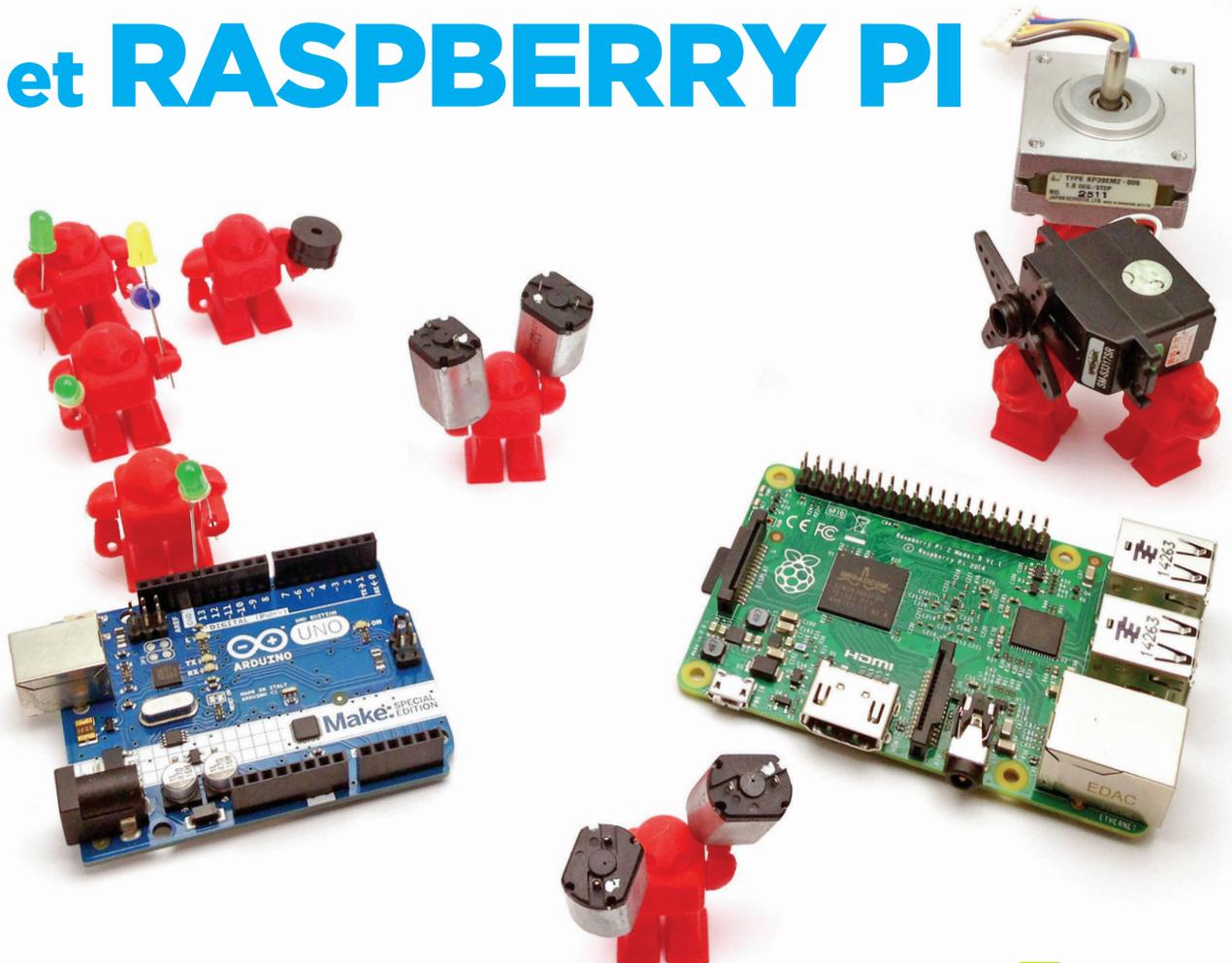


Simon Monk

Mouvement, lumière et son avec ARDUINO et RASPBERRY PI

Avec
30
projets
ludiques



EYROLLES

SERIAL
MAKERS

À l'action avec Arduino et Raspberry Pi !

Cet ouvrage à vocation pratique explique comment créer et contrôler des mouvements, de la lumière et du son à l'aide d'un Arduino et d'un Raspberry Pi. Avec à l'appui 30 projets ludiques à réaliser, il détaille comment utiliser ces deux plates-formes pour contrôler des LED, des moteurs de divers types, des bobines, des dispositifs à courant alternatif, des pompes, ou encore des systèmes d'affichage ou de production de sons. Il se clôt par des projets permettant de contrôler des mécanismes et des systèmes avec Internet, faisant ainsi pénétrer le lecteur dans le monde des objets connectés. Le maker, qui aura déjà eu l'occasion d'utiliser un Arduino ou un Raspberry Pi pour mesurer le monde réel à l'aide de capteurs, passera ici à l'action en découvrant les bases de l'automatisation.

À qui s'adresse ce livre ?

Aux makers, amateurs d'électronique, bricoleurs, bidouilleurs...

Dans ce livre, vous apprendrez notamment à :

- créer un système d'arrosage automatique de vos plantes avec Arduino
- mettre au point un rafraîchisseur de boissons
- fabriquer une marionnette qui danse en fonction de vos tweets
- concevoir un éclateur aléatoire de ballon

Sur www.serialmakers.com

Téléchargez le code source des programmes Arduino et Raspberry Pi présentés dans cet ouvrage.

Simon Monk est l'auteur de nombreux best-sellers d'électronique pour les makers. Il aide également sa femme, qui a lancé le site web MonkMakes.com, à concevoir et vendre des kits et d'autres articles liés à ses ouvrages. Vous pouvez le suivre sur Twitter et en savoir plus sur ses livres sur www.simonmonk.org.

Make:
makezine.com

**Mouvement,
lumière et son**
avec **ARDUINO**
et **RASPBERRY PI**

CHEZ LE MÊME ÉDITEUR

Dans la collection « Serial Makers »

C. PLATT. – **L'électronique en pratique (2^e édition).**

N°14425, 2016, 328 pages.

E. DE KEYSER. – **Filmer et photographier avec un drone.**

N°14241, 2016, 168 pages.

F. BOTTON. – **Les drones de loisir (2^e édition).**

N°14436, 2016, 224 pages.

R. JOBARD. – **Les drones (2^e édition).**

N°14189, 2016, 202 pages.

C. PLATT. – **L'électronique en pratique 2.**

N°14179, 2015, 336 pages.

E. BARTMANN. – **Le grand livre d'Arduino (2^e édition).**

N°14117, 2015, 612 pages.

C. BOSQUE, O. NOOR et L. RICARD. – **FabLabs, etc. Les nouveaux lieux de fabrication numérique.**

N°13938, 2015, 216 pages.

M. RICHARDSON et S. WALLACE. – **À la découverte du Raspberry Pi.**

N°13747, 2013, 176 pages.

B. PETTIS, A. KAZIUNAS FRANCE et J. SHERGILL. – **Imprimer en 3D avec la MakerBot.**

N°13748, 2013, 226 pages.

J. BOYER. – **Réparez vous-même vos appareils électroniques.**

N°13936, 2014, 384 pages.

A. KAZIUNAS FRANCE *et al.* – **Pratique de l'impression 3D.**

N°13924, 2014, 228 pages.

Simon Monk

**Mouvement,
lumière et son**
avec **ARDUINO**
et **RASPBERRY PI**

EYROLLES



ÉDITIONS EYROLLES
61, bld Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Authorized French translation of the English edition of *Make: Action* ISBN 978-1-457-18779-7 © 2016 Simon Monk. This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to sell the same.

Traduction autorisée de l'ouvrage en langue anglaise intitulé *Make: Action* de Simon Monk (ISBN : 978-1-457-18779-7), publié par Maker Media, Inc.

Adapté de l'anglais par Danielle Lafarge, relecture technique par Jean Boyer

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie, 20, rue des Grands Augustins, 75006 Paris.

Table des matières

1. Arduino vs Raspberry Pi	1
Arduino	1
Raspberry Pi	3
Quelle carte choisir ?	4
Les alternatives	5
Résumé	7
2. Arduino	9
Qu'est-ce qu'un Arduino ?	9
Installation de l'IDE Arduino	11
Téléversement d'un sketch	13
Le code du livre	14
Guide de programmation	15
Setup et Loop	15
Variables	16
Sorties numériques	16
Entrées numériques	17
Entrées analogiques	18
Sorties analogiques	19
If/Else	20
Boucles	21
Fonctions	22
Résumé	24

3. Raspberry Pi	25
Qu'est-ce qu'un Raspberry Pi ?	25
Configuration du Raspberry Pi	27
Préparation d'une carte microSD avec NOOBS	28
Configuration de SSH	29
SSH sur un ordinateur sous Windows	31
SSH sur Mac ou Linux	32
La ligne de commande Linux	32
Le code du livre	34
Guide de programmation	35
Hello, World	35
Tabulations et retraits	36
Variables	36
if, while, etc.	37
La bibliothèque RPi.GPIO	37
Le connecteur GPIO	37
Sorties numériques	38
Entrées numériques	39
Sorties analogiques	39
Résumé	39
4. Premières expériences	41
Plaque d'essai sans soudure (breadboard)	41
Fonctionnement d'une plaque d'essai	43
Raccordement d'une plaque d'essai à l'Arduino	44
Raccordement d'une plaque d'essai au Raspberry Pi	44
Téléchargement des programmes	45
Expérience : pilotage d'une LED	46
Composants nécessaires	46
Réalisation du circuit	46
Montage Arduino	47
Programme Arduino	48
Expérimentation Arduino	49
Montage Raspberry Pi	49
Programme Raspberry Pi	51
Expérimentation Raspberry Pi	52

Comparaison du code	52
Expérience : pilotage d'un moteur	53
Composants nécessaires	53
Réalisation du circuit	54
Expérimentation sans Arduino ni Raspberry Pi	55
Montage Arduino	56
Expérimentation Arduino	57
Montage Raspberry Pi	57
Expérimentation Raspberry Pi	58
Résumé	58
5. Notions d'électronique	59
Courant, tension et résistance	59
Courant	59
Tension	60
Masse	61
Résistance	61
Puissance	62
Principaux composants	63
Résistances	63
Transistors	64
Diodes	70
LED	70
Condensateurs	71
Circuits intégrés	71
Les tenants et les aboutissants des connexions	71
Sorties numériques	72
Entrées numériques	72
Entrées analogiques	72
Sorties analogiques	73
Communication série	73
Résumé	73
6. LED	75
LED ordinaires	76
Limitation de courant	76
Projet : feu tricolore	78

Composants nécessaires	80
Montage	80
Montage Arduino	80
Programme Arduino	81
Montage Raspberry Pi	82
Programme Raspberry Pi	83
MLI et LED	84
LED RGB	85
Expérience : arc-en-ciel de couleurs	86
Matériel	86
Composants nécessaires	88
Montage Arduino	88
Programme Arduino	89
Expérimentation Arduino	90
Montage Raspberry Pi	90
Programme Raspberry Pi	91
Expérimentation Raspberry Pi	93
Résumé	93
7. Moteurs, pompes et vérins	95
Régulation de la vitesse par MLI	96
Expérience : régulation de la vitesse d'un moteur CC	96
Matériel	97
Montage Arduino	97
Programme Arduino	98
Expérimentation Arduino	100
Montage Raspberry Pi	100
Programme Raspberry Pi	101
Expérimentation Raspberry Pi	102
Pilotage d'un moteur CC à l'aide d'un relais	102
Commutation d'un relais avec Arduino ou Raspberry Pi	104
Modules relais	105
Expérience : pilotage d'un moteur CC à l'aide d'un module de relais	106
Composants nécessaires	106
Câblage	106
Programme Arduino	107
Programme Raspberry Pi	108

Choix d'un moteur	108
Couple	109
Tr/min	109
Engrenages	110
Moteurs à engrenages	110
Pompes	110
Pompes péristaltiques	111
Pompes rotodynamiques	113
Projet : système d'arrosage Arduino pour plantes d'intérieur	114
Montage	114
Composants nécessaires	115
Construction	116
Programme	118
Application du projet	120
Vérins électriques	120
Solénoïdes	122
Résumé	123
8. Régulation avancée d'un moteur	125
Ponts en H	126
Pont en H sur un circuit intégré	128
Expérience : commande du sens et de la vitesse d'un moteur	130
Composants nécessaires	131
Réalisation du circuit	133
Expérimentations	134
Montage Arduino	136
Programme Arduino	138
Expérimentation Arduino	140
Raccordement du Raspberry Pi	140
Programme Raspberry Pi	141
Expérimentation Raspberry Pi	143
Autres circuits intégrés de ponts en H	144
L298N	144
TB6612FNG	148
Modules à ponts en H	148
Projet : compacteur de canettes Arduino	150
Composants nécessaires	150

Câblage	151
Construction	152
Programme Arduino	152
Résumé	153
3. Servomoteurs	155
Servomoteurs	155
Commande d'un servo	156
Expérience : commande de la position d'un servomoteur	157
Matériel	158
Composants nécessaires	158
Raccordement de l'Arduino	159
Programme Arduino	160
Expérimentation Arduino	161
Raccordement du Raspberry Pi	162
Programme Raspberry Pi	163
Expérimentation avec le Raspberry Pi	164
Projet : Pepe, la marionnette Raspberry Pi qui danse	165
Composants nécessaires	166
Montage	166
Construction	167
Programme	174
Utilisation de Pepe la marionnette	175
Résumé	176
10. Moteurs pas-à-pas	177
Moteurs pas-à-pas	178
Moteurs pas-à-pas bipolaires	178
Expérience : pilotage d'un moteur pas-à-pas bipolaire	181
Composants nécessaires	182
Montage	182
Arduino	183
Montage Arduino	183
Programme Arduino (version compliquée)	185
Programme Arduino (version simplifiée)	187
Expérimentation Arduino	188
Raspberry Pi	189

Montage Raspberry Pi	189
Programme Raspberry Pi	190
Expérimentation Raspberry Pi	192
Moteurs pas-à-pas unipolaires	193
Réseaux de transistors Darlington	194
Expérience : pilotage d'un moteur pas-à-pas unipolaire	195
Matériel	195
Composants nécessaires	196
Montage Arduino	197
Montage Raspberry Pi	197
Programme	198
Micropas	198
Expérience : micropas avec le Raspberry Pi	199
Composants nécessaires	200
Montage Raspberry Pi	200
Programme	201
Expérimentations	203
Moteurs CC sans balais	203
Résumé	204
11. Chauffage et refroidissement	205
Résistances chauffantes	205
Expérience : résistance chauffante	205
Composants nécessaires	206
Construction	206
Expérimentations	207
Projet : éclateur aléatoire de ballons Arduino	207
Composants nécessaires	208
Matériel	208
Programme	210
Utilisation de l'éclateur de ballon	211
Éléments chauffants	211
Puissance et énergie	212
De la puissance à la hausse de température	212
Eau bouillante	212
Éléments Peltier	213
Fonctionnement des éléments Peltier	213

Considérations pratiques	215
Projet : réfrigération de boisson	217
Composants nécessaires	217
Construction	218
Application du projet	219
Résumé	220
12. Boucles de commande	221
Le thermostat simple	221
Expérience : précision de la régulation thermostatique marche/arrêt	222
Composants nécessaires	223
Montage	224
Réalisation du circuit	225
Programme	226
Expérimentations	229
Hystérésis	230
Régulation PID	231
Proportionnel (P)	232
Intégral (I)	234
Dérivé (D)	234
Réglage d'un régulateur PID	234
Expérience : régulation thermostatique PID	236
Matériel	236
Programme Arduino	236
Expérimentation Arduino	239
Raccordement du Raspberry Pi	243
Programme Raspberry Pi	244
Expérimentation Raspberry Pi	248
Projet : système thermostatique de réfrigération de boisson	249
Matériel	250
Composants nécessaires	250
Montage	251
Construction	252
Programme Arduino	255
Résumé	258

13. Contrôle d'appareils à courant alternatif	259
Théorie de la commutation du courant alternatif	259
Qu'est-ce que le courant alternatif ?	260
Relais	261
Optocoupleurs	262
Triacs et optocoupleurs à sortie triac	262
Commutation de courant alternatif en pratique	264
Modules relais	264
Relais statiques	266
PowerSwitch Tail	267
Projet : interrupteur à minuterie Raspberry Pi	267
Composants nécessaires	268
Construction	268
Programme	269
Application du projet	270
Résumé	270
14. Afficheurs	271
Rubans LED	271
Expérience : pilotage d'un ruban de LED RGB	272
Composants nécessaires	273
Montage Arduino	273
Programme Arduino	274
Connexions Raspberry Pi	275
Programme Raspberry Pi	277
Écrans OLED I2C	279
Expérience : utilisation d'un module d'affichage I2C avec un Raspberry Pi	280
Composants nécessaires	281
Connexions	281
Programme	281
Expérimentation	283
Projet : ajout d'un écran au système de réfrigération de boisson	284
Composants nécessaires	284
Connexions	284
Programme	285
Résumé	286

15. Son	287
Expérience : enceinte non amplifiée et Arduino	287
Composants nécessaires.....	288
Réalisation du circuit	288
Programme Arduino.....	289
Expérimentation Arduino.....	290
Amplificateurs	290
Expérience : lecteur de fichiers son sur un Arduino	291
Composants nécessaires.....	291
Création des données audio	291
Code Arduino	293
Expérimentation Arduino.....	294
Raccordement d'un Arduino à un amplificateur	294
Lecteur de fichiers audio sur le Raspberry Pi	297
Projet : Pepe, la marionnette Raspberry Pi qui parle	297
Composants nécessaires.....	298
Réalisation du circuit	299
Programme.....	300
Utilisation de Pepe la marionnette	302
Résumé	302
16. L'Internet des objets	303
Raspberry Pi et Bottle	304
Projet : un interrupteur web Raspberry Pi	305
Matériel	305
Programme.....	306
Utilisation de l'interrupteur web	307
Arduino et les réseaux.....	307
Projet : Pepe la marionnette annonce l'arrivée de nouveaux messages	309
Connexion de Pepe à Internet	311
IFTTT (If This Then That).....	313
Utilisation du projet	316
Résumé	316

A. Fournisseurs et composants	317
Principaux fournisseurs	317
Résistances et condensateurs	318
Semi-conducteurs	319
Autres fournitures.....	320
Modules, moteurs et alimentations.....	320
Brochage des composants	322
B. Brochage du connecteur GPIO du Raspberry Pi	323
Index	325

Arduino vs Raspberry Pi

1

Il n'a jamais été aussi facile pour un amateur de faire ses premiers pas dans l'univers de l'électronique avec les plates-formes Arduino et Raspberry Pi. Vous pourrez tout aussi bien créer un système de domotique permettant de réguler l'éclairage et le chauffage de votre maison via le réseau Wi-Fi ou simplement piloter quelques moteurs.

Mais même si Arduino et Raspberry Pi sont tous les deux des circuits de la taille d'une carte de crédit, ils diffèrent par de nombreux aspects. Arduino est une carte à microcontrôleur rudimentaire qui n'est pas dotée d'un quelconque système d'exploitation, tandis que Raspberry Pi est un nano-ordinateur qui fonctionne sous Linux et qui peut être interfacé avec des composants électroniques externes.

Arduino

Il existe un vaste choix de cartes Arduino. Dans ce livre, nous nous concentrerons sur le modèle le plus répandu : Arduino Uno (figure 1-1). Une carte Arduino coûte un peu moins cher qu'un Raspberry Pi. Il vous faudra déboursier 25 € environ pour une Arduino Uno.

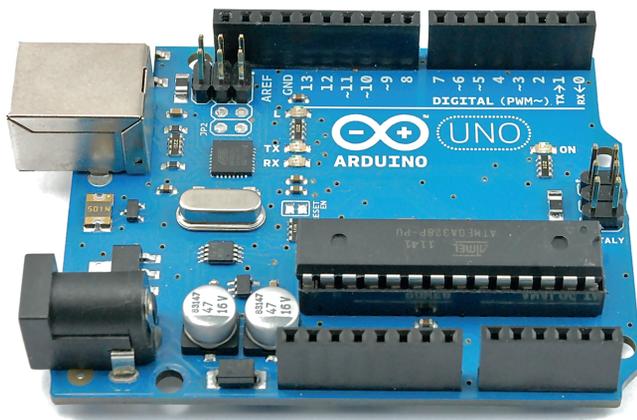


Figure 1-1. Une platine Arduino Uno Revision 3

Si vous avez l'habitude de travailler sur un PC ordinaire, les caractéristiques techniques d'Arduino vous paraîtront totalement insuffisantes. La carte ne dispose que de 34 Ko de mémoire (de différents types). Cela signifie que Raspberry Pi a environ 30 000 fois plus de mémoire, sans compter la mémoire flash de la carte SD du Pi ! En outre, Arduino Uno est équipé d'un processeur qui n'est cadencé qu'à 16 MHz. Il n'est pas possible de brancher un clavier, une souris ou un écran à la platine. Enfin, il n'y a pas non plus de système d'exploitation.

Peut-être vous interrogez-vous sur l'utilité de cette nanocarte. Le secret réside dans son extrême simplicité. Il n'y a pas de système d'exploitation à initialiser ou d'interfaces superflues qui ne font qu'accroître les coûts et consommer de l'énergie.

Alors que Raspberry Pi est un ordinateur polyvalent, Arduino ne joue qu'un seul rôle – la connexion et le pilotage de composants électroniques.

Pour programmer un Arduino, vous avez besoin d'un ordinateur ordinaire (ou d'un Raspberry Pi, si vous le souhaitez) sur lequel un IDE (*Integrated Development Environment*) a préalablement été installé. Cet environnement de programmation vous sert à écrire un programme qui sera ensuite transféré dans la mémoire flash d'Arduino.

L'Arduino ne peut exécuter qu'un seul programme à la fois. Une fois programmé, il le gardera en mémoire et l'exécutera automatiquement à chaque initialisation.

Arduino est conçu pour recevoir des shields, c'est-à-dire des cartes qui s'enfichent sur ses prises d'entrée-sorties afin de doter la platine de fonctionnalités matérielles supplémentaires, comme divers types d'écrans ou des adaptateurs Ethernet et Wi-Fi.

Un Arduino se programme à l'aide du langage de programmation C (vous en apprendrez plus sur la programmation et Arduino au chapitre 2).

Raspberry Pi

Si vous débutez dans le domaine de l'électronique, mais que vous savez utiliser un ordinateur, vous vous sentirez en terrain connu avec Raspberry Pi (figure 1-2). La nanocarte est une version réduite d'un ordinateur fonctionnant sous Linux. Elle est dotée de ports USB permettant de connecter un clavier et une souris, ainsi que d'une sortie audio et d'une sortie vidéo HDMI pour le branchement d'un écran.

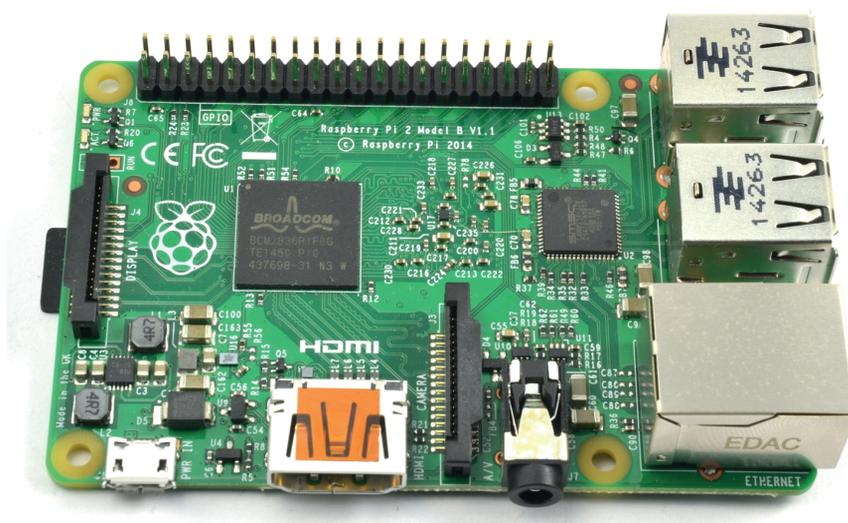


Figure 1-2. Un Raspberry Pi 2

Vous pouvez relier Raspberry Pi à un réseau via son port Ethernet ou par le biais d'adaptateurs Wi-Fi USB. La carte est alimentée par un port micro USB.

Il n'y a pas de disque dur ; une carte microSD est utilisée comme support de stockage. Elle contient le système d'exploitation, ainsi que tous vos documents et programmes.

À l'origine, Raspberry Pi a été créé au Royaume-Uni pour servir d'ordinateur bon marché destiné à l'enseignement des bases de l'informatique, et notamment de la programmation en Python, à des écoliers. D'ailleurs, « Pi » serait dérivé du « Py » de Python.

Qu'est-ce qui différencie le Raspberry Pi d'un ordinateur de bureau ou d'un ordinateur portable fonctionnant sous Linux ?

Quelle carte choisir ?

- Il coûte moins de 40 € (le modèle A+ qui est une version simplifiée du Raspberry Pi est encore meilleur marché et le modèle Zéro est quasiment donné).
- Il nécessite une alimentation de moins de 5 watts.
- Raspberry Pi est doté de deux rangées de broches GPIO (*General-Purpose Input/Output*, en haut à gauche sur la figure 1-1) auxquelles vous pouvez connecter directement des composants électroniques, tels que des LED, un écran, des moteurs, ainsi que les différents types de périphériques de sortie utilisés pour la réalisation des montages présentés dans ce livre.

Par ailleurs, Raspberry Pi peut être connecté à Internet via le Wi-Fi ou un câble LAN, ce qui permet de réaliser des projets dans le domaine de l'Internet des objets (voir le chapitre 16).

Raspberry Pi 2 (dernière et meilleure version au moment de l'écriture de ce livre) présente les caractéristiques suivantes :

- processeur ARM v7 900 MHz Quad-Core ;
- 1 Go de mémoire DDR2
- Ethernet 10.100 BaseT
- quatre ports USB 2.0
- sortie vidéo HDMI
- prise pour caméra
- connecteur GPIO 40 broches (qui fonctionnent toutes en 3,3 V)

Si vous découvrez Raspberry Pi, reportez-vous au chapitre 3 où nous présentons les composants matériels et le langage de programmation Python.

Quelle carte choisir ?

Ce livre explique comment connecter des composants électroniques à Arduino et Raspberry Pi parce que certains projets se prêtent mieux à l'un ou à l'autre. D'autres cartes qui se situent entre ces deux extrêmes partagent plus de points communs avec l'un ou l'autre et les explications fournies dans ce livre resteront valables.

Au moment de vous lancer dans un nouveau projet, je vous conseille d'opter en priorité pour une carte Arduino. Toutefois, si le projet présente l'une des exigences suivantes, il sera sans doute préférable de privilégier une carte Raspberry Pi :

- Internet ou connexion réseau
- grand écran
- clavier ou souris
- périphériques USB, tels qu'une webcam

En ne ménageant pas ses efforts et son porte-monnaie, il est possible d'étendre les fonctionnalités d'Arduino par des shields qui répondront à la plupart des exigences précédentes. Toutefois, les montages sont plus compliqués, car ce ne sont pas des fonctionnalités natives d'Arduino alors qu'elles le sont pour Pi.

Parmi les bonnes raisons d'utiliser un Arduino plutôt qu'un nano-ordinateur Raspberry Pi, il faut citer :

Coût

Un Arduino Uno est moins cher qu'un Raspberry Pi 2.

Temps de démarrage

Avec un Arduino, on n'a pas besoin d'attendre que le système d'exploitation démarre. Il y a un temps de latence d'une seconde environ pendant que le processeur vérifie si un nouveau programme est en cours de transfert. Ensuite, le module est prêt.

Fiabilité

Une platine Arduino est intrinsèquement beaucoup plus simple et résistante qu'un micro-ordinateur Raspberry Pi et elle n'est pas supervisée par un système d'exploitation.

Consommation d'énergie

Un Arduino consomme 1/10^e environ de l'énergie nécessaire au fonctionnement d'un Raspberry Pi. Si des piles ou l'énergie solaire sont utilisées comme source d'alimentation, mieux vaut opter pour une platine Arduino.

Courant de sortie GPIO

Les broches GPIO du Raspberry Pi doivent uniquement être utilisées pour fournir un courant maximum d'environ 16 mA. En revanche, les broches d'Arduino autorisent une sortie nominale de 40 mA. Dans certains cas, vous pouvez donc connecter directement un composant (comme une LED) à un Arduino alors que ce n'est pas possible avec un Raspberry Pi.

L'Arduino et le Raspberry Pi sont tous les deux d'excellentes cartes sur lesquelles baser vos projets. Dans une certaine mesure, le choix est aussi une question de préférence personnelle.

De manière générale, lorsque vous connectez des composants externes à un Raspberry Pi, souvenez-vous qu'il fonctionne en 3,3 V, alors qu'Arduino utilise une tension de 5 V. Si vous branchez un composant fonctionnant en 5 V à l'une des broches GPIO du Raspberry Pi, vous risquez d'endommager la broche, voire de détruire la carte.

Les alternatives

L'Arduino Uno et le Raspberry Pi se situent chacun à une extrémité de la palette des cartes permettant de piloter des éléments. Comme vous pouvez vous en douter, d'autres cartes intermédiaires ont été produites dans le but de pallier les faiblesses des unes et des autres.

De nouvelles platines voient régulièrement le jour. La nature open source d'Arduino permet de donner naissance à de nombreuses variations, ainsi qu'à des modules répondant à des besoins spécifiques, comme le contrôle de drones ou l'interfaçage avec des capteurs sans fils.

La figure 1-3 réunit quelques exemples de cartes les plus populaires dans ce domaine.

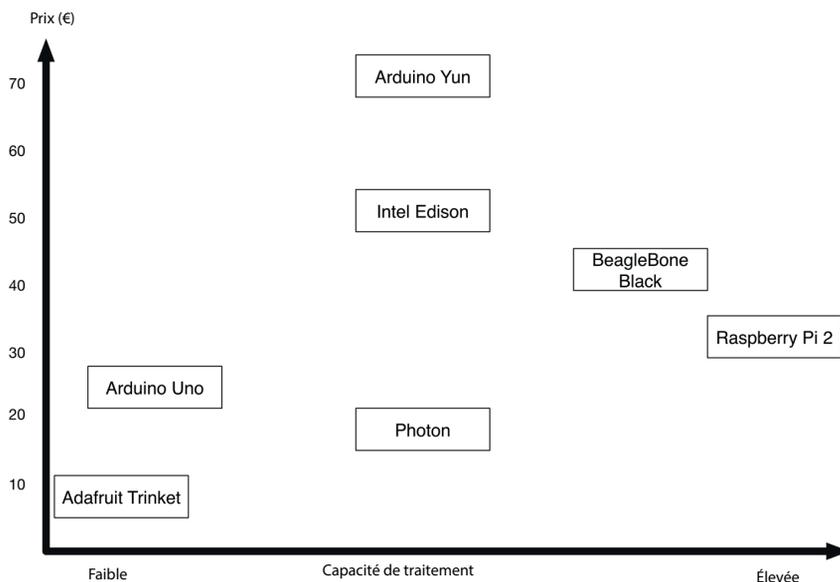


Figure 1-3. Plates-formes intégrées

Le modèle Adafruit Trinket se place avant l'Arduino Uno à la fois en termes de prix et de performance. Cette carte intéressante ne possède que quelques broches GPIO. Mais, elle est relativement compatible avec l'Arduino. Elle mérite donc d'être envisagée pour un projet ne nécessitant qu'une ou deux entrées ou sorties.

Il existe une catégorie de produits intermédiaires comprenant les cartes Arduino Yun, Intel Edison et Photon, qui intègrent toutes des fonctionnalités Wi-Fi et sont destinées à des projets dans le domaine de l'Internet des objets (IoT, *Internet of Things*, voir le chapitre 16). Parmi ces cartes, le modèle Photon présente probablement le meilleur rapport qualité-prix. Ces trois cartes se programment en Arduino C. Par conséquent, tout ce que vous aurez appris sur la carte Arduino s'appliquera aussi à ces cartes.

La carte BeagleBone Black est très proche du Raspberry Pi du point de vue de son concept. Il s'agit aussi d'un ordinateur monocarte et même si la version actuelle du BeagleBone Black arrive derrière le Raspberry Pi 2 en termes de puissance brute, elle présente néanmoins des avantages

sur ce dernier : elle compte davantage de broches GPIO, certaines pouvant même être utilisées comme des entrées analogiques, ce qui n'est pas possible sur le Raspberry Pi 2. En outre, la platine BeagleBone Black peut être programmée en Python, comme Raspberry Pi, ou en JavaScript.

Résumé

Dans ce chapitre, nous avons brièvement présenté Arduino et Raspberry Pi, avec leurs avantages et leurs inconvénients, et nous avons évoqué quelques alternatives. Dans les deux chapitres suivants, nous examinerons le fonctionnement et la programmation d'une carte Arduino, puis d'un nano-ordinateur Raspberry Pi.

Si vous avez déjà utilisé ces deux composants électroniques, vous pouvez aller directement au chapitre 4 où nous passerons à la pratique. Vous pourrez toujours revenir aux chapitres 2 et 3 si vous en éprouvez le besoin.

Ce chapitre est une version modifiée de l'annexe de mon livre *The Maker's Guide to the Zombie Apocalypse*, paru chez NoStarch Press. Je l'ai repris ici avec leur aimable autorisation.

Si vous faites vos premiers pas avec Arduino, ce chapitre vous expliquera les bases de l'utilisation de très petite carte.

Qu'est-ce qu'un Arduino ?

Il existe différents types de cartes Arduino, mais la plus répandue est l'Arduino Uno (figure 2-1). C'est celle qui est utilisée pour tous les projets de ce livre.

L'Arduino Uno a connu des versions successives, ou révisions. Le modèle illustré à la figure 2-1 est une révision 3 (R3). C'est la plus récente à l'heure où j'écris ces lignes.

Nous commencerons notre présentation d'Arduino par la prise USB qui joue plusieurs rôles : elle peut servir à l'alimentation de la carte, à sa programmation depuis l'ordinateur et, enfin, comme liaison de communication.

Le petit bouton rouge qui se trouve à côté de la prise USB est le bouton de réinitialisation. Lorsque vous appuyez dessus, l'Arduino redémarre et exécute le programme qui y est installé.

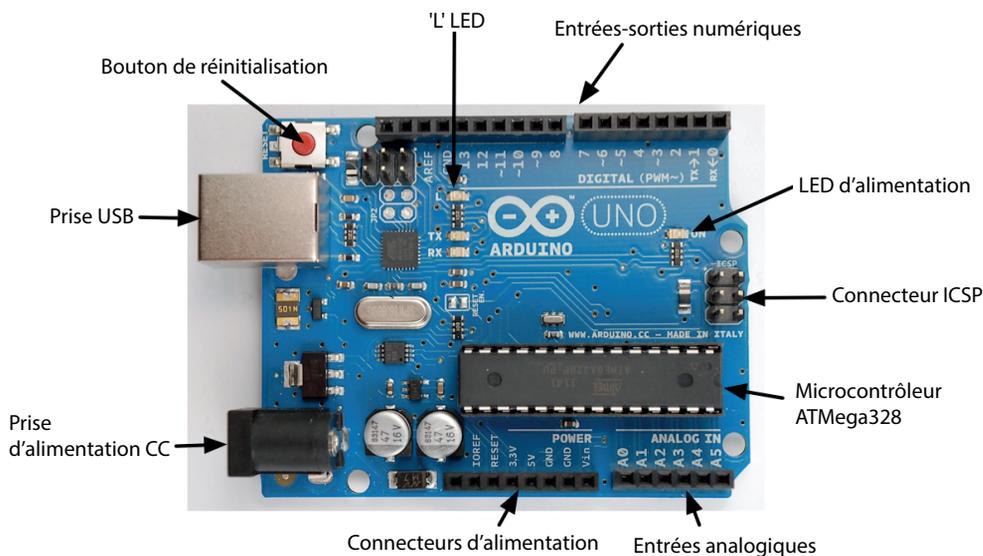


Figure 2-1. Un Arduino Uno R3

Le long des bords supérieurs et inférieurs de l'Arduino se trouvent des broches auxquelles il est possible de connecter des composants électroniques. Les entrées et sorties numériques, désignées par des chiffres compris entre 0 et 13, sont visibles le long du bord supérieur de la carte illustrée à la figure 2-1. Chaque broche peut être configurée dans vos programmes pour servir d'entrée ou de sortie. Si vous connectez un interrupteur à une entrée numérique, celle-ci indiquera s'il est enfoncé ou non. Ou bien, lorsque vous connectez une LED à une sortie numérique et que cette dernière passe de l'état haut à l'état bas, la LED s'allume. Une LED se trouve déjà sur la carte : il s'agit de la LED « L » qui est connectée à la broche numérique 13.

Au-dessous des broches d'E/S numériques, une LED d'alimentation indique simplement si la carte est sous tension. Le connecteur ICSP (*In-Circuit Serial Programming*) sert uniquement à la programmation avancée de carte Arduino qui ne passe pas par la connexion USB. La majorité des utilisateurs ne s'en servent jamais.

L'ATmega328 (le cerveau de l'Arduino) est un microcontrôleur à circuit intégré (IC, *Integrated Circuit*). Les 32 Ko de mémoire flash de la puce contiennent le programme nécessaire au fonctionnement de la carte.

Au-dessous de l'ATmega328, il y a une rangée de broches d'entrées analogiques étiquetées d'A0 à A5. Alors que les entrées numériques détectent seulement si un composant est allumé ou éteint, les entrées analogiques peuvent mesurer la tension reçue par la broche (à condition que cette tension soit comprise entre 0 et 5 V). Cette tension peut notamment provenir d'un capteur. Si vous n'avez pas assez d'entrées et sorties numériques, ces broches d'entrées analogiques peuvent aussi être utilisées comme des entrées numériques.

À côté de ces entrées se trouve une rangée de connecteurs d'alimentation qui peuvent être utilisés de façon alternative pour l'alimentation de l'Arduino. Ils peuvent également être utilisés pour alimenter d'autres composants électroniques contrôlés depuis la carte.

L'Arduino dispose également d'une prise d'alimentation en courant continu. Elle peut recevoir une tension comprise entre 7 et 12 V CC et utilise un régulateur de tension pour fournir la tension de 5 V nécessaire à l'Arduino. La platine acceptera automatiquement l'alimentation provenant de la prise USB ou du connecteur d'alimentation, selon le cas.

Installation de l'IDE Arduino

L'Arduino ne correspond pas tout à fait à l'image que l'on se fait d'un ordinateur. Il n'a pas de système d'exploitation et on n'y connecte pas de clavier, de moniteur ou de souris. Un seul programme peut y être exécuté à la fois et il faut préalablement le charger dans la mémoire flash de la carte à l'aide d'un ordinateur. L'Arduino peut être reprogrammé autant de fois que vous le souhaitez.

Pour pouvoir le programmer, vous devez installer l'IDE Arduino sur votre ordinateur. Ce logiciel peut fonctionner sur différentes plates-formes – Windows, Mac et Linux – et c'est l'une des raisons de la popularité de l'Arduino. De plus, l'IDE vous permet de programmer la carte via l'interface USB sans qu'il soit nécessaire d'utiliser de matériel particulier.

Pour installer l'IDE Arduino sur votre ordinateur, téléchargez le logiciel et suivez les instructions affichées sur le site web Arduino (<http://arduino.cc/en/Guide/HomePage>).

Sous Windows et Mac, il faut également installer des pilotes USB afin que l'IDE puisse communiquer avec la carte Arduino.

Une fois que tout est installé, démarrez l'IDE. La figure 2-2 présente la fenêtre de l'interface.

Comme son nom l'indique, le bouton **Téléverser** permet de transférer le sketch sur la carte Arduino. Avant que le sketch ne soit téléversé, l'IDE convertit le code de programmation textuel en code exécutable par l'Arduino. Si des erreurs s'y sont glissées, elles sont affichées dans la fenêtre de messagerie. Le bouton **Vérifier** remplit le même rôle, mais il ne lance pas le transfert du programme sur la carte.

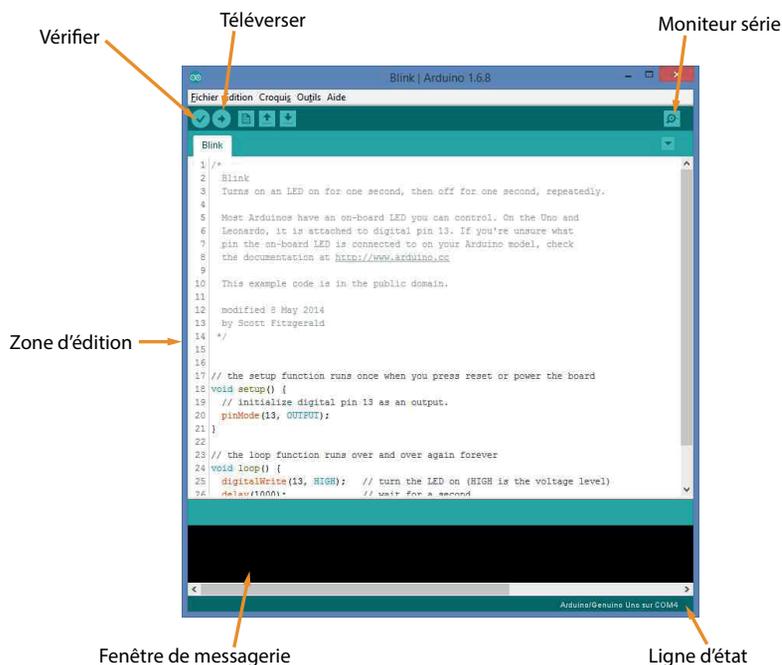


Figure 2-2. L'IDE Arduino

Le bouton **Moniteur série** ouvre la fenêtre du même nom qui sert à communiquer avec l'Arduino. Vous vous servirez de cette interface pour de nombreuses expériences proposées dans ce livre, car c'est un excellent moyen de transmettre des commandes à l'Arduino depuis votre ordinateur. Le moniteur série permet de communiquer dans les deux sens : vous pouvez envoyer des messages texte à l'Arduino et recevoir des réponses de la part de la carte.

La ligne d'état dans le bas de la fenêtre indique le type de carte Arduino et le port série utilisé pour sa programmation lorsque vous cliquez sur le bouton **Téléverser**. Le port illustré à la figure 2-2 (/dev/cu.usbmodem411) s'affiche généralement lorsque l'on utilise un ordinateur sous Mac ou Linux. Si vous utilisez un ordinateur Windows pour programmer l'Arduino, il s'agira du port COM4, ou des lettres COM suivies du numéro du port alloué par Windows à l'Arduino lorsque vous avez connecté la carte.

Enfin, la partie principale de l'IDE Arduino est la zone d'édition dans laquelle vous saisissez le code du programme que vous voulez téléverser sur l'Arduino.

Dans le monde de l'Arduino, les programmes se nomment des sketches (ou croquis dans la version française de l'IDE). Depuis le menu **Fichier** de l'IDE Arduino, vous pouvez **Ouvrir** et **Enregistrer** des sketches à la façon d'un document dans un traitement de texte. Le menu **Fichier** contient aussi un sous-menu **Exemples** à partir duquel vous pouvez charger des exemples de sketches fournis avec l'IDE.

Téléversement d'un sketch

Pour tester votre carte Arduino et pour vous assurer que l'IDE Arduino est correctement installé, commencez par ouvrir le sketch intitulé *Blink* qui se trouve dans **Fichier>Exemples>01. Basics**. (Le sketch *Blink* est affiché dans la zone d'édition illustrée à la figure 2-2.)

Utilisez un câble USB pour raccorder la carte à l'ordinateur qui servira à programmer l'Arduino. La LED d'alimentation de la carte s'allume et des LED clignent.

Lorsque l'Arduino est connecté, vous devez indiquer à l'IDE quel type de carte est programmé (Arduino Uno) et le port série auquel elle est connectée. Définissez le type de carte à l'aide de la commande **Outils>Type de carte>Arduino Uno**.

Définissez le port série en sélectionnant la commande **Outils>Port**. Si vous utilisez un ordinateur sous Windows, le sous-menu ne devrait pas comporter beaucoup d'options. Il est même possible que vous n'y trouviez que l'option COM4. Sur un ordinateur fonctionnant sous Mac ou Linux, vous devriez avoir le choix entre divers ports USB et vous aurez du mal à savoir lequel correspond à celui auquel l'Arduino est connecté. La liste doit proposer une entrée commençant par `dev/tty.usbmodemNNN`, NNN étant un nombre. À la figure 2-3, la carte Arduino connectée à mon Mac a été sélectionnée.

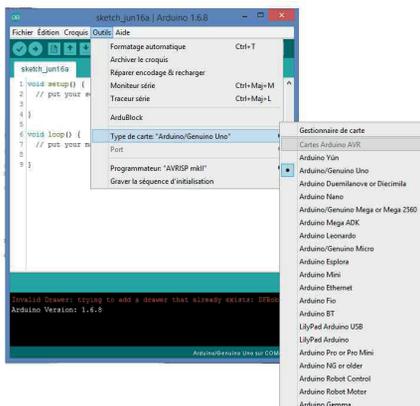


Figure 2-3. Sélection du port série Arduino/Genuino Uno

Si votre Arduino n'apparaît pas dans la liste, c'est généralement dû à un problème de pilote USB. Dans ce cas, essayez de réinstaller les pilotes.

Lorsque vous êtes prêt à transférer le sketch sur la carte, cliquez sur le bouton [Téléverser](#). Des messages s'affichent dans la partie inférieure de la fenêtre. Puis, au bout de quelques secondes, les LED intitulées « TX » et « RX » se mettent à clignoter pendant que le programme est transmis à la carte.

Si tout se déroule comme prévu, un message ressemblant à celui illustré à la figure 2-4 devrait s'afficher lorsque la transmission est terminée.

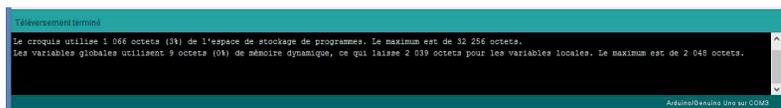


Figure 2-4. Transmission réussie

Ce message indique que le sketch a été transmis et qu'il utilise 1 066 octets sur les 32 256 octets disponibles.

Une fois la transmission terminée, vous devriez voir clignoter la LED « L » intégrée à la carte. C'est le signe que le sketch Blink fonctionne correctement.

Le code du livre

Tous les programmes de ce livre – que ce soit les sketches Arduino ou les programmes Python pour le Raspberry Pi – sont disponibles en anglais sur la page GitHub du livre (https://github.com/simonmonk/make_action). Pour importer ces fichiers sur votre ordinateur sous Mac, Linux ou Windows, cliquez sur le bouton [Clone or download](#) de la page GitHub, puis sur [Download ZIP](#). Les sketches Arduino en français sont disponibles sur www.serialmakers.com.

Enregistrez l'archive téléchargée sur le Bureau ou à un autre emplacement facile d'accès. Lorsque vous décompressez l'archive, vous obtenez un dossier intitulé `make_actionmaster`. Le code Arduino se trouve dans un dossier `arduino`. À l'intérieur, vous trouverez les deux sous-dossiers `experiments` et `projects`.

Chaque programme est rangé dans un dossier spécifique qui contient généralement un seul fichier qui correspond au programme proprement dit. Par exemple, à l'intérieur du dossier `experiments`, vous trouverez le dossier `ex_01_basic_motor_control` qui contient l'unique fichier `basic_motor_control.ino`. Si l'IDE Arduino est déjà installé sur votre ordinateur, vous pouvez ouvrir directement le fichier dans l'environnement de programmation.

Une autre façon d'accéder à ces sketches est de copier les dossiers d'expériences et de projets dans votre sketchbook Arduino. Il s'agit du dossier intitulé `Arduino`, qui se trouve dans votre dossier de documents habituel (Mes documents sous Windows, ou Documents sous Mac).

Si les fichiers sont copiés dans le dossier sketchbook, vous pourrez y accéder à l'aide de la commande [Fichier>Carnet de croquis](#) dans l'IDE Arduino.

Guide de programmation

Nous allons passer en revue les principales commandes pour vous aider à mieux comprendre les sketches utilisés dans le livre.

Setup et Loop

Tous les sketches Arduino doivent contenir une fonction `setup()` et une fonction `loop()` (les fonctions sont des blocs de code de programme qui réalisent une tâche). Pour comprendre le fonctionnement de `setup()` et `loop()`, nous allons disséquer l'exemple du sketch *Blink* que nous avons transmis à la carte Arduino :

```
int led = 13;
// la fonction d'initialisation est exécutée une fois après l'action sur le bouton de
réinitialisation :
void setup() {
  // initialise la broche numérique comme sortie.
  pinMode(led, OUTPUT);
}

// la fonction de boucle est exécutée en continu :
void loop() {
  digitalWrite(led, HIGH); // allume la LED (le niveau de tension est HIGH)
  delay(1000);             // pause 1 seconde
  digitalWrite(led, LOW);  // éteint la LED (le niveau de tension est LOW)
  delay(1000);            // pause 1 seconde
}
```

Notez qu'une grande partie du texte qui a été traduit en français pour les besoins de l'explication est précédé des signes `//`. Cela signifie que le texte qui suit `//` jusqu'à la fin de la ligne doit être considéré comme un commentaire. Ce n'est pas un code de programme, il sert simplement à expliquer au lecteur ce qu'il se passe dans le programme.

Les lignes de code à l'intérieur de la fonction `setup()` ne sont exécutées qu'une seule fois (ou, plus précisément, chaque fois que la carte Arduino est mise sous tension ou que vous appuyez sur le bouton de réinitialisation). Par conséquent, on utilise la fonction `setup()` pour effectuer toutes les choses qui ne doivent être faites qu'une seule fois au démarrage du programme. Dans le cas de *Blink*, cela consiste simplement à préciser que la broche LED est définie comme sortie.

Les commandes à l'intérieur de la fonction `loop()` sont exécutées en boucle – dès que la dernière ligne de commande `loop()` a été exécutée, elle recommence à la première ligne.

Je n'ai pas expliqué le rôle des commandes à l'intérieur de `setup()` et de `loop()`, mais j'y arrive bientôt.

Variables

Les variables permettent de nommer des valeurs. La première ligne du sketch *Blink* (si l'on ne tient pas compte des commentaires) est la suivante :

```
int led = 13;
```

Cela définit une variable nommée `led` en lui donnant une valeur initiale de 13. La valeur 13 a été choisie, car elle correspond au nom de la broche à laquelle la LED L est connectée et `int` désigne le type de variable. Le mot `int` est l'abréviation d'*integer* (« nombre entier », en anglais).

Même s'il n'est pas nécessaire d'utiliser un nom variable pour toutes les broches utilisées, il est judicieux de le faire, car il est alors plus facile de savoir à quoi sert la broche. En outre, si vous voulez utiliser une autre broche, il vous suffit de changer la valeur de la variable à un seul endroit.

Peut-être avez-vous remarqué que dans les sketches qui accompagnent ce livre, lorsque l'on déclare des variables comme celle-ci qui définit la broche utilisée, la ligne commence par `const` :

```
const int led = 13;
```

Le mot-clé `const` indique à l'IDE Arduino que la variable n'en est pas vraiment une, mais plutôt une constante ; en d'autres termes, sa valeur demeurera 13. Cela aboutit à des sketches légèrement plus petits qui sont exécutés plus rapidement. C'est généralement considéré comme une bonne habitude à prendre.

Sorties numériques

Le sketch *Blink* offre un bon exemple de sortie numérique. La broche 13 est configurée comme sortie dans la fonction `setup()` par cette ligne (la variable `led` ayant préalablement été définie sur 13) :

```
pinMode(led, OUTPUT);
```

Cette commande se situe dans la fonction `setup()`, car la configuration n'a besoin d'être effectuée qu'une seule fois. Lorsque la broche a été définie comme sortie, elle le restera jusqu'à ce qu'une autre instruction la configure différemment.

Pour clignoter, la LED doit être allumée et éteinte plusieurs fois. Par conséquent, le code est le suivant :

```
digitalWrite(led, HIGH);    // allume la LED (le niveau de tension est HIGH)
delay(1000);                // pause 1 seconde
digitalWrite(led, LOW);     // éteint la LED (le niveau de tension est LOW)
delay(1000);                // pause 1 seconde
```

La fonction `digitalWrite()` a deux paramètres (entre parenthèses et séparés par une virgule). Le premier paramètre est la broche Arduino qui sera écrite et le second paramètre est la valeur qui y sera écrite. Par conséquent, la valeur `HIGH` appliquera une tension de 5 V à la sortie (la LED s'allume) et la valeur `LOW` met la broche hors tension, à 0 V (la LED s'éteint).

La fonction `delay()` interrompt le programme pendant la durée (en millisecondes) précisée comme paramètre. Il y a 1 000 millisecondes dans 1 seconde, donc chacune des fonctions `delay()` présentées interrompt le programme pendant 1 seconde.

Dans l'expérience « Pilotage d'une LED », décrite au chapitre 4, au lieu de la LED de la carte, vous utiliserez une sortie numérique connectée à une LED externe que vous ferez clignoter.

Entrées numériques

Dans ce livre, nous nous intéressons davantage aux sorties qu'aux entrées, donc nous utilisons surtout la fonction `digitalWrite()`. Les entrées numériques permettent de connecter des boutons et des capteurs à une carte Arduino.

Vous pouvez définir une broche Arduino comme entrée numérique à l'aide de la fonction `pinMode()`. L'exemple suivant définit la broche 7 comme entrée. Vous pouvez évidemment aussi utiliser un nom de variable à la place du chiffre 7.

Comme pour une sortie, la broche d'entrée est définie dans la fonction `setup()`, car il est rare de changer le mode d'une broche en cours d'exécution du sketch :

```
pinMode(7, INPUT)
```

Une fois la broche définie comme entrée, vous pouvez la lire pour déterminer si la tension y est plus proche de 5 V (HIGH) ou de 0 V (LOW). Dans cet exemple, la LED s'allume si l'entrée est LOW au moment de la lecture (une fois allumée, la LED le reste, car le code ne précise pas qu'il faut l'éteindre) :

```
void loop()
{
  if (digitalRead(7) == HIGH)
  {
    digitalWrite(led, LOW)
  }
}
```

À partir de là, le code se complique. Nous le parcourons donc ligne par ligne.

À la deuxième ligne, nous trouvons le symbole `{`. Il arrive qu'il soit placé sur la même ligne que `loop()` ou sur la ligne suivante. Ce n'est qu'une question de préférence personnelle et cela n'a aucune incidence sur l'exécution du code. Le symbole `{` signale le début d'un bloc de code qui se termine par le symbole `}` correspondant. C'est une façon de regrouper toutes les lignes de code qui appartiennent à la fonction `loop`.

La première de ces lignes utilise l'instruction `if` qui est immédiatement suivie d'une condition. Dans ce cas, la condition est `(digitalRead(7) == HIGH)`. Le double signe égal (`==`) permet de comparer les deux valeurs qui figurent de chaque côté. Donc, ici, si la broche 7 est égale à HIGH, alors le bloc de code délimité par `{ et }` qui figure après `if` est exécuté. Sinon, il ne l'est pas. L'alignement des accolades permet d'associer plus facilement les paires.

Nous avons déjà examiné le code qui est exécuté si la condition est remplie. Il s'agit de la fonction `digitalWrite()` utilisée pour allumer la LED.

Dans cet exemple, on suppose que l'entrée numérique est strictement HIGH ou LOW. Si vous avez connecté un interrupteur à une entrée numérique, il ne pourra que fermer la connexion. D'ordinaire, il faut connecter l'entrée numérique à GND (0 V). Si la liaison établie par l'interrupteur est ouverte, on dira que l'entrée numérique est flottante. Au sens strict, elle n'est pas raccordée électriquement à un composant. L'entrée percevra un bruit électrique et elle alternera souvent entre l'état haut et l'état bas. Pour l'éviter, on utilise généralement une résistance pull-up, comme illustré à la figure 2-5.

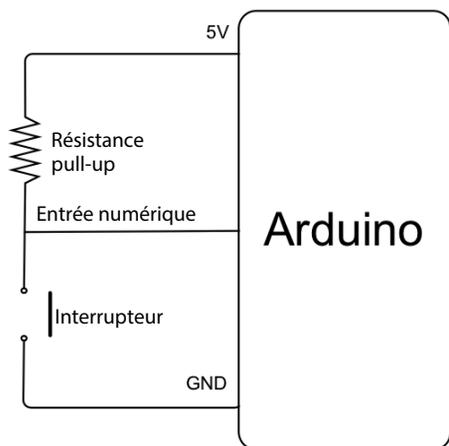


Figure 2-5. Utilisation d'une résistance pull-up avec une entrée numérique

Quand l'interrupteur est ouvert (comme illustré à la figure 2-5), la résistance tire la broche d'entrée à 5 V. Quand l'interrupteur est fermé, le tirage faible de l'entrée est contré puisque l'interrupteur relie l'entrée numérique à la masse (GND).

Même si vous pouvez effectivement utiliser une résistance pull-up, sachez que les entrées de la carte Arduino intègrent déjà des résistances pull-up internes d'environ 40 kΩ qui sont activées lorsque le mode de la broche est défini sur `INPUT_PULLUP` au lieu de `INPUT`. Le code suivant montre comment définir le mode d'une entrée numérique connectée à un interrupteur sans employer de résistance pull-up externe comme illustré à la figure 2-5 :

```
pinMode(switchPin, INPUT_PULLUP);
```

Entrées analogiques

Les entrées analogiques étiquetées de A0 à A5 peuvent mesurer une tension comprise entre 0 et 5 V. Contrairement aux entrées et sorties numériques, il n'est pas nécessaire d'utiliser `pinMode()` dans la fonction `setup` pour une entrée analogique.