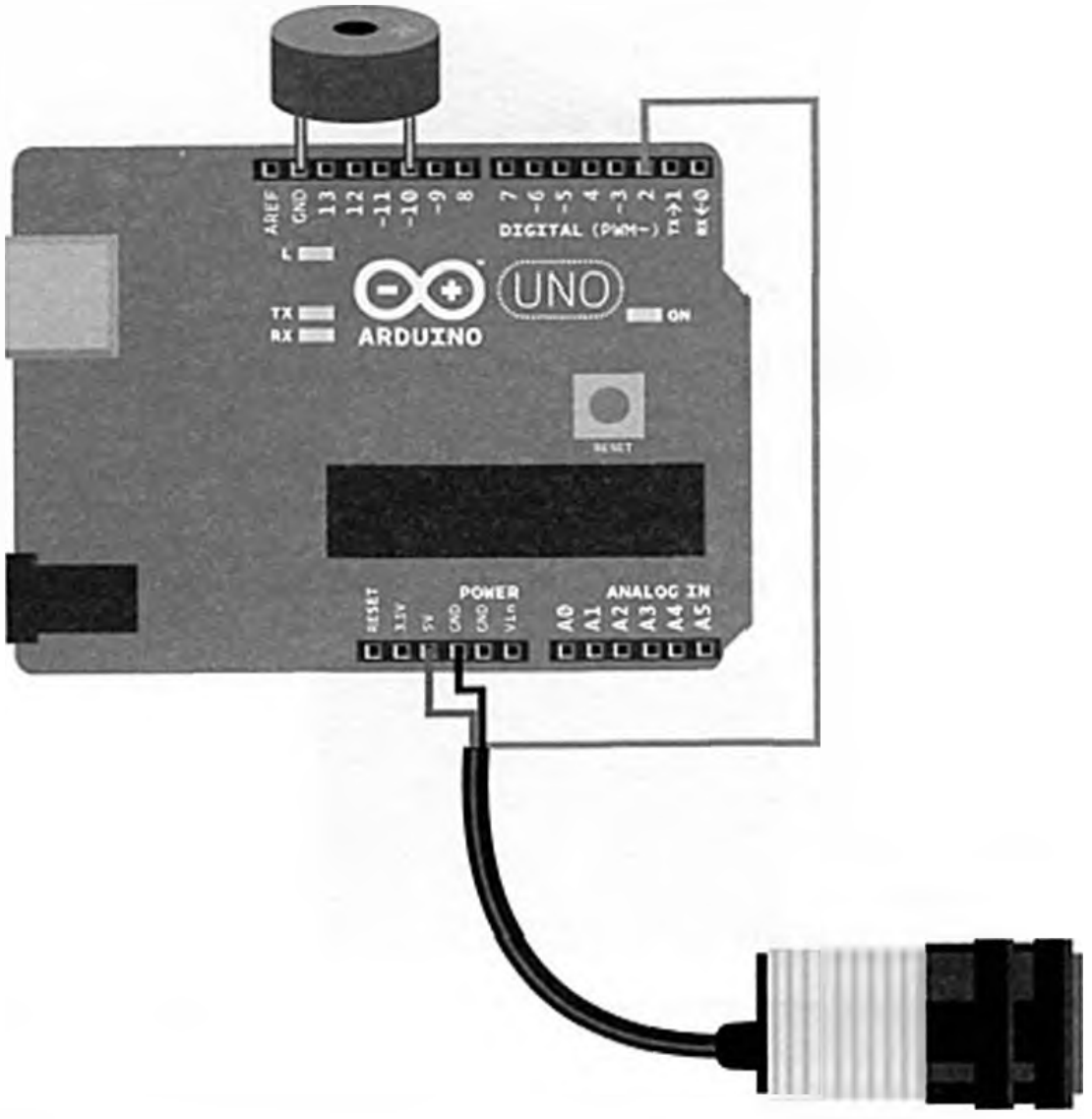


Figure 3.21 Schéma de l'alarme posturale

Exemple 3.12. posture_alarm.ino

```
// posture_alarm.ino - sonne l'alarme quand un capteur infrarouge détecte
une mauvaise posture
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int speakerPin = 10;
const int sensorPin = 2;
int switchState = 0;
void wave(int pin, float frequency, int duration)       // 1
{
  float period=1/frequency*1000*1000; // microsecondes (µs)
  long int startTime=millis();
  while(millis()-startTime < duration) {
    digitalWrite(pin, HIGH);
    delayMicroseconds(period/2);
    digitalWrite(pin, LOW);
    delayMicroseconds(period/2);
  }
}
void alarm() // 2
{
  wave(speakerPin, 440, 40);
  delay(25);
  wave(speakerPin, 300, 20);
  wave(speakerPin, 540, 40);
  delay(25);
  wave(speakerPin, 440, 20);
  wave(speakerPin, 640, 40);
  delay(25);
  wave(speakerPin, 540, 20);
}
void setup()
{
  pinMode(speakerPin, OUTPUT);
  Serial.begin(115200);
  pinMode(sensorPin, INPUT);
}
void loop()
{
  switchState = digitalRead(sensorPin);
  Serial.println(switchState,BIN);
  if (switchState==0) { // 3
    alarm(); // 4
  }
  delay(10);
}
```

1 Utilise la même fonction wave() déjà employée précédemment pour le bip et l'alarme.
2 Utilise la même fonction alarm() que précédemment.

3 Utilise la fonction digitalRead() pour obtenir une valeur du détecteur de distance infrarouge, comme cela a été fait auparavant.
4 Si quelque chose est détecté dans la zone de l'écran, on sonne l'alarme. En pratique, si vous placez votre tête trop près de l'écran, l'alarme va sonner.

On place le tout dans un joli boîtier

Les prototypes sont plus impressionnants et plus robustes quand on les met à l'intérieur d'un boîtier.

Nous avons utilisé un boîtier pour Arduino fabriqué par SmartProjects, car il convenait exactement à notre projet.

Nous l'avons d'abord peint en noir puis nous avons fait un trou de 19 mm pour laisser passer le capteur infrarouge (Figure 3.22).

Figure 3.22 Trou pour le capteur infrarouge

Nous avons supprimé la petite trappe à l'arrière du boîtier pour que le capteur tienne plus facilement et pour pouvoir ajuster la vis de réglage de la distance ultérieurement (Figure 3.23).

Nous avons fixé le capteur avec les écrous en plastique qui sont livrés. Vous pouvez glisser l'Arduino dans les tétons du châssis où il sera correctement fixé (Figure 3.24).

Figure 3.23 Depuis l'arrière du châssis, vous pouvez ajuster la vis de réglage de la distance

Fermez le boîtier en poussant l'une sur l'autre les deux moitiés et vous êtes prêt à protéger la posture de votre dos (voir la Figure 3.18). Le boîtier a un trou pour le passage du câble USB et vous disposez à présent d'un joli gadget USB pour décorer votre espace de travail.

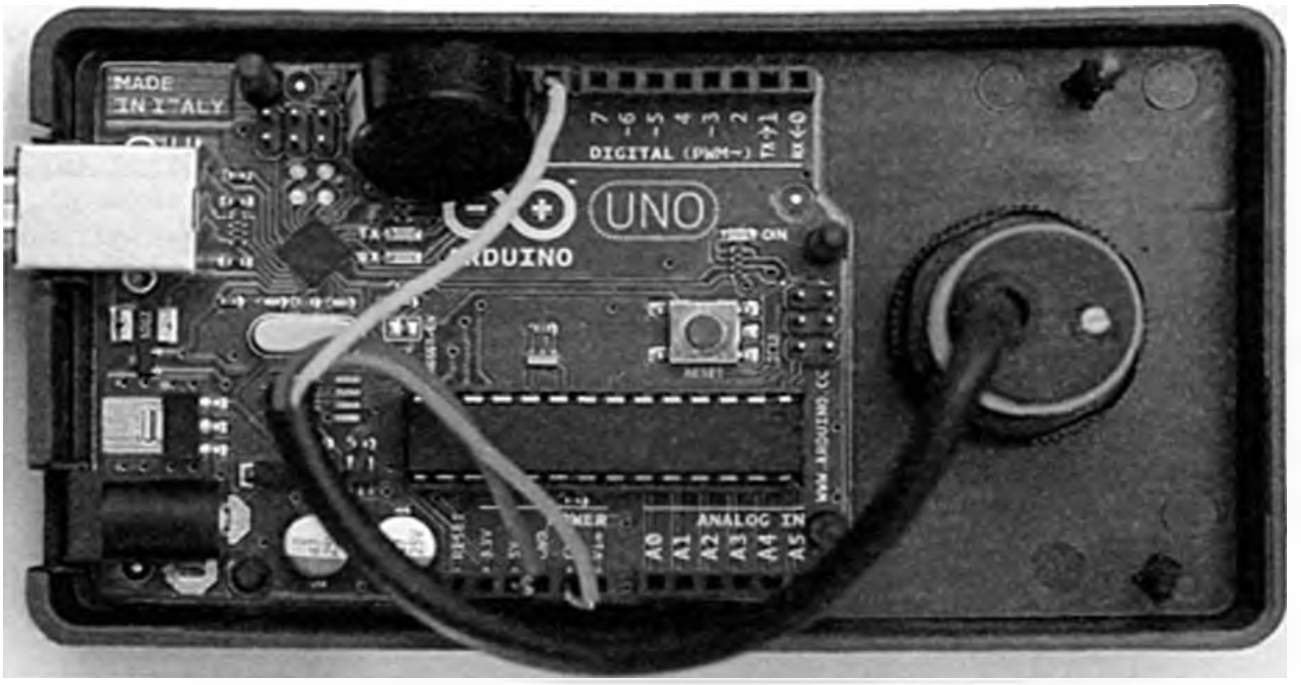


Figure 3.24 **Agencement du boîtier d'alarme posturale**

Vous avez appris à mesurer la distance selon différentes méthodes. Vos projets sont maintenant capables de déterminer si quelque chose est proche ou loin des objets qui l'entourent. Avec plusieurs capteurs, vous pouvez gérer des comportements sophistiqués. Par exemple, en combinant deux capteurs infrarouges avec un servo, vous pouvez faire tourner un robot dans la direction de l'objet le plus proche qu'il a détecté. En plaçant deux récepteurs infrarouges sur un robot mobile, vous pouvez lui faire suivre un rayon lumineux. Comment allez-vous utiliser les capteurs de distance dans vos futurs projets ?

4 Détecter fumées et gaz

Quand vous commencez à bailler au travail ou bien dans un cours passionnant, il se peut que le coupable soit le dioxyde de carbone (CO_2). Les capteurs des systèmes de ventilation des habitations peuvent détecter un niveau élevé de CO_2 et vous envoyer l'air frais dont vous avez besoin.

Les pompiers peuvent mesurer s'il y a une vapeur d'hydrocarbure dans l'air afin d'éviter qu'une explosion ne survienne. Il y a aussi un capteur de gaz à l'intérieur du moteur de votre voiture qui mesure le ratio essence/air. Un ratio correct garantit que toute l'essence brûle dans le cylindre (ce ne serait pas bon si de l'essence sortait du tuyau d'échappement).

> Il est facile et amusant de créer des prototypes avec des capteurs de gaz et de fumée bon marché. Il existe cependant des exigences strictes, des protocoles de test et des programmes de certification pour les produits de sécurité si bien que les projets de ce chapitre ne remplaceront pas de tels produits.

MQ est une série de capteurs de gaz bon marché. Il existe des capteurs pour de nombreux gaz dont certains sont listés dans le Tableau 4.1.

Capteur MQ	Gaz détectés
MQ-2	Gaz et fumée inflammable
MQ-3, MQ-303A	Alcool (éthanol)
MQ-4	Méthane (CH4)
MQ-7	Monoxyde de carbone
MQ-8	Hydrogène
MQ-9	Monoxyde de carbone, méthane, GPL (propane ou butane)

Tableau 4.1 Exemples de capteurs de gaz MQ

EXPÉRIENCE : DÉTECTEUR DE FUMÉE (CAPTEUR ANALOGIQUE DE GAZ)

Un capteur de fumée MQ-2 signale la fumée par le niveau de tension qu'il produit. Plus il y a de fumée, plus la tension est élevée. Le MQ-2 que nous avons utilisé a un potentiomètre intégré pour ajuster la sensibilité (Figure 4.1).

Figure 4.1 Capteur analogique de gaz

> Quel est le gaz le plus mortel ? Si l'on mesure le danger par le nombre de victimes, il s'agit du monoxyde de carbone (CO). Dans un incendie, la plupart des victimes meurent en respirant de la fumée avant que les flammes ne les atteignent.
>
> Le CO mortel est différent du dioxyde de carbone, le CO_2, qui est la plupart du temps inoffensif. Le corps humain crée du CO_2 quand il produit de l'énergie, et le sang transporte le CO_2 aux poumons où il est expiré.
>
> Le CO est un poison car il ne veut pas partir. Il se fixe à l'hémoglobine, empêchant ainsi le sang de transporter de l'oxygène et du CO_2 pendant des heures. Si toute l'hémoglobine est bloquée par le CO, les tissus ne sont plus capables d'obtenir de l'oxygène, et la mort s'ensuit.

Comme le MQ-2 a trois fils, il n'a pas besoin de résistance pull-up ou pull-down. L'Arduino considère le MQ-2 comme un potentiomètre dans une configuration à trois fils.

Le MQ-2 est relié à la masse (noir, 0 V) et au +5 V (rouge). Il mesure la fumée et augmente la tension de sa broche S (la plus proche du +5 V) quand il détecte de la fumée.

MQ-2 pour Arduino

Arduino possède un convertisseur analogique-numérique intégré, si bien que vous pouvez lire le MQ-2 avec un appel à analogRead(). Utilisez le potentiomètre sur la platine d'évaluation pour ajuster sa sensibilité.

Exemple 4.1. mq_x_smoke_sensor.ino

```
// mq_x_smoke_sensor.ino - affiche le niveau de fumée sur le port série
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int sensorPin = A0;
int smoke_level = -1;   // 1
void setup() {
  Serial.begin(115200); // bit/s
  pinMode(sensorPin, INPUT);
}
void loop() {
  smoke_level = analogRead(sensorPin);  // 2
  Serial.println(smoke_level);
  if(smoke_level > 120) {       // 3
    Serial.println("Fumée détectée");
  }
  delay(100); // ms
}
```

1 Initialise le niveau de fumée à une valeur impossible pour aider au débogage. Si vous voyez la valeur -1 sur la sortie, vous savez qu'elle n'a jamais été remplacée par une valeur lue par analogRead().
2 Le MQ-2 est un capteur simple de résistance analogique et vous pouvez donc lire la valeur avec analogRead(). La valeur retournée est comprise entre 0 (0 V) et 1023 (+5 V).
3 Même s'il y a une valeur de seuil dans le code, il est préférable d'ajuster la sensibilité avec le potentiomètre du capteur.

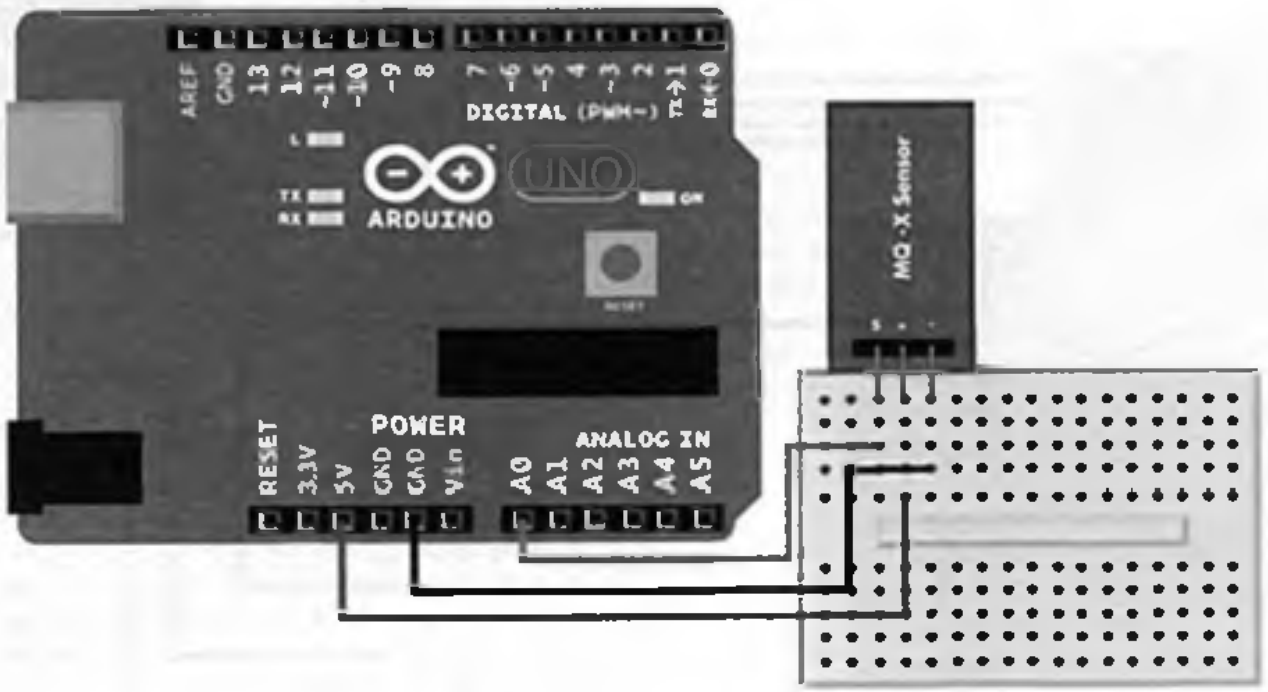


Figure 4.2 Circuit du capteur MQ-2 pour Arduino

Une alarme incendie n'a pas à être un boîtier passe-partout, comme le prouve l'alarme Lento conçue par Paola Suhonen (Figure 4.3). Une forme d'insecte donne à cet objet du quotidien un aspect totalement nouveau. Gardez cela en tête quand vous concevez vos propres gadgets.

Figure 4.3 Alarme Lento conçue par Paola Suhonen

MQ-2 pour Raspberry Pi

Le Raspberry Pi a besoin d'un convertisseur analogique-numérique externe pour lire le MQ-2. Pour cette conversion, vous pouvez utiliser une puce MCP3002 bon marché qui est similaire à ce que nous avons déjà employé (voir *Œil composé pour Raspberry Pi* au chapitre 3).

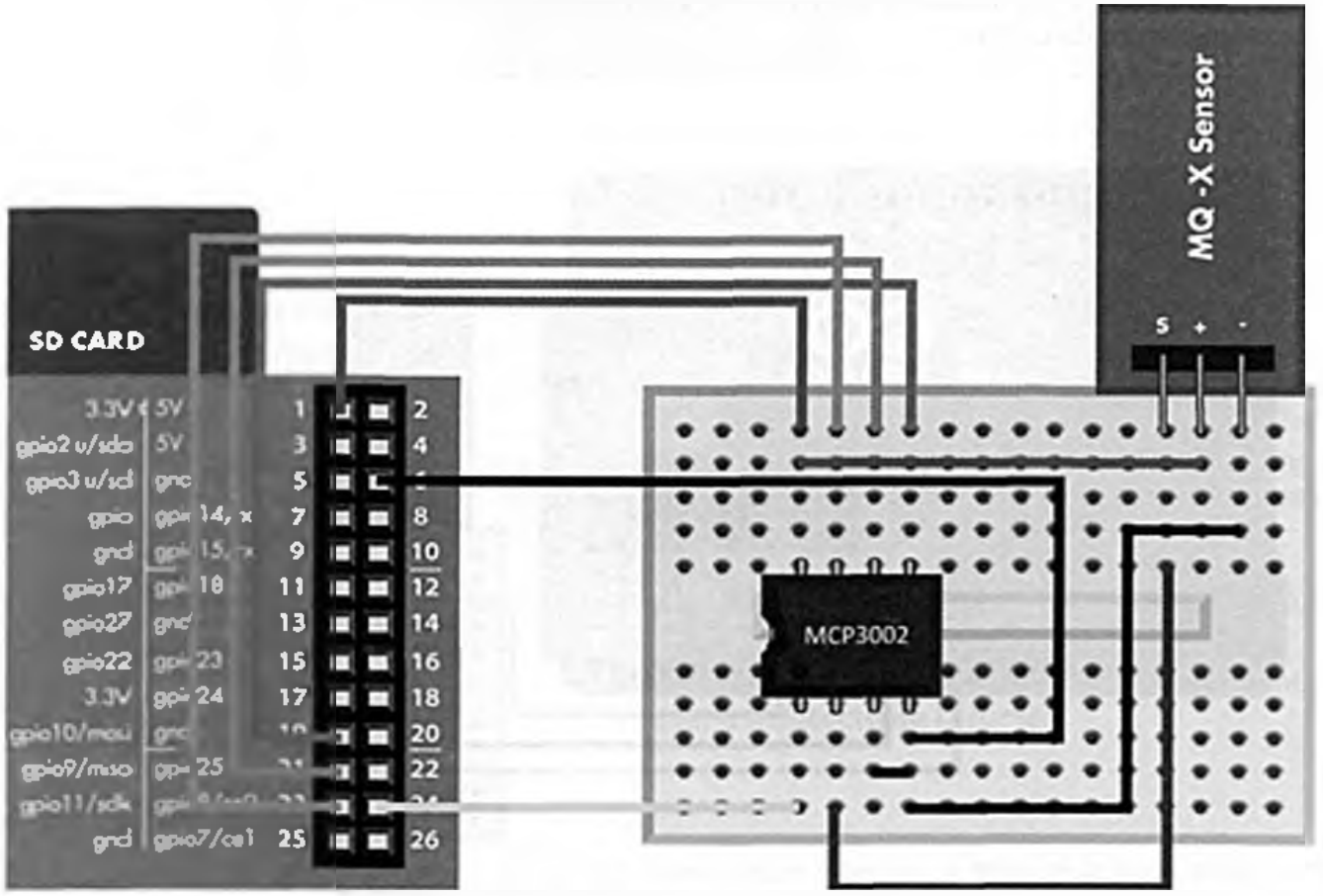


Figure 4.4 Circuit du capteur MQ-2 pour Raspberry Pi

Exemplo 4.2. mq_x_smoke_sensor.py

```
# mq_x_smoke_sensor.py - affiche le niveau de fumée
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp   # 1
smokeLevel = 0
def readSmokeLevel():
        global smokeLevel
        smokeLevel = mcp.readAnalog()   # 2
def main():
  while True:   # 3
    readSmokeLevel()    # 4
    print("Le niveau actuel de fumée est %i " % smokeLevel)    # 5
    if smokeLevel > 120:
      print("Fumée détectée")
```

```
    time.sleep(0.5) # 3
if __name__ == '__main__':
  main()
```

1 La bibliothèque botbook.com pour le MCP3002 économise de nombreuses lignes de code et facilite la lecture des valeurs analogiques. La bibliothèque (*botbook_mcp3002.py*) doit se trouver dans le même répertoire que *mq_x_smoke_sensor.py*. Vous devez d'abord installer la bibliothèque *spidev*, qui est importée par *botbook_mcp3002*. Pour de plus amples informations, lisez les commentaires au début du fichier *botbook_mcp3002/botbook_mcp3002.py* ou bien la section *Installation de SpiDev* au chapitre 3.
2 Lit la tension du premier appareil connecté au convertisseur MCP3002. Les paramètres de readAnalog(device=0, channel=0) sont les valeurs par défaut si bien que vous pouvez les ignorer.
3 La répétition d'un programme à l'infini est courante pour les périphériques embarqués. Quand on utilise while True, on doit toujours ajouter un certain délai à la fin de la boucle. Vous pouvez arrêter le programme par un Ctrl+C quand vous avez terminé.
4 Il est pratique d'insérer la tâche principale du programme à l'intérieur de votre propre fonction. Comme son nom l'indique, readSmokeLevel() lit le niveau de fumée ; on peut utiliser ce module comme une brique de construction quand on crée des projets plus importants impliquant plusieurs capteurs et sorties.
5 Crée la chaîne à afficher avec une chaîne de mise en forme. La valeur de smokeLevel est traitée comme un entier et remplace %i dans la chaîne de mise en forme.

Expérience : la fumée monte

Il y a une bonne raison pour que les alarmes incendie soient montées au plafond plutôt que sur le plancher. La fumée provient en général du feu si bien qu'elle est plus chaude que l'air ambiant. La chaleur rend les gaz moins denses, c'est-à-dire, plus légers.

Pourquoi l'air chaud est-il plus léger que l'air froid ? La chaleur implique le mouvement : les molécules sont secouées et rebondissent les unes sur les autres. Plus elles deviennent chaudes, plus les molécules s'activent. Quand elles s'entrechoquent, elles se repoussent les unes les autres, ce qui rend le gaz moins dense. En ayant moins de molécules dans un même volume de gaz, le gaz devient plus léger. Et quand il est entouré par de l'air plus lourd, le gaz plus léger monte.

Figure 4.5 La fumée monte

Utilisez le programme du capteur de fumée et éteignez une allumette à côté du capteur. Jusqu'à quelle hauteur pouvez-vous maintenir l'allumette éteinte et obtenir encore une mesure ? Quand vous atteindrez le même niveau que le capteur, vous ne verrez probablement plus de différence avec l'air normal. Comme le titre de cette expérience l'indique, la fumée monte. C'est la raison pour laquelle il n'y a rien à mesurer si vous laissez la fumée s'échapper bien au-dessus du capteur.

Expérience : éthylotest (capteur d'alcool MQ-303A)

Un éthylotest est utilisé pour contrôler si une personne a de l'alcool dans le sang, et plus spécifiquement pour vérifier la présence d'éthanol, qui est l'alcool que l'on trouve dans le vin, la bière et les boissons alcoolisées.

Le capteur d'alcool MQ-3 ressemble beaucoup aux autres capteurs de gaz de la série MQ (Figure 4.6).

Figure 4.6 Capteur d'alcool MQ-3

> Avant de faire une pause et de vous verser un verre de Château Margaux, jetez au moins un coup d'œil à l'expérience *Testez sans vous alcooliser* un peu plus loin dans ce chapitre.

Tout comme l'échange de gaz dans les poumons vous apporte de l'oxygène et se débarrasse du dioxyde de carbone, de l'alcool est libéré dans l'air quand vous expirez. C'est cet alcool qui est mesuré par un éthylomètre.

Plus il y a d'éthanol dans votre sang, plus il y en a dans l'air que vous expirez. L'alcoolémie donne une bonne indication du degré d'ivresse d'une personne.

Plus une personne a de l'alcool dans le sang, plus elle est ivre, mais l'ivresse varie cependant d'une personne à l'autre. Pour établir la réglementation, une valeur moyenne est choisie comme limite. Par exemple, en France, la limite légale pour conduire un véhicule est de 0,5 gramme d'alcool par litre de sang (ou 0,25 mg d'alcool par litre d'air expiré).

Les éthylomètres officiels sont calibrés périodiquement pour obtenir des mesures fiables. Les éthylomètres utilisent une formule intégrée pour estimer l'alcoolémie à partir de l'alcool contenu dans l'air expiré.

La Figure 4.7 illustre le circuit de l'Arduino pour le MQ-3, et le sketch Arduino est listé dans l'Exemple 4.3. Vous trouverez le circuit correspondant pour le Raspberry Pi à la Figure 4.8 et le code Python dans l'Exemple 4.4.

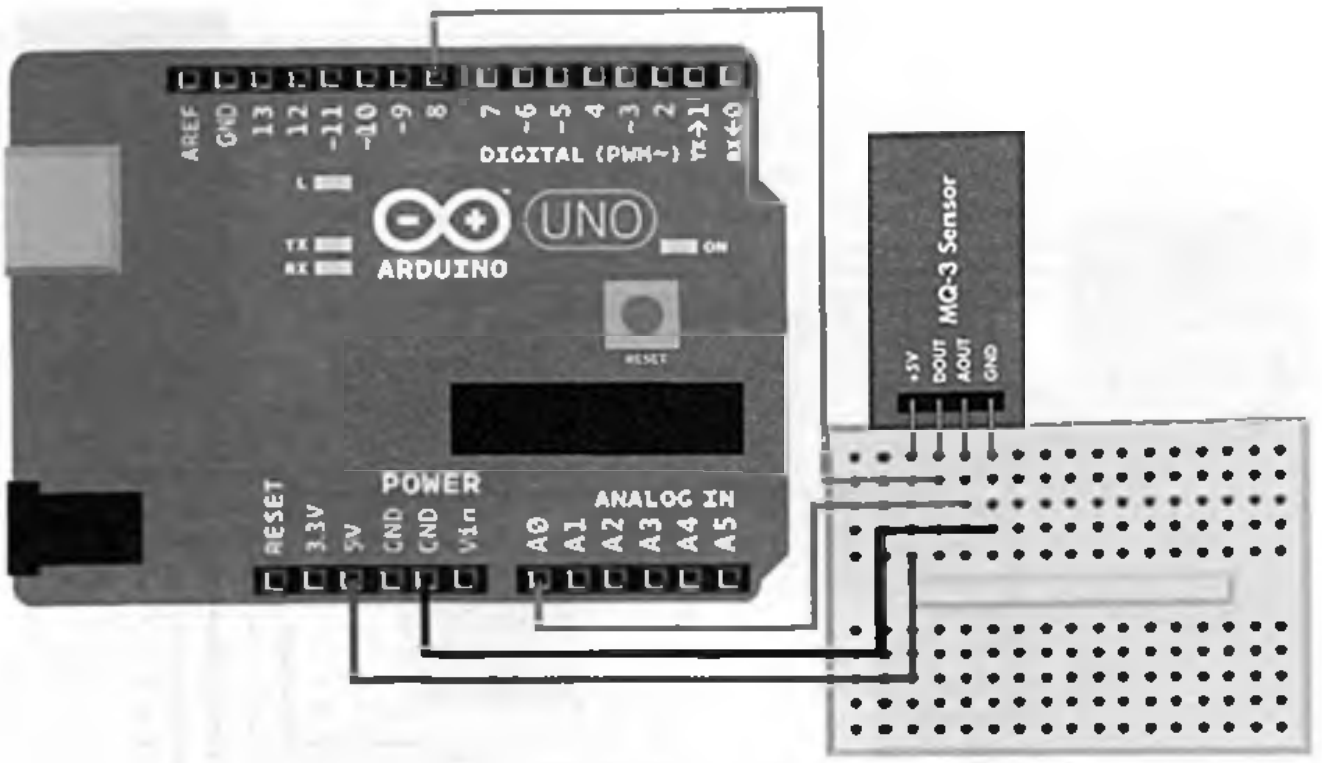


Figure 4.7 Circuit du capteur MQ-3 pour Arduino

Exemple 4.3. mq_3_alcohol_sensor.ino

```
// mq_3_alcohol_sensor.ino - affiche la valeur d'alcool et la limite
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int analogPin = A0;
const int digitalPin = 8;
int limit = -1;
int value = 0;
void setup() {
  Serial.begin(115200);
  pinMode(digitalPin,INPUT);
}
void loop()
{
  //Lit la valeur analogique
  value = analogRead(analogPin);
  //Vérifie si la limite est dépassée
  limit = digitalRead(digitalPin);
  Serial.print("Valeur d'alcool : ");
  Serial.print(value);
  Serial.print("Limite : ");
  Serial.println(limit);
  delay(100);
}
```

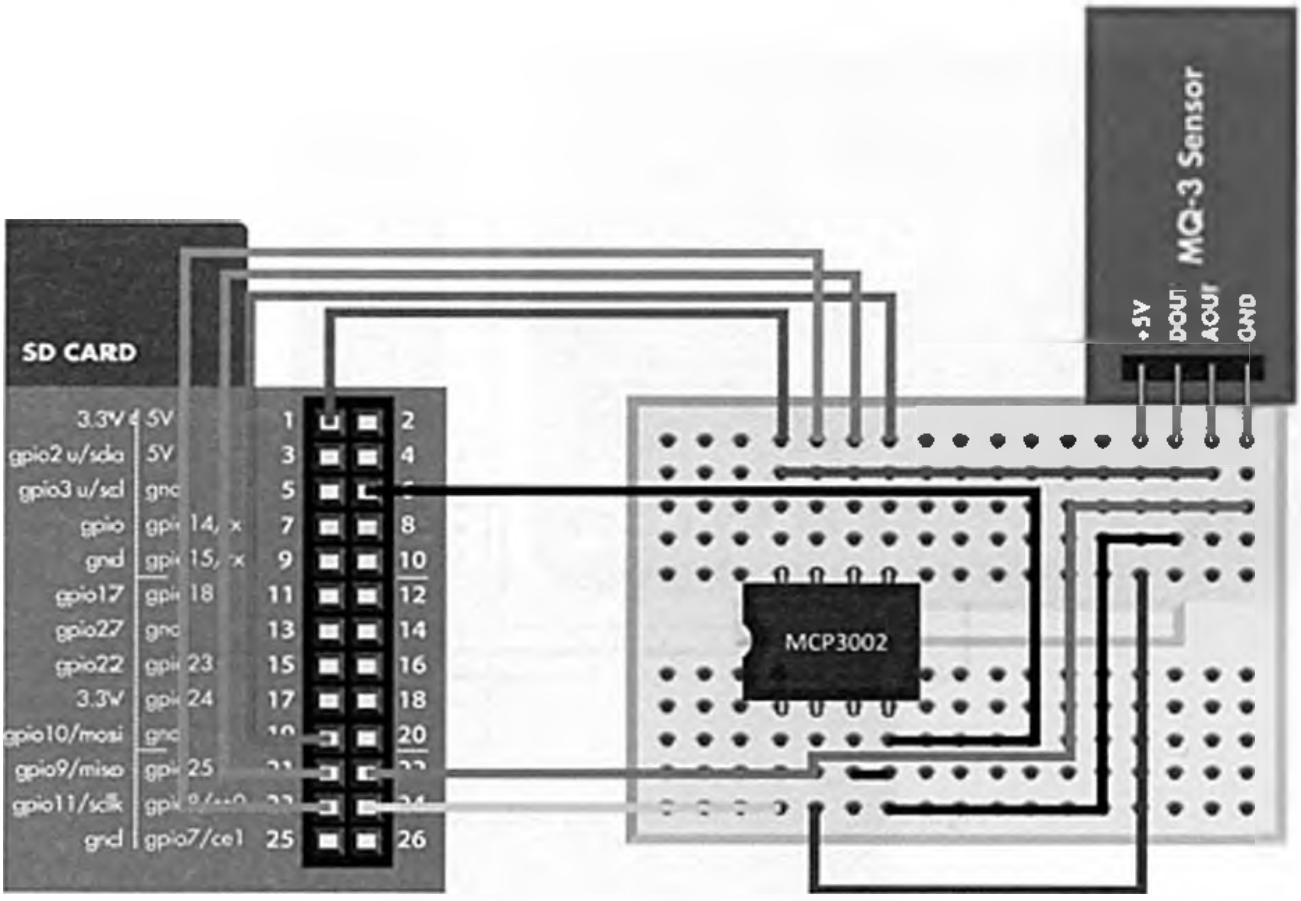


Figure 4.8 Circuit du capteur MQ-3 pour Raspberry Pi

Exemple 4.4. mq_3_alcohol_sensor.py

```
# mq_3_alcohol_sensor.py - lit la sortie numérique à partir du capteur
d'alcool
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio
def readLimit():
  limitPin = 23
  gpio.setMode(limitPin,"in")
  return gpio.read(limitPin)
def main():
  while True:
    if readLimit() == gpio.HIGH:
      print("Limite dépassée :")
    time.sleep(0.5)
if __name__ == "__main__":
  main()
```

Expérience : testez sans vous alcooliser

Boire de l'alcool quand on essaye d'apprendre l'électronique peut se révéler contre-productif. La bonne nouvelle est que vous n'avez pas besoin d'une bouteille de pur malt pour tester votre capteur. Vous pouvez obtenir des résultats impressionnants en utilisant d'autres produits qui contiennent de l'alcool. Les solutions pour bain de bouche ou les bonbons à la liqueur fonctionnent parfaitement.

Utilisez le code de l'éthylotest et ouvrez le Moniteur série. Prenez un bain de bouche (sans l'avaler) et soufflez dans le capteur. La mesure doit instantanément monter en flèche. Vous risquez d'obtenir des niveaux supérieurs à la normale plusieurs minutes après votre bain de bouche.

PROJET : ENVOI D'UN COURRIEL EN CAS DE FUMÉE

Que diriez-vous de recevoir un courriel quand votre gadget détecte de la fumée ?

Figure 4.9 Alarme prête à envoyer un courriel en cas de fumée

Ce que vous allez apprendre

Dans ce projet, vous allez apprendre :

- À réagir aux modifications de l'environnement : s'il y a de la fumée, on envoie un courriel d'avertissement.
- À envoyer automatiquement un courriel à partir du Raspberry Pi.
- Les bases de l'envoi de courriels.

Python pour le courriel et les réseaux sociaux

C'est un exemple de projet qui est facile avec le Raspberry Pi. L'envoi automatique de courriels à partir d'un Raspberry Pi n'est pas différent de l'envoi automatique à partir d'un système Linux.

Python est réputé pour avoir des « munitions » en ce sens où il existe une bibliothèque pour chaque chose. L'envoi de courriels est facile avec l'aide des bibliothèques existantes. De la même manière, il existe des bibliothèques pour l'envoi et la réception de données par le Web à l'aide du protocole HTTP.

Qu'en est-il des réseaux sociaux ? Vous pouvez adapter le programme pour envoyer des messages sur Twitter, Facebook, ou tous les services similaires de réseau social. Mais tous ces services utilisent des protocoles qui, bien qu'ils soient basés sur des technologies ouvertes, sont propriétaires et par conséquent verrouillés. Ces mastodontes ont des règles d'exécution de leur service, règles qu'ils peuvent changer du jour au lendemain de manière unilatérale. Par exemple, Twitter peut modifier le nombre de requêtes autorisées par jour, ou bien supprimer la fonctionnalité dont votre application dépend. Ne construisez jamais votre château de sable sur la plage privée de quelqu'un d'autre ! Ou faites-le au moins en toute connaissance de cause.

Montage

Connectez le capteur de fumée MQ-2 au Raspberry Pi (voir *MQ-2 pour Raspberry Pi* plus haut dans ce chapitre). C'est une bonne idée de tester d'abord le capteur de fumée avec seulement le code de l'Exemple 4.2.

Une fois que vous êtes certain qu'il fonctionne, vous pouvez passer à l'envoi de courriels dans votre code. Testez votre compte de courrier électronique normalement avant de l'utiliser dans ce programme. Par exemple, vous pouvez le saisir dans un client de messagerie comme Thunderbird, KMail, Claws ou, dans le pire des cas, Outlook. Même si notre exemple utilise un compte Gmail, n'importe quel compte fera l'affaire.

> N'utilisez pas votre compte personnel de courriel électronique pour ce type de test. Si vous faites une erreur dans votre code et que vous envoyez trop de courriels, votre serveur de messagerie peut interpréter cela comme une tentative de spam. Il est préférable de créer un compte de courrier électronique uniquement pour cette fonction. Pour autant, des services comme SendGrid (http://sendgrid.com/) et Amazon SES (http://aws.amazon.com/ses/) sont conçus exactement pour ce type de scénario, et vous pouvez donc envisager leur utilisation si vous comptez envoyer beaucoup de messages.

Comment fonctionne le courrier électronique ?

Est-ce que vous vous souvenez encore comment fonctionne le courrier électronique ? À l'heure de Gmail et des autres clients webmail, il est si facile d'oublier les bases.

Nous allons décrire ici le fonctionnement d'un courriel du point de vue d'un logiciel de messagerie s'exécutant sur votre ordinateur local, et non pas un webmail. Pour faire simple, nous allons nous contenter de montrer les points importants en prenant un exemple.

Il est essentiel de se rappeler que l'envoi et la réception de courriels impliquent deux serveurs différents (le serveur d'envoi et le serveur de réception). Il est courant que ces serveurs soient situés sur des réseaux différents, voire sur des continents différents.

Quand on clique sur le bouton Envoyer de son logiciel de messagerie, l'ordinateur contacte le serveur SMTP. Le serveur SMTP peut se trouver sur le réseau local, s'exécuter chez votre FAI (comme `smtp.orange.fr` ou `smtp.free.fr`) et peut accepter toutes les connexions en provenance de son réseau sans mot de passe ni identifiant. Il peut aussi être exécuté par un service

de messagerie (comme `smtp.gmail.com`, `mail.gmx.com`), et nécessite un cryptage TLS/SSL et un identifiant. Quand votre courriel est accepté, le serveur SMTP envoie votre message au destinataire.

Quand votre destinataire relève son courrier électronique, son logiciel de messagerie contacte un serveur IMAP. Celui-ci est exécuté par le service de messagerie (comme Gmail ou GMX) ou le FAI. La lecture des messages privés nécessite bien sûr un identifiant et un mot de passe, et le bon sens commande que l'on utilise un cryptage (SSL/TLS) quand on se connecte (la plupart des services de messagerie nécessitent un cryptage). À partir d'un serveur IMAP, un logiciel de messagerie peut télécharger les en-têtes de message et des copies complètes des messages. Et lorsque quelqu'un vous envoie un courriel, c'est l'inverse qui se produit : son logiciel de messagerie contacte son serveur SMTP, l'envoie à votre serveur de messagerie, et quand vous relevez votre boîte aux lettres, vous récupérez une copie du courriel à partir de votre serveur IMAP.

> Les messages sont en général laissés sur le serveur IMAP, et c'est la raison pour laquelle vous pouvez relever vos messages sur votre ordinateur et sur votre téléphone mobile et voir la même liste de messages sur les deux appareils.

Arduino peut-il envoyer des courriels ? Pas si simple !

Si vous voulez créer un projet similaire avec un Arduino, il faudra utiliser un ordinateur externe. L'Arduino peut lire le capteur de fumée et communiquer la valeur à un ordinateur de bureau en utilisant le port série via le port USB. Sur l'ordinateur, un programme Python peut lire le port série (avec l'API pySerial) et envoyer le courriel comme nous allons le voir dans le prochain exemple.

Mais même avec un shield Ethernet ou WiFi, il vous sera difficile d'envoyer un courriel à partir d'un Arduino. Bien que le protocole de courrier électronique soit, en apparence, assez simple, il y a tellement d'incertitudes quand on envoie un courriel qu'il serait difficile d'écrire une bibliothèque SMTP fiable pour Arduino (de plus, de nombreux serveurs requièrent SSL, et l'Arduino Uno et ses modèles similaires ne prennent pas en charge cette technologie).

Bien entendu, il est plus facile de réaliser ce projet avec un Raspberry Pi, ou un microcontrôleur similaire comme le BeagleBone (ou l'hybride Arduino/Linux, l'Arduino Yún).

Code pour Raspberry Pi

Exemple 4.5. smoke_alarm.py

```
# smoke_alarm.py - envoie un courriel toutes les 5 minutes quand de la fumée
est détectée
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp
import smtplib  # 1
from email.mime.text import MIMEText     # 2
# Adresses électroniques
email_to = 'exemple@gmail.com'  # 3
email_from = 'exemple@gmail.com'         # 4
# Paramètres du serveur SMTP
server = 'smtp.gmail.com'        # 5
mail_port = 587
user = 'exemple@gmail.com'       # 6
```

```
password = 'mot_de_passe'    # 7
gracePeriod = 5 * 60 # secondes  # 8
def sendEmail(subject, msg):     # 9
        msg = MIMEText(msg)      # 10
        msg['Subject'] = subject         # 11
        msg['To'] = email_to     # 12
        msg['From'] = email_from
        smtp = smtplib.SMTP(server,mail_port)    # 13
        smtp.starttls()
        smtp.login(user, password)       # 14
        smtp.sendmail(email_from, email_to, msg.as_string())     # 15
        smtp.quit()      # 16
def main():
        while True:
                smokeLevel = mcp.readAnalog()
                print("Le niveau actuel de fumée est %i " % smokeLevel)
                if smokeLevel > 120:
                        print("Fumée détectée")
                        sendEmail("Fumée","Le niveau de fumée était %i" %
smokeLevel) # 17
                        time.sleep(gracePeriod) # 18
                time.sleep(0.5) # s      # 19
if __name__ == "__main__":
        main()
```

1 Python a une bibliothèque intégrée pour la communication SMTP. Vous n'avez pas à programmer les détails de bas niveau du protocole.

2 On n'écrit que du texte. Les en-têtes (une par ligne), une ligne vide, puis le corps du message. Mais de nos jours, on s'attend à ce que les caractères étrangers (non ASCII) fonctionnent. C'est la raison pour laquelle le corps du message est souvent encodé en MIME, tout comme les pièces jointes.

3 L'adresse To est le destinataire du message. Il faudra utiliser votre propre adresse pour recevoir les messages d'avertissement. Il s'agit de variables globales, visibles par toutes les fonctions.

4 L'adresse From peut en théorie être n'importe quoi. Mais à l'époque des filtres antispam, le fait de choisir une adresse bizarre pourrait affecter la livraison du courriel. Vous devez utiliser votre adresse électronique car son nom de domaine va en général correspondre à celle du serveur SMTP que vous utilisez.

5 Le serveur SMTP est le serveur d'envoi. Comme ce programme n'envoie que des courriels et n'en reçoit jamais, le serveur SMTP est le seul serveur nécessaire.

6 Votre identifiant SMTP. Il peut s'agir de votre adresse électronique (<exemple@gmail.com>) ou d'un identifiant plus court (jdoe).

7 Votre mot de passe SMTP doit être le même que celui que vous utilisez pour vous connecter au webmail ou à votre client de messagerie.

8 Même quand la fumée est détectée, vous ne souhaitez probablement pas recevoir 60 courriels par minute. Un tel rythme vous ferait sans doute blacklister par le serveur. C'est pourquoi vous devez ajouter un délai de grâce, 5 minutes, qui est converti en secondes en multipliant 60 secondes par le nombre de minutes (programmez des calculs simples dans le code, plutôt que des nombres magiques sur le papier ; évitez la tentation de répéter la valeur dans un commentaire).

9 Le code qui envoie le courriel réside dans sa propre fonction, pour une plus grande clarté. L'objet (le sujet du message) et le message (le corps du courriel) sont passés en paramètres à la fonction.
10 Le corps du message est encodé en MIME pour gérer convenablement les caractères non ASCII. Comme vous pouvez le voir à partir de la première lettre en majuscule, `MIMEText` est une classe. Le constructeur `MIMEText()` retourne un nouvel objet qui est stocké dans la variable `msg`. `MIMEText` ne sert qu'à créer le message ; elle ne s'occupe pas de connecter quoi que ce soit ou d'envoyer quelque chose.
11 Pour modifier les en-têtes des courriels, vous pouvez utiliser l'objet MIMEText `msg` comme un type de données dictionnaire (http://bit.ly/1jFV3V8).
12 Le destinataire et l'expéditeur proviennent de variables globales.
13 Crée un nouvel objet de la classe SMTP qui sera utilisé pour se connecter au serveur SMTP. Le préfixe (smtplib) est l'espace de noms, qui est créé automatiquement par import smtplib.
14 Connecte au serveur SMTP, en utilisant l'identifiant et le mot de de passe à partir de variables globales.
15 Envoie le courrier, avec la méthode `sendmail()` de l'objet `smtp` de la classe `smtplib.SMTP`. La méthode `smtp.sendmail()` s'attend à voir une chaîne, si bien que l'objet `msg` est transformé en chaîne avec sa méthode intégrée.
16 Ferme la connexion et fait le ménage. La fonction `quit()` est une méthode de l'objet `smtp`.
17 Utilise une chaîne de mise en forme pour créer l'objet `msg`, en remplaçant « %i » par la valeur de `smokeLevel`.
18 Attend 5 minutes après l'envoi du courriel pour en envoyer un autre.
19 Évite d'exécuter la boucle à vitesse maximale en faisant une pause d'une demi-seconde.

Packaging

Nous avons utilisé un boîtier générique pour conditionner notre alarme de fumée (Figure 4.10). Ces boîtiers paraissent rudimentaires, mais ils sont facilement disponibles et leurs trous recouverts par du caoutchouc rendent les modifications simples et rapides.

Figure 4.10 Boîtier à usage général

Le trou pour le capteur de fumée est le seul trou que vous devez percer (Figure 4.11). D'autres trous peuvent être faits à l'aide d'un couteau en perçant soigneusement les ouvertures recouvertes par du caoutchouc. En fait, vous pouvez aussi utiliser ces ouvertures pour le capteur, mais nous préférons les utiliser pour le passage des câbles. Une mèche de 19 mm a servi à percer le

trou pour le capteur et il n'y a pas besoin d'adhésif pour qu'il tienne en place, comme cela est illustré à la Figure 4.12.

Figure 4.11 Trou pour le capteur de fumée

Figure 4.12 Capteur en place

Pour empêcher le Raspberry Pi de se déplacer dans le boîtier, on l'a sécurisé avec du Velcro au fond (Figure 4.13). De cette manière, il est facile de l'enlever pour le débogage ou pour l'utiliser dans d'autres projets.

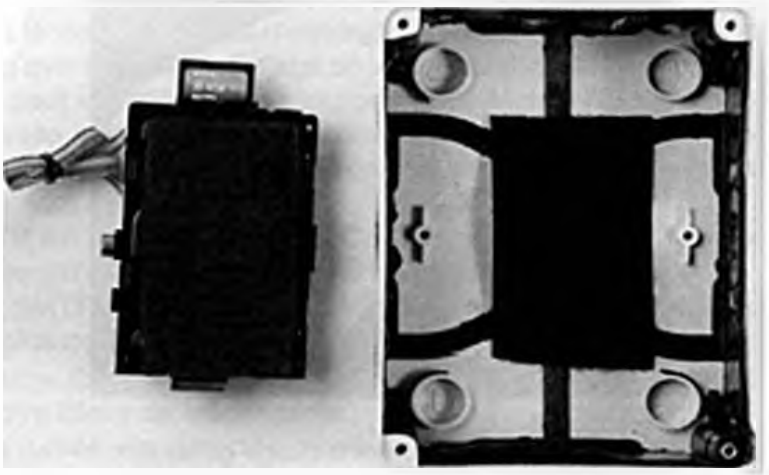


Figure 4.13 Sécurisation du Raspberry Pi avec du Velcro

On a collé une mini-platine de test au fond du couvercle du boîtier (Figure 4.14). Ensuite, nous avons utilisé une attache mono-usage (Serflex) pour regrouper les câbles proprement.

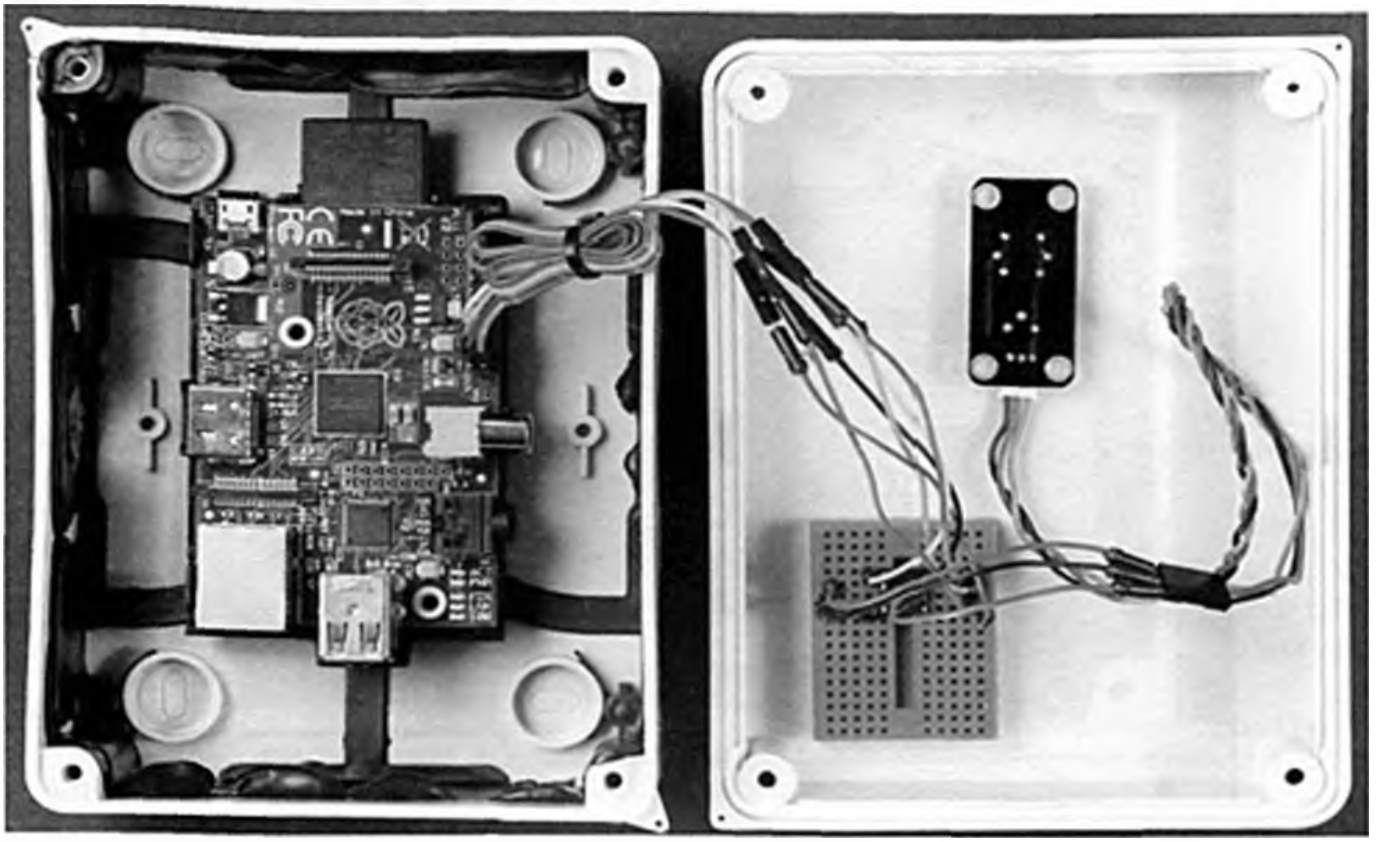


Figure 4.14 Mini-platine de test avec les câbles regroupés par un Serflex

On a fait un trou plus large (~2 cm) sur le côté pour le câble réseau et le câble d'alimentation (Figure 4.15). Nous souhaitions les trous les plus petits possibles pour la version finale du gadget, si bien que nous n'avons pas d'ouvertures pour les autres câbles (Figure 4.16).

Figure 4.15 Trou pour les câbles

Vous pouvez à présent fermer le couvercle et vous êtes prêt à effectuer les tests (voir plus haut la Figure 4.9). Quand vous placez votre alarme, souvenez-vous du sens dans lequel va la fumée.

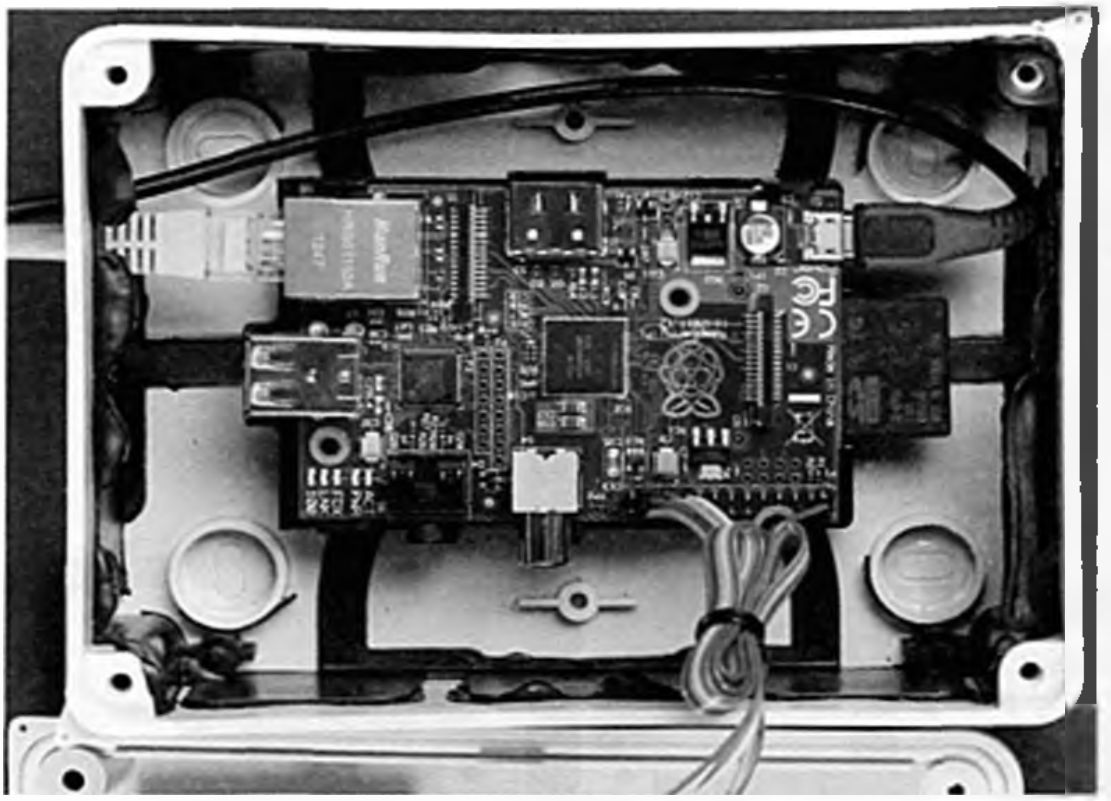


Figure 4.16 Câbles attachés

Votre gadget a maintenant un « nez électronique ». En utilisant la famille des capteurs MQ, vous pouvez détecter les gaz nocifs (CO), inflammables (hydrocarbures), explosifs (hydrogène), ou qui révèlent une intoxication (éthanol dans l'haleine).

Dans ce projet, vous avez écrit un programme Python pour réagir à une modification de l'environnement. Vous pouvez adapter ce programme pour qu'il réagisse à des événements émanant d'autres capteurs. La réaction à ces événements pourrait aussi être autre chose qu'un courriel, comme l'appel d'un service web ou l'activation d'une sortie.

Les capteurs de gaz détectent des choses qui sont difficiles ou impossibles à percevoir par les sens humains. Dans le chapitre suivant, vous allez percevoir un phénomène plus visible : le toucher.

5 Percevoir le toucher

Dans ce chapitre, vous allez faire plus ample connaissance avec un bouton, avec un capteur de pression et avec un capteur de toucher capacitif. Vous apprendrez à utiliser des résistances pull-up afin d'éviter l'instabilité des broches. Enfin, vous construirez une sonnette hantée pour tester vos compétences tactiles dans un projet.

- Le capteur le plus basique qui soit est le bouton si bien que c'est également le capteur tactile le plus courant. Quand on appuie sur un bouton, ses fils se connectent, ce qui ferme un circuit. Un microrupteur est un type de petit bouton résistant.
- Un capteur de pression FlexiForce détecte la quantité de force qui est appliquée. Vous pouvez l'utiliser pour tester la puissance de vos doigts ou pour détecter si quelqu'un est assis sur une chaise.
- Un capteur tactile perçoit le toucher sans composants mobiles et le plus étonnant est qu'il ne nécessite même pas de toucher un objet ! Vous apprendrez à détecter une main placée sur une table, à travers la table.

Même si les capteurs tactiles poursuivent le même but dans vos projets, leurs méthodes de mesure du toucher diffèrent. Un bouton connecte simplement deux fils. Un capteur de pression FlexiForce est basé sur de fines couches d'encre sensible à la pression. Un capteur tactile mesure la capacitance, c'est-à-dire la faculté de stocker de l'électricité.

EXPÉRIENCE : ALLUMAGE D'UNE LED AVEC UN BOUTON

Les boutons (Figure 5.1) sont les capteurs les plus simples. En appuyant sur un bouton, on connecte ses fils, de telle sorte que le bouton agit comme un interrupteur. En relâchant le bouton, on interrompt le circuit. Tous les interrupteurs numériques (comme un contact en ampoule ou un interrupteur à bascule) fonctionnent comme des boutons.

Figure 5.1 Bouton-poussoir

Il existe des boutons de différentes tailles et formes. Quand on utilise une platine de test, il est pratique d'employer un bouton avec quatre fils. Les fils fonctionnent par paires, si bien que deux fils adjacents sont toujours connectés l'un à l'autre. Quand on retourne un bouton, on peut voir les fils qui sont connectés. Il y a une ligne biseautée qui indique les fils connectés (Figure 5.2).

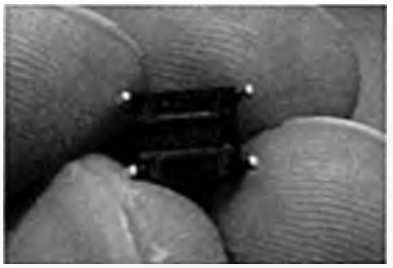


Figure 5.2 Vue de dessous d'un bouton-poussoir

Résistance pull-up

Pour lire une broche de données, elle doit être connectée à quelque chose.

Vous ne devez pas lire une broche qui n'est pas connectée car l'état de la broche est instable (en anglais, on parle de *floating pin*, ce qui signifie que la broche « flotte » entre l'état HIGH et l'état LOW au gré du hasard). Une résistance pull-up met la broche à HIGH quand elle n'est pas connectée à quelque chose, ce qui évite son instabilité.

L'état d'une broche de données peut être lu avec des commandes comme `digitalRead()`, `analogRead()`, `botbook_gpio.read()`, `botbook_mcp2003.readAnalog()`.

Une broche connectée à la masse ou à la valeur HIGH est un exemple évident : quand on lit son état, il est facile à déterminer. Si la broche est connectée à la masse (GND, 0 V), une lecture numérique retourne LOW. Si la broche est connectée à HIGH (+5 V ou +3,3 V), la lecture retourne HIGH.

Si on lit une broche de données non connectée, on obtiendra une valeur « flottante » indéfinie. Cela pourra être HIGH, LOW, changer à chaque seconde ou bien rester identique. De telles valeurs indéfinies sont sans intérêt et il n'y a donc pas lieu de lire une broche instable.

Réfléchissez à un circuit avec un bouton entre une broche de données et la masse (Figure 5.3). Quand on appuie sur le bouton, la broche de données est connectée à la masse et elle renvoie LOW.

Que se passe-t-il quand on relâche le bouton ? Vous devez utiliser une résistance pull-up pour mettre la broche de données à HIGH.

La résistance pull-up est grande. On en utilise souvent une dans la plage des milliers d'ohms. De cette manière, quand on appuie sur le bouton et que la broche de données est connectée à la fois à la masse et à la résistance pull-up, il n'y a pas de court-circuit entre le +5 V et la masse car la voie de la moindre résistance se situe entre la broche de données et la masse, plutôt qu'entre le +5 V et la masse.

Par facilité, le code ci-dessous utilise les résistances pull-up intégrées. Pour l'Arduino, vous pouvez activer les résistances pull-up de la carte avec une seule ligne de code. Pour le Raspberry Pi, vous allez utiliser une des broches qui a une résistance pull-up toujours activée.

Code et connexion pour l'Arduino

Montez le circuit illustré à la Figure 5.3 et chargez le code l'Exemple 5.1.

Appuyez sur le bouton pour allumer la LED montée en surface.

Si l'appui sur le bouton n'allume pas du tout la LED, vérifiez que vous avez connecté correctement le bouton.

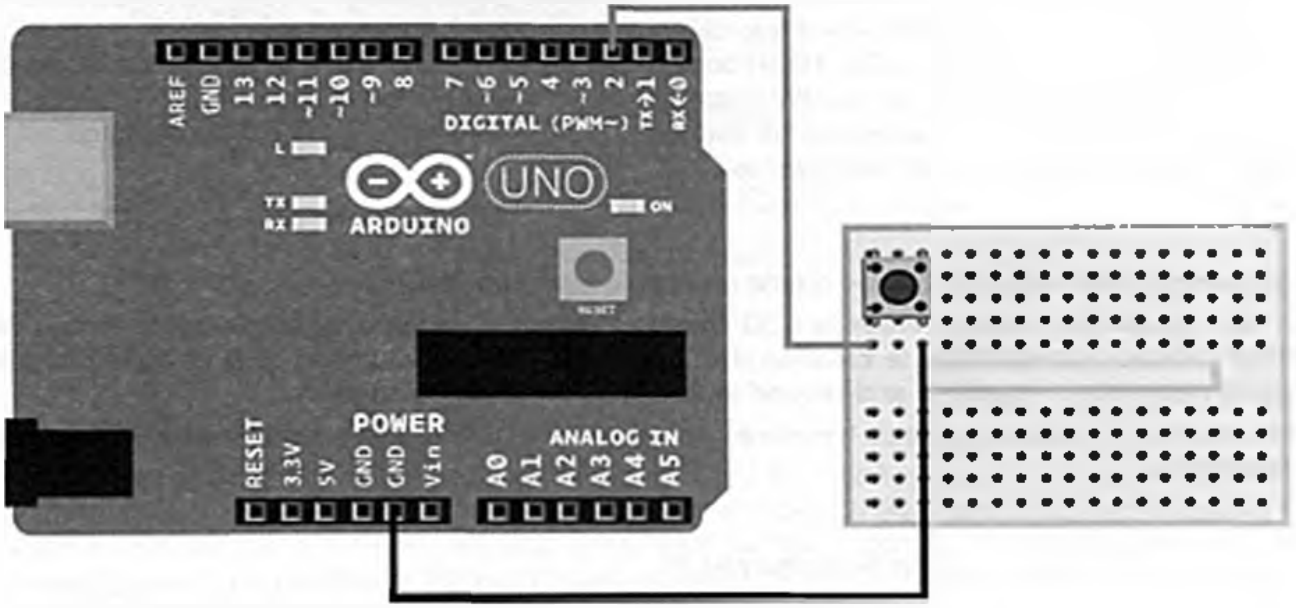


Figure 5.3 Circuit du bouton pour Arduino

Exemple 5.1. button.ino

```
// button.ino - allumo une LED en appuyant sur un bouton
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int buttonPin=2;
int ledPin=13;
int buttonStatus=-1;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT); // 1
  digitalWrite(buttonPin, HIGH); // pull-up interne // 2
}
void loop() {
  buttonStatus=digitalRead(buttonPin); // 3
  if (LOW==buttonStatus) { // 4
    digitalWrite(ledPin, HIGH); // 5
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(20);     // 6
}
```

1 Met la broche en mode INPUT afin de pouvoir contrôler plus tard sa tension avec digitalRead().
2 Connecte la broche D2 à GND avec une grosse résistance interne (20 kΩ). Cela empêche l'instabilité de la broche quand le bouton est ouvert. Les résistances pull-up internes économisent la peine de connecter une résistance supplémentaire à la platine de test. Les dernières versions d'Arduino proposent une nouvelle option pour pinMode(), `INPUT_PULLUP`, qui permet de combiner cette ligne et la précédente en une seule commande : pinMode(buttonPin, INPUT_PULLUP);. Mais l'ancienne méthode fonctionne parfaitement et vous la verrez dans de nombreux programmes écrits avant que ce nouveau mode ait été introduit.
3 Lit la tension de la broche D2. HIGH correspond au +5 V, LOW à 0 V, la tension de la masse.
4 Si la tension du bouton est à LOW (signifiant qu'on a appuyé sur le bouton)...
5 ...on allume une LED. La broche de données D13 est connectée à la LED de l'Arduino.
6 Petite pause pour ne pas pénaliser le CPU.

Ce code maintient la LED allumée quand on appuie sur le bouton.

Si vous souhaitez plutôt inverser la LED chaque fois que vous appuyez sur le bouton, vous devez filtrer l'entrée. Par exemple, la création d'un appareil où un clic allume la LED et où le clic suivant l'éteint nécessite un système anti-rebond (appelé debouncing en anglais).

Vous trouverez un exemple de ce type de programme dans l'IDE Arduino dans **Fichier→Exemples→2.Digital→Debounce**.

Code et connexion pour le Raspberry Pi

Câblez le Raspberry Pi selon le schéma de la Figure 5.4 et exécutez le programme Python de l'Exemple 5.2.

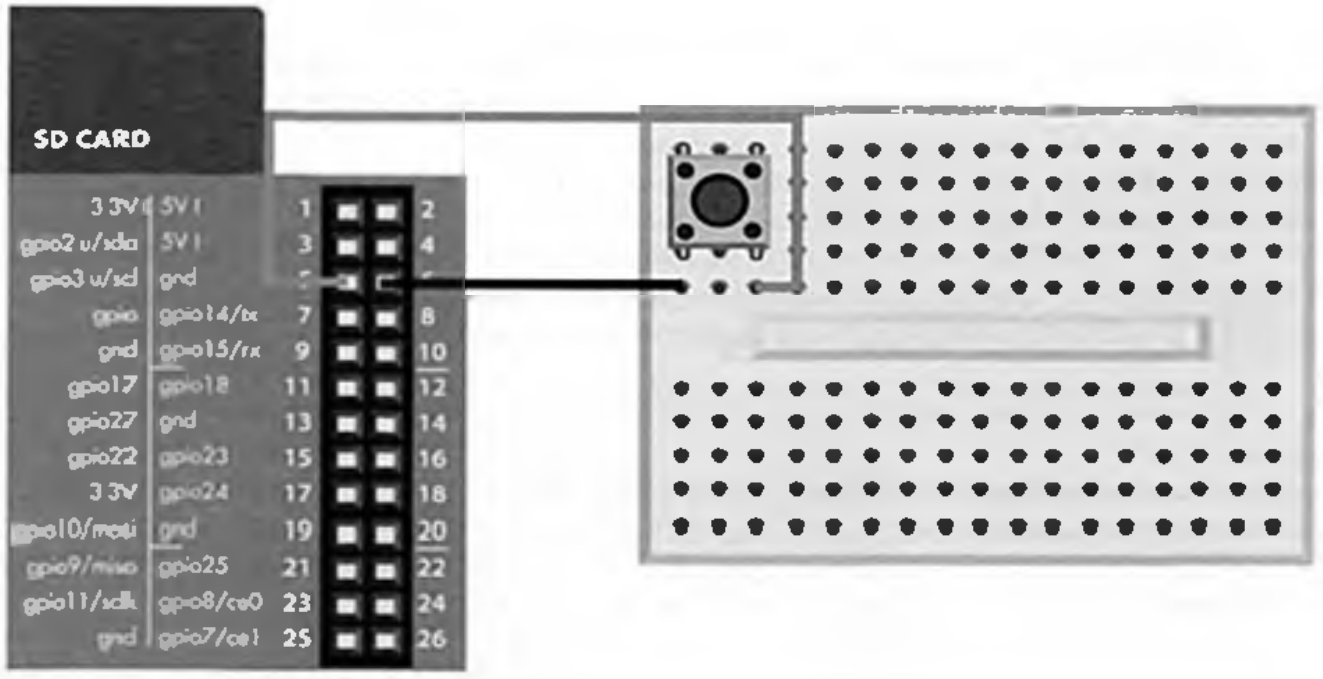


Figure 5.4 Circuit du bouton pour Raspberry Pi

Exemple 5.2. button.py

```
# button.py - écrit à l'écran l'état du bouton
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio      # 1
def main():
        buttonpin = 3    # pull-up interne # 2
        gpio.mode(buttonpin, "in")       # 3
        while (True):    # 4
                buttonUp = gpio.read(buttonpin) # 5
                if(buttonUp == gpio.HIGH):
                        print "Bouton pas pressé"
                else:
                        print "Bouton pressé"
                time.sleep(0.3) # secondes        # 6
if __name__ == "__main__":
        main()
```

1 Assurez-vous qu'il y a une copie de la bibliothèque *botbook_gpio.py* dans le même répertoire que ce programme. Vous pouvez télécharger cette bibliothèque avec tous les exemples de code à partir de http://botbook.com. Voir *GPIO sans root* au chapitre 1 pour des informations sur la configuration du Raspberry Pi pour l'accès au port GPIO.
2 La broche gpio3 a une résistance pull-up interne (elle est connectée au +3,3 V *via* une résistance de 1800 ohms).
3 Définit la broche gpio3 en mode « in » pour lire plus tard sa tension avec gpio.read().
4 Continue à tourner tant que l'utilisateur n'interrompt pas le programme avec Ctrl+C.
5 Lit la valeur de la broche (True ou False).
6 Fait une pause pendant un moment pour empêcher la boucle `while(True)` d'accaparer 100% du CPU. La petite pause permet aussi de lire plus facilement le texte affiché.

EXPÉRIENCE : MICRORUPTEUR

Un microrupteur (*microswitch* en anglais) est un type de bouton (Figure 5.5). Quand on presse le bouton, les fils sont connectés. Cette expérience écrit « 0 » sur le port série quand on appuie sur le microrupteur.

Figure 5.5 Microrupteur

On peut utiliser un microrupteur comme un bouton normal (voir l'*Expérience : allumage d'une LED avec un bouton* au début de ce chapitre). Tout comme avec un bouton, le modèle spécifique

de microrupteur n'a pas d'importance et vous pouvez utiliser le même code et le même schéma de connexion pour tout microrupteur courant.

Les microrupteurs sont populaires en raison de leur faible coût, de leur petite taille et de leur résistance. Un microrupteur classique pourra supporter plus d'un million d'appuis. Le cliquetis provient d'un petit bras tournant autour d'un axe.

Cette connexion évite l'instabilité de la broche en utilisant une résistance pull-up (voir *Résistance pull-up* au début de ce chapitre).

Microrupteur pour Arduino

Montez le circuit illustré à la Figure 5.6 et chargez le code de l'Exemple 5.3.

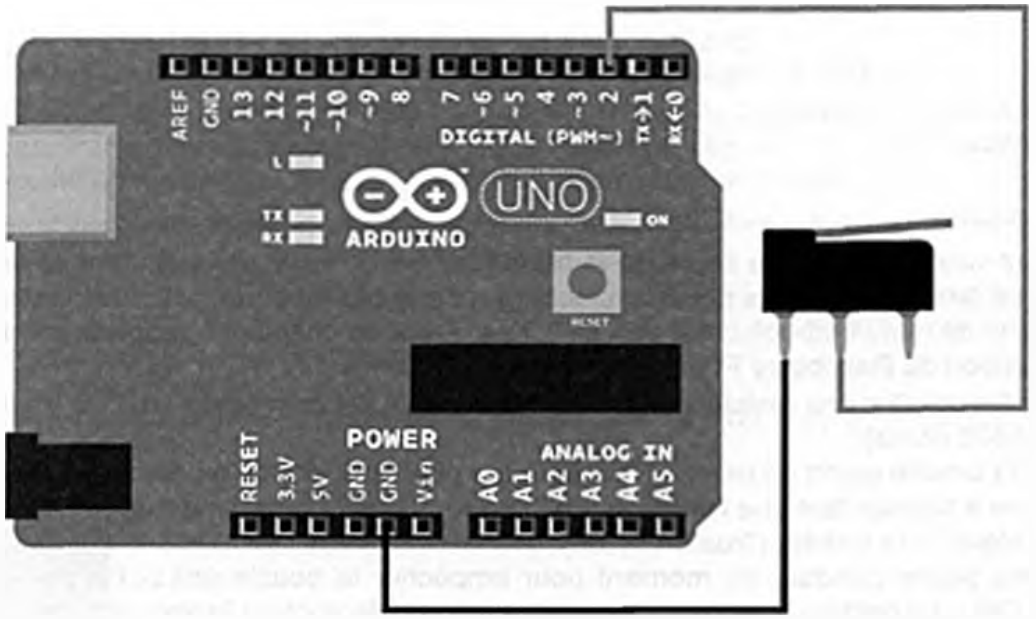


Figure 5.6 Circuit de microrupteur pour Arduino

Exemple 5.3. microswitch.ino

```
// microswitch.ino - affiche l'état du microrupteur sur le port série
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int switchPin = 2;
int switchState = -1;   // 1
void setup() {
  Serial.begin(115200);
  pinMode(switchPin, INPUT);
  digitalWrite(switchPin, HIGH); // pull-up interne // 2
}
void loop() {
  switchState = digitalRead(switchPin);
  Serial.println(switchState);  // 3
  delay(10);
}
```

1 Initialise à une valeur impossible.
2 L'écriture de HIGH sur la broche d'INPUT la connecte au +5 V avec une grosse résistance (20 kΩ).
3 Si on appuie sur le bouton, la broche D2 est directement connectée à la masse *via* le bouton, et switchState est à LOW. Quand D2 est directement connectée à la masse, la résistance de 20 kΩ de la résistance pull-up est si grande qu'elle n'affecte pas l'état de D2. Si on n'appuie pas sur le bouton, la résistance pull-up interne la met à HIGH (+5 V).

Vous pouvez réaliser des antennes simples pour votre robot avec un microrupteur auquel vous aurez collé un Serflex (Figure 5.7).

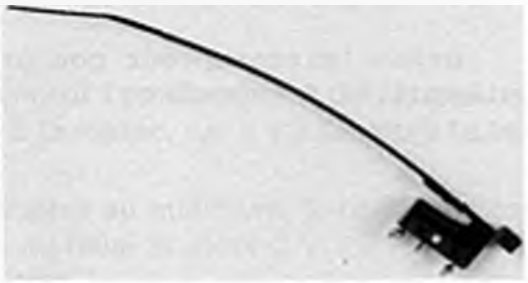


Figure 5.7 Antenne réalisée avec un microrupteur

Microrupteur pour Raspberry Pi

Assemblez le Raspberry Pi selon le schéma de la Figure 5.8 et exécutez le programme Python de l'Exemple 5.4.

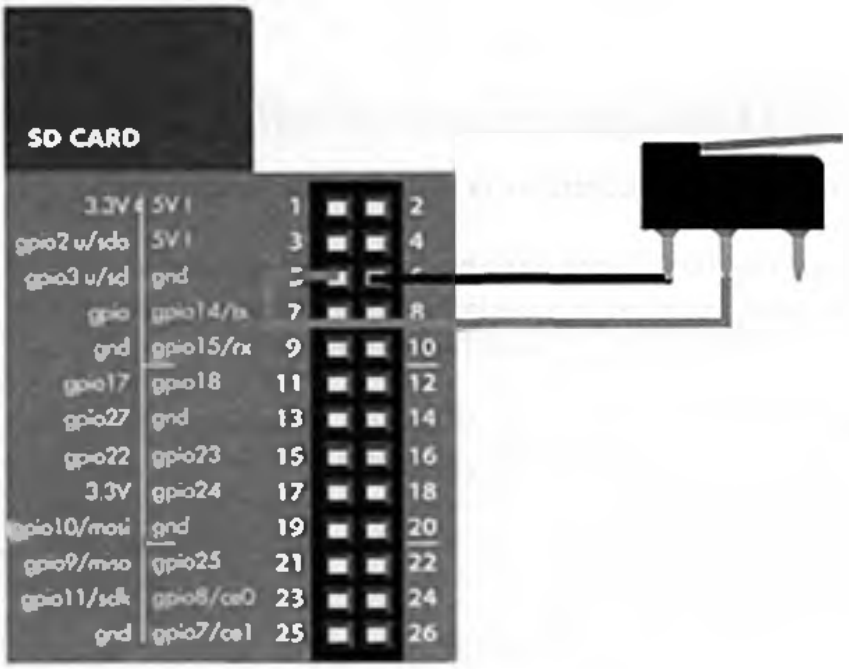


Figure 5.8 Circuit de microrupteur pour Raspberry Pi

Exemple 5.4. microswitch.py

```
# microswitch.py - écrit à l'écran l'état du microrupteur
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio      # 1
def main():
        switchpin = 3    # pull-up interne # 2
        gpio.mode(switchpin, "in")
        while (True):
                switchState = gpio.read(switchpin)       # 3
                if(switchState == gpio.LOW):
                        print "Microrupteur pressé"
                else:
                        print "Microrupteur non pressé"
                time.sleep(0.3) # secondes
if __name__ == "__main__":
        main()
```

1 Assurez-vous qu'il y a une copie de la bibliothèque *botbook_gpio.py* dans le même répertoire que ce programme. Vous pouvez télécharger cette bibliothèque avec tous les exemples de code à partir de http://botbook.com. Voir *GPIO sans root* au chapitre 1 pour des informations sur la configuration du Raspberry Pi pour l'accès au port GPIO.

2 Les broches gpio2 et gpio3 ont des résistances pull-up toujours activées.

3 Quand on appuie sur le bouton, la broche gpio3 est connectée à la masse et reçoit ainsi la valeur LOW. La résistance de la résistance pull-up interne est si grande que la résistance pull-up n'affecte pas l'état de la broche gpio3 quand gpio3 est connectée directement à la masse. Quand on n'appuie pas sur le bouton, la résistance pull-up la met à HIGH.

EXPÉRIENCE : POTENTIOMÈTRE (RÉSISTANCE VARIABLE)

Dans cette expérience, vous allez contrôler la cadence de clignotement d'une LED en tournant un potentiomètre.

Un potentiomètre est une résistance variable (on tourne un bouton pour modifier la résistance).

Une résistance normale, qui n'est pas variable, ne possède que deux fils. Si vous voulez utiliser un potentiomètre à la place d'une seule résistance, utilisez le fil du milieu et l'un des deux fils qui sont sur le côté.

Pourquoi un potentiomètre (que l'on abrège parfois en pot ou potar) a-t-il trois fils ? Quand on lit une broche de données, la broche doit toujours être connectée à quelque chose.

Le moyen le plus simple d'utiliser un potentiomètre est de connecter un côté au positif (+5 V) et l'autre côté à la masse (GND). Ensuite, le fil du milieu du potentiomètre est connecté à la broche de données. La tension du fil du milieu oscille ensuite entre +5 V et GND quand vous tournez le bouton. Le potentiomètre agit alors comme un réducteur de tension, qui est un circuit qui diminue une tension en la divisant entre deux résistances (une connectée à la tension positive, l'autre à la masse). Dans le cas d'un potentiomètre, les résistances de chaque côté du potentiomètre changent quand vous tournez le bouton.

Vous pouvez voir un diagramme de potentiomètre à la Figure 5.9. Le fil vert du milieu est connecté à une broche de données qui mesure la tension. Les fils de côté sont connectés à la masse (noir) et au +5 V (rouge). Il y a une résistance ronde en forme d'arc (rouge-noir) qui va du positif au négatif.

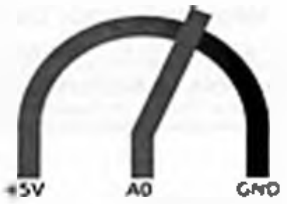


Figure 5.9 Diagramme de potentiomètre

En tournant le bouton, on choisit l'emplacement de l'arc où le balai touche la résistance. Le positif n'est pas court-circuité à la masse, car il y a toujours la longue résistance en forme d'arc entre le positif et la masse.

Si vous tournez le potentiomètre au minimum, le balai vert touche la masse noire (GND). La broche de données connectée au balai lit alors 0 V. La totalité de la résistance en forme d'arc sépare le balai de la borne positive.

Si vous tournez le potentiomètre au maximum, le balai vert touche la borne rouge positive (+5 V). La broche de données connectée au balai lit alors +5 V.

Au fur et à mesure que vous tournez le balai sur la résistance en forme d'arc, vous pouvez sélectionner une tension comprise entre 0 V et +5 V.

> Cet exemple utilise le +5 V de l'Arduino. Un potentiomètre fonctionne de la même manière avec le Raspberry Pi, mais vous devez utiliser le +3,3 V, sinon vous allez endommager la broche de données.

Potentiomètre pour Arduino

La Figure 5.10 illustre le schéma de connexion pour Arduino. Montez le circuit en suivant le schéma puis chargez et exécutez le sketch de l'Exemple 5.5.

Exemple 5.5. pot.ino

```
// pot.ino - contrôle la vitesse de clignotement de la LED avec un
potentiomètre
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int potPin=A0; // 1
int ledPin=13;
int x=0; // 0..1023
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(potPin, INPUT);
}
void loop() {
  x=analogRead(potPin); // 2
  digitalWrite(ledPin, HIGH);
  delay(x/10); // ms    // 3
```

```
    digitalWrite(ledPin, LOW);
    delay(x/10);
}
```

1 Vous devez utiliser des broches analogiques pour lire les capteurs de résistance analogique.
2 Maintenant x a une valeur brute analogRead(), allant de 0 (0 V) à 1023 (5 V).
3 La résistance du potentiomètre contrôle la cadence de clignotement. Ce délai varie de 0 ms à 100 ms (1023/10).

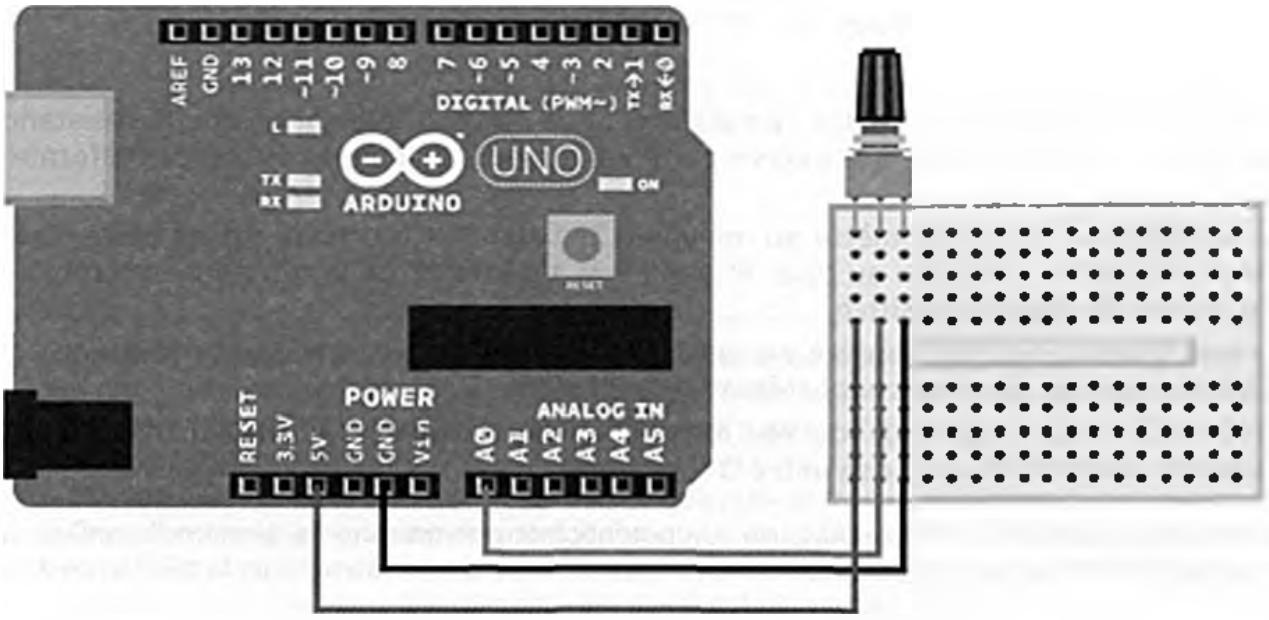


Figure 5.10 Circuit d'un potentiomètre pour Arduino

Potentiomètre pour Raspberry Pi

La Figure 5.11 illustre le circuit d'utilisation d'un potentiomètre avec un Raspberry Pi. Montez les composants et exécutez le programme de l'Exemple 5.6.

À la différence d'Arduino, Raspberry Pi ne possède pas de convertisseur analogique-numérique intégré. Cela signifie que tous les circuits utilisant des capteurs de résistance analogique sont plus complexes avec un Raspberry Pi qu'un Arduino.

Un potentiomètre est un composant très simple, comme vous pouvez le voir en examinant le code Arduino. La lecture de la valeur analogique est un peu plus compliquée avec un Raspberry Pi.

Au chapitre 3, vous vous êtes simplifié la vie pour lire les valeurs analogiques avec notre bibliothèque botbook_mcp3002 (*Œil composé pour Raspberry Pi*). Mais vous êtes-vous posé la question de savoir comment la bibliothèque botbook_mcp3002 fonctionne ?

Ce code Raspberry Pi introduit le code sous-jacent nécessaire à la communication avec le convertisseur analogique-numérique MCP3002. Auparavant, vous vous êtes contenté d'utiliser la bibliothèque sans vous poser de questions, mais ce code va vous montrer comment lire des valeurs

directement à partir de la puce grâce à l'interface SPI, ce qui facilitera votre compréhension du fonctionnement de la bibliothèque.

D'autres exemples de code utilisent cette bibliothèque, ce qui les rend plus simples.

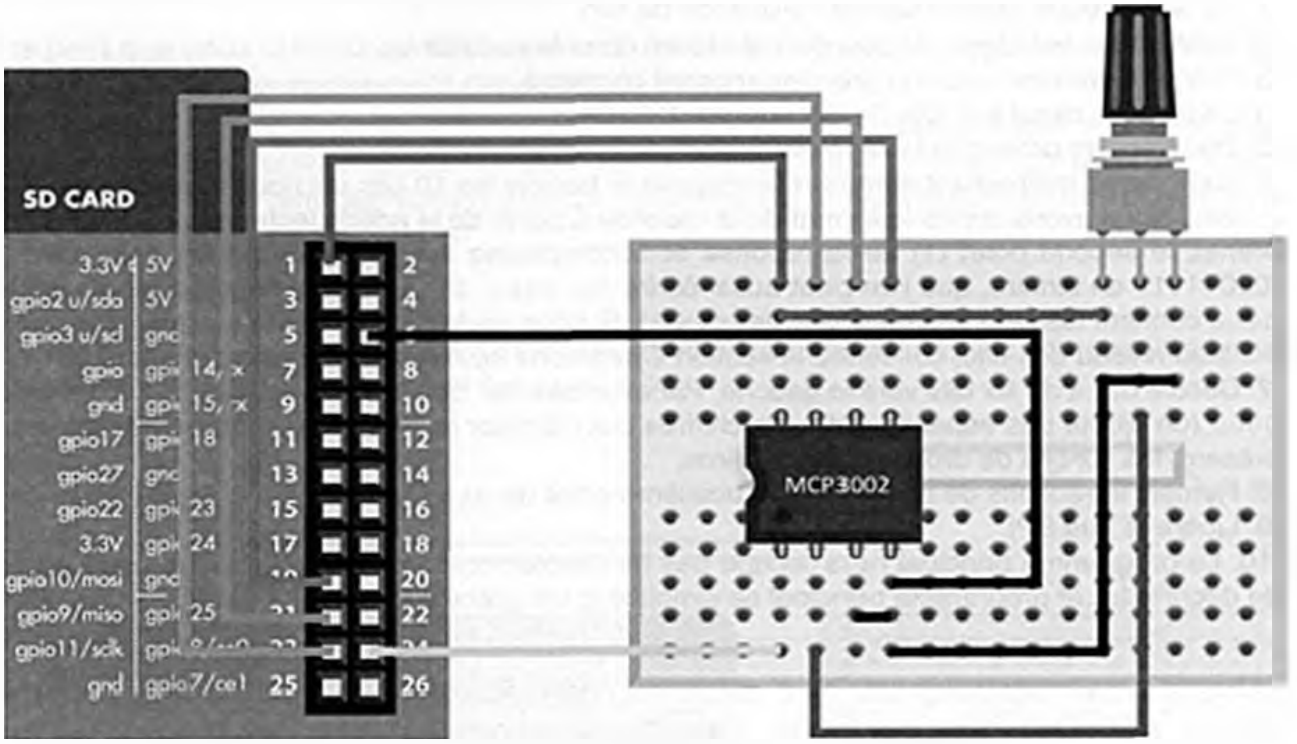


Figure 5.11 Circuit d'un potentiomètre pour Raspberry Pi

Exemple 5.6. pot.py

```
# pot.py - potentiomètre avec un convertisseur mcp3002
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import spidev # aide à l'installation dans botbook_mcp3002.py  # 1
import time
def readPotentiometer():
        spi = spidev.SpiDev()    # 2
        spi.open(0, 0)   # 3
        command = [1, 128, 0]    # 4
        reply = spi.xfer2(command)       # 5
        # on analyse 10 bits des 24 bits de la réponse
        data = reply[1] & 31     # 6
        data = data << 6         # 7
        data = data + (reply[2] >> 2)    # 8
        spi.close()      # 9
        return data
def main():
        while True:
                pot = readPotentiometer()      # 10
                print("La valeur du potentiomètre est %i " % pot)
                time.sleep(0.5)
```

```
if __name__ == "__main__":
        main()
```

1 La bibliothèque spidev facilite l'utilisation de SPI.
2 Crée un nouvel objet `SpiDev` qui est stocké dans la variable `spi`.
3 Ouvre le premier canal du premier appareil connecté.
4 Le premier canal est 128 ; le second est 128+64.
5 Transfère un octet et lit la réponse.
6 Vous devez maintenant analyser la réponse et trouver les 10 bits qui nous intéressent sur les 24 bits. Nous avons appris le format de la réponse à partir de la notice technique du MCP3002 : prenez le second octet (1) de la réponse et accomplissez au niveau du bit un AND avec 31 (00011111 en binaire, que l'on peut aussi écrire `0b 0001 1111`). Après l'opération, la variable `data` contient les cinq bits de droite de reply[1]. Si vous voulez comprendre en détail les opérations au niveau des bits, consultez la section *Opérations au niveau des bits* du chapitre 8.
7 Décale `data` de six bits vers la gauche. Par exemple, `0b 0001 1001` devient `0b 0110 0100 0000` (on ajoute des espaces entre les chiffres pour faciliter la lecture des nombres binaires). À présent, les six bits de droite sont des zéros.
8 Remplit les six bits de droite avec le troisième octet de `data`, data[2].
9 Libère le bus SPI.
10 Le programme principal ne s'occupe pas de l'implémentation des détails que nous venons de décrire. Ici, le programme principal se simplifie la vie grâce à un appel à readPotentiometer().

Cela vous paraît contraignant ? Le reste du code de cet ouvrage utilise la bibliothèque `botbook_mcp3002`, si bien que vous pouvez vous concentrer sur l'essentiel et laisser la bibliothèque gérer toutes les opérations au niveau des bits.

EXPÉRIENCE : PERCEVOIR LE TOUCHER SANS TOUCHER (CAPTEUR DE TOUCHER CAPACITIF QT113)

Le secret d'un capteur de toucher capacitif est qu'il ne perçoit pas du tout le toucher. En fait, il mesure le temps mis pour charger un morceau de câble électriquement. S'il y a un être humain (un grand sac rempli d'eau) dans les parages, le câble met plus longtemps à se charger.

Figure 5.12 Circuit intégré du capteur de toucher capacitif QT113

Le QT113 est un circuit intégré pour la détection capacitive. Le protocole est simple : quand le toucher est détecté, la broche de sortie passe à LOW.

Un bon capteur capacitif a besoin d'une surface. Dans cette expérience, nous allons utiliser un câble métallique enroulé en spirale. Un morceau de papier d'aluminium pourrait aussi fonctionner. Le circuit utilise également un condensateur de 10-500 nF.

Il y a plusieurs façons de réaliser une détection capacitive :

- Un câble et une horloge simple
- Un câble et la bibliothèque CapSense
- Une puce spécialisée (comme le QT113)

Il se peut que vous utilisiez quotidiennement un capteur capacitif ; si vous avez un smartphone, son écran tactile est probablement capacitif.

Ce type de détection capacitive nécessite une réelle mise à la terre et une pile n'est pas suffisante. Par exemple, l'utilisation d'un Arduino alimenté par un ordinateur portable ne fonctionne pas de manière fiable tant que vous n'avez pas branché votre ordinateur sur une prise électrique.

QT113 pour Arduino

La Figure 5.13 illustre le circuit utilisant le QT113 avec Arduino. Montez les composants en suivant le schéma et exécutez le sketch de l'Exemple 5.7.

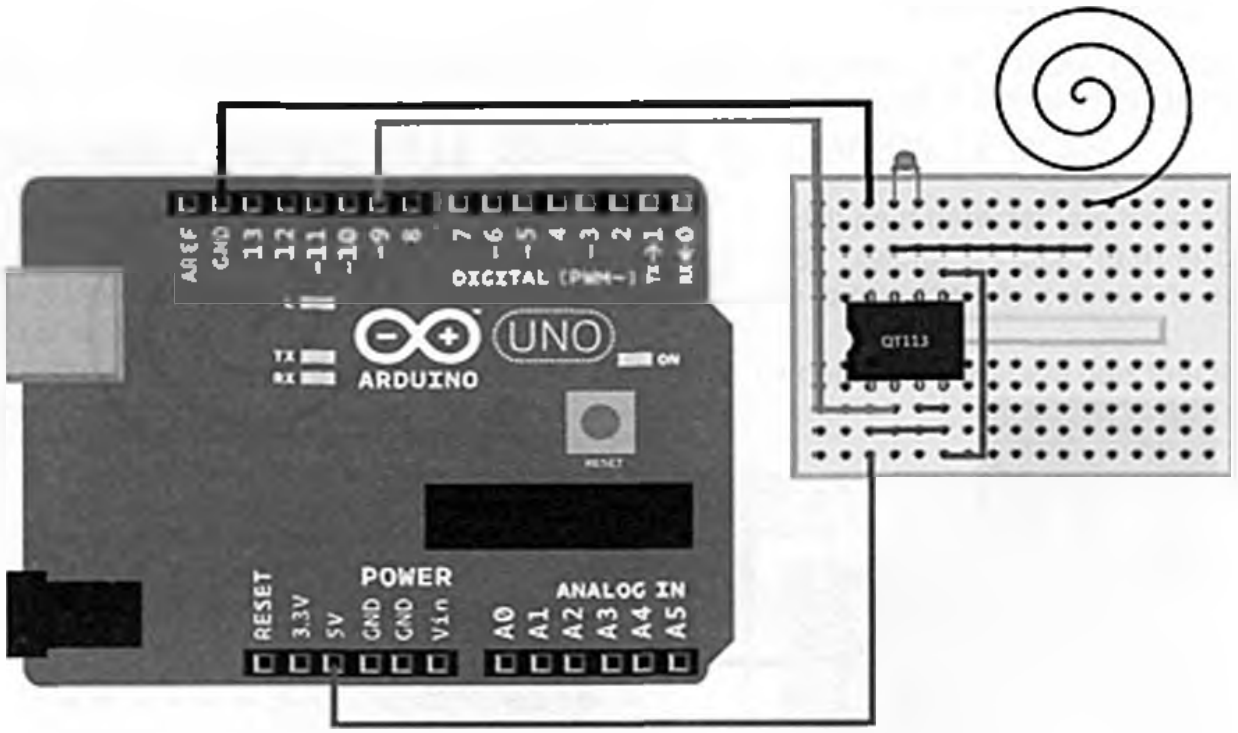


Figure 5.13 Circuit QT113 pour Arduino

Exemple 5.7. qt113.ino

```
// qt113.ino - capteur tactile qt113
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int sensorPin = 9;
void setup()
{
  pinMode(sensorPin, INPUT);
  Serial.begin(115200);
}
```

```
void loop()
{
  int touch = digitalRead(sensorPin);    // 1
  if(touch == LOW) {
    Serial.println("Toucher détecté");
  } else {
    Serial.println("Pas de toucher détecté");
  }
  delay(100);
}
```

1 Il s'agit d'un simple interrupteur numérique.

QT113 pour Raspberry Pi

La Figure 5.14 illustre les connexions pour utiliser un Raspberry Pi avec le QT113. Câblez les composants et exécutez le programme de l'Exemple 5.8.

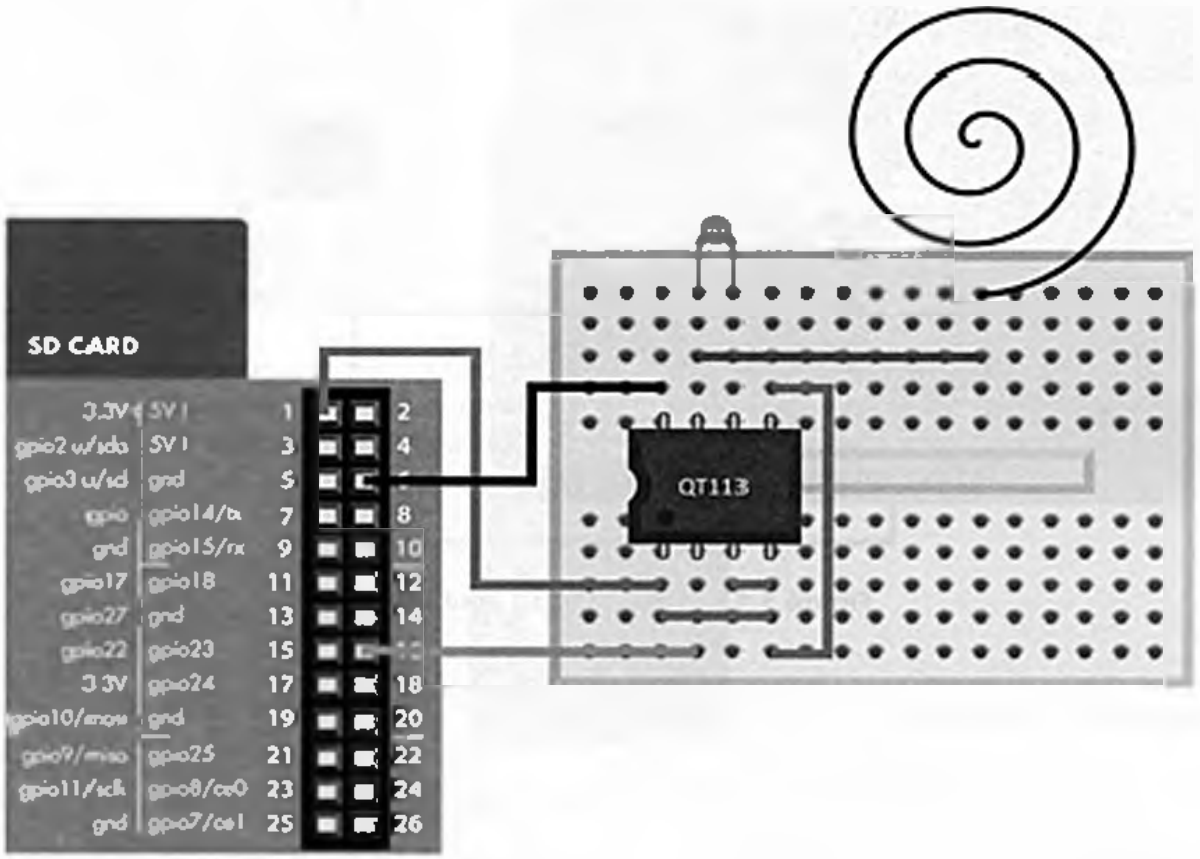


Figure 5.14 Circuit QT113 pour Raspberry Pi

Exemple 5.8. qt113.py

```
# qt113.py - lit les informations du QT113
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio
def main():
        limitPin = 23
        gpio.setMode(limitPin, "in")
        while True:
                if gpio.read(limitPin) == gpio.LOW:      # 1
                        print("Toucher détecté !")
                time.sleep(0.5)
if __name__ == "__main__":
  main()
```

1 Il s'agit d'un simple interrupteur numérique.

EXPÉRIENCE : DÉTECTER LE TOUCHER À TRAVERS LE BOIS

Quand il est correctement réglé, un capteur de toucher capacitif peut détecter une main à travers des objets solides. Le résultat est très convaincant car c'est comme si l'objet était le capteur (Figure 5.15). Par exemple, nous avons réalisé une étagère qui modifie la couleur de ses LED quand on la touche.

En réalité, le bois n'affecte pas la détection capacitive. Le capteur derrière le bois fonctionne à travers le bois presque aussi bien que dans l'air ambiant.

Figure 5.15 Pour l'utilisateur, on dirait que le bois est le capteur

Pour essayer cela, utilisez le code et les connexions du QT113 étudiées précédemment. Cette fois-ci, vous allez devoir modifier le câble pour amplifier sa capacité à détecter les modifications du champ électromagnétique. Fondamentalement, il vous faut un morceau de métal plus gros à l'extrémité du câble. Le fait de faire une spirale ou de fixer du papier d'aluminium a très bien fonctionné pour nous (Figure 5.16).

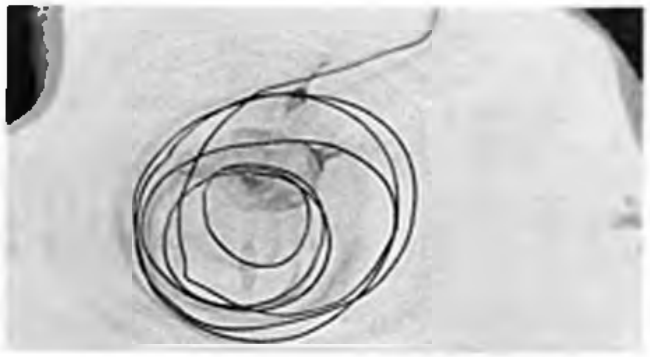


Figure 5.16 Spirale pour détecter le toucher, masquée à l'utilisateur

EXPÉRIENCE : SENTEZ LA PRESSION (FLEXIFORCE)

Le capteur FlexiForce mesure la pression exercée sur sa tête arrondie (Figure 5.17).

Le FlexiForce est un simple capteur de résistance analogique. Plus vous appuyez, plus la résistance diminue. Il n'a que deux fils. Le troisième fil du milieu n'est pas connecté, mais sa présence facilite les connexions. Comme il ne s'agit que d'une résistance, il n'a pas de polarité et vous pouvez donc le connecter dans n'importe quel sens. Vous pouvez même le tester avec un multimètre placé sur la position résistance.

Nos étudiants utilisent un FlexiForce pour élaborer un réveil dont la sonnerie ne s'éteint que lorsque vous sortez du lit (le FlexiForce mesure la pression que votre corps exerce sur le lit). Si vous avez besoin d'une plus grande surface pour effectuer les mesures, vous pouvez placer une pièce en bois par-dessus le FlexiForce.

Figure 5.17 FlexiForce

FlexiForce pour Arduino

Le FlexiForce est un capteur de résistance analogique. Comme Arduino a un convertisseur analogique-numérique intégré, la lecture de la valeur se fait par un simple appel à analogRead().

Une résistance pull-up de 1 MΩ est utilisée pour éviter l'instabilité de la broche. Pour une explication des résistances pull-up, consultez la section consacrée à ce sujet au début de ce chapitre.

La Figure 5.18 illustre le schéma de câblage, et l'Exemple 5.9 liste le sketch. Après avoir câblé l'Arduino, chargez le sketch et exécutez-le.

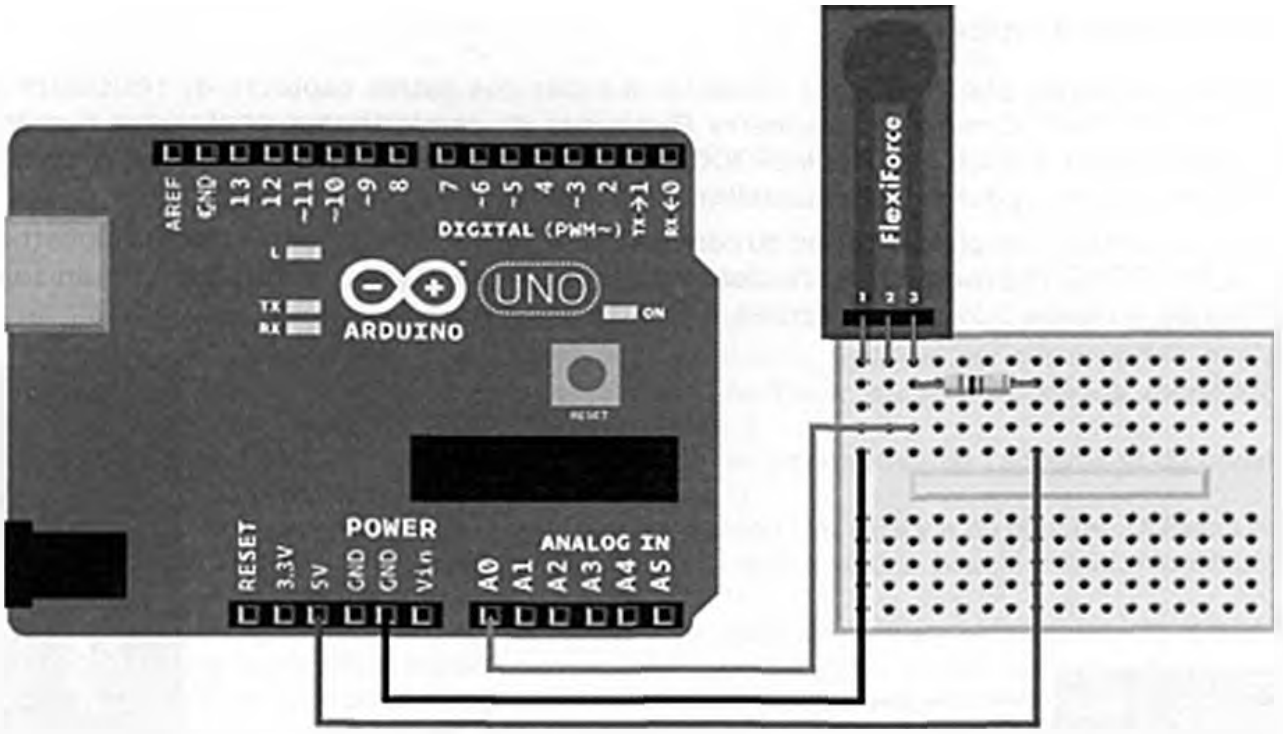


Figure 5.18 Connexions du FlexiForce pour Arduino

Exemple 5.9. flexiforce_25.ino

```
// flexiforce_25.ino - envoie les valeurs de pression du FlexiForce sur le
Moniteur série de l'ordinateur
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int squeezePin=A0;      // 1
int x=-1; // 0..1023
void setup() {
  pinMode(squeezePin, INPUT);
  Serial.begin(115200); // bit/s
}
void loop() {
  x=analogRead(squeezePin);  // 2
  Serial.println(x);
  delay(500); // ms
}
```

1 Les constantes prédéfinies d'Arduino (`A0`, `A1`, `A2`...) sont le meilleur moyen pour faire référence aux broches analogiques de l'Arduino.
2 Comme avec tout capteur de résistance analogique, `analogRead()` retourne la tension de la broche, de 0 (0 V) à 1023 (+5 V).

FlexiForce pour Raspberry Pi

Les connexions du Raspberry Pi sont similaires à celles des autres capteurs de résistance analogique (Figure 5.19). Comme le Raspberry Pi n'a pas de convertisseur analogique-numérique intégré, on utilise une puce externe MCP3002. Comme le FlexiForce n'a que deux fils, on utilise une résistance pull-up pour éviter l'instabilité de la broche.

Comme les détails d'implémentation du convertisseur MCP3002 sont gérés dans la bibliothèque botbook_mcp3002, le programme principal est assez simple. Câblez le Raspberry Pi en suivant le schéma de la Figure 5.19, puis exécutez le code de l'Exemple 5.10.

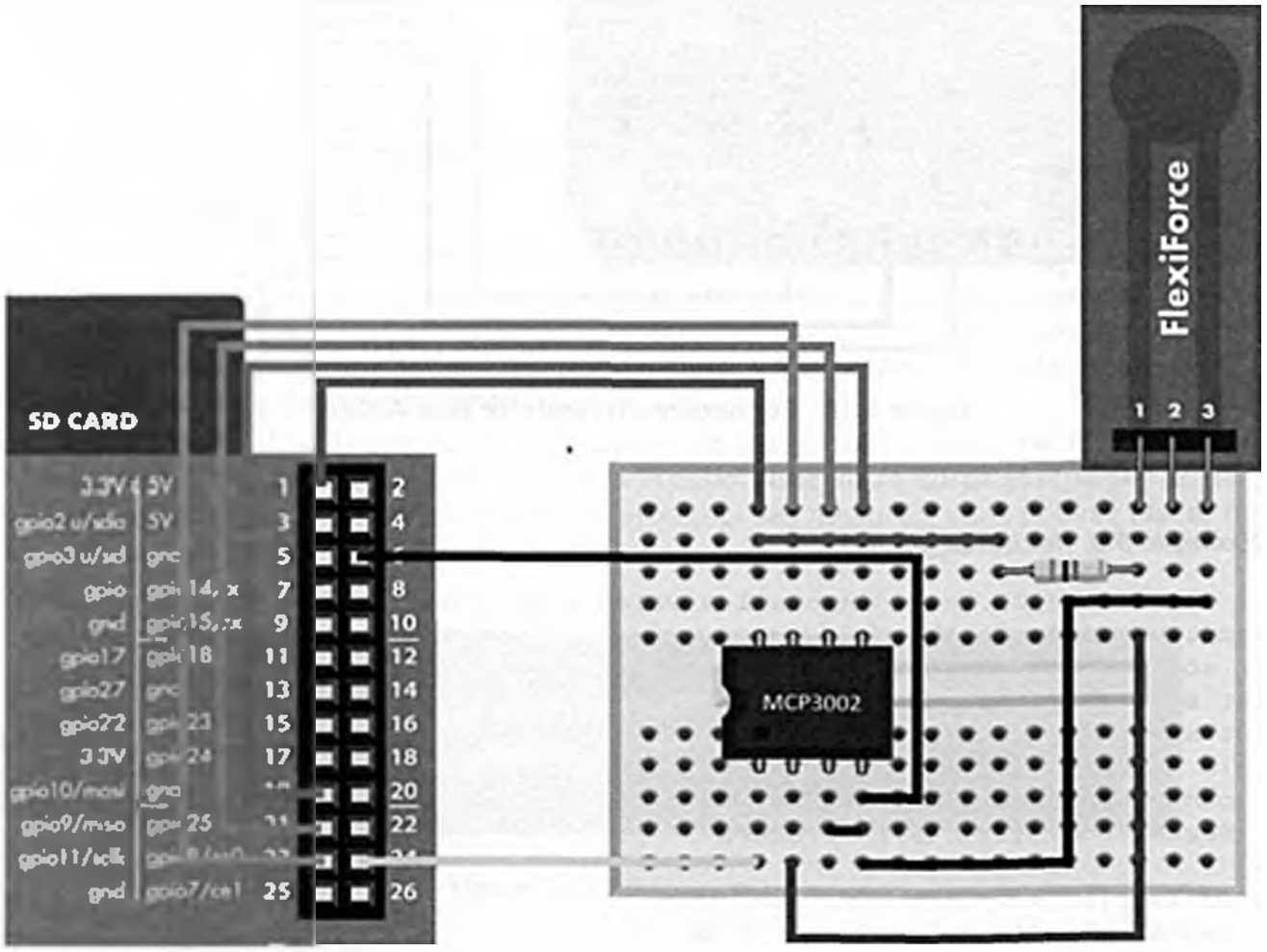


Figure 5.19 Connexions du FlexiForce pour Raspberry Pi

Exemple 5.10. flexiforce.py

```
# flexiforce.py - détecte la force et affiche la valeur à l'écran.
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp # 1
def readFlexiForce():
        return mcp.readAnalog() # 2
def main():
        while True:     # 3
```

```
                f = readFlexiForce()     # 4
                print("La force est %i " % f)       # 5
                time.sleep(0.5) # s      # 6
if __name__ == "__main__":
        main()
```

1 La bibliothèque *botbook_mcp3002.py* doit se trouver dans le même répertoire que *flexiforce.py*. Vous devez aussi installer la bibliothèque *spidev*, qui est importée par *botbook_mcp3002*. Lisez les commentaires du fichier *botbook_mcp3002/botbook_mcp3002.py* ou lisez la section *Installation de SpiDev* au chapitre 3.
2 Lit le premier capteur connecté au MCP3002. Les paramètres de l'appareil et du canal ne sont pas nécessaires quand ils sont tous les deux à 0.
3 Les périphériques embarqués s'exécutent en général tant qu'il y a du courant. Utilisez Ctrl+C si vous voulez arrêter le programme. Quand vous exécutez une boucle infinie, n'oubliez pas d'ajouter delay() à la fin de la boucle.
4 Vous pouvez facilement utiliser readFlexiForce() dans des projets plus importants quand elle se trouve dans une fonction séparée, comme dans cet exemple. Le nom de la fonction est suffisamment explicite pour qu'aucun commentaire ne soit nécessaire.
5 Affiche la force. La chaîne affichée est formatée avec une chaîne de mise en forme, où la valeur de f remplace %i.
6 Un délai est nécessaire pour les longues boucles (pour éviter l'utilisation de 100% du CPU) et aussi pour permettre de lire le texte affiché à l'écran.

EXPÉRIENCE : CONSTRUISEZ VOTRE PROPRE CAPTEUR TACTILE

Si la détection capacitive n'est que la mesure du temps qu'un composant met à se charger électriquement, est-il possible de créer soi-même un tel dispositif en se passant de la puce QT113 ? On peut en effet percevoir le toucher simplement avec une feuille de papier aluminium, une résistance et une bonne terre reliée à une carte Arduino.

La détection capacitive est une question d'ordre pratique : même si le principe est simple, l'implémentation est affaire de précision. Outre une bonne mise à la terre, la mesure doit utiliser des moyennes glissantes pour lisser toute fluctuation aléatoire.

Pour bénéficier d'une mise à la terre fiable, la source d'alimentation de l'Arduino doit être connectée à une prise murale. Si vous utilisez un ordinateur portable, connectez son alimentation sur le secteur. Vous pouvez aussi utiliser un chargeur USB branché sur une prise murale.

Pour la mesure, la feuille d'aluminium doit être connectée entre deux broches de données. La connexion de quelque chose entre des broches de données de la sorte est plutôt rare (on connecte habituellement un câble à la masse ou au +5 V et l'autre à une broche de données).

Le code charge une broche de données, et attend jusqu'à ce que l'autre broche de données atteigne la même charge (même tension). Comme la capacitance est la faculté de tenir la charge, un gros objet comme un être humain dans les parages va affecter la capacitance et modifier le temps de chargement.

La plus petite résistance de 10 kΩ (marron-noir-orange) protège contre l'électricité statique.

La grosse résistance qui va de 1 MΩ à 50 MΩ sélectionne la sensibilité. Plus la résistance est grosse, plus elle permet de détecter loin un être humain. L'inconvénient des grandes distances de détection est que la vitesse de lecture est plus lente.

La Figure 5.20 illustre le schéma de connexion pour Arduino, et l'Exemple 5.11 liste le sketch. Câblez l'Arduino en suivant le schéma, puis chargez et exécutez le code.

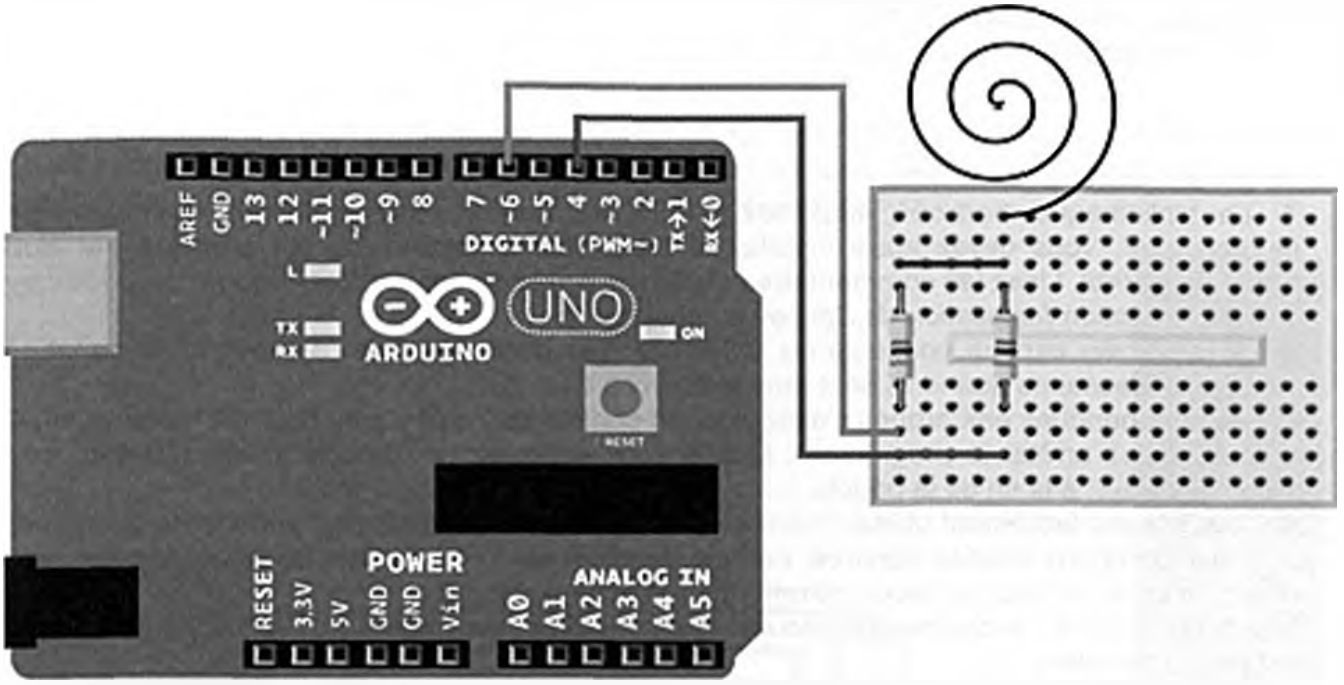


Figure 5.20 Connexion du capteur de capacitance pour Arduino

Exemple 5.11. diy_capacitive_sensor.ino

```
// diy_capacitivo_sensor.ino - mesuro le toucher
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int sendPin = 4;
const int readPin = 6;
void setup() {
   Serial.begin(115200);
   pinMode(sendPin,OUTPUT);
   pinMode(readPin,INPUT);
   digitalWrite(readPin,LOW);
}
void loop() {
  int time = 0;
  digitalWrite(sendPin,HIGH);
  while(digitalRead(readPin) == LOW) time++;
  Serial.println(time);
  digitalWrite(sendPin,LOW);
  delay(100);
}
```

Capteur de capacitance pour Raspberry Pi

La Figure 5.21 illustre le schéma de câblage pour Raspberry Pi. Câblez en suivant le schéma, et exécutez le code de l'Exemple 5.12.

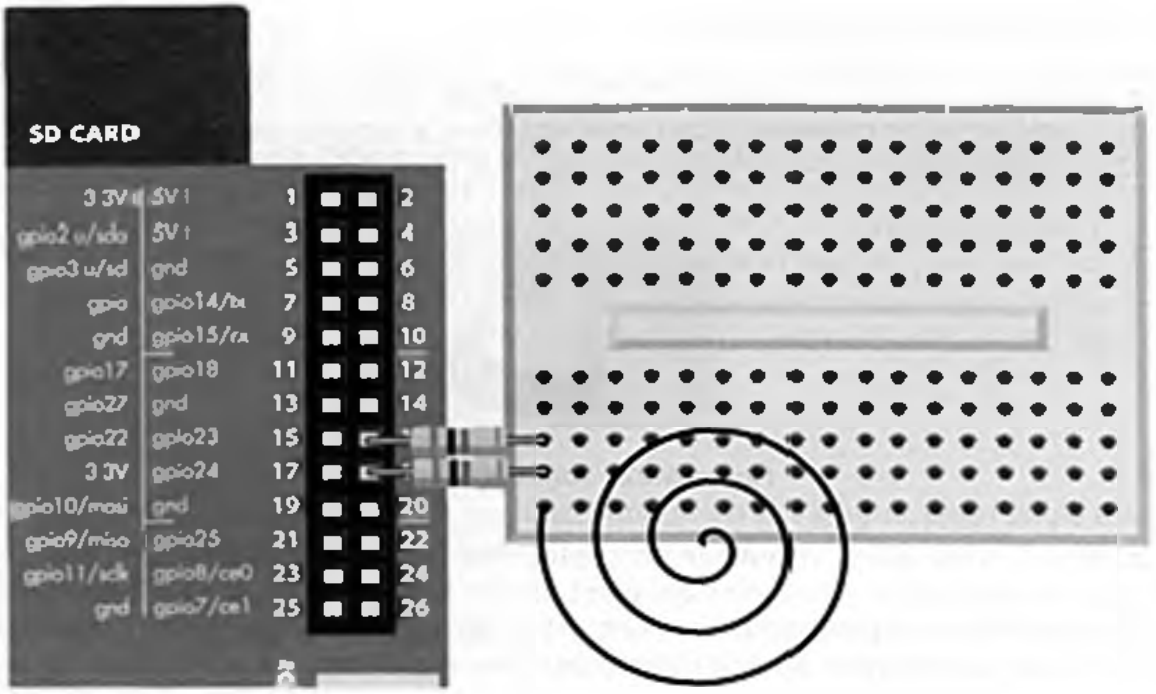


Figure 5.21 Connexion du capteur de capacitance pour Raspberry Pi

Exemple 5.12. diy_capacity_sensor_simple.py

```
# diy_capacity_sensor_simple.py - lit le toucher à partir d'un capteur de
capacitance maison
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio
def sample(count):
        sendPin = 23
        recievePin = 24
        gpio.mode(sendPin,"out")
        gpio.mode(recievePin,"in")
        gpio.write(sendPin,0)
        total = 0
        # set low
        for x in xrange(1,count):
                time.sleep(0.01)
                gpio.write(sendPin,gpio.HIGH)
                while(gpio.read(recievePin) == False):
                        total += 1
                gpio.write(sendPin,gpio.LOW)
        return total
def main():
        while True:
                touch = sample(30)
                print("Toucher : %d" % touch)
                time.sleep(0.5)
if __name__ == "__main__":
        main()
```

PROJET : SONNETTE HANTÉE

Vous vous présentez devant le comptoir d'une réception vide et il n'y a personne en vue pour vous donner la clé de votre chambre. Vous vous apprêtez à appuyer sur la sonnette du comptoir (Figure 5.22) quand elle se met à sonner sans que vous l'ayez touchée...

Figure 5.22 Sonnette hantée

Dans ce projet, nous allons combiner un capteur de toucher capacitif avec une sonnette à l'ancienne. Le résultat est une sonnette qui émet un son juste avant qu'on ne la touche. Même le bouton de la sonnette se déplace tout seul, et c'est ce qui donne à ce gadget un charmant aspect fantomatique. Vous apprendrez aussi à utiliser un nouveau composant polyvalent : le servo.

Ce que vous allez apprendre

Dans le projet de Sonnette hantée, vous allez apprendre à :

- Construire un gadget qui réagit à votre main avant que vous ne le touchiez.
- Faire bouger des choses avec des servos.
- Contrôler des servos.
- Créer un beau package pour que votre projet ressemble à un innocent objet du quotidien.

Servos

Un servo est un moteur que l'on peut précisément contrôler (Figure 5.23). On peut dire à un servo standard de tourner d'un angle spécifique, comme 90 degrés. Il existe aussi des servos exerçant une rotation complète pour lesquels on peut simplement contrôler la direction et la vitesse de rotation.

Figure 5.23 Différents servos

Si vous pensez avoir besoin d'un moteur pour un projet, commencez d'abord par envisager l'emploi d'un servo. Le servo diffère des moteurs classiques en courant continu qui sont difficiles

à contrôler et qui nécessitent des composants supplémentaires pour changer de direction. La plupart des objets qui se déplacent dans les projets de prototypage Arduino et Raspberry Pi sont réalisés avec des servos.

Un servo a trois câbles : la masse (0 V) en noir, le positif (+5 V) en rouge, et le fil de contrôle (jaune ou blanc). L'Arduino envoie un flux continu d'impulsions au fil de contrôle. La longueur des impulsions dit au servo l'amplitude de l'angle de déplacement. Le servo effectue sa rotation selon l'angle voulu et maintient sa position tant que les impulsions continuent à arriver de l'Arduino.

L'onde carrée (l'impulsion) est facile à créer en faisant alterner rapidement une broche de sortie numérique entre LOW et HIGH. Pour contrôler le servo, Arduino doit envoyer 50 impulsions par seconde (en d'autres termes, la vitesse est de 50 Hz).

```
50/s = 50 * 1/s = 50 * Hz = 50 Hz
```

Les impulsions durent entre 1 ms et 2 ms. Plus l'impulsion est longue, plus l'angle est grand. La durée d'impulsion entre l'angle minimum et maximum centre le servo. Le centre se situe souvent aux alentours de 1,5 ms.

Pour la plupart des servos, il n'est pas facile de trouver les notices techniques. Le nombre d'impulsions par seconde est cependant habituellement identique et il n'a pas à être exact.

La durée d'impulsion réelle varie entre les servos, mais il est facile de la trouver de façon expérimentale. Pour obtenir une table de correspondance de la durée d'impulsion et l'angle de rotation, consultez les valeurs du Tableau 5.1.

Durée d'impulsion en ms	Durée d'impulsion en µs	Angle	Commentaire
0,5 ms	500 µs	< –90 deg	On tente d'aller au-delà de la plage ce qui engendre un vilain bruit dans les engrenages
1 ms	1 000 µs	–90 deg	Extrême gauche
1,5 ms	1 500 µs	0 %	Centré
2 ms	2 000 µs	90 %	Extrême droite
2,5 ms	2 500 µs	> 90 deg	Au-delà de la plage, d'où un vilain bruit

Tableau 5.1 Exemples de valeurs d'angles en fonction de la durée d'impulsion

Comment trouver la plage du servo

Même si vous savez déjà que les servos utilisent des impulsions comprises entre 1 ms et 2 ms, en quoi cela va-t-il vous aider pour le servo spécifique que vous avez ? Vous pouvez trouver la plage du servo de manière expérimentale. Ce code est une sorte de « Hello World » pour servos.

Nous testons toujours la plage d'un servo quand nous en achetons un nouveau. En théorie, la durée d'impulsion doit être disponible sur le site web du fabricant, mais dans la pratique de nombreux servos n'ont pas de notice technique, ou une notice technique difficile à trouver.

Nous testons chaque durée d'impulsion significative, de la plus petite à la plus grande (Exemple 5.13). Comme le code affiche la durée d'impulsion à l'écran, on note les valeurs essentielles quelque part (voir l'exemple du Tableau 5.2). Le schéma de câblage est illustré à la Figure 5.24.

Exécutez le programme. Ouvrez le Moniteur série dans l'IDE Arduino et notez les durées d'impulsion qui s'affichent (**Outils→Moniteur série**). Définissez le Moniteur série à la même vitesse (« baud », bits/s) que celle qui est utilisée dans le code.

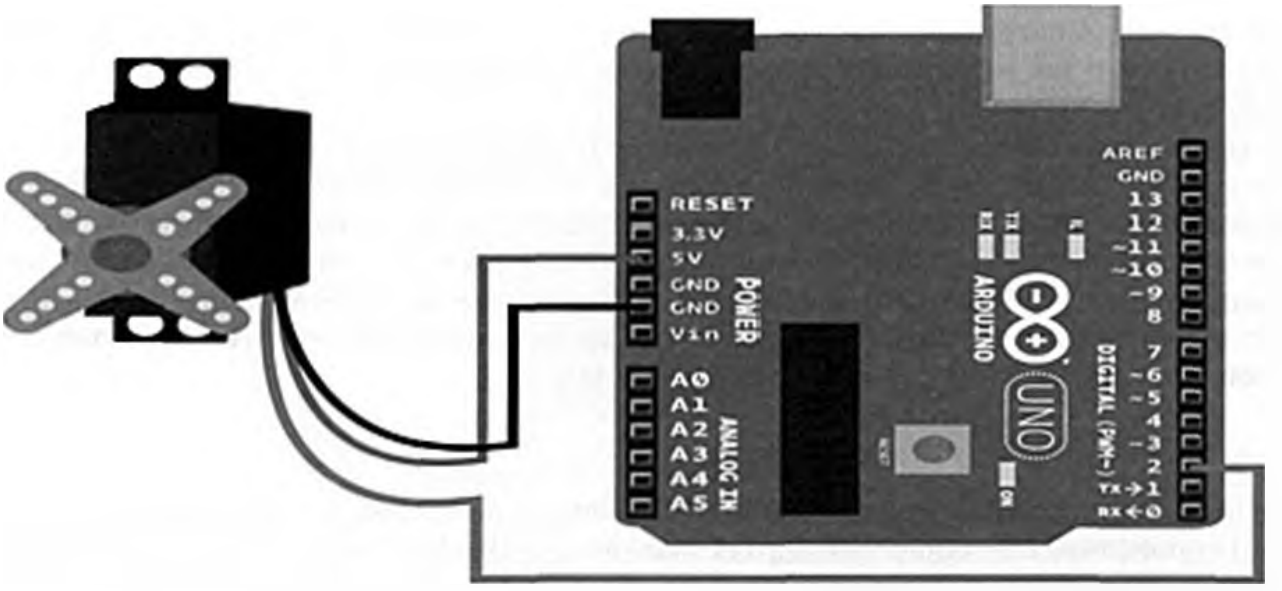


Figure 5.24 Servo connecté à la broche D2 pour le programme servo_range.ino

Durée d'impulsion en µs	Angle	Commentaire
.....	-90 deg	Extrême gauche (test avec le code)
.....	0 deg	Milieu, moyenne des deux extrêmes
.....	+90 deg	Extrême droite (test avec le code)

Tableau 5.2 Durées d'impulsion importantes pour le calibrage du servo

Au début, l'impulsion est trop courte et elle demande au servo de tourner au-delà de sa plage. Le servo ne tourne pas, mais il peut vibrer un peu et les engrenages peuvent produire un vilain bruit, ce qui n'endommage pas le servo. Quand le servo commence à tourner, notez la valeur sur le Moniteur série. Il s'agit de la durée d'impulsion pour -90 degrés qui est à l'extrême gauche.

Quand le servo s'arrête de tourner, notez la valeur de la durée d'impulsion. Il s'agit de la durée d'impulsion pour +90 degrés qui est à l'extrême droite.

La durée d'impulsion pour le milieu est la moyenne de la valeur maximale à gauche et à droite. Par exemple, si les extrêmes sont 1 ms et 2 ms, le milieu est 1,5 ms.

Certains servos ont un potentiomètre intégré pour définir le centre. Si vous avez un servo de ce genre, vous pouvez envoyer l'impulsion pour le milieu et tester avec le potentiomètre.

Exemple 5.13. servo_range.ino

```
// servo_range.ino - fait tourner le servo et affiche les valeurs sur le
port série
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int servoPin=2;
void pulseServo(int servoPin, int pulseLenUs)   // 1
{
        digitalWrite(servoPin, HIGH);   // 2
```

```
        delayMicroseconds(pulseLenUs);   // 3
        digitalWrite(servoPin, LOW);     // 4
        delay(15);       // 5
}
void setup()
{
        pinMode(servoPin, OUTPUT);
        Serial.begin(115200);
}
void loop()
{
        for (int i=500; i<=3000; i=i+2) {        // 6
                pulseServo(servoPin, i);         // 7
                Serial.println(i);
        }
}
```

1 Pour envoyer une impulsion courte, on appelle pulseServo(). La fonction n'est pas exécutée à l'endroit où elle est définie, mais uniquement quand elle est appelée ultérieurement. Comme la broche du servo est passée en paramètre, plusieurs appels à cette fonction peuvent contrôler plusieurs servos.

2 La broche est censée être d'abord à LOW, ce qui signifie qu'aucune impulsion n'est envoyée. Puis on la met à HIGH, et l'impulsion démarre.

3 Attend un tout petit moment. Comme le nom de la variable le laisse supposer, l'unité est la microseconde (µs), c'est-à-dire le millionième de seconde. Les valeurs typiques figurent dans le Tableau 5.1. Le but de ce sketch est de trouver les valeurs exactes de votre servo.

4 Met la broche à LOW, ce qui met fin aux impulsions.

5 Attend un moment. Le délai de 15 ms est très court pour un être humain, mais c'est presque 10 fois plus long que la durée d'une impulsion. Les impulsions doivent être envoyées au moins 50 fois par seconde, si bien qu'une impulsion doit être envoyée toutes les 1/50 s = 0,02 s = 20 ms. On choisit d'attendre un peu moins que cela.

6 Exécute le corps de la boucle en incrémentant les durées d'impulsion. On commence par une impulsion trop courte de 500 µs (0,5 ms), et on termine par une impulsion trop longue de 3 000 µs (3 ms). Si vous avez oublié le fonctionnement des boucles `for`, lisez l'encadré suivant.

7 Envoie une courte impulsion au servo. Comme un délai est inclus dans `pulseServo()`, vous n'avez pas à programmer un temps d'attente dans la boucle principale.

BOUCLES FOR

Une boucle for est un moyen facile de dire « exécute ce code un certain nombre de fois ». Voici la syntaxe :

```
for(initialisation; condition;
valeur_incrément)
{
  corps
}
```

Par exemple :

```
for(int i=0; i<3; i++) {
```

```
  Serial.print(i);
}
```

Quand la boucle démarre, l'initialisation a lieu une seule fois. C'est aussi l'endroit où l'on déclare la variable de la boucle.

Quand la boucle va démarrer, la condition est vérifiée. Si la condition est fausse, on sort de la boucle for. Si la condition est vraie, alors le corps de la boucle est exécuté.

Pour finir, on incrémente le compteur de la boucle, puis la condition est vérifiée à nouveau

Il est également possible de contrôler les servos avec la bibliothèque intégrée Arduino, *Servo.h*. *Servo.h* est livrée avec l'IDE Arduino et fournit une interface orientée objet pour les servos. Elle permet de contrôler les servos en spécifiant des degrés, par exemple, myservo.write(180). Cette bibliothèque pourra se révéler pratique si vous avez besoin de contrôler plusieurs servos standard. Cependant, l'utilisation de la bibliothèque Arduino Servo désactive la fonction analogWrite() sur les broches 9 et 10, et comme nous préférons conserver cette fonctionnalité nous n'utilisons pas cette bibliothèque. Consultez http://arduino.cc/en/reference/servo pour de plus amples informations.

Sonnette hantée pour Arduino

Connectez le servo et le capteur de toucher capacitif en suivant les exemples précédents (Figure 5.25). Rappelons que vous avez besoin d'utiliser un condensateur de 10-500 nF avec le QT113. La meilleure valeur pour nous a été 300 nF.

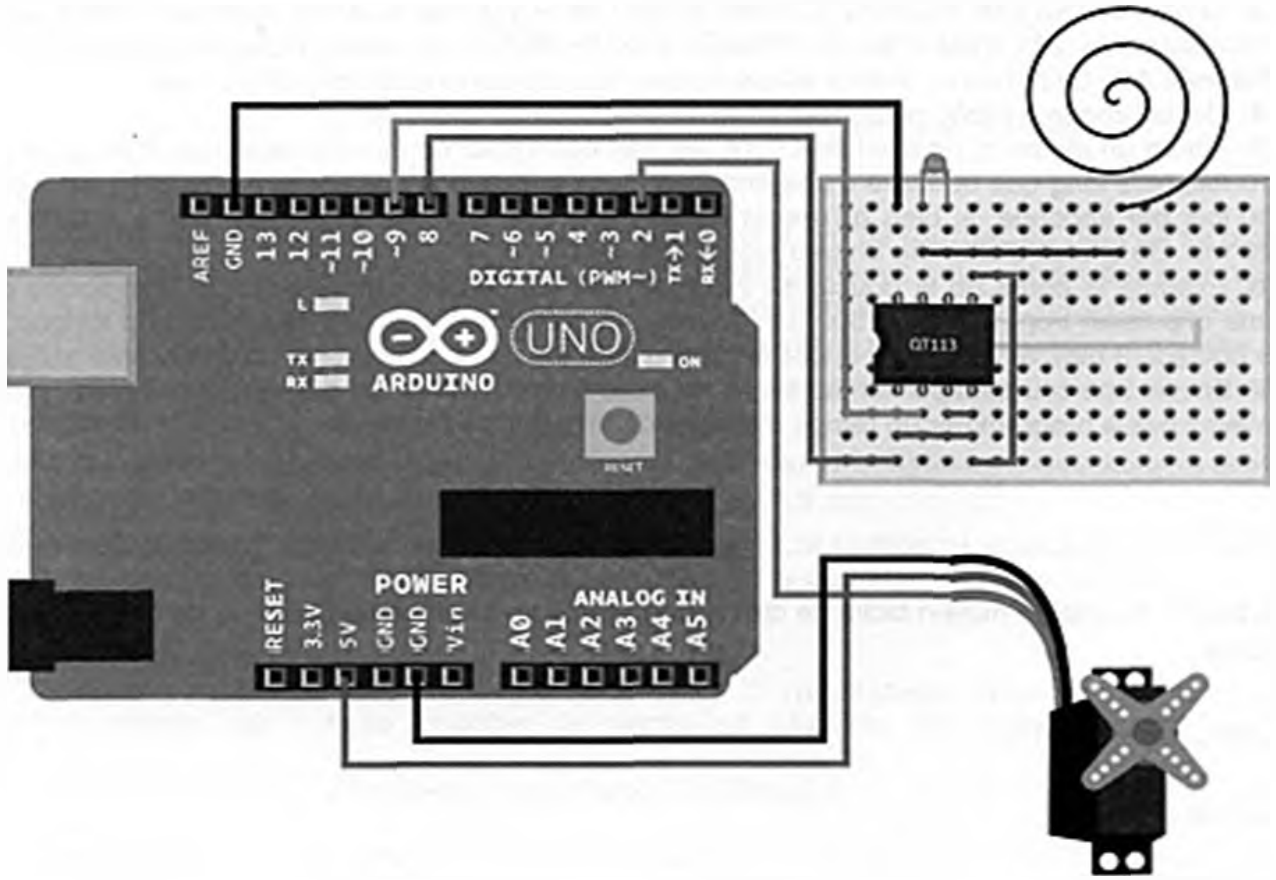


Figure 5.25 Connexions pour la sonnette hantée

Exemple 5.14. haunted_bell.ino

```
// haunted_bell.ino - la sonnette sonne juste avant qu'on ne la touche
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int servoPin=2;
int sensorPin = 9;
int sensorPowerPin = 8;
int hasRang = 0;          // 1
void pulseServo(int servoPin, int pulseLenUs)   // 2
{
  digitalWrite(servoPin, HIGH);
  delayMicroseconds(pulseLenUs);
  digitalWrite(servoPin, LOW);
  delay(20);
}
void cling()     // 3
{
    for (int i=0; i<=3; i++) {  // 4
      pulseServo(servoPin, 2000);
    }
    for (int i=0; i<=100; i++) {
      pulseServo(servoPin, 1000);
    }
}
void setup()
{
  pinMode(servoPin, OUTPUT);
  pinMode(sensorPowerPin, OUTPUT);
  digitalWrite(sensorPowerPin,HIGH);
  pinMode(sensorPin,INPUT);
}
void loop()
{
  int touch = digitalRead(sensorPin);   // 5
  if(touch == HIGH) {   // 6
    hasRang = 0;
  }
  if(touch == LOW && hasRang == 0) {    // 7
    cling();    // 8
    hasRang = 1;          // 9
    digitalWrite(sensorPowerPin,LOW);
    delay(1);
    digitalWrite(sensorPowerPin,HIGH);
  }
  delay(100);
}
```

1 La variable `hasRang` sera à 1 si la sonnette a sonné. Cela permet de s'assurer que la sonnette ne sonne qu'une fois (nous avons utilisé l'entier 1 pour true et 0 pour false afin d'éviter

l'introduction d'une logique booléenne lourde, mais si le cœur vous en dit, vous pouvez modifier le programme pour utiliser des booléens).

2 Le contrôle du servo est expliqué dans l'Exemple 5.13.

3 Fonction pour faire sonner la sonnette une fois. Elle est dans une fonction séparée pour rendre la fonction loop() principale plus facile à lire, et parce que c'est une chose simple à faire.

4 Pour donner au servo du temps pour se déplacer et attendre la sonnette, vous devez envoyer plusieurs impulsions. Comme une fonction pulseServo() prend approximativement 20 ms, 100 itérations durent 2 secondes. Une fonction ring() dure donc à peu près deux secondes.

5 SensorPin est à LOW quand il y a un toucher.

6 S'il n'y a pas de toucher (HIGH), on réinitialise hasRang. La sonnette va à présent sonner à nouveau si une main s'approche.

7 Si une main s'approche *et* (&&) qu'elle n'a pas encore sonné pour ce toucher...

8 ...on fait sonner une fois.

9 Rappelle que la sonnette a déjà sonné, afin d'éviter de sonner deux fois pour la même main qui s'approche.

Fixation du servo à la sonnette

Utilisez de la colle à chaud pour fixer le servo à l'intérieur de la sonnette (Figure 5.26). Avant de coller, assurez-vous que le bras du servo pousse la partie mobile à l'intérieur de la sonnette de telle sorte qu'il produit un son puissant. Le mouvement du servo doit aussi permettre au bouton de la sonnette de se déplacer naturellement. Il faudra peut-être plusieurs essais pour fixer le servo au bon endroit. Votre sonnette hantée est maintenant prête à effrayer d'innocentes victimes qui ne se doutent de rien.

Figure 5.26 Servo à l'intérieur de la sonnette

Vous avez vu à présent plusieurs moyens de contrôler le toucher ou l'approche. Vous avez joué avec des boutons, des microrupteurs, des interrupteurs tactiles et même utilisé des capteurs tactiles sans qu'on les touche. Une fois que vous aurez bien effrayé vos amis avec votre sonnette, vous pouvez commencer à appliquer ces compétences à vos propres projets. Pensez à tous les appareils électroniques qui vous entourent : la plupart sont tactiles, d'une manière ou d'une autre.

6 Détecter les mouvements

Quand vous vous promenez le soir dans votre jardin et qu'une lumière s'allume automatiquement à votre passage, c'est qu'un capteur a détecté la chaleur de votre corps en mouvement.

Quand vous jouez avec votre console de jeux, les capteurs de votre joystick convertissent les mouvements en signaux électriques.

Quand vous baissez le volume de votre radio, le mouvement de rotation du bouton de réglage du son agit comme un potentiomètre.

EXPÉRIENCE : DÉTECTER LE SENS AVEC UN CAPTEUR D'INCLINAISON

Si vous décidez de construire un flipper, il est préférable de détecter les mouvements excessifs de la machine afin de la faire tilter.

On utilisera aussi un capteur d'inclinaison comme alarme anti-intrusion.

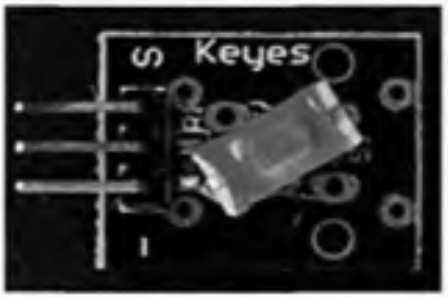


Figure 6.1 Capteur d'inclinaison

Capteur d'inclinaison pour Arduino

La Figure 6.2 illustre le circuit d'un capteur d'inclinaison pour Arduino. Montez le circuit et exécutez le code de l'Exemple 6.1.

Exemple 6.1 tilt_sensor.ino

```
// tilt_sensor.ino - détecte le tilt et affiche sur le port série
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int tiltPin = 8;
int tilted = -1;
void setup() {
```

```
  Serial.begin(115200);
  pinMode(tiltPin, INPUT);
  digitalWrite(tiltPin, HIGH);
}
void loop() {
  tilted = digitalRead(tiltPin);          // 1
  if(tilted == 0) {
    Serial.println("Le capteur a tilté");
  } else {
    Serial.println("Le capteur n'a pas tilté");
  }
  delay(100);
}
```

1 Il s'agit d'un simple interrupteur numérique.

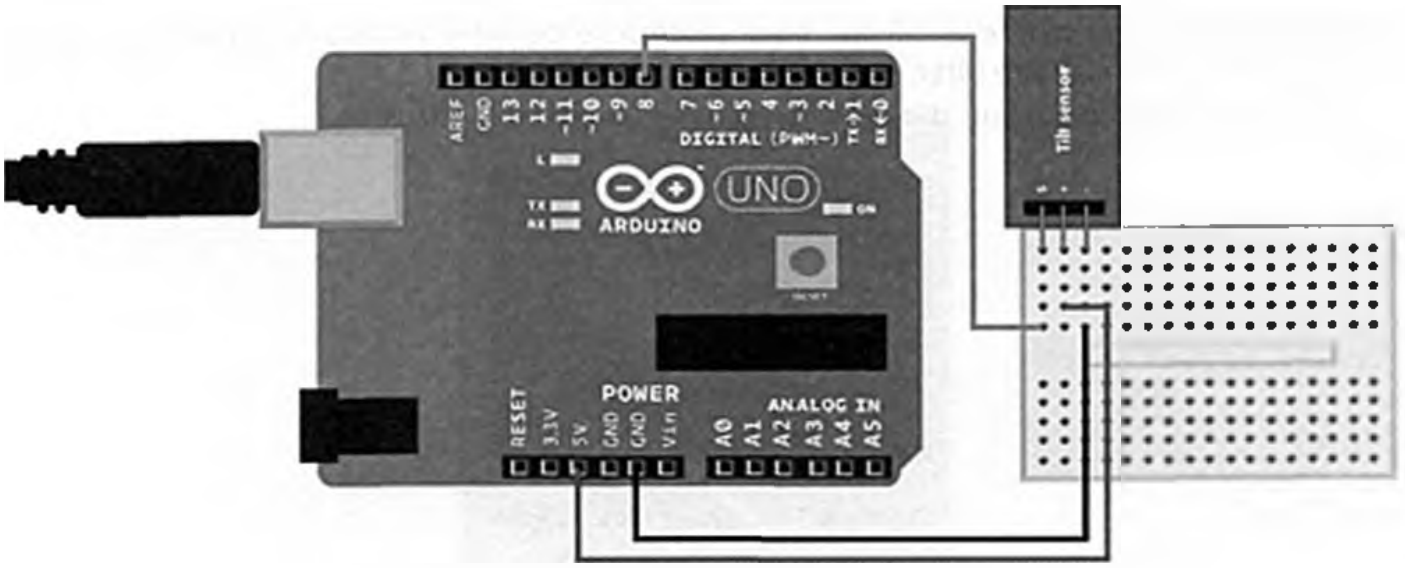


Figure 6.2 Circuit de capteur d'inclinaison pour Arduino.

Capteur d'inclinaison pour Raspberry Pi

La Figure 6.3 illustre les connexions pour le Raspberry Pi. Effectuez le montage et exécutez le code de l'Exemple 6.2.

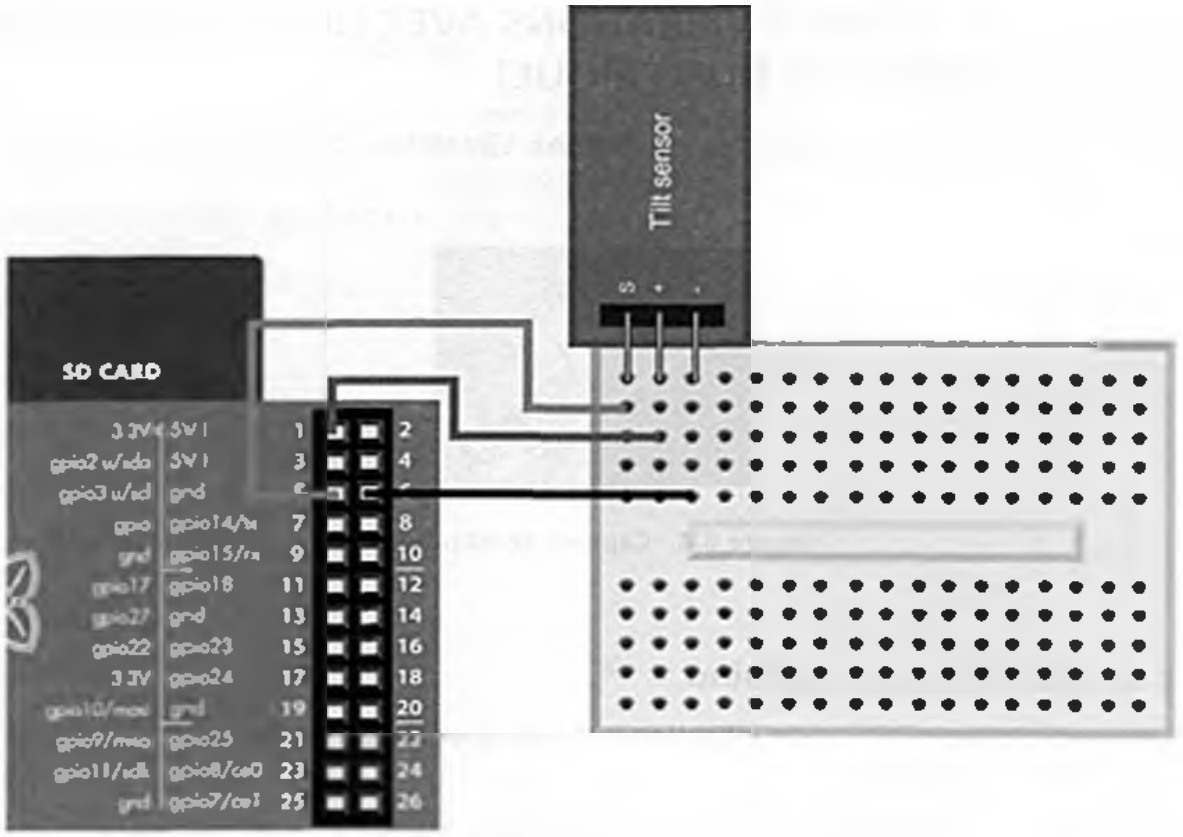


Figure 6.3 Circuit de capteur d'inclinaison pour Raspberry Pi.

Exemple 6.2 tilt_sensor.py

```
# tilt_sensor.py - avertit si le capteur a tilté
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio
def main():
        tiltpin = 3     # pull-up interne # 1
        gpio.mode(tiltpin, "in")
        while (True):
                isNotTilted = gpio.read(tiltpin) # 2
                if(isNotTilted == gpio.LOW):
                        print "Le capteur a tilté"
                else:
                        print "Le capteur n'a pas tilté"
                time.sleep(0.3) # secondes
if __name__ == "__main__":
        main()
```

1 Le Raspberry Pi a des résistances internes pull-up sur gpio2 et gpio3 qui sont activées en permanence.

2 Un capteur d'inclinaison est un simple interrupteur numérique.

EXPÉRIENCE : DE BONNES VIBRATIONS AVEC UNE INTERRUPTION (CAPTEUR DE VIBRATION NUMÉRIQUE)

Un capteur de vibration peut détecter de petites vibrations, comme des secousses du sol (Figure 6.4).

Figure 6.4 Capteur de vibration.

Capteur de vibration pour Arduino

Le signal envoyé par un capteur de vibration est très court si bien que vous devez utiliser une interruption pour l'intercepter.

Pour la plupart des capteurs, il suffit de les interroger, ce qui signifie que l'on vérifie l'état de la broche numérique, que l'on attend un peu et que l'on vérifie à nouveau.

Avec une interruption, vous pouvez dire à Arduino d'exécuter une fonction quand un événement se produit. Les interruptions rendent plus difficile le suivi du flux du code, mais elles permettent d'intercepter des événements très courts, comme une broche dont l'état change très rapidement.

La Figure 6.5 illustre le circuit du capteur de vibration. Câblez-le en suivant le schéma et exécutez le code de l'Exemple 6.3.

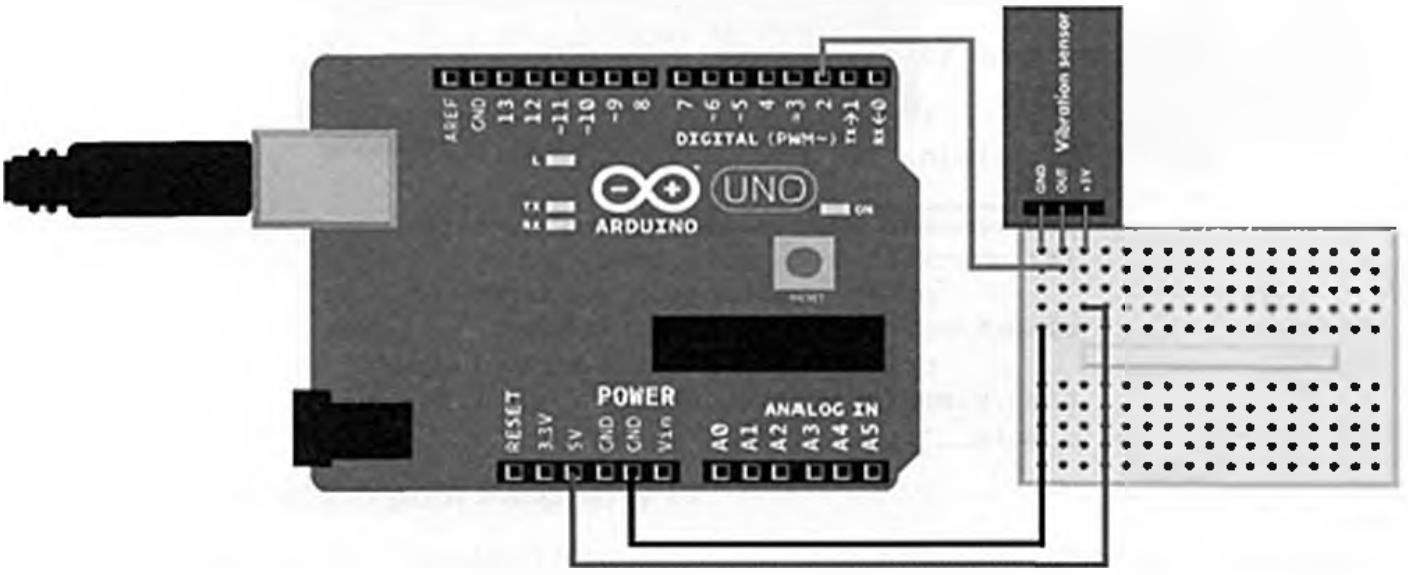


Figure 6.5 Circuit du capteur de vibration pour Arduino.

Exemple 6-3. vibration_sensor.ino

```
// vibration_sensor.ino - détecte la vibration à l'aide d'une interruption
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int sensorPin = 0; //UNO,Mega broche 2, Leonardo broche 3
volatile int sensorState = -1;
void setup() {
  Serial.begin(115200);
  attachInterrupt(sensorPin, sensorTriggered, RISING);  // 1
}
void loop() {
  if(sensorState == 1) {          // 2
    Serial.println("Vibrations détectées");
    delay(1);   // ms
    sensorState = 0;
  }
  delay(10);
}
void sensorTriggered() {          // 3
  sensorState = 1;  // 4
}
```

1 Dit à l'Arduino d'appeler sensorTriggered() quand sensorPin passe de LOW à HIGH. On appelle cela un callback.
Notez que le nom de la fonction dans le callback n'a pas de parenthèses. Sans parenthèses, un nom de fonction est un pointeur vers la fonction (alors qu'avec des parenthèses la fonction ne serait appelée qu'une seule fois et retournerait sa valeur). Après tout, attachInterrupt() a besoin de connaître la fonction, et elle n'est pas prête à l'exécuter pour l'instant.
2 La boucle principale peut tracer sa route aussi lentement qu'elle le souhaite. Elle vérifie la valeur d'une variable globale.
3 Quand l'interruption est déclenchée, sensorTriggered() est appelée. L'interruption réagit très rapidement.
4 La fonction sensorTriggered() définit simplement une variable globale qui peut être lue à loisir par la boucle principale.

Capteur de vibration pour Raspberry Pi

La Figure 6.6 illustre le schéma de câblage de la version Raspberry Pi du circuit de ce capteur. Montez-le comme cela est indiqué à la Figure 6.6 et exécutez l'Exemple 6.4.

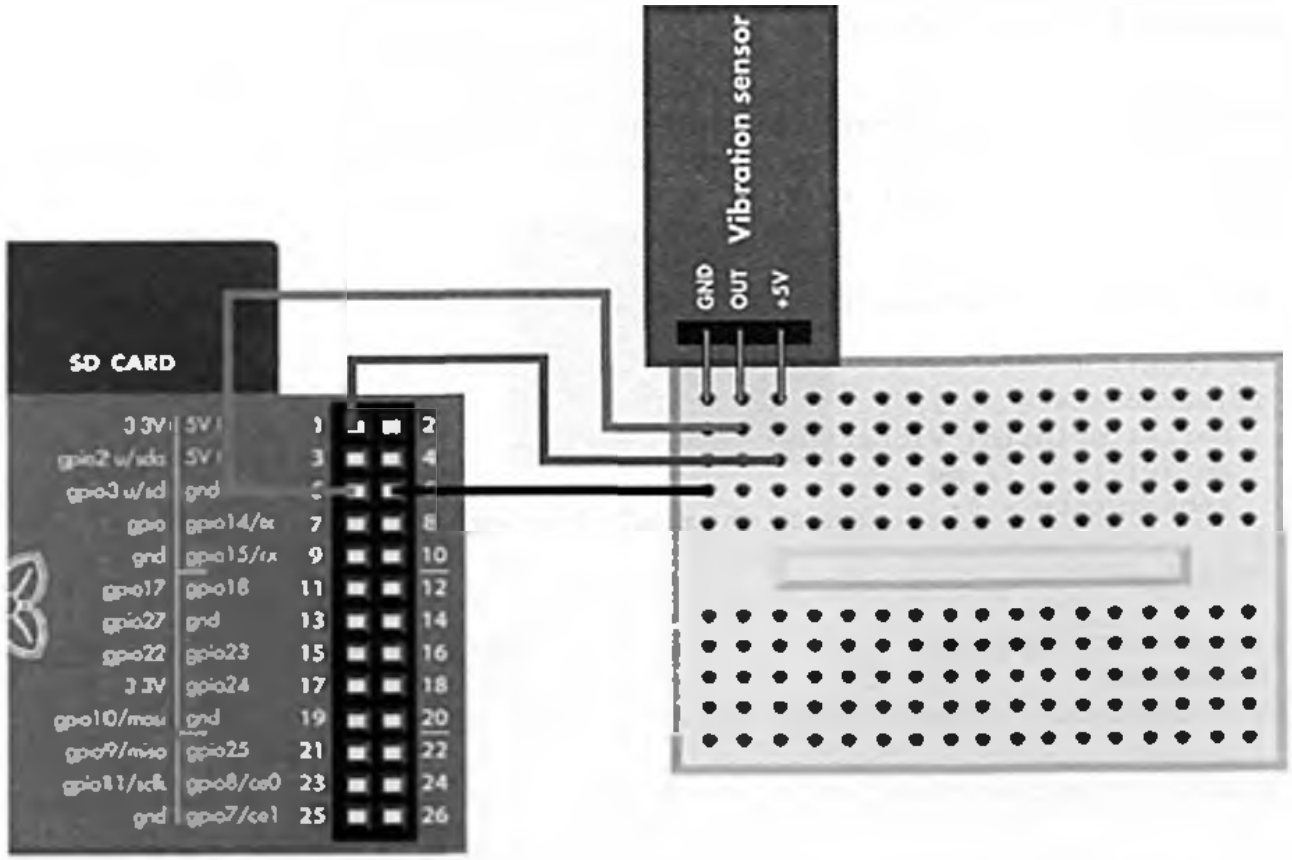


Figure 6.6 Circuit d'un capteur de vibration pour Raspberry Pi.

Exemple 6.4 vibration_sensor.py

```
# vibration_sensor.py - détecte la vibration
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio      # 1
def main():
        vibPin = 3
        gpio.mode(vibPin, "in")
        while (True):
                vibrationInput = gpio.read(vibPin)   # 2
                if(vibrationInput == gpio.LOW):
                        print "Vibration"
                        time.sleep(5)    # 3
                time.sleep(0.01) # secondes    # 4
if __name__ == "__main__":
        main()
```

1 Comme pour les exemples précédents, cette bibliothèque doit être dans le même répertoire que *vibration_sensor.py*.

2 Le capteur de vibration est utilisé comme un interrupteur numérique. Vous pouvez le considérer comme un bouton.
3 Si la vibration a été détectée, on ne l'affiche qu'une seule fois au lieu de remplir l'écran avec la chaîne « Vibration ».
4 On utilise un délai très court de 10 ms, si bien que vibPin est interrogé 100 fois par seconde (100 Hz).

Le code Raspberry Pi ne peut pas détecter des vibrations aussi courtes que l'Arduino. Le code n'utilise pas d'interruption car cela compliquerait le code qui nécessiterait quand même une boucle rapide et ne pourrait toujours pas concurrencer l'Arduino. Si vous avez besoin d'une détection de vibration très précise avec un Raspberry Pi, vous pouvez connecter un Arduino à un Raspberry Pi (voir la section *Communiquer avec Arduino à partir de RaspBerry Pi* du chapitre 11).

EXPÉRIENCE : TOURNEZ LE BOUTON

Un encodeur rotatif mesure la rotation (voir la Figure 6.7). Un encodeur rotatif **absolu** donne la position (par exemple 83 degrés), et un encodeur rotatif **relatif** indique le changement (on a tourné de 23 degrés par rapport à la position précédente).

L'encodeur rotatif que l'on utilise ici est un encodeur relatif. Quand on tourne le bouton, l'encodeur envoie des impulsions d'horloge sur la broche d'horloge. Sur le front montant (l'horloge passe de LOW à HIGH), une impulsion de données arrive sur la broche de données ; une autre arrive sur le front descendant (on passe de HIGH à LOW).

Si l'impulsion de données est à HIGH, c'est qu'on tourne à droite (dans le sens des aiguilles d'une montre, direction négative). Si l'impulsion de données est à LOW, c'est que l'on tourne à gauche.

Figure 6.7 Encodeur rotatif.

Encodeur rotatif pour Arduino

Ce code utilise des interruptions si bien qu'il vous semblera peut-être différent des programmes Arduino que vous avez déjà rencontrés. Câblez le circuit selon le schéma de la Figure 6.8, et exécutez le sketch de l'Exemple 6.5.

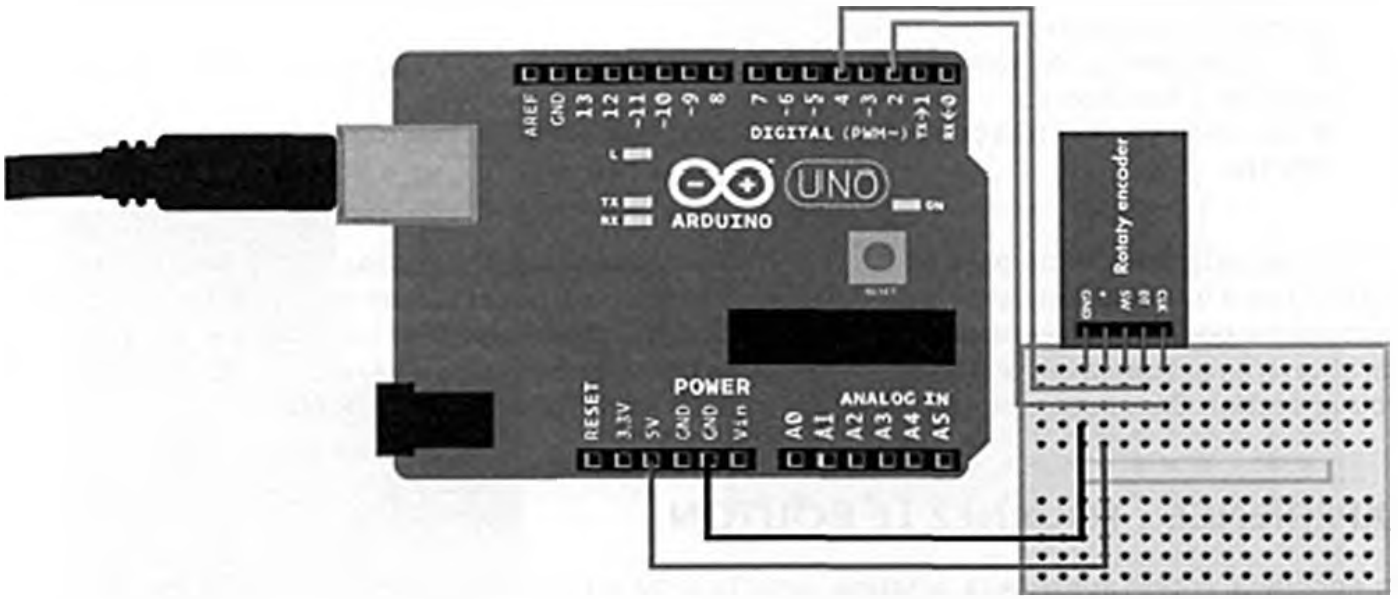


Figure 6.8 Circuit d'un encodeur rotatif pour Arduino.

Exemple 6.5 rotary_encoder.ino

```
// rotary_encoder.ino - affiche la position de l'encodeur
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int clkPin = 2;
const int dtPin = 4;
volatile unsigned int encoderPos = 0;    // 1
void setup()
{
  Serial.begin(115200);
  pinMode(clkPin, INPUT);
  digitalWrite(clkPin, HIGH); // pull-up // 2
  pinMode(dtPin, INPUT);
  digitalWrite(dtPin, HIGH); // pull-up
  attachInterrupt(0, processEncoder, CHANGE);    // 3
}
void loop()
{
  Serial.println(encoderPos);
  delay(100);
}
void processEncoder()    // 4
{
  if(digitalRead(clkPin) == digitalRead(dtPin)) // 5
  {
    encoderPos++;          // tourne à droite
  } else {
```

```
        encoderPos--;
    }
}
```

1 EncoderPos est définie en tant que volatile, car elle est modifiée par la fonction d'interruption. Le mot-clé **volatile** permet au compilateur Arduino de savoir que la valeur peut changer « dans le dos » de la boucle principale du sketch Arduino.
2 Le fait d'écrire HIGH sur une broche d'entrée la connecte au +5 V *via* une résistance *pull-up* de 20 kΩ. Cela est fait pour éviter l'instabilité de la broche.
3 Chaque fois qu'il y a un changement (front montant ou descendant), cela appelle la fonction processEncoder. Avec un Arduino Uno, l'interruption 0 contrôle la broche numérique 2. La fonction callback processEncoder n'a pas de parenthèses car ce n'est pas nécessaire pour qu'elle s'exécute pour le moment. Elle s'exécutera plus tard, quand elle sera appelée par l'interruption (cela se produit en arrière-plan).
4 Tout le reste est mis en attente pendant l'exécution de la la fonction d'interruption.
5 Quand dtPin est dans le même état que la broche d'horloge, cela indique que l'on a tourné le bouton à droite. Dans le cas contraire, on a tourné à gauche.

Encodeur rotatif pour Raspberry Pi

La Figure 6.9 illustre le câblage de l'utilisation d'un encodeur rotatif avec un Raspberry Pi. Montez le circuit puis exécutez le programme de l'Exemple 6.6.

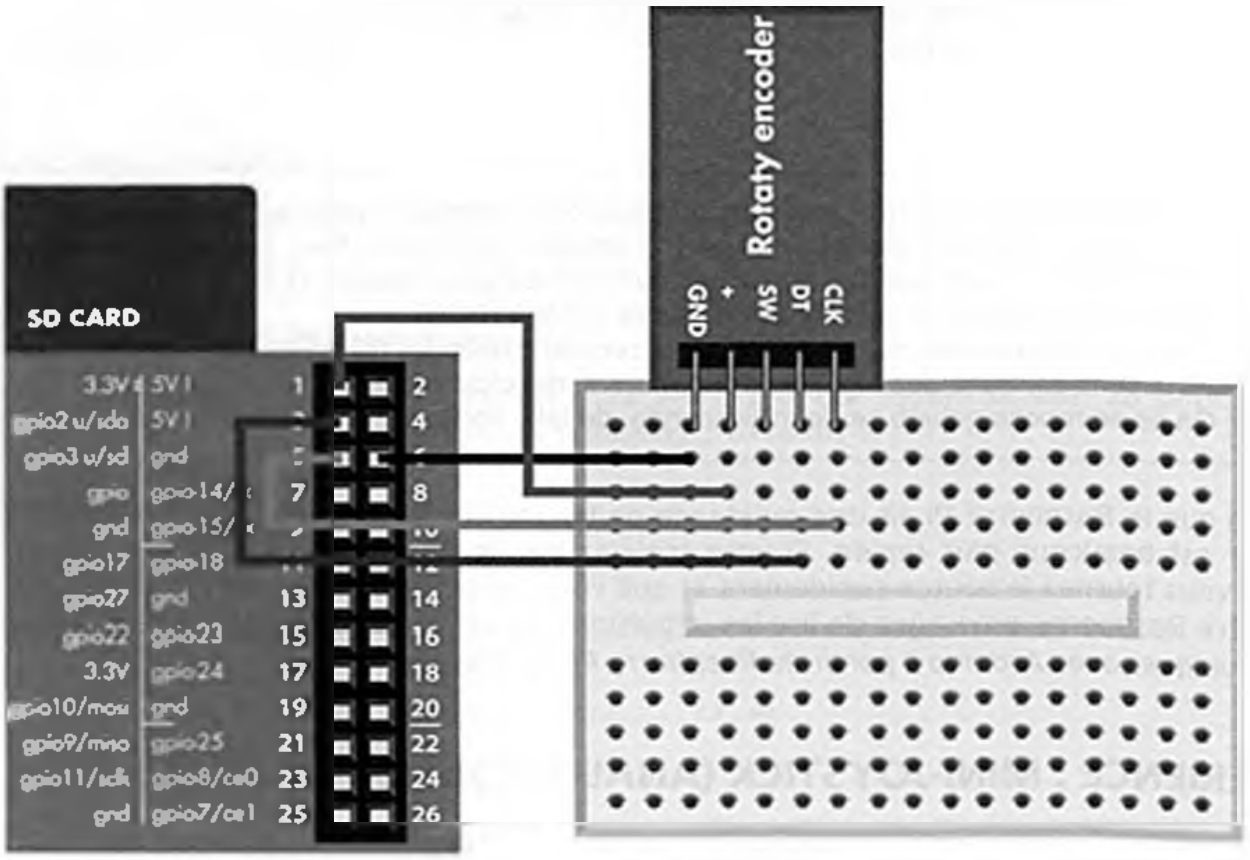


Figure 6.9 Circuit d'un encodeur rotatif pour Raspberry Pi.

Exemple 6.6. rotary_encoder.py

```
# rotary_encoder.py - lit un encodeur rotatif
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio      # 1
def main():
        encoderClk = 3
        encoderDt = 2
        gpio.mode(encoderClk, "in")
        gpio.mode(encoderDt, "in")
        encoderLast = gpio.LOW
        encoderPos = 0
        lastEncoderPos = 0
        while True:
                clk = gpio.read(encoderClk)
                if encoderLast == gpio.LOW and clk == gpio.HIGH: # 2
                        if(gpio.read(encoderDt) == gpio.LOW):    # 3
                                encoderPos += 1
                        else:
                                encoderPos -= 1
                encoderLast = clk
                if encoderPos != lastEncoderPos:
                        print("Position encodeur %d" % encoderPos)
                lastEncoderPos = encoderPos
                time.sleep(0.001) # s    # 4
if __name__ == "__main__":
        main()
```

1 Assurez-vous qu'il y a une copie de la bibliothèque *botbook_gpio.py* dans le même répertoire que ce programme. Reportez-vous à la section *GPIO sans Root* du chapitre 1 pour des informations sur la configuration de l'accès du GPIO de votre Raspberry Pi.
2 Si un front montant est détecté (on passe de LOW à HIGH)...
3 ... quand les données sont à LOW sur la broche d'horloge, l'encodeur est tourné à gauche (dans le sens contraire des aiguilles d'une montre, direction positive).
4 On ne se repose qu'une seule milliseconde, de telle sorte que l'on ne rate aucun clic.

Bien que le Raspberry Pi ait une puissance de traitement supérieure à celle de l'Arduino, ce dernier est beaucoup plus rapide. Si vous trouvez que le Raspberry Pi rate trop d'impulsions quand vous tournez le bouton rapidement et que vous exécutez beaucoup d'autres programmes sur votre Raspberry, envisagez de lire les impulsions *via* un Arduino. Reportez-vous à la section *Communiquer avec Arduino à partir de RaspBerry Pi* du chapitre 11.

EXPÉRIENCE : MINI-JOYSTICK (ANALOGIQUE DEUX AXES)

Si vous avez déjà joué sur une console de jeux vidéo, vous avez utilisé un **joystick**. Au début, on se servait d'un gros joystick qu'on empoignait fermement. Les consoles de jeux modernes comme la Xbox, la PlayStation, ou la Ouya emploient plusieurs mini-joysticks.

Figure 6.10 Mini-joystick deux axes.

En général, un joystick a deux potentiomètres (pot ou potar en abrégé), qui sont des résistances variables. En déplaçant le joystick le long de l'axe vertical (*y*), on modifie la résistance d'un potentiomètre et on change la résistance de l'autre potentiomètre en déplaçant le joystick sur l'axe horizontal (*x*).

Dans de nombreux joysticks, les potentiomètres ont trois fils : un fil est relié à la masse, l'autre au +5 V, et le dernier du milieu procure une tension variable entre 0 V et +5 V. Dans cette configuration à trois fils, il n'y a pas besoin de résistance pull-up ou pull-down.

> Les téléphones mobiles et les consoles Wii peuvent utiliser des accéléromètres à la place des joysticks. Le comportement de l'appareil est évalué par rapport à la gravitation, ce qui permet de contrôler les jeux (voir l'*Expérience : détournement d'un Nunchuk (avec I2C)* du chapitre 8). La gravitation a le même effet que l'accélération. Pour obtenir des mesures précises du comportement, reportez-vous au chapitre 8.

Joystick pour Arduino

La Figure 6.11 illustre le circuit utilisant un Arduino avec un joystick. Montez-le et exécutez le sketch de l'Exemple 6.7.

Exemple 6.7 ky_023_xyjoystick.ino

```
// ky_023_xyjoystick.ino - affiche la position du joystick sur le port série
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int VRxPin = 0;    // 1
const int VRyPin = 1;
const int SwButtonPin = 8;
int button = -1; // LOW ou HIGH // 2
int x = -1; // 0..1023
int y = -1; // 0..1023
void readJoystick() { // 3
  button = digitalRead(SwButtonPin);     // 4
  x = analogRead(VRxPin);
  y = analogRead(VRyPin);
}
void setup()  {
  pinMode(SwButtonPin, INPUT);
  digitalWrite(SwButtonPin, HIGH); // résistance pull-up // 5
```

```
  Serial.begin(115200);
}
void loop()  {
  readJoystick();        // 6
  Serial.print("X: ");
  Serial.print(x);
  Serial.print(" Y: ");
  Serial.print(y);
  Serial.print(" Bouton : ");
  Serial.println(button);
  delay(10);
}
```

1 Stocke les numéros des broches dans des constantes globales : une résistance variable pour *x*, une résistance variable pour *y*, et un bouton.
2 Ces variables globales stockent l'état du bouton et l'inclinaison du joystick le long des axes *x* et *y*. On initialise ces variables à des valeurs impossibles (des valeurs qui ne seront pas générées par des appels à analogRead ou digitalRead) afin de faciliter le débogage. La plage des valeurs attendues est indiquée dans les commentaires.
3 readJoystick() ne retourne pas de valeur (il s'agit d'un type void). En C++, langage sur lequel est basé Arduino, les fonctions ne peuvent pas facilement retourner plusieurs valeurs. Au lieu de cela, readJoystick() met à jour les variables globales.
4 L'état du bouton est stocké dans la variable globale button.
5 Active la résistance interne pull-up pour éviter l'instabilité de la broche.
6 Met à jour les variables globales qui stockent l'état du joystick.

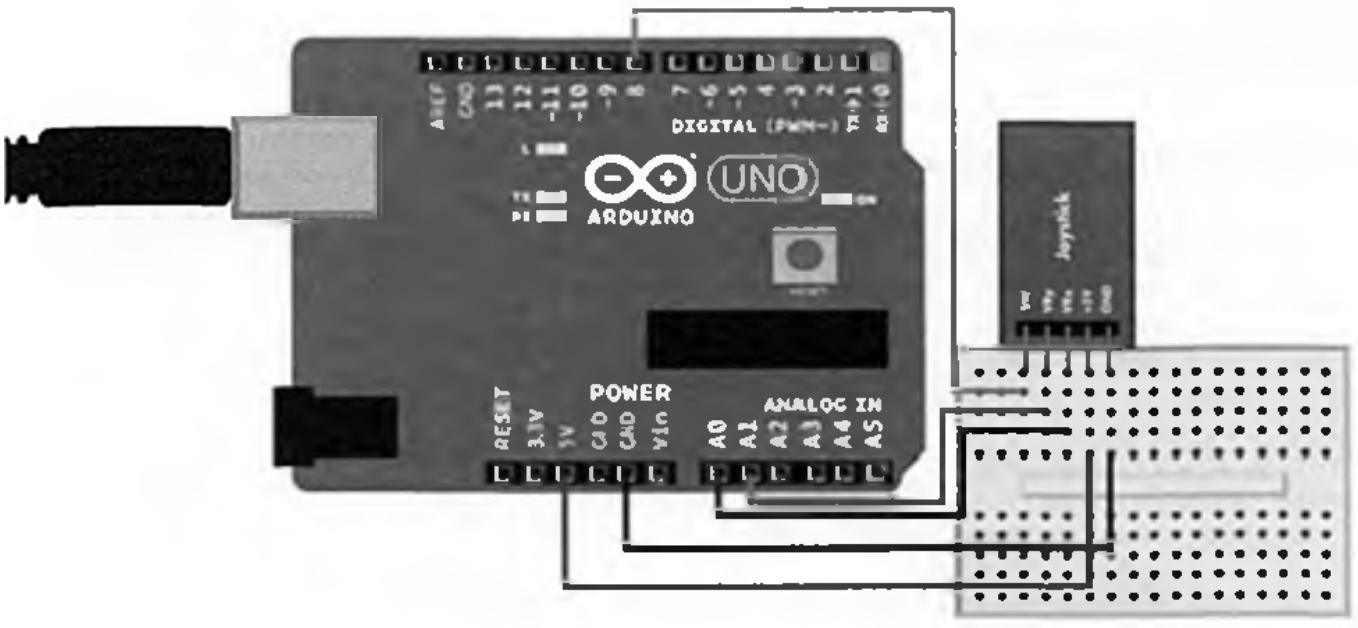


Figure 6.11 Circuit de joystick pour Arduino.

Joystick pour Raspberry Pi

La Figure 6.12 illustre le circuit d'un joystick avec un Raspberry Pi. Montez l'ensemble selon le schéma, puis exécutez le programme de l'Exemple 6.8.

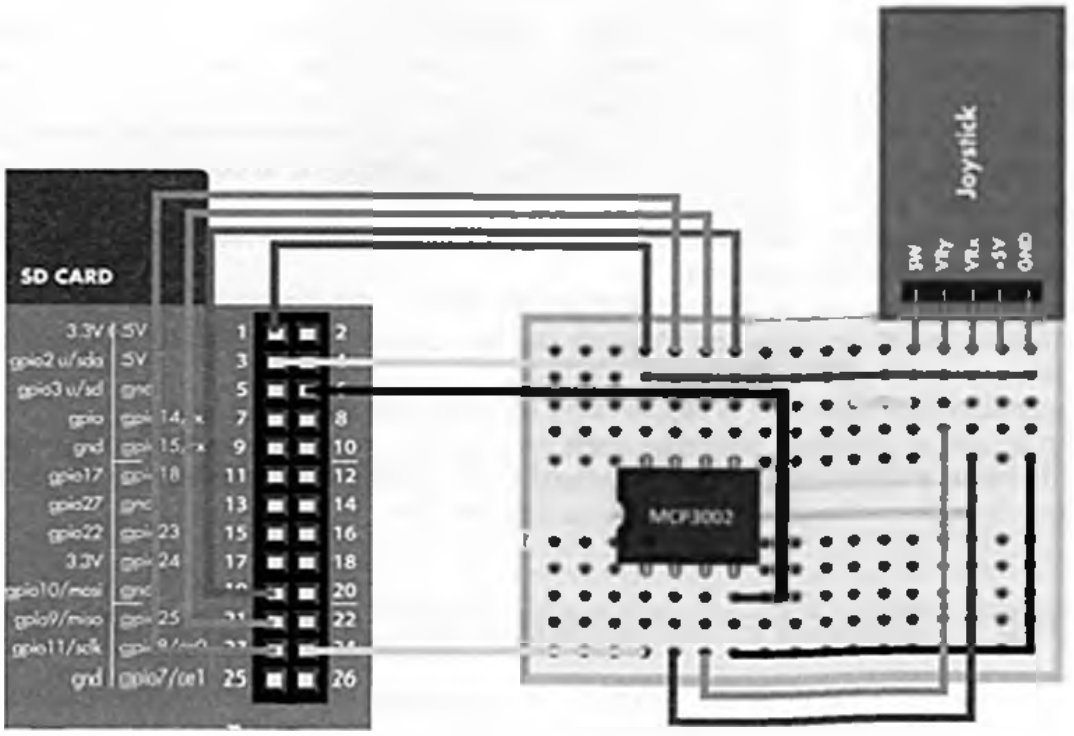


Figure 6.12 Circuit de joystick pour Raspberry Pi.

Exemple 6.8 xy_joystick.py

```
# xy_joystick.py - affiche l'inclinaison et l'état d'un joystick KY 023
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002  as mcp   # 1
import botbook_gpio as gpio      # 2
def readX():     # 3
        return mcp.readAnalog(0, 0)
def readY():
        return mcp.readAnalog(0, 1)      # 4
def readButton():
        buttonPin = 25
        gpio.modo(buttonPin, "in")
        return gpio.read(buttonPin)
def main():
        while True:      # 5
                xAxel = readX() # 6
                yAxel = readY()
                button = readButton()
                print("X: %i, Y: %i, Bouton : %r" % (xAxel, yAxel, button))
# 7
```

```
                time.sleep(0.5)
if __name__ == "__main__":
        main()
```

1 Importe la bibliothèque du convertisseur analogique-numérique MCP3002. La bibliothèque *botbook_mcp3002.py* doit être dans le même répertoire que *xy_joystick.py*. Vous devez aussi installer la bibliothèque spidev, qui est importée par botbook_mcp3002. Lisez les commentaires au début de *botbook_mcp3002/botbook_mcp3002.py* ou *Installation de SpiDev* au chapitre 3.
2 On utilise un espace de noms plus concis pour ne pas alourdir le code. Le mot-clé as permet d'écrire gpio.read() au lieu de botbook_gpio.read().
3 On écrit des fonctions pour chaque tâche. L'objet de readX() (lecture de l'axe *x*) deviendra évident dans le programme principal.
4 Mesure la tension sur le second canal (numéro 1).
5 Continue à tourner tant que l'on n'appuie pas sur Ctrl+C pour arrêter le programme (ou que l'on n'éteint pas le Raspberry Pi).
6 Lit l'inclinaison sur l'axe *x* et la stocke dans une nouvelle variable xAxel.
7 Quand on utilise plusieurs variables dans une chaîne, les variables doivent former un **tuple** (groupe de valeurs séparées par des virgules). Vous pouvez vous représenter un tuple (1, 2, 3) comme une liste [1, 2, 3] que vous ne pouvez pas modifier.

EXPÉRIENCE : DÉVELOPPEMENT DURABLE AVEC UNE MANETTE

Si vous avez une manette d'une vieille console de jeux (Xbox, PlayStation, etc.) qui traîne quelque part, vous pouvez recycler deux capteurs pour les utiliser avec votre Arduino ou votre Raspberry Pi (Figure 6.13). L'ouverture d'une manette est assez facile, mais vous aurez besoin d'un fer à souder pour enlever les composants du circuit.

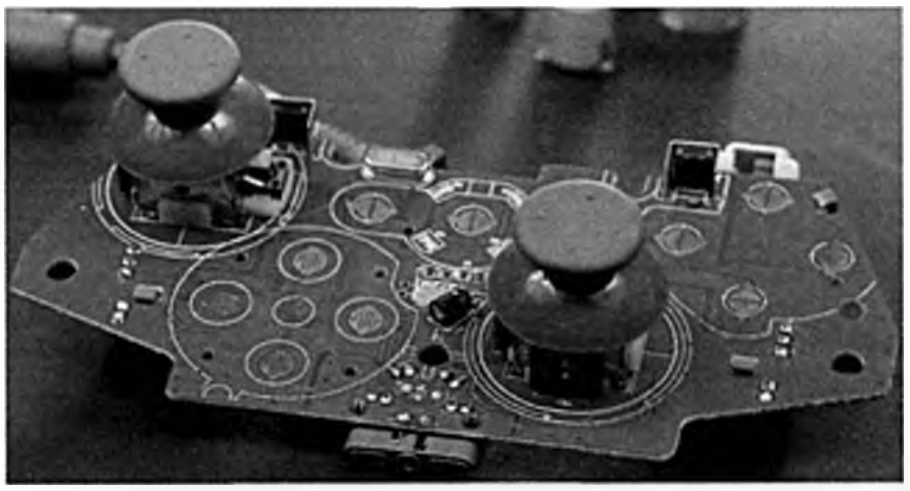


Figure 6.13 Mini-joysticks d'une manette Xbox.

Pour enlever les composants, vous pouvez aussi utiliser avec profit les outils suivants : pompe à dessouder, tresse à dessouder, voire stylo flux de réparation.

De nombreuses manettes ont aussi un système simple de retour de force. Par exemple, si un joueur subit des dégâts, le retour de force fait vibrer la manette (grâce à des moteurs de vibration

excentriques à courant continu qui bougent quand ils sont alimentés). Enlevez-les aussi pour leur donner une nouvelle vie dans votre propre gadget à retour de force (Figure 6.14).

Figure 6.14 Récupération de moteurs de vibration.

EXPÉRIENCE : ALARME ANTIVOL (CAPTEUR INFRAROUGE PASSIF)

Un capteur passif infrarouge (abrégé en anglais par PIR pour *Passive InfraRed*) est probablement l'alarme antivol la plus courante. Tous les objets chauds émettent une lumière infrarouge invisible. Un capteur PIR réagit au changement de lumière infrarouge. Cette modification est en général provoquée par la chaleur d'un être humain qui se déplace.

Assurez-vous d'abord d'adapter votre capteur PIR à son environnement. Comme un capteur PIR ne détecte que les modifications, il doit d'abord connaître la chaleur de la pièce quand il n'y a pas d'effraction.

Après la mise sous tension, le capteur a besoin de 30 secondes pour s'adapter à son environnement. Il ne doit y avoir aucun mouvement ni personne dans la zone observée durant la période d'adaptation.

Vous pouvez utiliser un boîtier pour limiter la zone observée par un capteur PIR. Si vous voulez tester rapidement un capteur PIR, placez-le dans un boîtier de telle sorte qu'il ne puisse voir que vers le haut.

Pendant que le capteur apprend à découvrir son environnement, vous pouvez continuer à coder sans vous préoccuper d'être parfaitement immobile. Quand vous voulez provoquer une alarme afin de tester votre code, passez la main au-dessus du boîtier.

Le capteur PIR Parallax a deux modes de fonctionnement :

- H pour rester à la valeur HIGH tant qu'il y a du mouvement,
- L pour renvoyer une valeur LOW après une alarme.

En mode L, le PIR se contente alors d'envoyer une impulsion même si le mouvement se poursuit après sa première détection.

Le capteur PIR a un petit cavalier pour sélectionner le mode de fonctionnement. Le cavalier est un petit rectangle recouvert de plastique noir. Dans ce projet, mettez le cavalier en mode H.

Alarme antivol pour Arduino

La Figure 6.15 illustre le circuit pour Arduino. Câblez le montage et exécutez l'Exemple 6.9.

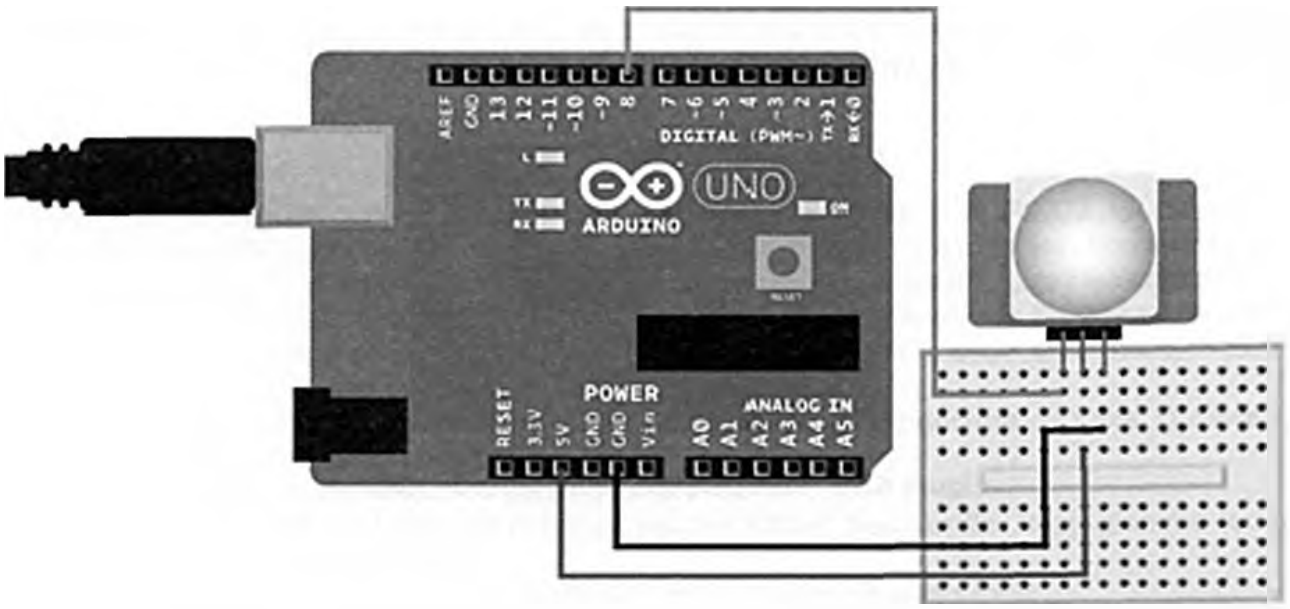


Figure 6.15 Circuit du capteur Parallax PIR Rev A pour Arduino.

Exemple 6.9 parallax_pir_reva.ino

```
// parallax_pir_reva.ino - affiche la détection du mouvement
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int sensorPin = 8;
const int ledPin = 13;
const int learningPeriod = 30*1000; // ms  // 1
int learningCompleted = 0;
void setup() {
  Serial.begin(115200);
  pinMode(sensorPin, INPUT);
  Serial.println("Apprentissage de l'environnement calme..."); // 2
  pinMode(ledPin, OUTPUT);
}
void loop() {
   if(millis() < learningPeriod) {      // 3
     delay(20); // ms   // 4
     return;    // 5
   }
   if(learningCompleted == 0) { // 6
     learningCompleted = 1;
     Serial.println("Apprentissage terminé.");
   }
   if(digitalRead(sensorPin) == HIGH) { // 7
     Serial.println("Mouvement détecté");
     digitalWrite(ledPin,HIGH);
   } else {
     Serial.println("Aucun mouvement détecté");
     digitalWrite(ledPin,LOW);
```

```
    }
    delay(100);
}
```

1 Le PIR a besoin d'une période calme pour s'adapter. Ici, on utilise 30 000 ms (30 secondes). La valeur est stockée dans une constante globale. Évitez la tentation de répéter la valeur dans le commentaire et utilisez plutôt le commentaire pour indiquer l'unité (millisecondes) de la variable. De cette manière, si vous modifiez une valeur, vous n'aurez pas besoin de changer le commentaire. De plus, si la valeur standard de l'unité (secondes) est le résultat d'un calcul (nombre de secondes * 1 000), écrivez le calcul dans le code (30*1000).

2 Affiche les instructions sur le port série. Pour ouvrir le Moniteur série, utilisez **Outils→ Moniteur série**. N'oubliez pas de sélectionner la même vitesse (« bauds », bits/s) à la fois dans le Moniteur série et votre code.

3 C'est un procédé courant pour mesurer le temps avec un Arduino. La fonction millis() retourne le nombre de millisecondes depuis le dernier démarrage.

4 Un court délai empêche que les appels répétés à loop() ne consomment 100 % du temps CPU.

5 Une instruction return termine la fonction loop(). Comme toujours, une fois que loop() se termine, elle est automatiquement appelée à nouveau.

6 La variable learningCompleted garantit que l'on n'affiche qu'une seule fois « Apprentissage terminé ». Ici, 1 et 0 sont utilisés comme des valeurs *true* et *false*.

7 Quand le capteur PIR détecte la chaleur d'un mouvement, sensorPin passe à HIGH.

Alarme antivol pour Raspberry Pi

La Figure 6.16 illustre le circuit pour Raspberry Pi. Assemblez-le selon le schéma et exécutez le code de l'Exemple 6.10.

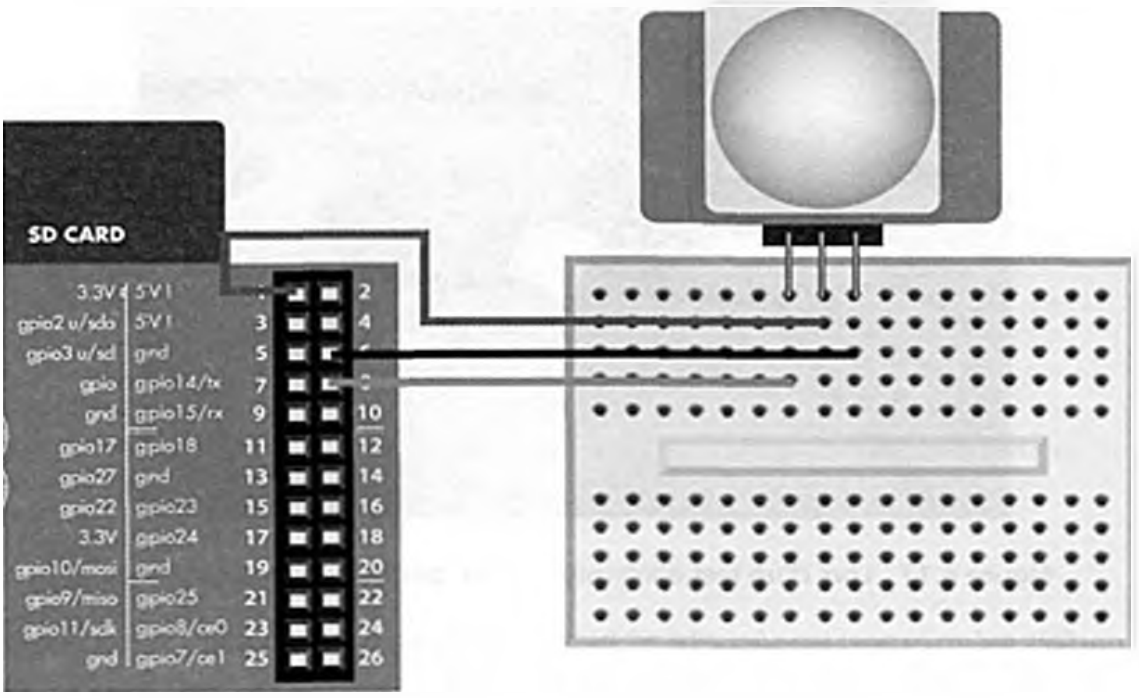


Figure 6.16 Circuit du capteur Parallax PIR sensor Rev A pour Raspberry Pi.

Exemple 6.10 parallax_pir_reva.py

```
# parallax_pir_reva.py - écrit à l'écran quand un mouvement est détecté
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio
learningPeriod = 60
def main():
        pirPin = 14
        gpio.mode(pirPin, "in")
        #Période d'apprentissage
        time.sleep(learningPeriod) # 1
        while (True):
                movement = gpio.read(pirPin) # 2
                if(movement == gpio.HIGH):
                        print "Mouvement détecté"
                else:
                        print "Aucun mouvement détecté"
                time.sleep(0.3)
if __name__ == "__main__":
        main()
```

1 Les capteurs ont besoin d'une période calme pour s'adapter. Ici, on utilise 60 secondes, mais vous pouvez essayer différentes valeurs.
2 La broche passe à HIGH quand un mouvement est détecté.

Expérience : leurrer une alarme

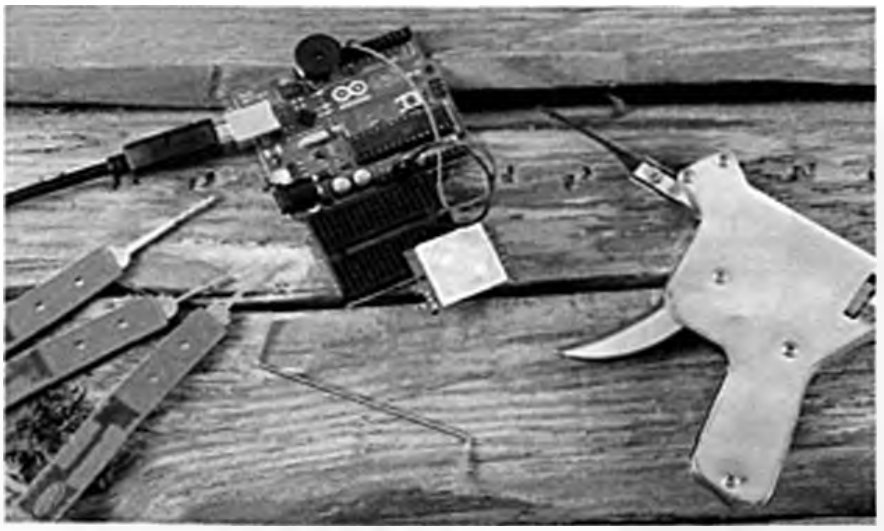


Figure 6.17 Lors d'un test d'intrusion, il faut parfois leurrer les alarmes.

Il arrive parfois qu'un expert en sécurité ait besoin de réaliser un test d'intrusion afin de prouver l'existence de vulnérabilités, et qu'il doive passer devant des alarmes sans se faire remarquer. Vous allez pouvoir réaliser cette expérience dans l'intimité de votre domicile ; cela vous permettra

de ne pas vous soucier des clients qui vous regardent (si vous êtes un professionnel de la sécurité) ni de finir en prison (si une carrière criminelle vous tente).

Vous pouvez tenter de leurrer votre capteur PIR, mais comme vous le constaterez bientôt, cela n'est pas aussi facile que dans les films d'action. Nous utiliserons le même code que précédemment, mais nous ajouterons un haut-parleur piézo. De cette manière, il est plus facile de savoir quand le mouvement est détecté. Connectez le piézo selon les instructions du diagramme de la Figure 6.18 et chargez le code de l'Exemple 6.11.

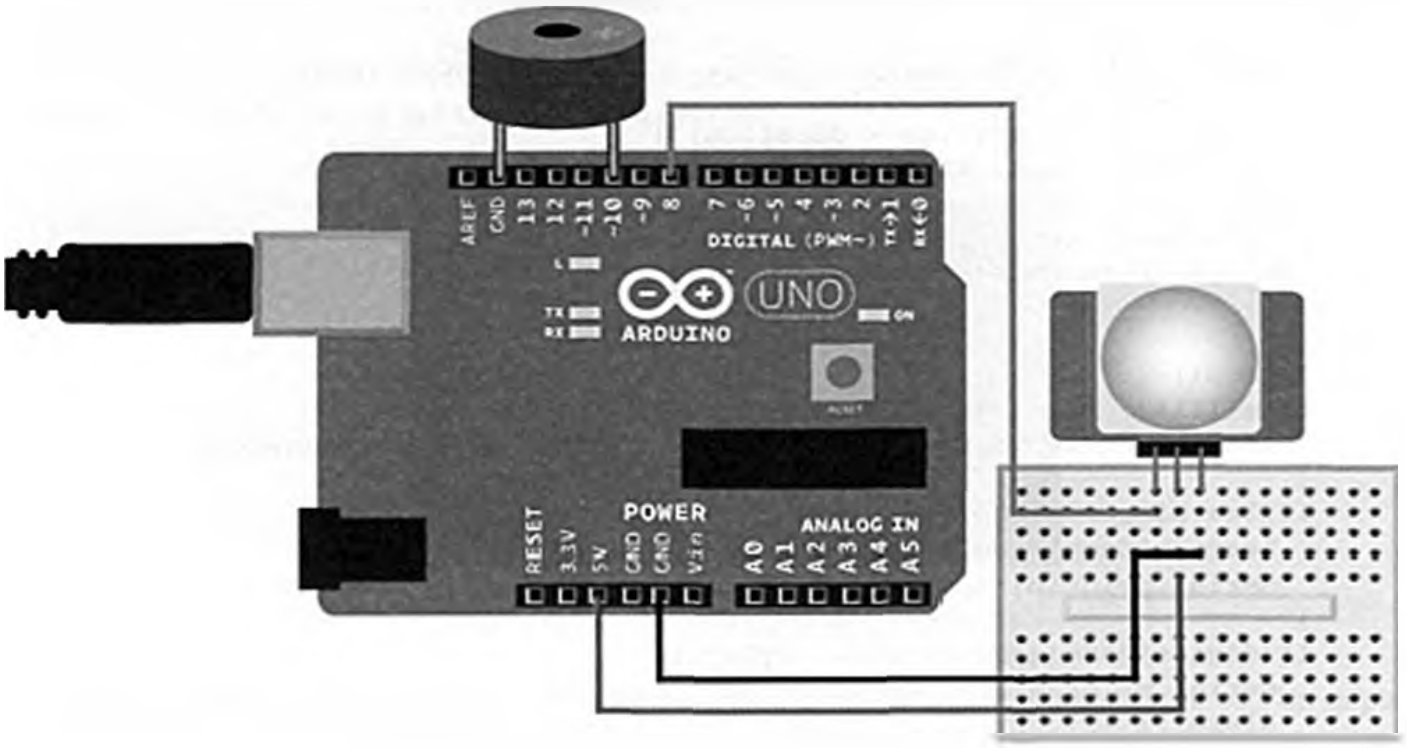


Figure 6.18 Circuit du capteur Parallax PIR Rev A pour Arduino avec une LED et un haut-parleur.

Exemple 6.11 parallax_PIR_revA_cheating_pir.ino

```
// parallax_PIR_revA_cheating_pir.ino - déclenche une alarme quand un
mouvement est détecté
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int sensorPin = 8;
const int ledPin = 13;
int speakerPin = 10;
const int learningPeriod = 30*1000; // 30 s pour l'apprentissage.
int learningCompleted = 0;
void setup() {
  Serial.begin(115200);
   pinMode(speakerPin, OUTPUT);
  pinMode(sensorPin, INPUT);
  Serial.println("Apprentissage pendant 30 secondes.");
  pinMode(ledPin, OUTPUT);
}
```

```
void alarm()
{
  wave(speakerPin, 440, 40);
  delay(25);
  wave(speakerPin, 300, 20);
  wave(speakerPin, 540, 40);
  delay(25);
}
void wave(int pin, float frequency, int duration)
{
  float period=1/frequency*1000*1000; // microsecondes (µs)
  long int startTime=millis();
  while(millis()-startTime < duration) {
    digitalWrite(pin, HIGH);
    delayMicroseconds(period/2);
    digitalWrite(pin, LOW);
    delayMicroseconds(period/2);
  }
}
void loop() {
   if(millis() < learningPeriod) {
     return; // le capteur n'a pas encore appris son environnement
   }
   if(learningCompleted == 0) {
     learningCompleted = 1;
     Serial.println("Apprentissage terminé.");
   }
   if(digitalRead(sensorPin) == HIGH) {
     Serial.println("Mouvement détecté");
     alarm();
     digitalWrite(ledPin, HIGH);
   } else {
     Serial.println("Aucun mouvement détecté");
     digitalWrite(ledPin, LOW);
   }
   delay(100);
}
```

Vous devez à présent entendre une alarme quand vous déplacez votre main au-dessus du capteur. Essayez d'approcher le capteur de quelques mètres sans le déclencher. Vous devrez vous déplacer très lentement et même comme cela, c'est difficile. Si vous utilisez un drap ou un grand torchon, c'est beaucoup plus facile. Couvrez-vous entièrement de carton (à la manière d'un fantôme, mais sans faire de trous pour les yeux) et rapprochez-vous doucement du capteur. Le fait de vous couvrir de carton limite la capacité du capteur à détecter la chaleur de votre corps. De la sorte, vous pouvez presque toucher le capteur avant qu'il ne vous remarque.

Dans les tests réels d'intrusion, vous aurez affaire à des alarmes qui combinent un capteur de distance à ultrasons et un capteur infrarouge passif. Et bien entendu, les caméras de sécurité détectent automatiquement les modifications de l'image...

Vous avez déjà appris à brouiller un capteur de distance à ultrasons dans la section *Expérience : des objets invisibles* du chapitre 3. Même si les caméras peuvent se révéler difficiles à tromper, vous pouvez vous faufiler entre elles. De nombreuses caméras utilisent l'illumination infrarouge

active pour voir dans le noir. Vous avez appris à voir la lumière infrarouge dans la section *Expérience : comment voir l'infrarouge* du chapitre 3.

PROJET : PONG

La détection du mouvement est bien plus intéressante quand vous pouvez présenter cette information à l'utilisateur. Dans ce projet, vous allez apprendre à utiliser des données d'un capteur pour déplacer des objets à l'écran. Pour simplifier les choses, on utilise un joystick comme périphérique d'entrée d'un jeu de Pong dans notre exemple. Ce que vous apprendrez ici pourra facilement être adapté à d'autres projets. Tout capteur peut servir de périphérique d'entrée et seule votre imagination va limiter ce qui sera affiché à l'écran.

Figure 6.19 On peut commencer à jouer !

ong, qui a été commercialisé à l'origine par Atari en 1972, est le jeu classique où l'on déplace une raquette de haut en bas. Votre but est que la balle ne traverse pas votre camp.

Figure 6.20 Plateau de jeu de Pong.

PCe projet permet de présenter **pyGame**, l'une des bibliothèques les plus simples pour programmer des jeux. Vous allez créer votre propre console de jeux et apprendre à utiliser les entrées du capteur pour déplacer des choses sur un grand écran. Si vous voulez améliorer les effets, utilisez un vidéoprojecteur à la place d'un écran.

Ce projet est facile à réaliser avec un Raspberry Pi, car vous pouvez connecter votre téléviseur ou un vidéoprojecteur à la sortie HDMI du Raspberry Pi. Faire la même chose avec un Arduino ne serait pas aussi aisé.

Cela ne veut pas dire que c'est impossible. À l'époque de Pong, les CPU étaient plus lents et avaient moins de RAM que l'Arduino. Ils traçaient les lignes de balayage sur un écran et

accomplissaient leurs calculs pendant les suppressions de trame. Vous trouverez un simple projet Arduino de Pong à l'adresse http://bit.ly/1f0GgHt. Si vous voulez réaliser toutes sortes de jeux vidéo passés de mode avec un Arduino, allez sur le site Wayne and Layne (http://bit.ly/QD3zeL).

Ce que vous allez apprendre

Dans le projet Pong, vous allez apprendre à :

- Utiliser les données d'un capteur pour déplacer des objets à l'écran.
- Afficher des graphiques haute définition avec un Raspberry Pi.
- Rendre le Raspberry Pi plus réactif en dessinant directement à l'écran sans passer par l'environnement de bureau ou le système X Window.
- Programmer un jeu simple avec pyGame.
- Démarrer automatiquement votre programe quand le Raspberry Pi démarre.

La Figure 6.21 illustre le schéma de câblage du projet Pong. Assemblez le tout comme indiqué, puis exécutez le programme de l'Exemple 6.12. Assurez-vous que la bibliothèque *botbook_gpio.py* est dans le même répertoire que le programme *pong.py*.

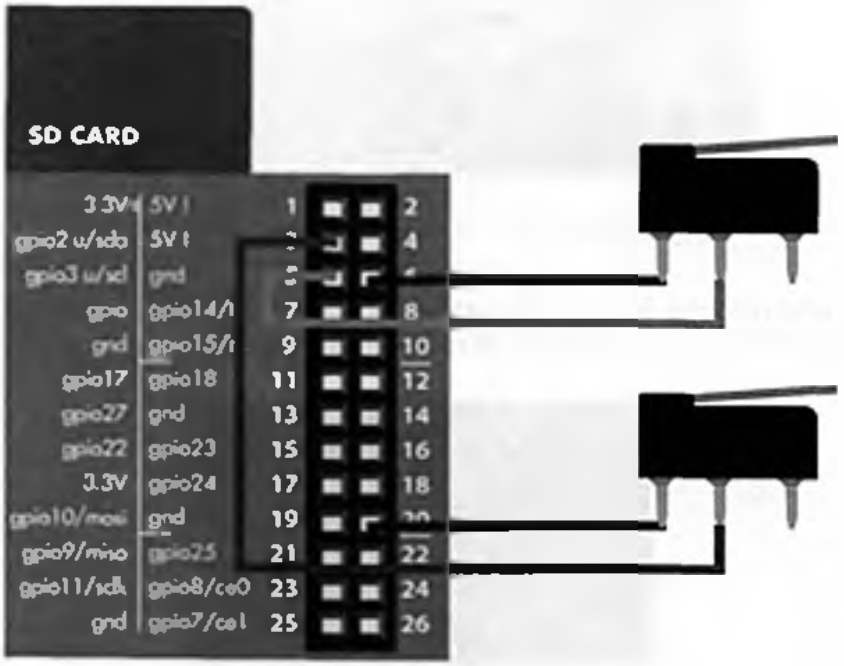


Figure 6.21 Connexions Pong pour Raspberry Pi.

Exemple 6.12 pong.py

```
# pong.py - joue un jeu classique avec un joystick et un grand écran
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import sys
import pygame
import botbook_gpio as gpio
from pygame.locals import *
```

```
print "Chargement de Pong..."
pygame.init()    # 1
width = pygame.display.Info().current_w    # 2
height = pygame.display.Info().current_h
size = width, height    # 3
background = 0, 0, 0    # 4
screen = pygame.display.set_mode(size,pygame.FULLSCREEN)    # 5
normalSpeed = 512
ballrect = Rect(width/2, height/2, 16, 16)    # 6
computerrect = Rect(width-20, 0, 20, 120)    # 7
playerrect = Rect(0, 0, 20, 120)    # 8
# mouvement différent en x et y
# la balle ne peut se déplace qu'avec des angles à 45 degrés
speed = [normalSpeed, normalSpeed]    # 9
clock = pygame.time.Clock()    # 10
pygame.mouse.set_visible(False)
mainloop = True
uppin = 2
downpin = 3
gpio.mode(uppin, "in")
gpio.mode(downpin, "in")
while mainloop:    # 11
    seconds = clock.tick(30) / 1000.0 # sec. depuis dernier frame # 12
    # saisie utilisateur
    for event in pygame.event.get():    # 13
        if event.type == pygame.QUIT: mainloop = False    # 14
        if (event.type == KEYUP) or (event.type == KEYDOWN):
            if event.key == K_ESCAPE: mainloop = False
    # mouvement et collisions
    playerspeed = 0
    if gpio.read(uppin) == gpio.LOW:
        playerspeed = -normalSpeed
    if gpio.read(downpin) == gpio.LOW:
        playerspeed = normalSpeed
    ballrect.x += speed[0] * seconds    # 15
    ballrect.y += speed[1] * seconds
    if ballrect.left < 0 or ballrect.right > width:    # 16
        ballrect.x = width/2;
    if ballrect.top < 0 or ballrect.bottom > height:
        speed[1] = -speed[1]
    computerrect.y = round(ballrect.y)    # 17
    playerrect.y += playerspeed * seconds    # 18
    if playerrect.top < 0: playerrect.top = 0    # 19
    if playerrect.bottom > height: playerrect.bottom = height    # 20
    if computerrect.colliderect(ballrect):    # 21
        speed[0] = -normalSpeed
    if playerrect.colliderect(ballrect):
        speed[0] = normalSpeed
    # on dessine le frame
    screen.fill(background)
    pygame.draw.circle(screen, (255, 255, 255), (int(round(ballrect.x+8)),
                                int(round(ballrect.y+8))), 10)    # 22
```

```
    pygame.draw.rect(screen, (255, 255, 255), computerrect)     # 23
    pygame.draw.rect(screen, (255, 255, 255), playerrect)
    pygame.display.update() # 24
```

1 Pour que pyGame fonctionne, il faut d'abord l'initialiser.

2 pyGame utilise des dimensions prédéfinies pour le canvas. Cela permet de travailler avec des pixels réels de l'écran, au lieu d'unités intermédiaires. Ces deux lignes récupèrent la largeur et la hauteur.

3 La taille du canvas est spécifiée sous la forme d'un tuple : (width, height).

4 Les couleurs dans pyGame sont rouge, vert et bleu (RGB). Vous avez déjà probablement travaillé avec des couleurs RGB si vous avez déjà défini des couleurs dans des pages web. (0, 0, 0) est égal à noir.

5 screen est l'objet où le dessin réel se produit. Il sera utilisé vers la fin de la boucle principale.

6 Crée des rectangles pour les objets du jeu à l'écran : Rect(x, y, width, height). La balle démarre dans le coin supérieur gauche (y==0, x==0). La balle est un rectangle de 16 × 16 pixels.

7 Cela place la raquette du joueur ordinateur près du bord droit du canvas. La raquette fait 20 pixels de large et 120 pixels de haut.

8 La raquette du joueur humain démarre à partir du coin gauche (0,0). Elle a les mêmes dimensions que la raquette du joueur ordinateur.

9 Vecteur de vitesse de la balle. À chaque seconde, la balle se déplace de 64 pixels sur l'axe *x* et de 64 pixels sur l'axe *y*.

10 Crée un nouvel objet **Clock** stocké dans la nouvelle variable déclarée clock. Cela sera utilisé pour garder la vitesse constante même si vous *overclockez* votre Raspberry Pi.

11 pyGame utilise un style de programmation avec une boucle principale, ce qui est très courant dans les jeux. Comme les boucles typiques Arduino loop() ou les structures while(True) que vous avez vues dans les exemples Python, la boucle principale redémarre chaque fois qu'elle se termine. Dans ces boucles typiques, on trouve en général la saisie de l'utilisateur, le déplacement des objets à l'écran, le contrôle des collisions, et enfin le dessin d'un frame à l'écran.

12 clock.tick() retourne l'heure depuis son dernier appel. Quand on l'appelle une fois dans un frame (à chaque exécution de boucle principale), on obtient le temps écoulé depuis le dernier frame. Le paramètre 60 spécifie un taux de frame maximum à 60 Hz, c'est-à-dire 60 frames par seconde. Si le jeu va trop vite, tick() fera patienter. Pour que les unités soient plus lisibles, on convertit les millisecondes (1/1 000 s) retournées par tick() en secondes.

13 La saisie au clavier est gérée par l'objet pygame.event. La fonction pygame.event.get() retourne un objet des événements qui contient une collection d'éléments que l'on peut parcourir. La boucle « for ITEM in LIST » parcourt un à un les éléments de la liste. Elle définit ITEM comme premier élément de la LIST lors de la première itération, puis comme second élément de la liste dans la deuxième itération, et ainsi de suite, jusqu'à la fin.

14 Compare chaque événement individuel aux constantes pyGame prédéfinies, et réagit en cas de correspondance. Par exemple, si le joueur accomplit une action d'interface utilisateur qui termine le jeu, alors l'événement QUIT est activé et c'est le moment de quitter le jeu.

15 Déplace la balle. Comme on prend en compte le temps écoulé entre chaque frame, le déplacement à 64 pixels par seconde est identique en dépit des modifications du taux de frame par seconde.

16 Si la balle atteint les limites du plateau de jeu, elle rebondit !

17 Déplace la raquette du joueur ordinateur exactement à la même hauteur que la balle. Vous avez donc un sérieux adversaire !

18 Déplace la raquette du joueur humain verticalement selon l'accélération des touches d'entrée. L'abréviation a+=2 est identique à a=a+2.

```
    19 Si le mouvement du joueur amène la raquette en haut du canvas, on se contente de rester
en haut.
    20 Si le mouvement du joueur amène la raquette en bas du canvas, on se contente de rester
en bas.
    21 Contrôle si la balle rentre en collision avec l'une des raquettes. Si tel est le cas, on définit la
direction de la balle avec une direction opposée à la raquette qu'elle a frappée.
    22 Dessine la balle à l'endroit de sa position calculée.
    23 La raquette du joueur ordinateur est tracée à la position calculée plus haut. Comme sa forme
est un rectangle, la boîte correspond exactement à l'objet lui-même.
    24 Affiche tout en même temps dans ce frame.
```

Conseils de packaging

Nous avons utilisé un boîtier en aluminium très robuste pour notre console de jeux (Figure 6.22). Cette construction change avantageusement des gadgets en plastique léger que l'on rencontre habituellement.

Au fond du boîtier, on a percé un trou pour les câbles d'alimentation et HDMI (Figure 6.23). Pour réaliser un trou large comme le nôtre, percez deux gros trous séparément, puis enlevez le métal entre les deux trous à l'aide d'une lame de scie.

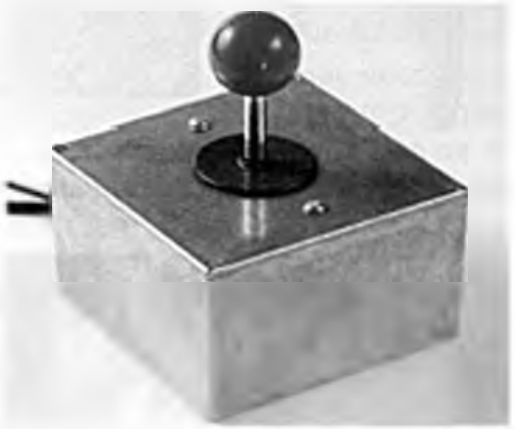


Figure 6.22 Boîtier du jeu Pong.

Figure 6.23 Trou pour l'alimentation et le port HDMI.

Un joystick de jeu d'arcade traditionnel fait parfaitement l'affaire pour ce boîtier indestructible (Figure 6.24). Pour relier le joystick, on a besoin de trois trous : deux de 5 mm pour monter le cadre et un trou de 10 mm pour faire passer le manche à travers le couvercle (Figure 6.25).

Figure 6.24 Joystick d'arcade.

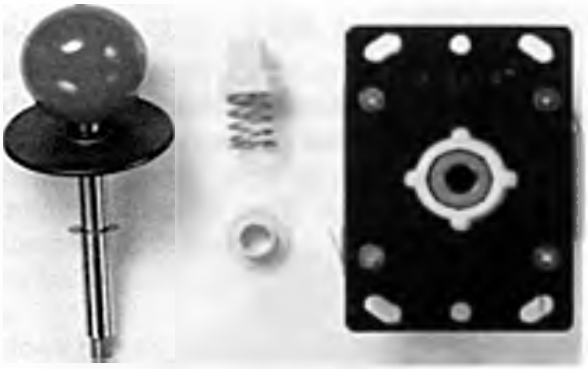


Figure 6.25 Pièces du joystick d'arcade.

Figure 6.26 Soudure des câbles.

Comme nous ne contrôlons que les mouvements de haut en bas, on n'a besoin de souder que deux microswitchs du joystick, comme cela est indiqué à la Figure 6.26.

Le Raspberry Pi a été simplement collé à chaud au fond de notre boîtier d'aluminium (voir la Figure 6.27).

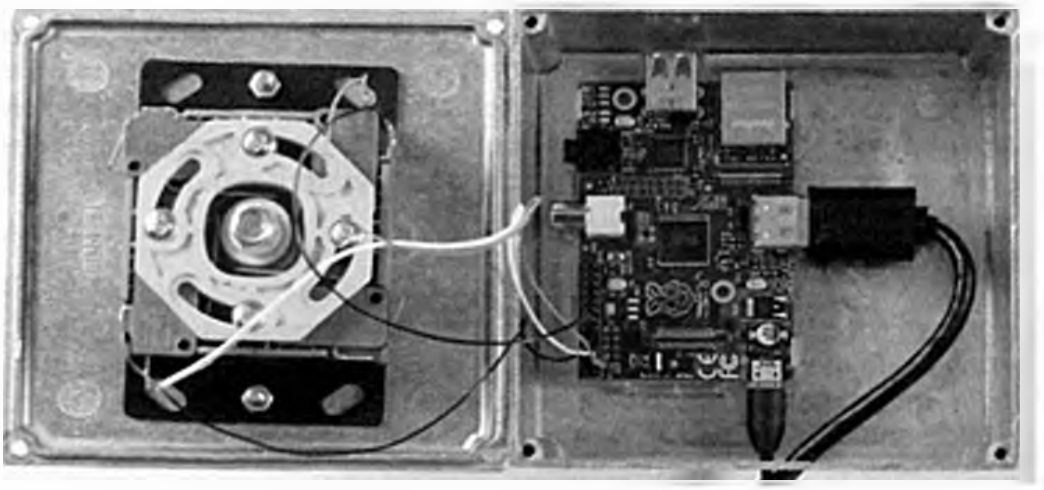


Figure 6.27 Montage complet.

Démarrage automatique du jeu lors du boot du Raspberry Pi

Maintenant que nous avons une magnifique manette de jeu, il est temps de nous préoccuper de l'expérience utilisateur. Ne serait-ce pas génial si le jeu démarrait automatiquement ? Sans clavier, il est difficile de saisir `python pong.py` chaque fois que vous voulez démarrer le jeu.

Pour démarrer le jeu en tant qu'utilisateur normal, vous devez paramétrer le système pour qu'il se connecte automatiquement en tant qu'utilisateur `pi`. Ensuite, vous pourrez configurer les choses pour que le jeu démarre dans le script de connexion de l'utilisateur. De cette manière, le jeu démarre immédiatement lors du boot du Raspberry Pi.

Exécution du jeu à la connexion

Quand on se connecte, **bash** s'ouvre. Bash est votre shell : il interprète les commandes que vous saisissez à l'invite de commande (`$`).

> Si votre Raspberry Pi démarre automatiquement en environnement graphique (comportement par défaut), vous devez modifier cette option pour qu'il démarre avec le shell, car le démarrage automatique du jeu à la connexion ne peut avoir lieu qu'en mode texte. Ouvrez **LXTerminal** et exécutez `sudo raspi-config`. Vous pouvez alors choisir l'option **Enable Boot To Desktop** et sélectionner **Console Text**.

La première chose que bash fait est d'exécuter des scripts comme `.profile`, `.login`, `.bash_profile` et `.bash_log- in`. Comme tous les fichiers de configuration utilisateur, ce sont des fichiers cachés du répertoire *home* de l'utilisateur. Pour les voir, vous devez utiliser le paramètre `-a` avec `ls`. Si vous n'êtes pas déjà dans votre répertoire *home*, vous pouvez rapidement le modifier (*/home/pi/*) avec `cd` sans arguments. Voici un ensemble de commandes (ne saisissez pas le `$`, puisqu'il s'agit de l'invite du shell) qui vous place dans votre répertoire home et liste tous les fichiers qu'il contient :

```
$ cd
$ ls -a
```

Le fichier *.bash_login* est un script de shell : il a des commandes à exécuter, les unes après les autres. Avant d'ajouter cette commande au fichier, testez la ligne de commande suivante :

```
$ python /home/pi/makesensors/pong/pong.py
```

Remplacez makesensors par le chemin de l'emplacement où vous avez enregistré le programme *pong.py*. Appuyez sur la touche Echap pour quitter le jeu. Si tout fonctionne parfaitement, vous pouvez à présent ouvrir le fichier *.bash_login* avec l'éditeur nano :

```
$ nano .bash_login
```

Ajoutez la ligne python /home/pi/makesensors/pong/pong.py à *.bash_login*. S'il y a déjà d'autres lignes dans le fichier, supprimez-les. L'Exemple 6.13 montre ce à quoi doit ressembler le fichier *.bash_login* maintenant. Enregistrez le fichier en saisissant Ctrl+X, puis pressez la touche y quand on vous demande d'enregistrer, puis appuyez sur la touche Entrée pour confirmer le nom du fichier.

Exemple 6.13 bash_login

```
# /home/pi/.bash_login - démarre automatiquement le jeu à la connexion
python /home/pi/makesensors/pong/pong.py
```

Déconnectez-vous :

```
$ exit
```

Ensuite, reconnectez-vous. Le jeu démarre automatiquement.

Connexion automatique

Il faut maintenant faire en sorte que votre utilisateur « pi » se connecte automatiquement lors du démarrage. Comme le jeu s'exécute immédiatement après la connexion de pi, cela démarre le jeu après l'allumage du Raspberry Pi. Si vous êtes toujours dans le jeu, appuyez sur Echap pour en sortir.

Le programme init contrôlant le démarrage du système, nous allons modifier ses paramètres. Tous les paramètres système figurent dans */etc/*, et le fichier de configuration démarre en général avec le nom de la chose : */etc/init** (dans ce cas, */etc/inittab*). Comme la connexion des utilisateurs est un processus qui nécessite des privilèges complets, vous devez modifier le fichier en tant que root à l'aide de la commande sudoedit :

```
$ sudoedit /etc/inittab
```

> Pour modifier des fichiers texte en tant que root, on utilise sudoedit à la place de nano avec sudo. De cette manière, on n'a pas d'erreurs sur les droits d'appartenance du fichier historique de nano quand on se sert plus tard de nano en tant qu'utilisateur normal.

Modifiez la ligne qui gère le premier « terminal virtuel » de telle sorte qu'elle vous connecte automatiquement (n'ajoutez pas de ligne, mais modifiez celle qui démarre par `1:2345:respawn:/sbin/getty`) :

```
1:2345:respawn:/sbin/getty -noclear 38400 tty1 -autologin pi
```

Enregistrez avec Ctrl+X, puis appuyez sur la touche y quand on vous demande de sauvegarder, et pressez enfin la touche Entrée pour confirmer le nom du fichier. L'Exemple 6.14 présente un fichier complet */etc/inittab* modifié.

Arrêtez le Raspberry Pi :

```
$ sudo shutdown -P now
```

Déconnectez puis reconnectez l'alimentation USB.

Relaxez-vous pendant que vous êtes connecté automatiquement. Le jeu démarre et votre propre console de jeux est prête. Il est temps de jouer à Pong !

Exemple 6.14 inittab

```
# /etc/inittab - connecte automatiquement l'utilisateur pi au démarrage
(désactive aussi la console série)
id:2:initdefault:
si::sysinit:/etc/init.d/rcS
~~:S:wait:/sbin/sulogin
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
z6:6:respawn:/sbin/sulogin
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
# modifié :
1:2345:respawn:/sbin/getty -noclear 38400 tty1 -autologin pi
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
# on supprime la console série pour que le port série puisse être utilisé
avec les capteurs
# T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Connectez-vous en tant qu'utilisateur normal quand c'est possible. N'accomplissez que l'administration système (par exemple, apt- get, raspi-config) en tant que root. Il est particulièrement important que les jeux s'exécutent sous l'utilisateur normal car ils ont souvent besoin d'un accès direct au matériel, et les programmes qui rentrent profondément dans le système peuvent faire plus de dégâts s'ils ont un bug sérieux. C'est la raison pour laquelle vous devez utiliser l'autologin et un script de connexion pour démarrer le jeu (comme cela est décrit ici), au lieu de simplement placer le jeu dans un script de démarrage comme rc.local (qui s'exécute toujours en tant que root).

Pour arrêter de jouer et retourner à la ligne de commandes, appuyez simplement sur Echap.

Vous avez à présent testé le mouvement de nombreuses manières différentes, en commençant par les boutons classiques et les potentiomètres. Ces capteurs sont les archétypes des capteurs de résistance. Le bouton est le plus simple capteur de résistance numérique (*on/off* ou résistance zéro/infinie), et le potentiomètre est le plus simple capteur de résistance analogique. Vous êtes maintenant capable d'utiliser des circuits et du code avec de nombreux autres capteurs.

Il existe des capteurs plus spécialisés qui détectent le toucher et même le bois. Vous avez également des capteurs de pression qui peuvent se révéler utiles pour savoir si un lit ou un siège est occupé.

Dans le prochain chapitre, nous allons mesurer une forme d'énergie moins tangible : la lumière.

7 Détecter la lumière

Certains robots sont capables de suivre apparemment sans difficulté un chemin compliqué. Quand on regarde plus attentivement, on s'aperçoit qu'ils suivent une ligne colorée sur le sol. La couleur n'étant rien d'autre que de la lumière réfléchie, certains capteurs peuvent détecter la couleur d'une surface. D'autres, munis de tubes spéciaux, peuvent aussi détecter la direction de la lumière.

> Vous souhaitez mesurer le mouvement d'un être humain avec une lumière infrarouge ? Reportez-vous à l'*Expérience antivol ! (capteur infrarouge passif)* du chapitre 6. Vous avez besoin de savoir si un objet est situé à proximité en utilisant l'infrarouge ? Reportez-vous à l'*Expérience : détection des obstacles avec l'infrarouge (capteur de distance infrarouge)* du chapitre 3.

EXPÉRIENCE : DÉTECTION DE FLAMME (CAPTEUR DE FLAMME)

Les flammes émettent une plage de lumière infrarouge qui n'est pas très courante dans la lumière ambiante. Le capteur de flamme KY-026 indique le niveau de lumière infrarouge par une modification de la résistance (Figure 7.1).

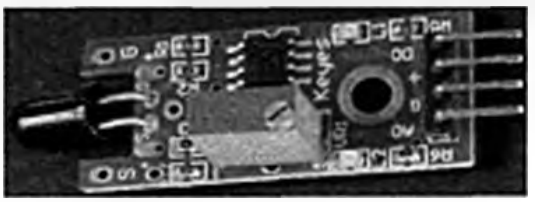


Figure 7.1 Capteur de flamme

Le code que vous allez écrire pour la détection de flamme est identique à celui que vous avez utilisé avec un capteur de résistance analogique : on utilise analogRead() pour lire la tension d'une broche.

Le capteur de flamme KY-026 offre deux façons de mesurer une flamme : digitalRead() et analogRead(). Même si le code de cette expérience implémente ces deux méthodes, vous pouvez choisir celle qui vous convient le mieux dans votre propre code.

L'utilisation du seul mode numérique est particulièrement pratique avec le Raspberry Pi car il ne possède pas de convertisseur analogique-numérique.

Capteur de flamme pour Arduino

La Figure 7.3 illustre le schéma de câblage du capteur de flamme avec Arduino. Câblez selon le schéma puis exécutez le sketch de l'Exemple 7.1.

Figure 7.2 Prototype de robot suivant une flamme

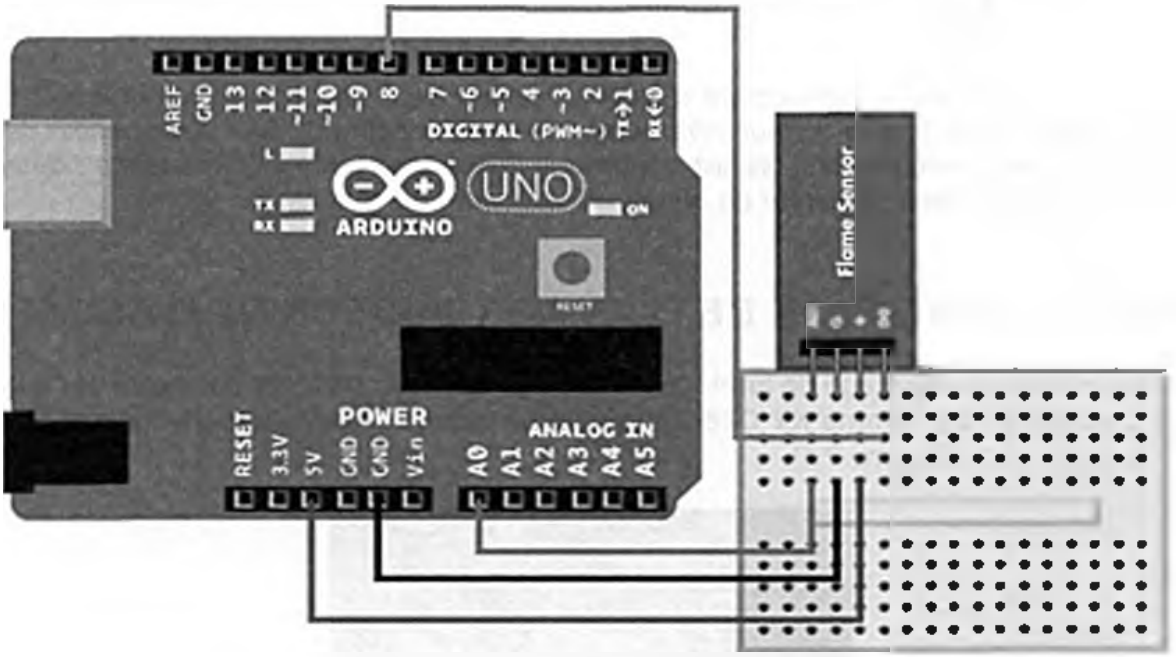


Figure 7.3 Circuit du capteur de flamme pour Arduino

Exemple 7.1. ky_026_flame.ino

```
// ky_026_flame.ino - indique le niveau de lumière infrarouge de la flamme
sur le port série
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int analogPin = A0;
const int digitalPin = 8;
const int ledPin = 13;
void setup() {
  Serial.begin(115200);
  pinMode(digitalPin,INPUT);
  pinMode(ledPin,OUTPUT);
}
void loop()
```

```
{
  int threshold = -1; // HIGH ou LOW
  int value = -1; // 0..1023
  value = analogRead(analogPin);          // 1
  threshold = digitalRead(digitalPin);    // 2
  Serial.print("Brut : ");
  Serial.print(value);
  Serial.print(" Au-dessus du seuil : ");
  Serial.println(threshold);
  delay(10);
  if(threshold==HIGH) { // 3
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

1 Lit la tension brute à partir de A0. C'est un entier dans la plage 0 (0 V) à 1 023 (+5 V).
2 Lit la tension de D8 : LOW (0 V) ou HIGH (+5 V).
3 Allume la LED intégrée (D13) si une flamme est détectée.

Capteur de flamme pour Raspberry Pi

La Figure 7.4 illustre le circuit de connexion du capteur au Raspberry Pi.
Câblez selon le schéma et exécutez le programme de l'Exemple 7.2.

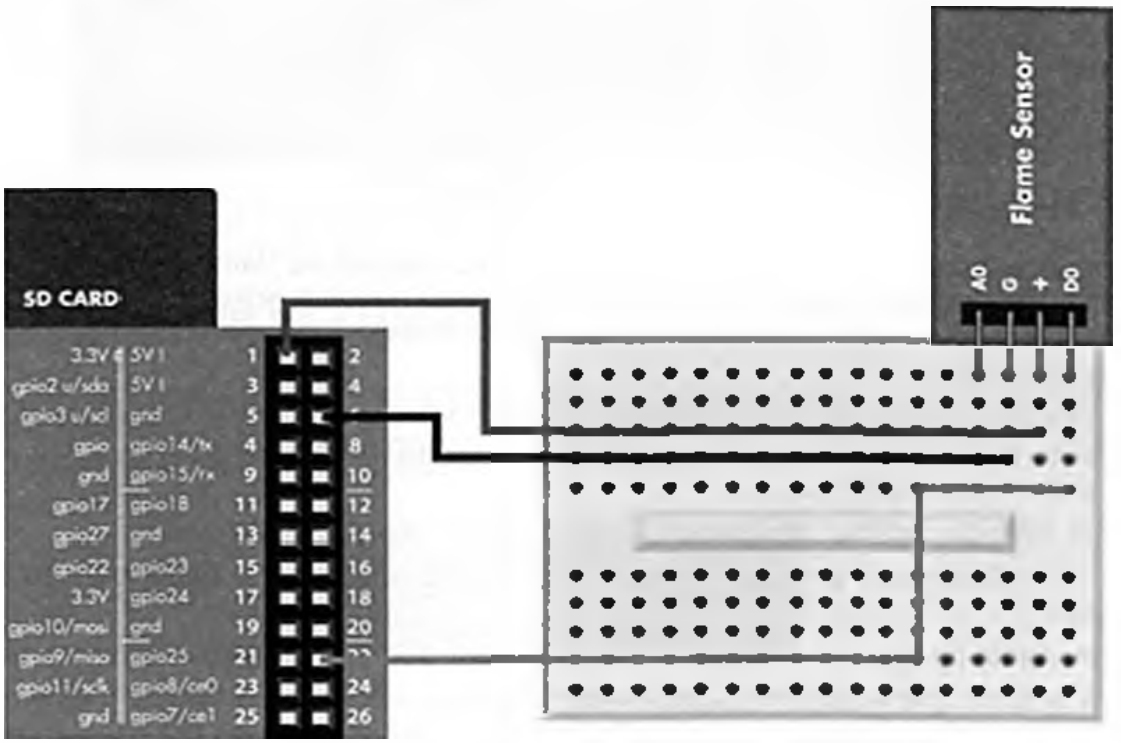


Figure 7.4 Circuit du capteur de flamme pour Raspberry Pi

Exemple 7.2. ky_026_flame.py

```
# ky_026_flame.py - Indique la présence d'une lumière infrarouge dans une
flamme
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio      # 1
def main():
        triggerPin = 25
        gpio.mode(triggerPin, "in")      # 2
        flame = gpio.read(triggerPin)    # 3
        if(flame == gpio.HIGH): # 4
                print "Flamme détectée"
        else:
                print "Pas de flamme détectée"
        time.sleep(0.5)
if __name__ == "__main__":
        main()
```

1 Assurez-vous qu'une copie de la bibliothèque *botbook_gpio.py* figure bien dans le même répertoire que ce programme. Vous pouvez télécharger cette bibliothèque ainsi que tous les exemples de code à partir de http://botbook.com. Voir la section *GPIO sans root* du chapitre 1 pour des informations sur la configuration de l'accès GPIO du Raspberry Pi.
2 La broche est définie en mode In pour lire sa tension à la ligne de code suivante.
3 Lit l'état de la broche 23. La valeur est True pour HIGH (+3,3 V) et False pour LOW (0 V).
4 Notez qu'une valeur booléenne (true ou false) peut être utilisée dans un if sans opérateur de comparaison. En d'autres termes, vous pouvez écrire if(b) à la place de if(b==True).

EXPÉRIENCE : PRÉCISION DE LA FLAMME

Le potentiomètre intégré de réglage de la sensibilité du capteur de flamme est très utile, notamment pour l'ajuster de telle sorte que la lumière ambiante ne déclenche pas le capteur constamment. Il peut aussi être utilisé pour régler le capteur de flamme pour qu'il réagisse à un niveau de flamme très spécifique.

Commencez par placer une LED supplémentaire entre GND et la broche 13 (Figure 7.5). Comme il est plus facile de distinguer une LED complète que la LED embarquée sur la carte, l'activation du capteur se remarquera mieux.

Chargez le code du capteur de flamme (Exemple 7.1) sur votre Arduino.

Tournez complètement à droite le potentiomètre, puis tournez-le à gauche jusqu'à ce que la LED s'éteigne.

Il est souhaitable de tirer les rideaux car le soleil peut dominer les autres sources de lumière.

Allumez à présent une allumette. La LED doit s'allumer à nouveau. Essayez d'ajuster le potentiomètre de sensibilité de telle sorte que le capteur ne réagisse qu'à une grande flamme.

Ne vous laissez pas distraire par le spectacle de la LED, sinon vous allez vous brûler les doigts quand l'allumette se consume.

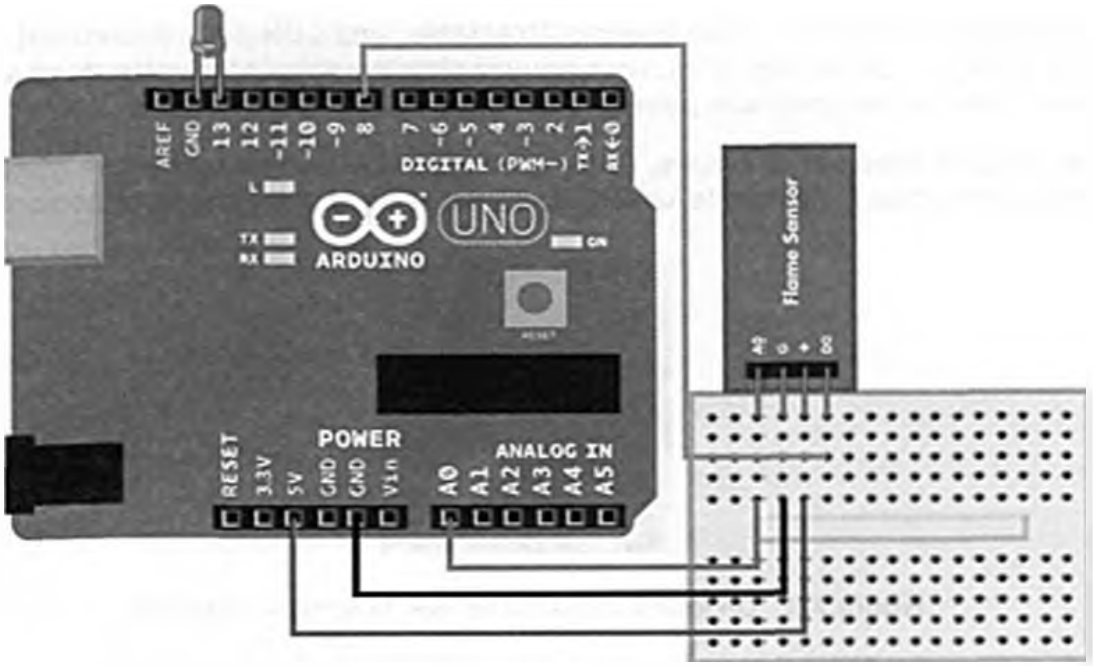


Figure 7.5 Capteur de flamme avec une LED

Figure 7.6 Ajustement et test de la sensibilité du capteur de flamme

EXPÉRIENCE : VOIR LA LUMIÈRE (PHOTORÉSISTANCE)

Une photorésistance (on emploie également l'acronyme anglais LDR, pour *Light-Dependent Resistor*, résistance dépendante de la lumière) modifie sa résistance selon le niveau de lumière visible (Figure 7.7). Sa résistance est plus faible en pleine lumière.

Figure 7.7 Photorésistance

Les photorésistances peuvent allumer les lumières quand il fait sombre, détecter si un boîtier est ouvert dans une pièce éclairée, et faciliter la création de robots qui aiment la lumière. Si

l'on sait correctement utiliser du tube thermorétractable, une LDR peut également détecter la direction de la lumière. Lors de vos tests, vous pouvez simplement placer votre doigt sur une LDR pour faire l'obscurité, ou au contraire pointer une lampe de poche sur la LDR.

Pour tester un robot attiré par la lumière, essayez de le recouvrir d'une couverture (Figure 7.8). De la sorte, vous n'avez pas à éteindre la lumière ou à vous réfugier dans une pièce sombre.

Figure 7.8 On joue à cache-cache avec la lumière ambiante

LDR pour Arduino

Une photorésistance n'est qu'une résistance variable avec deux pattes. Avec Arduino, il suffit de configurer le capteur avec une autre résistance qui agit comme réducteur de tension (voir l'*Expérience : potentiomètre (résistance variable)* du chapitre 5), puis d'utiliser analogRead(). De cette manière, une photorésistance est similaire aux autres capteurs de résistance analogique.

Montez la LDR en suivant le schéma de la Figure 7.9, puis exécutez le sketch de l'Exemple 7.3.

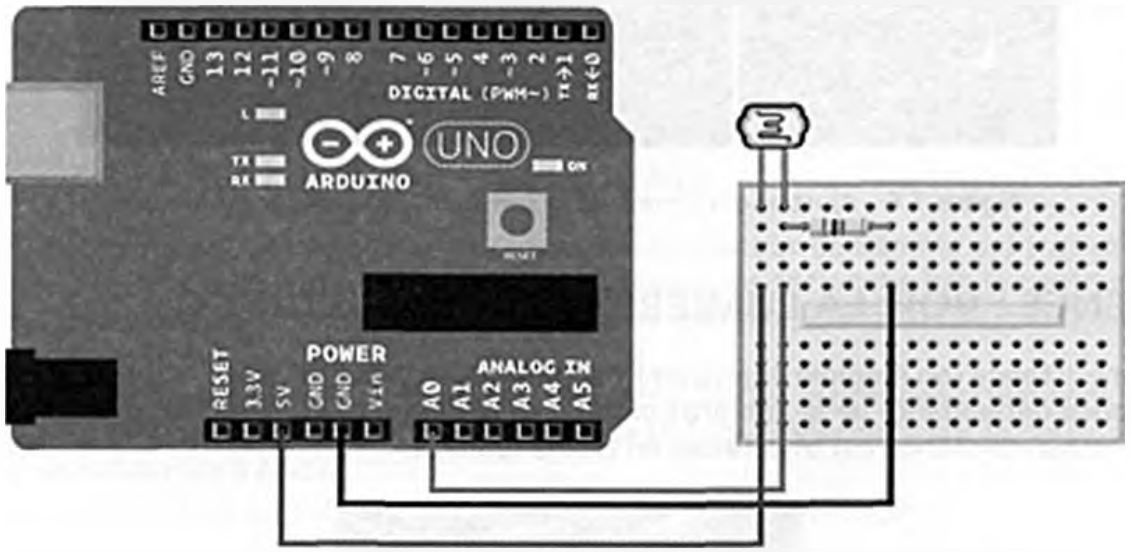


Figure 7.9 Circuit de photorésistance pour Arduino

Exemple 7.3. ldr_light_sensor.ino

```
// ldr_light_sensor.ino - affiche le niveau élevé de lumière avec la LED
intégrée
```

```
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int lightPin = A0;
const int ledPin = 13;
int lightLevel = -1;
void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
}
void loop() {
  lightLevel = analogRead(lightPin);     // 1
  Serial.println(lightLevel);
  if(lightLevel < 400) {          // 2
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(10);
}
```

1 Pour mesurer la tension de la broche A0, il suffit d'utiliser analogRead().
2 Si l'affichage de la valeur sur le port série n'est pas suffisant, vous pouvez aussi allumer une LED quand le niveau de lumière excède une valeur choisie de façon expérimentale.

LDR pour Raspberry Pi

Connectez les composants selon le schéma de la Figure 7.10, puis exécutez le code de l'Exemple 7.4.

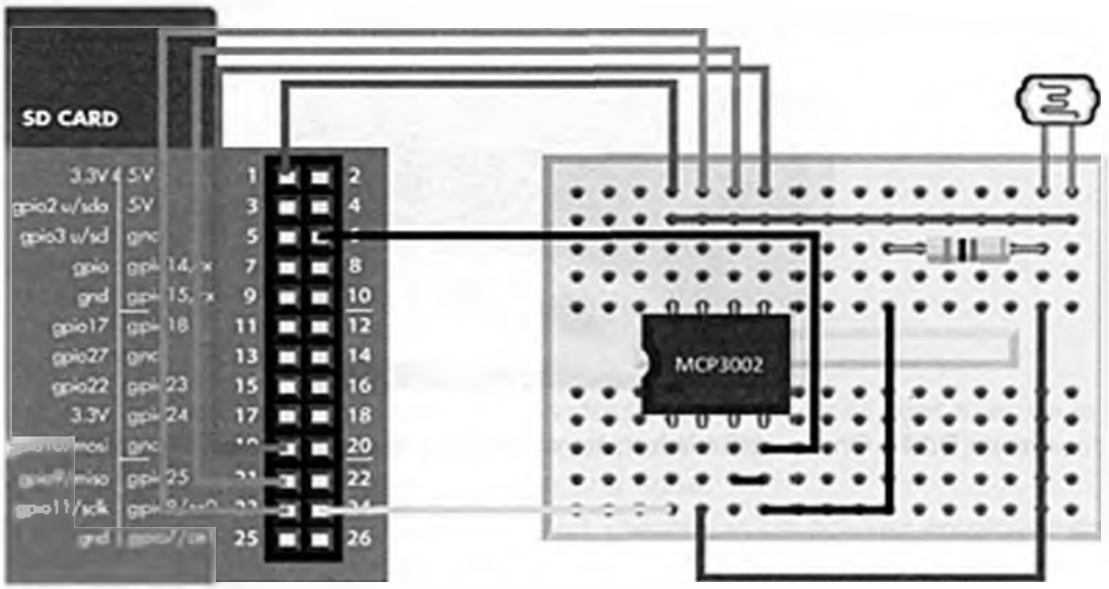


Figure 7.10 Circuit de photorésistance pour Raspberry Pi

Exemple 7.4. ldr.py

```
# ldr.py - perçoit le niveau de lumière et l'affiche à l'écran
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp
def main():
        while True:
                lightLevel = mcp.readAnalog()
                print("Le niveau actuel de lumière est %i " % lightLevel)
                time.sleep(0.5)
if __name__ == "__main__":
        main()
```

La bibliothèque *botbook_mcp3002.py* doit se trouver dans le même répertoire que ce programme. Vous devez aussi installer la bibliothèque *spidev*, qui est importée par *botbook_mcp3002*. Reportez-vous aux commentaires au début de *botbook_mcp3002/botbook_mcp3002.py* ou à la section *Installation de SpiDev* au chapitre 3.

EXPÉRIENCE : UNE DIRECTION

Aimeriez-vous connaître la direction d'où la lumière provient plutôt que son intensité ?

Une photorésistance nue réagit à la lumière environnante et vous ne pouvez donc pas l'utiliser, par exemple, pour diriger un robot vers une source de lumière. Il y a cependant une solution très simple à ce problème.

Prenez un morceau de tube thermorétractable et faites-en un manchon pour le capteur, comme cela est illustré à la Figure 7.11. Cela empêche le capteur de voir la lumière émanant de toutes les directions.

Figure 7.11 On empêche la lumière d'atteindre le capteur depuis les côtés

Vous pouvez utiliser un autre matériel que du tube thermorétractable tant qu'il bloque la lumière en provenance des côtés (Figure 7.12).

Quand vous avez plusieurs photorésistances, c'est une bonne idée de placer les trois câbles et la résistance à l'intérieur d'un tube thermorétractable, comme cela est illustré à la Figure 7.13.

Figure 7.12 Photorésistances à l'intérieur d'un dispositif en plastique servant d'œillères

Figure 7.13 Avec du tube thermorétractable, vous pouvez conditionner les câbles proprement

EXPÉRIENCE : SUIVEZ LA LIGNE

Le fait de suivre une ligne est un moyen facile de déplacer un robot le long d'un chemin prédéfini. Pour ce faire, on crée en général des « rails » avec du scotch noir. Nous avons utilisé l'évitement de ligne pour limiter le champ d'action de notre robot. La Figure 7.14 illustre un détecteur de ligne.

Figure 7.14 Capteur de suivi de ligne

Les détecteurs de ligne éclairent la surface au-dessus de laquelle ils sont avec de la lumière, en général de l'infrarouge. La surface est considérée comme « blanche » si suffisamment de lumière est réfléchie ; tout le reste est considéré comme une ligne.

Pour savoir si votre robot se détourne du droit chemin, vous pouvez employer deux ou trois détecteurs de ligne côte à côte (par exemple, si le détecteur du milieu voit une ligne, mais que les deux autres voient du blanc, vous savez que vous suivez la ligne).

Il existe des capteurs de détection de ligne qui combinent plusieurs capteurs en un seul dispositif.

Capteur de ligne pour Arduino

Comme nous utilisons ici un détecteur de ligne avec trois fils, aucune résistance pull-up n'est nécessaire. Il y a un fil pour le positif, un autre pour le signal, et le troisième pour la masse. La circuiterie du capteur inclut toutes les résistances et les composants nécessaires.

Montez le circuit selon le schéma de la Figure 7.15, puis exécutez le sketch de l'Exemple 7.5.

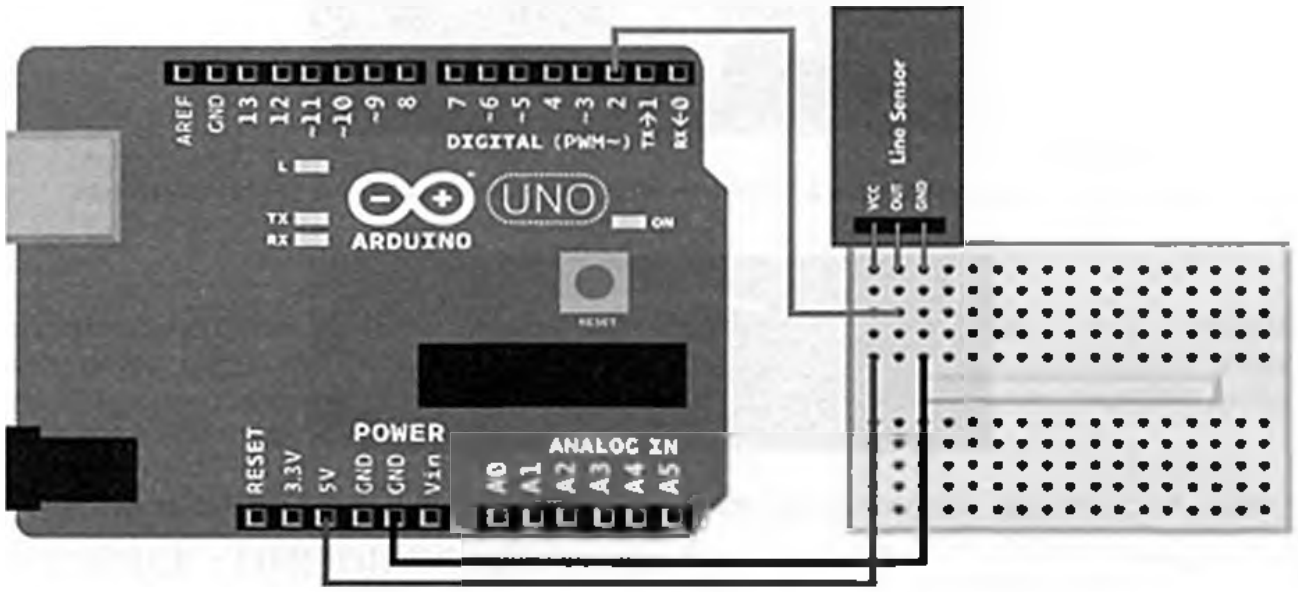


Figure 7.15 Circuit de capteur de ligne pour Arduino

Exemple 7.5. line_sensor.ino

```
// line_sensor.ino - affiche sur le port série si l'on est sur une ligne
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int sensorPin = 2;
const int ledPin = 13;
int lineFound = -1;
void setup() {
  Serial.begin(115200);
  pinMode(sensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
}
void loop() {
  lineFound = digitalRead(sensorPin);   // 1
  if(lineFound == HIGH) {
    Serial.println("Capteur sur la ligne");
    digitalWrite(ledPin, HIGH);
  } else {
    Serial.println("Capteur en dehors de la ligne");
    digitalWrite(ledPin, LOW);
  }
  delay(10);
}
```

1 digitalRead(8) retourne HIGH si le capteur s'écarte de la ligne.

Capteur de ligne pour Raspberry Pi

Comme un capteur de ligne est un capteur numérique, sa connexion au Raspberry Pi est aussi simple qu'avec Arduino. La Figure 7.16 illustre le diagramme du circuit. Câblez selon le schéma puis exécutez le code de l'Exemple 7.6.

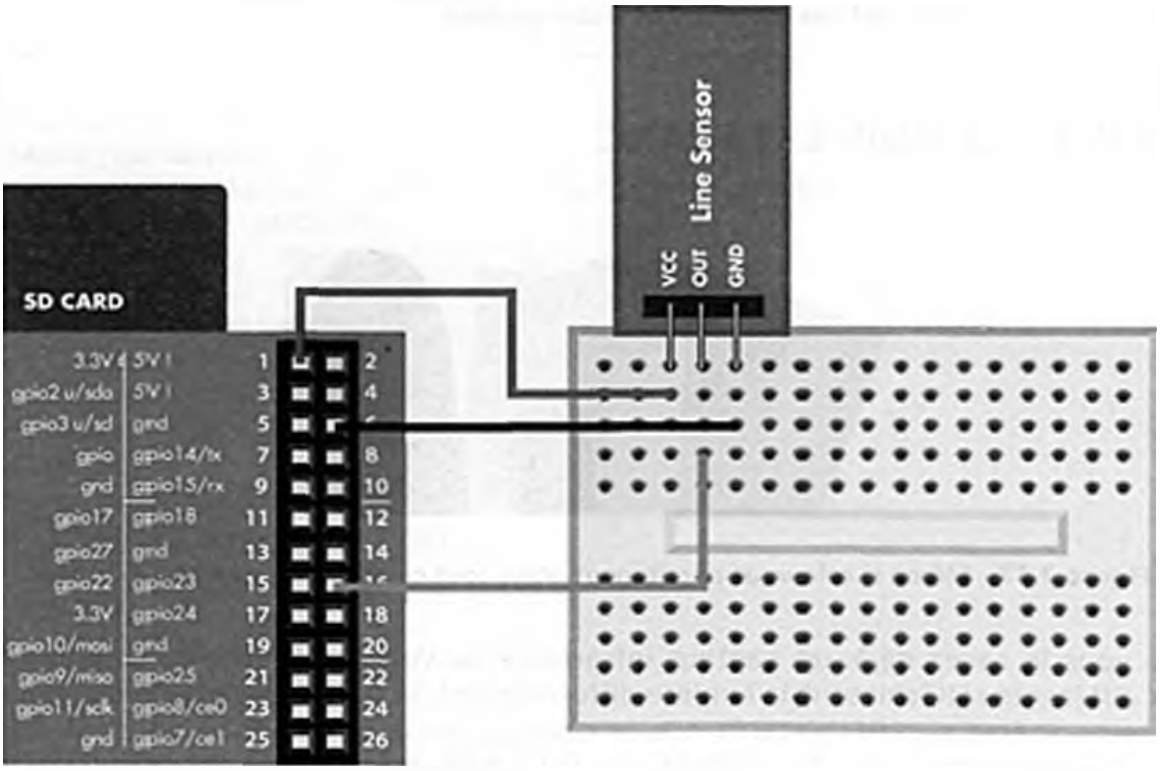


Figure 7.16 Circuit de capteur de ligne pour Raspberry Pi

Exemple 7.6. line_sensor.py

```
# line_sensor.py - affiche si l'on est sur une ligne
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import os
import botbook_gpio as gpio      # 1
def main():
        linePin = 23
        gpio.mode(linePin, "in")
        while True:
                lineState = gpio.read(linePin)  # 2
                if( lineState == gpio.HIGH ):
                        print "Capteur sur la ligne"
                else:
                        print "Capteur en dehors de la ligne"
                time.sleep(0.5)
```

```
if __name__ == "__main__":
        main()
```

1 Assurez-vous qu'il y a une copie de la bibliothèque *botbook_gpio.py* dans le même répertoire que ce programme. Vous pouvez télécharger cette bibliothèque ainsi que tous les exemples de code à partir de http://botbook.com. Reportez-vous à la section *GPIO sans root* du chapitre 1 pour des informations sur la configuration de l'accès GPIO du Raspberry Pi.
2 La valeur numérique est lue comme tout autre capteur.

EXPÉRIENCE : LE NOIR EST BLANC

Figure 7.17 Même si cela vous paraît incroyable, tout ce que vous voyez ici est blanc

Comme vous le savez déjà, un capteur infrarouge ne voit pas le monde comme nous. Les matériaux différents réfléchissent la lumière différemment, et parfois les objets qui apparaissent sombres peuvent être si réfléchissants que le capteur pense qu'ils sont blancs. Si votre robot suiveur de ligne agit bizarrement, cela peut être la texture de la surface qui vous joue des tours, mais pas le code.

En général, noir c'est noir et blanc c'est blanc, mais c'est carrément l'inverse qui nous est arrivé. Lorsque nous avons présenté notre robot contrôlé par la pensée à la Foire des *makers*, nous avions apporté du scotch et du carton pour réaliser une plateforme avec des bordures noires et un centre blanc. Notre idée était que le robot fasse un demi-tour quand il voyait du noir, ce qui lui permettait de rester sur la plateforme. Bizarrement, le détecteur de ligne a vu le scotch et le carton en couleurs inversées car l'une était très réfléchissante et l'autre très mate.

Vous pouvez ajuster la sensibilité du suiveur de ligne en réglant le potentiomètre illustré à la Figure 7.18.

Figure 7.18 Ajustement de la sensibilité du détecteur de ligne

Chargez le code de l'Exemple 7.7 dans l'Arduino. Nous avons légèrement modifié le code de l'exemple précédent de telle sorte que le Moniteur série dit à présent NOIR ou BLANC en fonction de ce qu'il voit.

Exemple 7.7. line_sensor_black_or_white.ino

```
// line_sensor_black_or_white.ino - capteur de suivi de ligne
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int sensorPin = 2;
const int ledPin = 13;
int lineFound = -1;
void setup() {
  Serial.begin(115200);
  pinMode(sensorPin, INPUT);
  // Pas besoin de pull-up car le capteur en a déjà.
  pinMode(ledPin, OUTPUT);
}
void loop() {
  lineFound = digitalRead(sensorPin);
  if(lineFound == 1) {
    Serial.println("NOIR");
    digitalWrite(ledPin, HIGH);
  } else {
    Serial.println("BLANC");
    digitalWrite(ledPin, LOW);
  }
  delay(50);
}
```

Ce code combine l'affichage sur le port série avec le code de la section *Capteur de ligne pour Arduino*.

Testez différents matériaux. Arrivez-vous à en trouver qui soient noirs pour un être humain et blancs pour le capteur ?

EXPÉRIENCE : TOUTES LES COULEURS DE L'ARC-EN-CIEL

Un capteur couleur mesure la couleur d'une surface et retourne les valeurs RVB (Figure 7.19). Pour chaque couleur de base (rouge, vert et bleu), un capteur couleur a un filtre de couleur au-dessus d'une photodiode. Le capteur de chaque couleur est lu comme n'importe quel capteur de résistance analogique.

L'expérience affiche les valeurs RGB (rouge, vert et bleu ; en français, vous trouverez aussi l'acronyme RVB comme traduction de RGB) de la lumière qu'elle voit, une valeur pour chaque couleur. Vous pouvez utiliser ce capteur pour réaliser le Dôme caméléon (voir dans ce chapitre, le *Projet : Dôme caméléon*).

Les couleurs sont simplement des noms pour des longueurs d'ondes spécifiques de lumière. Par exemple, 555 nanomètres (nm, qui décrit la longueur d'onde de la lumière) ou 540 térahertz (THz, sa fréquence) correspondent au vert. Certains animaux peuvent voir des couleurs que les humains ne perçoivent pas, comme celles dans la plage de l'infrarouge ou de l'ultraviolet.

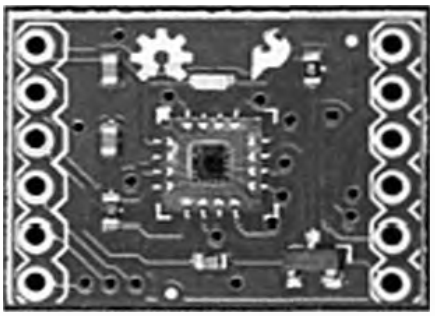


Figure 7.19 Capteur couleur

Les couleurs de base sont vraiment fondamentales pour les humains. Les seules couleurs que l'œil humain peut percevoir sont le rouge, le vert et le bleu. Il y a trois types de cônes rétiniens dans la rétine, un type pour chaque couleur.

Capteur couleur pour Arduino

La Figure 7.20 illustre le diagramme du circuit du capteur couleur pour Arduino. Câblez selon le schéma et exécutez le sketch de l'Exemple 7.8.

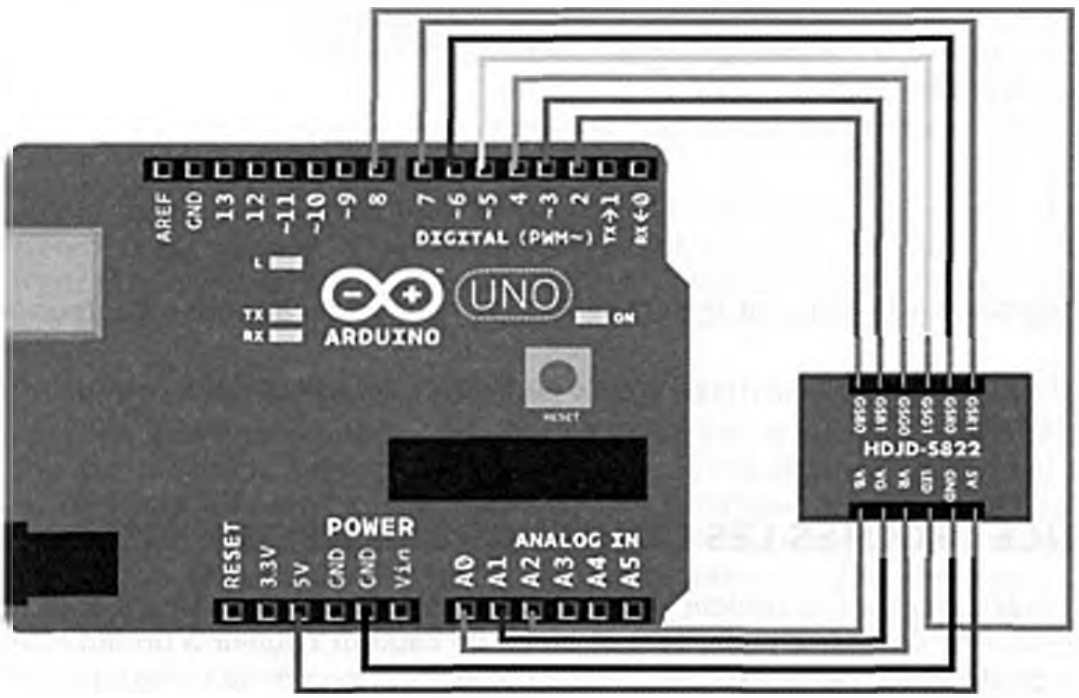


Figure 7.20 Circuit de capteur couleur pour Arduino

Exemple 7.8. color_sensor.ino

```
// color_sensor.ino - perçoit la couleur avec le HDJD-S822-QR999 et affiche
la valeur RGB
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int gsrlPin = 7;  // 1
```

```
const int gsr0Pin = 6;
const int gsg1Pin = 5;
const int gsg0Pin = 4;
const int gsb1Pin = 3;
const int gsb0Pin = 2;
const int ledPin = 8;    // 2
const int redPin = A0;   // 3
const int greenPin = A1;
const int bluePin = A2;
int red = -1;    // 4
int green = -1;
int blue = -1;
void setup() {
  Serial.begin(115200);
  pinMode(gsr1Pin, OUTPUT);
  pinMode(gsr0Pin, OUTPUT);
  pinMode(gsg1Pin, OUTPUT);
  pinMode(gsg0Pin, OUTPUT);
  pinMode(gsb1Pin, OUTPUT);
  pinMode(gsb0Pin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);    // 5
  digitalWrite(gsr1Pin, LOW);    // 6
  digitalWrite(gsr0Pin, LOW);
  digitalWrite(gsg1Pin, LOW);
  digitalWrite(gsg0Pin, LOW);
  digitalWrite(gsb1Pin, LOW);
  digitalWrite(gsb0Pin, LOW);
}
void loop() {
  int redValue = analogRead(redPin);     // 7
  int greenValue = analogRead(greenPin);
  int blueValue = analogRead(bluePin);
  redValue = redValue * 10 / 1.0;  // 8
  greenValue = greenValue * 10 / 0.75;
  blueValue = blueValue * 10 / 0.55;
  Serial.print(redValue); Serial.print(" ");     // 9
  Serial.print(greenValue); Serial.print(" ");
  Serial.print(blueValue); Serial.println(" ");
  delay(100);
}
```

1 Spécifie les broches de sélection du gain. Pour calibrer les couleurs individuelles, vous pouvez ajouter plus tard du gain au rouge (gsr1Pin, gsr0Pin), au vert (gsg1Pin, gsg0Pin), ou au bleu (gsb1Pin, gsb0Pin) si nécessaire. De cette manière, vous pouvez augmenter la sensibilité des couleurs individuelles en cas de besoin.

2 Cette LED qui éclaire la surface est critique pour l'exploitation de ce capteur ; elle est intégrée au capteur.

3 Broches pour lire les niveaux de rouge, vert et bleu.

4 Variables pour les valeurs lues par le capteur. Les variables sont initialisées à des valeurs impossibles pour faciliter le débogage (si vous voyez ces valeurs pendant l'exécution du code, c'est qu'il y a un problème).
5 Allume la LED intégrée du capteur pour éclairer la surface.
6 Désactive la sélection de gain pour toutes les couleurs. Chaque couleur a deux bits (0 et 1), ce qui autorise quatre niveaux (y compris la désactivation) de sélection de gain.
7 La lecture des valeurs est un simple appel à analogRead().
8 Les éléments des différentes couleurs ont une sensibilité différente. D'un autre côté, vous avez besoin que toutes les valeurs RGB soient à la même échelle si vous voulez recréer la couleur. Les facteurs de conversion (10 rouge, 14 vert, 17 bleu) ont été déduits de la notice technique du HDJD-S822-QR999.
9 Affiche chacune des valeurs RGB sur le Moniteur série (**Outils→Moniteur série**).

Capteur couleur pour Raspberry Pi

La Figure 7.21 illustre le circuit pour Raspberry Pi.

Montez les composants en suivant le schéma et exécutez le code de l'Exemple 7.9.

Des câbles en pagaille...

Est-ce que le circuit pour capteur couleur du Raspberry Pi vous fait criser ? Le nombre important de câbles facilite les erreurs de branchement.

Il y a deux solutions de base à ce problème :
– soit on simplifie le circuit,
– soit on combine un Arduino et un Raspberry Pi.

La solution qui a notre préférence est de combiner les deux plateformes.

Vous pouvez lire le capteur avec un Arduino, puis envoyer les valeurs RGB au Raspberry Pi en utilisant la liaison USB-série. Avec la fonction intégrée d'Arduino analogRead(), le circuit est simple (Figure 7.20). Puis on utilise l'USB pour connecter l'Arduino au Raspberry Pi.

En Raspberry Pi, vous pouvez lire les données série envoyées par le port USB avec pySerial.

Une troisième méthode pour réduire le câblage consiste à utiliser un convertisseur analogique-numérique doté de plus de canaux, comme le MCP3008. Pour employer une telle puce, il faut modifier la bibliothèque *botbook_mcp3002*.

Si vous ne comptez pas utiliser les broches de gain, vous pouvez tenter de les connecter à la masse, ce qui les conserve à LOW et désactive tous les gains. Cela peut aussi simplifier les connexions.

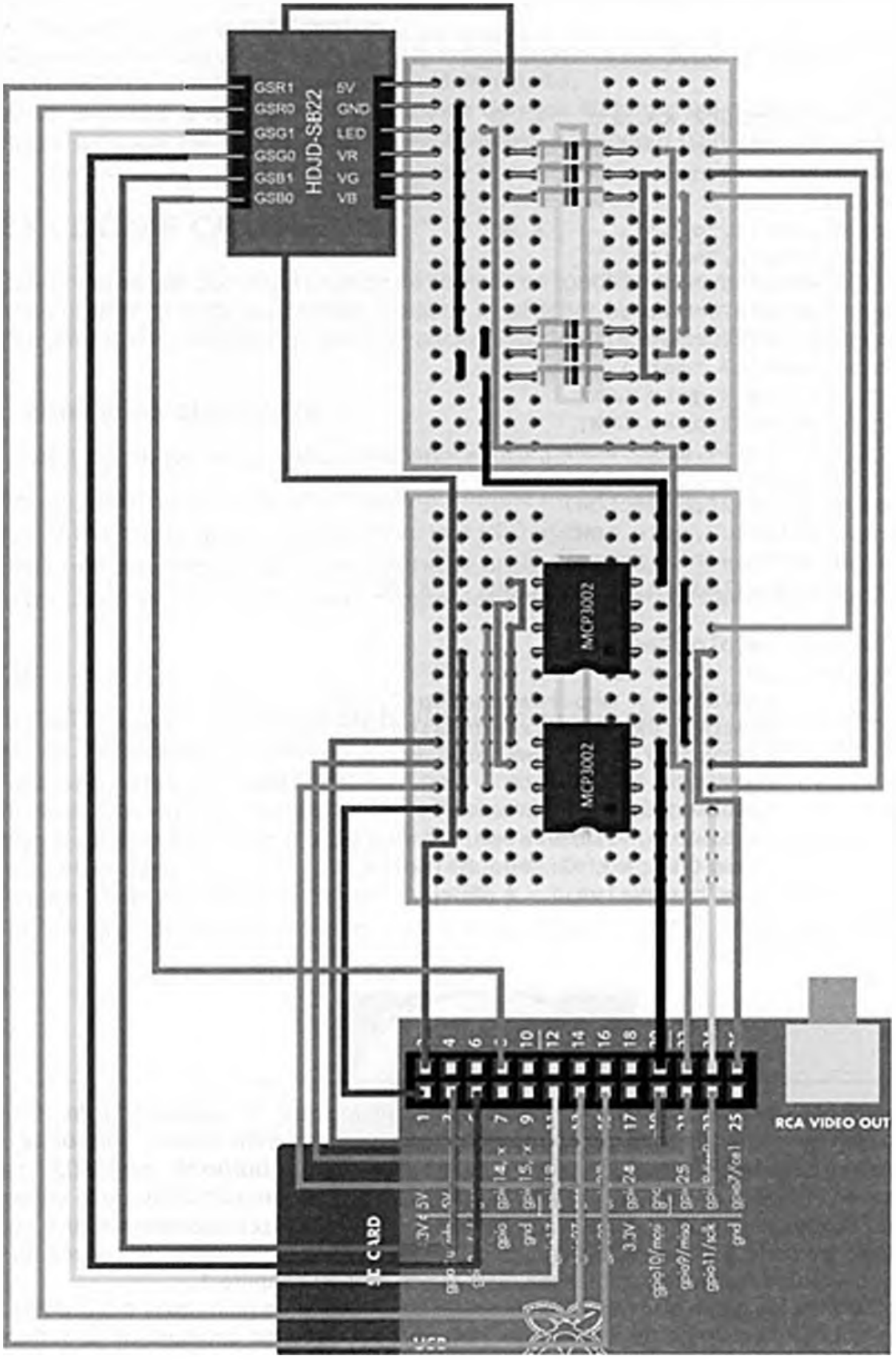


Figure 7.21 Le circuit du capteur couleur pour Raspberry Pi est complexe

Exemple 7.9. color_sensor.py

```
# color_sensor.py - perçoit la couleur et affiche les valeurs RGB
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp    # 1
import botbook_gpio as gpio
def initializeColorSensor():
        ledPin = 25
        gpio.mode(2,"out")       # 2
        gpio.mode(3,"out")
        gpio.mode(14,"out")
        gpio.mode(17,"out")
        gpio.mode(22,"out")
        gpio.mode(27,"out")
        gpio.write(2,gpio.LOW)
        gpio.write(3,gpio.LOW)
        gpio.write(14,gpio.LOW)
        gpio.write(17,gpio.LOW)
        gpio.write(22,gpio.LOW)
        gpio.write(27,gpio.LOW)
        gpio.mode(ledPin,"out")
        gpio.write(ledPin, gpio.HIGH)    # 3
def main():
        initializeColorSensor()
        while True:      # 4
                redValue = mcp.readAnalog(0, 0)
                greenValue = mcp.readAnalog(0, 1)
                blueValue = mcp.readAnalog(1, 0)          # 5
                redValue = redValue * 10 / 1.0; # 6
                greenValue = greenValue * 10 / 0.75;
                blueValue = blueValue * 10 / 0.55;
                print("R: %d, G: %d, B: %d" %
(redValue,greenValue,blueValue))  # 7
                time.sleep(0.1) # s
if __name__ == "__main__":
        main()
```

1 Les deux bibliothèques, *botbook_mcp3002.py* (analogique) et *botbook_gpio.py* (numérique), doivent se trouver dans le même répertoire que ce programme (*color_sensor.py*). Vous devez aussi installer la bibliothèque *spidev*, qui est importée par *botbook_mcp3002*. Lisez les commentaires au début du fichier *botbook_mcp3002/botbook_mcp3002.py* ou la section *Installation de SpiDev* au chapitre 3. Vous pouvez télécharger les deux bibliothèques ainsi que tous les exemples de code à partir de http://botbook.com. Pour la configuration de l'accès au GPIO sans avoir besoin de l'utilisateur root, lisez *GPIO sans root* au chapitre 1.

2 Désactive tous les gains en plaçant les broches de sélection de gain (gs*) à 0 (LOW).

3 Allume la LED d'éclairage du capteur qui s'en sert pour voir les couleurs de la surface qu'il mesure.

4 Le programme s'exécute tant que l'on n'appuie pas sur Ctrl+C.

```
5 Lit la deuxième (1, parce que l'on a commencé à partir de 0) puce MCP3002 du premier (0)
canal: readAnalog(device=1, channel=0). Les commandes précédentes utilisent une combinai-
son différente de puces et de canaux.
6 Égalise les valeurs de couleurs selon la notice technique du HDJD-S822-QR999. Après l'éga-
lisation, toutes les couleurs emploient la même échelle.
7 Crée la chaîne à afficher avec une chaîne de mise en forme qui n'accepte qu'un seul para-
mètre, si bien que les valeurs multiples sont placées à l'intérieur d'un tuple, (a, b).
```

PROJET : DÔME CAMÉLÉON

Notre projet final est un dôme qui change de couleur en fonction de la surface sur laquelle il repose. Nous allons utiliser le code du capteur couleur et afficher la couleur perçue avec une LED RGB. Quand l'ensemble est conditionné dans un packaging robuste, le résultat est très impressionnant.

Ce que vous allez apprendre

Dans le projet *Dôme caméléon*, vous allez apprendre à :

- Créer un appareil qui change de couleur comme un caméléon.
- Afficher n'importe quelle couleur avec une LED RGB.
- Utiliser une moyenne mobile pour filtrer le bruit aléatoire.
- Utiliser une fonction easing pour mapper les valeurs d'entrées avec les valeurs de sortie.

LED RGB

Une LED RGB (Figure 7.22) rassemble trois LED en un seul package. Elle ressemble à une seule LED, mais en mélangeant le rouge, le vert et le bleu, elle peut afficher n'importe quelle couleur.

L'œil humain a des cellules réceptrices pour trois couleurs : le rouge, le vert et le bleu. C'est la raison pour laquelle ces couleurs sont utilisées dans les téléviseurs et les autres écrans. La perception humaine a une caractéristique étrange (ou un *bug*) en ce sens qu'elle voit des combinaisons de fréquences comme une autre fréquence. Cela signifie que vous pouvez mélanger la lumière rouge avec du vert pour obtenir du jaune.

Une LED RGB a en général trois fils : un fil pour chaque couleur et un fil commun aux autres fils.

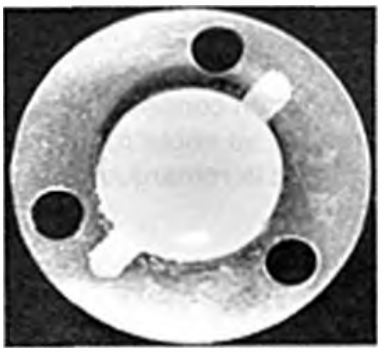


Figure 7.22 LED RGB

Contrairement à ce que vous pouvez penser, dans de nombreuses LED RGB, le fil commun est souvent positif. Un fil commun positif est également appelé anode commune. Comme le fil

commun est positif, vous devrez mettre chaque fil de couleur à LOW (0 V ou GND) pour faire briller chaque couleur.

Et la cathode commune ?

Si votre LED RGB a une cathode commune (le fil commun est négatif), vous aurez besoin de modifier le circuit et votre code. En ce qui concerne le circuit, il faut connecter le fil commun à la masse et non pas au +5 V. Dans le code, au lieu d'inverser la valeur d'intensité de la lumière par `255-couleur` (par exemple `255-rouge`) avec `analogWrite()`, vous devez supprimer la partie `255-` (par exemple `analogWrite(redPin, red)`).

Mais comment savoir quel fil est l'anode commune et quels sont les fils de chaque couleur ?

Déterminer les fils d'une LED RGB

Vous pouvez trouver les fils de manière expérimentale.

Allumez l'Arduino en le connectant au port USB. Comme vous allez utiliser simplement le +5 V et GND, le code qui s'exécute sur l'Arduino n'a pas d'importance.

> Nous utilisons une LED de 5 V. Si vous n'avez qu'une LED de moindre tension, vous pouvez utiliser une LED de 3,3 V quand vous testez les couleurs et une résistance quand vous la connectez à l'Arduino.

Sur l'Arduino, connectez un câble rouge au +5 V et un câble noir à la masse (GND).

Connectez les deux câbles à n'importe quel fil adjacent de la LED RGB.

Essayez de garder les fils adjacents autour de la LED jusqu'à ce qu'elle s'allume.

Testez les deux autres fils jusqu'à ce que vous allumiez les éléments rouge, vert et bleu de la LED RGB séparément. Notez que pour ce faire un fil (pour une LED à anode commune, ce sera le fil positif) devra rester connecté au câble rouge +5 V, pendant que vous connecterez les trois autres fils, chacun à leur tour, au câble noir de la masse.

Repérez le fil de l'anode commune (positif).

Si vous remarquez que le fil commun est plus long que les autres, mémorisez cette information pour le retrouver plus facilement plus tard. Sinon, marquez le fil commun avec un morceau de scotch et inscrivez quelque chose dessus (A pour anode, + pour positif, ou toute information permettant de vous souvenir).

> Si au lieu de cela, vous gardez un fil commun connecté au câble noir de la masse et devez connecter à tour de rôle chacun des trois autres fils au câble rouge du +5 V, c'est que vous avez une LED à cathode commune. Si tel est le cas, lisez la remarque *Et la cathode commune ?* un peu plus haut.

Marquez chacun des fils restants avec sa couleur correspondante (R, G ou B).

RGB pour Arduino

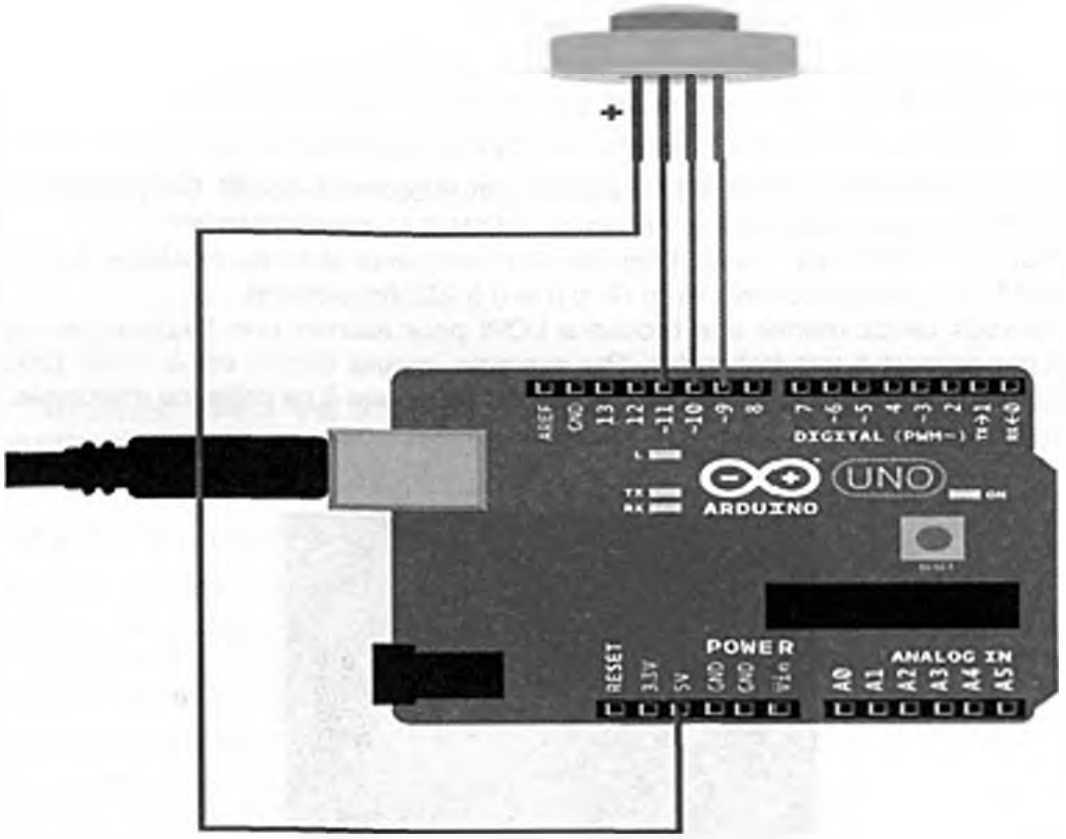


Figure 7.23 LED RGB connectée à Arduino

Exemple 7.10. hellorgb.ino

```
// hellorgb.ino - mélange des couleurs avec une LED RGB
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int redPin=11;    // 1
const int greenPin=10;
const int bluePin=9;
void setup()
{
        pinMode(redPin, OUTPUT);
        pinMode(greenPin, OUTPUT);
        pinMode(bluePin, OUTPUT);
}
void loop()
{
        setColor(255, 0, 0);    // 2
        delay(1000);
        setColor(255, 255, 255);
        delay(1000);
}
void setColor(int red, int green, int blue)
```

```
{
        analogWrite(redPin, 255-red);    // 3
        analogWrite(greenPin, 255-green);
        analogWrite(bluePin, 255-blue);
}
```

1 Comme de nombreuses LED RGB, celle-ci a un fil commun positif. Cela signifie que toutes les broches de données devront être négatives (LOW, 0 V) pour s'allumer.
2 Pour allumer la LED RGB, il suffit d'appeler setColor() avec la couleur désirée. Les paramètres sont R, G et B, où chaque couleur varie de 0 (rien) à 255 (maximum).
3 Comme vous devez mettre une broche à LOW pour allumer une couleur, les choses sont inversées par rapport à vos habitudes. Par exemple, quand redPin est à HIGH (255), la LED rouge est à off. Quand redPin est à LOW (0), la LED rouge est à sa brillance maximale.

Figure 7.24 Prototype de robot changeant de couleur

Vous voulez seulement réaliser votre Dôme caméléon ? Sautez les explications mathématiques et allez directement à la section *Assemblage des programmes*.

Moyenne mobile

Le Dôme caméléon est joli à regarder quand il modifie en douceur ses couleurs, mais il aurait l'air affreux s'il passait brusquement d'une couleur à une autre. C'est cependant le danger quand on travaille avec des capteurs : parfois un détail minuscule peut faire en sorte qu'un rouge apparaisse vert pour le capteur.

Le bruit aléatoire est un problème courant dans toute mesure et les capteurs ne font pas exception à ce phénomène.

Prenons l'exemple de la mesure d'un arbuste. Nous avons réalisé plusieurs mesures pour réduire le risque d'erreur et nous avons finalement obtenu les valeurs suivantes :

```
102 cm, 100 cm, 180 cm, 103 cm, 105 cm
```

La plupart d'entre nous considéreraient la mesure de 180 cm comme une erreur de saisie et, heureusement, l'ordinateur peut réaliser cela à notre place.

De prime abord, vous pensez sans doute à stocker les valeurs dans un tableau, puis à calculer la moyenne de chacune des valeurs du tableau. Même si cela fonctionne, cela nécessite beaucoup de code pour une si petite chose : un tableau, un pointeur, une moyenne et des calculs à chaque itération.

La moyenne mobile vient à notre secours ! Vous pouvez calculer la moyenne de la valeur en cours et de la valeur précédente pour obtenir une moyenne lissée. Pour traiter plus de données sans utiliser de tableau, vous pouvez employer une moyenne mobile pondérée.

Vous pouvez donner une pondération de 70% à la nouvelle entrée, en laissant 30% (100% moins 70%) aux anciennes valeurs.

```
entrée = 0.7*entrée + 0.3*anciennes
anciennes = entrée
```

Comme l'ancienne valeur est affectée par les anciennes données, vous n'avez pas besoin d'utiliser un tableau.

La LED RGB affiche n'importe quelle couleur

Pour obtenir une grande variété de couleurs, vous avez besoin de mélanger du rouge, du vert et du bleu dans des proportions variables. Si vous définissez les trois couleurs à leur brillance maximale, vous obtenez quelque chose qui se rapproche de la lumière blanche. Si vous définissez les trois couleurs à leur brillance minimale, vous n'obtenez rien.

Arduino n'est pas capable de véritablement baisser l'intensité d'une LED, car en pratique on ne peut pas faire baisser une LED. Si vous diminuez la tension, la brillance va diminuer, mais si vous abaissez suffisamment la tension, la LED va tout simplement s'éteindre (en général bien avant d'atteindre le 0 V).

Pour contourner ce problème, Arduino utilise la modulation de largeur d'impulsion (on utilise volontiers l'acronyme anglais PWM, pour *Pulse Width Modulation*, qui est parfois aussi traduit par modulation d'impulsion en durée). Pour faire en sorte qu'une LED paraisse avoir 10% de sa brillance, la PWM va utiliser un cycle de service de 10%. Cela signifie que l'Arduino va garder la LED allumée pendant 10% du temps, et éteinte pendant 90% du temps. Mais l'Arduino allume et éteint la LED si rapidement (chaque cycle dure 2 millisecondes ou 2 000 microsecondes) que vous ne remarquez pas le clignotement. Lors d'un cycle de service de 10%, Arduino va allumer la LED pendant 200 microsecondes, l'éteindre pendant 1 800 microsecondes, puis recommencer le cycle d'allumage et d'extinction.

Vous n'avez cependant pas à vous inquiéter des détails dans votre code. Vous pouvez faire comme si vous vous contentiez d'envoyer une plage de tension (de 0 à 255) qui correspond à différents niveaux de brillance. Comme le fil commun est à HIGH et que chaque fil de couleur est à sa brillance maximale quand vous mettez le fil à LOW, vous aurez la couleur rouge la plus brillante avec analogWrite(redPin, 0).

Pour toutes les valeurs comprises entre le minimum et le maximum, on utilise la valeur de la couleur (comme le rouge, r) soustraite de 255 :

```
analogWrite(redPin, 255-r);
```

Pour connaître les couleurs de base, consultez le Tableau 7.1.

Vous pouvez obtenir toutes les couleurs de l'arc-en-ciel et il vous suffit d'expérimenter des mélanges de lumière rouge, verte et bleue.

Nom	Couleur RGB	Valeurs des broches de données	Commentaire
Noir	(0,0,0)	(255,255,255)	Toutes les couleurs sont éteintes
Rouge	(255,0,0)	(0,255,255)	
Vert	(0,255,0)	(255,0,255)	
Bleu	(0,0,255)	(255,255,0)	
Blanc	(255,255,255)	(0,0,0)	Brillance max. de toutes les LED
(formule)	(r,g,b)	(255-r, 255-g, 255-b)	

Tableau 7.1 Valeurs des couleurs et des broches des LED RGB

Traitement de l'entrée avec une fonction easing

Les entrées et les sorties peuvent avoir différents types de valeurs que votre code doit donc convertir.

Dans le cas le plus simple, la sortie n'est que l'entrée multipliée par un nombre. Dans ce cas, la conversion n'est qu'une affaire de multiplication.

Par exemple, la fonction Arduino analogRead() a une plage allant de 0 à 1 023, mais la plage de analogWrite() ne va que de 0 à 255. Pour convertir ces deux plages, on doit d'abord calculer le pourcentage, p, de la valeur d'entrée maximale qu'une valeur d'entrée donnée (in) représente :

```
p = in/1023
```

Alors, la sortie mappée est p pourcent de la valeur maximale de la sortie :

```
out = p•255
```

La bibliothèque Arduino a même une fonction pour réaliser cela :

```
out = map(in, 0, 1023, 0, 255)
```

Des exemples de conversion linéaire sont listés dans le Tableau 7.2.

Entrée analogRead()	Pourcentage	Sortie analogWrite()
0	0.0 %	0.0
234	23 %	58
511	50 %	127
1023	100 %	255

Tableau 7.2 Mappage linéaire de l'entrée vers la sortie avec la fonction map()

Cependant, certaines sorties ne fonctionnent pas bien avec une sortie qui augmente de manière linéaire, si bien que vous devez utiliser une fonction easing. La LED RGB de ce projet est un bon exemple de sortie qui fonctionne mieux avec une fonction easing.

La plupart des mélanges de couleurs de LED RGB ont besoin de valeurs faibles. Quand on approche de la plage supérieure des valeurs de sortie, tout devient blanc et les couleurs individuelles sont difficiles à discerner.

Une fonction easing mappe les entrées vers les sorties d'une manière non linéaire. Pour une LED RGB, une fonction exponentielle convient parfaitement. Sinon, la plupart des valeurs deviendraient un blanc éclatant au lieu de couleurs comme l'orange ou le violet.

On calcule d'abord le pourcentage de l'entrée :

```
p = in/1023
```

Puis on crée la sortie de manière non linéaire. Les deux extrémités, le 0 en bas et le 255 au sommet, sont toujours représentées dans la plage des valeurs possibles. Le Dôme caméléon utilise une fonction exponentielle à titre de fonction easing :

$$out = 255 \cdot p^4$$

Comme le pourcentage p a un maximum de 1,0 (100%), les valeurs de l'exposant p^4 sont toujours comprises entre 0,0 (0%) et 1,0 (100%). La fonction exponentielle crée la classique figure en forme de canne de hockey. Vous pouvez voir des exemples de valeurs mappées dans le Tableau 7.3.

p	p**4	analogWrite()	Commentaire
0%	0.0%	0	minimum
	20%	0.2%	0
	40%	2.6%	6
	50%	6.2%	15
moitié	60%	13.0%	33
	80%	41.0%	104
	90%	65.6%	167
	100%	100.0%	255

Tableau 7.3 Fonction easing avec une fonction exponentielle

Les fonctions easing sont aussi utilisées dans l'animation.

Quand les objets se déplacent puis s'arrêtent, les vitesses d'accélération et de décélération sont calculées à l'aide d'une fonction easing.

Assemblage des programmes

Le Dôme caméléon combine une LED RGB avec le capteur couleur étudié plus haut dans la section *Capteur couleur pour Arduino*.

Connectez le capteur couleur et la LED RGB à l'Arduino selon le schéma de la Figure 7.27.

Les câbles du connecteur Dupont (Figure 7.25) combinés avec un ScrewShield facilitent grandement la vie quand vous devez utiliser beaucoup de broches (Figure 7.26).

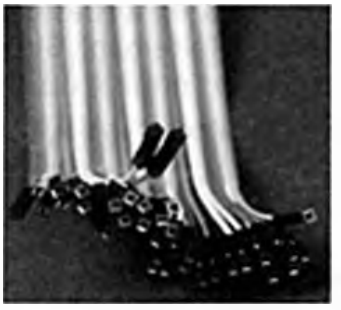


Figure 7.25 Câble de connecteur Dupont

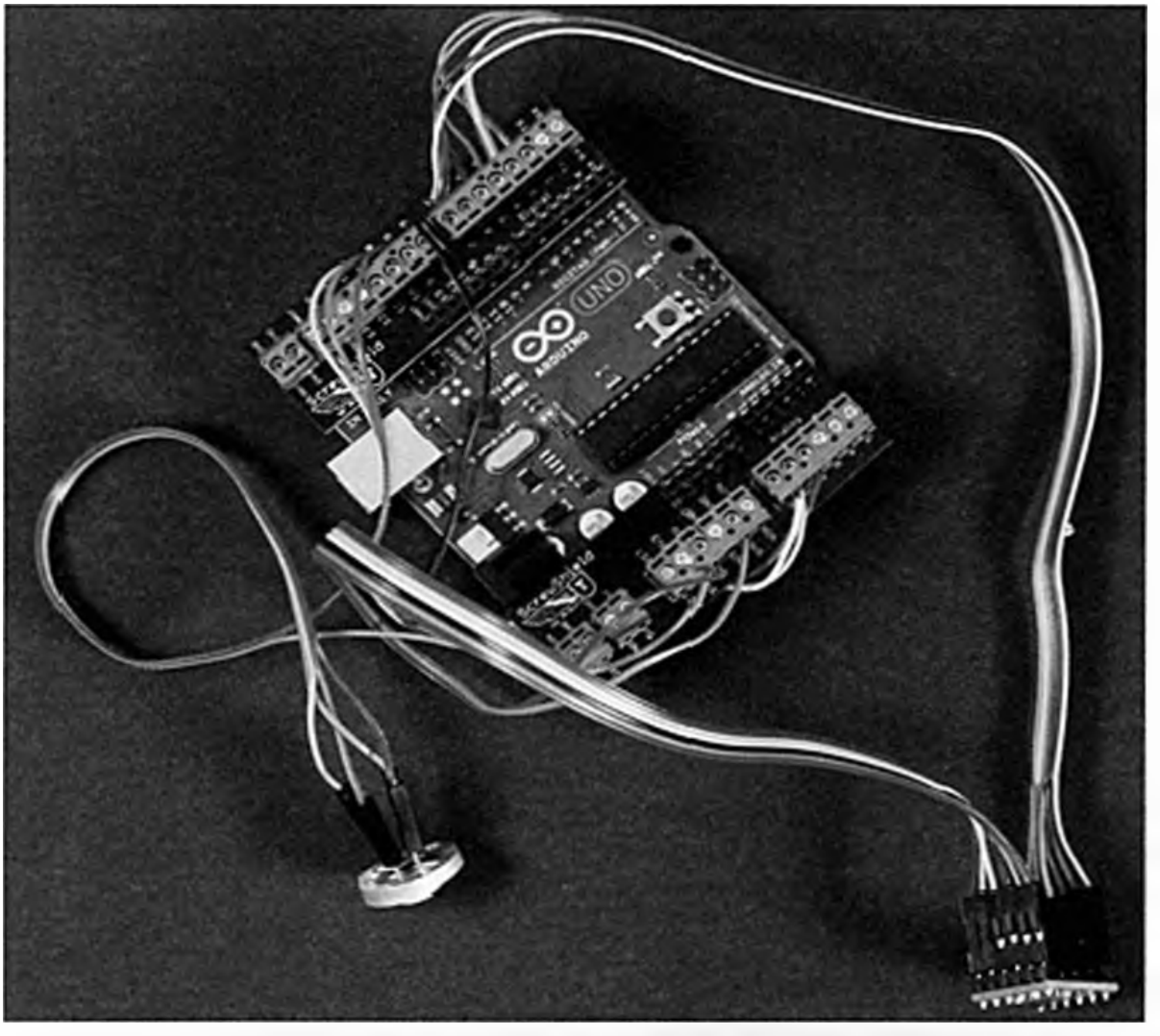


Figure 7.26 Capteur couleur connecté à une LED RGB

Le sketch du Dôme caméléon est listé dans l'Exemple 7.11. Après avoir réalisé toutes les connexions (Figure 7.27), exécutez ce sketch sur l'Arduino.

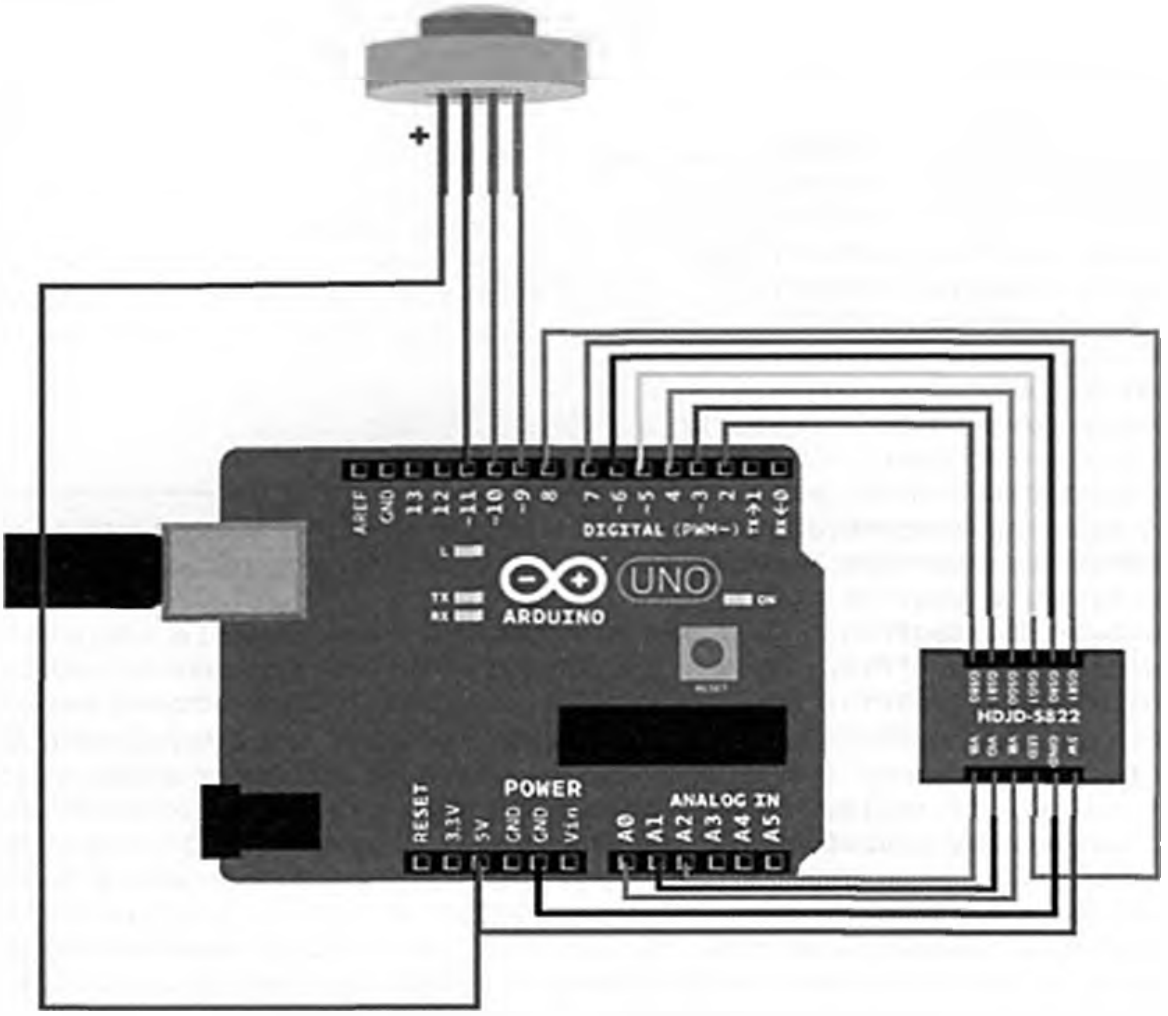


Figure 7.27 Connexions du Dôme caméléon

Exemple 7.11. chameleon_cube.ino

```
// chameleon_dome.ino - le cube change de couleur en fonction de la surface
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int gsr1Pin = 7;  // 1
const int gsr0Pin = 6;
const int gsg1Pin = 5;
const int gsg0Pin = 4;
const int gsb1Pin = 3;
const int gsb0Pin = 2;
const int ledPin = 8;   // 2
const int redInput = A0;        // 3
const int greenInput = A1;
const int blueInput = A2;
const int redOutput = 11;       // 4
const int greenOutput = 10;
```

```
const int blueOutput = 9;
int red = -1;   // 5
int green = -1;
int blue = -1;
const float newWeight = 0.7;     // 6
void setup() {
  Serial.begin(115200);
  pinMode(gsr1Pin, OUTPUT);
  pinMode(gsr0Pin, OUTPUT);
  pinMode(gsg1Pin, OUTPUT);
  pinMode(gsg0Pin, OUTPUT);
  pinMode(gsb1Pin, OUTPUT);
  pinMode(gsb0Pin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(redOutput, OUTPUT);
  pinMode(greenOutput, OUTPUT);
  pinMode(blueOutput, OUTPUT);
  digitalWrite(ledPin, HIGH);   // 7
  digitalWrite(gsr1Pin, LOW);
  digitalWrite(gsr0Pin, LOW);
  digitalWrite(gsg1Pin, LOW);
  digitalWrite(gsg0Pin, LOW);
  digitalWrite(gsb1Pin, LOW);
  digitalWrite(gsb0Pin, LOW);
}
void loop() {
  int redValue = analogRead(redInput);  // 8
  int greenValue = analogRead(greenInput);
  int blueValue = analogRead(blueInput);
  redValue = redValue * 10 / 1.0;
  greenValue = greenValue * 10 / 0.75;
  blueValue = blueValue * 10 / 0.55;
  redValue = map(redValue, 0, 1023, 0, 255);     // 9
  greenValue = map(greenValue, 0, 1023, 0, 255);
  blueValue = map(blueValue, 0, 1023, 0, 255);
  if(redValue > 255) redValue = 255;     // 10
  if(greenValue > 255) greenValue = 255;
  if(blueValue > 255) blueValue = 255;
  red = runningAverage(redValue, red);  // 11
  green = runningAverage(greenValue, green);
  blue = runningAverage(blueValue, blue);
  Serial.print(red); Serial.print(" ");
  Serial.print(green); Serial.print(" ");
  Serial.print(blue); Serial.println(" ");
  if(red < 200 || green < 180 || blue < 180) {
    green = green - red * 0.3;  // 12
    blue = blue - red * 0.3;
  }
  red = easing(red);     // 13
  green = easing(green);
  blue = easing(blue);
  setColor(red,green,blue);     // 14
```

```
  delay(100);
}
int runningAverage(int input, int old) {
  return newWeight*input + (1-newWeight)*old;    // 15
}
int easing(int input) { // 16
  float percent = input / 255.0f;
  return 255.0f * percent * percent * percent * percent;
}
int setColor(int r, int g, int b) {      // 17
  analogWrite(redOutput, 255-r);         // 18
  analogWrite(greenOutput, 255-g);
  analogWrite(blueOutput, 255-b);
}
```

1 Ce sont les broches de gain de chaque couleur. Ce code n'utilise pas le gain, mais initialise toutes les broches de gain à LOW. Si vous décidez de les utiliser, reportez-vous à l'Exemple 7.8 pour les détails.

2 Broche pour la LED du capteur qui éclaire la surface.

3 Broches d'entrées pour lire les valeurs RGB.

4 Broches de sortie des LED RGB.

5 Variables globales pour stocker les valeurs des couleurs manipulées. Elles sont initialisées avec des valeurs impossibles pour faciliter le débogage (si vous voyez ces valeurs apparaitre lors de l'exécution du code, c'est qu'il y a un problème).

6 Valeur de pondération à appliquer à la nouvelle entrée (voir *Moyenne mobile*).

7 Éclaire la surface avant de prendre la mesure.

8 Lit la couleur rouge. analogRead() retourne une valeur entière brute comprise entre 0 et 1023.

9 Mappe les valeurs (0..1023) analogRead() vers les valeurs (0..255) analogWrite().

10 11 Comme les valeurs de couleur sont égalisées (par exemple, le rouge est multiplié par 10), elles peuvent devenir supérieures à la valeur maximale utilisée pour le mappage. Le résultat peut être une valeur de rouge supérieure à 255. Plus tard, vous allez cependant contrôler la LED avec analogWrite(), si bien que vous devez limiter la valeur à 255.

12 Quelques modifications de couleurs esthétiques, avec des valeurs trouvées de façon empirique.

13 Rend les modifications de couleurs un peu moins rapides, ce qui évite de passer d'un coup du rouge au vert.

14 Définit la LED RGB avec une valeur calculée. L'utilisateur peut maintenant apprécier le résultat.

15 Une moyenne mobile permet de lisser le bruit aléatoire de l'entrée. Voir *Moyenne mobile*.

16 Easing est un terme d'animation. Les animations Flash ou JavaScript qui déplacent des choses en accélérant ou en décélérant sont souvent réalisées avec des fonctions easing. Ici, la fonction rend les valeurs inférieures à 255 (100%). Les valeurs plus petites sont plus affectées. Le but est de rendre le changement de couleur plus lent et sans à-coups.

17 Modifie la couleur de la LED RGB avec la valeur RGB donnée. Les paramètres sont des entiers dans la plage de 0 à 255.

18 Pour une anode commune (le fil commun est positif), les broches de données sont les broches négatives, et les valeurs doivent être inversées. Voir *La LED RGB affiche n'importe quelle couleur*.

Conseils pour la réalisation du dôme

Nous avons trouvé un parfait boîtier pour notre Dôme caméléon chez IKEA. Avec quelques menus changements, la lampe Solvinden convient parfaitement. Bien entendu, vous pouvez utiliser n'importe quel boîtier translucide ou dôme qui vous convienne. Il existe un choix immense de lampes différentes et vous pouvez même choisir un emballage de produit surgelé.

Commencez par ouvrir la lampe Solvinden (Figure 7.28) et enlevez le couvercle (Figure 7.29).

Figure 7.28 Commencez par enlever le dôme

Figure 7.29 Ouvrez le couvercle

Ôtez la partie électronique pour faire de la place pour notre gadget (Figure 7.30).

Figure 7.30 Supprimez l'électronique

Il y a une pointe en plastique au centre du fond et vous devez la couper (Figure 7.31). Vous devez percer deux trous, l'un de 19 mm pour que le capteur puisse voir ce qu'il y a dessous, et un autre de 3 mm pour fixer l'Arduino et la LED RGB (Figure 7.32).

Utilisez de la colle à chaud pour fixer le capteur qui pointe vers le bas et assurez-vous qu'il est centré sur le trou comme cela est illustré à la Figure 7.33.

Figure 7.31 Coupez la petite pointe au centre

Figure 7.32 Percez un trou de 19 mm pour le capteur et un trou de 3 mm pour qu'une vis maintienne en place l'Arduino et la LED RGB

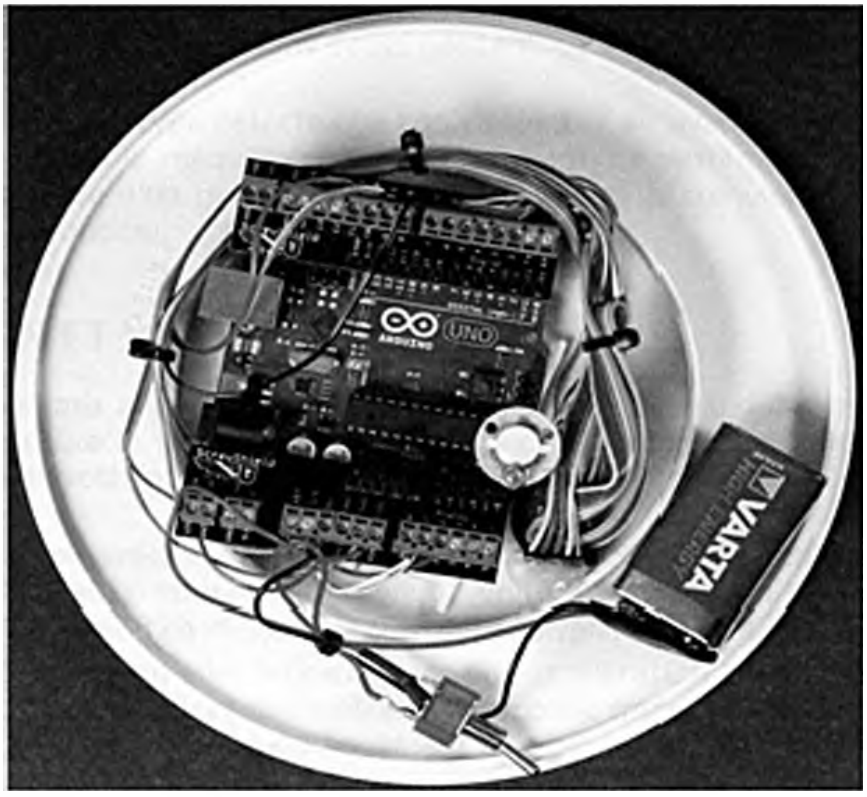


Figure 7.33 Fixation de l'électronique du Dôme caméléon

Utilisez une vis de 3 mm pour sécuriser l'Arduino sur le fond (Figure 7.33). Au sommet de la même vis qui maintient l'Arduino, placez la LED RGB. Notre LED a déjà des trous, mais on a besoin d'en percer un plus gros pour qu'elle tienne dans la vis de 3 mm.

Placez à présent la pile dans son logement, mettez le courant à l'aide de l'interrupteur, fermez le dôme et appréciez votre Dôme caméléon (Figure 7.34).

Figure 7.34 Le dôme est prêt à fonctionner avec son électronique embarquée

Vos appareils peuvent maintenant voir la lumière de nombreuses façons : ils peuvent détecter sa présence et sa direction, mesurer son intensité et même déterminer sa couleur.

8 Mesurer l'accélération

Quand vous inclinez votre smartphone, son écran passe probablement du mode portrait au mode paysage. Comment fait-il ? Les smartphones ont un accéléromètre intégré, et comme il n'est pas possible de distinguer la pesanteur de l'accélération vers le bas, alors l'attraction vers le bas de la pesanteur indique au téléphone le sens dans lequel il est orienté.

Certains jeux sont contrôlés par l'inclinaison d'un téléphone dont on utilise l'accéléromètre. La console de jeux Wii utilise un accéléromètre pour ses contrôleurs. Dans ce chapitre, nous allons détourner l'usage d'un contrôleur Wii pour nous servir de ses capteurs.

Les disques durs que l'on trouve dans les portables et les ordinateurs de bureau peuvent à présent subir beaucoup de mauvais traitements. Un grand nombre d'entre eux sont certifiés pouvoir encaisser un choc de 150 g (quand ils ne sont pas en fonctionnement), accélération qui serait immédiatement fatale à n'importe quel être humain. Pour supporter l'impact, certains disques s'éteignent automatiquement : quand l'accéléromètre du disque dur détecte une chute, il déplace le bras articulé en dehors des zones sensibles.

Avez-vous déjà essayé de monter sur un appareil qui s'équilibre tout seul, comme un Segway ou un Solowheel ? Après un départ sans doute hésitant, cela tient du miracle que l'appareil reste debout.

Quand un appareil de ce type détecte que l'on va tomber en avant, il déplace rapidement ses roues vers l'avant, ce qui le rééquilibre automatiquement. Le système d'équilibrage mesure la vitesse angulaire à l'aide d'un gyroscope (un accéléromètre accumulerait trop d'erreurs pour fonctionner sur un gyropode).

ACCÉLÉRATION ET VITESSE ANGULAIRE

L'accélération est le taux auquel change la vitesse d'un objet (quand il ralentit ou accélère) La vitesse angulaire mesure la vitesse de rotation d'un objet, ainsi que celle de l'axe sur lequel il tourne. En fonction de votre projet, vous pouvez avoir besoin d'accélération, de vitesse angulaire, voire des deux.

L'accélération est mesurée en g, qui est un multiple de l'accélération causée par la gravitation de la terre. On utilise aussi couramment comme unité d'accélération les mètres par seconde carrée (m/s^2). L'accélération en chute libre (1 g) est équivalente à 9,81 m/s^2.

Pourquoi les secondes sont-elles au carré dans une accélération ? L'accélération est un changement de vitesse. Si vous utilisez des mètres par seconde (m/s) comme unité de vitesse, alors on représente l'unité d'accélération (changement de vitesse) par des mètres par seconde par seconde, que l'on abrège en m/s^2.

Les gyroscopes mesurent la vitesse angulaire, c'est-à-dire la vitesse de rotation d'un capteur autour d'un axe. Par exemple, un gyroscope peut indiquer qu'il tourne de 10 degrés par seconde. Les gyroscopes sont utilisés dans les gyropodes et les compas gyroscopiques des avions.

Capteur	Mesure	Signification	Unité	Gravitation
Accéléromètre	Accélération	Changement de vitesse, accélération ou freinage	m/s / s = m/s^2	Oui, 1 g vers le bas
Gyroscope	Vitesse angulaire	Changement d'angle, rotation	rad/s (SI), souvent deg/s ou tours/min	Ignore la pesanteur

Tableau 8.1 Accéléromètre et gyroscope

EXPÉRIENCE : ON ACCÉLÈRE AVEC LE MX2125

Le MX2125 est un capteur d'accélération biaxial simple (Figure 8.1) qui indique l'accélération sous la forme d'une longueur d'impulsion, ce qui simplifie l'interface et le code.

Figure 8.1 Capteur MX2125

Dans la réalité du monde physique, il y a trois dimensions. Les objets peuvent se déplacer de haut en bas (y), de gauche à droite (x), et d'avant en arrière (z). Un capteur biaxial ne mesure que deux de ces axes.

> Le MX2125 ne mesure que jusqu'à 3 g par axe, alors que certains capteurs peuvent mesurer une accélération extrême. Par exemple, l'accélération maximale mesurée par le capteur ADXL377 (200 g) est bien supérieure à celle qui tuerait n'importe quel être humain. Il peut mesurer une accélération supérieure à celle ressentie lors du lancement d'une navette spatiale ou lors de brusques changements de trajectoires effectués par des avions de chasse. Quand nous avons créé un prototype de capteur solaire pour un satellite Aalto-1, même les spécifications du satellite n'exigeaient pas une accélération aussi forte.
>
> Il est fort peu probable que vous ayez à mesurer une telle accélération extrême, ce qui ne serait d'ailleurs pas possible avec une platine de test (car l'accélération à tester détruirait votre installation...). Notez cependant que son coût n'est pas prohibitif, mais plus la plage de l'accélération mesurée est grande (de -250 g à +250 g), moins le capteur est précis.

Décodage de la longueur d'impulsion du MX2125

Habituellement, les formules de conversion d'un accéléromètre se trouvent sur la notice technique. Dans notre cas, il faut effectuer quelques recherches.

COMMENT TROUVER LES NOTICES TECHNIQUES DES CAPTEURS ?

Quand on recherche une notice technique, il paraît évident de saisir dans la requête Internet le nom de code du composant suivi des mots « notice technique » (ou de son équivalent anglais *datasheet*). Mais vous pouvez également trouver la notice technique sur le site web où vous avez acheté le composant (c'est en général dans la fiche détaillée du produit).

Par exemple, si l'on saisit dans un moteur de recherche « MX2125 datasheet », on trouve la fiche du produit **Memsic 2125 Dual-axis Accelerometer** et dans la rubrique **Downloads & Documentation**, un document intitulé « Memsic MXD2125 Datashoot ».

Il arrive parfois que l'on ne trouve pas aussi facilement ce que l'on cherche et il faut alors employer des voies détournées. Par exemple, pour notre capteur, on peut aussi trouver l'information en saisissant le nom de code de la puce qui est embarquée sur la carte, à savoir « MXD2125 ».

Le MX2125 fonctionne en chauffant une bulle de gaz à l'intérieur de l'appareil, puis en mesurant la manière dont la bulle d'air se déplace.

Quand il est alimenté, le MX2125 signale l'accélération sur chaque axe, à un rythme de 100 impulsions par seconde. Les signaux consécutifs HIGH et LOW forment une onde carrée de 100 Hz. Plus il y a d'accélération, plus l'onde passe de temps dans la partie HIGH, et moins elle passe de temps dans la partie LOW. Vous pouvez lire ces impulsions pour déterminer l'accélération.

Une onde complète comprend un HIGH et un LOW. Le temps écoulé pendant une onde (HIGH+LOW) est appelé période (T). Nous appellerons tHIGH le temps de la partie HIGH.

Le cycle de service indique la quantité d'onde qui est à la valeur HIGH. Le cycle de service s'exprime sous la forme d'un pourcentage, par exemple 50% (0,50) ou 80% (0,80).

```
dutyCycle = tHIGH / T
```

Selon la notice technique et les autres documents, la période T est initialisée à 10 ms par défaut.

```
dutyCycle = tHIGH / 10 ms
```

Voici la formule d'accélération qui figure dans la notice technique :

```
A = (tHIGH/T-0.50)/20%
```

On peut aussi remplacer tHIGH/T par dutyCycle et 20% par .2 :

```
A = (dutyCycle-0.50)/.2
```

On peut maintenant l'écrire de la manière suivante (car x/.2 est égal à 5*x) :

```
A = 5*(dutyCycle-0.50)
```

ou :

```
A = 5*(tHIGH/T-0.50)
```

Quand il n'y a pas d'accélération (0 g), le cycle de service est de 50% :

```
0    = 5*(dutyCycle-0.50)
0/5  = dutyCycle-0.50
0    = dutyCycle-0.50
.50  = dutyCycle
```

Lors de notre première recherche, les documentations de Parallax et de Memsic différaient sur le multiplicateur : la documentation de Memsic employait 1/20% (5), et Parallax 1/12.5% (8). Lors de nos propres expériences, nous avons trouvé que 1/12.5% (8) donnait de bons résultats avec la platine d'évaluation de chez Parallax. En fait, quand nous avons consulté à nouveau la notice technique du MXD2125 de chez Memsic qui était hébergée sur le site web de Parallax, les deux documents indiquaient la même valeur, 1/12.5%. C'est la raison pour laquelle vous devez être prudent avec les notices techniques que vous trouvez sur Internet : vérifiez toujours les valeurs empiriquement. Dans notre exemple, nous utiliserons donc le multiplicateur 8 :

```
A = 8*(tHIGH/T-0.50)
```

Comme la fonction Arduino pulseIn() retourne la longueur d'impulsion en microsecondes (1 µs = 0,001 ms = 1e-6 s), la formule peut être modifiée pour utiliser les microsecondes :

```
A = 8 * ( tHIGH/(10*1000) - 0.5)
```

L'unité de A est alors g, qui est égal à 9,81 m/s^2.

Par exemple, si tHIGH est égal à 5 000 µs (5 ms), le cycle de service est :

```
dutyCycle = 5 ms / 10 ms = 0.50 = 50%
```

Ce qui équivaut à 0 g :

```
A = 8*(0.50-0.5) = 8*0 = 0  // g
```

Prenons un autre exemple avec une valeur de tHIGH à 6 250 µs (6,25 ms) :

```
A = 8*(6250/10000-0.5) = 8*(0.625-0.5) = 8*0.125 = 1  // g
```

Une impulsion de 6,25 ms équivaut donc à une accélération de 1 g.

Accéléromètre pour Arduino

La Figure 8.2 illustre le circuit pour Arduino. Câblez selon le schéma et chargez le code de l'Exemple 8.1.

Exemple 8.1. mx2125.ino

```
// mx2125.ino - mesure l'accélération sur deux axes avec le MX2125 et
affiche sur le port série
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int xPin = 8;
const int yPin = 9;
void setup() {
  Serial.begin(115200);
```

```
  pinMode(xPin, INPUT);
  pinMode(yPin, INPUT);
}
void loop() {
  int x = pulseIn(xPin, HIGH);  // 1
  int y = pulseIn(yPin, HIGH);
  int x_mg = ((x / 10) - 500) * 8;      // 2
  int y_mg = ((y / 10) - 500) * 8;
  Serial.print("Axes x : ");
  Serial.print(x_mg);
  Serial.print(" y : ");
  Serial.println(y_mg);
  delay(10);
}
```

1 La longueur de l'impulsion indique l'accélération. pulseIn() retourne la longueur d'impulsion en microsecondes (µs). 1 µs = 0,001 ms = 0,000001 s = 10^{-6} s.
2 Convertit la sortie en millig, un millième de la constante do gravitation g.

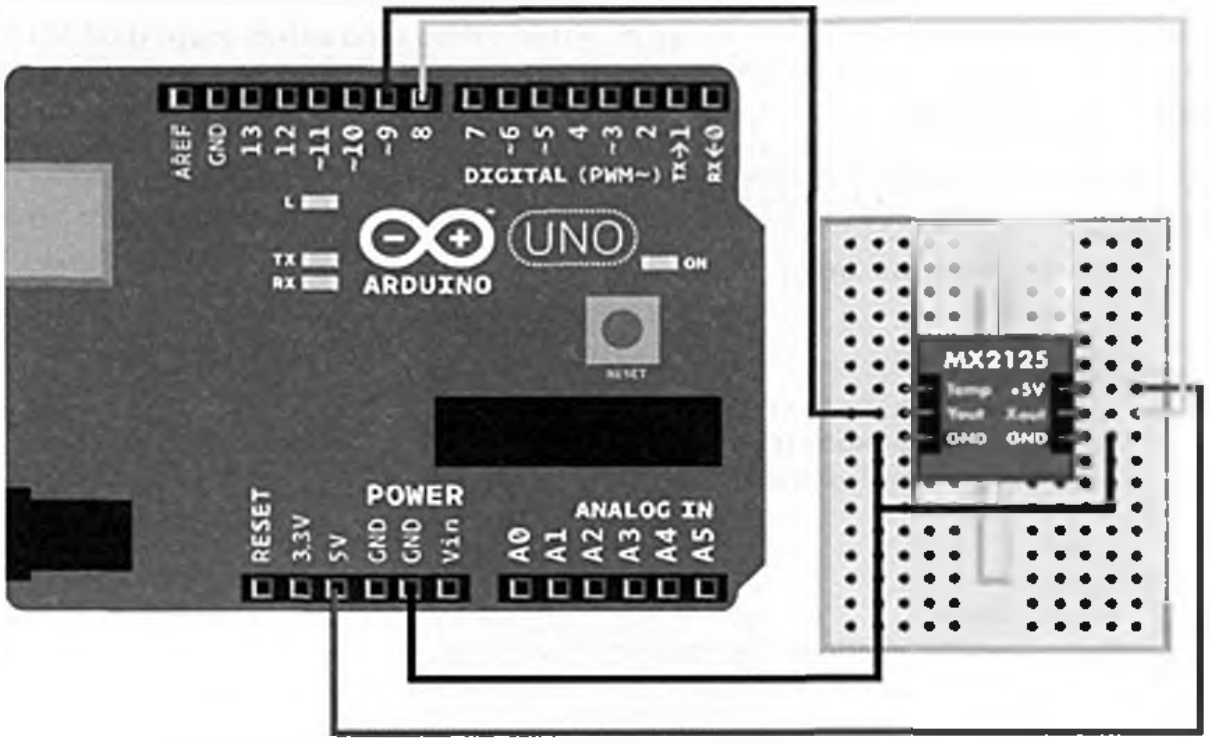


Figure 8.2 Circuit de l'accéléromètre biaxial MX2125 pour Arduino

Accéléromètre pour Raspberry Pi

La Figure 8.3 illustre le schéma de câblage pour Raspberry Pi. Branchez selon le schéma puis exécutez le code de l'Exemple 8.2.

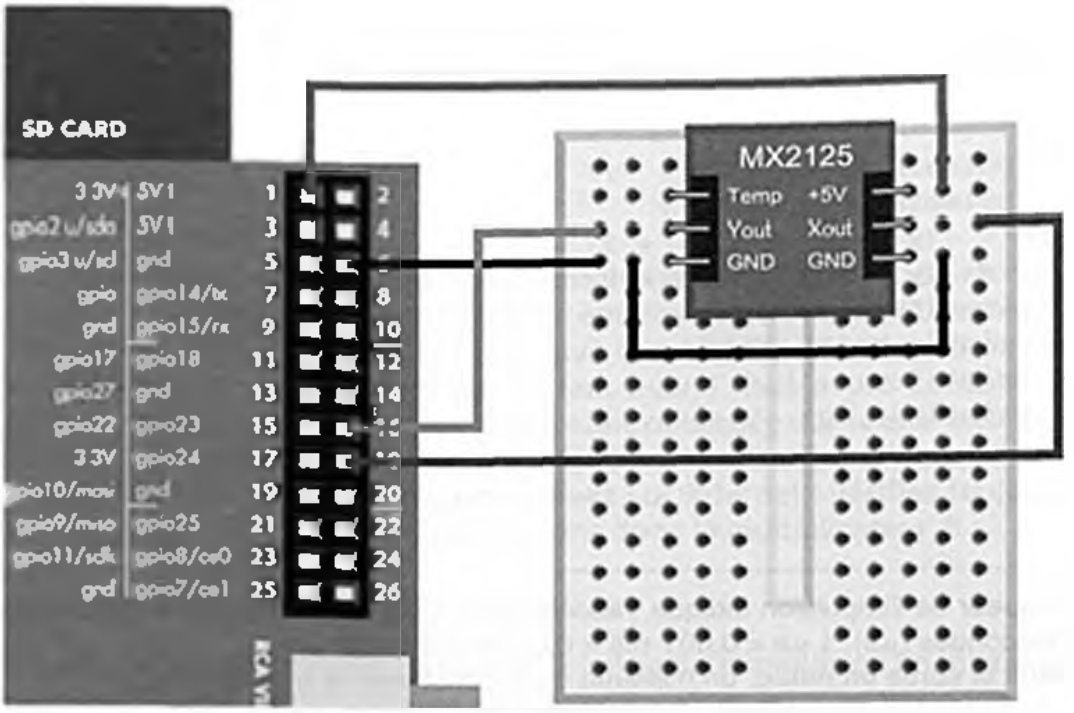


Figure 8.3 Circuit de l'accéléromètre biaxial MX2125 pour Raspberry Pi

Exemple 8.2. mx2125.py

```
# mx2125.py - affiche les valeurs de l'accélération des axes.
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio
xPin = 24
yPin = 23
def readAxel(pin):
        gpio.mode(pin, "in")
        gpio.interruptMode(pin, "both")
        return gpio.pulseInHigh(pin)    # 1
def main():
        x_g = 0
        y_g = 0
        while True:
                x = readAxel(xPin) * 1000
                y = readAxel(yPin) * 1000
                if(x < 10):     # 2
                        x_g = ((x / 10) - 0.5) * 8      # 3
                if(y < 10):
                        y_g = ((y / 10) - 0.5) * 8
                print ("Axes x : %fg, y : %fg" % (x_g, y_g))    #
                time.sleep(0.5)
if __name__ == "__main__":
        main()
```

1 Mesure la longueur d'impulsion, c'est-à-dire la durée pendant laquelle la broche est à HIGH.
2 Ignore les mesures qui sont bien en dehors de la plage.
3 Calcule l'accélération le long de l'axe x en g. Un g est l'accélération provoquée par la pesanteur de la terre, 9,81 mètres par seconde carrée.

EXPÉRIENCE : ON COMBINE UN ACCÉLÉROMÈTRE ET UN GYROSCOPE

Quand un accéléromètre ne bouge pas, il détecte la pesanteur et peut indiquer où se situe le bas. Un gyroscope peut donner l'orientation de manière fiable, même si vous le faites tourner sur lui-même, mais il ignore cependant la pesanteur.

Est-il possible de combiner les avantages d'un accéléromètre et d'un gyroscope ? La réponse est affirmative !

Une centrale inertielle combine plusieurs capteurs et (en option) des programmes pour obtenir des informations plus précises et plus fiables sur le déplacement. Dans cette expérience, vous allez travailler directement avec les fonctionnalités de base du MPU 6050.

En général, les centrales inertielles sont plus coûteuses et plus précises que de simples accéléromètres et gyroscopes. Elles utilisent aussi des protocoles de communication plus avancés, comme I2C, au lieu d'un simple protocole fournissant la largeur d'impulsion.

Le MPU 6050 (Figure 8.4) a un accéléromètre, un gyroscope et un microcontrôleur sur la même puce. Même si l'espace ne constitue pas un réel bonus quand on est au stade de prototypage sur la platine de test, il est plaisant de savoir que toutes ces fonctionnalités tiennent sur un petit composant monté en surface, juste au cas où vous manqueriez de place pour votre projet. Par exemple, nous avons réussi à faire tenir un prototype de capteur solaire dans un boîtier cubique de 10 cm de côté, le produit fini devant se placer dans une zone toute plate de 5 mm par 5 mm sur la surface d'un satellite.

Figure 8.4 MPU 6050

Le MPU 6050 utilise le protocole I2C. Grâce à la bibliothèque python-smbus, le code Raspberry Pi est bien plus simple et plus facile que le code équivalent pour Arduino. En général, Raspberry Pi gère les protocoles complexes plus simplement que l'Arduino.

PROTOCOLES STANDARD INDUSTRIELS

La plupart des appareils utilisent l'un des protocoles standard industriels au lieu d'inventer le leur.

I2C est l'un des protocoles les plus simples. Comme il est défini strictement et qu'il inclut la manière dont les données doivent être encodées et décodées, c'est en général le plus facile à utiliser. Vous trouverez plusieurs exemples d'usage d'I2C dans ce chapitre.

SPI est aussi un protocole courant, mais comme il laisse une grande latitude dans l'implémentation, ce peut être une tâche immense que d'écrire l'interface d'un nouveau composant SPI. D'un autre côté, s'il y a un exemple de code ou une implémentation de référence, il suffit de copier et de coller puisque quelqu'un d'autre a fait le travail difficile à votre place. Vous trouverez un exemple d'emploi d'un composant SPI sans référence d'implémentation à http://botbook.com/satellite.

On retrouve aussi souvent des formes déguisées du protocole série : série sur USB, série sur Bluetooth, série sur des câbles de liaison. Le bon vieux port série que vous utilisez avec le Moniteur série Arduino est en fait très courant. Il ne résout cependant qu'une partie du problème : comment envoyer des caractères par câble, mais le programmeur qui implémente doit toujours décider la manière dont les nombres sont encodés et décodés. Vous trouverez un exemple de détournement du port série sur USB dans notre ouvrage *Make a Mind-Controlled Arduino Robot*.

MPU 6050 pour Arduino

La Figure 8.5 illustre le schéma de câblage pour Arduino. Branchez en suivant le schéma et exécutez le code de l'Exemple 8.3.

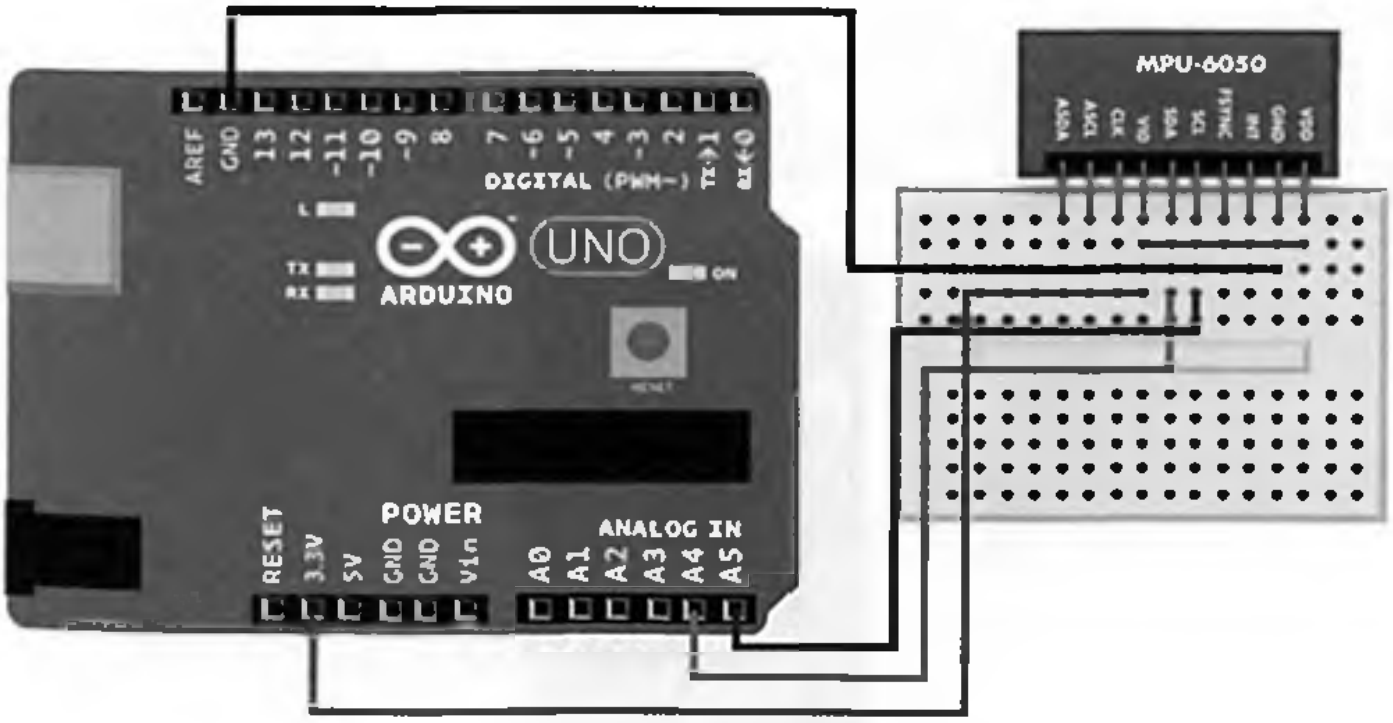


Figure 8.5 Circuit du MPU 6050 (accéléromètre + gyroscope) pour Arduino

Exemple 8.3. mpu_6050.ino

```
// mpu_6050.ino - affiche l'accélération (m/s**2) et la vitesse angulaire
(gyro, deg/s)
```

```
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
#include <Wire.h>        // 1
const char i2c_address =  0x68;          // 2
const unsigned char sleep_mgmt = 0x6B;  // 3
const unsigned char accel_x_out = 0x3B;
struct data_pdu // 4
{
  int16_t x_accel;       // 5
  int16_t y_accel;
  int16_t z_accel;
  int16_t temperature;  // 6
  int16_t x_gyro;       // 7
  int16_t y_gyro;
  int16_t z_gyro;
};
void setup() {
  Serial.begin(115200);
  Wire.begin(); // 8
  write_i2c(sleep_mgmt,0x00);   // 9
}
int16_t swap_int16_t(int16_t value)      // 10
{
  int16_t left = value << 8;     // 11
  int16_t right = value >> 8;    // 12
  right = right & 0xFF; // 13
  return left | right;  // 14
}
void loop() {
  data_pdu pdu; // 15
  read_i2c(accel_x_out, (uint8_t *)&pdu, sizeof(data_pdu));      // 16
  pdu.x_accel = swap_int16_t(pdu.x_accel);       // 17
  pdu.y_accel = swap_int16_t(pdu.y_accel);
  pdu.z_accel = swap_int16_t(pdu.z_accel);
  pdu.temperature = swap_int16_t(pdu.temperature);       // 18
  pdu.x_gyro = swap_int16_t(pdu.x_gyro);
  pdu.y_gyro = swap_int16_t(pdu.y_gyro);
  pdu.z_gyro = swap_int16_t(pdu.z_gyro);
  float acc_x = pdu.x_accel / 16384.0f; // 19
  float acc_y = pdu.y_accel / 16384.0f;
  float acc_z = pdu.z_accel / 16384.0f;
  Serial.print("Accéléromètre : x,y,z (");
  Serial.print(acc_x,3); Serial.print("g, ");   // 20
  Serial.print(acc_y,3); Serial.print("g, ");
  Serial.print(acc_z,3); Serial.println("g)");
  int zero_point = -512 - (340 * 35);   // 21
  double temperature = (pdu.temperature - zero_point) / 340.0;  // 22
  Serial.print("Température (C): ");
  Serial.println(temperature,2);
  Serial.print("Gyroscope : x,y,z (");
  Serial.print(pdu.x_gyro / 131.0f); Serial.print(" deg/s, ");  // 23
  Serial.print(pdu.y_gyro / 131.0f); Serial.print(" deg/s, ");
  Serial.print(pdu.z_gyro / 131.0f); Serial.println(" deg/s)");
```

```
  delay(1000);
}
void read_i2c(unsigned char reg, uint8_t *buffer, int size)     // 24
{
  Wire.beginTransmission(i2c_address);  // 25
  Wire.write(reg);       // 26
  Wire.endTransmission(false);  // 27
  Wire.requestFrom(i2c_address,size,true);       // 28
  int i = 0;     // 29
  while(Wire.available() && i < size) {  // 30
    buffer[i] = Wire.read();     // 31
    i++;
  }
  if(i != size) {        // 32
    Serial.println("Erreur de lecture i2c");
  }
}
void write_i2c(unsigned char reg, const uint8_t data)    // 33
{
  Wire.beginTransmission(i2c_address);  // 34
  Wire.write(reg);
  Wire.write(data);
  Wire.endTransmission(true);
}
```

1 Wire.h est la bibliothèque Arduino pour le protocole I2C. Wire.h est livrée avec l'IDE Arduino, de sorte que vous pouvez facilement l'inclure dans votre code. Vous n'avez pas besoin d'installer séparément la bibliothèque ou de copier des fichiers de bibliothèque manuellement.

2 Adresse I2C du capteur MPU 6050. Un bus de trois câbles peut avoir de nombreux esclaves. Chaque esclave est reconnu par son adresse. En général, un bus I2C peut avoir 128 (2^7) esclaves et les câbles I2C ne peuvent pas excéder deux mètres de long. Le nombre est écrit en hexadécimal (voir la section *Hexadécimal, binaire et autres systèmes de numération* dans ce chapitre pour une explication de cette notation).

3 Les registres des commandes se trouvent dans la documentation du MPU 6050. Si vous avez besoin d'une liste complète des commandes, cherchez sur le Web « notice technique MPU 6050 » et « MPU 6050 register map ». Les nombres sont en hexa mais ils peuvent être écrits en décimal si vous préférez.

4 Structure pour décoder la réponse du capteur. Une structure combine plusieurs valeurs. La structure C est réservée pour les données et ne peut pas contenir de fonctions. Cela différencie les structures des objets et des classes que l'on trouve dans les autres langages. struct data_pdu déclare un nouveau type de données que vous allez plus tard employer pour déclarer des variables de type data_pdu. Cette structure a des variables qui ont exactement la même taille que les champs de données du protocole utilisé par le capteur. Plus tard, vous allez lire les octets du capteur directement dans la structure. Ensuite, vous accéderez aux variables incorporées dans la structure pour obtenir les valeurs. Voici un mécanisme astucieux !

5 Variable pour stocker l'accélération sur l'axe horizontal x. Le type `int16_t` est un entier spécifiquement dimensionné et défini par avr-libc (la bibliothèque C utilisée par le compilateur Arduino). Il s'agit d'un entier signé (négatif ou positif) sur 16 bits (deux octets). Comme la structure est utilisée pour décoder les données brutes du capteur, les types de données correctement dimensionnés sont nécessaires.

6 Le capteur indique aussi la température. Même si vous n'en avez pas besoin, vous devez avoir une variable pour cela de telle sorte que la structure `data_pdu` soit à la bonne taille.

7 Vitesse angulaire autour de l'axe x, lu par le gyroscope du MPU 6050.

8 Initialise la communication I2C en utilisant la bibliothèque Wire.h.

9 Réveille le capteur en écrivant la commande 0 sur le registre de gestion de la veille 0x6B. Le MPU 6050 démarre en mode veille si bien qu'il s'agit d'une étape nécessaire.

10 Permute les deux octets de la valeur du paramètre. Le MPU 6050 est au format big endian alors que l'Arduino est au format little endian comme la plupart des processeurs. L'endianness[1] doit être convertie entre les plateformes. Voir plus bas la section *Endianness* pour des détails.

11 Cette nouvelle variable sur deux octets (16 bits) left finit à la place de l'octet de droite du paramètre value. Après un décalage à gauche (< <) d'un octet (8 bits), l'octet de gauche de value est abandonné car il ne tient pas dans les deux octets de left. L'octet de droite de left est comblé avec des zéros dans l'opération de décalage des bits.

12 Cette nouvelle variable sur deux octets (16 bits) right est à présent l'octet de gauche de value. L'octet de gauche de `right` est comblé avec des zéros.

13 Met à zéro l'octet de gauche de right, pour s'assurer simplement qu'il est vide. Voir aussi plus bas la section *Masquage de bits avec AND & au niveau des bits.*

14 Combine les octets de gauche et de droite. La variable left est composée en fait de deux octets (16 bits), avec l'octet de droite (8 bits) rempli par des zéros. Voir aussi plus bas la section *OR | au niveau des bits.*

15 Crée une nouvelle variable pdu de type data_pdu. Il s'agit du type structure créé précédemment.

16 Remplit la structure pdu avec les données du capteur. Le premier paramètre est le registre à lire (accel_x_out, 0x3B). Le second paramètre est une référence à la structure pdu. Il est passé en tant que référence, si bien que la fonction peut modifier la structure au lieu de retourner une valeur. Le dernier paramètre contient le nombre d'octets à lire. Vous pouvez utiliser la taille de la structure pour spécifier le nombre d'octets à lire.

17 Convertit le nombre au format big endian du capteur en format little endian utilisé par Arduino.

18 Vous pouvez faire référence aux variables de la structure avec nom_structure.var, comme dans pdu.temperature.

19 La valeur de l'accélération brute est convertie dans l'unité que l'on utilise en pratique (g). La pesanteur standard g est de 9,81 m/s^2. Le facteur de conversion provient de la notice technique. Pour avoir un résultat en virgule flottante (décimal), le diviseur doit être en virgule flottante.

20 Affiche l'accélération sur le Moniteur série. L'unité est g, l'accélération de la pesanteur. 1 g = 9,81 m/s^2.

21 Calcule le point zéro pour convertir la mesure brute en température en Celsius, grâce aux informations de la notice technique. La température varie de -40°C à +85°C. Une valeur brute de -512 indique 35°C. À partir de ce point, chaque changement de 1 °C est représenté par une valeur brute de 340. Ainsi, pour trouver la valeur brute du point du 0°C, on prend -512 et on soustrait le produit de 340 * 35 (on enlève la valeur brute de 35°C, soit 340 par degré C). Le calcul est -512 - (340 * 35), soit -12412. Mais au lieu d'écrire le résultat de la valeur calculée -12412, vous devez écrire le calcul indiqué dans la notice technique, de manière à ce que le code soit plus clair.

22 Convertit la mesure brute de température en Celsius, en utilisant la formule de la notice technique.

1. Certains traducteurs utilisent le terme « boutisme » pour traduire le concept d'*endianness*. Vous apprendrez tout sur l'origine et la signification de ce terme en consultant l'article suivant : http://fr.wikipedia.org/wiki/Endianness.

23 Affiche la vitesse angulaire mesurée par le gyroscope. Le facteur de conversion de degré/s est 1/131.0 selon la notice technique. Pour avoir un résultat en virgule flottante (décimal), le diviseur doit aussi être en virgule flottante.
24 Fonction pour lire la taille du nombre d'octets du registre point. Le résultat est écrit dans la structure, qui est représentée par le pointeur *buffer. En raison du pointeur, la valeur réelle de la structure est modifiée, au lieu de retourner une valeur.
25 Envoie la commande I2C à l'appareil (à l'adresse 0x69 du capteur MPU 6050).
26 Spécifie l'adresse du registre à lire. Dans ce programme, read_i2c() n'est utilisé que pour lire accel_x_out (0x3B).
27 Maintient la connexion ouverte, de telle sorte que l'on puisse lire les données des prochaines lignes.
28 Demande des octets de données du capteur. Plus haut, vous avez déclaré vouloir démarrer à partir du registre accel_x_out (0x3B). Le paramètre true (troisième argument, qui s'appelle stop dans la documentation Arduino) signifie que la connexion est fermée après la fin de la lecture, ce qui libère le bus I2C pour une future utilisation.
29 Déclare une nouvelle variable pour la future boucle while. La variable i de la boucle stocke le compteur du nombre d'octets lus. Ce compteur i est égal au nombre d'itérations parcourues par la boucle.
30 N'entre dans la boucle que s'il y a des octets à lire et que vous n'avez pas encore lu tous les octets demandés. De la façon dont read_i2c() est appelée dans ce programme, la variable size sera toujours de la longueur de la structure data_pdu.
31 Lit un octet (8 bits) et le stocke dans le buffer. En examinant la manière dont read_i2c() est appelée dans ce programme, nous pouvons parcourir les premières itérations. Le pointeur *buffer pointe sur le premier octet de pdu, qui est une structure de type data_pdu. À la première itération, i est égal à 0, si bien que buffer[i] pointe sur le premier octet de pdu. Comme pdu a été passé à la fonction avec un pointeur, le contenu réel de pdu (la variable du programme principal) est écrasé. Aucune valeur de retour n'étant nécessaire, le type de read_i2c() est void. Lors de la seconde itération, buffer[1] pointe sur le second octet de pdu. Cela continue pour la totalité du buffer (pdu). Quand i == size, on ne rentre pas à nouveau dans la boucle while, et l'exécution continue avec le code qui vient après la boucle while.
32 S'il n'y a pas suffisamment d'octets disponibles, la variable i de la boucle est inférieure à size. Comme i a été déclarée en dehors de la boucle, elle est disponible après la boucle.
33 Écrit un octet de données dans le registre reg du capteur.
34 L'adresse du capteur provient de la variable globale i2c_address.

Quel code difficile ! Le code pour le MPU 6050 contient plus de concepts de programmation complexes que la plupart des autres exemples de ce livre. Si vous trouvez que l'ordre des bits, le décalage des bits et les structures sont difficiles, vous pouvez simplement utiliser le code et jouer avec les valeurs. Il n'est pas obligatoire de comprendre le code en profondeur pour l'utiliser.

Si vous voulez comprendre le code, reportez-vous, plus loin dans ce chapitre, aux explications des sections *Hexadécimal, binaire et autres systèmes de numération* et *Opérations au niveau des bits*.

MPU 6050 pour Raspberry Pi

La Figure 8.6 illustre le schéma de câblage pour Raspberry Pi. Branchez selon le schéma puis exécutez le code de l'Exemple 8.4.

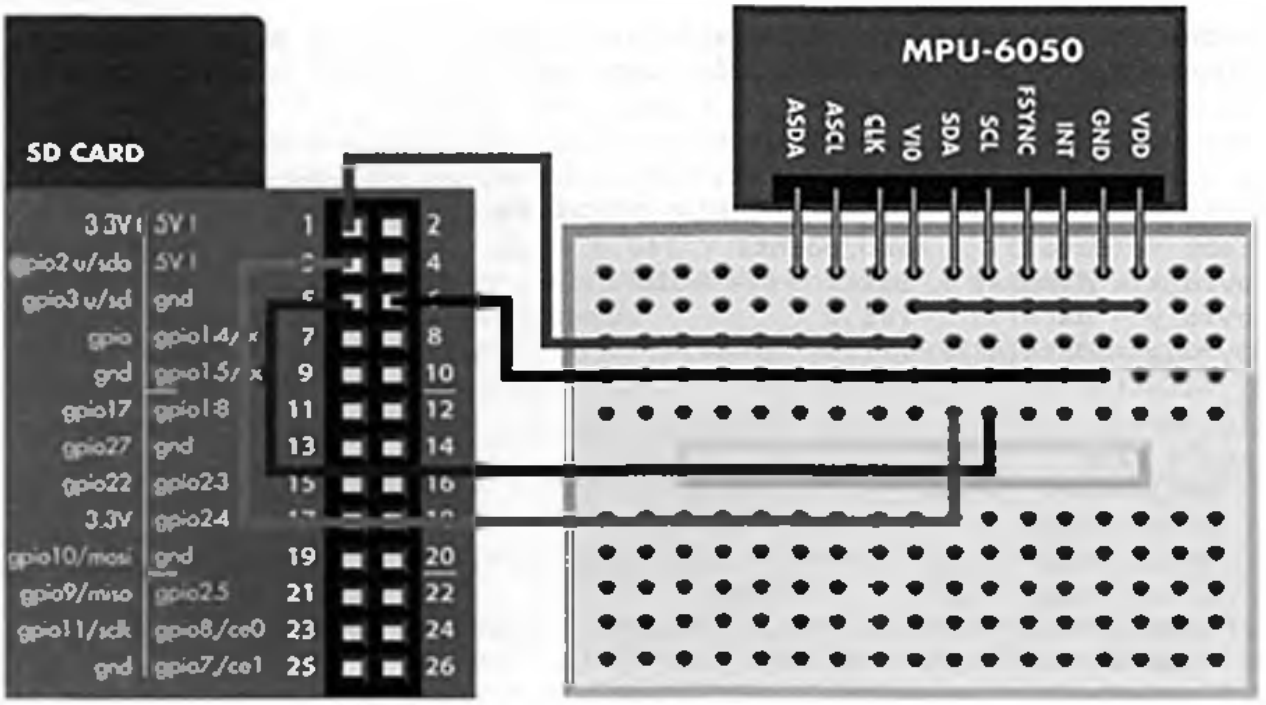


Figure 8.6 Circuit de l'accéléromètre MPU 6050 à six axes pour Raspberry Pi

Exemple 8.4. mpu_6050.py

```
# mpu_6050.py - affiche l'accélération (m/s**2) et la vitesse angulaire
(gyro, deg/s)
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import smbus # sudo apt-get -y install python-smbus  # 1
import struct
i2c_address = 0x68       # 2
sleep_mgmt = 0x6B        # 3
accel_x_out = 0x3B       # 4
bus = None       # 5
acc_x = 0
acc_y = 0
acc_z = 0
temp = 0
gyro_x = 0
gyro_y = 0
gyro_z = 0
def initmpu():
  global bus     # 6
  bus = smbus.SMBus(1)  # 7
  bus.write_byte_data(i2c_address, sleep_mgmt, 0x00)     # 8
def get_data():
  global acc_x,acc_y,acc_z,temp,gyro_x,gyro_y,gyro_z
  bus.write_byte(i2c_address, accel_x_out)      # 9
  rawData = ""
  for i in range(14):    # 10
```

```
    rawData += chr(bus.read_byte_data(i2c_address,accel_x_out+i)) # 11
  data = struct.unpack('>hhhhhhh', rawData)      # 12
  acc_x = data[0] / 16384.0      # 13
  acc_y = data[1] / 16384.0
  acc_z = data[2] / 16384.0
  zero_point = -512 - (340 * 35)          # 14
  temp = (data[3] - zero_point) / 340.0 # 15
  gyro_x = data[4] / 131.0       # 16
  gyro_y = data[5] / 131.0
  gyro_z = data[6] / 131.0
def main():
  initmpu()
  while True:    # 17
    get_data()   # 18
    print("DATA:")
    print("Acc (%.3f,%.3f,%.3f) g, " % (acc_x, acc_y, acc_z))    # 19
    print("temp %.1f C, " % temp)
    print("gyro (%.3f,%.3f,%.3f) deg/s" % (gyro_x, gyro_y, gyro_z))
    time.sleep(0.5) # s # 20
if __name__ == "__main__":
  main()
```

1 SMBus implémente un sous-ensemble du protocole I2C. La bibliothèque SMBus rend le programme Raspberry Pi plus court que son équivalent Arduino. Le paquet `python-smbus` doit être installé sur le Raspberry Pi (voir plus bas les instructions de la section *SMBus et I2C sans root*).

2 Adresse I2C du capteur MPU 6050, trouvée sur la notice technique. Le nombre est représenté en hexadécimal (voir la section *Hexadécimal, binaire et autres systèmes de numération* dans ce chapitre).

3 Adresse du registre pour les commandes. On peut trouver la liste des registres en cherchant sur le web « MPU 6050 register map ».

4 L'adresse du registre X d'accélération est l'adresse de départ des valeurs qui nous intéressent : accélération, température et vitesse angulaire.

5 Fait de `bus` une variable globale visible pour toutes les fonctions.

6 Pour modifier la valeur d'une variable globale d'une fonction, vous devez indiquer qu'elle est globale au début de la fonction.

7 Initialise le protocole SMBus (I2C). Stocke le nouvel objet de la classe SMBus dans la variable globale `bus`.

8 Le MPU 6050 démarre en mode veille. On le réveille avant de faire quoi que ce soit. Les commandes envoyées au capteur le sont par le protocole I2C (SMBus) avec l'adresse de l'appareil, le registre et la valeur à écrire dans le registre.

9 On demande des données, en partant de l'adresse X d'accélération.

10 On répète 14 fois, avec une valeur de i, allant de 0 à 13.

11 Lit l'octet en cours, le convertit en ASCII, puis l'ajoute à la chaine `rawData`.

12 Convertit la chaine `rawData` en un tuple Python. Les caractères de la chaine de mise en format indiquent le format big endian `>`, un entier court signé sur 2 octets (16 bits).

13 Convertit l'accélération brute en unités g. La pesanteur standard g est égale à 9,81 m/s^2.

14 Calcule le point zéro de la température. Le Pi réalise cela très rapidement et l'écriture de la formule complète à cet endroit facilite la lecture du code et minimise les erreurs de saisie.

15 Convertit la température brute en Celsius. Pour obtenir un résultat en virgule flottante, le diviseur doit aussi être en virgule flottante. Les formules de conversion sont extraites de la notice technique (Google « notice technique MPU 6050 »).
16 Convertit la vitesse angulaire brute en degrés par seconde.
17 Le programme s'exécute tant que l'on n'appuie pas sur Ctrl+C.
18 get_data() met à jour les variables globales, si bien que l'on n'a pas besoin de retourner une valeur à partir de la fonction.
19 Affiche l'accélération en utilisant une chaîne de mise en forme. Le modèle %.3f indique une valeur en virgule flottante avec trois décimales.
20 Pour laisser à l'utilisateur le temps de lire les valeurs affichées et éviter d'accaparer 100% du temps CPU, on ajoute ici un petit délai.

Vous pouvez maintenant marcher, courir ou sauter pour voir l'effet que cela a sur vos mesures !

SMBus et I2C sans root

Le code Raspberry Pi utilise la bibliothèque Python smbus pour I2C. Heureusement, son installation sous Linux est un jeu d'enfant. Vous pouvez installer n'importe quel logiciel sous Raspbian comme vous le feriez avec une distribution Debian, Ubuntu ou Mint. Faites un double-clic sur l'icône du LXTerminal dans le coin gauche du bureau Raspbian, et saisissez dans le terminal :

```
$ sudo apt-get update
$ sudo apt-get install python-smbus
```

Pour activer la prise en charge d'I2C, vous aurez besoin d'activer les modules i2c. Commencez par vous assurer qu'ils ne sont pas désactivés. Modifiez le fichier */etc/modprobe.d/raspi-blacklist.conf* avec la commande sudoedit /etc/modprobe.d/raspi-blacklist.conf puis supprimez cette ligne :

```
blacklist i2c-bcm2708
```

Enregistrez le fichier : appuyez sur Ctrl+X, saisissez y, puis appuyez sur Entrée.

Modifiez ensuite le fichier */etc/modules* avec la commande sudoedit /etc/modules et ajoutez ces deux lignes :

```
i2c-bcm2708
i2c-dev
```

Enregistrez le fichier : appuyez sur Ctrl+X, saisissez y, puis appuyez sur Entrée.

Pour se servir d'I2C sans avoir besoin d'être root, créez le fichier de règles udev *99-i2c.rules* (listé dans l'Exemple 8.5) et copiez-le au bon endroit (pour éviter les erreurs de saisie, vous pouvez télécharger une copie du fichier *99-i2c.rules* à http://botbook.com).

```
$ sudo cp 99-i2c.rules /etc/udev/rules.d/99-i2c.rules
```

Exemple 8.5. 99-i2c.rules

```
# /etc/udev/rules.d/99-i2c.rules - I2C sans root sur Raspberry Pi
# Copyright 2013 http://BotBook.com
SUBSYSTEM=="i2c-dev", MODE="0666"
```

Redémarrez votre Raspberry Pi, ouvrez LXTerminal, et vérifiez que vous pouvez voir les périphériques I2C et que leur appartenance est correcte :

```
$ ls -l /dev/i2c*
```

Le listing doit afficher deux fichiers qui doivent avoir les autorisations `crw-rw-rwT`. Si tel n'est pas le cas, recommencez la procédure.

Hexadécimal, binaire et autres systèmes de numération

Un même nombre peut être représenté de plusieurs façons. Par exemple, le nombre décimal 65 s'écrit 0x41 en hexadécimal et 0b1000001 en binaire. Vous êtes probablement plus familier avec le système décimal où 5+5 est égal à 10.

Les différentes représentations sont indiquées par un préfixe avant le nombre, mais dans le système décimal normal, il n'y a pas de préfixe. Les nombres hexadécimaux commencent par 0x, les nombres binaires par 0b, et les nombres en octal par 0.

Les différents systèmes de numération sont comparés dans le Tableau 8.2.

Préfixe	Système de représentation	Base	Utilisation	Exemple	Calcul
	Décimal	10	Système normal	10	$0*10^0 + 1*10^1$
0x	Hexadécimal	16	Code C et C++, notices techniques	0xA	$10*16^0$
0b	Binaire	2	Protocoles de bas niveau, bit-banging	0b1010	$0*2^0 + 1*2^1 + 0*2^2 + 1*2^3$
0	Octal	8	Permissions Chmod en Linux	012	$2*8^0 + 1*8^1$

Tableau 8.2 Représentations des nombres

Prenons l'exemple du nombre 42 qui peut se représenter sous quatre formes différentes :

```
42 = 0x2a = 0b101010 = 052
```

Seule la base du nombre change ; dans le système décimal normal auquel vous êtes habitué, il s'agit de la base 10. En commençant par la droite, vous commencez à compter les unités, puis les dizaines.

```
2*1 + 4*10 = 42
```

S'il y a un grand nombre, comme 1917, la dizaine est évidente. Après les unités, il y a les dizaines (10), les centaines (10*10) et les milliers (10*10*10). Vous pouvez facilement écrire ces nombres sous la forme de puissances de 10 :

$$10*10 = 10^2$$
$$10*10*10 = 10^3$$

Et pour les unités ? Comme tout nombre élevé à la puissance zéro est égal à 1, on a donc :

$10^0 = 1$

Ainsi, le nombre 42 devient :

$42 = 4*10^1 + 2*10^0$

La représentation hexadécimale de 42 s'écrit 0x2A. En hexa, les nombres supérieurs à 9 s'écrivent avec des lettres : A=10, B=11... F=15. En commençant par la droite, notez que A est égal à 10 et comptez :

$0x2A = 10*1 + 2*16 = 10 + 32 = 42$

Si l'on emploie les puissances, on peut alors écrire :

$10*16^0 + 2*16^1 = 0x2A$

Essayez sur d'autres nombres. Pour vérifier vos calculs, utilisez la console Python (voir plus bas l'encadré *Console Python* ou le Tableau 8.3). Pouvez-vous également utiliser vos compétences pour faire des conversions de nombres en binaire ?

Vous pouvez vous entraîner à travailler avec des nombres dans la console Python. La représentation des nombres (1 == 0x1 == 0b1) est identique en Python, C et C++. Vous pouvez exécuter n'importe quelle commande Python dans la console :

```
>>> print("Botbook.com")
Botbook.com
>>> 2+2
4
```

Tous les nombres saisis sont convertis en décimal lors de l'affichage :

```
>>> 0x42
66
>>> 66
66
>>> 0b1000010
66
```

Il y a des fonctions pour convertir les nombres en binaire, hexadécimal, octal et caractères ASCII :

```
>>> bin(3)
'0b11'
>>> hex(10)
'0xa'
>>> oct(8)
'010'
>>> chr(0x42)
'B'
```

Décimal	Hexadécimal	Binaire	Octal	ASCII
0	0x0	0b0	0	*\0* NUL, caractère de fin de chaîne
1	0x1	0b1	01	
2	0x2	0b10	02	
3	0x3	0b11	03	
4	0x4	0b100	04	EOT (*end of text*, fin de texte), CTRL+D en Linux
5	0x5	0b101	05	
6	0x6	0b110	06	
7	0x7	0b111	07	
8	0x8	0b1000	010	
9	0x9	0b1001	011	
10	0xA	0b1010	012	*\n* nouvelle ligne
11	0x8	0b1011	013	
16	0x10	0b10000	020	
17	0x11	0b10001	021	
32	0x20	0b100000	040	' ' espace, premier caractère imprimable
48	0x30	0b110000	060	*0* caractère ASCII zéro qui n'est pas le chiffre zéro
65	0x41	0b1000001	0101	*A*
97	0x61	0b1100001	0141	*a*
126	0x7e	0b1111110	0176	~ le tilde est le dernier caractère ASCII imprimable

Tableau 8.3 Exemples de nombres dans différentes représentations

Saisissez la commande `man ascii` dans le terminal pour afficher un tableau des caractères ASCII en Linux ou OS X. Par exemple, vous pouvez voir que le caractère ASCII *A* est le nombre décimal 65, hexadécimal 0x41, et octal 0101. Les pages du manuel sont affichées dans l'utilitaire less, si bien que l'appui sur la barre d'espacement fait avancer d'une page, l'appui sur la touche b recule d'une page et l'appui sur q quitte le programme.

Console Python

La console Python est démarrée en saisissant la commande `python` dans le terminal. En Linux, le terminal (bash, shell) se trouve en bas à gauche du menu principal : **Accessoires→Terminal**, ou en recherchant Dash pour Terminal. Avec un Macintosh, le terminal se trouve dans */Applications/Utilitaires/Terminal*. Sous Windows, vous pouvez démarrer le terminal (**Applications→Accessoires→Invite de commande**) ou bien ouvrir IDLE à partir du menu Démarrer (si vous avez installé Python).

L'invite Python `>>>` permet de savoir que vos commandes seront interprétées comme du code Python. Pour terminer votre session Python, saisissez la commande Python `exit()`.

Si vous désirez une console Python vraiment efficace avec une aide interactive et l'auto-complétion, essayez `ipython`.

Opérations au niveau des bits

Certains capteurs envoient des données sous la forme d'une série de bits qui composent un octet, par exemple 01010000. Pour travailler avec des données binaires, vous devez effectuer des opérations au niveau des bits, que vous manipulerez en traitant plusieurs bits en même temps.

> Un bit est un 1 ou un 0. Un bit peut représenter une valeur de vérité, 0 pour false ou 1 pour true.
>
> Un octet se compose de huit bits, par exemple, 0b1010100. Un octet peut représenter un caractère ASCII, par exemple *T*, *3*, ou *x*. Il peut aussi représenter un nombre, comme 256 ou –127.

Les opérations au niveau des bits sont des fonctions de très bas niveau (on travaille presque avec la mémoire de l'ordinateur dans son format natif), et il peut être difficile de lire le code qui les utilise, si bien que vous ne devez y avoir recours que lorsque cela est nécessaire. D'un autre côté, on rencontre parfois des capteurs qui nécessitent une arithmétique binaire pour fonctionner. L'unité de déplacement intégrée du MPU 6050 fait partie de ces capteurs.

Les opérations d'arithmétique binaire les plus courantes sont le décalage de bits et les opérations booléennes.

Pour vous habituer aux opérations sur les bits, entraînez-vous dans la console Python. Les opérations sur les bits fonctionnent de la même manière en C et C++ (Arduino), mais ces langages n'ont pas de consoles interactives, si bien qu'il est plus facile de s'entraîner en Python.

Vous connaissez sans doute déjà l'algèbre booléenne classique. « AND » signifie que les deux conditions doivent être vraies pour que l'expression complète soit considérée comme vraie.

« OR » signifie que l'une des deux conditions doit être vraie pour que l'expression complète soit vraie.

```
>>> if (True): print "Oui"
...
Oui
>>> if (False): print "Oui"
...     # rien ne s'affiche
>>> if (True and True): print "Oui"
...
Oui
```

> En Python, True et False doivent avoir une majuscule.

Les valeurs de vérité peuvent être employées même sans construction « if » :

```
>>> True
True
>>> False
False
>>> True and True
True
```

Dans la plupart des langages, 1 est égal à True et 0 à False :

```
>>> 1 and 1
1
>>> 1 and 0
0
```

Avec les opérations au niveau des bits, vous pouvez faire la même chose à chacun des zéros et des uns qui composent un octet.

> Vous ne pouvez pas utiliser les mots anglais « and » ou « or » pour les opérations au niveau des bits en Python. Vous devez à la place employer les caractères & (AND au niveau du bit) et | (OR au niveau du bit).

```
>>> 0b0101 & 0b0110
4
```

La réponse est peut-être plus facile à comprendre en binaire :

```
>>> bin(0b101 & 0b110)
'0b100'
```

Le AND au niveau du bit (&) prend simplement chaque bit et applique l'opération booléenne AND à chaque paire :

```
    0b0101
    0b0110
==========
&   0b0100
```

En partant de la gauche, 0-false et 0-false donne 0-false. 1-true et 1-true donne 1-true. 0-false et 1-true donne 0-false. 1-true et 0-false donne 0-false.

Masquage de bits avec AND & au niveau des bits

Le masquage de bits permet de n'obtenir que les bits souhaités. Par exemple, on peut récupérer les quatre chiffres de gauche de 0b 1010 1010 grâce à l'opération suivante :

```
>>> bin(0b10101010 & 0b11110000)
'0b10100000'
```

Voilà comment cela fonctionne :

```
0b 1010 1010     # nombre
0b 1111 0000     # masque
==========================
0b 1010 0000     # & (AND au niveau des bits)
```

AND n'est vrai que si les deux entrées sont vraies (Tableau 8.4).

a	b	a AND b
0	0	0
1	0	0
0	1	0
1	1	1

Tableau 8.4 Table de vérité AND

OR | au niveau des bits

Si vous construisez la partie gauche et la partie droite d'un octet séparément, comment allez-vous reconstituer l'ensemble ? En Python, vous pouvez utiliser une opération OR (|) :

```
>>> gauche=0b10100000
>>> droite=0b1111
>>> bin(gauche|droite)
'0b10101111'
```

Examinons les bits en détail :

```
0b 1010 0000    # gauche, doit avoir les zéros sur la droite
0b      1111    # droite, les zéros sur la gauche n'ont pas d'importance
==========================
0b 1010 1111    # OR | au niveau des bits
```

Le OR « | » au niveau des bits n'est pas identique au plus « + ». Par exemple, étudiez l'expression 0b1 + 0b1.

Décalage de bits < <

Le décalage de bits déplace des bits vers la droite ou vers la gauche. Vous pouvez tester cela en Python :

```
>>> bin(0b0011<<1)
'0b110'
>>> bin(0b0011<<2)
'0b1100'
```

Le déplacement des bits au-delà du côté droit abandonne les bits supplémentaires :

```
>>> bin(0b0011>>2)
'0b0'
```

Endianness

La plupart des matériels fonctionnent en little endian, comme les nombres avec lesquels vous travaillez quotidiennement. Les nombres normaux sont en little endian : dans le nombre 1991, les milliers viennent en premier, puis les centaines, les dizaines et pour finir les unités. Le chiffre le plus significatif vient en premier.

La plupart des ordinateurs sont également en little endian, si bien que c'est l'octet le plus significatif qui vient en premier. Votre station de travail et votre ordinateur portable fonctionnent probablement avec une architecture x86, amd64, ou x86_64, si bien qu'ils sont en little endian. Arduino est basé sur le processeur Atmel AVR qui est en little endian. Raspberry Pi et Android Linux (la plateforme leader du marché des smartphones) s'exécutent sous un processeur ARM qui est en little endian.

Le nombre 135 peut être stocké en little endian (cas le plus typique) ou en big endian :

```
0b 1000 0111    # little endian, cas général, Arduino, Pi, ordinateur
0b 0111 1000    # big endian, MPU 6050
```

Certains appareils, qui ne sont pas majoritaires, ont une endianness atypique. C'est le cas du capteur MPU 6050 qui est en big endian, ce qui veut dire que les octets les plus significatifs viennent en dernier. Lors d'une communication entre un Arduino et un MPU 6050, l'endianness doit être permutée entre little endian et big endian.

L'endianness n'a d'importance que lorsque l'on effectue des opérations de bas niveau, comme quand on travaille au niveau des bits et des octets. En programmation de haut niveau, on se contente d'assigner une valeur à une variable en virgule flottante et on obtient en retour un nombre en virgule flottante. C'est donc l'environnement de programmation qui gère l'endianness et les autres détails à votre place.

EXPÉRIENCE : DÉTOURNEMENT D'UN NUNCHUK (AVEC I2C)

Aimeriez-vous avoir un accéléromètre, un joystick et un bouton, le tout dans un package économique ? Ne cherchez plus, car le contrôleur Nunchuk de la console de jeux Wii est fait pour vous ! Si vous voulez baisser les coûts, il existe aussi des copies compatibles encore meilleur marché.

Le Nunchuk Wii nous donne également une importante leçon de hacking : les ingénieurs sont des êtres humains. Et comme le reste de l'humanité, les ingénieurs veulent travailler avec de véritables protocoles éprouvés qui possèdent des bibliothèques et des outils. La Wii a son propre connecteur propriétaire qui, à l'origine, était très peu documenté. Mais sous le capot, on trouve en fait le protocole standard I2C. Comme nous l'avons déjà vu (voir plus haut l'encadré *Protocoles standard industriels*), I2C est notre norme industrielle favorite en matière de protocole de communication à courte portée.

Nintendo produit le Nunchuk Wii en grandes séries, ce qui en fait un produit de qualité et bon marché. La grande disponibilité du Nunchuk signifie aussi que de nombreux exemples de code sont disponibles. Vous n'avez même pas besoin de couper le cordon car il existe un adaptateur WiiChuck que vous pouvez placer sur le connecteur du Nunchuk pour le connecter à un Arduino ou à une platine de test (Figure 8.7).

Le Nunchuk Wii peut utiliser le protocole SMBus pour communiquer. Il s'agit d'un protocole simplifié car c'est un sous-ensemble du standard I2C, ce qui rend le processus de communication encore plus intelligible.

Nunchuk pour Arduino

La Figure 8.8 illustre le diagramme de connexion pour Arduino. Assemblez selon le schéma et exécutez le sketch de l'Exemple 8.6.

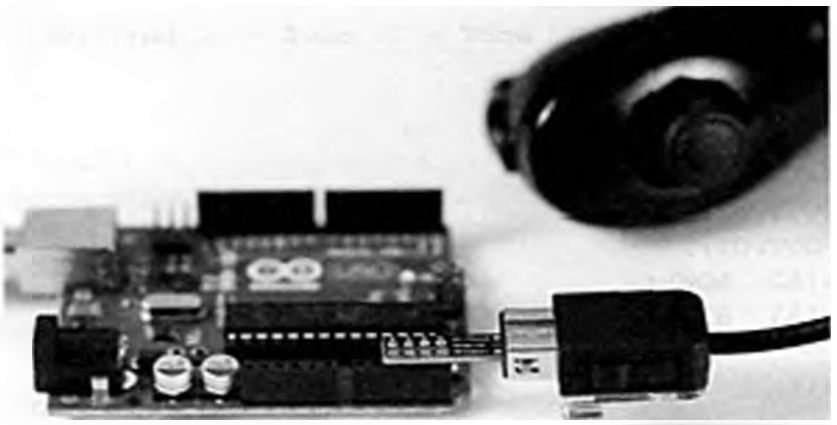


Figure 8.7 Nunchuk connecté à un Arduino avec un connecteur WiiChuck

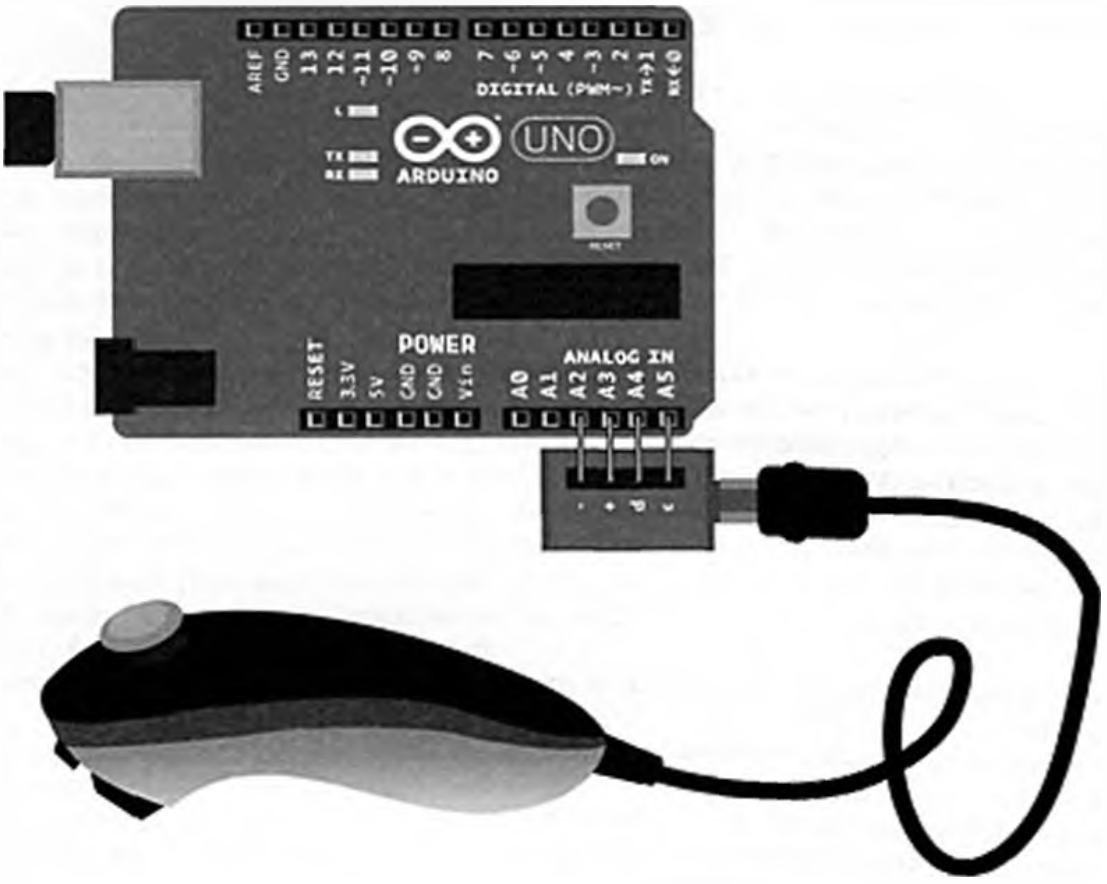


Figure 8.8 Circuit WiiChuck pour Arduino

Exemple 8.6. wiichuck_adapter.ino

```
// wiichuck_adapter.ino - affiche les données du joystick, de
l'accéléromètre et du bouton sur le port série
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
#include <Wire.h>
const char i2c_address = 0x52;
unsigned long lastGet=0; // ms
```

```
int jx = 0, jy = 0, accX = 0, accY = 0, accZ = 0, buttonZ = 0, buttonC = 0;
// 1
void setup() {
  Serial.begin(115200);
  Wire.begin();
  pinMode(A2, OUTPUT);
  pinMode(A3, OUTPUT);
  digitalWrite(A2, LOW);          // 2
  digitalWrite(A3, HIGH);         // 3
  delay(100);
  initNunchuck();         // 4
}
void loop() {
  if(millis() - lastGet > 100) {          // 5
    get_data(); // 6
    lastGet = millis(); // 7
  }
  Serial.print("Bouton Z : ");
  Serial.print(buttonZ);          // 8
  Serial.print(" Bouton C : ");
  Serial.print(buttonC);
  Serial.print(" Joystick : (x,y) (");
  Serial.print(jx);       // 9
  Serial.print(",");
  Serial.print(jy);
  Serial.print(") Accélération (x,y,z) (");
  Serial.print(accX);     // 10
  Serial.print(",");
  Serial.print(accY);
  Serial.print(",");
  Serial.print(accZ);
  Serial.println(")");
  delay(10); // ms
}
void get_data() {
  int buffer[6]; // 11
  Wire.requestFrom(i2c_address, 6);       // 12
  int i = 0;      // 13
  while(Wire.available()) {       // 14
    buffer[i] = Wire.read();      // 15
    buffer[i] ^= 0x17;  // 16
    buffer[i] += 0x17;  // 17
    i++;
  }
  if(i != 6) {  // 18
    Serial.println("Erreur de lecture i2c");
  }
  write_i2c_zero();       // 19
  buttonZ = buffer[5] & 0x01;     // 20
  buttonC = (buffer[5] >> 1) & 0x01;      // 21
  jx = buffer[0];         // 22
  jy = buffer[1];
```

```
  accX = buffer[2];
  accY = buffer[3];
  accZ = buffer[4];
}
void write_i2c_zero() {
  Wire.beginTransmission(i2c_address);
  Wire.write(0x00);
  Wire.endTransmission();
}
void initNunchuck()
{
  Wire.beginTransmission(i2c_address);
  Wire.write(0x40);
  Wire.write(0x00);
  Wire.endTransmission();
}
```

1 Déclare des variables globales. Comme les fonctions Arduino ne peuvent pas facilement retourner plusieurs valeurs, les données sont passées de fonction en fonction grâce à des variables globales.
2 Utilise la broche analogique A2 comme masse (GND, 0 V, LOW). De cette manière, le WiiChuck peut être branché sur le connecteur Arduino sans nécessiter de platine de test ou de câbles de liaison.
3 Utilise A3 comme alimentation +5 V (HIGH, positif, VCC).
4 Initialise le Nunchuk Wii en appelant initNunchuck(), qui envoie 0x40 et 0x00 sur le bus I2C.
5 Récupère les données toutes les 100 ms. Il s'agit d'un modèle courant de programme pour accomplir quelque chose toutes les *x* millisecondes. La fonction millis() retourne le temps de service de l'Arduino (depuis combien de temps est-il allumé ?) en millisecondes. La variable lastGet contient l'heure depuis la dernière fois où get_data() a été appelé, en millisecondes de temps de service (lors de la première itération, lastGet est à 0). La différence entre millis() et lastGet est le temps écoulé depuis le dernier appel à get_data(). Si plus de 100 ms se sont écoulées, Arduino exécute le bloc suivant.
6 Comme get_data() met à jour les variables globales, elle n'a pas besoin de retourner de valeur.
7 Met à jour l'heure du dernier appel à get_data().
8 Les états du bouton sont 0 quand il est pressé, et 1 quand il est relâché.
9 Les deux axes du joystick (`jx`, `jy`) sont des valeurs brutes allant de 30 à 220.
10 Tous les axes de l'accéléromètre (`accX`, `accY`, `accZ`) sont des valeurs brutes allant de 80 à 190.
11 Déclare un nouveau tableau de six entiers.
12 Demande six octets de données au Nunchuk.
13 La variable i de la boucle contient le nombre d'octets traités.
14 N'entre dans la boucle que s'il y a des octets à lire.
15 Lit un octet et le stocke dans la cellule en cours de `buffer[]`. Lors de la première itération, il s'agit de `buffer[0]`. Lors de la dernière itération, il s'agit de `buffer[5]`.
16 Accomplit un XOR sur la valeur en cours avec 0x17, et remplace la valeur en cours par le résultat (on appelle cela une opération en place). Le ou exclusif (XOR) est une opération booléenne similaire à un OR, mais elle n'est vraie que si a ou b est vrai, et non pas quand les deux sont vrais.
17 Ajoute 0x17 à la valeur en cours.

18 Si le nombre d'octets lus n'est pas égal à 6, on affiche un avertissement.
19 Demande une autre lecture en envoyant 0x00 par le protocole I2C.
20 Récupère le dernier bit du dernier octet. Le dernier octet est `buffer[5]`. Le dernier bit est extrait par masquage de bits ; le code accomplit sur l'octet un AND au niveau des bits avec la valeur 0b 0000 0000 0000 0001. Voir la section *Masquage de bits avec AND & au niveau des bits* et le Tableau 8.5. Les états du bouton sont 0 quand il est pressé, et 1 quand il est relâché.

Octet	Usage	Octet	Usage
0	Joystick X	3	Accéléromètre Y
1	Joystick Y	4	Accéléromètre Z
2	Accéléromètre X	5	ZCxxyyzz : boutons « Z » et « C », précision de l'accéléromètre

Tableau 8.5 Bloc de données des 6 octets du Nunchuk

21 Récupère l'avant-dernier bit du dernier octet. L'avant-dernier bit est déplacé en dernière place avec le décalage de bits `>>1`. Ensuite, le dernier bit est extrait comme dans la ligne précédente. Voir la section *Décalage de bits* `<<`.
22 L'axe x du joystick `jx` est constitué d'un octet complet. Le reste des axes du joystick et de l'accéléromètre est lu de façon similaire.

Nunchuk pour Raspberry Pi

La Figure 8.9 illustre le diagramme de connexion pour Raspberry Pi. Branchez suivant le schéma et exécutez le programme de l'Exemple 8.7.

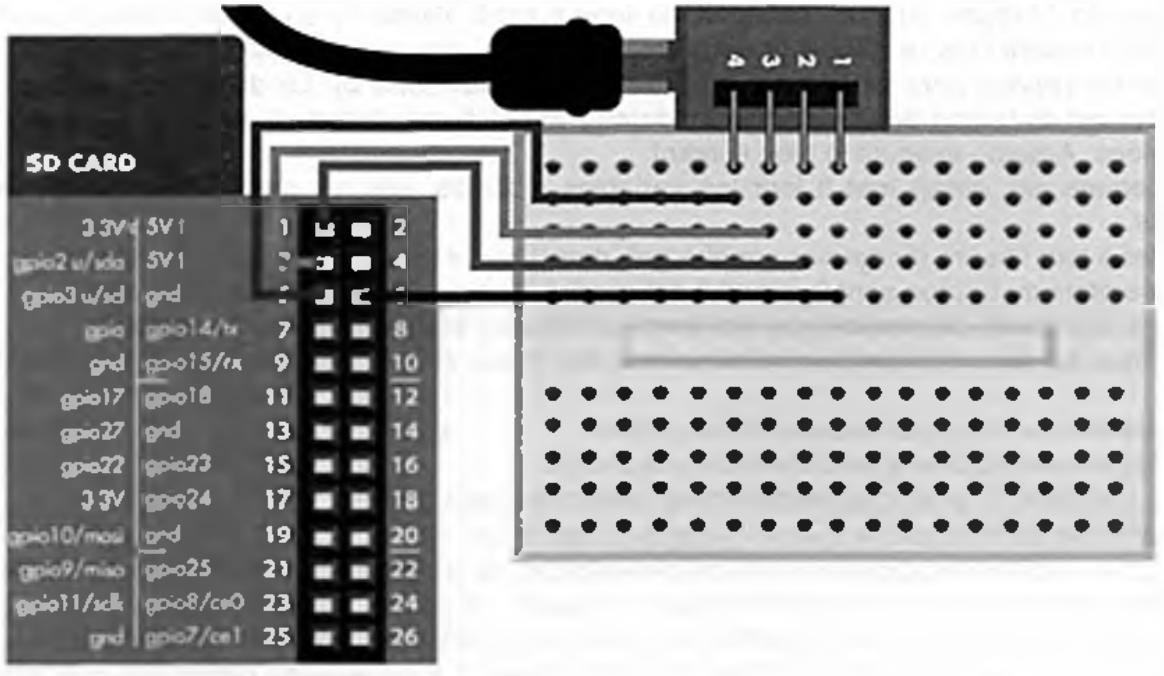


Figure 8.9 Circuit WiiChuck pour Raspberry Pi

Exemple 8.7. wiichuck_adapter.py

```
# wiichuck_adapter.py - affiche les données d'accélération et du joystick du
Nunchuck
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import smbus # sudo apt-get -y install python-smbus    # 1
bus = None
address = 0x52    # 2
z = 0    # 3
c = 0
joystick_x = 0
joystick_y = 0
ax_x = 0
ax_y = 0
ax_z = 0
def initNunchuck():
    global bus
    bus = smbus.SMBus(1) # 4
    bus.write_byte_data(address, 0x40, 0x00)    # 5
def send_request():
    bus.write_byte(address, 0x00)    # 6
def get_data():
    global bus, z, c, joystick_x, joystick_y, ax_x, ax_y, ax_z
    data = [0]*6
    for i in range(len(data)): # 7
        data[i] = bus.read_byte(address)
        data[i] ^= 0x17
        data[i] += 0x17
    z = data[5] & 0x01    # 8
    c = (data[5] >> 1) & 0x01    # 9
    joystick_x = data[0]
    joystick_y = data[1]
    ax_x = data[2]
    ax_y = data[3]
    ax_z = data[4]
    send_request()
def main():
    initNunchuck()
    while True:
        get_data()
        print("Bouton Z : %d Bouton C : %d joystick (x,y) (%d,%d) \
              accélération (x,y,z) (%d,%d,%d)" \
              % (z,c,joystick_x,joystick_y,ax_x, ax_y, ax_z))
        time.sleep(0.1)
if __name__ == "__main__":
    main()
```

1 La bibliothèque python-smbus doit être installée sur le Raspberry Pi (voir plus haut *SMBus et I2C sans root*).

2 Adresse du Nunchuk Wii. La valeur est en hexa. Voir plus haut la section *Hexadécimal, binaire et autres systèmes de numération* pour des informations détaillées sur les valeurs hexadécimales.
3 Variable globales pour l'un des boutons.
4 Crée un nouvel objet de classe SMBus, et le stocke dans la nouvelle variable bus. Le constructeur SMBus() a un paramètre, le numéro du périphérique. Le numéro 1 correspond au fichier */dev/i2c-1*. Ce numéro de périphérique est courant avec les cartes modernes Raspberry Pi. Si vous avez une ancienne carte Raspberry Pi (révision 1), vous serez peut-être obligé d'utiliser à la place le numéro 0 qui correspond à */dev/i2c-0*.
5 Le Nunchuk doit être initialisé avec des commandes I2C. `bus.write_byte_data(addr=0x52, cmd=0x40, val=0x00)` envoie au Nunchuk la commande `0x40` avec la valeur `0x00`.
6 Vous pouvez demander au Nunchuk la prochaine série de valeurs avec le caractère null `0x00`, qui est équivalent à `\0` ou simplement 0.
7 Lit six octets de données. Cela contient le bloc de données décrit dans le Tableau 8.5. Le reste de la fonction décode les données.
8 Récupère l'état du bouton « Z » : 1 pour un bouton enfoncé, 0 quand il est relâché. Le masque de bits ne concerne qu'un bit, `0b1`. Quand il est utilisé avec un AND (`&`) au niveau des bits, il s'agit du bit d'extrême droite, et tout ce qui se trouve à gauche est considéré comme étant à zéro. Un AND au niveau des bits avec `0b1` retourne donc simplement le bit d'extrême droite. Voir aussi la section *Masquage de bits avec AND & au niveau des bits*.
9 Récupère l'état du bouton « C » : 1 pour un bouton enfoncé, 0 quand il est relâché. Pour obtenir l'avant-dernier bit, on déplace celui-ci à la dernière place (`x > > 1`) et on utilise un masque de bits pour ne récupérer que le dernier bit.

PROJET : MAIN ROBOTISÉE CONTRÔLÉE PAR LE NUNCHUK WII

Comme vous savez déjà lire l'accélération et la position d'un joystick, vous pouvez très simplement faire tourner des servos en fonction de ces valeurs.

Si vous ajoutez un bras mécanique, vous obtiendrez une main robotisée contrôlée par un Nunchuk.

Figure 8.10 Robot contrôlé par un Nunchuk

Ce que vous allez apprendre

Dans le projet de Main robotisée, vous allez apprendre à :

- Utiliser un accéléromètre et un joystick mécanique avec des sorties.
- Combiner des servos pour des mouvements complexes.

Vous allez aussi rafraîchir vos compétences sur le contrôle des servos et filtrer le bruit avec des moyennes glissantes (voir la section *Servos* du chapitre 5 et la section *Moyenne mobile* du chapitre 7).

Commencez simplement avec les servos (Figure 8.11). Une fois que vous arrivez à gérer le mouvement, vous pouvez continuer avec le mécanisme de la main.

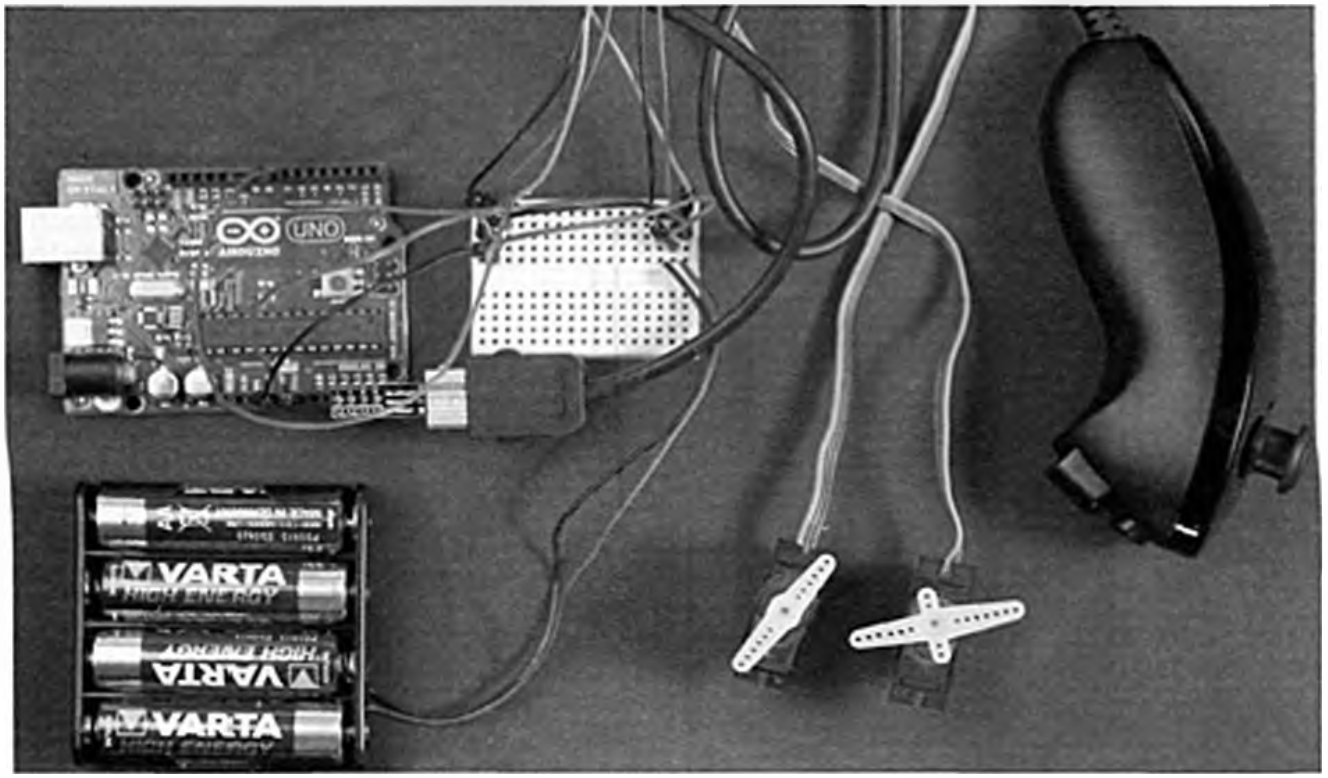


Figure 8.11 Un Nunchuk contrôle deux servos avec un Arduino

La Figure 8.12 illustre le schéma de câblage. Branchez en suivant le schéma puis exécutez le code de l'Exemple 8.8.

> Pour en savoir plus sur le Nunchuk Wii avec Arduino, reportez-vous plus haut à la section *Nunchuk pour Arduino*.

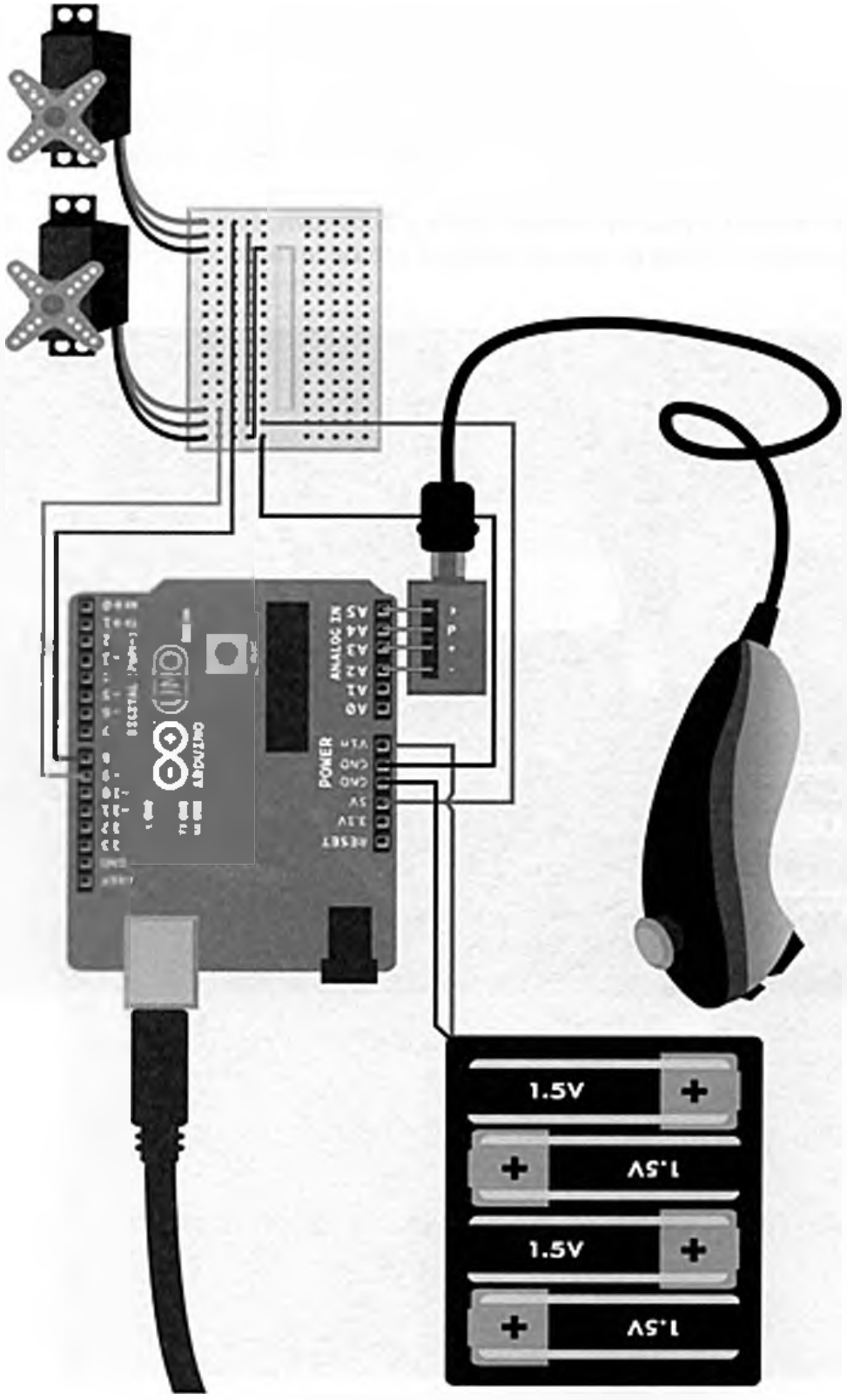


Figure 8.12 Circuit pour Arduino

Exemple 8.8. wiichuck_adapter_claw.ino

```
// wiichuck_adapter_claw.ino - contrôle une main robotisée avec un Nunchuck
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
#include <Wire.h>
const int clawPin = 8;
const int armPin = 9;
int armPos=0, clawPos=0;
float wiiP = 0.0;        // 1
float wiiPAvg = 0.0;     // 2
int lastarmPos = 350;
const char i2c_address = 0x52;
int jx = 0, jy = 0, accX = 0, accY = 0, accZ = 0, button2 = 0, buttonC = 0;
void setup() {
  Serial.begin(115200);
  // Nunchuck
  Wire.begin();
  pinMode(A2, OUTPUT);
  pinMode(A3, OUTPUT);
  digitalWrite(A2, LOW);
  digitalWrite(A3, HIGH);
  delay(100);
  initNunchuck();
  // Servos
  pinMode(clawPin, OUTPUT);
  pinMode(armPin, OUTPUT);
}
void loop() {
  get_data();    // 3
  wiiP = (accZ-70.0)/(178.0-70.0);       // 4
  if (accY>120 && accZ>100) wiiP=1;
  if (accY>120 && accZ<100) wiiP=0;
  if (wiiP>1) wiiP=1;
  if (wiiP<0) wiiP=0;
  wiiPAvg = runningAvg(wiiP, wiiPAvg);  // 5
  armPos = map(wiiPAvg*10*1000, 0, 10*1000, 2200, 350);
  clawPos = map(jy, 30, 220, 1600, 2250);        // 6
  pulseServo(armPin, armPos);    // 7
  pulseServo(clawPin, clawPos);
  printDebug();
}
float runningAvg(float current, float old) {
  float newWeight=0.3;
  return newWeight*current + (1-newWeight)*old; // 8
}
// servo
void pulseServo(int servoPin, int pulseLenUs)    // 9
{
  digitalWrite(servoPin, HIGH);
  delayMicroseconds(pulseLenUs);
  digitalWrite(servoPin, LOW);
  delay(15);
```

```
}
// i2c
void get_data() {
  int buffer[6];
  Wire.requestFrom(i2c_address,6);
  int i = 0;
  while(Wire.available()) {
    buffer[i] = Wire.read();
    buffer[i] ^= 0x17;
    buffer[i] += 0x17;
    i++;
  }
  if(i != 6) {
    Serial.println("Erreur de lecture i2c");
  }
  write_i2c_zero();
  buttonZ = buffer[5] & 0x01;
  buttonC = (buffer[5] >> 1) & 0x01;
  jx = buffer[0];
  jy = buffer[1];
  accX = buffer[2];
  accY = buffer[3];
  accZ = buffer[4];
}
void write_i2c_zero() {
  Wire.beginTransmission(i2c_address);
  Wire.write((byte)0x00);
  Wire.endTransmission();
}
void initNunchuck()
{
  Wire.beginTransmission(i2c_address);
  Wire.write((byte)0x40);
  Wire.write((byte)0x00);
  Wire.endTransmission();
}
// débogage
void printDebug()
{
  Serial.print("accZ:");
  Serial.print(accZ);
  Serial.print("        wiiP:");
  Serial.print(wiiP);
  Serial.print("        wiiPAvg:");
  Serial.print(wiiPAvg);
  Serial.print("        jy:");
  Serial.print(jy);
  Serial.print("        clawPos:");
  Serial.println(clawPos);
}
```

1 `WiiP` contient l'inclinaison de la Wii sous la forme d'un pourcentage. En arrière est égal à 0,0 et en avant à 1,0.
2 Moyenne glissante de `WiiP`.
3 La lecture du Nunchuk Wii via i2c requiert un délai de 20 ms entre les lectures. Les deux appels à pulseServo() plus tard dans loop() fournissent ce délai.
4 Calcule un pourcentage à partir de valeurs brutes lues sur la Wii. Cette formule est similaire à la fonction intégrée map().
5 Pour filtrer les pics aléatoires, on utilise la moyenne des deux derniers échantillons de l'accélération de l'axe z.
6 Prend la valeur brute du joystick (de 30 à 220) et la mappe vers l'impulsion du servo (de 1500 à 2400). Par exemple, la valeur du joystick 30 mappe vers la longueur d'impulsion du servo de 1500 µs.
7 Envoie une impulsion au servo contrôlant le bras. Pour que le servo continue à tourner, vous devez envoyer un flux continu d'impulsions au servo.
8 Pour obtenir une moyenne glissante sur plusieurs valeurs, on utilise une moyenne pondérée. De cette manière, seul un point de données précédent a besoin d'être stocké, mais les anciennes valeurs peuvent toujours affecter la moyenne.
9 Envoie une impulsion au servo. Pour que le contrôle du servo soit fiable, cette fonction doit être appelée à peu près 50 fois par seconde. Voir la section *Servos* du chapitre 5.

Ajout du mécanisme de la main

Vous savez déjà comment contrôler les moteurs du servo avec le Nunchuk Wii, et il est facile d'adapter cela aux bras et aux mains des robots que l'on trouve dans le commerce. Le choix est très varié et nous avons trouvé le nôtre sur le site http://dx.com (cherchez dans le catalogue à « DAGU VA2 »).

Quel que soit le modèle que vous choisissiez, c'est une bonne idée de le monter solidement sur une base afin de l'empêcher de basculer. Nous avons fixé notre bras à une épaisse planche de bois avec des vis (Figure 8.13).

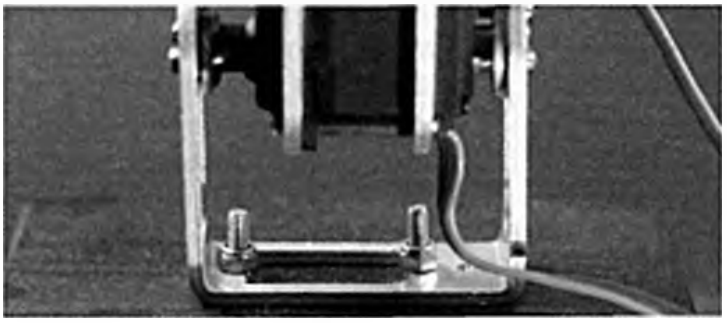


Figure 8.13 Bras du robot fixé à une base

Tout le code est déjà finalisé (voir l'Exemple 8.8). Connectez les servos de manière à pouvoir contrôler la pince avec le bouton du Nunchuk, et le mouvement du bras avec l'accéléromètre. La Figure 8.14 illustre le résultat final.

Vous savez à présent mesurer l'accélération et la vitesse angulaire. Vous connaissez l'orientation de votre appareil et dans quel sens il tourne. Vous pouvez mesurer l'un de ces paramètres ou bien employer une unité mobile intégrée pour mesurer tous les paramètres si cela s'avère nécessaire pour votre projet.

Pour utiliser ces capteurs, vous avez probablement remarqué qu'il est possible de se servir des exemples de code qui sont prêts à l'emploi, mais si vous ne renoncez pas aux difficultés et apprenez les opérations au niveau des bits pour comprendre comment le code fonctionne, vous pourrez être fier de vous. Ces compétences sont utiles si vous travaillez avec un nouveau capteur complexe livré sans exemples de code.

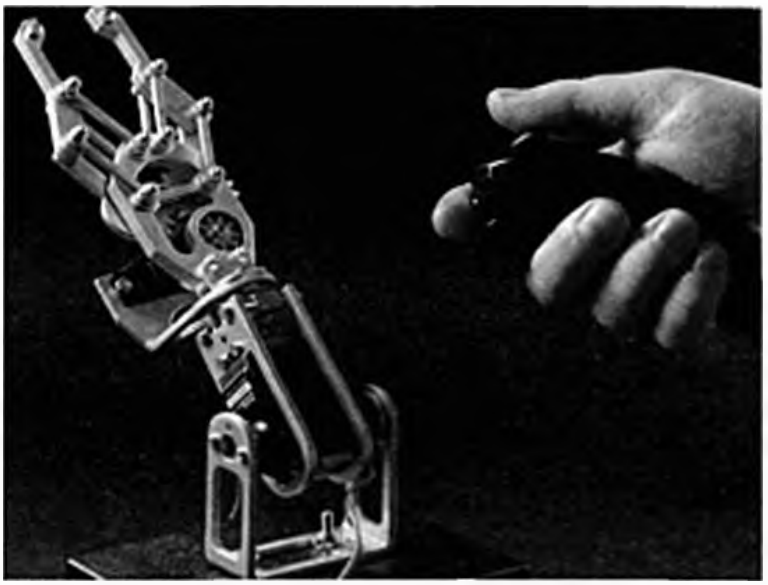


Figure 8.14 Bras du robot contrôlé par la Wii

9 Électricité et magnétisme

Les capteurs à effet Hall détectent les champs magnétiques. Il en existe différentes sortes : certains ne détectent que la présence d'un aimant, alors que d'autres indiquent la force du champ magnétique en teslas.

Les capteurs de courant et de tension fonctionnent comme un multimètre en mesurant l'électricité qui les traverse. Ils peuvent facilement mesurer le courant qui pourrait endommager la broche d'entrée analogique d'un microcontrôleur.

Une boussole électronique indique où se trouve le nord magnétique. Les plus évoluées sont couplées à un accéléromètre, de telle sorte qu'elles peuvent pointer le nord, même quand elles sont inclinées.

> Les pôles Nord et Sud du Soleil s'inversent à peu près tous les onze ans. Au moment où nous écrivons ce livre, nous attendons d'une semaine à l'autre cette inversion. Ne vous faites pas de souci cependant à propos de l'inversion des pôles de la Terre, la dernière fois qu'elle s'est produite, c'était il y a 780 000 années...

EXPÉRIENCE : TENSION ET COURANT

Dans cette expérience, vous allez utiliser l'AttoPilot pour mesurer la tension d'un bloc de piles.

Est-ce que vos piles sont à plat et combien de courant le moteur de votre robot consomme-t-il ? L'AttoPilot peut répondre à ces questions car il mesure la tension et le courant et transmet ces informations à sa sortie analogique.

L'AttoPilot sait mesurer une grande quantité de courant. Le modèle le plus puissant est certifié pour 50 V et 180 A. Rappelons que la puissance (*P* mesurée en watt) est une fonction de la tension (*U* mesurée en volt) et de l'intensité (*I* mesurée en ampère) :

```
P = UI = 50 V • 180 A = 9000 VA = 9000 W = 9 kW
```

On peut aussi exprimer cette formule en disant que les watts (W) sont une fonction du voltage (V) et de l'intensité du courant (A) :

```
W = VA = 50 V • 180 A = 9000 W = 9 kW
```

> Le modèle 50 V/ 180 A du capteur AttoPilot peut supporter 9 kilowatts, ce qui n'est pas votre cas. Restez prudent et n'utilisez que des tensions raisonnables, comme des piles ou une alimentation USB.

Modèle	Tension U, sortie/mesure	Intensité I, sortie/mesure	Commentaire
13,6 V / 45 A	242,3 mV / V	73,20 mV / A	Utilisé dans cette expérience
50 V / 90 A	63,69 mV / V	36,60 mV / A	
50 V / 180 A	63,69 mV / V	18,30 mV / A	9 kW

Tableau 9.1 Tensions et courants mesurés par l'AttoPilot et facteurs de conversion

Comme de telles puissances ne peuvent pas être gérées en toute sécurité dans la plupart des projets, nous avons utilisé le modèle 13 V / 45 A pour ce test.

```
P = UI = 13 V * 45 A = 585 W     # modèle utilisé ici
```

L'AttoPilot dispose de deux sorties analogiques. L'une indique le courant et l'autre la tension. La sortie maximale est de 3,3 V, ce qui est considérablement moins que le maximum de 50 V mesuré. Les facteurs de conversion du modèle 13,6 V / 45 A utilisé ici sont calculés à partir des spécifications maximales du composant. Notre modèle d'exemple utilise la formule suivante pour le courant :

```
3,3 V sortie / 45 A mesurés = 73,3 mV sortie / A mesurés
```

Mais dans votre code, vous allez réorganiser les choses différemment :

```
45 A mesurés / 3,3 V sortie = 13,6363 A mesurés / V sortie
```

Ainsi, quand vous voulez calculer le courant mesuré, vous pouvez multiplier la tension lue sur la sortie du capteur par 13,6363 :

```
0,05 V sortie * 13,6363 ≈ 0,6818 A = 681,8 mA
```

Et voici la formule pour la tension :

```
3,3 V sortie / 13,6 V mesurés = 242,6 mV sortie / V mesurés
```

On exprime cela différemment dans le code :

```
13,6 V mesurés / 3,3 V sortie = 4,1212 V mesurés / V sortie
```

Quand on veut calculer la tension mesurée, on peut multiplier la tension lue sur la sortie du capteur par 4,1212 :

```
1,213 V sortie * 4,1212 ≈ 5 V
```

AttoPilot pour Arduino

La Figure 9.1 illustre les connexions pour Arduino. Câblez selon le schéma puis exécutez le sketch de l'Exemple 9.1.

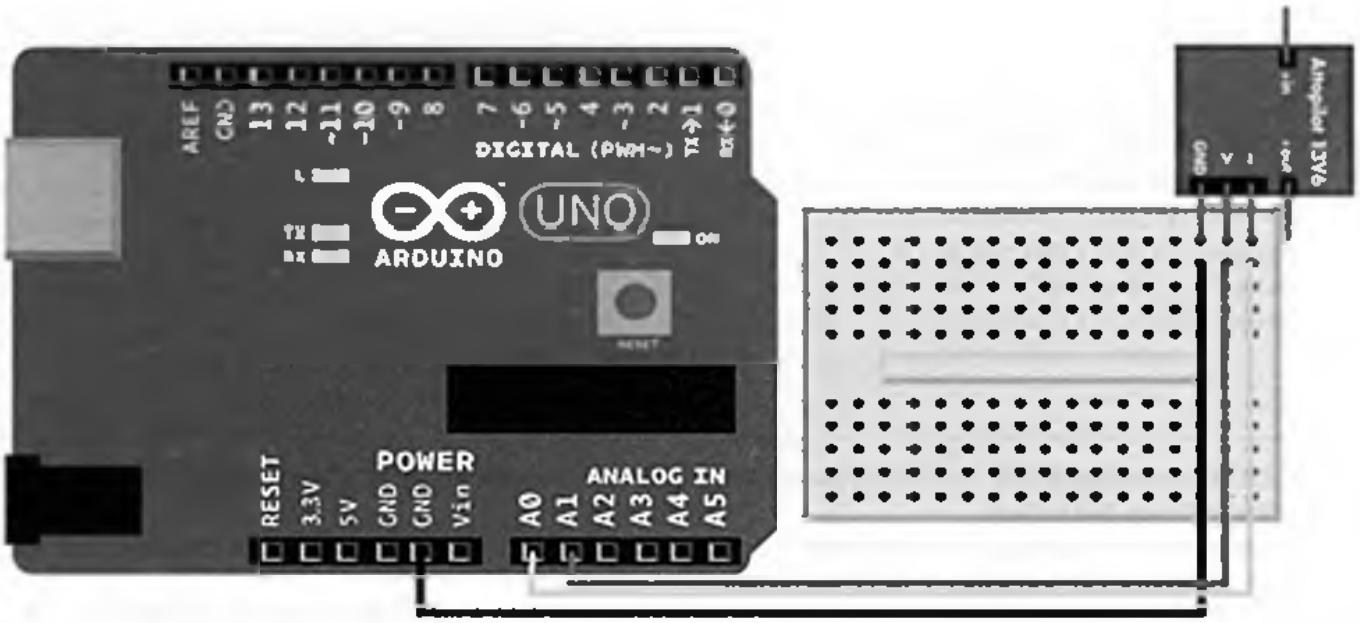


Figure 9.1 Connexions de l'AttoPilot pour Arduino

Exemple 9.1. attopilot_voltage.ino

```
// attopilot_voltage.ino - mesure le courant et la tension avec un AttoPilot
13,6V/45A
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int currentPin = A0;
int voltagePin = A1;
void setup()
{
  Serial.begin(115200);
  pinMode(currentPin, INPUT);
  pinMode(voltagePin, INPUT);
}
float current()
{
  float raw = analogRead(currentPin);
  Serial.println(raw);
  float percent = raw/1023.0;    // 1
  float volts = percent*5.0;     // 2
  float sensedCurrent = volts * 45 / 3.3;       // A/V  // 3
  return sensedCurrent; // A     // 4
}
float voltage()
{
  float raw = analogRead(voltagePin);
  float percent = raw/1023.0;
  float volts = percent*5.0;
  float sensedVolts = volts * 13.6 / 3.3;       // V/V  // 5
  return sensedVolts;    // V
}
```

```
void loop()
{
  Serial.print("Courant : ");
  Serial.print(current(),4);
  Serial.println(" A");
  Serial.print("Tension : ");
  Serial.print(voltage());
  Serial.println(" V");
  delay(200); // ms
}
```

1 La mesure maximale d'analogRead() est 1023 ; on calcule donc la mesure comme un pourcentage de ce maximum.
2 Cinq volts est la tension maximale d'analogRead(), ce qui correspond à une valeur brute de 1023.
3 Le facteur de conversion est 45 A mesurés / 3,3 V sortie ≈ 13,7 A/V. Voir aussi le Tableau 9.1.
4 Retourne le courant en ampères. C'est toujours une bonne idée d'inclure l'unité dans les commentaires.
5 Le facteur de conversion de la tension est `V mesurés max / V sortie max`.

AttoPilot pour Raspberry Pi

La Figure 9.2 illustre les connexions pour le Raspberry Pi et l'AttoPilot. Branchez selon le schéma puis exécutez le programme de l'Exemple 9.2.

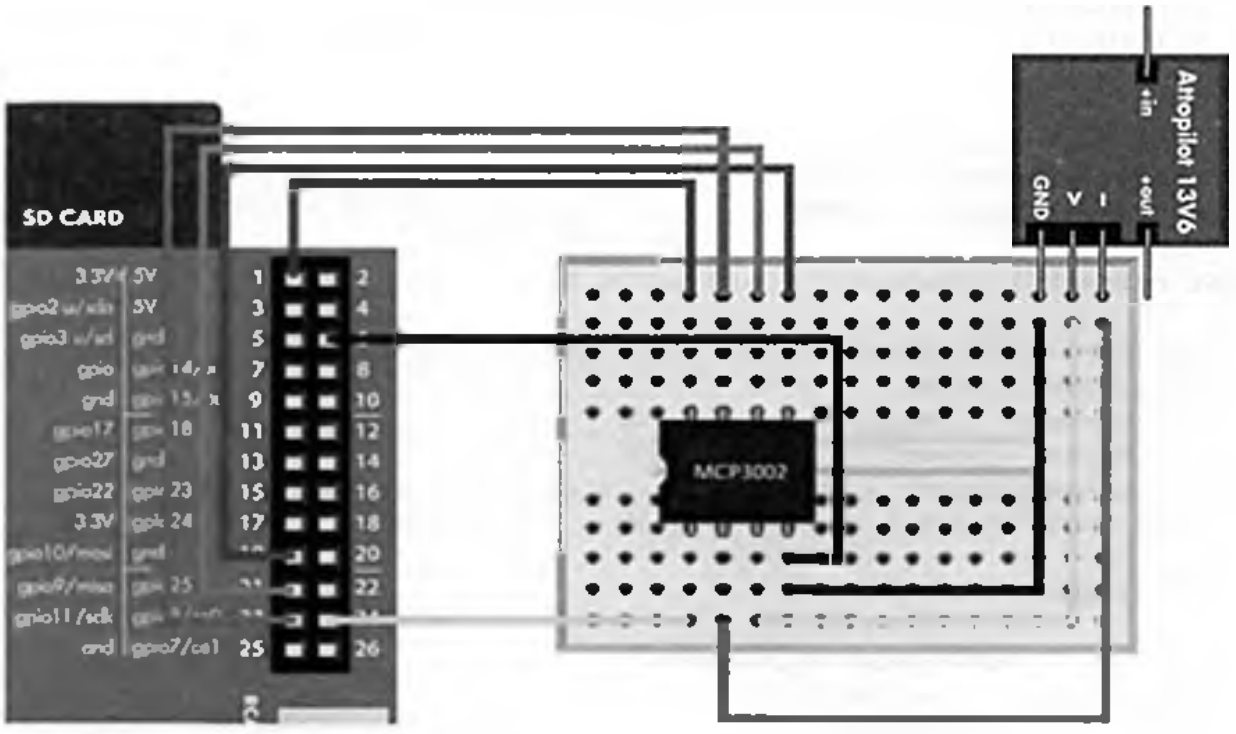


Figure 9.2 Connexions du Raspberry Pi pour l'AttoPilot

Exemple 9.2. attopilot_voltage.py

```
# attopilot_voltage.py - mesure le courant et la tension avec un AttoPilot
13,6V/45A
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp   # 1
def readVoltage():
        raw = mcp.readAnalog(0,1)       # 2
        percent = raw / 1023.0  # 3
        volts = percent * 3.3   # 4
        sensedVolts = volts * 13.6 / 3.3        # V/V   # 5
        return sensedVolts      # V
def readCurrent():
        raw = mcp.readAnalog(0,0)
        percent = raw / 1023.0
        volts = percent * 3.3
        sensedCurrent = volts * 45.0 / 3.3      # A/V # 6
        return sensedCurrent    # A
def main():
        while True:
                voltage = readVoltage()
                current = readCurrent()
                print("Courant %.2f A" % current)
                print("Tension %.2f V" % voltage)
                time.sleep(0.5) # s
if __name__ == "__main__":
        main()
```

1 Importe la bibliothèque du convertisseur analogique-numérique MCP3002. La bibliothèque *botbook_mcp3002.py* doit se trouver dans le même répertoire que ce programme (*attopilot_voltage.py*).
2 Lit le second canal. Les sorties de la tension et du courant de l'AttoPilot sont connectées à des canaux différents.
3 1023 est la valeur maximale de readAnalog(). Elle correspond à 3,3 V.
4 La tension maximale du port GPIO du Raspberry Pi est 3,3 V.
5 Les facteurs de conversion sont calculés comme pour l'Arduino. Le facteur de conversion de la tension est `V mesurés max / V sortie max`.
6 Le facteur de conversion est 45 A mesurés / 3,3 V sortie ≈ 13.7 A/V. Voir aussi le Tableau 9.1.

EXPÉRIENCE : EST-CE MAGNÉTIQUE ?

Un capteur à effet Hall mesure un champ magnétique. Les capteurs à effet Hall sont utilisés sur les vélos pour les compteurs de vitesse, où un aimant sur la roue permet au capteur de comptabiliser le nombre de tours.

Un champ magnétique fait dévier les électrons de leur chemin naturel, ce qui provoque une modification de la tension dans un conducteur. Il s'agit de l'effet Hall.

Le capteur indique un champ magnétique sous la forme d'une tension qui peut être lue comme un capteur de résistance analogique, avec analogRead() ou botbook_mcp3002.readAnalog(). Cet effet est implémenté dans les capteurs d'un grand nombre de fabricants.

Nous avons utilisé le module de détection magnétique KY-024 (article n°232563) de chez http://dx.com, qui est illustré à la Figure 9.3.

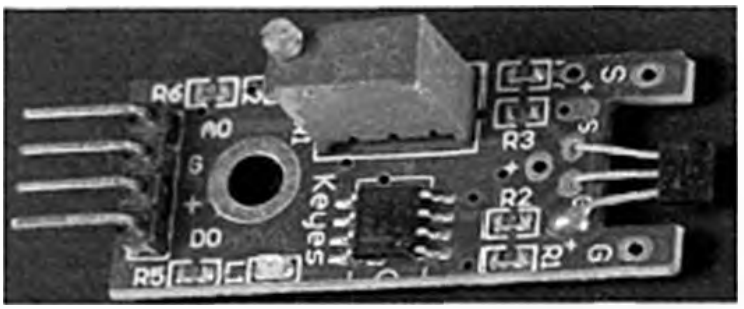


Figure 9.3 Capteur de détection magnétique KY-024

Capteur à effet Hall pour Arduino

La Figure 9.4 illustre les connexions pour Arduino.

Câblez selon le schéma puis exécutez le sketch de l'Exemple 9.3.

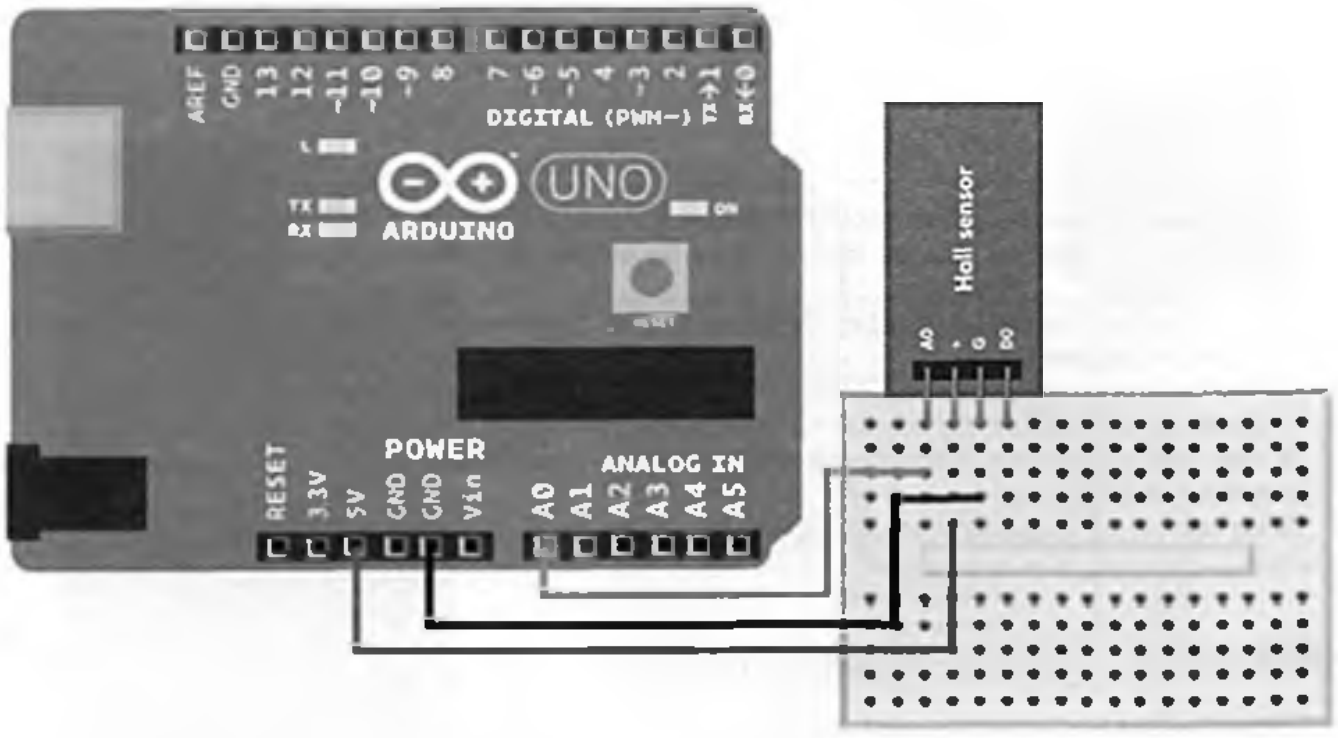


Figure 9.4 Connexions Arduino pour le capteur à effet Hall

Exemple 9.3. hall_sensor.ino

```
// hall_sensor.ino - affiche la valeur brute et le pôle magnétique
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int hallPin = A0;
int rawMagneticStrength = -1;   // 1
int zeroLevel = 527;    // 2
void setup() {
  Serial.begin(115200);
  pinMode(hallPin, INPUT);
}
void loop() {
  rawMagneticStrength = analogRead(hallPin);    // 3
  Serial.print("Force brute : ");
  Serial.println(rawMagneticStrength);
  int zeroedStrength = rawMagneticStrength - zeroLevel;
  // Si vous connaissez la formule de conversion du capteur à effet Hall
  // de tension en gauss, alors vous pouvez l'effectuer ici
  // zeroedStrength * conversion
  Serial.print("Force remise à zéro : ");
  Serial.println(zeroedStrength);
  if(zeroedStrength > 0) {
    Serial.println("Pôle Sud ");
  } else if(zeroedStrength < 0) {
    Serial.println("Pôle Nord ");
  }
  delay(600); // ms
}
```

1 Initialise à une valeur impossible comme résultat du capteur. Si cette valeur apparaît lors de l'exécution du code, vous savez donc qu'il y a un problème et qu'il faut déboguer.

2 C'est la valeur brute d'analogRead() quand il n'y a pas de champ magnétique. Notre capteur indique 527 quand il n'y a pas de champ magnétique. La spécification du fabricant indique 500 (valeur brute), sans doute pour un niveau logique de 5 V. Si vous voyez des valeurs différentes dans le Moniteur série Arduino quand vous n'avez pas de champ magnétique, il se peut que vous ayez à modifier la variable zeroLevel.

3 Le capteur à effet Hall fonctionne comme un capteur de résistance analogique, même s'il ne s'agit pas d'une résistance.

Capteur à effet Hall pour Raspberry Pi

La Figure 9.5 illustre les connexions pour Raspberry Pi.

Branchez selon le schéma et exécutez le code de l'Exemple 9.4.

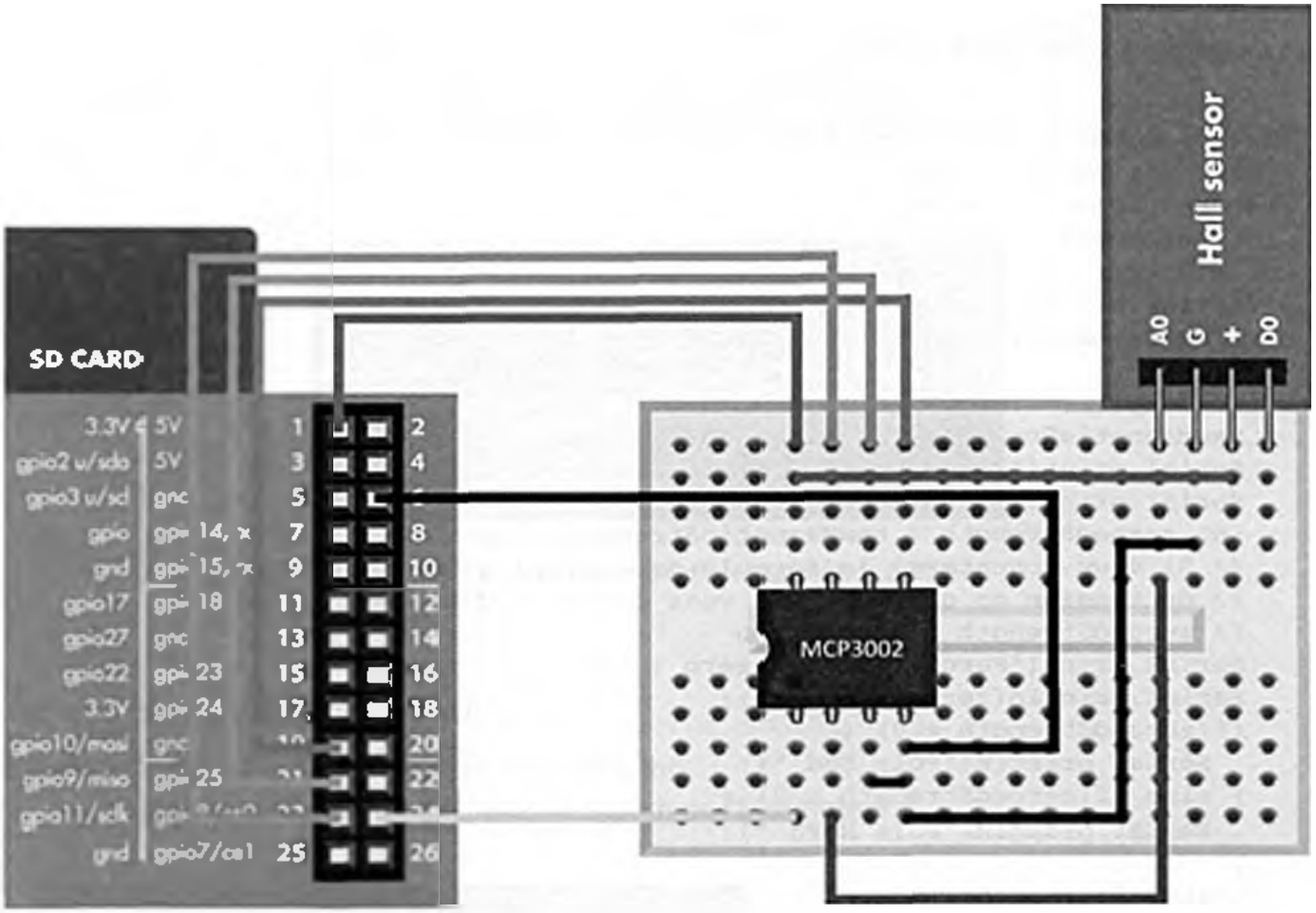


Figure 9.5 Connexions du Raspberry Pi avec un capteur à effet Hall

Exemple 9.4. hall_sensor.py

```
# hall_sensor.py - affiche la valeur brute et le pôle magnétique
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp   # 1
zeroLevel = 388 # 2
def main():
  while True:
        rawMagneticStrength = mcp.readAnalog()
        print("Force brute :  %i " % rawMagneticStrength)
        zeroedStrength = rawMagneticStrength - zeroLevel
        print("Force remise à zéro : %i " % zeroedStrength)
        if(zeroedStrength > 0):
                print("Pôle Sud")
        elif(zeroedStrength < 0):
                print("Pôle Nord")
        time.sleep(0.5)
if __name__ == "__main__":
  main()
```

1 La bibliothèque (*botbook_mcp3002.py*) doit se trouver dans le même répertoire que ce programme. Vous devez aussi installer la bibliothèque *spidev*, qui est importée par *botbook_mcp3002*. Lisez les commentaires de *botbook_mcp3002/botbook_mcp3002.py* ou la section *Installation de SpiDev* du chapitre 3.

2 `zeroLevel` est la sortie brute de `readAnalog()` quand aucun champ magnétique n'affecte le capteur. Notre test nous a indiqué la valeur 388 pour notre capteur. Les spécifications du fabricant indiquent une valeur brute de 500 pour un niveau logique de 5 V, ce qui nous donne une valeur de 330 pour `zeroLevel` à 3,3 V : `500 / (3.3/5) = 330`. Si vous constatez une valeur brute différente dans la sortie du programme quand il n'y a pas d'aimant, il se peut que vous ayez besoin de modifier la valeur de la variable `zeroLevel`.

EXPÉRIENCE : NORD MAGNÉTIQUE AVEC LA BOUSSOLE-ACCÉLÉROMÈTRE LSM303

La boussole-accéléromètre LSM303 (Figure 9.6) indique la direction du nord magnétique. Avec l'accéléromètre, elle peut corriger la mesure de son orientation.

Si vous avez l'habitude des courses d'orientation, vous savez comment tenir votre boussole horizontalement quand vous faites votre trajet ou une visée, mais avec ce capteur, on peut trouver le nord même s'il est penché.

> Les capteurs de boussole, comme le LSM303, sont sensibles aux interférences externes. Tenez la boussole éloignée des câbles d'alimentation et des grosses masses métalliques.

La carte LSM303 que nous avons utilisée ne comporte pas d'indication pour le nord. Pour la marquer vous-même, tournez la carte de telle sorte que le texte « SA0 » soit orienté afin de pouvoir être lu de gauche à droite. Le nord est le côté droit de la carte quand le texte « SA0 » est correctement orienté.

Dans les valeurs de sortie, le nord est indiqué par un cap égal à 0.

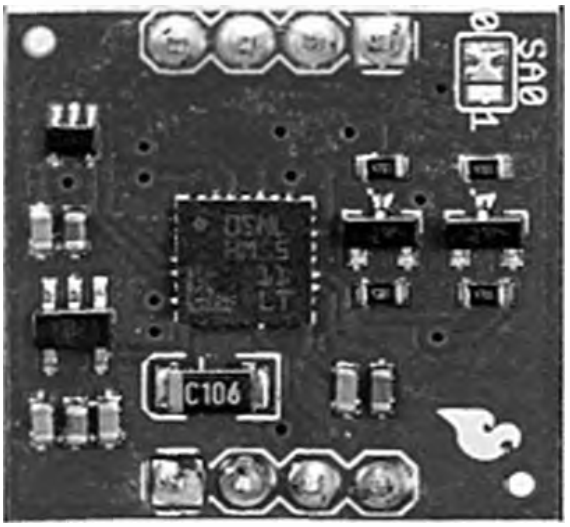


Figure 9.6 Boussole-accéléromètre LSM303 de chez SparkFun

> Vous n'avez pas de boussole sous la main ?
>
> Dans l'hémisphère nord, les antennes satellites pointent vers le sud. La plupart d'entre elles, comme celles que l'on utilise pour recevoir la télévision par satellite, sont installées de manière statique. Cela signifie qu'elles pointent vers des satellites géostationnaires qui tournent autour de la Terre à la même vitesse que la rotation terrestre.
>
> Les satellites sont donc en orbite à l'est au-dessus de l'équateur et les antennes paraboliques pointent vers l'équateur.

Calibrage du module

La boussole doit être calibrée pour fournir des valeurs correctes. Vous pouvez la tester sans la calibrer et vous obtiendrez des valeurs, mais elles ne seront correctes qu'une fois que vous l'aurez calibrée.

Voici les étapes du calibrage :

- Montez le circuit de la plateforme choisie (Arduino ou Raspberry Pi).
- Exécutez le programme pour voir les valeurs.
- Modifiez `runningMode = 0` dans le code pour placer l'appareil en mode de calibrage. Dans ce mode, le programme n'affichera que les valeurs minimale et maximale de chaque axe.
- Définissez au départ les valeurs maximales à zéro. Pour le Raspberry Pi, modifiez `magMax[]` et `magMin[]`. Pour Arduino, modifiez chacune des six variables `magMax_x`, `magMax_y` ... `magMin_z`.
- Faites tournoyer l'appareil en lui faisant dessiner la figure d'un huit dans l'air. Continuez jusqu'à ce que les valeurs se stabilisent. Si vous n'arrivez pas à trouver de valeur supérieure ou de valeur inférieure, c'est que vous avez trouvé le min et le max.
- Codez en dur ces valeurs dans votre code. Il s'agit des mêmes valeurs maximales que vous avez mises à zéro précédemment, `magMax[]` et `magMin[]` ou `magMax_x`...

Une fois le calibrage effectué, modifiez `runningMode = 0` afin d'utiliser le capteur normalement et visualisez les caps.

Essayez d'exécuter le code et de calibrer l'appareil. Une fois que vous avez testé la boussole, vous pouvez lire les détails d'implémentation (si vous le souhaitez) dans la section *Protocole du LSM303*.

LSM303 pour Arduino

La Figure 9.7 illustre le circuit pour Arduino. Branchez en suivant le schéma et exécutez le sketch de l'Exemple 9.5.

> N'oubliez pas de calibrer votre module (voir plus haut la section *Calibrage du module*).

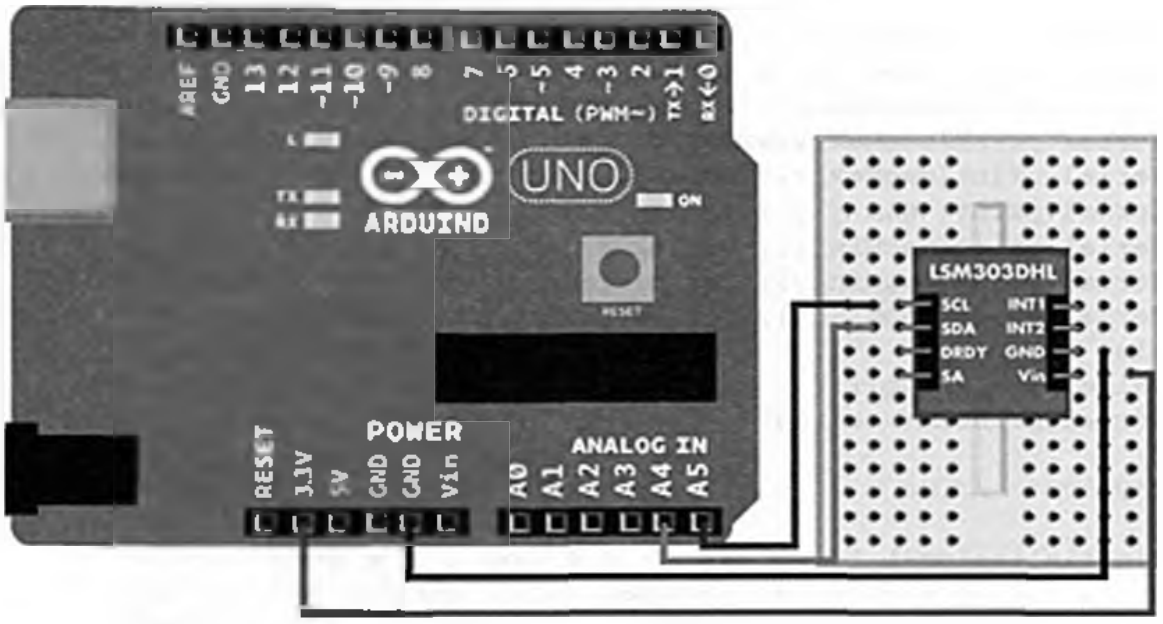


Figure 9.7 Circuit de la boussole-accéléromètre LSM303 pour Arduino

Exemple 9.5. lsm303.ino

```
// lsm303.ino - utilisation normale et calibrage de la
boussole-accéléromètre LSM303DLH
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
#include <Wire.h>
const char accelerometer_address = 0x30 >> 1;    // 1
const char magnetometer_address = 0x3C >> 1;
const int runningMode = 0;       // 2
float magMax_x = 0.1; float magMax_y = 0.1; float magMax_z = 0.1;  // 3
float magMin_x = -0.1; float magMin_y = -0.1; float magMin_z = -0.1;
float acc_x = 0; float acc_y = 0; float acc_z = 0; // 4
float mag_x = 0; float mag_y = 0; float mag_z = 0;
int heading = 0;
void setup() {
  Serial.begin(115200);
  Wire.begin(); // 5
  Serial.println("Initialise la boussole");
  initializelsm();       // 6
  delay(100);
  Serial.println("Début de la lecture du cap");
}
void loop() {
  updateHeading();       // 7
  if(runningMode == 0) {          // 8
    magMax_x = max(magMax_x, mag_x);
    magMax_y = max(magMax_y, mag_y);
    magMax_z = max(magMax_z, mag_z);
    magMin_x = min(magMin_x, mag_x);
    magMin_y = min(magMin_y, mag_y);
```

```
    magMin_z = min(magMin_z, mag_z);
    Serial.print("Max x y z : ");
    Serial.print(magMax_x); Serial.print(" ");
    Serial.print(magMax_y); Serial.print(" ");
    Serial.print(magMax_z); Serial.print(" ");
    Serial.print("Min x y z : ");
    Serial.print(magMin_x); Serial.print(" ");
    Serial.print(magMin_y); Serial.print(" ");
    Serial.print(magMin_z); Serial.println(" ");
  } else {
    calculateHeading();
    Serial.println(heading);     // 9
  }
  delay(100); // ms
}
void initializeIsm() {
    write_i2c(accelerometer_address, 0x20, 0x27);          // 10
    write_i2c(magnetometer_address, 0x02, 0x00);           // 11
}
void updateHeading() {
  updateAccelerometer();
  updateMagnetometer();
}
void updateAccelerometer() {     // 12
  Wire.beginTransmission(accelerometer_address);
  Wire.write(0x28 | 0x80);       // 13
  Wire.endTransmission(false);
  Wire.requestFrom(accelerometer_address, 6, true);      // 14
  int i = 0;
  while(Wire.available() < 6) {
    i++;
    if(i > 1000) {
      Serial.println("Erreur de lecture de l'accéléromètre i2c");
      return;
    }
  }
  uint8_t  axel_x_l = Wire.read();       // 15
  uint8_t  axel_x_h = Wire.read();
  uint8_t  axel_y_l = Wire.read();
  uint8_t  axel_y_h = Wire.read();
  uint8_t  axel_z_l = Wire.read();
  uint8_t  axel_z_h = Wire.read();
  acc_x = (axel_x_l | axel_x_h << 8) >> 4; // 16
  acc_y = (axel_y_l | axel_y_h << 8) >> 4;
  acc_z = (axel_z_l | axel_z_h << 8) >> 4;
}
void updateMagnetometer() { // 17
  Wire.beginTransmission(magnetometer_address);
  Wire.write(0x03);
  Wire.endTransmission(false);
  Wire.requestFrom(magnetometer_address, 6, true);
  int i = 0;
```

```
  while(Wire.available() < 6) {
    i++;
    if(i > 1000) {
      Serial.println("Erreur de lecture du magnétomètre i2c");
      return;
    }
  }
  uint8_t  axel_x_h = Wire.read();
  uint8_t  axel_x_l = Wire.read();
  uint8_t  axel_y_h = Wire.read();
  uint8_t  axel_y_l = Wire.read();
  uint8_t  axel_z_h = Wire.read();
  uint8_t  axel_z_l = Wire.read();
  mag_x = (int16_t)(axel_x_l | axel_x_h << 8);
  mag_y = (int16_t)(axel_y_l | axel_y_h << 8);
  mag_z = (int16_t)(axel_z_l | axel_z_h << 8);
}
//Cap vers le nord
void calculateHeading() {
  // Vecteur unitaire
  float dot = acc_x*acc_x + acc_y*acc_y + acc_z*acc_z;  // 18
  float magnitude = sqrt(dot);  // 19
  float nacc_x = acc_x / magnitude;      // 20
  float nacc_y = acc_y / magnitude;
  float nacc_z = acc_z / magnitude;
  // Application du calibrage
  mag_x = (mag_x - magMin_x) / (magMax_x - magMin_x) * 2 - 1.0; // 21
  mag_y = (mag_y - magMin_y) / (magMax_y - magMin_y) * 2 - 1.0;
  mag_z = (mag_z - magMin_z) / (magMax_z - magMin_z) * 2 - 1.0;
  // Est
  float ex = mag_y*nacc_z - mag_z*nacc_y;        // 22
  float ey = mag_z*nacc_x - mag_x*nacc_z;
  float ez = mag_x*nacc_y - mag_y*nacc_x;
  dot = ex*ex + ey*ey + ez*ez; // 23
  magnitude = sqrt(dot);
  ex /= magnitude;
  ey /= magnitude;
  ez /= magnitude;       // 24
  // Projection
  float ny = nacc_z*ex - nacc_x*ez;      // 25
  float dotE = -1 * ey; // 26
  float dotN = -1 * ny;
  // Angle
  heading = atan2(dotE, dotN) * 180 / M_PI;      // 27
  heading = round(heading);
  if (heading < 0) heading +=360; // 28
}
void write_i2c(char address, unsigned char reg, const uint8_t data)
{
  Wire.beginTransmission(address);
  Wire.write(reg);
```

```
    Wire.write(data);
    Wire.endTransmission();
}
```

1 Les adresses de l'accéléromètre et du magnétomètre ont besoin d'être décalées au niveau des bits (voir la section Opérations au niveau des bits du chapitre 8).
2 Le mode d'exploitation normal de ce sketch (affichage du cap de la boussole) est runningMode=1. Pour calibrer la boussole, voir précédemment la section *Calibrage du module*.
3 Les données de calibrage initial vous aident à obtenir des mesures même avant d'avoir trouvé les valeurs exactes de votre appareil (avec runningMode=0).
4 Les variables du cap en cours sont initialisées à zéro.
5 *Wire.h* est la bibliothèque standard I2C pour Arduino.
6 Appelle la fonction d'initialisation.
7 Cette fonction met à jour les variables globales, et il n'y a donc pas besoin de valeur de retour.
8 Le mode de calibrage n'affiche que le max et le min.
9 Le mode d'exploitation normal affiche l'accélération et le cap corrigé.
10 Active l'accéléromètre avec les valeurs par défaut qui sont extraites de la notice technique de ce module.
11 Active le magnétomètre et l'initialise en conversion continue.
12 Lit le capteur selon le protocole décrit dans la notice technique, puis met à jour les variables globales.
13 0x80 est égal à 0b10000000 : le bit de poids fort est 1, et les sept autres bits sont à zéro. Un XOR au niveau du bit de 0x80 et de n'importe quel autre nombre a pour résultat de modifier le bit de poids fort à 1. 0x28 est le registre OUT_X_L_A, d'après la notice technique du composant.
14 Lit six octets.
15 Chaque valeur brute est décomposée en deux octets, la partie inférieure (axel_x_l) et la partie supérieure (axel_x_h) qui est celle de poids fort.
16 Chaque valeur brute est composée de 8 + 4 = 12 bits. L'octet de poids fort contient les 8 bits de poids fort. La partie inférieure contient les quatre derniers bits qui sont ignorés. Voir aussi la section Opérations au niveau des bits du chapitre 8.
17 Les valeurs brutes du magnétomètre (boussole) sont lues de la même manière qu'avec updateAccelerometer().
18 Prend un produit scalaire du vecteur...
19 ...pour calculer sa magnitude (longueur).
20 Divise chaque dimension par la longueur pour obtenir un vecteur dont la longueur est un. On a donc un vecteur orienté vers le haut (nacc_x, nacc_y, nacc_z).
21 Applique les données du calibrage.
22 L'est est à 90 degrés du nord, et également à 90 degrés du haut. Un calcul de vecteur de produit cartésien fournit le vecteur qui est perpendiculaire à ces deux autres vecteurs.
23 L'est est normalisé, comme on l'a fait pour le haut.
24 Après la normalisation, le vecteur (ox, oy, oz) a une longueur d'un et pointe vers l'est. La signification du symbole /= est similaire à +=. Par exemple test /= 2 est identique à test = test/2.
25 Utilise un produit cartésien pour trouver le vecteur nord dans le plan horizontal de la gravitation.
26 Projection du vecteur nord N du plan NE vers le plan XY.
27 Trouve l'angle entre l'axe *y* et le vecteur nord projeté. Convertit les radians en degrés (deg=rad/(2*pi)*360).
28 On fait en sorte que le cap de la boussole soit compris entre 0 et 360 degrés.

LSM303 pour Raspberry Pi

La Figure 9.8 illustre le circuit pour Raspberry Pi. Câblez en suivant le schéma puis exécutez le code de l'Exemple 9.6.

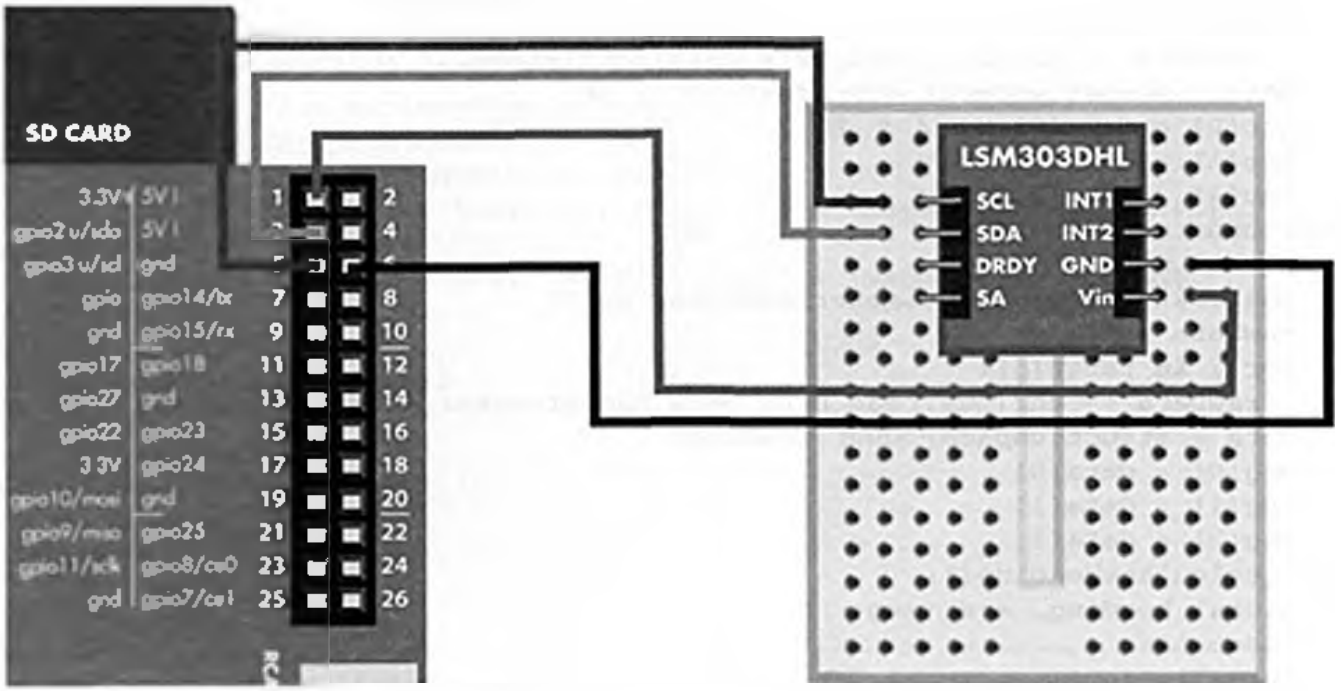


Figure 9.8 Circuit de la boussole-accéléromètre LSM303 pour Raspberry Pi

N'oubliez pas de calibrer le module (voir plus haut la section *Calibrage du module*).

Exemple 9.6. lsm303.py

```
# lsm303.py - utilisation normale et calibrage de la boussole-accéléromètre
LSM303DLH
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import smbus # sudo apt-get -y install python-smbus
import struct
import math
accelerometer_address = 0x30 >> 1
magnetometer_address = 0x3C >> 1
calibrationMode = True  # 1
magMax = [ 0.1, 0.1, 0.1 ]      # 2
magMin = [ 0.1, 0.1, 0.1 ]
acc = [ 0.0, 0.0, 0.0 ] # 3
mag = [ 0.0, 0.0, 0.0 ]
heading = 0
def initlsm():
  global bus
  bus = smbus.SMBus(1)
  bus.write_byte_data(accelerometer_address, 0x20, 0x27)        # 4
```

```
  bus.write_byte_data(magnetometer_address, 0x02, 0x00)  # 5
def updateAccelerometer():
  global acc    # 6
  bus.write_byte(accelerometer_address, 0x28 | 0x80)    # 7
  rawData = ""
  for i in range(6):
    rawData += chr(bus.read_byte_data(accelerometer_address, 0x28+i)) # 8
  data = struct.unpack('<hhh',rawData) # 9
  acc[0] = data[0] >> 4 # 10
  acc[1] = data[1] >> 4
  acc[2] = data[2] >> 4
def updateMagnetometer():       # 11
  global mag
  bus.write_byte(magnetometer_address, 0x03)
  rawData = ""
  for i in range(6):
    rawData += chr(bus.read_byte_data(magnetometer_address, 0x03+i))
  data = struct.unpack('>hhh',rawData)
  mag[0] = data[0]
  mag[1] = data[1]
  mag[2] = data[2]
def calculateHeading():
  global heading, acc, mag
  #normalise
  normalize(acc)          # 12
  #utilise les données de calibrage
  mag[0] = (mag[0] - magMin[0]) / (magMax[0] - magMin[0]) * 2.0 - 1.0 # 13
  mag[1] = (mag[1] - magMin[1]) / (magMax[1] - magMin[1]) * 2.0 - 1.0
  mag[2] = (mag[2] - magMin[2]) / (magMax[2] - magMin[2]) * 2.0 - 1.0
  e = cross(mag, acc)    # 14
  normalize(e)  # 15
  n = cross(acc,e)        # 16
  dotE = dot(e,[0.0, -1.0, 0.0])          # 17
  dotN = dot(n,[0.0, -1.0, 0.0])
  heading = round(math.atan2(dotE, dotN) * 180.0 / math.pi)      # 18
  if heading < 0:         # 19
    heading += 360        # 20
def normalize(v):         # 21
  magnitude = math.sqrt(dot(v,v))
  v[0] /= magnitude
  v[1] /= magnitude
  v[2] /= magnitude
def dot(v1, v2):          # 22
  return v1[0]*v2[0] + v1[1]*v2[1] + v1[2]*v2[2]
def cross(v1, v2):        # 23
  vr = [0.0, 0.0, 0.0]
  vr[0] = v1[1] * v2[2] - v1[2] * v2[1]
  vr[1] = v1[2] * v2[0] - v1[0] * v2[2]
  vr[2] = v1[0] * v2[1] - v1[1] * v2[0]
  return vr
def main():
  initIsm()
```

```
  while True:
    updateAccelerometer()          # 24
    updateMagnetometer()
    if calibrationMode: # 25
      magMax[0] = max(magMax[0], mag[0])
      magMax[1] = max(magMax[1], mag[1])
      magMax[2] = max(magMax[2], mag[2])
      magMin[0] = min(magMin[0], mag[0])
      magMin[1] = min(magMin[1], mag[1])
      magMin[2] = min(magMin[2], mag[2])
      print("magMax = [ %.1f, %.1f, %.1f ]" % (magMax[0], magMax[1],
magMax[2]))
      print("magMin = [ %.1f, %.1f, %.1f ]" % (magMin[0], magMin[1],
magMin[2]))
    else:
      calculateHeading()           # 26
      print(heading)
    time.sleep(0.5)
if __name__ == "__main__":
  main()
```

1 Quand on définit `calibrationMode` à True, c'est le processus de calibrage qui est exécuté au lieu du mode normal d'exploitation. Voir la section *Calibrage du module* pour plus d'information.

2 Nous vous fournissons des valeurs de calibrage pour que vous puissiez tester votre boussole sans calibrage (mais il faut quand même la calibrer).

3 Les valeurs du capteur de la dernière lecture et les valeurs calculées sont stockées dans les variables globales acc, mag, et heading.

4 Active l'accéléromètre en utilisant les codes de contrôle extraits de la notice technique.

5 Active le magnétomètre.

6 Pour mettre à jour une variable globale, elle doit être explicitement déclarée en tant que globale au début de la fonction.

7 Envoie un message où le premier bit est 1 (0x80 == 0b10000000) et la fin est l'adresse du registre OUT_X_L_A (0x28 == 0b 10 1000). Une opération de OR exclusif (XOR) combine les deux à 0b10101000. Voir la section *Opérations au niveau des bits* du chapitre 8 pour de plus amples informations sur ce type d'opérations.

8 Lit six octets, en partant de l'adresse de l'accéléromètre 0x28.

9 Extrait les valeurs des entiers des octets lus à partir du capteur (rawData). Les paramètres d'extraction sont les suivants : little endian (<), entier court (2 octets sur 16 bits, h). Struct.unpack() est nécessaire pour que vous puissiez définir la longueur exacte des entiers que vous lisez.

10 data[0] se compose de 16 bits, mais la valeur est contenue dans les 12 premiers bits. Les 4 derniers bits doivent être ignorés. Le décalage des bits par la droite réalise cela. Voir aussi la section *Opérations au niveau des bits* du chapitre 8.

11 updateMagnetometer() lit les valeurs comme updateAccelerometer().

12 Normalise le vecteur d'accélération ; ce qui signifie qu'on le convertit en un vecteur unitaire qui pointe vers le haut, mais après sa normalisation sa longueur est égale à un.

13 Applique les données du calibrage.

14 L'est e est à 90 degrés du nord (mag) et à 90 degrés du haut (acc).

15 Après la normalisation, e est un vecteur unitaire (longueur un), pointant vers l'est.

16 Le nouveau vecteur n est maintenant perpendiculaire aux deux valeurs de l'accélération et de l'est. À ce moment-là, le plan NE est aligné (exactement au niveau) avec une gravitation horizontale. Il n'est pas encore aligné avec le plan horizontal XY de l'appareil.
17 Utilise un produit scalaire pour projeter le vecteur nord sur le plan horizontal XY de l'appareil. Le vecteur est est nécessaire car le plan entier NE contrôle la projection.
18 L'angle entre Y et le nord est le cap. Il est converti de radians (où un cercle complet est égal à 2*pi) en degrés (où un cercle complet est égal à 360 degrés) par une simple division.
19 Si le cap est en dessous de zéro...
20 ...on fait en sorte que le cap de la boussole soit toujours compris entre 0 et 360 degrés.
21 La normalisation des vecteurs crée un vecteur unitaire qui a une longueur d'un (magnitude), et qui pointe dans la même direction que le vecteur original.
22 La formule du produit scalaire du vecteur est extraite d'un livre de maths. Ce programme utilise le produit scalaire pour calculer la magnitude (length) d'un vecteur. La magnitude mag du vecteur v est égale à sqrt(dot(v)).
23 La formule du produit cartésien du vecteur provient aussi d'un livre de maths. Dans ce programme, on l'utilise pour trouver un vecteur perpendiculaire aux deux autres vecteurs. Cela signifie que pour c=cross(a,b), c est perpendiculaire à la fois à a et b.
24 updateAccelerometer() et updateMagnetometer() fonctionnent en modifiant des variables globales si bien qu'elles n'ont pas besoin de valeur de retour.
25 En définissant calibrationMode à 0 le programme exécute le processus de calibrage.
26 C'est le mode d'exploitation normal où l'on affiche le cap en degrés de la boussole avec une inclinaison corrigée.

Protocole du LSM303

> Avant de lire cette section, il est préférable de construire le projet et d'exécuter le code. Vous pouvez utiliser le code sans vous soucier de la manière dont il marche, mais si vous êtes curieux, reprenez la lecture de cette section.

Pour fournir les valeurs mesurées, le LSM303 utilise le protocole standard I2C. Comme I2C est assez strictement défini, il est en général plus facile que les protocoles similaires comme SPI.

Le LSM303 est un esclave I2C, de telle sorte que le microcontrôleur (Arduino ou Raspberry Pi) est le maître. Le maître initialise la communication en demandant des valeurs. La communication I2C se compose simplement d'envois et de lectures de valeurs qui sont spécifiées dans la notice technique du composant. Vous trouverez la notice technique de ce composant en cherchant sur le web « LSM303DLH datasheet ».

Si vous voulez comprendre le protocole, la communication I2C a été expliquée en détail à la Figure 8.3. Comme de nombreux programmes I2C, le code de la boussole utilise des nombres hexadécimaux (0xA == 10) et le décalage de bits (0b01 << 1 == 0b10). Ces concepts sont expliqués au chapitre 8 dans les sections *Hexadécimal, binaire et autres systèmes de numération* et *Opérations au niveau des bits*.

Calcul du cap de la boussole

Le capteur sait déjà où se trouve le nord. Il fournit cette valeur sous la forme d'un vecteur tridimensionnel n = [nx, ny, nz]. Le capteur de la boussole est réellement tridimensionnel et ce vecteur pointant vers le nord est déjà correct. Dans ces conditions, pourquoi avoir besoin de ce calcul ?

> Le capteur fonctionnera même si vous êtes ignorant des vecteurs, mais la lecture de ces explications nécessite une connaissance minimale des vecteurs.

On souhaite en général que le cap d'une boussole soit un nombre compris entre 0 et 360 degrés. Un cap a deux dimensions. Le cap de la boussole indique de combien on a tourné à droite à partir du nord. La rotation est mesurée en degrés, dans le sens des aiguilles d'une montre, quand on tourne à droite.

Pour convertir le vecteur tridimensionnel du nord en degré, on utilise le vecteur du haut de l'accéléromètre.

Tous les vecteurs sont relatifs à leur appareil, si bien que Y pointe dans la direction avant de l'appareil. Ainsi, le vecteur [0, 1, 0] pointe directement vers l'avant de l'appareil.

Arduino	Raspberry Pi	Valeurs	Explication
heading	heading	int, 0 deg .. 360 deg	Angle entre le nord et l'avant (y). De combien de degrés a-t-on tourné à droite ?
mag_x, mag_y...	mag[]	entier signé, liste d'entiers signés	Vecteur tridimensionnel pointant vers le nord
acc_x, acc_y...	acc[]	entier signé, liste d'entiers signés	Vecteur tridimensionnel pointant vers le haut
[0, 0, 1]			Vecteur pointant vers le haut de l'appareil (Z)
[0, 1, 0]			Vecteur pointant vers l'avant (Y)
[1, 0, 0]			Vecteur pointant vers la droite (X)

Tableau 9.2 Variables et définitions du LSM303

Le code Arduino et Raspberry Pi utilisent la même logique. Dans un souci de clarté, nous allons employer Python ici, mais le code Arduino fonctionne de la même manière et se sert de noms de variables similaires.

On commence par lire l'accélération et le nord à partir du capteur et on obtient ces vecteurs :

- acc (gravitation vers le haut, 3D)
- mag (nord, 3D)

L'angle formé par ces deux vecteurs peut représenter n'importe quelle valeur.

Le calibrage est ensuite appliqué à mag, le vecteur pointant vers le nord.

On doit ensuite calculer le vecteur est qui est perpendiculaire à la fois aux vecteurs mag et acc. Il y a donc un angle de 90 degrés entre acc (gravitation vers le haut) et l'est, et il y a aussi un angle de 90 degrés entre mag et l'est. Vous pouvez ainsi calculer le vecteur est avec un produit cartésien de vecteurs.

```
e = cross(mag, acc)
e = normalize(e)
```

Après la normalisation, e est un vecteur unitaire (de longueur 1) pointant vers l'est. Cet est est perpendiculaire à la gravitation vers le haut.

À présent, le vecteur nord (n) et le vecteur est (e) forment un plan NE qui est probablement toujours dans un angle du plan XY horizontal par rapport à l'appareil.

Pour aligner les plans n et e à la gravitation horizontale,

```
n = cross(acc, e)
```

Vous avez maintenant le nord n et l'est e dans le plan de la gravitation horizontale.

Le cap de la boussole doit être calculé pour l'utilisateur. Le cap indique le nombre de degrés dont l'appareil a tourné à droite par rapport au nord. Un vecteur est projeté à partir de l'appareil sur le plan NE. Ensuite, on peut calculer l'angle entre le vecteur projeté et le nord :

```
dotE = dot(e,[0.0, -1.0, 0.0])
dotN = dot(n,[0.0, -1.0, 0.0])
headingRad = math.atan2(dotE, dotN)
headingDeg = headingRad / (2*math.pi) * 360
```

L'utilisateur obtient pour finir le cap de la boussole en degrés.

EXPÉRIENCE : CAPTEUR À EFFET HALL

Un capteur à effet Hall (Figure 9.9) détecte la proximité d'un aimant et il est souvent utilisé pour mesurer la vitesse de rotation d'une roue. Avant le GPS, de nombreux compteurs de vitesse de vélo utilisaient des capteurs à effet Hall.

Quand vous placez un aimant près d'un capteur à effet Hall, le code de cette expérience affiche « capteur déclenché ».

Il existe de nombreux capteurs à effet Hall disponibles chez plusieurs fabricants et nous avons employé un capteur magnétique robuste et bon marché de chez http://dx.com (article n°141363). Ce capteur a une jolie LED intégrée qui s'allume quand un aimant est proche. Les couleurs du câblage sont un peu étranges car le noir est pour les données (le fil noir va sur D2 et non pas à la masse comme d'habitude), le bleu va à la masse et le marron au +5 V.

Figure 9.9 Capteur à effet Hall

Capteur à effet Hall pour Arduino

La Figure 9.10 illustre le schéma de câblage pour Arduino. Branchez en suivant le schéma et exécutez le code de l'Exemple 9.7.

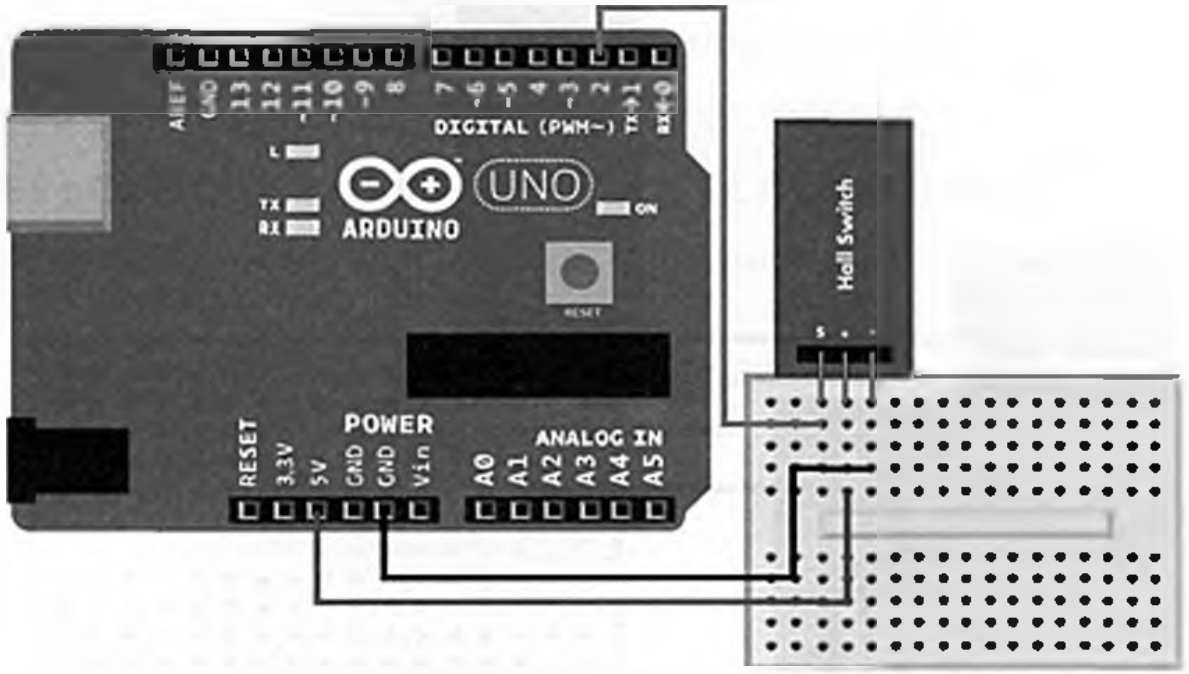


Figure 9.10 Circuit Arduino pour un capteur à effet Hall

Exemple 9.7. hall_switch.ino

```
// hall_switch.ino - écrit sur le port série si un aimant déclenche le
capteur
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int switchPin=2;
void setup() {
  Serial.begin(115200);
  pinMode(switchPin, INPUT);
  digitalWrite(switchPin, HIGH);
}
void loop() {
  int switchState=digitalRead(switchPin);        // 1
  if (switchState == LOW) {
    Serial.println("OUI, aimant à proximité");
  } else {
     Serial.println("non");
  }
  delay(50);
}
```

1 Il s'agit d'un simple interrupteur numérique, comme un bouton.

Capteur à effet Hall pour Raspberry Pi

La Figure 9.11 illustre le circuit pour Raspberry Pi. Câblez en suivant le schéma et exécutez le code de l'Exemple 9.8.

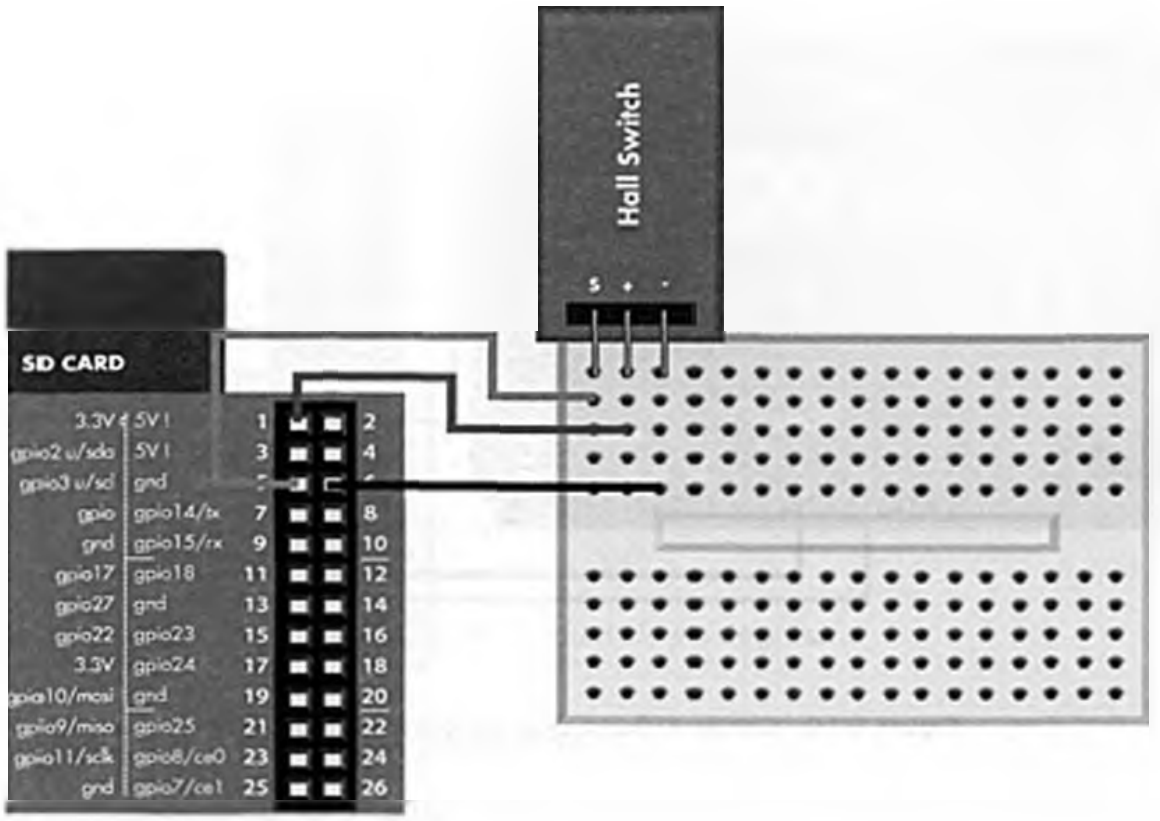


Figure 9.11 Circuit Raspberry Pi pour capteur à effet Hall

Exemple 9.8. hall_switch.py

```
# hall_switch.py - écrit à l'écran si l'aimant déclenche le capteur
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio      # 1
def main():
        switchPin = 3    # pull-up interne # 2
        gpio.mode(switchPin, "in")
        while (True):
                switchState = gpio.read(switchPin)       # 3
                if(switchState == gpio.LOW):
                        print "Capteur déclenché"
                time.sleep(0.3)
if __name__ == "__main__":
        main()
```

1 Assurez-vous qu'il y a une copie de la bibliothèque *botbook_gpio.py* dans le même répertoire que ce programme. Vous pouvez télécharger cette bibliothèque ainsi que tous les exemples de code à partir de http://botbook.com. Voir la section *GPIO sans root* du chapitre 1 pour des informations sur la configuration de l'accès GPIO du Raspberry Pi.

2 Pour éviter l'instabilité de la broche, on a besoin d'une résistance pull-up. Heureusement, les résistances pull-up sont incluses sur les broches 2 et 3 du connecteur GPIO du Raspberry Pi.

3 Un capteur à effet Hall est un simple interrupteur numérique et il fonctionne donc comme un bouton.

PROJET : CONTRÔLE D'UNE PILE SOLAIRE PAR INTERNET

Transformez votre Raspberry Pi en serveur web et surveillez la tension de vos piles solaires à distance (Figure 9.12).

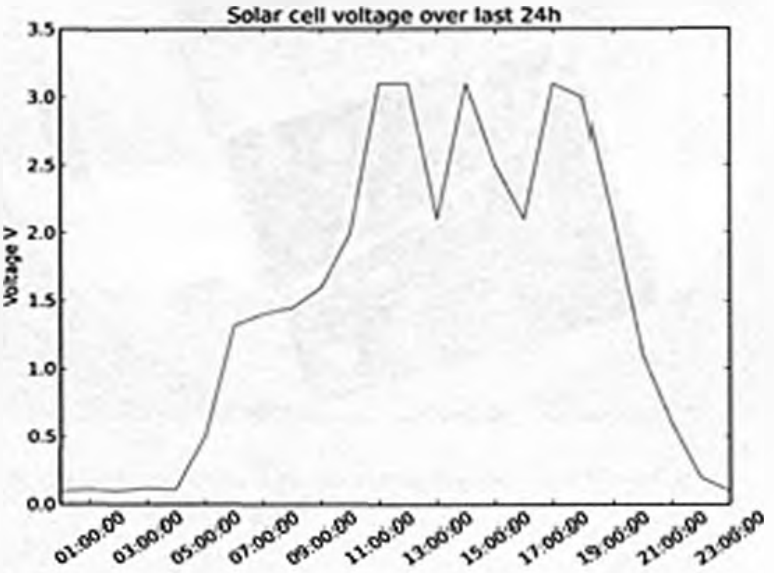


Figure 9.12 Graphique de la puissance de piles solaires

Ce que vous allez apprendre

Dans le projet *Contrôle d'une pile solaire par Internet*, vous allez apprendre à :

- Mesurer la tension de vos piles solaires, et l'afficher sur votre propre serveur web.
- Transformer le Raspberry Pi en serveur web (avec le serveur web le plus populaire au monde).
- Créer des tâches planifiées avec cron, un planificateur qui continue à s'exécuter même si le Raspberry Pi est redémarré.
- Tracer des graphiques avec la bibliothèque Python matplotlib.

Les grands serveurs web publics utilisent les mêmes techniques que vous allez apprendre ici. Nous enseignons Apache et cron à nos étudiants qui apprennent Linux pour exploiter des serveurs, si bien que ces techniques ne sont pas spécifiques aux systèmes embarqués ni aux robots.

Connexion des piles solaires

Est-ce que vous vous souvenez de la lampe Solvinden de chez IKEA que nous avons utilisée pour construire notre Dôme caméléon au chapitre 7 ? Dans ce projet, nous allons utiliser les piles solaires que nous n'avions pas employées.

Si vous ne vous êtes pas servi d'une lampe Solvinden, il suffit de modifier les instructions pour qu'elles conviennent aux piles solaires que vous allez utiliser.

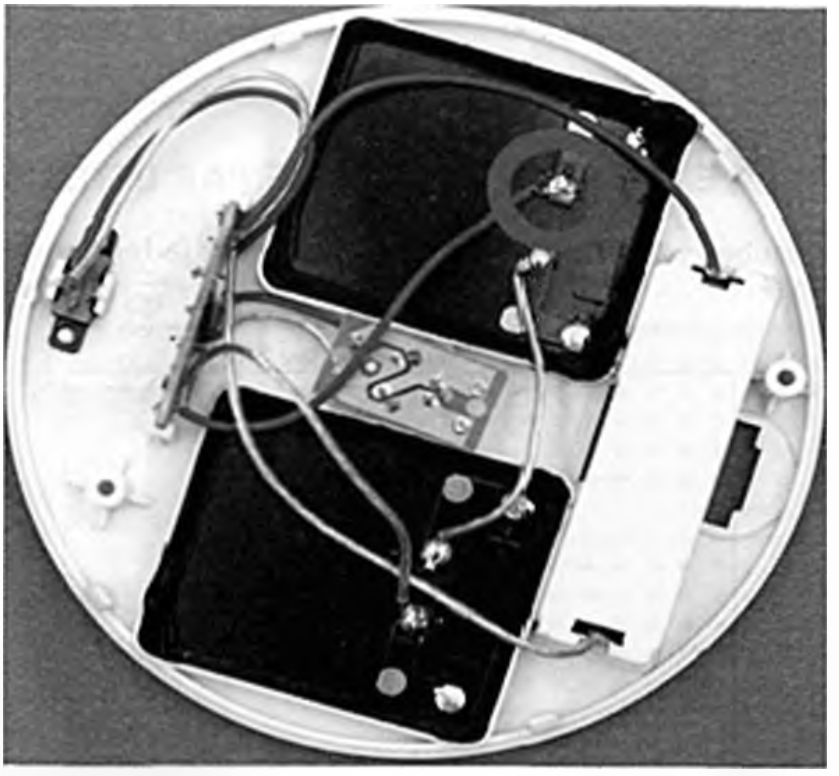


Figure 9.13 Il faut dessouder le câble rouge entouré en bleu

Pour commencer, dessoudez le câble rouge illustré par un cercle bleu à la Figure 9.13.

> Vous n'avez pas besoin de la lampe Solvinden IKEA pour réaliser ce projet. Il suffit de connecter une pile solaire selon le circuit de la Figure 9.17.
>
> Comme on utilise un capteur de courant avec une tension mesurée maximale de 13,6 V, assurez-vous que la pile solaire ne délivre pas une tension supérieure à 13,6 V.
>
> Si vous utilisez une cellule solaire différente, elle est probablement livrée avec des câbles d'alimentation si bien que vous n'avez pas besoin de suivre les instructions de démontage de la lampe Solvinden.

Coupez les câbles de liaison illustrés à la Figure 9.14 et soudez-les au capteur de courant et aux piles solaires (Figure 9.15). Le produit final est illustré à la Figure 9.16.

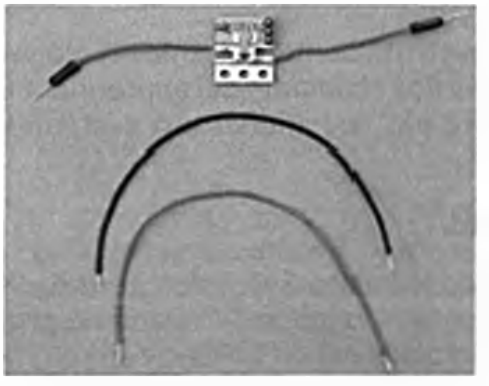


Figure 9.14 Câbles de liaison

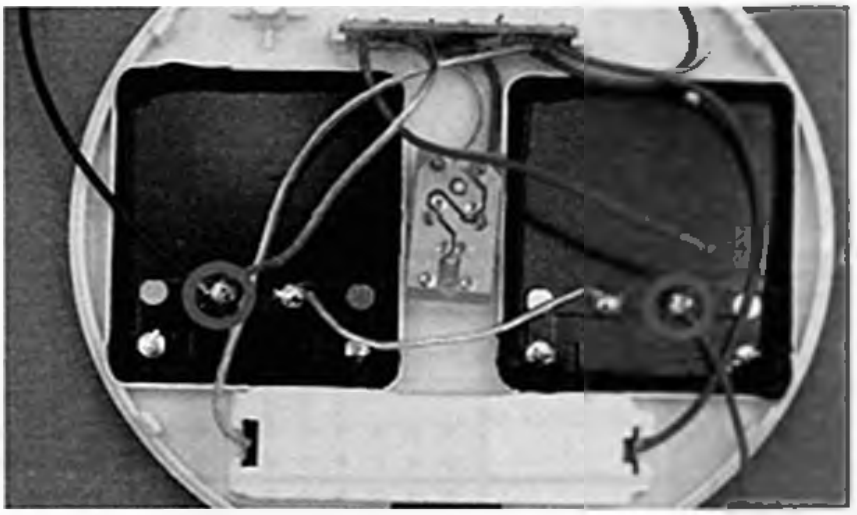


Figure 9.15 Câbles de liaison soudés aux piles solaires

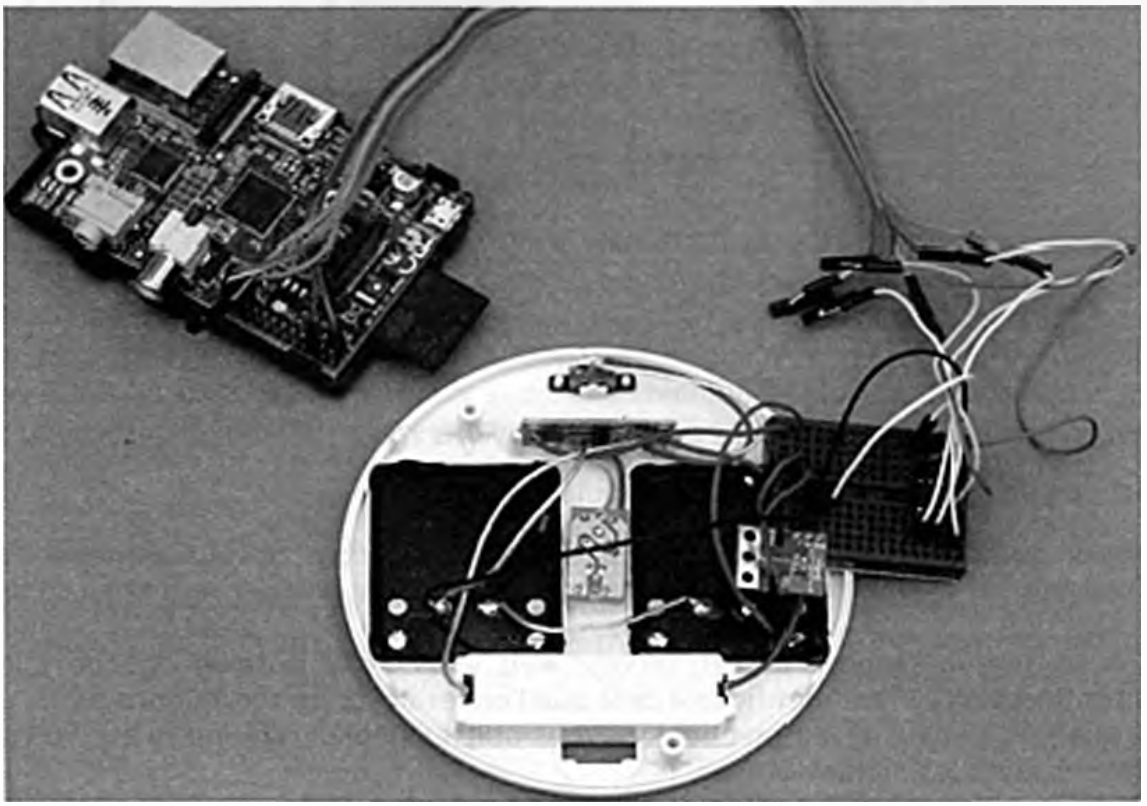


Figure 9.16 Connexions du produit fini

Utilisez le code du capteur de courant AttoPilot que vous avez exécuté précédemment pour tester si votre connexion fonctionne. Utilisez une ampoule pour voir combien de courant vos piles solaires collectent. Nous avons obtenu une tension de 1 à 3 V des nôtres.

Branchez le Raspberry Pi au capteur de courant en suivant le schéma de la Figure 9.17.

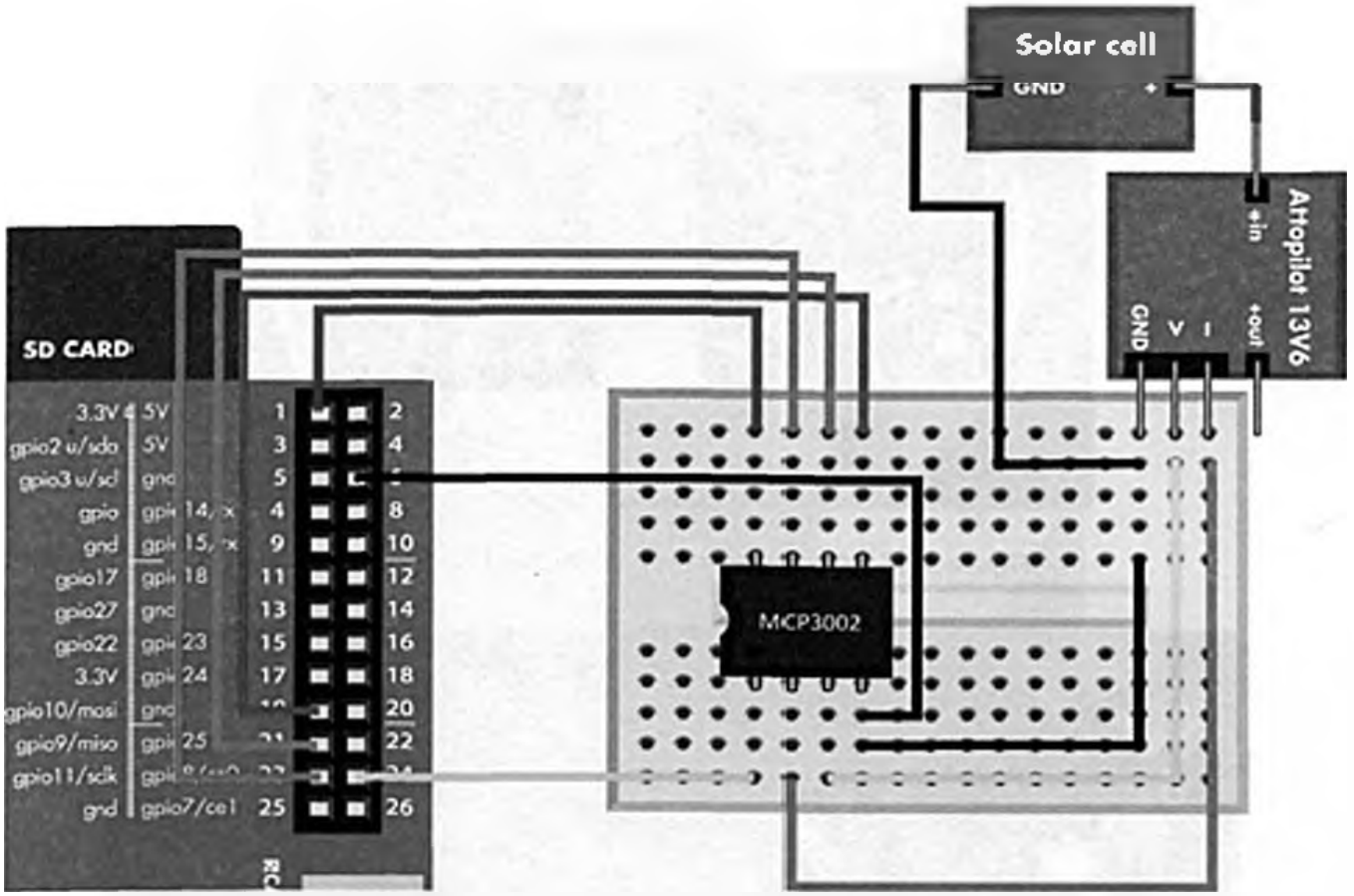


Figure 9.17 Connexion des piles solaires

Transformation du Raspberry Pi en serveur web

Apache est l'un des serveurs web les plus populaires au monde.

> L'installation du serveur web comprend de nombreuses instructions. Si vous connaissez bien Linux, vous pouvez vous contenter de lire les commandes qui réalisent l'installation en peu de temps.

Pour transformer le Raspberry Pi en serveur web, vous devez installer Apache. L'installation d'Apache sur Raspberry Pi est identique à celle que l'on ferait sur un serveur physique, un serveur virtuel ou une machine de développement. Si vous utilisez Debian ou Ubuntu sur votre portable, vous pouvez essayer les mêmes étapes.

Sur le bureau de Raspbian, ouvrez l'interface en ligne de commandes (LXTerminal, icône dans le coin gauche du bureau).

Mettez à jour la liste des paquets disponibles, puis installez Apache :

```
$ sudo apt-get update
$ sudo apt-get -y install apache2
```

Le serveur web est à présent installé et s'exécute. Testez-le avec un navigateur web, comme Midori. L'icône de Midori se situe sur le côté gauche du bureau. Saisissez l'URL :

```
http://localhost/
```

Voyez-vous une page web ? Félicitations ! Vous exécutez à présent un serveur web.

Recherche de votre adresse IP

Pour accéder à votre serveur web à partir d'un autre ordinateur, vous devez trouver votre adresse IP :

```
$ ifconfig
```

Regardez inet addr dans la sortie. Habituellement, elle se trouve sous l'interface eth0 ou wlan0.

> Votre adresse IP n'est pas 127.0.0.1. C'est l'adresse de localhost, et chaque ordinateur fait référence à lui-même sous le nom localhost.

Vous pouvez également essayer de vous rendre à cette adresse. Il suffit d'ouvrir Midori, et de saisir « http:// » suivi de l'adresse IP sous la forme d'une URL. Par exemple, http://10.0.0.1. Bien entendu, vous devez utiliser votre propre adresse.

Cette adresse peut aussi fonctionner sur votre réseau local. Vous pouvez tenter de vous rendre à cette adresse à partir d'autres ordinateurs de votre réseau. Pouvez-vous voir votre serveur web à partir de votre portable ou de votre station de travail ?

Création de la page d'accueil sur le Raspberry Pi

Il est facile de créer et de maintenir une page web si on la crée en tant que page d'accueil utilisateur. Cela se fait en deux étapes : on active les utilisateurs pour qu'ils créent leur page d'accueil et on crée les pages d'accueil.

Nous allons autoriser les utilisateurs à créer leurs pages d'accueil. Comme cela modifie les paramètres système, on a besoin d'utiliser sudo. Les commandes suivantes activent le module du répertoire utilisateur d'Apache, puis redémarrent Apache :

```
$ sudo a2enmod userdir
$ sudo service apache2 restart
```

Il est temps à présent de créer le répertoire de votre page d'accueil (pi). Cette étape ne nécessite pas de privilèges sudo. Le nom du répertoire de votre page d'accueil, public_html, se compose du mot « public », d'un trait de soulignement (« _ »), et du mot « html ». Il doit être écrit tel quel.

```
$ cd /home/pi/
$ mkdir public_html
```

Si vous le souhaitez, vous pouvez créer une page de test :

```
$ echo "botbook">/home/pi/public_html/hello.txt
```

Saisissez l'URL suivante dans le navigateur Midori :

```
http://localhost/~pi/hello.txt
```

Si vous aimez l'aventure, saisissez votre adresse IP à la place de localhost. Vous pouvez même tenter d'aller sur votre page à partir d'un autre ordinateur du même réseau.

Parfait ! Vous avez maintenant transformé votre Raspberry Pi en serveur web et vous avez déjà installé une page d'accueil. Jetons un coup d'œil au programme, et nous le paramétrerons ensuite pour qu'il s'exécute régulièrement.

Contrôle de pile solaire pour Raspberry Pi

Ce code nécessite matplotlib, une grande bibliothèque Python libre pour les graphiques mathématiques. Installez-la avec les commandes suivantes :

```
$ sudo apt-get update
$ sudo apt-get -y install python-matplotlib
```

> Il existe aussi un autre outil très pratique pour la visualisation des données, Plotly (https://plot.ly/). Vous pouvez voir un exemple d'utilisation sur le site Instructables (http://bit.ly/1f0Goqp).

L'exécution de *voltage_record.py* enregistre de nouvelles données et crée un graphique, puis quitte le programme. Aucune boucle n'est nécessaire car nous allons voir comment utiliser cron pour exécuter le programme toutes les cinq minutes.

L'historique des mesures est conservé dans */home/pi/record.csv*.

Le fichier du graphique généré est placé dans */home/pi/public_html/history.png*. Comme vous avez déjà installé Apache, ce fichier est publié à l'URL http://localhost/~pi/history.png. Pour visiter cette page à partir d'un autre ordinateur, vous devez remplacer localhost par l'adresse IP de votre Raspberry Pi.

Exemple 9.9. voltage_record.py

```
# voltage_record.py - enregistre la tension d'une pile solaire et affiche
l'historique dans un graphique
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import os
import matplotlib       # 1
matplotlib.use("AGG")   # 2
import matplotlib.pyplot as plt
import numpy
from datetime import datetime
from datetime import date
from datetime import timedelta
import attopilot_voltage        # 3
import shelve
historyFile = "/home/pi/record" # 4
plotFile = "/home/pi/public_html/history.png"   # 5
def measureNewDatapoint():
        return attopilot_voltage.readVoltage()  # 6
def main():
        history = shelve.open(historyFile)      # 7
        if not history.has_key("x"):    # 8
                history["x"] = []       # 9
                history["y"] = []
        history["x"] += [datetime.now()]        # 10
        history["y"] += [measureNewDatapoint()] # 11
```

```
        now = datetime.now()      # 12
        sampleCount = 24 * 60 / 5
        history['x'] = history["x"][-sampleCount:]
        history['y'] = history["y"][-sampleCount:]
        plot = plt.figure()       # 13
        plt.title('Tension de la pile solaire sur une période de 24h')
        plt.ylabel('Voltage V')
        plt.xlabel('Heure')
        plt.setp(plt.gca().get_xticklabels(), rotation=35)
        plt.plot(history["x"], history["y"], color='#4884ff')    # 14
        plt.savefig(plotFile)     # 15
        history.close()  # 16
if __name__ == '__main__':
        main()
```

1 La bibliothèque doit être installée en suivant les directives énoncées plus haut.
2 AGG (Anti-Grain Geometry) est un bon module matplotlib pour enregistrer les fichiers au format PNG.
3 L'exemple que vous avez créé plus haut doit se trouver dans le répertoire *attopilot_voltage/* qui est stocké dans le même répertoire que ce programme (*voltage_record.py*). Voir précédemment la section *Expérience : tension et courant*.
4 C'est le fichier de stockage où les données numériques sont conservées. La fonction Python shelve (que nous allons voir dans un moment) facilite grandement le stockage des valeurs.
5 Graphique (image au format PNG) à créer. Pour la publier en tant que page web, on la met dans un dossier qui est géré par le serveur web Apache.
6 C'est la véritable commande qui effectue la mesure. Pour mesurer n'importe quoi d'autre, il suffit de changer cette commande.
7 Ouvre le fichier historique. Le fichier est créé automatiquement si nécessaire.
8 Si nos variables ne sont pas déjà dans le stockage...
9 ...on les crée. Le fichier historique est un dictionnaire. Les clés x et y stockent une liste chacune.
10 Ajoute l'horodatage à x. Notez que nous n'avons pas à nous soucier de la mise en forme de la date ou de son analyse.
11 Ajoute la valeur mesurée à y. Les dictionnaires x et y sont complètement différents, mais les valeurs ayant le même indice sont liées. Par exemple, les données stockées dans y[12] correspondent à l'heure stockée dans x[12].
12 On abandonne les enregistrements au bout de 24 heures. Une alternative plus simple consisterait à ne stocker qu'une centaine de données et à les abandonner après.
13 Crée un nouveau graphique matplotlib à tracer.
14 Trace le graphique.
15 Enregistre le graphique sous la forme d'une image PNG.
16 Ferme le stockage pour écrire les valeurs sur le disque.

Tâches planifiées avec cron

Cron est notre outil favori pour accomplir des tâches planifiées. Même si votre Raspberry Pi s'éteint, les tâches cron reprennent après le redémarrage.

Assurez-vous d'exécuter le programme Python en ligne de commande, avec un chemin complet. Utilisez le chemin de l'emplacement où vous avez téléchargé ou enregistré *voltage_record.py* (voir l'Exemple 9.9).

```
$ python /home/pi/voltage_record/voltage_record.py
```

> Vous ne vous souvenez plus où se trouve *voltage_record.py* ? Exécutez cd /home/pi; find -iname voltage_record.py.

Si tout s'est bien passé, il faut à présent ajouter une tâche planifiée à cron. Modifiez votre fichier utilisateur cron :

```
$ crontab -e
```

Ajoutez les lignes suivantes à la fin du fichier :

```
*/5 * * * * /home/pi/voltage_record/voltage_record.py
*/1 * * * * touch /tmp/botbook-cron
```

Enregistrez le fichier. Si vous l'avez modifié avec nano, appuyez sur Ctrl+X, saisissez y, puis appuyez sur la touche Entrée.

La première ligne dit à cron d'exécuter le programme quand les minutes sont divisibles par 5 (:00, :05...), en ignorant l'heure, le jour, le mois et le jour de la semaine.

La deuxième ligne crée simplement un fichier vide */tmp/botbook-cron* toutes les minutes. Cela permet de vérifier le fonctionnement de cron, et vous pouvez supprimer cette ligne ultérieurement si vous le souhaitez. Attendez une minute, puis vérifiez la présence du fichier :

```
$ ls /tmp/botbook*
/tmp/botbook-cron
```

Est-ce que ls affiche votre fichier ? Parfait ! Vous avez ajouté avec succès une tâche planifiée à cron.

Quand vous voulez que votre Raspberry Pi fasse automatiquement quelque chose, utilisez cron. Même si l'alimentation est coupée, les tâches planifiées sont exécutées à nouveau quand le Raspberry Pi redémarre.

> Si la tâche cron écrit trop d'informations dans vos journaux, vous pouvez ajouter >/dev/null à votre commande dans crontab. Cela supprime tout ce que la commande affiche sur la sortie standard.

POUR ALLER PLUS LOIN

Vous avez joué avec l'électricité dans ce chapitre. Vos gadgets peuvent mesurer des tensions et des courants qui atteignent même des niveaux qui endommageraient la carte de votre microcontrôleur si elle était connectée directement. Les champs magnétiques peuvent être détectés simplement, car un capteur à effet Hall détecte un aimant et vous pouvez même utiliser une boussole 3D.

Le programme de contrôle par Internet d'une pile solaire peut être adapté à n'importe quel projet de capteur. Il suffit de modifier la fonction de mesure et vous pouvez stocker les données de n'importe quel capteur. Dans votre projet, vous publiez le graphique sur le Web et vous pouvez tenter d'ajouter une protection par mot de passe, ou même d'utiliser ssh afin de visualiser le graphique pour les projets qui nécessitent une plus grande confidentialité.

Dans le prochain chapitre, nous allons passer de l'électromagnétisme aux ondes sonores. Il est temps de fournir des oreilles à nos gadgets !

10 Entendre les sons

Les ondes sonores sont une compression et une décompression de l'air. Ce phénomène peut être détecté et analysé, par exemple à l'aide d'un Raspberry Pi. Dans un des projets de ce chapitre nous allons utiliser une transformée de Fourier rapide pour décomposer le son en fréquences. Ce calcul peut extraire les fréquences de n'importe quelle onde sonore. Avec un Arduino nous allons utiliser un microphone pour mesure le volume du son.

EXPÉRIENCE : ENTENDRE DES VOIX ET PERCEVOIR UN NIVEAU SONORE

Un microphone peut détecter un niveau sonore. Cette première expérience se contente de lire et d'afficher les valeurs perçues par un microphone.

Dans de nombreux projets, il est souhaitable de manipuler les valeurs lues par un microphone. On peut définir un seuil, appliquer une moyenne mobile, ou bien détecter la valeur minimale et la valeur maximale. D'autres exemples dans ce chapitre vous apprendront à définir un seuil sonore.

Nous avons utilisé un composant de chez Sparkfun illustré à la Figure 10.1 : une platine d'évaluation embarquant un microphone à électret (référence BOB-09964).

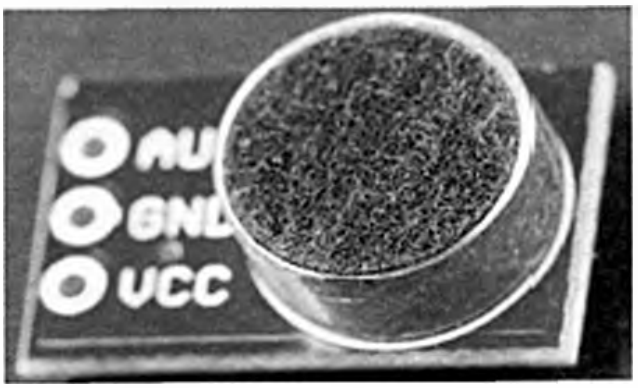


Figure 10.1 Microphone sur une platine d'évaluation

Microphone pour Arduino

La Figure 10.2 illustre le circuit Arduino du microphone. Câblez en suivant le schéma et exécutez le code de l'Exemple 10.1.

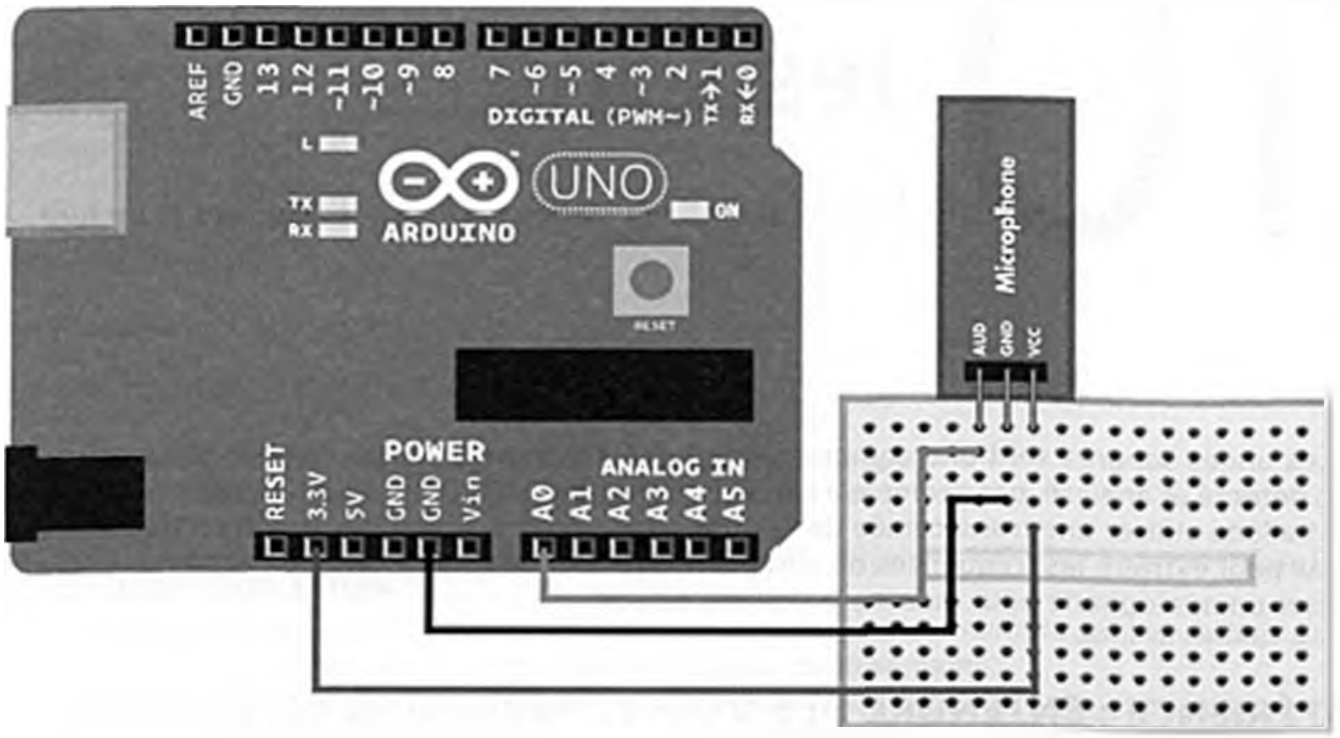


Figure 10.2 Circuit Arduino pour microphone

Exemple 10.1. microphone.ino

```
// microphone.ino - affiche le niveau du volume sonore sur le port série.
// Affiche "Son" si un grand bruit est détecté.
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int audioPin = A0;
const int sensitivity = 850;
void setup() {
  Serial.begin(115200);
}
void loop() {
  int soundWave = analogRead(audioPin); // 1
  Serial.println(soundWave);
  if (soundWave>sensitivity) {  // 2
    Serial.println("Son !");
    delay(500);
  }
  delay(10);
}
```

1 Vous pouvez lire le microphone de la platine d'évaluation comme tout capteur de résistance analogique. analogRead() retourne des valeurs brutes comprises entre 0 et 1023, et la valeur est proportionnelle au volume sonore. La connexion et le programme sont similaires à la connexion d'un potentiomètre.

2 Action en cas de bruit suffisamment fort.

Microphone pour Raspberry Pi

La Figure 10.3 illustre le schéma de câblage du Raspberry Pi. Branchez en suivant le schéma puis exécutez le code de l'Exemple 10.2.

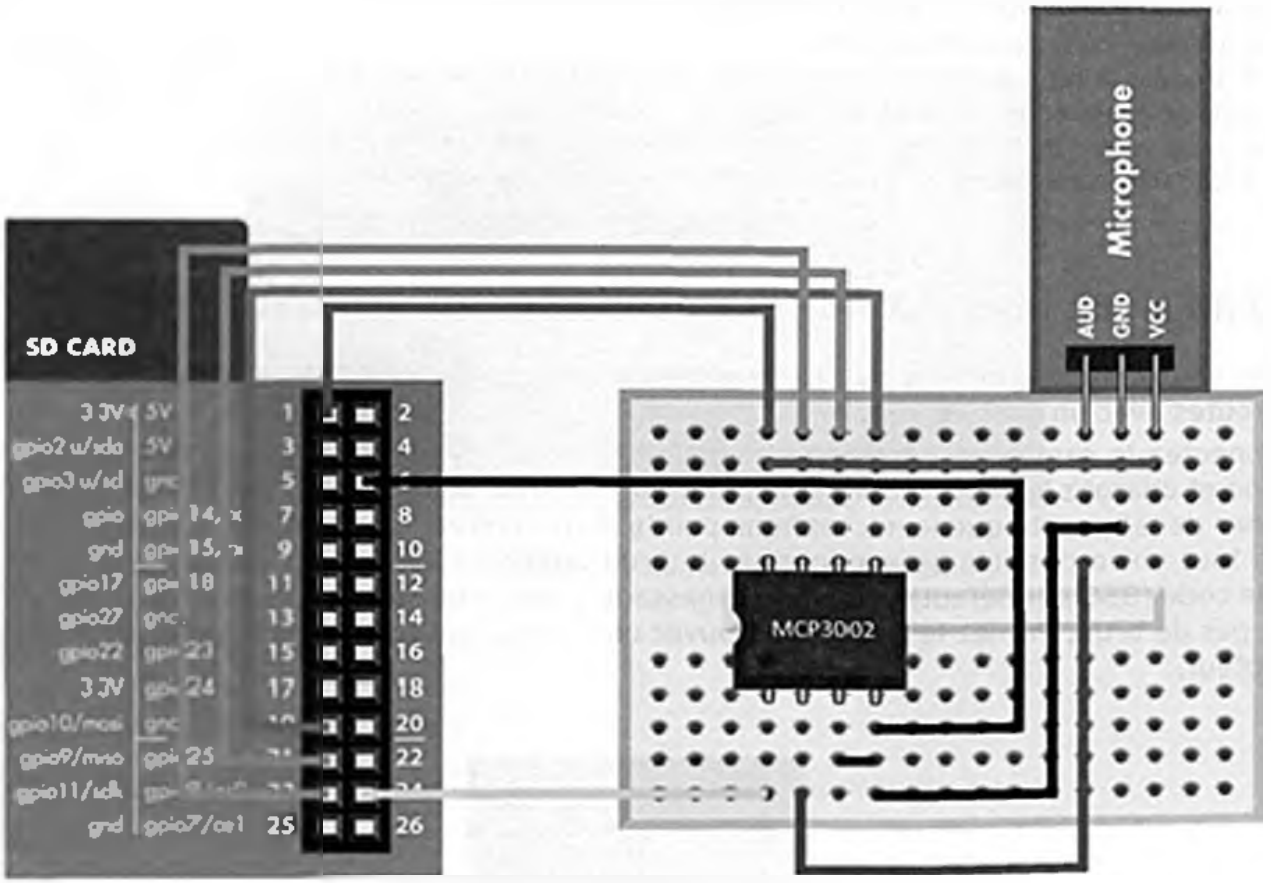


Figure 10.3 Circuit Raspberry Pi pour microphone

Exemple 10.2. microphone.py

```
# microphone.py - lit le son de la sortie analogique et l'affiche
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp    # 1
def readSound(samples):
        buff = []        # 2
        for i in range(samples):        # 3
                buff.append(mcp.readAnalog())    # 4
        return buff
def main():
        while True:
                sound = readSound(1024) # 5
                print(sound)
                time.sleep(1)    # 6
if __name__ == "__main__":
        main()
```

1 La bibliothèque (botbook_mcp3002.py) doit se trouver dans le même répertoire que ce programme.
Vous devez aussi installer la bibliothèque spidev, qui est importée par botbook_mcp3002.
Lisez les commentaires au début du fichier botbook_mcp3002/botbook_mcp3002.py ou bien la section *Installation de SpiDev* du chapitre 3.
2 Déclare une nouvelle liste vide.
3 Répète le bloc de la ligne suivante de telle sorte que l'on assigne à i chacune des valeurs. Dans ce programme, cela est équivalent à : for i in [0, 1, 2 ... 1023]:
4 Lit la valeur du microphone. Ajoute un nouvel élément à la liste (`buff`).
5 Lit 1024 échantillons et récupère une liste des valeurs retournées.

EXPÉRIENCE : POUVEZ-VOUS ENTENDRE UNE ÉPINGLE TOMBER ?

Pouvez-vous entendre le bruit d'une épingle qui tombe ? Réglons cette question une bonne fois pour toutes avec un capteur sonore.

Connectez le capteur à l'Arduino comme vous l'avez fait dans la section *Microphone pour Arduino* et chargez le code. Placez le capteur sur un plan solide comme une table en bois ou un plancher de telle sorte que le microphone pointe dans la direction où vous allez lâcher l'épingle. Faites tout votre possible pour minimiser le bruit ambiant. Changez la valeur de la sensibilité dans le code (850 par défaut) pour que le message « Son » ne soit pas déclenché quand vous ne faites pas de bruit. Prenez le temps de trouver une valeur qui soit à la limite du déclenchement du capteur.

Figure 10.4 Épingle tombant sur un plancher

Ouvrez le Moniteur série dans l'IDE Arduino et faites tomber une épingle.

Avez-vous obtenu le message « Son » sur le Moniteur série ? Si tel est le cas, vous avez réellement pu entendre tomber une épingle. Si vous n'avez pas de réaction du capteur, assurez-vous qu'il n'y ait pas de bruit dans la pièce et faites tomber l'épingle plus près du capteur, tout en vérifiant que vous avez correctement ajusté la sensibilité du capteur.

La sensibilité aux sons légers dépend du niveau de bruit ambiant. Mettez de la musique un peu forte et réajustez la sensibilité avant de faire tomber l'épingle. Est-ce que le capteur peut toujours entendre l'épingle tomber ?

PROJET : VISUALISEZ LE SON SUR LE PORT HDMI

N'avez-vous jamais rêvé d'avoir un égaliseur graphique sur un écran de 50 pouces ? Dans ce projet, vous allez analyser un son avec un Raspberry Pi et afficher le résultat sur votre télévision. Les fréquences sonores sont affichées sous la forme d'un graphique animé en couleur (Figure 10.5).

Figure 10.5 Graphique animé sur un grand écran

Ce que vous allez apprendre

Dans le projet *Visualisez le son sur le port HDMI*, vous allez apprendre à :

- Analyser le son numériquement.
- Faire des calculs très rapides en Python, en utilisant SciPy.
- Extraire des fréquences avec une transformée de Fourier rapide.

Vous allez aussi rafraîchir vos compétences pyGame et tracer des graphiques en HD sur la sortie HDMI, en vous servant de votre téléviseur ou d'un vidéoprojecteur (voir le *Projet Pong* du chapitre 6).

Activation du port série du Raspberry Pi

Pour utiliser le port série du Raspberry Pi, vous devez d'abord l'activer. Dans le cas contraire, il est utilisé par un shell auquel vous pouvez vous connecter par un câble série :

```
$ sudoedit /etc/inittab
```

Mettez en commentaire la dernière ligne qui concerne le port série. On place un commentaire sur une ligne en plaçant un signe dièse à son début ; la ligne est alors ignorée.

```
# T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Redémarrez le Raspberry Pi.

Visualiseur pour Raspberry Pi

Il faut d'abord installer quelques programmes prérequis. Sur le Raspberry Pi, ouvrez le terminal et exécutez ces commandes :

```
$ sudo apt-get update
$ sudo apt-get -y install python-pygame python-numpy
```

La Figure 10.6 illustre le diagramme du circuit pour Raspberry Pi. Branchez en suivant le schéma et exécutez le code de l'Exemple 10.3.

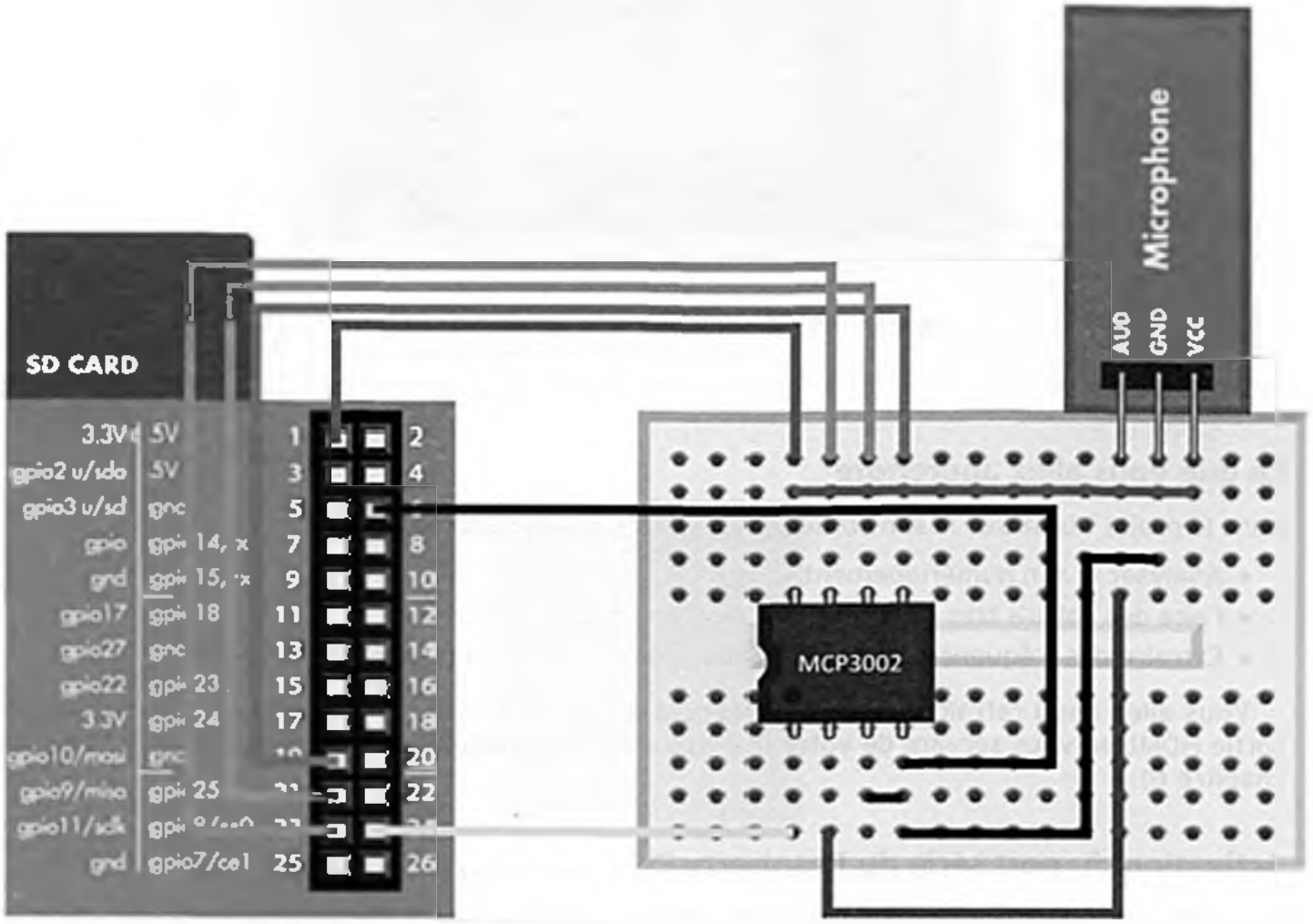


Figure 10.6 Circuit de microphone Raspberry Pi pour un égaliseur

Exemple 10.3. equalizer.py

```
# equalizer.py - affiche un égaliseur basé sur l'entrée du microphone
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import pygame  # sudo apt-get -y install python-pygame
import math
import numpy  # sudo apt-get -y install python-numpy
import microphone    # 1
```

```
from pygame.locals import *
pygame.init()
width = 800
height = 640
size = width, height
background = 0, 0, 0
screen = pygame.display.set_mode(size, pygame.FULLSCREEN)
fullBar = pygame.image.load("equalizer-full-small4.jpg")    # 2
emptyBar = pygame.image.load("equalizer-empty-small4.jpg")
clock = pygame.time.Clock()
pygame.mouse.set_visible(False)
mainloop = True
barHeight = 36
barWidth = 80
barGraphHeight = 327
barPos = [55, 130]
sampleLength = 16
def fftCalculations(data):    # 3
    data2 = numpy.array(data) / 4    # 4
    fourier = numpy.fft.rfft(data2)    # 5
    ffty = numpy.abs(fourier)    # 6
    ffty = ffty / 256.0        # 7
    return ffty
while mainloop:
    buff = microphone.readSound(sampleLength)    # 8
    barData = fftCalculations(buff)    # 9
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            mainloop = False
        if (event.type == KEYUP) or (event.type == KEYDOWN):
            if event.key == K_ESCAPE:
                mainloop = False
    screen.fill(background)
    # Trace les données en colonnes
    for i in range(8):    # 10
        bars = barData[i+1]    # 11
        bars = int(math.floor(bars*10))    # 12
        if bars > 10:
            bars = 10
        bars -= 1
        screen.blit(emptyBar, (barPos[0]+i*(barWidth+10), barPos[1]),
                              (0, 0, barWidth, barHeight*(10-bars)))
        if bars >= 0:
            barStartPos = (barPos[0] + i * (barWidth + 10),
                           barPos[1] + barGraphHeight - barHeight * bars+6)
            barSourceBlit = (0, barGraphHeight - barHeight * bars+6,
                             barWidth, barHeight*bars)
            screen.blit(fullBar, barStartPos, barSourceBlit)    # 13
    pygame.display.update()
```

1 Le programme microphone.py de la section *Microphone pour Raspberry Pi* doit se trouver dans le même répertoire que ce programme (equalizer.py).
2 PyGame peut charger des images normales au format JPG. Vous pouvez les dessiner dans n'importe quel programme, comme Inkscape, Gimp, Photoshop, ou Illustrator.
3 Quand cette fonction est appelée à partir du programme principal, data contient 16 échantillons, chacun dans la plage de 0 à 1024.
4 Divise les échantillons pour les mettre dans la plage de 0 à 256.
5 Accomplit une transformée de Fourier sur l'échantillon sonore enregistré. Cela donne les plages des fréquences de l'échantillon. La fonction rfft() retourne un tableau qui a 16/2+1 (9) cellules, chacune représentant une fréquence.
6 On se débarrasse de la partie imaginaire des nombres, de telle sorte qu'on puisse les tracer plus tard. Par exemple, abs(1+1j) est approximativement 1,4.
7 Convertit les valeurs grâce à la transformée de Fourier en pourcentages (dans la plage de 0,0 à 1,0).
8 Lit 16 échantillons à partir du microphone, grâce à la bibliothèque microphone de botbook.com.
9 Enregistre la transformée de Fourier de l'échantillon sonore dans barData.
10 Pour chacune des huit barres verticales (0 à 7)...
11 ...on récupère le pourcentage de la fréquence. En raison du i+1 ici, le premier numéro de cellule (0) est ignoré. La première cellule ignorée contient le composant de courant continu, la valeur moyenne entre les pics positif et négatif de l'onde de courant alternatif.
12 Compte le nombre de barres nécessaires (par exemple, 1.0 (100 %) donne le nombre maximal de barres, à savoir neuf).
13 Trace les barres complètes à l'écran.

Transformée de Fourier rapide

Une transformée de Fourier rapide (que l'on abrège en anglais par FFT pour *Fast Fourier Transform*) extrait des fréquences individuelles à partir d'une onde. On l'utilise pour créer des diagrammes de fréquences dans les égaliseurs graphiques, les analyseurs de spectres et les oscilloscopes.

Imaginez que vous jouez deux notes, une basse (5 Hz) et une haute (40 Hz).

> Dans la vie réelle, on ne perçoit les sons que dans la plage allant de 20 Hz à 20 000 Hz, mais on a choisi des valeurs plus petites pour cet exemple.

À la Figure 10.7, l'axe horizontal du temps augmente vers la droite. L'axe vertical est l'amplitude, qui représente la compression et la décompression de l'air.

Quand ces deux notes sont jouées en même temps, elles forment une seule onde.

Cette onde combinée est semblable à une petite ondulation au sommet d'une grande vague. La note basse, à 5 Hz, forme la grande onde. La petite ondulation est la fréquence la plus élevée à 40 Hz. L'onde combinée n'est que la somme des deux ondes.

Quand on enregistre un son, on obtient ce type d'onde combinée. Avec un tel type d'onde, comment retrouve-t-on les deux ondes sinusoïdales originales ? C'est à ce moment que l'on a besoin de Fourier !

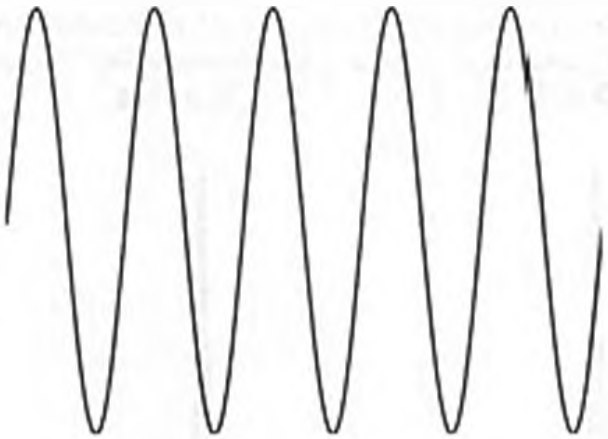


Figure 10.7 Note basse, onde sinusoïdale à 5 Hz

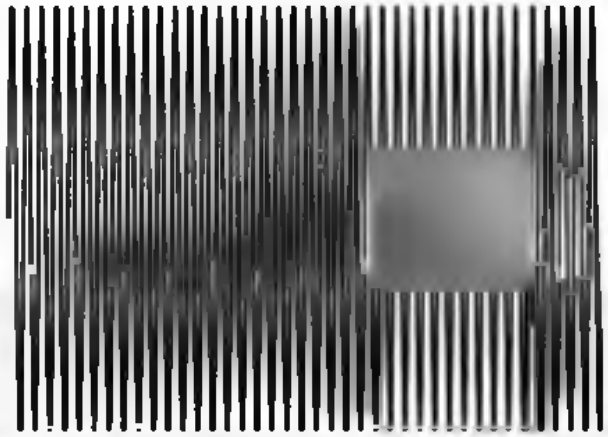


Figure 10.8 Note haute, onde sinusoïdale à 40 Hz

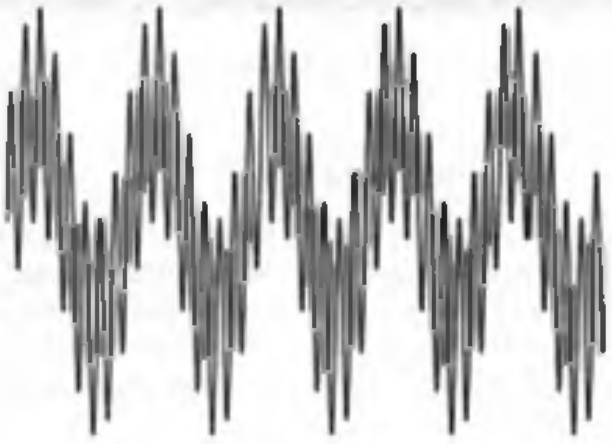


Figure 10.9 Les deux notes combinées

On accomplit alors une transformée de Fourier rapide de l'onde combinée. Comme vous l'avez vu dans le code de ce projet, il suffit d'appeler une fonction FFT en lui passant en paramètre les données de l'onde (Figure 10.10).

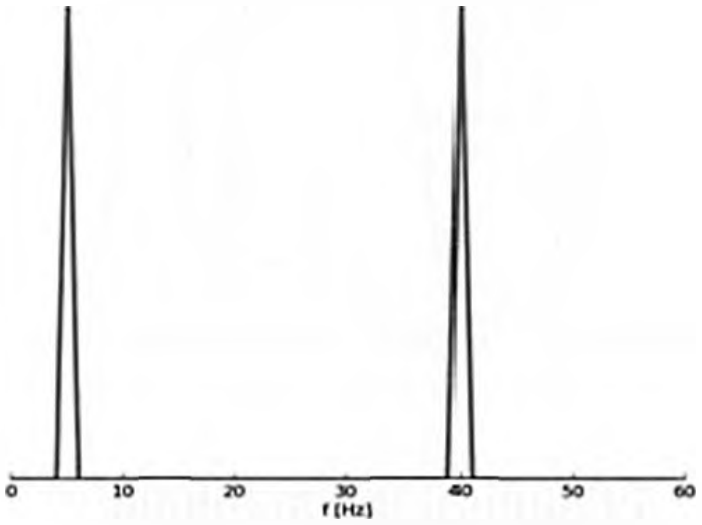


Figure 10.10 La transformée de Fourier révèle les fréquences des notes originales

La FFT fournit les fréquences des ondes sinusoïdales originales, 5 Hz et 40 Hz. Vous avez à présent divisé une onde complexe en fréquences. La Figure 10.11 illustre l'affichage d'un égaliseur.

Figure 10.11 Égaliseur

POUR ALLER PLUS LOIN

Dans ce chapitre, vous avez joué avec le son. Avec Arduino, vous avez détecté le niveau sonore et repéré un volume dépassant un certain seuil. Avec Raspberry Pi, vous avez enregistré un son et l'avez analysé en temps réel. Quand vous avez créé votre projet, vous avez amélioré vos compétences pyGame et utilisé une transformée de Fourier rapide pour diviser une onde en composants.

Dans le prochain chapitre, nous allons regarder le ciel et mesurer la température et l'humidité d'une pièce pour finir par prédire la météo locale.

11 Construire une station météo

La météo reste l'une des rares choses que les hommes ne peuvent pas contrôler. Pour l'instant... Pour prévoir le temps on a besoin de mesurer la pression atmosphérique, la température et l'humidité.

EXPÉRIENCE : EST-CE QU'IL FAIT CHAUD ICI ?

Un capteur LM35 indique la température grâce à une variation de la tension. Comme il s'agit d'un capteur de résistance analogique, il est facile à utiliser avec Arduino.

L'emploi avec un Raspberry Pi nécessite un convertisseur analogique-numérique.

Le LM35 a trois fils (Figure 11.1), comme un potentiomètre. Le fil de sortie de la tension U est converti en température T par une simple formule :

```
T = 100•U
```

Des exemples de valeurs sont listés dans le Tableau 11.1. À partir du tableau, vous pouvez voir que le LM35 peut mesurer des températures comprises entre 2°C et 150°C.

La tension de sortie varie entre approximativement 0 V et 1,5 V.

> Avez-vous besoin de mesurer des températures négatives ? Utilisez le LM36 ou bien consultez la notice technique du LM35 pour d'autres options de connexion.

Température en °C	Tension en volts	Commentaire
2°C	0,02 V	Température minimale mesurée
20°C	0,2 V	Température de la pièce
100°C	1,0 V	Eau bouillante
150°C	1,5 V	Température maximale mesurée

Tableau 11.1 Le LM35 indique la température en mesurant la tension

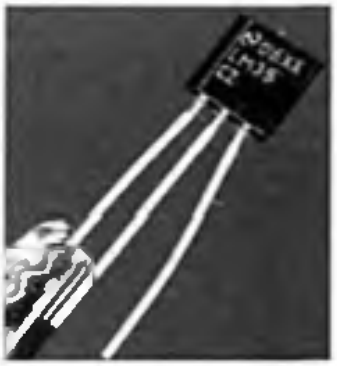


Figure 11.1 LM35

LM35 pour Arduino

La Figure 11.2 illustre le schéma de câblage pour Arduino. Branchez en suivant le schéma puis exécutez le code de l'Exemple 11.1.

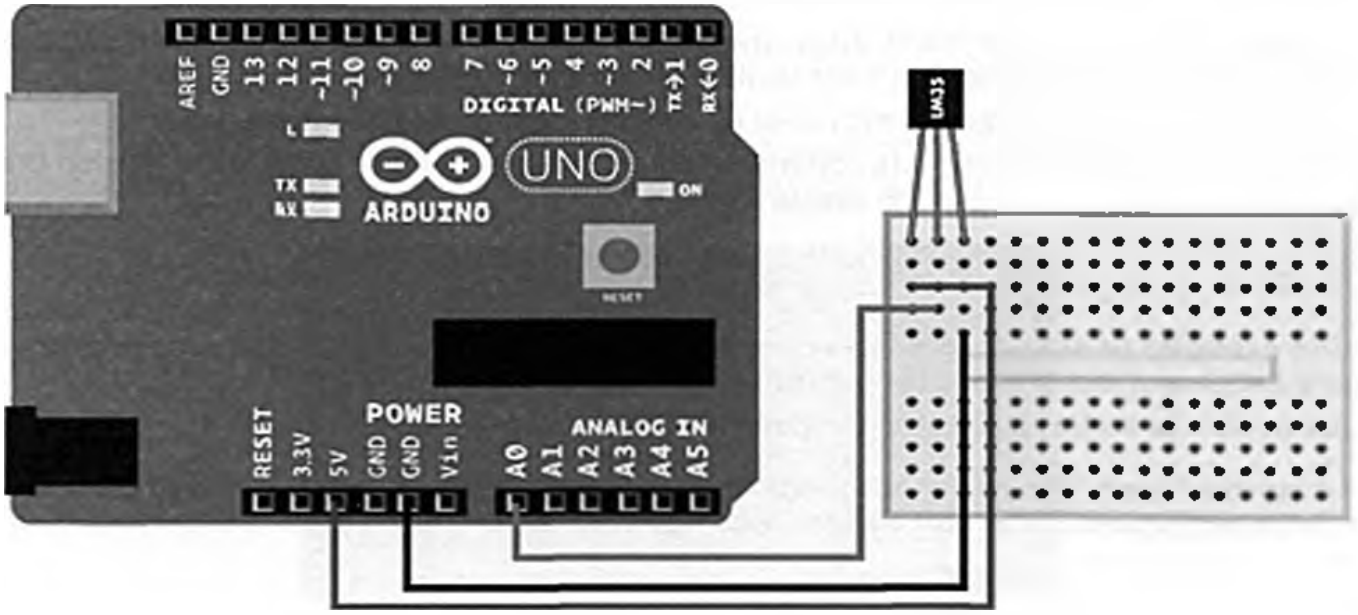


Figure 11.2 Connexions Arduino pour le LM35

Exemple 11.1. temperature_lm35.ino

```
// temperature_lm35.ino - mesure et affiche la température (Celsius) avec le
LM35
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int lmPin = A0;
void setup()
{
  Serial.begin(115200);
  pinMode(lmPin, INPUT);
}
float tempC()
```

```
{
  float raw = analogRead(lmPin);  // 1
  float percent = raw/1023.0; // 2
  float volts = percent*5.0; // 3
  return 100.0*volts;  // 4
}
void loop()
{
  Serial.println(tempC());
  delay(200); // ms
}
```

1 Le LM35 n'est qu'un capteur de résistance analogique, et vous pouvez donc mesurer sa tension avec analogRead(). Il retourne une valeur brute comprise entre 0 et 1023.
2 Convertit la valeur brute (de 0 à 1023) en un pourcentage de la tension HIGH (5 V). La variable percent sera un nombre en virgule flottante compris entre 0,0 et 1,0.
3 Convertit le pourcentage en volts (entre 0 et 5 V).
4 Retourne la température en degrés Celsius. La formule utilisée provient de la notice technique du LM35.

LM35 pour Raspberry Pi

La Figure 11.3 illustre le diagramme de connexion pour Raspberry Pi. Câblez en suivant le schéma puis exécutez le code de l'Exemple 11.2.

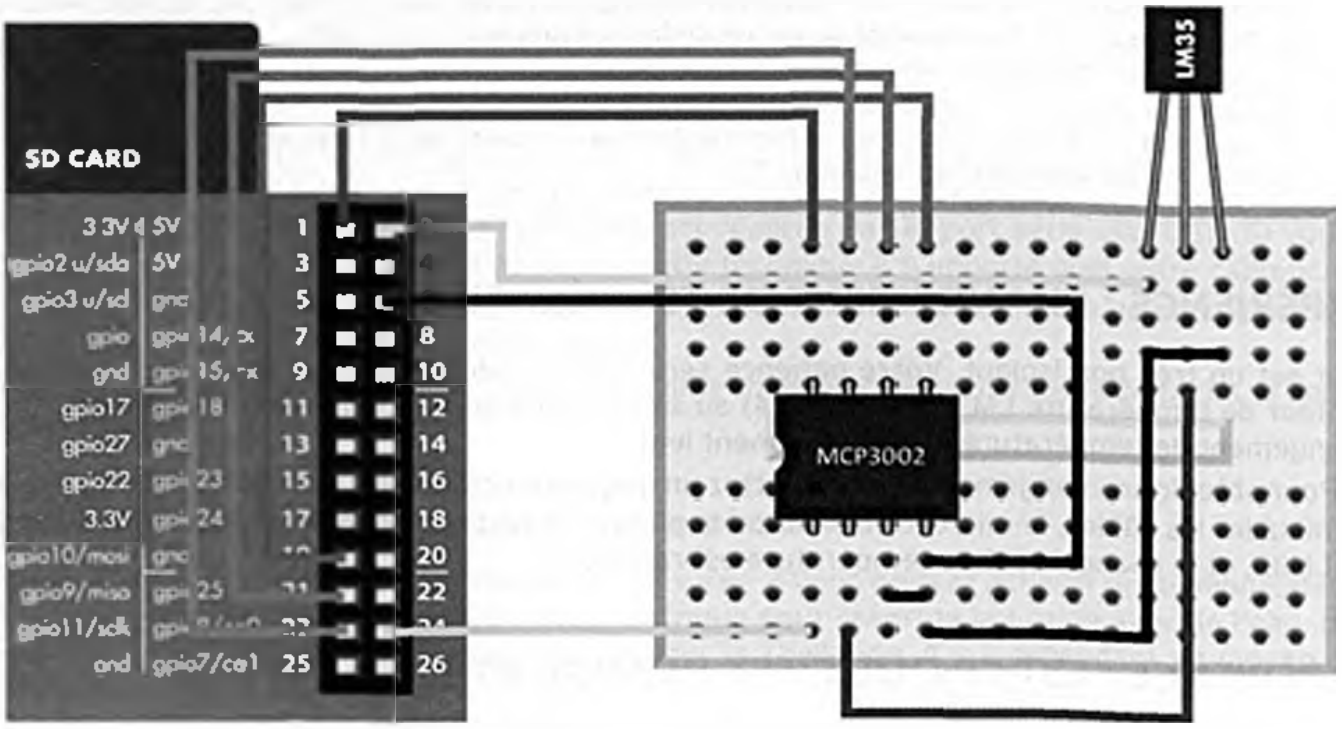


Figure 11.3 Connexions Raspberry Pi pour le LM35

Comme la sortie maximale de la tension du LM35 est de 1,5 V à 150°C, il n'y a aucun danger de dépassement du niveau d'entrée maximal de la tension du Raspberry Pi qui est 3,3 V.

Exemple 11.2. temperature_lm35.py

```
# temperature_lm35.py - affiche la température
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp    # 1
# La tension de la broche de données du LM35 reste en dessous de 2 V
# Il n'y a donc pas besoin de conversion de niveau
def main():
        while True:
                rawData = mcp.readAnalog()       # 2
                percent = rawData / 1023.0       # 3
                volts = percent * 3.3    # 4
                temperature = 100.0 * volts      # 5
                print("Température actuelle : %.1f C " % temperature)
                time.sleep(0.5)
if __name__ == "__main__":
        main()
```

1 La bibliothèque *botbook_mcp3002.py* doit se trouver dans le même répertoire que ce programme (*temperature_lm35.py*). Vous devez aussi installer la bibliothèque *spidev*, qui est importée par *botbook_mcp3002*. Lisez les commentaires au début du fichier *botbook_mcp3002/botbook_mcp3002.py* ou la section *Installation de SpiDev* du chapitre 3.
2 Lit la tension en utilisant un convertisseur analogique-numérique MCP3002 et la bibliothèque *botbook_mcp3002*. La valeur brute est un entier compris entre 0 et 1023.
3 Convertit la valeur brute en un pourcentage de 5 V.
4 Convertit percent en une tension (entre 0 et 3,3 V).
5 Convertit en degrés Celsius en utilisant la formule trouvée dans la notice technique du composant. Voir les exemples du Tableau 11.1.

EXPÉRIENCE : CHANGEMENT DE TEMPÉRATURE

L'air est un très bon isolant. Votre patience sera mise à rude épreuve si vous emportez votre capteur de température LM35 (Figure 11.4) au sauna, dans un frigidaire ou sur le balcon, car le changement de température est extrêmement lent.

Pour obtenir un changement rapide, mettez un glaçon directement sur le LM35, tout en évitant de mouiller les câbles, le microcontrôleur ou la platine de test. La chaleur est à présent évacuée du LM35, et le changement de température est rapide.

EXPÉRIENCE : EST-CE QUE C'EST HUMIDE ICI ?

Le capteur DHT11 mesure l'humidité et la température (Figure 11.5).

Figure 11.4 Mesure de températures froides

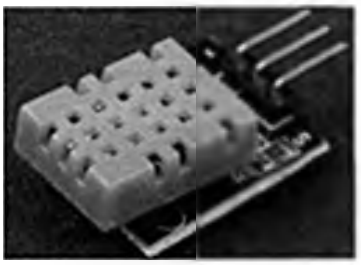


Figure 11.5 Capteur d'humidité DHT11

Le protocole utilisé par le DHT11 est étrange. Il envoie très rapidement des bits sous la forme d'impulsions très courtes. Arduino n'a pas de système d'exploitation et il est donc plus rapide que le Raspberry Pi et peut alors facilement lire ces impulsions. Mais l'Arduino a cependant besoin d'être programmé spécialement car même la fonction intégrée pulseIn() n'est pas assez rapide.

Le Raspberry Pi est moins rapide et a des difficultés à lire les impulsions de manière fiable. C'est la raison pour laquelle nous allons connecter l'Arduino au Raspberry Pi et utiliser l'Arduino pour lire les données du DHT11. La connexion est réalisée par le port série via l'USB. Si vous le souhaitez, vous pouvez facilement appliquer cette technique à n'importe quel autre capteur.

La broche de données est normalement à HIGH. Pour démarrer la lecture, le microcontrôleur envoie une impulsion LOW de 18 millisecondes.

Le DHT11 envoie des paquets de 5 octets. Comme chaque octet est composé de 8 bits, le paquet compte 40 bits.

Quel est le degré d'humidité de votre respiration ?

Comment pouvons-nous modifier l'humidité mesurée ? Vous pouvez acheter un humidificateur à ultrasons qui utilise un transducteur piézoélectrique pour créer de la brume avec de l'eau, mais il est également possible de réserver un vol pour la Thaïlande. Il y a cependant un moyen plus facile. Vous avez un humidificateur intégré dans vos poumons. Approchez le capteur près de votre bouche et respirez (Figure 11.6). Vérifiez la modification des valeurs sur le Moniteur série Arduino.

Figure 11.6 Respiration face à un capteur d'humidité

DHT11 pour Arduino

La Figure 11.7 illustre les connexions pour Arduino. Câblez en suivant le schéma puis exécutez le code de l'Exemple 11.3.

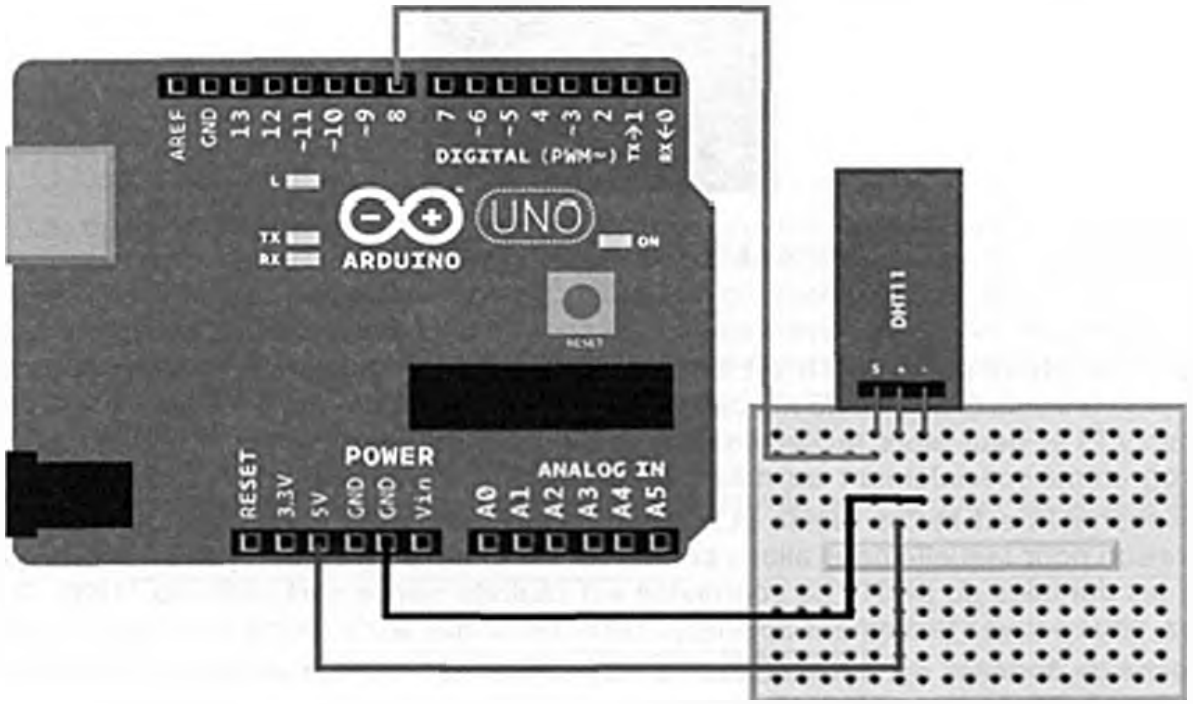


Figure 11.7 Capteur d'humidité DHT11 connecté à Arduino

Exemple 11.3. dht11.ino

```
// dht11.ino - affiche l'humidité et la température avec le capteur DHT11
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int dataPin = 8;
int temperature = -1;
int humidity = -1;
void setup() {
  Serial.begin(115200);
```

```
}
int readDHT11() {
  uint8_t recvBuffer[5];          // 1
  uint8_t cnt = 7;
  uint8_t idx = 0;
  for(int i = 0; i < 5; i++) recvBuffer[i] = 0; // 2
  // Début des communications avec une impulsion LOW
  pinMode(dataPin, OUTPUT);
  digitalWrite(dataPin, LOW);
  delay(18);
  digitalWrite(dataPin, HIGH);
  delayMicroseconds(40);          // 3
  pinMode(dataPin, INPUT);
  pulseIn(dataPin, HIGH);   // 4
  // Lecture du paquet de données
  unsigned int timeout = 10000; // itérations de la boucle
  for (int i=0; i<40; i++)        // 5
  {
          timeout = 10000;
          while(digitalRead(dataPin) == LOW) {
                  if (timeout == 0) return -1;
                  timeout--;
          }
          unsigned long t = micros();   // 6
          timeout = 10000;
          while(digitalRead(dataPin) == HIGH) { // 7
                  if (timeout == 0) return -1;
                  timeout--;
          }
          if ((micros() - t) > 40) recvBuffer[idx] |= (1 << cnt); // 8
          if (cnt == 0)   // octet suivant ?
          {
                  cnt = 7;    // redémarre à l'octet de poids fort
                  idx++;      // octet suivant !
          }
          else cnt--;
  }
  humidity = recvBuffer[0];       // %    // 9
  temperature = recvBuffer[2];  // C
  uint8_t sum = recvBuffer[0] + recvBuffer[2];
  if(recvBuffer[4] != sum) return -2;   // 10
  return 0;
}
void loop() {
  int ret = readDHT11();
  if(ret != 0) Serial.println(ret);
  Serial.print("Humidité : "); Serial.print(humidity); Serial.println(" %");
  Serial.print("Température : "); Serial.print(temperature);
Serial.println(" C");
  delay(2000); // ms
}
```

1 La longueur du paquet reçu à partir du DHT11 est de 5 octets (40 bits).
2 Initialise le buffer avec des zéros.
3 Attend un tout petit peu que le DHT11 commence à envoyer des données.
4 Reçoit le signal de réponse du DHT11.
5 Pour chacun des 40 bits, on accomplit les opérations à l'intérieur des crochets.
6 Temps utilisable en microsecondes (µs) ; stocké pour mesurer la durée de l'impulsion.
7 Mesure l'impulsion avec une boucle.
8 Si l'impulsion est supérieure à 40 µs, c'était un 1. Sinon, le bit garde sa valeur d'initialisation, 0. Pour inverser le bit correspondant dans l'octet correspondant, un nouveau nombre est créé par décalage de bits (par exemple, `1<<2 == 0b100`). Ce bit est ensuite combiné avec la valeur de l'octet précédent grâce à un OR en place au niveau des bits. Dans ces conditions, si la valeur était déjà à 1, elle reste à 1. Si c'était un zéro, mais que le nouveau nombre (0b100) est un 1, alors la valeur passe à 1. Voir la section *Opérations au niveau des bits* du chapitre 8 pour plus de détails.
9 L'humidité est le premier octet du paquet reçu.
10 La somme de contrôle (octet 4) est la somme de l'humidité et de la température.

DHT11 pour Raspberry Pi

Aimeriez-vous combiner un Arduino et un Raspberry Pi pour obtenir le meilleur des deux mondes ? Nous l'espérons car le DHT11 n'aime pas le Raspberry Pi. Le code montre comment lire les données à partir de l'Arduino, de manière à ce que vous puissiez facilement l'adapter à d'autres projets.

Même s'il est possible d'utiliser le DHT11 à partir du Raspberry Pi avec un code complexe, le résultat ne sera pas satisfaisant. Heureusement, Arduino peut être employé pour des tâches de bas niveau, ce qui permet au Raspberry Pi de se concentrer sur ce qui est important.

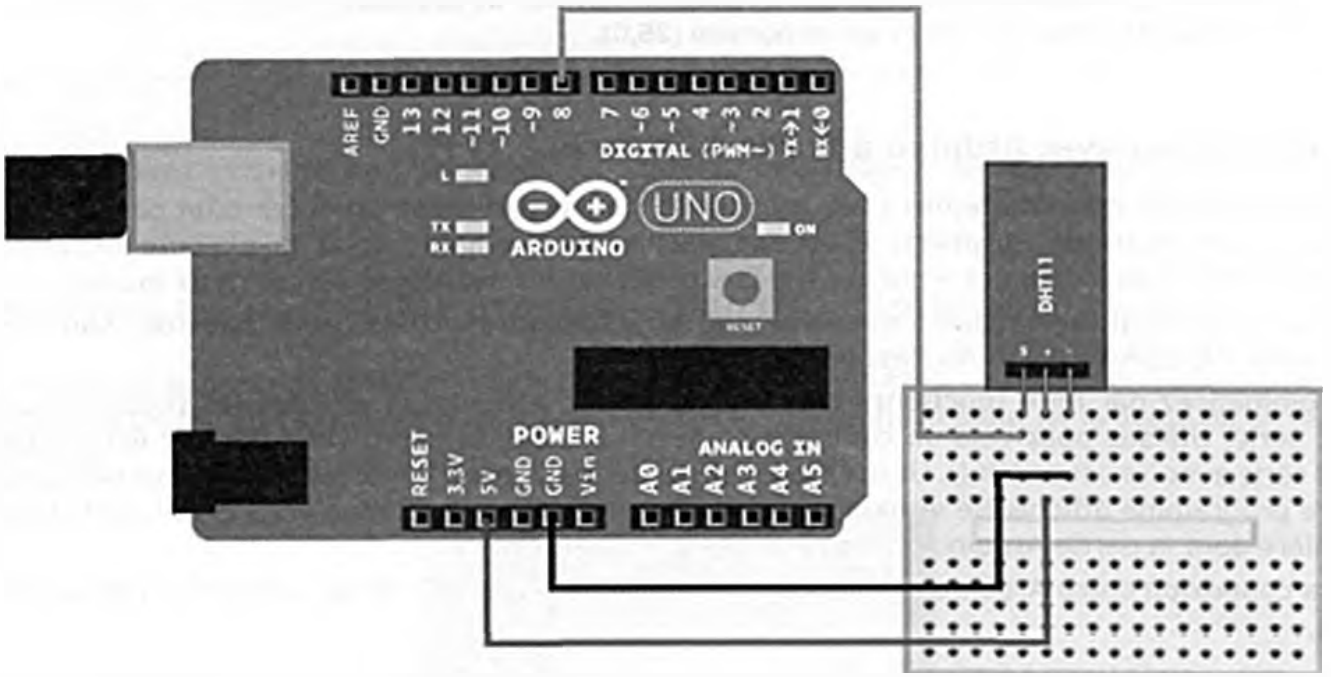
Après avoir testé le code, vous devriez lire plus bas la section *Communiquer avec Arduino à partir d'un Raspberry Pi*.

Pour utiliser le port série du Raspberry Pi, vous devez d'abord le libérer de son usage premier en tant que terminal de connexion. Voir la section *Activation du port série du Raspberry Pi* au chapitre 10 pour les instructions. Câblez selon le schéma illustré à la Figure 11.8, puis exécutez le code de l'Exemple 11.4.

Exemple 11.4. dht11_serial.py

```
# dht11_serial.py - affiche humidité et température avec capteur DHT11
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import serial    # 1
def main():
    port = serial.Serial("/dev/ttyACM0", baudrate=115200, timeout=None) # 2
    while True:
                line = port.readline()  # 3
                arr = line.split()      # 4
                if len(arr) < 3:        # 5
                        continue        # 6
                dataType = arr[2]
                data = float(arr[1])    # 7
                if dataType == '%':
                        print("Humidité : %.1f %%" % data)
                else:
```

```
                print("Température : %.1f C" % data)
            time.sleep(0.01)
if __name__ == "__main__":
    main()
```



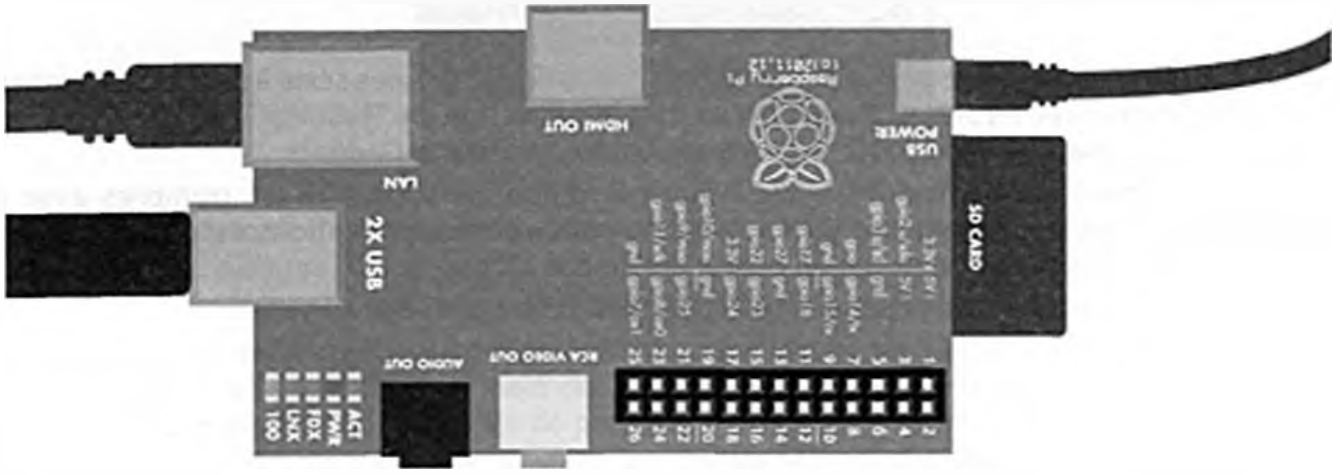


Figure 11.8 Arduino connecté au Raspberry Pi

1 Il est possible que vous ayez besoin d'exécuter ces commandes pour installer cette bibliothèque : sudo apt-get update et sudo apt-get -y install python-serial.
2 Ouvre le port série USB en lecture. Le programme Arduino doit utiliser la même vitesse, 115 200 bits/s pour communiquer.

```
    3 Lit une ligne jusqu'à atteindre une nouvelle ligne. La fonction readline() est une fonction
    bloquante, ce qui signifie qu'elle attend ici jusqu'à ce que quelque chose soit lu.
    4 Décompose la chaîne de caractères en une liste. Cela facilite l'analyse de l'entrée interpré-
    table par l'utilisateur de l'Arduino. Par exemple, "Humidité : 25 %".split() crée une liste pratique :
    ['Humidité :', '25', '%'].
    5 Si la longueur de ligne est incorrecte...
    6 ...on ignore cette ligne et on passe à l'itération suivante de la boucle while.
    7 Convertit la chaîne (« 25 ») en un nombre (25.0).
```

Communiquer avec Arduino à partir d'un Raspberry Pi

Arduino est une machine temps réel, robuste et simple. Il dispose en outre d'un convertisseur analogique-numérique. Raspberry Pi est une machine de plus haut niveau qui exécute un système d'exploitation Linux complet. Pourquoi ne pas combiner les avantages de ces deux machines ?

Vous pouvez utiliser Arduino pour interagir directement avec des capteurs et des sorties, et contrôler l'Arduino à partir du Raspberry Pi.

Commencez par faire fonctionner les capteurs avec Arduino, et affichez les mesures sur le port série. Utilisez pour cela un ordinateur normal et l'EDI Arduino. Vous pouvez écrire sur le port série avec Serial.println(). Utilisez le Moniteur série (**Outils→Moniteur série**) pour voir ce que votre programme affiche. Ne vous occupez du Raspberry Pi que lorsque vous êtes satisfait de la manière dont la partie Arduino de votre projet fonctionne.

La connexion entre Arduino et Raspberry Pi est simple : il suffit de les connecter par un câble USB.

> Pour une combinaison plus étroite de l'Arduino et du Raspberry Pi, jetez un coup d'œil au produit AlaMode for Raspberry Pi (http://bit.ly/1icPd0z), qui incorpore une carte microcontrôleur compatible Arduino dans une extension Raspberry Pi qui se fixe au-dessus du Pi. Il n'y a aucun besoin de connexion USB, puisqu'on utilise le connecteur d'extension du Raspberry Pi.

Arduino communique par le port série via l'USB. Pour lire les données série à partir du Raspberry Pi, on utilise Python et PySerial. Pour un exemple d'utilisation de PySerial, reportez-vous à la Figure 11.8 ou au chapitre 7 de notre livre *Make: Arduino Bots and Gadgets*.

Vous allez obtenir un texte normal d'affichage Arduino, puis extraire les nombres avec les fonctions de gestion de chaînes de Python. Dans de nombreuses applications de prototypage, on n'a pas besoin de concevoir un nouveau protocole de bas niveau.

Si l'on prend l'exemple d'un programme Arduino affichant le libellé suivant :

```
Humidité : 83 %
```

Avec Python, on utiliserait split() pour séparer les mots en une liste :

```
>>> s="Humidité : 83 %"
>>> s.split()
['Humidité :', '83', '%']
```

Vous pouvez faire référence aux éléments de la liste avec des crochets, []. Le premier élément a le numéro zéro. La commande suivante affiche le deuxième élément :

```
>>> x = s.split()[1]
>>> x
'83'
```

Enfin, il faut convertir la chaîne « 83 » en un entier 83 de manière à pouvoir le traiter comme un nombre, car on peut accomplir des opérations mathématiques et des comparaisons avec un nombre, mais pas avec une chaîne.

```
>>> int(x)
83
```

Vous pouvez ainsi utiliser n'importe quel capteur compatible Arduino à partir d'un Raspberry Pi.

CAPTEUR DE PRESSION ATMOSPHÉRIQUE GY65

La mesure de la pression atmosphérique permet de prévoir la météo. Une pression élevée annonce du soleil et un temps clair, alors qu'une pression basse signifie de la pluie, de la neige ou un temps nuageux.

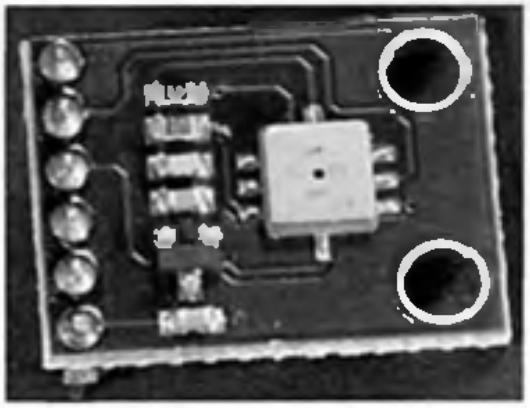


Figure 11.9 Capteur de pression barométrique GY65

La pression normale de l'air se situe aux environs de 1 000 hPa (hectopascals). La valeur exacte dépend de votre altitude.

1 000 hPa = 100 000 Pa = 1 000 mbar = 1 bar

La pression ne varie pas beaucoup, même quand le temps se dégrade fortement. Les records mondiaux sont une pression maximale de 1 084 hPa et minimale de 870 hPa (au centre d'un typhon). D'habitude, les changements de pression d'une météo normale ne sont que de quelques hectopascals.

Les modifications de la pression dans la basse atmosphère sont légères si on les compare à la pression exercée par une pompe à vélo (4 000 hPa) ou bien à la pression qu'il fait à l'extérieur d'un avion quand il vole à 10 000 mètres d'altitude (moins de 300 hPa).

Dans ces conditions, qu'appelle-t-on une pression atmosphérique élevée ? Les météorologues parlent d'une pression élevée quand elle est relativement haute par rapport à la pression des zones environnantes.

Les changements de pression dans le temps permettent de prédire la météo. Si la pression augmente, la météo va s'améliorer et plus le changement est rapide, plus il y aura de soleil.

Pour une prévision plus simple, vous pouvez comparer la pression actuelle à la pression qui est attendue à votre altitude. Vous pouvez trouver votre altitude avec un GPS ou Google Maps. Selon SparkFun, le distributeur du module GY65 que nous avons utilisé, une augmentation de 2,5 hPa signifie un beau temps ensoleillé. À l'inverse, une diminution de la pression de 2,5 hPa en dessous de la pression normale annonce un temps pluvieux.

Le GY65 est une platine d'évaluation pour le capteur BMP085. Vous trouverez sa notice technique en cherchant « notice technique BMP085 ».

GY65 pour Arduino

La Figure 11.10 illustre le diagramme de connexion pour Arduino. Branchez en suivant le schéma et exécutez le code de l'Exemple 11.5.

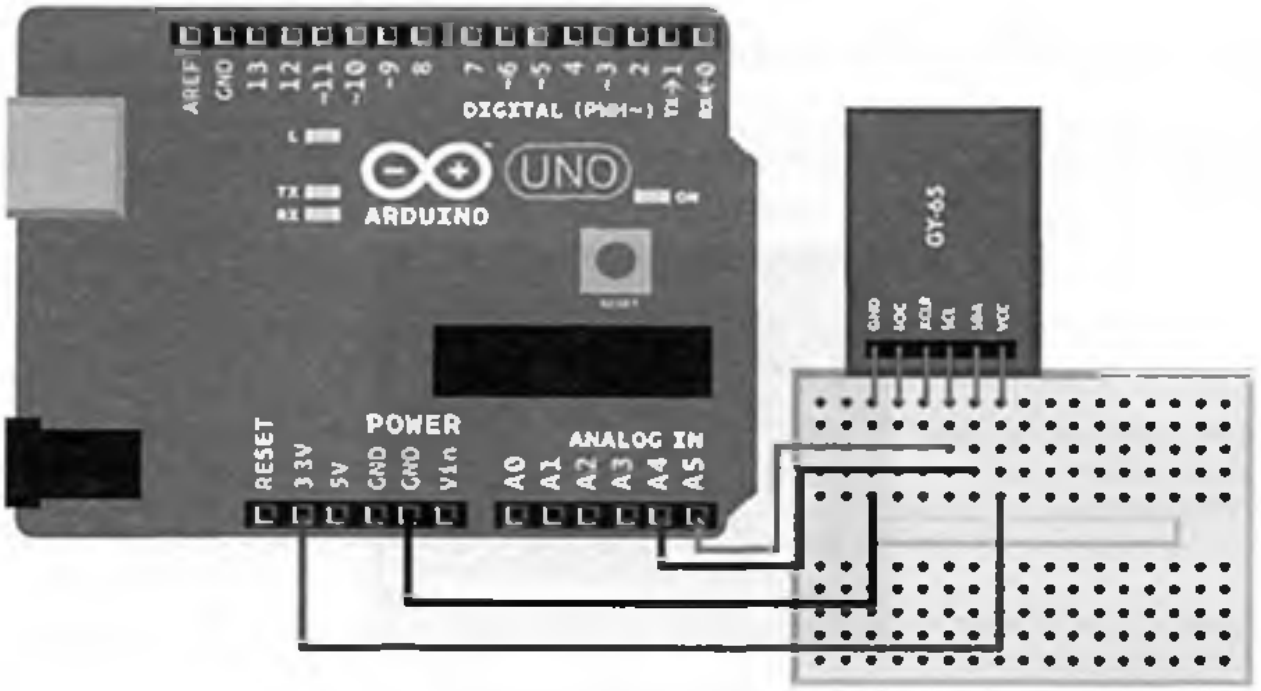


Figure 11.10 Capteur de pression atmosphérique GY65 connecté à un Arduino

Le code Arduino utilise la bibliothèque *gy_65* du site http://botbook.com. Si vous souhaitez une étude technique détaillée de la manière dont le protocole I2C/SMBus fonctionne, lisez plus bas la section *GY65 pour Raspberry Pi*.

Exemple 11.5. gy_65.ino

```
// gy_65.ino - affiche l'altitude, la pression et la température avec un
GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
#include <Wire.h>
#include <gy_65.h>      // 1
void setup() {
  Serial.begin(115200);
  readCalibrationData(); // 2
}
```

```
void loop() {
  float temp = readTemperature();
  float pressure = readPressure();       // 3
  float altitude = calculateAltitude(pressure); // 4
  Serial.print("Altitude : ");
  Serial.print(altitude,2);
  Serial.println(" m");
  Serial.print("Pression : ");
  Serial.print(pressure,2);
  Serial.println(" Pa");
  Serial.print("Température : ");
  Serial.print(temp,2);
  Serial.println("C");
  delay(1000);
}
```

1 La bibliothèque *gy_65* facilite l'usage de ce capteur. Elle masque la complexité du protocole I2C qu'utilise le capteur. Le dossier de la bibliothèque *gy_65* doit se trouver dans le même répertoire que le programme principal, *gy_65.ino*.
2 Met à jour les variables globales.
3 Avec la bibliothèque, c'est un jeu d'enfant de récupérer la pression en pascals.
4 L'air est plus léger quand vous prenez de l'altitude si bien que la pression peut être utilisée pour estimer l'altitude.

Utilisation des bibliothèques Arduino

Si vous avez beaucoup de code, il est préférable de le décomposer en plusieurs fichiers. Le code Arduino du capteur GY65 est dans ce cas-là et vous allez le réutiliser lors du projet *Prévision météo sur papier électronique*.

La bibliothèque est dans un dossier, à l'intérieur du même répertoire que le programme principal (vous avez déjà sans doute travaillé avec des bibliothèques qui ont besoin d'être installées dans le sous-répertoire des sketchs Arduino, *libraries*). Voici la structure du dossier :

```
gy_65.ino          # programme principal
gy_65/             # dossier contenant la bibliothèque
gy_65/gy_65.cpp    # code de la bibliothèque
gy_65/gy_65.h      # prototypes de chaque fonction de la bibliothèque
```

L'emplacement du programme principal, *gy_65.ino*, doit vous sembler normal. Chaque projet Arduino a un programme principal.

La bibliothèque a son propre dossier. Le code de la bibliothèque se trouve dans le fichier *gy_65.cpp*. Ce code ressemble aux autres programmes Arduino que vous avez déjà vus.

Le fichier d'en-tête, *gy_65.h*, contient les prototypes des fonctions du fichier cpp. Les prototypes ne sont que des copies de la première ligne de chaque fonction se trouvant dans *gy_65.cpp*. Par exemple, le fichier d'en-tête comporte la ligne suivante :

```
float readTemperature();
```

Explication de la bibliothèque Arduino GY65

Vous pouvez utiliser la bibliothèque GY65 sans vous préoccuper de ses détails d'implémentation, mais si vous souhaitez bien comprendre la manière dont le protocole de communication du GY65 fonctionne, il vous faut lire ce qui suit.

Vous pouvez apprendre le protocole de communication en étudiant la bibliothèque ou le programme Raspberry Pi (Figure 11.11), voire les deux.

Cette section présente aussi la création de vos propres bibliothèques C++ pour Arduino. Il est souhaitable de consulter également la documentation Arduino sur l'écriture des bibliothèques (http://bit.ly/1iHgFCr).

Le fichier d'en-tête *gy_65.h* (voir l'Exemple 11.6) a des prototypes pour chaque fonction. Il s'agit d'une simple liste de fonctions se trouvant dans le fichier cpp où vous trouverez le véritable code de ces fonctions.

Exemple 11.6. gy_65.h

```
// gy_65.h - bibliothèque pour gérer l'altitude, la pression et la
température avec un GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
void readCalibrationData();
float readTemperature();
float readPressure();
float calculateAltitude(float pressure);
```

L'implémentation des définitions du fichier h se trouve dans un fichier cpp, *gy_65.cpp* (Exemple 11.7). Si certaines parties de ce code vous paraissent complexes, vous pouvez relire les explications du code I2C à la Figure 8.5, ainsi que les sections du chapitre 8 *Hexadécimal, binaire et autres systèmes de numération* et *Opérations au niveau des bits*.

Exemple 11.7. gy_65.py

```
// gy_65.cpp - bibliothèque pour gérer l'altitude, la pression et la
température avec un GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
#include <Arduino.h>
#include <Wire.h>
#include "gy_65.h"
const char i2c_address = 0x77;
int OSS = 0; // Oversampling
const long atmosphereSeaLevel = 101325; // Pa
struct calibration_data // 1
{
  int16_t ac1;
  int16_t ac2;
  int16_t ac3;
  int16_t ac4;
  int16_t ac5;
  int16_t ac6;
  int16_t b1;
  int16_t b2;
  int16_t mb;
```

```
  int16_t mc;
  int16_t md;
};
calibration_data caldata;
long b5;
int16_t swap_int16_t(int16_t value)     // 2
{
  int16_t left = value << 8;
  int16_t right = value >> 8;
  right = right & 0xFF;
  return left | right ;
}
unsigned char read_i2c_unsigned_char(unsigned char address)     // 3
{
  unsigned char data;
  Wire.beginTransmission(i2c_address);
  Wire.write(address);
  Wire.endTransmission();
  Wire.requestFrom(i2c_address,1);
  while(!Wire.available());
    return Wire.read();
}
void read_i2c(unsigned char point, uint8_t *buffer, int size)
{
  Wire.beginTransmission(i2c_address);
  Wire.write(point);
  Wire.endTransmission();
  Wire.requestFrom(i2c_address,size);
  int i = 0;
  while(Wire.available() && i < size) {
    buffer[i] = Wire.read();
    i++;
  }
  if(i != size) {
    Serial.println("Erreur de lecture i2c");
  }
}
int read_i2c_int(unsigned char address) {
    int16_t data;
    read_i2c(address,(uint8_t *)&data,sizeof(int16_t));
    data = swap_int16_t(data);
    return data;
}
void readCalibrationData()      // 4
{
  Wire.begin();
  read_i2c(0xAA,(uint8_t *)&caldata,sizeof(calibration_data)); // 5
  uint16_t *p = (uint16_t*)&caldata;    // 6
  for(int i = 0; i < 11; i++) { // 7
    p[i] = swap_int16_t(p[i]);
  }
}
```

```
float readTemperature() {        // 8
  // Lecture de la température brute
  Wire.beginTransmission(i2c_address);
  Wire.write(0xF4); // Registre
  Wire.write(0x2E); // Valeur
  Wire.endTransmission();
  delay(5); // 9
  unsigned int rawTemp = read_i2c_int(0xF6);
  // Calcule la véritable température
  long x1,x2;
  float t;
  x1 = (((long)rawTemp - (long)caldata.ac6) * (long)caldata.ac5) /
pow(2,15);
  long mc = caldata.mc;
  int md = caldata.md;
  x2 = (mc * pow(2,11)) / (x1 * md);
  b5 = x1 + x2;
  t = (b5 + 8) / pow(2,4);
  t = t / 10;
  return t;      // Celsius
}
long getRealPressure(unsigned long up){ //  10
  long x1, x2, x3, b3, b6, p;
  unsigned long b4, b7;
  int b1 = caldata.b1;
  int b2 = caldata.b2;
  long ac1 = caldata.ac1;
  int ac2 = caldata.ac2;
  int ac3 = caldata.ac3;
  unsigned int ac4 = caldata.ac4;
  b6 = b5 - 4000;
  x1 = (b2 * (b6 * b6) / pow(2,12)) / pow(2,11);
  x2 = (ac2 * b6) / pow(2,11);
  x3 = x1 + x2;
  b3 = (((ac1*4 + x3) << OSS) + 2) / 4;
  x1 = (ac3 * b6) / pow(2,13);
  x2 = (b1 * ((b6 * b6) / pow(2,12)) ) / pow(2,16);
  x3 = ((x1 + x2) + 2) / 4;
  b4 = (ac4 * (unsigned long)(x3 + 32768) ) / pow(2,15);
  b7 = ((unsigned long)up - b3) * (50000 >> OSS);
  if (b7 < 0x80000000)  p = ( b7 * 2 ) / b4;
  else p = (b7 / b4) * 2;
  x1 = (p / pow(2,8)) * (p / pow(2,8));
  x1 = (x1 * 3038) / pow(2,16);
  x2 = (-7357 * p) / pow(2,16);
  p += (x1 + x2 + 3791) / pow(2,4);
  long temp = p;
  return temp;
}
float readPressure() {  // 11
  // Lit la pression non compensée
  Wire.beginTransmission(i2c_address);
```

```
  Wire.write(0xF4); // Register
  Wire.write(0x34 + (OSS << 6)); // Paramètre de suréchantillonnage
  Wire.endTransmission();
  delay(2 + (3 << OSS));
  unsigned char msb,lsb,xlsb;
  unsigned long rawPressure = 0;
  msb = read_i2c_unsigned_char(0xF6);
  lsb = read_i2c_unsigned_char(0xF7);
  xlsb = read_i2c_unsigned_char(0xF8);
  rawPressure = (((unsigned long) msb << 16) |
        ((unsigned long) lsb << 8) |
        (unsigned long) xlsb) >> (8-OSS);
  return getRealPressure(rawPressure);
}
float calculateAltitude(float pressure) {        // 12
  float pressurePart = pressure / atmosphereSeaLevel;
  float power = 1 / 5.255;
  float result = 1 - pow(pressurePart, power);
  float altitude = 44330*result;
  return altitude; // m
}
```

1 Structures et variables globales. Les données de calibrage seront lues à partir de la mémoire EEPROM (non volatile) du capteur.

2 Prend un entier sur deux octets, et permute l'octet gauche avec l'octet de droite. Cela est nécessaire car le capteur a une représentation des nombres différente de celle de l'Arduino.

3 Fonctions utilitaires : `read_i2c_unsigned_char()`, `read_i2c()`, et `read_i2c_int()`. Elles offrent les fonctionnalités de *Wire.h* afin de faciliter son usage dans ce programme.

4 Lit les données de calibrage à partir de la mémoire EEPROM du capteur. Le format est décrit dans la notice technique du BMP085.

5 L'astuce est d'utiliser votre propre structure calibration_data pour décoder les données de l'EEPROM. Les données de l'appareil correspondent à la structure qui réside dans la mémoire de l'Arduino. La structure est créée de telle sorte que chaque variable ait une longueur qui corresponde à chaque donnée lue dans l'EEPROM. Après cette ligne, caldata est remplie avec des octets du capteur, et chaque donnée est accessible via la structure.

6 Comme le capteur a une endianness différente de celle de l'Arduino, vous devez permuter les octets de chaque entier sur 16 bits. D'abord, on crée un pointeur sur deux octets au début de caldata...

7 ...puis on parcourt caldata et on permute les octets. Vous remarquerez la manière dont le pointeur sur 16 bits pointe correctement sur un nouvel entier sur deux octets à chaque itération, au lieu de pointer bêtement sur des octets uniques.

8 `readTemperature()` lit d'abord la température brute avec I2C. Puis elle calcule la véritable température en utilisant la formule de la notice technique.

9 La notice technique spécifie qu'un délai d'au moins 4,5 ms est nécessaire.

10 `getRealPressure()` prend la température brute lue par `readPressure()`, puis retourne le résultat en pascals (Pa). Elle utilise les données de calibrage lues par `readCalibrationData()`, et applique la formule de la page 13 de la notice technique du BMP085.

11 Lit la pression brute avec I2C. Les valeurs du registre et le format des données sont extraits de la notice technique.

12 Estimo l'altitude à partir de la pression. La formule est la formule barométrique internationale qui est disponible à la page 14 de la notice technique du BMP085.

GY65 pour Raspberry Pi

Ce code Raspberry Pi est facile à utiliser et vous devez l'exécuter avant de l'étudier. Si vous le parcourez ligne par ligne, vous constaterez que le code est parfois difficile à comprendre. Avec Raspberry Pi, la communication I2C avec le capteur GY65 ne se trouve pas dans un fichier séparé si bien que le code est plus long que dans l'exemple Arduino. Le code utilise certaines techniques que vous avez déjà vues, comme celles décrites dans les sections du chapitre 8, *Hexadécimal, binaire et autres systèmes de numération* et *Opérations au niveau des bits*.

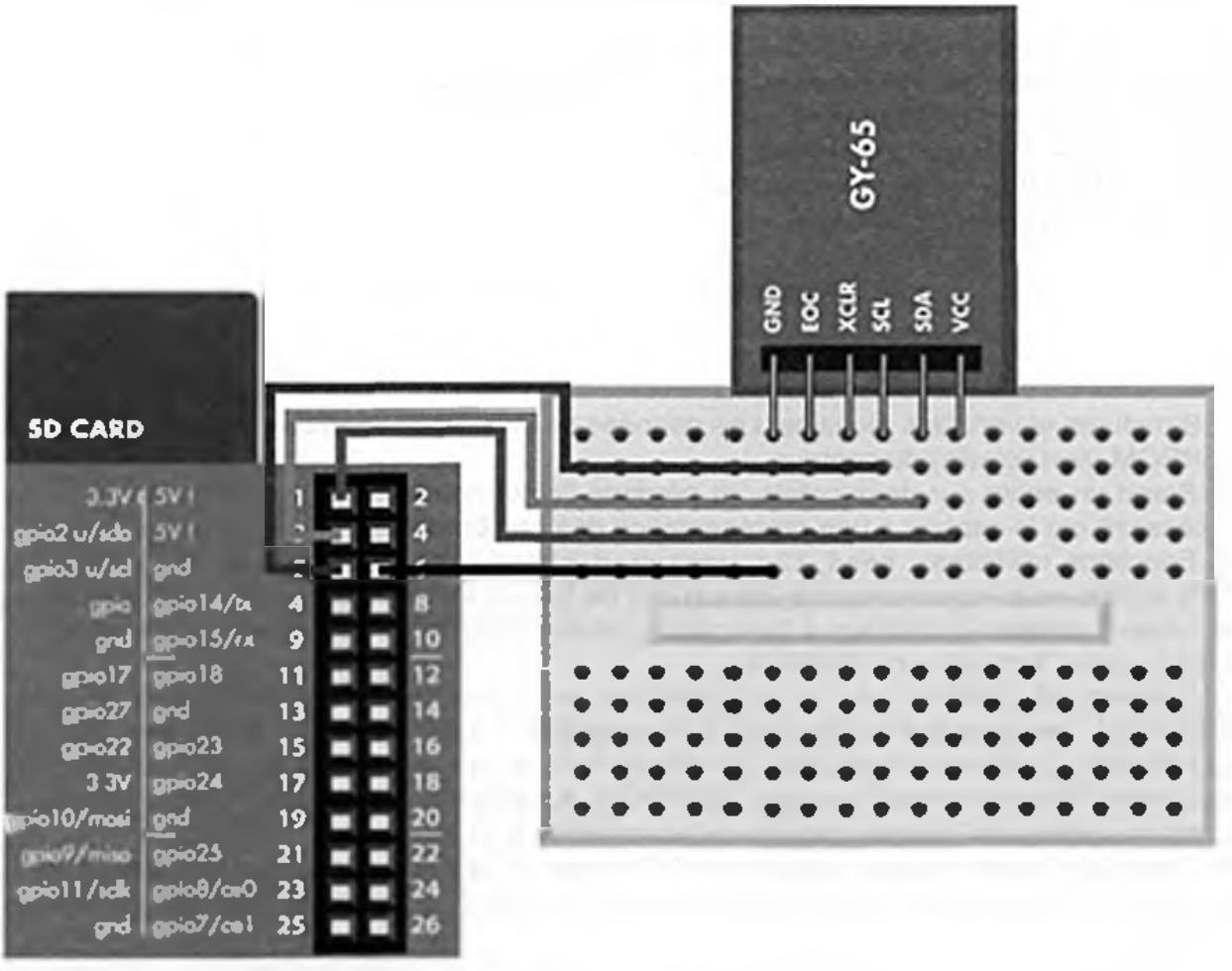


Figure 11.11 Capteur de pression atmosphérique GY65 connecté au Raspberry Pi

Exemple 11.8. gy_65.py

```
# gy_65.py - affiche l'altitude, la pression et la température
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import smbus # sudo apt-get -y install python-smbus      # 1
import time
import struct
```

```
bus = None
address = 0x77
caldata = None
atmosphereSeaLevel = 101325.0
OSS = 0
b5 = 0
def readCalibrationData():      # 2
  global bus, caldata
  bus = smbus.SMBus(1)
  rawData = ""
  for i in range(22):
    rawData += chr(bus.read_byte_data(address, 0xAA+i))  # 3
  caldata = struct.unpack('>hhhhhhhhhhh', rawData)  # 4
def readTemperature():
  global b5
  bus.write_byte_data(address, 0xF4, 0x2E)      # 5
  time.sleep(0.005)     # 6
  rawTemp = bus.read_byte_data(address, 0xF6) << 8          # 7
  rawTemp = rawTemp | bus.read_byte_data(address, 0xF7)
  x1 = ((rawTemp - caldata[5]) * caldata[4]) / 2**15
  x2 = (caldata[9] * 2**11) / (x1 + caldata[10])
  b5 = x1 + x2
  temp = (b5 + 8) / 2**4
  temp = temp / 10.0
  return temp
def readPressure():     # 8
  bus.write_byte_data(address, 0xF4, 0x34 + (OSS << 6))
  time.sleep(0.005)
  rawPressure = bus.read_byte_data(address, 0xF6) << 16
  rawPressure = rawPressure | bus.read_byte_data(address, 0xF7) << 8
  rawPressure = rawPressure | bus.read_byte_data(address, 0xF8)
  rawPressure = rawPressure >> (8 - OSS)
  # Calcule la pression réelle
  b6 = b5 - 4000
  x1 = (caldata[7] * ((b6 * b6) / 2**12 )) / 2**11
  x2 = caldata[1] * b6 / 2**11
  x3 = x1 + x2
  b3 = (((caldata[0] * 4 + x3) << OSS) + 2) / 4
  x1 = caldata[2] * b6 / 2**13
  x2 = (caldata[6] * ((b6 * b6) / 2**12 )) / 2**16
  x3 = ((x1 * x2) + 2) / 2**2
  b4 = (caldata[3] * (x3 + 32768)) / 2**15
  b4 = b4 + 2**16 # convertit de signé en non signé
  b7 = (rawPressure - b3) * (50000 >> OSS)
  if b7 < 0x80000000:
    p = (b7 * 2) / b4
  else:
    p = (b7 / b4) * 2
  x1 = (p / 2**8) * (p / 2**8)
  x1 = (x1 * 3038) / 2**16
  x2 = (-7357 * p) / 2**16
  p = p + (x1 * x2 + 3791) / 2**4
```

```
  return p
def calculateAltitude(pressure):          # 9
  pressurePart = pressure / atmosphereSeaLevel;
  power = 1 / 5.255;
  result = 1 - pressurePart**power;
  altitude = 44330*result;
  return altitude
def main():
  readCalibrationData()
  while True:
    temperature = readTemperature()
    pressure = readPressure()
    altitude = calculateAltitude(pressure)
    print("Altitude %.2f m" % altitude)
    print("Presion %.2f Pa" % pressure)          # 10
    print("Température %.2f C" % temperature)
    time.sleep(10)
if __name__ == "__main__":
  main()
```

1 Le standard SMBus est un sous-ensemble d'I2C. La bibliothèque *smbus* facilite son utilisation. Le paquet *python-smbus* doit être installé sur le Raspberry Pi (voir la section du chapitre 8 *SMBus et I2C sans root*).

2 Le capteur est livré avec 176 bits de données de calibrage stockés dans son EEPROM.

3 Lit les données à partir de l'adresse 0xAA, 0xAB, jusqu'à 0xBF. Ajoute les valeurs caractères à la chaîne. La chaîne se termine en ayant 22 octets (176 bits). Vous n'avez pas à vous soucier du nombre de bits de la structure Arduino quand vous réalisez cela, car la longueur de la structure est définie par la longueur des variables qui sont à l'intérieur.

4 Récupère 11 entiers courts sur deux octets (h) au format big endian (>). Cela consomme tous les octets de la chaîne.

5 Écrit la commande de « lecture de température » (0x2E) sur le registre de contrôle du capteur (0xF4).

6 Attend la fin de la mesure.

7 Lit la réponse et démarre la conversion du nombre brut en une température en degrés Celsius. Les formules sont extraites de la notice technique.

8 La fonction `readPressure()` fonctionne comme `readTemperature()`.

9 Plus on monte, plus la pression diminue. En fonction de cela, la formule barométrique internationale peut fournir une estimation de votre altitude.

10 100 000 Pa = 1 bar

EXPÉRIENCE : EST-CE QUE VOTRE PLANTE A BESOIN D'ÊTRE ARROSÉE ? (CAPTEUR D'HUMIDITÉ DE SOL)

Un capteur d'humidité de sol est un simple capteur de résistance analogique. Plantez-le dans le sol pour voir si votre plante a besoin d'être arrosée.

L'eau du robinet et l'eau des nappes phréatiques contiennent des sels dilués et d'autres substances, ce qui rend l'eau conductrice. Le capteur d'humidité de sol (Figure 11.12) mesure simplement cette conductivité.

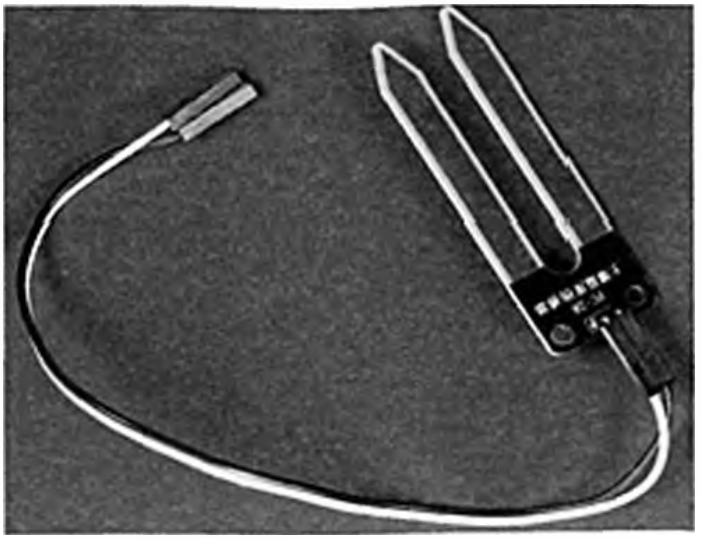


Figure 11.12 Capteur d'humidité de sol

Certains capteurs d'humidité de sol ont un circuit intégré. Le capteur utilisé ici n'en a pas si bien que vous pouvez créer le vôtre avec deux morceaux de métal (pour les sondes du capteur) et utiliser Arduino ou Raspberry Pi pour mesurer la résistance. Le circuit utilise une résistance de 1 MΩ (marron-noir-vert).

Capteur de sol pour Arduino

La Figure 11.13 illustre le schéma de câblage pour Arduino. Branchez en suivant le schéma et exécutez le code de l'Exemple 11.9.

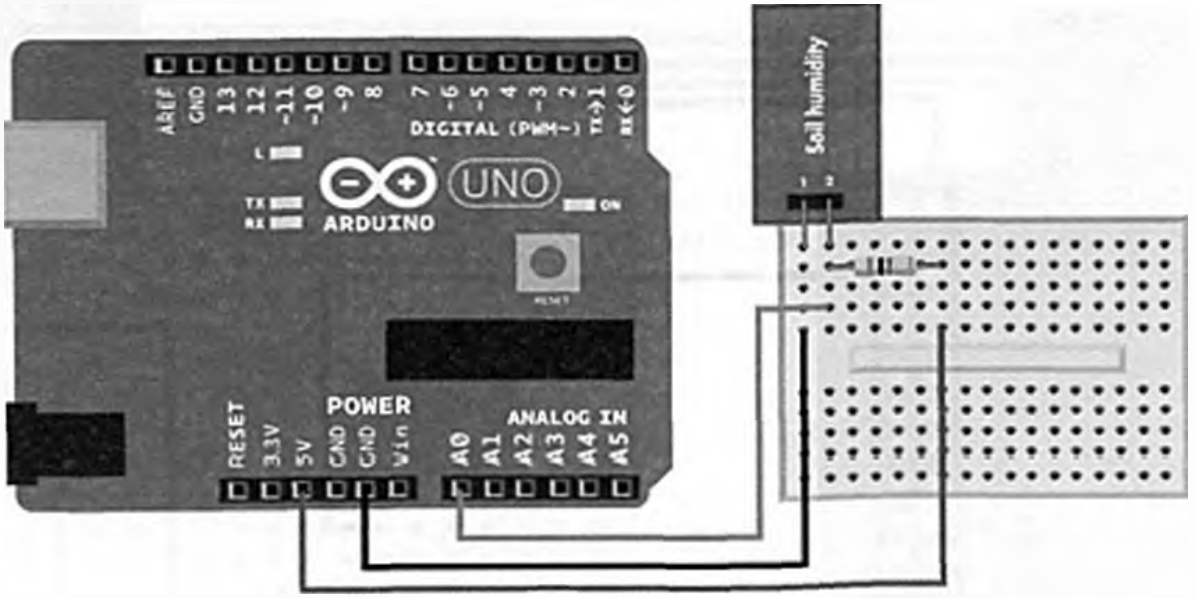


Figure 11.13 Capteur d'humidité de sol connecté à Arduino

Exemple 11.9. soil_humidity_sensor.ino

```
// soil_humidity_sensor.ino - lit l'humidité du sol en mesurant sa
résistance.
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int sensorPin = A0;
int soilHumidity = -1;
void setup() {
  Serial.begin(115200);
}
void loop() {
  soilHumidity = analogRead(sensorPin); // 1
  Serial.println(soilHumidity);
  delay(100);
}
```

1 C'est un simple capteur de résistance analogique.

Capteur de sol pour Raspberry Pi

La Figure 11.14 illustre le diagramme de connexion pour Raspberry Pi. Câblez en suivant le schéma et exécutez le code de l'Exemple 11.10.

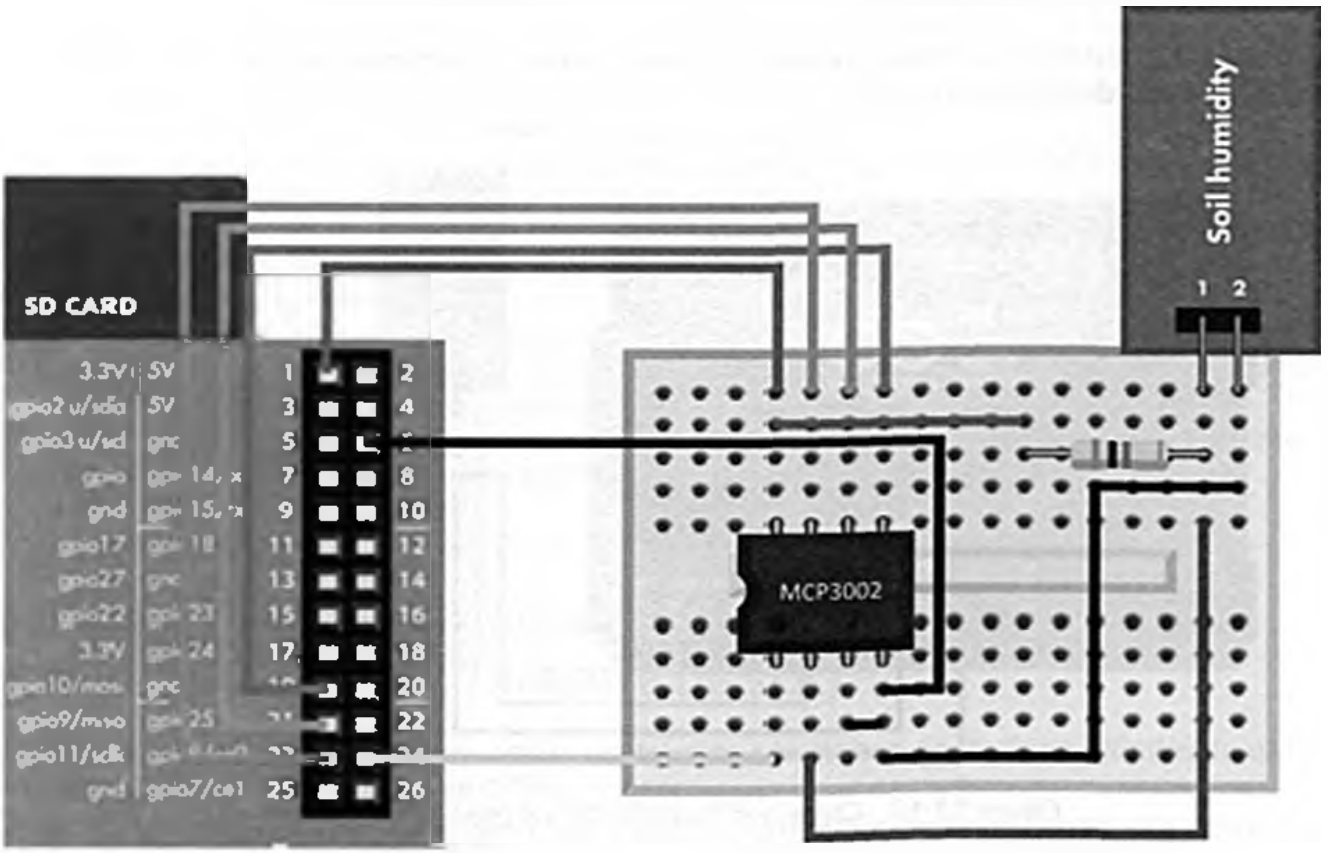


Figure 11.14 Capteur d'humidité de sol connecté à Raspberry Pi

Exemple 11.10. soil_humidity_sensor.py

```
# soil_humidity_sensor.py - lit l'humidité du sol en mesurant sa résistance.
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp
def main():
        while True:
                h = mcp.readAnalog()     # 1
                h = h / 1024 * 100       # 2
                print("Humidité actuelle : %d %%" % h)
                time.sleep(5)
if __name__ == "__main__":
        main()
```

1 C'est un simple capteur de résistance analogique. Comme pour tout autre capteur de résistance analogique de cet ouvrage, la bibliothèque *botbook_mcp3002.py* doit se trouver dans le même répertoire que ce programme. Vous devez aussi installer la bibliothèque *spidev*, qui est importée par *botbook_mcp3002*. Lisez les commentaires au début du fichier *botbook_mcp3002/botbook_mcp3002.py* ou la section *Installation de SpiDev* du chapitre 3.
2 La valeur brute est convertie en pourcentage de la mesure maximale. Pour faciliter l'affichage, h est en fait égal à 100 fois le pourcentage, c'est-à-dire que 0,53 devient 53 pour qu'il soit affiché sous la forme de 53 %.

PROJET : PRÉVISION MÉTÉO SUR PAPIER ÉLECTRONIQUE

Créez votre propre station météo sur papier électronique. La prévision météo est basée sur les modifications de la pression atmosphérique. L'affichage sur papier électronique est assez spécial : il ne brille pas en pleine lumière, il ressemble à du papier et l'image reste affichée sans consommer d'électricité.

> Cette création constitue le projet le plus difficile du livre. Si vous n'avez pas encore testé les expériences et les projets plus simples, il est souhaitable de revenir en arrière et de commencer par les réaliser.

Ce que vous allez apprendre

Dans le projet *Prévision météo sur papier électronique*, vous allez apprendre à :

- Construire un boîtier qui affiche une prévision météo graphique.
- Prédire la météo en vous servant de la pression atmosphérique.
- Afficher un graphique sur du papier électronique qui ne consomme pas d'énergie.
- Mettre l'Arduino en veille pour préserver l'alimentation.

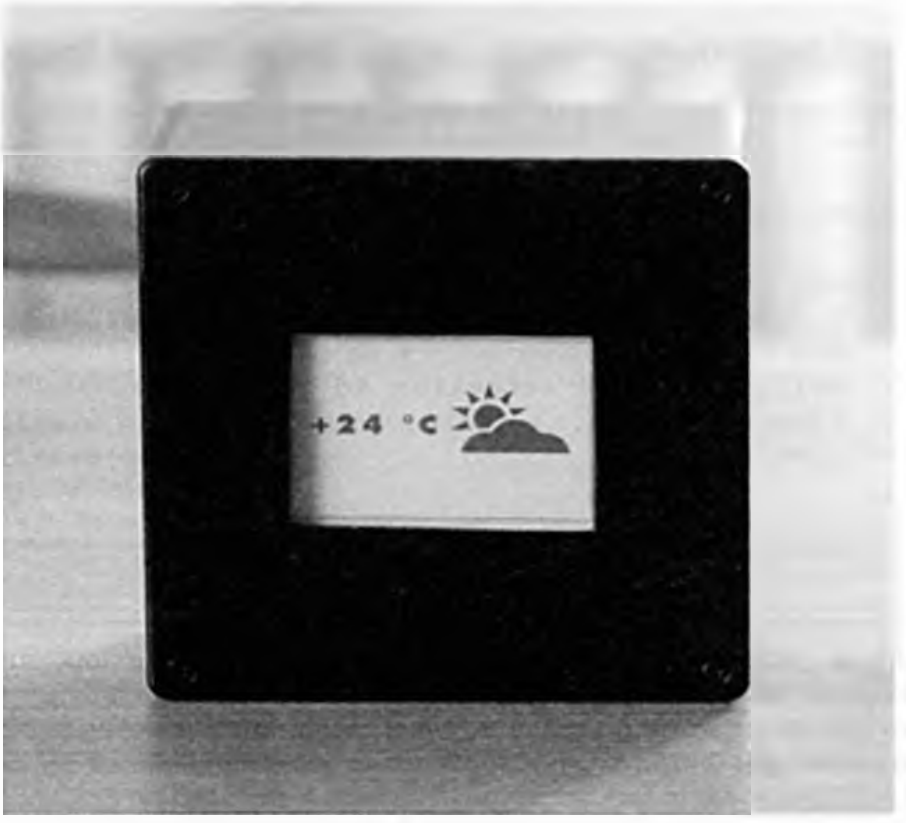


Figure 11.15 Prévision météo sur papier électronique

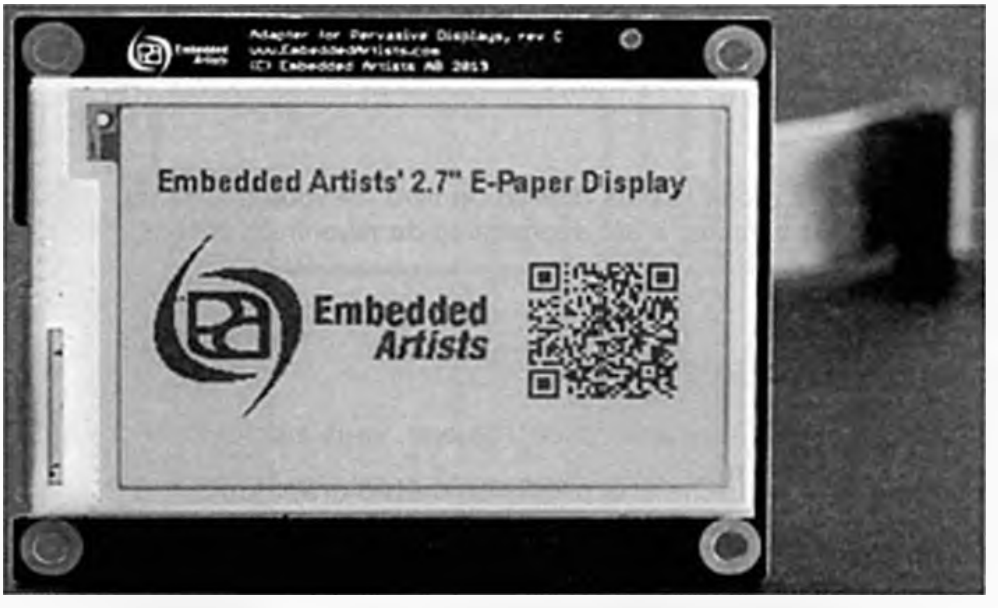


Figure 11.16 Affichage à encre électronique

Prévision météo pour Arduino

Le code combine plusieurs techniques. Vous pouvez simplement commencer par l'exécuter puis étudier par la suite les détails de son implémentation, une fois qu'il fonctionne.

Pour créer votre propre version, il n'est pas nécessaire de comprendre la totalité du code. Une fois que votre station météo fonctionne, jetez un coup d'œil à la fonction drawScreen(). Il s'agit de la fonction principale et vous pouvez, par exemple, modifier pos, l'emplacement où le signe plus est tracé :

```
int pos = 10;
drawCharacter(pos, 70, font,'+');
```

Plusieurs techniques sont utilisées dans ce code :

- Lecture des données du capteur de pression atmosphérique GY65 (voir plus haut *Capteur de pression atmosphérique GY65*).
- Travail avec des nombres hexadécimaux et binaires, et opérations au niveau des bits (voir les sections du chapitre 8 *Hexadécimal, binaire et autres systèmes de numération* et *Opérations au niveau des bits*).
- Mise en veille de l'Arduino pour économiser l'énergie. On utilise des commandes de bas niveau pour écrire directement dans les registres de l'ATmega (la puce de l'Arduino).
- Dessin sur un affichage à encre électronique, avec la bibliothèque EPD.
- Stockage des images sous la forme de fichiers d'en-tête (comme dans imagename.h). Cette technique est expliquée plus bas en détail dans la section *Stockage des images dans des fichiers d'en-tête*.

> Ce code utilise un Arduino Mega. Si vous travaillez avec une autre carte, il faudra modifier le code pour qu'il fonctionne.

La Figure 11.17 illustre les connexions pour Arduino Mega. Câblez en suivant le schéma et exécutez le code de l'Exemple 11.11.

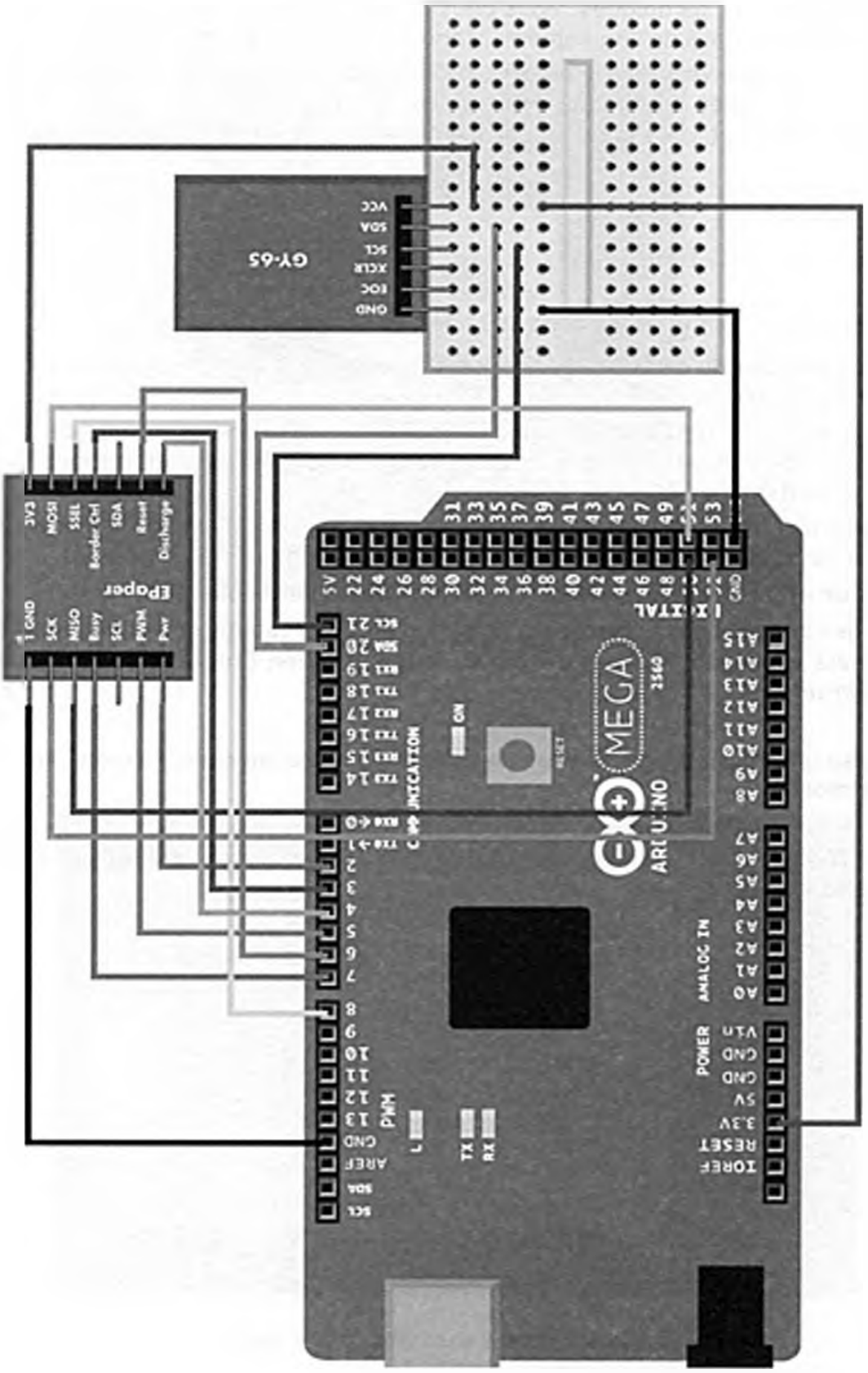


Figure 11.17 Connexions de l'Arduino Mega

Exemple 11.11. weather_station.ino

```
// weather_station.ino - affiche des données météo sur papier électronique
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
#include <inttypes.h>
#include <ctype.h>
#include <SPI.h>
#include <Wire.h>
#include <EPD.h>          // 1
#include <gy_65.h>        // 2
#include <avr/sleep.h>
#include <avr/power.h>
#include "rain.h"         // 3
#include "sun.h"
#include "suncloud.h"
#include "fonts.h"
uint8_t imageBuffer[5808]; // 264 * 176 / 8
const int pinPanelOn = 2;
const int pinBorder = 3;
const int pinDischarge = 4;
const int pinPWM = 5;
const int pinReset = 6;
const int pinBusy = 7;
const int pinEPDcs = 8;
EPD_Class EPD(EPD_2_7,
              pinPanelOn,
              pinBorder,
              pinDischarge,
              pinPWM,
              pinReset,
              pinBusy,
              pinEPDcs,
              SPI);
float weatherDiff;
float temperature;
const int sleepMaxCount = 10;   // min
volatile int arduinoSleepingCount = sleepMaxCount;
void setup() {
  Serial.begin(115200);
  pinMode(pinPanelOn, OUTPUT);
  pinMode(pinBorder, OUTPUT);
  pinMode(pinDischarge, INPUT);
  pinMode(pinPWM, OUTPUT);
  pinMode(pinReset, OUTPUT);
  pinMode(pinBusy, OUTPUT);
  pinMode(pinEPDcs, OUTPUT);
  digitalWrite(pinPWM, LOW);
  digitalWrite(pinReset, LOW);
  digitalWrite(pinPanelOn, LOW);
  digitalWrite(pinDischarge, LOW);
  digitalWrite(pinBorder, LOW);
  digitalWrite(pinEPDcs, LOW);
```

```
  SPI.begin();  // 4
  SPI.setBitOrder(MSBFIRST);     // 5
  SPI.setDataMode(SPI_MODE0);    // 6
  SPI.setClockDivider(SPI_CLOCK_DIV4);  // 7
  WDTCSR |= (1<<WDCE) | (1<<WDE);       // 8
  WDTCSR = 1<<WDP0 | 1<<WDP3;    // 9
  WDTCSR |= _BV(WDIE);  // 10
  MCUSR &= ~( 1 << WDRF);        // 11
  for(int i = 0; i < 5808; i++) // 12
    imageBuffer[i] = 0;
  readCalibrationData();         // 13
}
char characterMap[14] = {'+', '-', 'C', 'd',
                         '0', '1', '2', '3',
                         '4', '5', '6', '7',
                         '8', '9'};     // 14
void drawCharacter(int16_t x, int16_t y, const uint8_t *bitmap, char
character) {
  int charIndex = -1;
  for(int i = 0; i < 14; i++) { // 15
    if(character == characterMap[i]) {
      charIndex = i;
      break;
    }
  }
  if(charIndex == -1) return;
  drawBitmap(x,y,bitmap,charIndex*25,0,25,27,350);      // 16
}
void drawBitmap(int16_t x, int16_t y, const uint8_t *bitmap, int16_t x2,
    int16_t y2, int16_t w, int16_t h, int16_t source_width) {   // 17
  int16_t i, j, byteWidth = source_width / 8;
  for(j=y2; j<y2+h; j++) {       // 18
    for(i=x2; i<x2+w; i++ ) {
      byte b= pgm_read_byte(bitmap+j * byteWidth + i / 8);
      if(b & (128 >> (i & 7))) {
            drawPixel(x+i-x2, y+j-y2, true);
      }
    }
  }
}
void drawPixel(int x, int y, bool black) { // 19
  int bit = x & 0x07;
  int byte = x / 8 + y * (264 / 8);
  int mask = 0x01 << bit;
  if(black == true) {
    imageBuffer[byte] |= mask;
  } else {
    imageBuffer[byte] &= ~mask;
  }
}
void drawBufferToScreen() {     // 20
  for (uint8_t line = 0; line < 176 ; ++line) {
```

```
        EPD.line(line, &imageBuffer[line * (264 / 8)], 0, false,
EPD_inverse);
  }
  for (uint8_t line = 0; line < 176 ; ++line) {
        EPD.line(line, &imageBuffer[line * (264 / 8)], 0, false,
EPD_normal);
  }
}
void drawScreen() {      // 21
  EPD.begin();
  EPD.setFactor(temperature);
  EPD.clear();
  if(weatherDiff > 250) {        // Pa
    // Soleil
    drawBitmap(140,30,sun,0,0,117,106,117);      // 22
  } else if ((weatherDiff <= 250) || (weatherDiff >= -250)) {
    // Partiellement nuageux
    drawBitmap(140,30,suncloud,0,0,117,106,117);
  } else if (weatherDiff < -250) {
    // Pluie
    drawBitmap(140,30,rain,0,0,117,106,117);
  }
  // Affichage de la température
  String temp = String((int)temperature);
  int pos = 10;
  drawCharacter(pos,70,font,'+');        // 23
  pos += 25;
  drawCharacter(pos,70,font,temp.charAt(0));
  pos += 25;
  if(abs(temperature) >= 10) {
    drawCharacter(pos,70,font,temp.charAt(1));
    pos += 25;
  }
  drawCharacter(pos,70,font,'d');
  pos += 25;
  drawCharacter(pos,70,font,'C');
  drawBufferToScreen(); // 24
  EPD.end();
  for(int i = 0; i < 5808; i++) // 25
    imageBuffer[i] = 0;
}
void loop() {
  Serial.println(temperature);  // 26
  if(arduinoSleepingCount >= sleepMaxCount) {    // 27
    readWeatherData();  // 28
    drawScreen();
    arduinoSleepingCount = 0; // 29
    arduinoSleep(); // 30
  } else {
    arduinoSleep();
  }
}
```

```
const float currentAltitude = 40.00; // altitude actuelle en mètres
const long atmosphereSeaLevel = 101325; // Pa
const float expectedPressure = atmosphereSeaLevel * pow((1-currentAltitude /
44330), 5.255);
void readWeatherData(){
  temperature = readTemperature();
  float pressure = readPressure();
  weatherDiff = pressure - expectedPressure;
}
ISR(WDT_vect)   // 31
{
  arduinoSleepingCount++;
}
void arduinoSleep() { // 32
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable();
  sleep_mode(); // 33
  sleep_disable();      // 34
  power_all_enable();
} // 35
```

1 Bibliothèque pour affichage sur papier électronique. Vous pouvez télécharger la bibliothèque à https://github.com/repaperepaper/trees/master/Sketches/libraries. Suivez les instructions fournies à http://arduino.cc/en/Guide/Libraries pour installer la bibliothèque.

2 C'est la bibliothèque GY65 de *botbook.com*. On l'a déjà utilisée dans la section *Capteur de pression atmosphérique GY65*. Copiez le répertoire de la bibliothèque dans le dossier du sketch (si vous avez téléchargé les exemples de code de ce livre, il y figure déjà).

3 Les images sont enregistrées sous la forme de fichiers d'en-tête.

4 Prépare le protocole SPI avant de pouvoir communiquer avec l'affichage à encre électronique.

5 Configure SPI pour avoir le bit de poids fort en premier (voir la section du chapitre 8 *Opérations au niveau des bits*).

6 Configure SPI pour SPI_MODE0 : cela définit la polarité de l'horloge au mode CPOL 0, et la phase de l'horloge au mode CPHA 0. Cela signifie que les données sont capturées sur le front montant de l'horloge (quand le signal d'horloge passe de LOW à HIGH). Les données sont propagées sur le front descendant (quand le signal d'horloge passe de HIGH à LOW). Les modes (SPI_MODE0) sont listées dans la documentation Arduino (**Aide→Référence→Libraries→SPI**). Les paramètres SPI du capteur sont décrits dans sa notice technique.

7 SPI_CLOCK_DIV4 définit l'horloge SPI à un 1/4 de la fréquence du CPU de l'Arduino. Ce projet utilise un Arduino Mega, si bien que la fréquence d'horloge du SPI est de 16 MHz / 4 = 4 MHz.

8 Active l'horloge de surveillance. C'est ce qui va permettre à l'Arduino de se mettre en veille. On définit WDCE (Watch Dog Change Enable) et WDE (Watch Dog Enable) dans le registre WDTCSR (Watchdog Timer Control Register, registre de contrôle de l'horloge de surveillance). Ces registres sont tellement proches de l'architecture de bas niveau du microcontrôleur qu'on ne les trouve que dans la documentation de l'ATmega (et non pas dans la bibliothèque de base Arduino). L'Arduino Mega utilise la puce ATmega 1280, si bien que vous pouvez trouver sa documentation en cherchant « ATmega 1280 datasheet ». Le symbole |= indique un **XOR en place** au niveau des bits (on fait un XOR de la valeur actuelle de la variable et on remplace la valeur de la variable par le résultat). Le symbole << représente l'opération de décalage des bits. Voir la section du chapitre 8 *Opérations au niveau des bits*.

9 Définit la surveillance pour qu'elle réveille l'Arduino toutes les 8 secondes.
10 Active l'interruption de la surveillance. WDIE signifie « Watch Dog Interrupt Enable » (activation de l'interruption de la surveillance).
11 Efface le drapeau de réinitialisation du système de surveillance (WDRF) du registre d'état de l'unité du microcontrôleur (MCUSR).
12 Initialise le buffer d'image (où les dessins sont tracés avant leur affichage) avec des zéros.
13 Pour des détails sur l'utilisation du capteur GY65, reportez-vous plus haut aux explications du code de la section *Capteur de pression atmosphérique GY65*.
14 Liste des 14 caractères qui sont dans *font.h*, une grande image qui contient ces caractères.
15 Trouve l'indice du caractère dans l'image *font.h*.
16 Dessine un caractère à l'écran. En pratique, le fichier bitmap contient une police, une grande image avec des caractères côte à côte. Le calcul sélectionne simplement l'un de ces caractères.
17 `drawBitmap()` prend un fichier source bitmap, sa position, et ses dimensions. Il dessine ensuite chaque pixel de l'image dans un buffer d'image intermédiaire (sans l'afficher pour l'instant sur le papier électronique).
18 Traverse la zone 2D pour dessiner le bitmap pixel par pixel.
19 Dessine un pixel dans le buffer d'image (sans l'afficher pour l'instant sur le papier électronique). L'image qui est stockée contient un bit par pixel, si bien que chaque octet (8 bits) contient 8 pixels. La largeur de l'affichage est de 264 pixels, et chaque ligne contient donc 264/8 = 33 octets. Le code traverse ensuite chaque bit, en le modifiant à 1 ou 0 si nécessaire.
20 Transfère l'image déjà tracée de `imageBuffer` à l'écran en utilisant la bibliothèque EPD. L'affichage comporte 176 lignes de 264 points. Comme un bit représente un pixel, une ligne est stockée dans 33 octets. L'affichage de l'image est une simple question de traçage de chaque octet de `imageBuffer`, avec `EPD.line`.
21 `drawScreen()` utilise la bibliothèque GY65 pour mesurer l'environnement, et la bibliothèque EPD pour dessiner sur l'affichage à encre électronique. Plus la différence de pression est grande, plus la météo sera mauvaise. Si vous voulez créer votre propre version du programme, c'est la fonction `drawScreen()` que vous devez commencer à personnaliser.
22 Dessine l'image stockée dans *sun.h* dans `imageBuffer`. Pour afficher vos propres images à l'écran, utilisez cette fonction. Les paramètres sont `drawBitmap(imageBufferX, imageBufferY, sourceImage, sourceX, sourceY, sourceWidth, sourceHeight, totalSourceWidth)`. Les deux premiers sont la position de la cible (`imageBufferX`, `imageBufferY`). Le reste des paramètres concerne la source du fichier bitmap : le bitmap source (`sourceImage`), et ce que l'on doit extraire de la source (`sourceX`, `sourceY`, `sourceWidth`, `sourceHeight`). Enfin, la largeur totale du bitmap source est listée (`totalSourceWidth`).
23 Écrit un caractère dans le buffer d'image.
24 Affiche `imageBuffer` à partir de la mémoire sur le papier à encre électronique. Sans cela, aucune des images ne s'affiche.
25 Efface le buffer d'image.
26 Avec un composant aussi inhabituel que cet extraordinaire papier à encre électronique, c'est une bonne idée de vérifier les données du capteur séparément en écrivant sur la console série.
27 La surveillance réveille l'Arduino périodiquement. Si suffisamment de temps s'est écoulé...
28 ...il est temps d'exécuter le cœur du programme.
29 On réinitialise ensuite le compteur de veille...
30 ...et on se rendort pour économiser l'énergie.
31 L'interruption ISR (Interrupt Service Routine) est appelée automatiquement. ISR() s'exécute chaque fois que l'Arduino se réveille, juste avant que quelque chose d'autre ne s'exécute. Ce code réveille l'Arduino toutes les 8 secondes, si bien que ISR() s'exécute toutes les 8 secondes.
32 Met en veille l'Arduino pour économiser l'énergie. Comme vous l'avez vu plus haut, cela demande une certaine préparation.
33 Cette commande met en veille l'Arduino et arrête l'exécution du code. Plus rien ne se passe après cela, jusqu'à ce que le système de surveillance ne réveille l'Arduino.

34 Quand la surveillance réveille l'Arduino, le flux de code continue sur cette ligne et le réveil démarre.
35 Vous avez étudié tout ce code difficile ? Vous pouvez être fier de vous ! Vous êtes en passe de devenir un gourou !

EXPÉRIENCE : PAS D'ALIMENTATION !

Les affichages à encre électronique n'utilisent du courant que pour modifier l'image qui s'affiche à l'écran. Ils n'ont donc pas besoin d'énergie pour que l'image reste affichée à l'écran.

Vous pouvez essayer vous-même : prenez un Arduino pour afficher quelque chose sur votre écran à encre électronique, puis éteignez l'Arduino et déconnectez même l'écran. L'image demeure inchangée. En fait, il n'y a aucune différence visible entre le fait de laisser l'écran connecté ou déconnecté (Figure 11.18).

Figure 11.18 L'image reste identique sur un affichage à encre électronique même s'il n'est pas alimenté

STOCKAGE DES IMAGES DANS DES FICHIERS D'EN-TÊTE

Votre station météo peut déjà vous montrer le soleil, mais comment pouvez-vous afficher vos propres images ? Vous pouvez les dessiner dans un programme comme GIMP ou Photoshop, puis convertir les images enregistrées au format BMP en en-têtes de code C, comme ceux que nous avons utilisés dans le sketch.

Commencez par utiliser votre logiciel de dessin favori pour tracer une image. GIMP, qui est un logiciel libre, convient très bien. Enregistrez votre image au format BMP. Placez l'image dans le même dossier que les autres images de ce projet (*images/*).

Ensuite, il faut convertir les images BMP (*sun.bmp*) en un fichier d'en-tête C (*sun.h*). Vous pouvez utiliser le script inclus *image2c-botbook.py* pour cela.

Il faut d'abord installer les prérequis : Python et la bibliothèque Python Imaging. Les utilisateurs Windows peuvent télécharger un installateur gratuit pour Python à http://python.org. Les utilisateurs de Mac ont déjà une version de Python installée, mais les utilisateurs de Mac et de Windows auront besoin de télécharger la bibliothèque Python Imaging à partir de http://www.

pythonware.com/products/pil/. Les Linuxiens ont la vie plus facile : voici, par exemple, comment installer la bibliothèque nécessaire :

```
$ sudo apt-get update
$ sudo apt-get -y install python python-imaging
```

Si votre image source ne s'appelle pas *test.bmp*, modifiez le nom de fichier dans le script. Vous êtes ensuite prêt pour la conversion. Ces commandes fonctionnent sous Linux, Windows et Mac. Le signe dollar ($) représente l'invite du shell et vous n'avez donc pas à le saisir. Les utilisateurs de Windows ont probablement une invite d'aspect différent. La première commande présuppose que vous êtes dans le sous-répertoire qui contient le code de ce livre.

```
$ cd arduino/weather_station/
$ python image2c-botbook.py
BMP (117, 106) 1
```

Le fichier est à présent converti. Avec la commande ls ou dir, vous pouvez constater que le fichier *test.h* a été créé dans le répertoire de travail en cours. L'image BMP est maintenant convertie en un fichier d'en-tête C.

Dans *test.h*, votre image est à présent du code C :

```
// Fichier généré par image2c-botbook.py
const unsigned char sun [] PROGMEM= {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ..
0x7F, 0xF0, 0x00, 0xFF, 0xC0, 0x00, 0x1F, 0xF8, 0x00, // ..
```

Ce format est pratique pour la programmation. Chaque octet représente huit pixels, un pixel par bit. Par exemple, hex 0xFC est le nombre décimal 252, soit en binaire 0b11111100. Ces huit bits côte à côte représentent les points suivants : noir noir noir noir, noir noir blanc blanc.

Programme de conversion de BMP en C

Image2c-botbook.py convertit une image au format BMP en fichier d'en-tête C.

Si vous aimez les défis (voir http://codegolf.com), essayez de modifier ce programme pour qu'il fonctionne avec des fichiers PNG.

Ce code utilise des nombres hexadécimaux et des opérations au niveau des bits (voir les sections du chapitre 8 *Hexadécimal, binaire et autres systèmes de numération* et *Opérations au niveau des bits*).

Exemple 11.12. Conversion d'une image en en-tête C

```
# image2c-botbook - convertit une image BMP en C pour affichage sur écran
LCD ou encre électronique
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import Image     # 1
import math
imageName = "images/test.bmp"     # 2
outputName = "test.h"     # 3
im = Image.open(imageName)       # 4
print im.format, im.size, im.mode
width, height = im.size
```

```
pixels = list(im.getdata())      # 5
length = int(math.floor(width * height / 8.0))  # 6
carray = length * [0x00]         # 7
for y in xrange(0, height):      # 8
        for x in xrange(0, width):      # 9
                pixel = pixels[y * width + x]   # 10
                bit = 7 - x & 0x07      # 11
                byte = x / 8 + y * (width / 8)  # 12
                mask = 0x01 << bit      # 13
                if pixel == 0:
                        carray[byte] |= mask    # 14
                else:
                        carray[byte] &= ~mask
fileHandle = open(outputName, "w")      # 15
index = 0
fileHandle.write("// Fichier généré avec image2c-botbook.py\n")
fileHandle.write("const unsigned char sun [] PROGMEM= {\n")
for b in carray:
        fileHandle.write("0x%02X, " % b)        # 16
        index += 1
        if index > 15:
                fileHandle.write("\n")
                index = 0
fileHandle.write("};\n")
```

1 PIL, la bibliothèque Python Imaging, doit être installée comme cela est décrit plus haut.
2 L'image source au format BMP. Changez cela en fonction du nom et de l'emplacement de votre image.
3 Nom du fichier de sortie. Changez cela pour que ça corresponde à ce que vous voulez.
4 PIL peut ouvrir plusieurs formats.
5 Décompose l'image en pixels individuels. C'est la moitié du travail.
6 Longueur de l'image totale en octets (1 octet = 8 bits).
7 Crée le tableau C cible et l'initialise avec des zéros.
8 Pour chaque ligne...
9 ...et pour chaque pixel de la ligne en cours, on accomplit les opérations indentées.
10 Le pixel en cours.
11 L'indice du bit de l'octet en cours.
12 L'indice de l'octet en cours.
13 Crée le masquage de bits pour le bit en cours de l'octet en cours. Par exemple, le masque 0b00000001 est créé pour modifier le dernier bit d'un octet à zéro.
14 Change le bit.
15 Les fichiers d'en-têtes C ne sont que des fichiers texte.
16 Écrit chacun des octets sous la forme d'un code hexadécimal.

CONSEILS DE PACKAGING

Nous avons utilisé un boîtier en plastique de chez Hammond pour notre station météo. Mais comment faire un trou de la forme étrange dont nous avons besoin ici ? Commencez par tracer la forme que vous voulez découper sur le couvercle du boîtier. Percez des trous dans chaque angle

de la forme avec une mèche de 10 à 20 mm, puis découpez de coin en coin avec une lame de scie. Peaufinez le trou avec une lime et du papier de verre (Figure 11.19).

Figure 11.19 Trou pour l'écran

Collez à chaud l'écran au boîtier comme cela est illustré à la Figure 11.20.

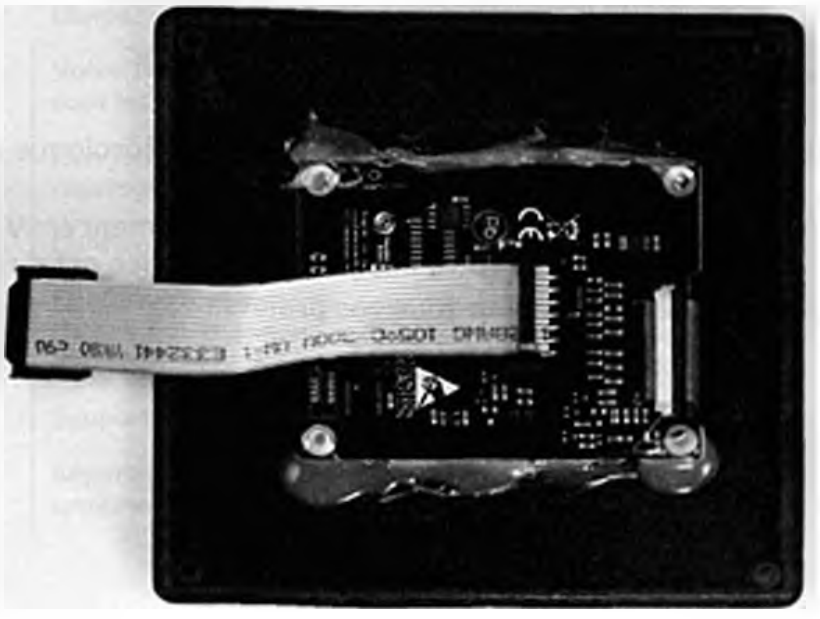


Figure 11.20 Écran collé à chaud

Sur l'arrière du boîtier, nous avons percé une grille formée de petits trous (Figure 11.21). Sans cela, l'air ne pourrait pas passer et atteindre le capteur. Le résultat final est illustré à la Figure 11.22.

Figure 11.21 Trous pour laisser passer l'air

Figure 11.22 Station météo finalisée

Félicitations ! Vous avez maintenant votre propre oracle météorologue avec un affichage à encre électronique.

C'est la fin de cet ouvrage, mais vos projets ne font que commencer. Vous savez à présent travailler avec tellement de capteurs que votre imagination créatrice est au pouvoir...

Nous vous souhaitons bonne chance pour tous vos projets !

Annexe

RÉFÉRENCES PRATIQUES SUR LA VERSION LINUX DU RASPBERRY PI

Le premier tableau indique les répertoires avec lesquels vous travaillerez la plupart du temps. Le second tableau liste quelques commandes Linux qui sont utiles.

Répertoires les plus importants

Répertoire	Usage
/home/pi/	Votre (vous êtes l'utilisateur nommé pi) répertoire d'accueil contient tous les fichiers du Pi
/var/log/	Contient tous les fichiers journaux du système, comme /var/log/syslog et /var/log/auth.log
/etc/	Contient tous les fichiers de configuration du système.
/sys/	Fichier virtuel système pour lire et modifier les données volatiles (les choses qui changent continuellement quand le système est en cours d'exécution, comme les broches d'entrée et de sortie)
/media/	Supports amovibles comme /media/cdrom/ ou /media/usbdisk/
/	Répertoire racine : contient chaque répertoire et chaque fichier du système

Commandes Linux utiles

Commande	Signification
pwd	Affiche votre répertoire de travail
ls	Liste les fichiers du répertoire de travail
cd Desktop	Positionne dans le répertoire *Desktop* qui est situé immédiatement sous votre répertoire en cours
cd ~/Desktop	Positionne dans le répertoire *Desktop* qui est situé immédiatement sous votre répertoire d'accueil
cd ~	Positionne dans votre répertoire d'accueil
nano test.txt	Modifie le fichier texte *test.txt*. Utilisez Ctrl+X, puis saisissez *y* suivi de la touche Entrée pour enregistrer le fichier.
passwd	Modifie votre mot de passe (demande d'abord l'ancien, mais ce que vous saisissez ne s'affiche pas à l'écran)
startx	Lance le bureau graphique à partir de la ligne de commande Linux
sudo apt-get update	Met à jour la liste des logiciels que vous pouvez installer (nécessite une connexion réseau)
sudo apt-get install ipython	Installe le programme *ipython*
sudo shutdown -P now	Prépare l'arrêt du Raspberry Pi en toute sécurité
sudoedit /etc/motd	Modifie un fichier en tant que root, avec plus de contrôles de sécurité que nano sudo /etc/motd
ifconfig	Affiche les adresse IP (127.0.0.1 est égal à localhost, qui est utilisé pour les connexions entre les programmes s'exécutant sur le Pi ; utilisez l'autre adresse qui est celle de l'adapteur Ethernet ou WiFi)
sudo raspi-config	Lance le menu de configuration des paramètres les plus courants du Raspberry Pi (il faut en général redémarrer après la modification des paramètres)
mkdir bar/	Crée un répertoire appelé *bar*
rm -r bar/	Supprime le répertoire *bar/* et son contenu (il n'y a pas de commande Annuler)
ssh login@exemple.com	Connecte à l'ordinateur distant exemple.com avec le nom d'utilisateur « login »
exit	Ferme le shell ou la connexion ssh
less /var/log/syslog	Affiche un fichier texte (appuyez sur la barre d'espace pour passer à la page suivante, et appuyez sur q pour quitter)
tail -F /var/log/syslog	Suit le fichier texte spécifié quand on y ajouter des lignes (utilisez Ctrl+C pour interrompre la commande *tail* et revenir au shell)
man ssh	Visualise la page du manuel (intégré à la documentation) de la commande « ssh » (appuyez sur la barre d'espace pour passer à la page suivante, et appuyez sur q pour quitter)

Index

Les capteurs pour Arduino et Raspberry Pi

Tutoriels et projets

Vous avez envie de concevoir des montages avec Arduino ou Raspberry Pi qui interagissent avec leur environnement ?

Pour cela vous avez besoin de capteurs, et cet ouvrage vous aidera à passer rapidement des idées à la réalisation.

Chaque chapitre est consacré à un type de capteur (mouvement, lumière, son, etc.) et comporte :

- des **expériences** qui expliquent la manière d'utiliser un capteur ;
- des **tests** de validation ;
- un **mini-projet** qui montre comment combiner différentes technologies pour obtenir un montage performant.

Les nombreux **exemples de code commentés** vous seront précieux pour créer vos propres projets.

Les montages que vous pourrez réaliser :

- ✔ un éthylotest personnel,
- ✔ un détecteur de fumée qui envoie un courriel d'alerte,
- ✔ une sonnette hantée qui sonne avant qu'on ne la touche,
- ✔ un jeu vidéo Pong,
- ✔ un dôme lumineux sensible à la couleur,
- ✔ un écran graphique qui réagit aux sons ambiants,
- ✔ une station météo...

RESSOURCES NUMÉRIQUES

Le code source des programmes et de nombreux liens et références utiles sont disponibles sur :
www.dunod.com/contenus-complementaires/9782100717934
ainsi que sur botbook.com, le site de référence de la version d'origine.

Tero Karvinen est professeur d'informatique à l'université d'Hel

Kimmo Karvine est directeur informatique dans une société spécialisée en automatique en Finlande.

Ville Valtokari est chef de proje en électronique, mais aussi bricol passionné et prolifique.

Traduit de l'américain par Dominique Manle

9 782100 717934

7900060
ISBN 978-2-10-071793-4

Make:

DUNOD
dunod.co