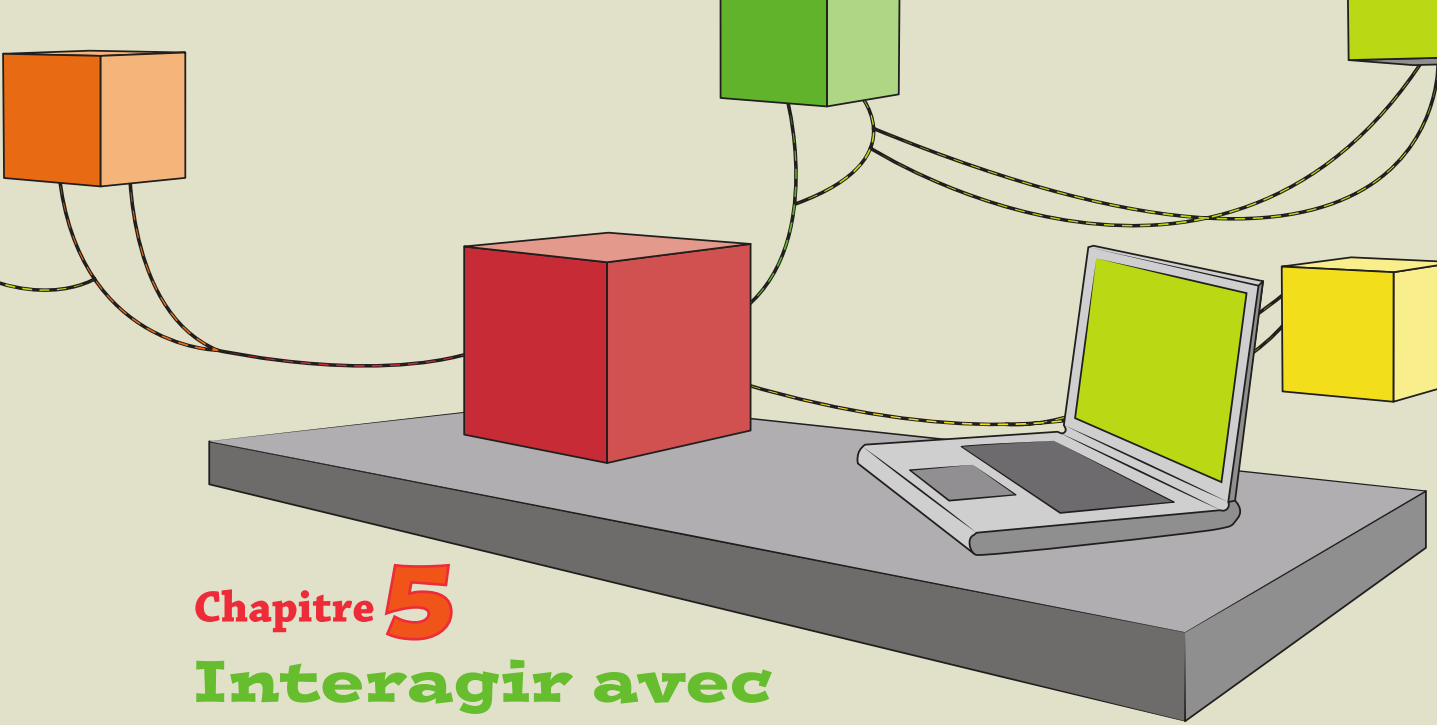

Chapitre 5
Interagir avec des circuits électroniques

LORSQUE TU JOUES à Minecraft, même via l'interface de programmation, tu évolues dans un monde virtuel. L'unique moyen que tu as d'interagir avec le jeu est d'utiliser ton clavier et ta souris pour manipuler les commandes définies par les programmeurs qui ont conçu le jeu.

Mais il existe une autre manière d'interagir avec Minecraft. Pour cela, il faut briser les barrières du jeu bac à sable et le relier au monde physique. Tu découvriras alors que la frontière qui sépare le monde virtuel du monde réel peut vite s'effacer, rendant ton jeu bien plus fertile et captivant.

Dans ce chapitre, tu vas apprendre à connecter Minecraft à de petits circuits électroniques en te servant de l'API. Tu vas commencer par faire clignoter une diode chaque fois que tu franchiras la porte de ta maison. Puis tu vas ajouter à ton dispositif ce qu'on appelle communément un afficheur 7 segments, qui te permettra d'activer un compte à rebours et bien d'autres informations sur ton jeu. Tu ajouteras ensuite un bouton-poussoir pour provoquer toutes sortes d'actions utiles dans Minecraft. Pour finir, tu assembleras le tout afin de créer un détonateur rouge entièrement fonctionnel doté d'un compte à rebours, comme celui que tu peux voir sur la figure 5-1. Grâce à cela, tu n'auras plus aucun mal à libérer de l'espace dans Minecraft et tes amis seront si ébahis qu'ils s'empresseront de te demander de leur en confectionner un !

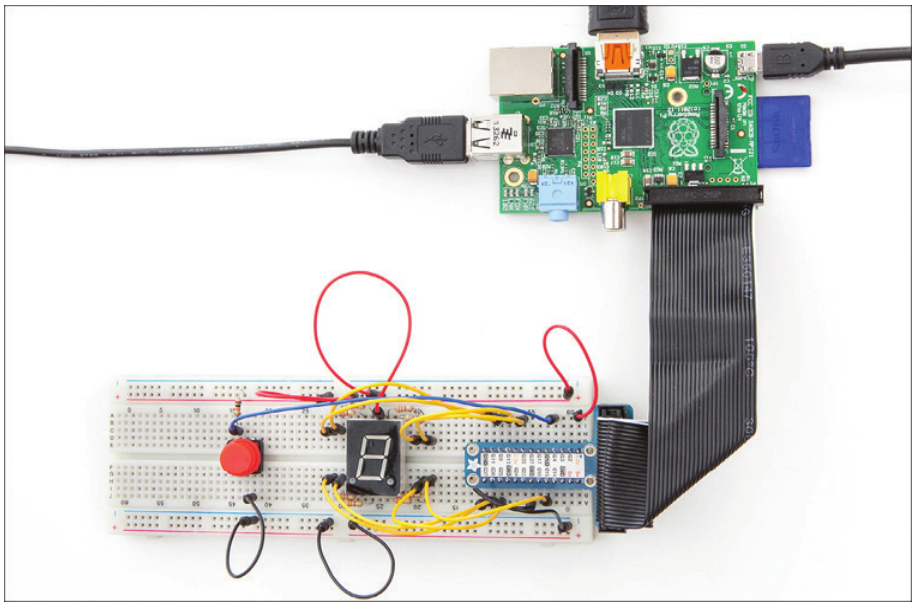

FIGURE 5-1 Un détonateur à bouton-poussoir connecté à un Raspberry Pi

Ce dont tu as besoin dans ce chapitre

Voici une liste complète de tous les éléments qui te serviront à construire tes circuits dans ce chapitre. Tu auras notamment besoin de composants électroniques, mais aussi d'autres outils te permettant de les connecter à ton ordinateur. Tous les composants requis sont exposés dans les figures 5-2 et 5-3 :

- 1 plaque d'essais ;
- au moins 4 boutons-poussoirs ;
- au moins 1 LED de la couleur de ton choix (mais plus tu auras de couleurs, plus l'exercice sera amusant !) ;
- 1 afficheur 7 segments ;
- au moins 10 résistances de 330 ohms à utiliser avec tes LED et ton afficheur ;
- au moins 4 résistances de 10 Kohms à utiliser avec tes boutons ;
- au moins 20 fils-jarretières avec connecteurs, ou des petits fils à âme rigide et une pince à dénuder ;
- 1 petit coupleur 2 piles AA ou AAA.

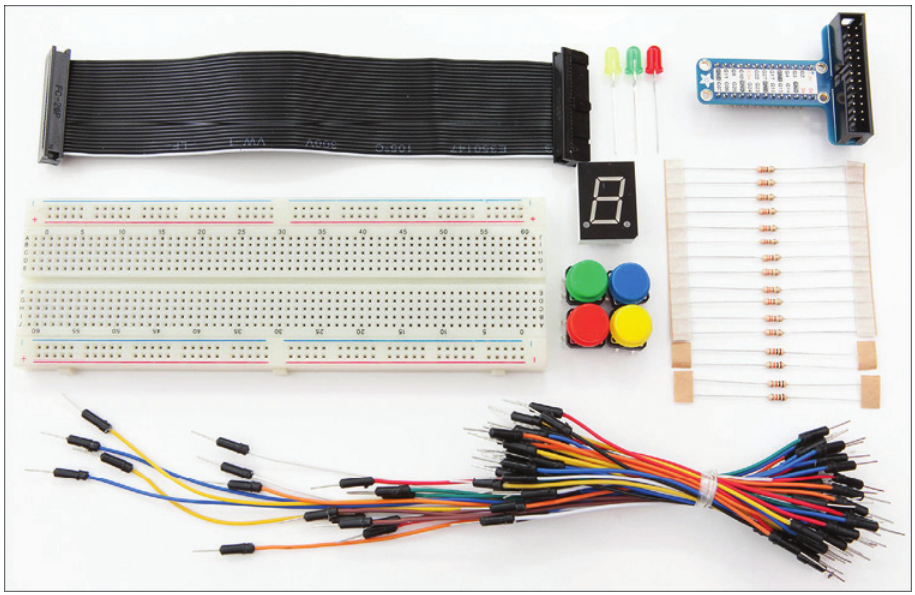

FIGURE 5-2 Les composants requis pour un Raspberry Pi

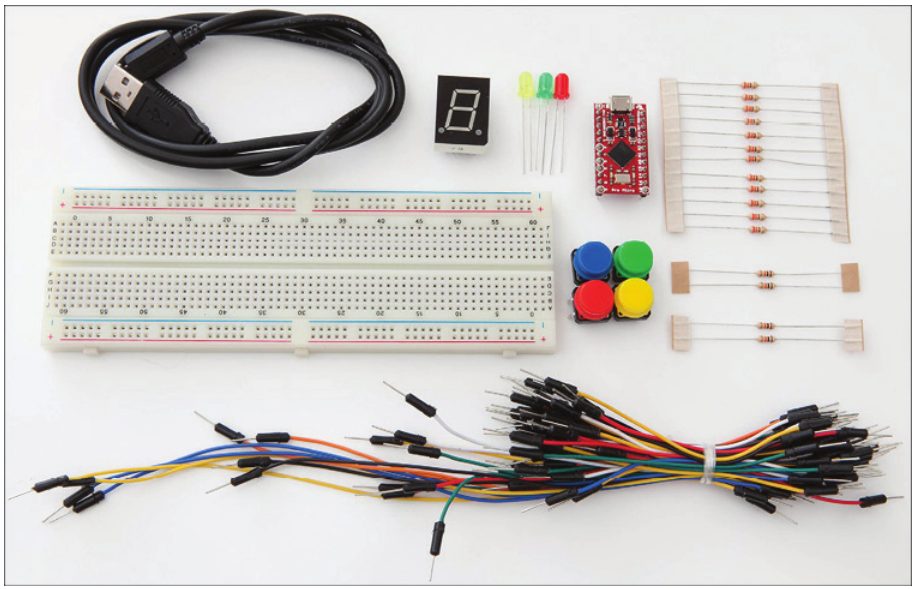

FIGURE 5-3 Les composants requis pour un PC ou un Mac

Sur Internet, les fournisseurs de composants électroniques ne manquent pas. Tu peux acheter les composants sur plusieurs sites, tels que : `www.selectronic.fr`, `www.conrad.fr`, `www.electronique-diffusion.fr`, ainsi que dans la plupart des magasins d'électronique comme Maplin Electronics (`www.maplin.co.uk`). Certains magasins vendent les

composants par sachets, surtout en ce qui concerne les LED et les résistances, mais tu pourras aussi trouver des sacs de fils-jarretières avec connecteurs. Si toutefois tu n'en trouves pas, tu peux toujours acheter du fil à âme rigide au mètre et une pince à dénuder pour créer tes propres fils-jarretières. Dans la suite de ce chapitre, nous parlerons de fils-jarretières, mais les instructions sont les mêmes si tu utilises des fils à âme rigide.

Sur Raspberry Pi

Utilise les connecteurs des fils-jarretières pour les rattacher directement aux broches de ton Raspberry Pi puis branche-les à ta plaque d'essais. Cette opération requiert beaucoup de minutie et compter toutes les broches peut s'avérer fastidieux. C'est pourquoi, dans ce chapitre, j'ai préféré utiliser un Pi-T-Cobbler d'Adafruit. Il se compose d'un câble plat qui part du Raspberry Pi et se branche directement à la plaque d'essais par des broches numérotées. Si tu sais souder, tu peux choisir un kit Pi-T-Cobbler et le souder toi-même, ce qui n'est pas plus mal, car cela te permettra d'économiser un peu d'argent. La figure 5-2 t'indique tous les composants dont tu as besoin pour réaliser les exercices de ce chapitre avec un Raspberry Pi.

Pour te procurer le Pi-T-Cobbler d'Adafruit, rends-toi directement sur les sites `www.adafruit.com` ou `www.lextronic.fr` et choisis le Pi-T-Cobbler pré-assemblé, sauf si tu souhaites t'amuser à souder les composants toi-même.

Au moment où nous avons rédigé ce livre, Raspberry Pi publiait son modèle B+. Tous les chapitres de ce livre sont donc écrits pour le Raspberry Pi B+, mais nous te conseillons tout de même d'acheter un fil supplémentaire avec ton Pi-T-Cobbler pour que tu puisses le connecter correctement au Raspberry Pi B+ à 40 broches. Le câble dont tu as besoin se termine par un embout à 40 broches qui se branche au Raspberry Pi B+. De l'autre côté, il possède un embout à 26 broches, que tu connectes au Pi-T-Cobbler. Pour te le procurer, tu peux te rendre sur `www.lextronic.fr` et l'acheter en même temps que le Pi-T-Cobbler.

Sur PC et Mac

Contrairement aux Raspberry Pi, les PC et les Mac ne sont pas dotés de broches intégrées. C'est pourquoi j'ai créé une petite plate-forme élaborée à partir d'un circuit Arduino que tu pourras utiliser sur tes machines. Pour effectuer les exercices de ce chapitre, tu peux utiliser n'importe quel circuit Arduino, à condition d'installer mon logiciel open source spécial qui permet de faire fonctionner la plate-forme comme s'il s'agissait d'un Raspberry Pi avec ses broches d'entrée et de sortie. Pour en savoir plus sur Arduino, rends-toi sur le site : `http://arduino.cc`. Dans ce chapitre, tu utiliseras un circuit Arduino Pro Micro. Tu auras également besoin d'un câble USB pour connecter le dispositif à ton ordinateur.

Pour acheter le circuit Arduino Pro Micro, connecte-toi à : https://www.sparkfun.com/products/12587 ou sur une autre boutique en ligne spécialisée. Le numéro de référence du produit est : DEV-12587. Mais pour qu'il soit complètement opérationnel, tu devras effectuer quelques soudures et le programmer à l'aide du logiciel conçu pour le faire fonctionner comme une plate-forme d'entrée et de sortie. Pour gagner du temps et t'éviter le travail de soudure, tu peux directement acheter la plate-forme soudée, testée et configurée avec le logiciel en te rendant sur la page (en anglais) : http://skpang.co.uk/catalog/pro-micro-33v8mhz-with-headers-and-anyio-firmware-p-1327.html. Le numéro de référence du produit est : AR-PROMICRO-ANYIO. Tu n'auras plus qu'à brancher le circuit à ta plaque d'essais et à le connecter à ton PC ou à ton Mac à l'aide d'un câble USB, pour le faire fonctionner comme une plate-forme d'entrée et de sortie. La figure 5-3 t'indique tous les composants dont tu as besoin pour réaliser les exercices de ce chapitre sur un PC ou un Mac.

Si tu as un Mac, il doit au moins disposer de la version 10.6 d'OS X pour être compatible avec le circuit Arduino.

Pour réaliser les exercices de ce chapitre, tu disposes d'un dossier de fichiers sources appartenant à un logiciel. Tous ces fichiers se trouvent dans le kit de démarrage que tu as téléchargé sur le site Internet du livre. Un peu plus loin, nous fournissons également une liste de sites Internet sur lesquels toi et tes amis pourrez télécharger la dernière version des fichiers sources du logiciel, si jamais tu souhaites les utiliser pour d'autres projets et faire quelques expérimentations en dehors de Minecraft.

Une fois que tu te seras procuré tous les éléments dont tu as besoin, il sera temps de passer aux choses sérieuses et de commencer à construire ton tout premier circuit électronique !

Créer un prototype à l'aide d'une plaque d'essais

Lorsqu'un ingénieur électronique conçoit un circuit pour un nouveau produit, il crée tout d'abord ce qu'on appelle un **prototype** du circuit, qu'il consolide ensuite pour le reproduire par milliers.

Un **prototype** est le modèle réduit d'un objet conçu pour réaliser des tests de fonctionnement. Il est bien plus simple, en effet, de résoudre les problèmes sur un seul objet plutôt que sur mille ou un million !

Sur Internet, tu trouveras de superbes photos des tout premiers prototypes du Raspberry Pi qu'Eben Upton a conçus en 2006, marquant la création de la Fondation Raspberry Pi. La plupart des circuits électroniques naissent sous la forme d'un prototype similaire et subissent de nombreuses modifications avant de ressembler au circuit que tu as sous les yeux. Pour voir les photos, rends-toi sur : `http://www.raspberrypi.org/raspberry-pi-2006-edition/`.

Sur la plupart des circuits électroniques, on soude les composants ensemble pour renforcer la connexion électrique et garantir un fonctionnement sur le long terme. Mais si les circuits soudés présentent l'avantage d'être plus solides, ils ne s'avèrent pas très pratiques lorsque tu souhaites te lancer dans des expérimentations impliquant la manipulation de plusieurs composants et la conception de plusieurs montages électroniques à tester, car tu passerais bien trop de temps à dessouder et ressouder tes composants. Pour éviter ce genre de problèmes, les ingénieurs utilisent généralement ce qu'on appelle une **plaque d'essais** sans soudure.

Une **plaque d'essais** est un outil réutilisable qui te permet de monter des circuits sans avoir recours à la soudure. Ces plaques contiennent plusieurs rangées de trous dans lesquels tu peux brancher des fils, des câbles ou des composants afin de créer des circuits. Les deux bandes de trous supérieure et inférieure de la plaque émettent un courant électrique. Normalement, elles sont encadrées d'une ligne rouge, désignant le côté positif de 3,3 volts (ce côté contient parfois l'indication « + » ou « VCC »), et d'une ligne noire ou bleue, désignant le côté négatif de 0 volt (parfois marqué d'un « - » ou de l'abréviation « GND »).

Sur la figure 5-4, tu peux voir deux exemples de plaques d'essais sans soudure, à gauche avec des composants branchés pour créer un circuit et à droite, l'intérieur de la plaque. Si tu regardes de plus près, tu peux voir les bandes de métal qui servent à réaliser les connexions. Il est très important de savoir comment une plaque d'essais fonctionne, car lorsque tu brancheras des composants dans les trous, tu sauras ainsi précisément comment les connexions interagiront dans tes circuits électriques.

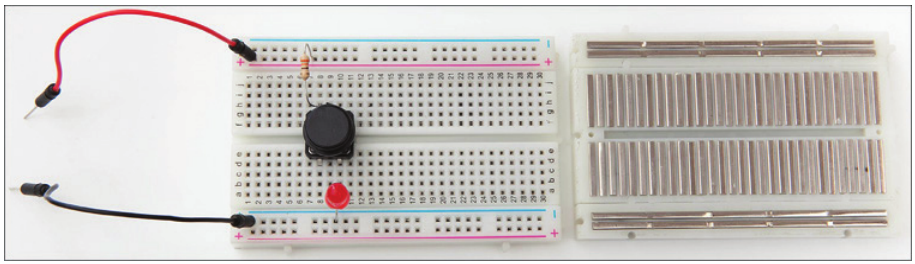

FIGURE 5-4 À gauche, une plaque d'essais ; à droite, l'intérieur d'une plaque d'essais avec ses connexions internes

Élaborer un circuit permettant d'allumer une LED

La figure 5-4 présente un circuit simple qui permet de tester des composants facilement. Pour faire fonctionner n'importe quel circuit, on doit le placer sous **tension**. Pour cela, tu dois connecter une pile 3 V aux bandes supérieure et inférieure de la plaque d'essais, appelées également *power-rails* (en anglais). Ce sont des barrettes longues qui assurent une connexion sur toute la longueur de la plaque et qui sont bien pratiques, car elles permettent d'apporter du courant à plusieurs composants à la fois. L'élément rouge que tu peux voir sur l'image est une **diode électroluminescente** (**LED**) ; une petite lumière que tu apprendras à connaître d'ici peu. Le composant aux rayures de couleur est une **résistance** et le dispositif noir situé en dessous est un bouton-poussoir. Si tu suis les câbles du circuit depuis la batterie jusqu'à la fin, tu retomberas sur la batterie. Cela veut dire qu'il s'agit d'un circuit électrique complet. Et lorsque tu appuies sur le bouton-poussoir, cela libère du **courant** électrique qui active la LED.

La **tension**, ou voltage, est la différence d'énergie électrique qui existe entre deux points d'un même circuit. Elle est équivalente à la pression qu'exerce l'eau dans les tuyaux de canalisation ; c'est cette même pression qui pousse le courant à s'écouler dans un circuit. Elle se mesure en volts (notés V).

Une **diode électroluminescente** (**LED**) s'allume lorsqu'un courant électrique la traverse. Une LED n'autorise le courant à passer que dans un seul sens et ne s'allume que si le courant passe dans le bon sens. Il existe des LED de toutes les couleurs. Elles sont dotées d'une patte courte (la cathode) et d'une patte longue (l'anode), qui t'aident à placer la diode dans le bon sens sur un circuit afin que le courant puisse la traverser.

Une **résistance** est un composant électrique qui permet de résister au courant dans un circuit. Par exemple, si le courant est trop puissant, cela peut endommager ta LED. Pour éviter cela, tu vas donc placer une résistance d'une valeur suffisamment élevée sur ton circuit pour réguler le courant qui passera à travers ta diode. La valeur d'une résistance se calcule en ohms, qu'on signale à l'aide du symbole oméga, Ω. Pour protéger ta diode, tu dois choisir la bonne valeur de résistance. Celle-ci est indiquée sur ton composant par des rayures de couleur répondant à un code précis, qui se lit de gauche à droite. Pour en savoir plus sur le code couleur des résistances, tu peux consulter la page : `https://fr.wikipedia.org/wiki/CEI_60757`.

PERCER LES SECRETS DU CODE

Sur le circuit de la figure 5-4, une résistance a été utilisée. Une LED est un composant délicat qui ne peut recevoir qu'un courant de faible intensité, ne dépassant généralement pas les 20 mA (soit 0,02 A, ce qui est plutôt faible). Si tu connectes une LED directement à une pile, le courant qui circulera dans le circuit sera bien trop fort et endommagera ta LED. Pour éviter cela, on ajoute une résistance entre la pile et la LED pour limiter le courant qui passe dans le circuit. La valeur de la résistance est signalée par les rayures colorées qui sont peintes sur son corps.

Si tu utilises des piles plus puissantes, tu dois alors placer une résistance d'une valeur plus élevée sur ton circuit pour réduire l'intensité du courant. Pour ce projet, j'ai calculé les valeurs de résistance à ta place, mais il existe une multitude de sites Internet qui t'aident à calculer la valeur des résistances en fonction de la puissance des piles, par exemple : `www.ohmslawcalculator.com/led_resistor_calculator.php`.

Le **courant** est le taux d'énergie électrique qui circule au-delà d'un point dans un circuit. C'est l'équivalent du débit d'eau dans un tuyau. Il se mesure en ampères (notés A). Pour les courants plus faibles, on utilisera le milliampère (mA) comme unité de mesure. Il existe plusieurs théories intéressantes au sujet du sens de circulation du courant dans un circuit. Pour les connaître, rends-toi sur : `www.allaboutcircuits.com/vol_1/chpt_1/7.html` (en anglais).

DÉFI

Essaye de construire le circuit de la figure 5-4 avec tes composants électroniques puis appuie sur le bouton pour allumer la diode. Tu auras besoin d'un petit coupleur de piles AA ou AAA. Fais attention à bien placer la LED dans le bon sens sur le circuit. Si elle ne s'allume pas, vérifie tes branchements et repositionne-la.

Connecter des composants électroniques à ton ordinateur

Dans ce chapitre, tu vas connecter une LED à ton ordinateur et l'activer à l'aide d'un programme Python. Or, pour prendre le contrôle de composants électroniques depuis ton ordinateur, tu dois d'abord trouver un moyen de connecter ton ordinateur aux circuits qui te permettront de les activer. Pour ce faire, tu as besoin de **broches GPIO** (**General Purpose Input/Output**).

Les **broches GPIO** permettent de recevoir ou d'émettre des signaux vers ou depuis le processeur central d'un système informatique. Elles ont été conçues pour répondre à un usage général et les concepteurs de circuits peuvent leur assigner une fonction précise, telle que le contrôle de circuits externes.

Certains ordinateurs, comme les Raspberry Pi, disposent d'un port GPIO intégré. Une broche GPIO peut être utilisée en mode entrée pour détecter la tension d'un dispositif externe ou en mode sortie pour réguler sa tension. La tension de ton Raspberry Pi ou de ton circuit Arduino s'élève à 3,3 volts, ce qui est plutôt bas, mais suffisant pour allumer une LED ou détecter un bouton-poussoir. À titre de comparaison, l'électricité domestique en France est limitée à 240 volts, ce qui est très élevé (et dangereux). Quant aux pylônes électriques disséminés un peu partout dans le pays, ils transportent un courant de pas moins de 230 000 volts ! Rien à voir, donc, avec le courant qui circule dans un circuit informatique.

Un ordinateur est une machine numérique, c'est-à-dire qu'il repose sur un système de numéros. Aujourd'hui, tous nos ordinateurs sont numériques et utilisent les chiffres 0 et 1. Lorsque tu utilises des broches GPIO sur un ordinateur, le 0 indique généralement une tension nulle, soit 0 volt, tandis que le 1 indique la présence de tension. Dans notre cas, cette dernière s'élève à 3,3 volts. Lorsque tu souhaites détecter la tension d'un dispositif externe à l'aide d'un port GPIO, l'ordinateur ne peut indiquer cette tension qu'avec les chiffres 0 ou 1. Une tension proche de 0 volt sera donc signalée par un 0,

tandis qu'une tension proche de 3,3 volts sera signalée par un 1. Toute tension qui se situe entre ces deux valeurs sera signalée par un 0 ou par un 1 dans ton ordinateur, bien qu'elle se situe dans une zone grise.

Paramétrer un PC ou un Mac pour contrôler des circuits électroniques

Si tu as un Raspberry Pi, tu peux ignorer cette section et te rendre directement à la suivante, intitulée « Contrôler une LED ». En effet, un Raspberry Pi est déjà doté de broches GPIO intégrées, ce qui en fait un excellent outil pour contrôler des circuits électroniques.

Les PC et les Mac, quant à eux, ne possèdent pas de broches GPIO intégrées. Tu dois donc les ajouter toi-même à l'aide d'un circuit externe. Il existe une multitude de manières de le faire, mais pour ce qui nous concerne, nous avons choisi d'utiliser un circuit Arduino préconfiguré. Tu n'apprendras rien sur Arduino dans ce livre, mais si tu souhaites en savoir plus sur le petit ordinateur, tu peux consulter l'annexe A, qui te renverra vers quelques sites d'intérêt.

Dans les instructions qui vont suivre, je considère que tu as acquis un circuit présoudé et préconfiguré, comme indiqué au début du chapitre. Si tu as choisi d'utiliser une plaque vierge pour la souder et la configurer toi-même, tu devras lire les instructions de ce guide pratique en anglais : `https://www.sparkfun.com/products/12587`. Puis tu devras installer le logiciel GPIO open source que j'ai spécialement conçu pour cela. Ce petit logiciel, couplé au module Python *anyio* que j'ai inclus dans le kit de démarrage, te permettra de faire fonctionner tes broches GPIO exactement comme sur un ordinateur Raspberry Pi.

Pour exécuter les programmes de ce chapitre, tu devras alors penser à les modifier légèrement afin qu'ils prennent en compte ton cas de figure. Si tu dois le programmer toi-même, le logiciel à installer dans le circuit Arduino se trouve dans le dossier `anyio/arduino/firmware` de ton kit de démarrage. Par ailleurs, si tu souhaites te procurer la dernière version du package anyio pour pouvoir l'utiliser avec tes amis dans le cadre d'autres projets, tu peux toujours te rendre sur ma page GitHub : `https://github.com/whaleygeek/anyio`.

Configurer les pilotes

Lorsque tu connectes un périphérique à ton ordinateur, celui-ci doit disposer d'un logiciel spécial qui lui permet de communiquer avec le périphérique en question. Ce logiciel est en fait un pilote informatique. Il t'est peut-être déjà arrivé d'utiliser un pilote

informatique sur ton ordinateur, par exemple si tu as installé une imprimante ou un autre périphérique. C'est exactement la même opération que tu vas réaliser ici.

1. Branche le câble USB sur le circuit Arduino et l'autre extrémité sur ton ordinateur.
2. Ensuite, suis les instructions qui s'affichent à l'écran (comme elles varient d'une version à une autre du système d'exploitation, je ne détaillerai malheureusement pas les étapes ici) et procède à l'installation des pilotes pour utiliser le Pro Micro sur ton ordinateur.

Sur Mac

Tu vas voir s'afficher un message t'indiquant qu'un nouveau clavier a été ajouté. Clique simplement sur la croix rouge pour le faire disparaître. Et voilà ! L'installation de ton circuit est terminée.

Sur PC

Windows va te demander de télécharger un pilote. Tu n'as pas besoin de le faire, car Windows dispose déjà de tous les pilotes nécessaires. En revanche, comme ton système d'exploitation ne reconnaît pas toujours les périphériques inconnus, j'ai fourni un fichier dans le kit de démarrage pour aider Windows à détecter le périphérique. Ce fichier s'appelle `ProMicro.inf` et il se trouve dans le dossier `anyio/arduino/firmware` du kit de démarrage, à l'intérieur du dossier `MyAdventures`. Recherche le fichier `ProMicro.inf` en question et clique sur `OK`. Windows devrait maintenant être en mesure de reconnaître le périphérique en tant que port USB.

> Si malgré tout ton ordinateur ne reconnaît pas le périphérique, consulte le guide pratique du site anglais SparkFun. Plutôt bien détaillé, il est mis à jour à chaque fois qu'une nouvelle version du système d'exploitation est déployée : `https://www.sparkfun.com/products/12587`.
>
> 
>

Obtenir le numéro du port série

Le circuit Arduino que tu viens de connecter à ton ordinateur est reconnu comme un port série. Sans t'attarder sur ce que cela signifie, tu dois t'assurer que le kit anyio choisisse le bon port avant de l'utiliser comme extension GPIO.

Pour t'aider, j'ai rédigé un petit programme d'aide que tu trouveras également dans le kit anyio. La figure 5-5 te montre à quoi il ressemble.

Suis donc ces indications pour choisir le bon port.

1. Ouvre IDLE et clique sur `File>Open` pour rechercher un programme Python.
2. Dans le dossier `MyAdventures`, ouvre le fichier `findPort.py`, qui s'affichera alors dans l'éditeur Python.
3. Exécute ce programme en sélectionnant `Run>Run Module` dans le menu. Tu verras alors des instructions s'afficher à l'écran.
4. Débranche le câble USB du circuit Arduino et appuie sur `Entrée` pour commencer l'analyse. Cette étape consiste à retirer le port USB de ton ordinateur pour que le programme puisse analyser tous les ports disponibles sur ta machine et en établir une liste. Le port que tu vas ajouter n'y figure pas.
5. Tu vas ensuite devoir brancher à nouveau le périphérique pour que le programme puisse recommencer l'analyse et détecter tous les nouveaux périphériques qui se sont ajoutés à la liste. Branche à nouveau ton câble, attends quelques secondes et appuie sur `Entrée`.
6. Si l'opération réussit, tu devrais obtenir un message indiquant le nom ou le numéro du port qui a été trouvé et le programme devrait te demander si tu souhaites le mémoriser. Saisis la lettre *y* (pour « yes », c'est-à-dire « oui » en français) et appuie sur `Entrée`. Comme tu peux le voir sur la figure 5-5, le programme `findPort.py` va créer un fichier nommé `findport.cache` qui mémorise le numéro de port de ton circuit Arduino. Plus tard, lorsque tu utiliseras les fonctions GPIO, ce fichier te permettra de retrouver le port du circuit Arduino sans difficulté.

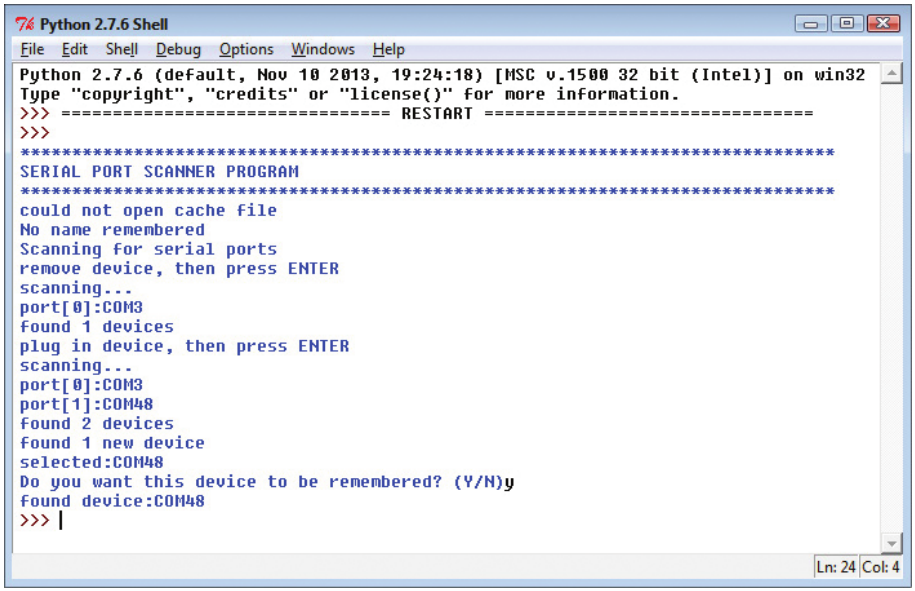

FIGURE 5-5 Le résultat obtenu suite à l'exécution du programme findPort.py

Si tu branches d'autres périphériques USB sur ton ordinateur et que tu redémarres ta machine ou rebranches ton circuit Arduino, il se peut que le numéro de port change et fasse échouer ton programme. Le cas échéant, il te suffira d'exécuter à nouveau le programme `findPort.py` ou de supprimer le fichier `findport.cache` et d'exécuter le programme pour effectuer une nouvelle analyse.

Il est assez difficile de rédiger un kit compatible avec toutes les versions d'un système d'exploitation, d'autant plus que les éditeurs les renouvellent sans cesse. Si tu rencontres des problèmes lors de l'installation de ton périphérique, fais un tour sur le site Internet du livre (`www.editions-eyrolles.com/dl/14292`) pour voir s'il existe une mise à jour du kit. Sinon, tu trouveras toujours les dernières actualités et mises à jour sur ma page GitHub à cette adresse : `https://github.com/whaleygeek/anyio`.

Contrôler une LED

Dans le chapitre 1, tu as rédigé un programme intitulé `bonjourMondeDeMinecraft.py` qui permettait d'afficher un message dans le chat de Minecraft. Avec des circuits électroniques, tu peux tout à fait rédiger un programme équivalent au « Bonjour le monde de minecraft » qui, au lieu d'afficher un message à l'écran, fait clignoter une diode. Ainsi, une fois que tu auras appris à vérifier que tout va bien en paramétrant ton ordinateur pour qu'il fasse clignoter une LED, tu seras en mesure de progresser et de créer des projets encore plus amusants.

La LED à laquelle nous avons affaire ici est un peu spéciale, car tu vas la relier au monde de Minecraft. Dans cette section, tu vas développer une idée que tu as déjà étudiée dans le chapitre 2, lorsque tu as rédigé le programme du paillasson magique. Cette fois-ci, ton paillasson sera connecté au circuit et la LED se mettra à clignoter dès que ton joueur marchera dessus. Mais si tu peux connecter ce circuit à une LED, alors il est également possible de le connecter à n'importe quel autre type d'objet. Imagine-toi donc tout ce que tu pourrais programmer et provoquer dans le monde réel en plaçant ton joueur sur le paillasson. Tu pourrais tout à fait allumer la lumière de ta chambre ou ouvrir les rideaux de ta fenêtre, ou que sais-je encore !

Si tu réalises ce projet sur un PC ou un Mac, tu vas utiliser le kit anyio. Tu le trouveras dans le kit de démarrage, mais tu peux aussi le télécharger en te rendant sur les sites mentionnés au début de ce chapitre.

Allumer une LED avec un ordinateur

Tu vas tout d'abord préparer ton circuit sur la plaque d'essais. Les figures 5-6 et 5-7 montrent comment la LED est raccordée à ton ordinateur.

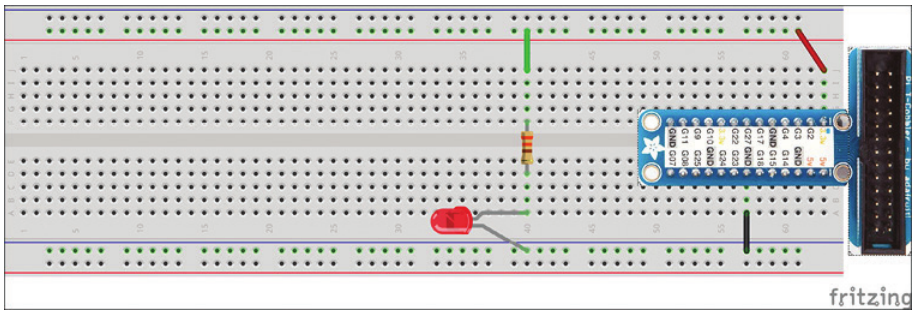

FIGURE 5-6 Une LED et une résistance raccordées à un Raspberry Pi

1. Prends la patte la plus courte de ta LED. Il s'agit de son pôle négatif, appelé cathode. Connecte-la à la barrette à tension nulle (0 V), située en bas de ta plaque d'essais.
2. L'autre patte de la LED, la plus longue, correspond au pôle positif, l'anode. Place l'une des pattes de ta résistance sur la même barrette que l'anode de la LED (dans la partie moyenne inférieure de la plaque). Ensuite, introduis l'autre patte de la résistance dans un trou de la bande moyenne supérieure de ta plaque d'essais. Pour rappel, la résistance est nécessaire, car elle régule le courant qui traverse la LED.
3. Pour tester la LED, place un fil électrique sur la patte de la résistance connectée à la partie moyenne supérieure de ta plaque d'essais et connecte l'autre extrémité du fil à la bande supérieure de la plaque d'essais. Tu le déplaceras plus tard.

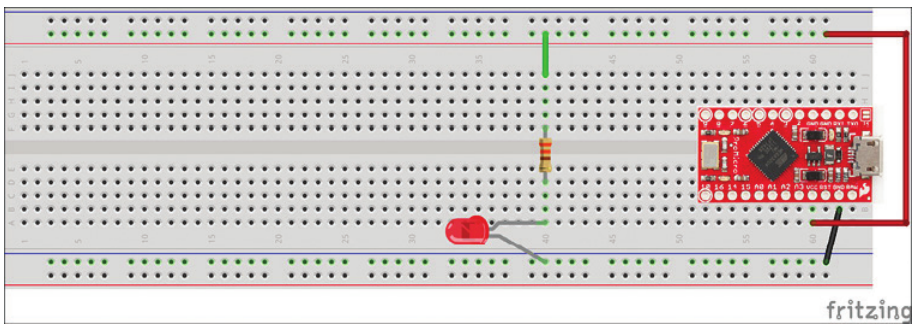

FIGURE 5-7 Une LED et une résistance raccordées à un circuit Arduino

Les instructions qui suivent sont différentes si tu utilises un Raspberry Pi ou un circuit Arduino sur PC ou Mac. Passe donc directement à la section qui te concerne.

Sur Raspberry Pi

1. Si tu disposes d'une étiquette autocollante pour le Pi-T-Cobbler, colle-la comme sur la figure 5-8. Cela t'aidera à reconnaître les broches GPIO plus facilement. Connecte le Pi-T-Cobbler à la plaque d'essais de façon à ce que la prise noire se retrouve sur la droite, à l'extérieur de la plaque. Introduis les broches du Pi-T-Cobbler dans les trous de la partie droite de la plaque, jusqu'au bord. La moitié des broches doit être branchée sur la partie moyenne supérieure de la plaque et l'autre moitié sur la partie moyenne inférieure. Appuie assez fort pour t'assurer que les broches soient bien enfoncées dans les trous.
2. Connecte le câble plat au Pi-T-Cobbler et au Raspberry Pi. Le connecteur est marqué d'une encoche et la prise du Pi-T-Cobbler contient une fente, ce qui signifie que tu ne peux brancher le câble que dans un sens. En branchant le câble plat au Raspberry Pi, fais attention de placer la cosse en plastique vers le haut par rapport au circuit, sinon les broches seront à l'envers et cela ne fonctionnera pas ! Tu peux voir tous les branchements sur la figure 5-8.

La dernière chose qu'il te reste à faire est de vérifier que ton ordinateur est capable d'allumer la LED. En effet, avant de la faire clignoter à l'aide du programme Python, tu dois d'abord t'assurer que la LED peut fonctionner avec le courant de ton ordinateur.

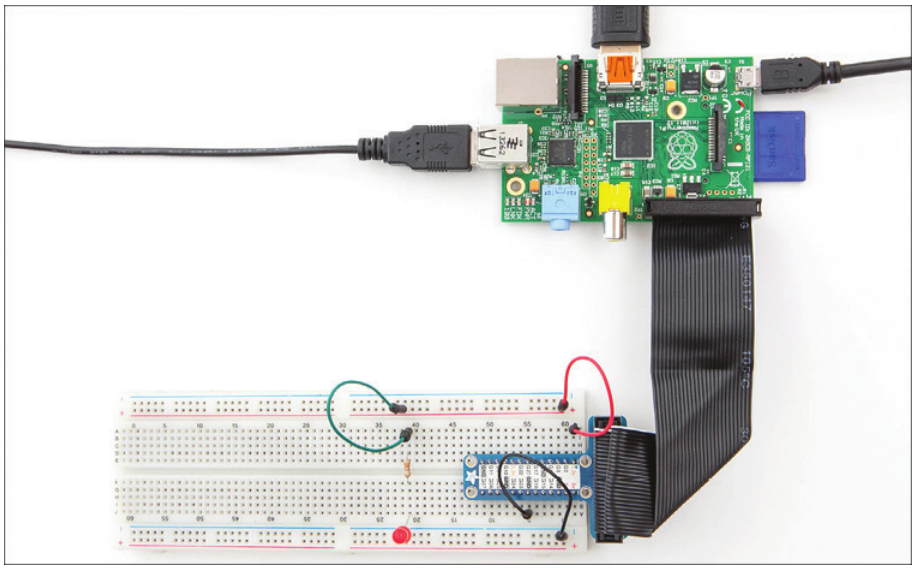

FIGURE 5-8 Le Pi-T-Cobbler est connecté au Raspberry Pi.

3. Branche un fil électrique sur la bande positive de la plaque (dans la partie supérieure) et relie-le à la broche 3,3 V du Pi-T-Cobbler.
4. Branche ensuite un fil sur la bande négative de la plaque (dans la partie inférieure) et relie-le à la broche 0 V du Pi-T-Cobbler.

Si ton Raspberry Pi est allumé, la LED devrait s'allumer, car elle est désormais connectée à l'alimentation de ton ordinateur.

Sur Arduino

1. Aligne ton circuit Arduino sur la plaque d'essais de sorte que la prise USB se retrouve sur le côté droit de la plaque, à l'extérieur. Il se peut que tu doives appuyer assez fort pour enfoncer toutes les broches dans la plaque d'essais. Une simple pression sur la totalité de la surface du circuit suffira et t'évitera d'endommager les broches. Tu peux voir tous les branchements sur la figure 5-9.
2. Connecte ensuite le circuit Arduino à ton ordinateur via le port USB.

La dernière chose qu'il te reste à faire est de vérifier que ton ordinateur est capable d'allumer la LED. En effet, avant de la faire clignoter à l'aide du programme Python, tu dois d'abord t'assurer que la LED peut fonctionner avec le courant de ton ordinateur.

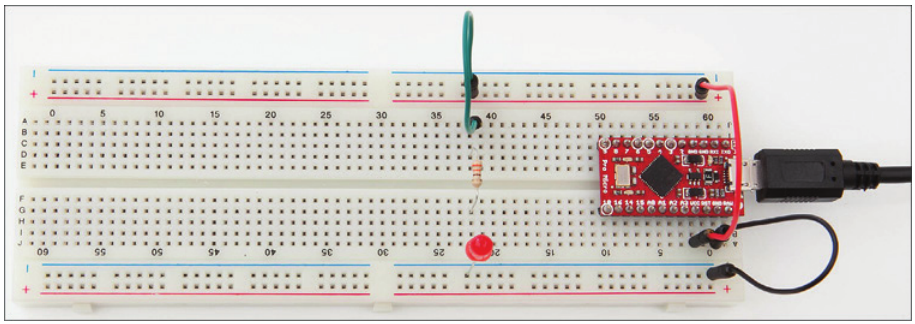

FIGURE 5-9 Le circuit Arduino est connecté à la plaque d'essais.

3. Branche un fil électrique sur la bande positive de la plaque (dans la partie supérieure) puis relie-le à la broche VCC du circuit Arduino (VCC est un terme technique qui désigne le pôle positif ; il équivaut à 3,3 volts sur ton circuit Arduino).
4. Branche ensuite un fil sur la bande négative de la plaque (dans la partie inférieure) puis relie-le à la broche GND du circuit Arduino (GND est un terme technique qui désigne le pôle négatif d'un circuit, soit un courant de 0 volt).

Si ton circuit Arduino est connecté, la LED devrait s'allumer, car elle est désormais connectée à l'alimentation de ton ordinateur.

Si ta LED ne s'allume pas, vérifie que tu l'as placée dans le bon sens. Inverse son branchement pour voir si elle fonctionne. Tu peux également retourner ta plaque d'essais pour voir comment elle se compose et retracer les lignes de ton circuit depuis la broche à 3,3 volts jusqu'à celle de 0 volt, en t'arrêtant à chaque composant pour voir s'ils sont bien branchés. Généralement, si une LED ne s'allume pas, c'est qu'elle est placée dans le mauvais sens ou que ses pattes ont été enfoncées dans les mauvais trous sur la plaque d'essais, ne formant pas un circuit complet.

Faire clignoter la LED

Maintenant que ta LED est bien raccordée et qu'elle s'allume comme prévu, il ne te reste plus qu'à rédiger un programme Python pour la faire clignoter.

1. Démarre IDLE et crée un nouveau programme en cliquant sur `File>New File`, puis nomme-le `testLED.py`.
2. Importe le module de temps que tu utiliseras pour introduire un délai entre chaque clignotement.

```
import time
```

Dans tous les scripts de ce chapitre, j'utilise les instructions `import RPi.GPIO as GPIO` ou `import anyio.GPIO as GPIO`, exactement comme s'il s'agissait de l'instruction `import mcpi.minecraft as minecraft`, que tu connais déjà. Le mot-clé `as` indique à Python de renommer le module et de lui attribuer le nom qui se situe à sa droite. Cela s'avère particulièrement utile lorsque les noms des modules sont très longs. Mais cela signifie aussi que tu n'auras qu'à modifier une ou deux lignes des instructions et programmes de ce chapitre pour communiquer avec ton circuit Arduino ou ton Raspberry Pi, ce qui est plutôt cool. Merci Python !

3. Importe le module qui te permet de prendre le contrôle du port GPIO et attribue une constante au numéro du port connecté à ta LED, comme ceci :

Sur Raspberry Pi

```
import RPi.GPIO as GPIO
LED = 15
```

Sur Arduino

```
import anyio.GPIO as GPIO
LED = 15
```

4. Détermine le mode de numérotation de la broche GPIO. Dans le code qui va suivre, tu vas voir que nous utilisons les lettres BCM. C'est l'abréviation du mot anglais « Broadcom », qui est le nom de la puce du processeur de ton Raspberry Pi. Ces lettres signifient que tu indiques à Raspberry Pi d'utiliser les numéros GPIO au lieu des numéros de broche qui figurent sur ton circuit. Définis maintenant ta broche GPIO en mode sortie pour que ton programme puisse modifier le courant d'alimentation de ta LED :

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED, GPIO.OUT)
```

5. Maintenant, passe à la rédaction de la fonction qui te permettra de provoquer un clignotement. On passe un paramètre `t` à la fonction pour pouvoir modifier la vitesse de clignotement de la LED. Quant à la fonction `output()`, elle servira à modifier la tension du GPIO. Lorsque tu utiliseras `True` la tension s'élèvera à 3,3 volts et lorsque tu utiliseras `False` elle tombera à 0 volt. Cela aura pour effet d'allumer et d'éteindre la LED :

```
def clignotement(t)
    GPIO.output(LED, True)
    time.sleep(t)
    GPIO.output(LED, False)
            time.sleep(t)
```

6. Rédige une boucle pour faire clignoter la LED, puis appelle la fonction `GPIO.cleanup()` à la fin du programme, afin d'effectuer un nettoyage et de remettre les broches GPIO dans leur état initial. Enfin, pour en savoir plus sur le bloc `try/finally`, je te conseille de lire l'encadré « Percer les secrets du code » plus loin.

```
try:
    while True:
        clignotement(0.5)
finally :
    GPIO.cleanup()
```

7. Pour que la LED puisse clignoter, elle doit être connectée à la broche GPIO sur ton ordinateur. Suis donc ces quelques étapes.

   Sur Raspberry Pi

   Déplace le fil qui relie la résistance à la barrette de 3,3 volts. Débranche l'extrémité connectée à la barrette et branche-la à la broche 15 du port GPIO (sur l'étiquette Pi-T-Cobbler, elle est nommée G15).

Sur Arduino

Déplace le fil qui relie la résistance à la barrette de 3,3 volts. Débranche l'extrémité connectée à la barrette et branche-la à la broche 15 du port GPIO sur le circuit Arduino.

Le circuit Arduino est lui-même doté de LED, la jaune permet de savoir lorsque le circuit reçoit des commandes par USB. Tu la verras clignoter deux fois plus vite que celle de ta plaque d'essais, car à chaque fois que cette dernière reçoit l'instruction de s'allumer ou de s'éteindre, la LED jaune de ton circuit Arduino se met à clignoter. Ne sois pas surpris ; cela veut dire que tout fonctionne correctement !

Exécuter un programme GPIO

Il est enfin temps que tu exécutes ton programme et que tu vérifies si la LED fonctionne !

Sur PC ou Mac

Exécute le programme comme d'habitude depuis IDLE. Le circuit Arduino est branché via un port USB et ton PC ou ton Mac peut désormais s'y connecter en continu.

Sur Raspberry Pi

Exécuter le programme sur un Raspberry Pi est un peu plus compliqué. En effet, bien que tu continues d'utiliser IDLE pour ouvrir et modifier tes programmes, ceux qui fonctionnent avec le GPIO devront s'exécuter différemment. Pour en savoir plus sur les raisons de ces manœuvres supplémentaires, consulte l'encadré « Percer les secrets du code ».

1. Ouvre le programme LXTerminal depuis le bureau ou le menu. Cela va ouvrir une fenêtre d'invite de commandes, que tu utiliseras pour saisir des instructions sur ton Raspberry Pi.
2. Accède au dossier `MyAdventures` en saisissant la commande `cd`.

```
cd MyAdventures
```

3. Exécute ton programme Python en utilisant la commande `sudo` (pour connaître la signification de ce terme, lis l'encadré « Percer les secrets du code ») :

```
sudo python testLED.py
```

Ta LED a-t-elle clignoté ? Cela peut te sembler anodin, mais tu viens d'effectuer quelque chose d'incroyable. Tu as non seulement réussi à prendre le contrôle d'un circuit électronique, mais ce faisant tu as également dépassé les limites de ton ordinateur !

Il se peut que le message d'erreur suivant s'affiche : « Channel already in use » (ce qui signifie littéralement « Canal déjà occupé »). Pas de panique, c'est normal. Tu vas bientôt apprendre à utiliser la fonction `GPIO.cleanup()` pour éviter ce message d'erreur.

La prochaine étape consistera à relier ce programme au monde de Minecraft.

Souviens-toi que lorsque tu fais clignoter une LED grâce à un programme Python, tu utilises un programme semblable à celui de « Bonjour le monde de Minecraft », que tu connais déjà. Lorsque tu raccordes un circuit électronique à un ordinateur, il est important de vérifier qu'il fonctionne correctement avant de t'aventurer plus loin. Si ta LED ne clignote pas, vérifie bien tous tes branchements et assure-toi que le fil qui relie ta LED à la broche GPIO est inséré dans le bon trou sur la plaque d'essais. Plus tôt, tu as vérifié que ta LED fonctionnait correctement en la raccordant à l'alimentation de ton ordinateur et en lui envoyant un courant de 3,3 volts. Si elle ne fonctionne pas ici, tu peux en déduire que ton programme Python contient une erreur ou que le fil qui relie la LED au port GPIO est mal branché.

PERCER LES SECRETS DU CODE

Dans les paragraphes précédents, tu as employé deux tours de magie qu'il serait bon d'éclaircir un peu ici.

Tout d'abord, à quoi sert l'instruction `try/finally` avec `GPIO.cleanup()` ?

Afin d'éviter les messages d'erreur à chaque fois que tu exécutes ton programme, n'oublie pas d'utiliser la fonction `GPIO.cleanup()` à la fin de ton script. Elle permet de redéfinir tes broches GPIO en mode entrée et de désactiver ton circuit. En d'autres termes, cela signifie que tes broches ne pourront plus prendre le contrôle du circuit. Si, par inadvertance, tu te trompes de broche lors des branchements, tu peux provoquer un court-circuit en allumant ton dispositif, ce qui pourrait endommager ton circuit. Pour éviter cela, nous te conseillons d'appeler la fonction `GPIO.cleanup()` à la fin de ton programme. Ainsi, les éventuels courts-circuits n'endommageront pas ton périphérique lorsque tu effectueras des modifications.

Mais voilà qu'un problème se pose : comme tu as utilisé la boucle `while True:` dans ton script, tu as créé une boucle de jeu infinie. Or, la seule façon d'arrêter un programme est d'appuyer sur `Ctrl+F6` ou d'aller dans le menu de

l'interpréteur de commandes IDLE et de cliquer sur `Restart Shell`, car quand tu utilises cette fonctionnalité, le programme s'arrête immédiatement.

Pas d'inquiétude, il existe une solution. C'est ici qu'entre en jeu le bloc `try/finally`, qui permet de lever ce que l'on appelle des exceptions. Je ne m'attarderai pas sur les exceptions dans ce livre, mais sache que tu ne peux pas t'en passer lorsque tu utilises un port GPIO. En gros, lorsque tu appuies sur `Ctrl+F6` ou que tu arrêtes le programme depuis le menu IDLE, Python exécute l'exception `KeyboardInterrupt`, qui provoque l'arrêt du programme. Pour permettre à Python d'anticiper un arrêt, les programmeurs ont recours à une petite astuce qui consiste à utiliser le bloc `try/finally`. Après le mot-clé `finally`, tu peux saisir plusieurs instructions que tu souhaites exécuter avant l'arrêt du programme. Ainsi, lorsque tu appuies sur `Ctrl+F6` ou que tu cliques sur `Restart Shell` dans le menu de l'interpréteur, le programme passe directement aux instructions qui se trouvent dans le bloc `finally:`, les exécute, puis s'arrête. Cela te permettra ainsi de toujours appeler la fonction `GPIO.cleanup()` pour remettre les broches GPIO en position de sécurité, c'est-à-dire dans leur état initial, à la fin de ton programme.

Je n'aborderai pas le sujet des exceptions plus en détail dans ce livre, mais si tu souhaites en savoir plus, tu peux toujours consulter la page : `https://wiki.python.org/moin/HandlingExceptions`.

Deuxièmement, à quoi sert le mot-clé `sudo` sur Raspberry Pi et pourquoi faut-il exécuter les programmes GPIO depuis LXTerminal ?

Tous les composants des ordinateurs Raspberry Pi sont protégés. Un utilisateur `pi` normal ne peut donc pas y accéder en tant que tel ; il doit avoir des privilèges de super-utilisateur. Le terme `sudo` signifie « substitute user and do » (en français, « se substituer à l'utilisateur et prendre le contrôle ») et permet de passer du statut d'utilisateur `pi` à celui d'utilisateur `root` (le super-utilisateur), puis d'exécuter le programme `testLED.py`.

Ce dernier peut ainsi accéder au périphérique GPIO, même s'il est protégé. Si tu n'exécutes pas tes programmes GPIO de cette manière sur Raspberry Pi, tu recevras un message d'erreur.

Lorsque j'ai écrit ce livre, j'ai essayé de trouver une explication claire au terme `sudo`. Mais étonnamment, à chaque fois que j'effectuais une recherche sur Google à ce sujet, je tombais sur une page de mon blog, qui est apparemment très appréciée ! Tu pourras donc lire tout ce qu'il faut savoir sur ce mot-clé et son utilisation ici : `http://blog.whaleygeek.co.uk/sudo-on-raspberry-pi-why-and-why-you-need-to-ask-kids-too`.

Rédiger le programme de paillasson magique avec une LED

La dernière étape qu'il te reste à franchir est de relier ta LED au jeu Minecraft. Heureusement, cette étape est facile à réaliser, car elle s'appuie sur des connaissances que tu maîtrises déjà.

1. Enregistre ton programme `testLED.py` sous un nom différent en cliquant sur `File>Save As` dans le menu de l'éditeur et nomme-le `bienvenueLED.py`.

2. Indique les modules Minecraft que tu dois importer au début du fichier et connecte-toi au jeu :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
mc = minecraft.Minecraft.create()
```

3. En dessous, pose des constantes indiquant les coordonnées de ton paillasson de bienvenue. Tu dois donc décider où tu souhaites poser ton paillasson dans le monde et choisir les coordonnées en conséquence. Sur Raspberry Pi, celles-ci s'affichent en haut à gauche de l'écran lorsque tu joues à Minecraft. Sur PC et Mac, tu dois appuyer sur `F3` pour les faire apparaître :

```
MAISON_X = 0
MAISON_Y = 0
MAISON_Z = 0
```

4. Juste après ces constantes, crée un bloc de laine au niveau de ton paillasson. Voici donc ton paillasson, exactement comme celui que tu as créé dans le chapitre 2 :

```
mc.setBlock(MAISON_X, MAISON_Y, MAISON_Z, block.WOOL.id, 15)
```

5. Modifie ta boucle en copiant les instructions ci-après. Fais particulièrement attention à l'indentation. Ce code permet de détecter en continu la position de ton joueur et utilise la technique du géorepérage pour savoir s'il se trouve sur le paillasson. Si c'est le cas, il active la LED qui clignote pour t'indiquer que tu es bien arrivé à la maison :

```
try:
    while True:
        pos = mc.player.getTilePos()
        if pos.x == MAISON_X and pos.z == MAISON_Z:
            clignotement(0.5)
finally :
    GPIO.cleanup()
```

Exécute ce programme en suivant les instructions indiquées dans la section « Exécuter un programme GPIO » et vérifie que la LED s'allume lorsque tu franchis le pas de ta porte.

Fantastique ! Tu es arrivé chez toi et tu as maintenant tous les outils en main pour repousser les limites du monde virtuel de Minecraft et le relier à des objets du monde réel. Si tu es capable d'allumer une LED à l'aide d'un programme, tu seras également capable de contrôler d'autres dispositifs électroniques. Laisse-toi guider par ton imagination !

DÉFI

Procure-toi des LED de couleurs différentes et branche-les sur d'autres broches de ton port GPIO. Choisis-en une bleue et ajoute-la à ton programme Python en appelant la fonction `getBlock()` pour la faire clignoter lorsque ton joueur touche l'eau. Ajoutes-en ensuite une verte et fais-la clignoter lorsque tu touches l'herbe. En faisant preuve de créativité, combien d'événements peux-tu encore signaler avec des LED dans Minecraft ?

Utiliser un afficheur 7 segments

Dans le prochain chapitre, tu auras recours à un type d'afficheur numérique dont l'utilisation est très répandue. Il te servira notamment à afficher des chiffres et d'autres symboles dès qu'un événement surviendra dans Minecraft. Tu pourras ainsi l'utiliser dans le dernier jeu que nous t'apprendrons à créer dans ce livre. Mais avant de te lancer dans l'aventure et de l'utiliser avec Minecraft, tu dois d'abord effectuer quelques manipulations pour vérifier qu'il fonctionne correctement. Une fois que tu auras passé cette étape de vérifications, tu pourras relier le dispositif au jeu, comme tu viens de le faire avec la LED.

Si tu joues sur un PC ou un Mac, tu dois utiliser le kit anyio pour réaliser ce projet. Ce kit est disponible dans le kit de démarrage, mais tu peux aussi le télécharger depuis les sites indiqués au début de ce chapitre.

Qu'est-ce qu'un afficheur 7 segments ?

Si tu observes autour de toi ou que tu te rends dans le magasin le plus proche, tu verras qu'une multitude d'objets divers et variés sont dotés d'afficheurs 7 segments. Montres numériques, micro-ondes à affichage digital, lecteurs CD et même boîtiers de contrôle de chauffage ; tous ces dispositifs utilisent un modèle à sept segments. Sur la figure 5-10, tu peux voir le schéma d'un afficheur 7 segments et son circuit interne.

Tous les afficheurs 7 segments sont dotés de plusieurs LED. L'affichage le plus commun, celui que tu utiliseras dans ce projet, est en fait doté de huit LED, l'une d'elles servant à afficher le séparateur décimal en bas à droite. Seules sept LED correspondent au modèle en question et forment un huit, comme tu peux le voir sur la figure 5-10.

Dans le cadre de ce projet, tu dois retenir deux informations importantes au sujet des afficheurs 7 segments :

- si tu sais rédiger un programme capable d'allumer et d'éteindre une LED, alors tu sauras aussi rédiger un programme capable de contrôler ce dispositif, car il contient huit LED séparées, soit huit connexions GPIO séparées ;
- lorsque tu utilises ce modèle d'afficheur 7 segments, tu peux afficher toute une succession de symboles en composant avec les différentes LED. Grâce à ces simples segments, tu as la possibilité d'afficher n'importe quel chiffre de 0 à 9.

Dans la figure 5-10, outre le circuit interne de l'afficheur 7 segments, tu disposes également d'un schéma indiquant comment les broches sont reliées aux segments. Tu remarqueras que chaque LED dispose d'une anode connectée à la même broche. C'est cette broche que tu brancheras sur la barrette positive de ta plaque d'essais. Quant à la flèche du symbole de la diode, elle t'indique le sens du courant, qui part du côté positif vers le côté négatif.

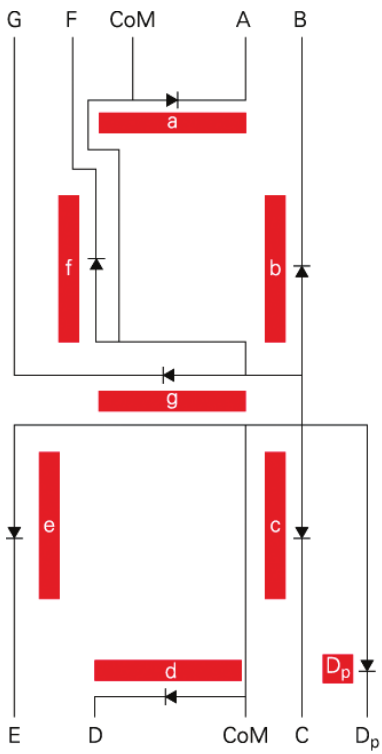

FIGURE 5-10 La disposition des segments et le circuit interne d'un afficheur 7 segments à anode commune

Dans l'industrie, on utilise généralement les lettres A B C D E F G pour désigner les segments de l'afficheur. Si tu observes celui-ci de plus près, tu remarqueras que les segments sont légèrement inclinés vers la droite. Cela n'a qu'une fonction esthétique, que les fabricants ont adoptée pour améliorer l'apparence de ces composants.

Il existe tout un tas d'afficheurs 7 segments en vente sur le marché, et bon nombre d'entre eux ont des circuits différents. Certains sont à cathode commune, c'est-à-dire que toutes les cathodes des LED sont reliées entre elles. D'autres sont à anode commune, ce qui veut dire que toutes les anodes des LED sont reliées entre elles. Tu trouveras également des circuits agencés différemment. Pour ce projet, j'utilise un afficheur 7 segments à anode commune un peu spécial que je me suis procuré sur le site `www.selectronic.fr`, mais si tu utilises un afficheur différent, vérifie bien la configuration du circuit avant de le connecter. Si c'est ton cas et que tu utilises, par exemple, un afficheur à cathode commune, j'ai ajouté quelques commentaires à ton attention dans les scripts de ce chapitre, à côté de la constante `ON`, pour que tu saches comment adapter tes instructions.

Brancher l'afficheur 7 segments

Toutes les conditions sont maintenant réunies pour que tu puisses brancher ton afficheur 7 segments à la plaque d'essais et vérifier qu'il fonctionne correctement. Suis ces quelques étapes :

1. L'afficheur 7 segments étant doté de cinq broches en haut et cinq broches en bas, insère-le au centre de ta plaque d'essais, de sorte que les dix broches se trouvent à distance égale.
2. Connecte un fil à la moitié supérieure de l'afficheur, en partant de la broche du milieu (l'anode commune) et relie-le à la barrette de 3,3 volts située dans la partie supérieure de la plaque d'essais. (Si tu utilises un afficheur différent du mien, par exemple un afficheur à cathode commune, tu devras connecter le fil à la moitié inférieure de l'afficheur, en partant de la broche du milieu, et le relier à la barrette de 0 volt située dans la partie inférieure de la plaque d'essais.)
3. Comme cet afficheur contient des LED, tu devras utiliser des résistances, à l'exemple de l'exercice précédent. Et comme il contient précisément huit LED, tu devras utiliser huit résistances. Prends chacune des résistances de 330 ohms (code couleur : orange, orange, marron), plie-leur les pattes et place-les de part et d'autre de l'afficheur, comme sur la figure 5-12. Afin d'éviter tout court-circuit, réfère-toi à la figure 5-4 pour te rappeler la configuration du circuit de la plaque d'essais.

4. Tu peux maintenant tester ton afficheur. Pour cela, tu vas utiliser un fil que tu brancheras temporairement au circuit. Connecte une extrémité du fil à la bande négative de la plaque (dans la partie inférieure) et utilise l'autre extrémité pour la connecter tour à tour aux pattes libres des résistances, c'est-à-dire aux pattes qui ne sont pas branchées à l'afficheur. Chaque fois que tu toucheras une patte, un segment de l'afficheur s'allumera. Compare les segments au schéma de la figure 5-10, qui correspond à l'afficheur que j'ai utilisé dans ce projet. Puis vérifie si le circuit interne de ton afficheur correspond au schéma de ce livre en t'aidant des lettres indiquées sur chaque segment. Si ton circuit est différent, note les lettres des segments qui correspondent aux résistances, car tu en auras besoin par la suite. (Voir la figure 5-11.)

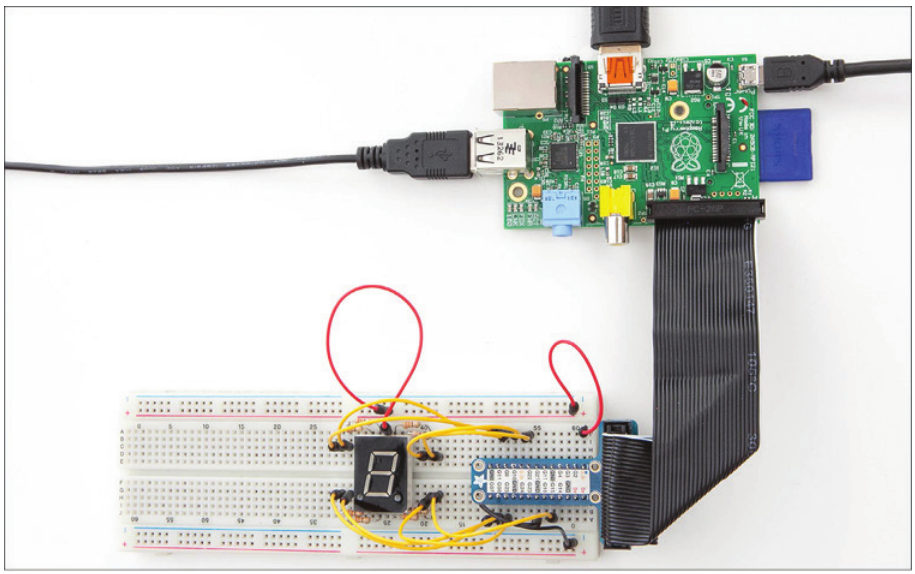

FIGURE 5-11 La LED à 7 segments est branchée à la plaque, prête à être testée sur Raspberry Pi.

Une fois que tu auras terminé les essais et que tu seras parvenu à faire fonctionner l'afficheur, la prochaine étape consistera à connecter ton afficheur au port GPIO de ton ordinateur. Tu pourras ainsi contrôler les LED du périphérique directement depuis ton interpréteur de commandes Python. Retire le fil de test de la plaque et munis-toi d'autres fils. Avec ces derniers, connecte chaque patte libre des résistances aux broches du port GPIO, en suivant le schéma de la figure 5-12 (pour un Raspberry Pi) ou de la figure 5-13 (pour un circuit Arduino). Si, lors du test précédent, tu as découvert que ton circuit était différent du mien, aide-toi des notes que tu as prises pour déterminer les lettres correspondant aux résistances et utilise ces informations pour connecter chaque segment aux broches GPIO qui conviennent. Si tu négliges cette étape, tu risques de voir des motifs bien étranges s'afficher lorsque tu exécuteras ton programme Python !

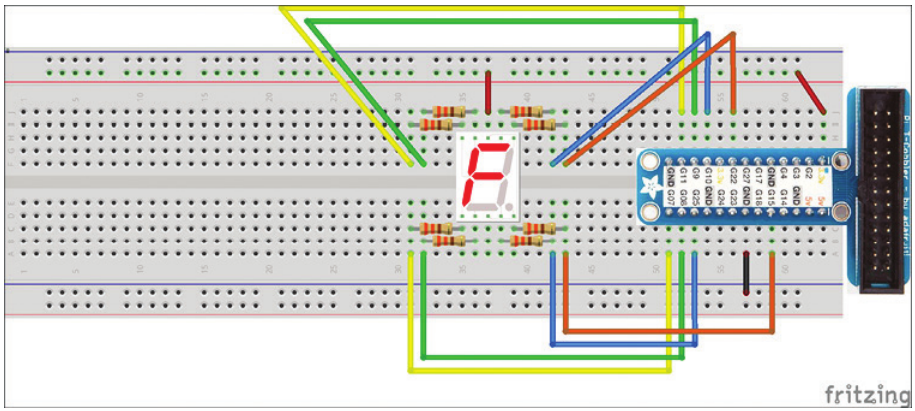

FIGURE 5-12 Schéma du circuit de l'afficheur 7 segments sur Raspberry Pi

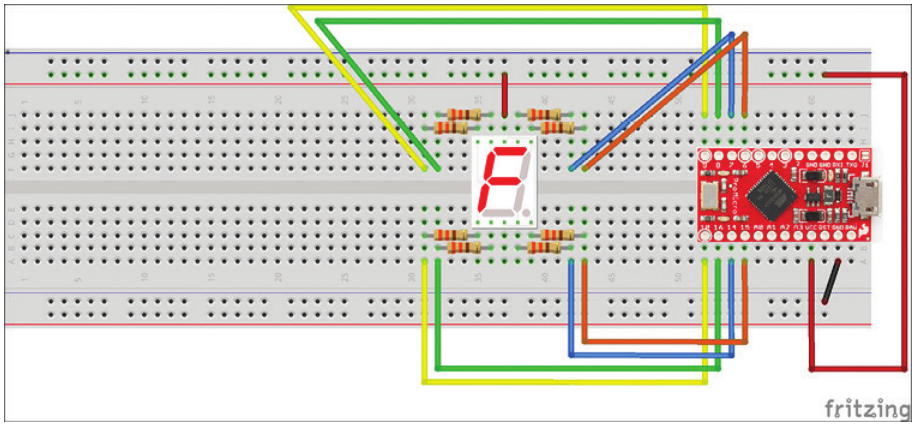

FIGURE 5-13 Schéma du circuit de l'afficheur 7 segments sur Arduino

Programmer en Python pour contrôler l'afficheur 7 segments

Maintenant que tu as testé ton afficheur 7 segments et que tu l'as correctement connecté aux huit broches GPIO de ton ordinateur, tu peux commencer à écrire des programmes Python pour l'activer.

1. Crée un nouveau programme en sélectionnant `File>New File`, puis nomme-le `testAfficheur.py`.
2. Importe tous les modules nécessaires et définis des constantes en leur affectant les numéros des huit broches GPIO :

Sur Raspberry Pi

```
import RPi.GPIO as GPIO
BROCHES_LED = [10,22,25,8,7,9,11,15] # ordre important
```

Sur Arduino

```
import anyio.GPIO as GPIO
BROCHES_LED = [7,6,14,16,10,8,9,15] # ordre important
```

3. Détermine le mode de numérotation du port GPIO :

```
GPIO.setmode(GPIO.BCM)
```

4. Choisis ton type d'afficheur (pour ce qui me concerne, j'ai utilisé un afficheur à anode commune dans mes circuits) :

```
ON = False # False = anode commune, True = cathode commune
```

5. Rédige une boucle configurant les huit broches en mode sortie :

```
for g in BROCHES_LED:
    GPIO.setup(g, GPIO.OUT)
```

6. Comme tu le sais déjà, les LED de ton afficheur seront soit allumées, soit éteintes. Il existe un super moyen de définir un beau modèle de fonctionnement pour ton afficheur : les listes. En effet, tu peux construire un superbe modèle en enregistrant huit valeurs `True` ou `False` dans une liste, chacun des index de la liste correspondant à une LED de l'afficheur. Te souviens-tu que tu as appris à créer des listes avec des index dans le chapitre 4 ? Eh bien, c'est exactement ce que tu vas faire ici. Dans la liste ci-dessous, tous les segments de l'afficheur (A, B, C, D, E, F et G) seront définis sur ON (à l'aide de la valeur `True` en guise d'index), à l'exception du séparateur décimal (DP) qui sera désactivé, car tu placeras la valeur `False` en fin de liste. L'index `[0]` dans la liste correspond au segment A, l'index `[1]` au segment B, et ainsi de suite.

```
modele = [True, True, True, True, True, True, True, False]
          # ABCDEFG (sauf DP)
```

7. Rédige une boucle qui passera en revue toutes les broches de ton modèle :

```
for g in range(8):
    if modele[g]:
        GPIO.output(BROCHES_LED[g], ON)
    else:
        GPIO.output(BROCHES_LED[g], not ON)
```

8. La fin du programme ne tient désormais plus qu'à une touche. Tu dois demander à ton programme d'attendre ton signal avant de s'arrêter, sinon toutes les LED

s'éteindront et il ne se passera rien. Pour cela, utilise la fonction `raw_input()`. Elle exige en effet que tu appuies sur la touche `Entrée` pour que ton programme passe à la ligne d'après :

```
raw_input("fini ?")
```

9. Enfin, demande au programme de nettoyer le port GPIO avant de s'arrêter :

```
GPIO.cleanup()
```

Exécute le programme et vois ce qu'il se passe. Si tu utilises un Raspberry Pi, n'oublie pas d'exécuter ce programme depuis LXTerminal à l'aide de la commande `sudo python testAfficheur.py`.

DÉFI

Essaye de calculer les valeurs à insérer dans ton modèle pour afficher les chiffres de 0 à 9. Rappelle-toi que les valeurs `True` et `False` dans la liste modèle sont agencées dans l'ordre suivant : A, B, C, D, E, F, G, DP. Teste-les sur ton afficheur. Combien de lettres de l'alphabet peux-tu faire apparaître sur ton afficheur 7 segments ? Pour t'aider, tu trouveras des exemples pratiques de modèles sur cette page Wikipédia : `https://fr.wikipedia.org/wiki/Afficheur_7_segments`.

Utiliser un module Python pour contrôler l'afficheur

Tu t'es probablement aperçu qu'il existe une multitude de modèles pratiques à utiliser sur ton afficheur 7 segments. Pour te faciliter la tâche, je t'ai confectionné un petit module Python que j'ai appelé `seg7.py` et que j'ai inclus dans le dossier `anyio` de ton kit de démarrage. Tu peux utiliser et modifier ce module autant que tu le souhaites et l'ajouter à tes propres jeux et programmes. En suivant ces quelques étapes, tu verras qu'il est très facile de l'insérer dans un programme existant.

1. Commence un nouveau programme en sélectionnant `File>New File`, puis enregistre-le sous le nom `testAfficheur2.py`.
2. Importe mon module d'afficheur ainsi que le module de temps pour pouvoir ajouter un délai.

```
import anyio.seg7 as display
import time
```

3. Copie toutes les lignes de ton script `testAfficheur.py` depuis le début jusqu'à la commande `ON=False`.

4. Ensuite, configure le module d'afficheur en lui donnant accès aux broches de ton périphérique GPIO et en lui fournissant une liste des numéros de broches à utiliser pour chaque segment :

```
display.setup(GPIO, BROCHES_LED, ON)
```

5. Rédige une boucle qui compte de 0 à 9 indéfiniment. Si j'ai choisi d'utiliser une boucle ici, c'est pour que tu puisses la laisser tourner indéfiniment pendant que tu modifies le câblage de ton afficheur, au cas où certains segments ne fonctionneraient pas. Ainsi, tu ne seras pas obligé d'exécuter ton programme à chaque fois.

```
try:
    while True:
        for d in range(10):
            display.write(str(d))
            time.sleep(0.5)
finally :
    GPIO.cleanup()
```

Enregistre le programme, puis exécute-le et observe ce qu'il se passe. L'afficheur devrait compter de 0 à 9 sans discontinuer. N'oublie pas que sur Raspberry Pi tu dois utiliser `sudo python testAfficheur2.py` dans une fenêtre LXTerminal.

Au lieu d'utiliser `time.sleep(0.5)` dans le script précédent, tu peux employer un délai plus long, de deux secondes par exemple. Tu peux même utiliser une fonction `raw_input()` pour que le modèle affiché reste dans la même position, pendant que tu manipules le câblage de ton circuit et que tu t'assures que tout est correctement en place.

DÉFI

Démarre le programme `seg7.py` qui se trouve dans le dossier `MyAdventures/anyio` et suis les instructions du fichier qui t'indiquent comment ajouter de nouveaux symboles. Parmi les lettres que tu as créées dans le défi précédent, identifie celles que tu peux coder dans ce module. Quels mots peux-tu épeler à l'aide de l'afficheur ? Peux-tu, par exemple, écrire ton prénom ? Sinon, quelles lettres te manque-t-il ?

Créer un détonateur

Dans le projet final de ce chapitre, tu vas construire un énorme détonateur rouge. Lorsque tu appuieras sur le bouton-poussoir, il déclenchera un compte à rebours sur l'afficheur 7 segments. Ton joueur aura donc assez de temps pour courir et se mettre à l'abri avant l'explosion. Puis, un énorme cratère apparaîtra à l'endroit où ton joueur se trouvait lorsque tu as actionné le détonateur. Ce nouveau programme constitue un accessoire bien pratique à ajouter à ta boîte à outils, car il te permet de faire de l'espace en un rien de temps dans Minecraft et de construire des structures. En outre, ce projet va te faire découvrir un autre mode de GPIO : le mode d'entrée. Il sert notamment à détecter lorsqu'un bouton est actionné, donc tu pourras mettre tes connaissances sur l'afficheur 7 segments en pratique !

Tu peux regarder une vidéo didactique sur la création d'un détonateur en te rendant sur le site Internet du livre : `www.editions-eyrolles.com/dl/14292`. Choisis la vidéo « Adventure 5 » (en anglais).

Pour les besoins de ce projet et si tu es sur Mac ou PC, tu devras utiliser le kit anyio. Il t'est fourni dans le kit de démarrage.

Réaliser le câblage d'un bouton

La première étape de ce projet consiste à réaliser le câblage du bouton-poussoir de ton détonateur. Choisis un bouton rouge pour qu'il ait l'air encore plus foudroyant !

Le type d'interrupteur que tu vas utiliser dans cet exercice est un bouton-poussoir tactile. Cela veut dire que le bouton ne laisse passer le courant que lorsque tu appuies dessus, refermant ainsi le circuit. Cependant, le bouton ne restera pas en position fermée indéfiniment ; dès que tu relèveras le doigt, cela coupera le courant.

Sur la figure 5-14, tu peux voir à quoi ressemble le circuit sur Raspberry Pi ; sur la figure 5-15, tu as le schéma du circuit sur Arduino. Pour réaliser ton câblage, tu peux soit te concentrer sur ces schémas, soit suivre les instructions énumérées ci-dessous, à toi de voir !

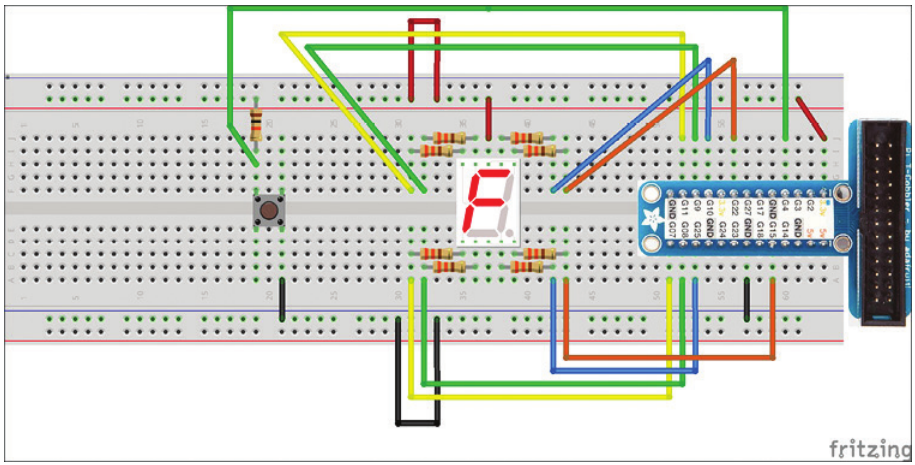

FIGURE 5-14 Schéma du circuit d'un bouton raccordé à un Raspberry Pi

1. Place ton bouton sur une partie encore libre de ta plaque d'essais. Le bouton que j'ai utilisé (voir la figure 5-16) est doté de quatre broches qui s'insèrent parfaitement au centre de la plaque. En appuyant sur le bouton, tu vas connecter les broches de gauche à celles de droite, à condition de les avoir placées sur les parties supérieure et inférieure de la plaque, comme tu peux le voir sur les figures 5-14 et 5-15.

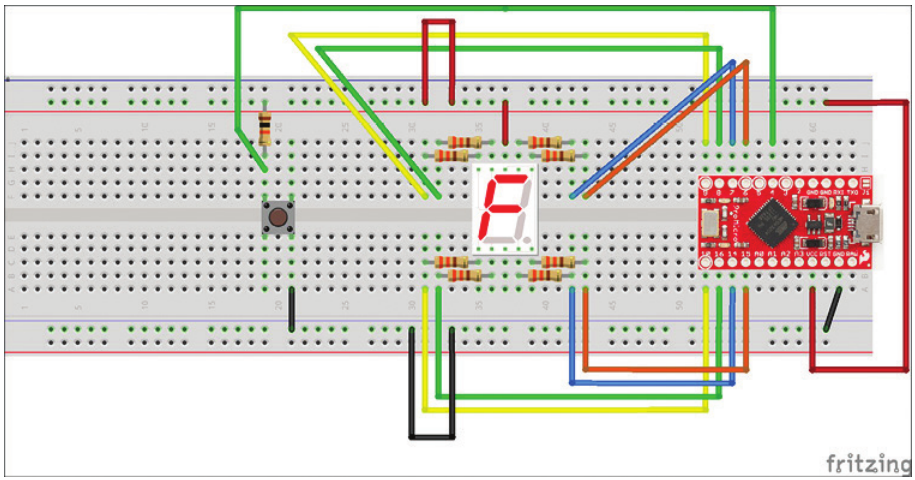

FIGURE 5-15 Schéma du circuit d'un bouton raccordé à un circuit Arduino

2. Tu vas avoir besoin d'une résistance pour tirer le courant d'entrée du bouton à 3,3 volts. Si tu n'utilises pas de résistance, ton bouton provoquera de fausses pressions et ton détonateur n'arrêtera pas de s'éteindre ! (Pour en savoir plus sur la nécessité d'utiliser une résistance dans ce cas précis, tu peux lire l'encadré « Percer les secrets du circuit ».) Place ta résistance de tirage 10 K (code couleur :

marron, noir, orange) en connectant une patte à la broche du côté gauche du bouton et l'autre patte à la bande de 3,3 volts située en haut de la plaque.

3. Pose un fil électrique entre la résistance et le bouton et branche l'autre extrémité à la broche 4 du GPIO.

4. Connecte ensuite un autre fil électrique à la broche du côté inférieur droit du bouton et relie-le à la bande de 0 volt située en bas de ta plaque. Pour savoir comment fonctionne ce raccordement, consulte l'encadré « Percer les secrets du circuit », mais de manière générale, tous les boutons sont raccordés aux ordinateurs de cette façon. La figure 5-16 te montre notamment comment raccorder un bouton à une broche GPIO.

Il existe des boutons-poussoirs en tous genres. Si tu utilises le même type de bouton que moi pour ce projet, tu remarqueras que les quatre broches sont légèrement inclinées pour mieux s'accorder à la plaque et faciliter la soudure. Fais très attention lorsque tu insères le bouton dans la plaque et veille à ne pas casser les broches. Pour cela, il te suffit d'appuyer de façon uniforme et assez fermement sur le bouton. Si tu possèdes une petite pince universelle, tu peux également l'utiliser pour aplatir les broches et faciliter ainsi leur insertion dans la plaque d'essais.

PERCER LES SECRETS DU CIRCUIT

Après avoir inséré ton bouton sur la plaque, tu as utilisé une résistance qui est appelée « résistance de tirage », car elle tire le courant de la broche jusqu'à 3,3 volts.

Lorsque le bouton n'est pas actionné et que le circuit est ouvert, le fil reliant la broche GPIO au bouton fournit un courant de 3,3 volts, que la broche du GPIO traduira numériquement comme le chiffre 1. En revanche, lorsque tu appuies sur le bouton, son mécanisme interne va refermer le circuit en reliant ses deux broches gauches à ses deux broches droites. Le fil du port GPIO va alors être tiré fortement vers le bas, vers un courant de 0 volt, à savoir celui de la bande inférieure de la plaque. À ce moment-là, la broche GPIO va traduire ce courant comme un 0. La résistance est puissante (10 Kohms équivalent à 10 000 ohms), mais cette puissance est nécessaire, car sans elle, le bouton provoquerait un court-circuit en connectant la bande de 3,3 volts à celle de 0 volt, ce qui ferait redémarrer ton ordinateur ! Cela est fortement déconseillé, car dans certains cas, un tel redémarrage pourrait endommager ton ordinateur. Quant à la dénomination 10 K, il s'agit d'une abréviation que les ingénieurs utilisent pour désigner une valeur de 10 000 ohms.

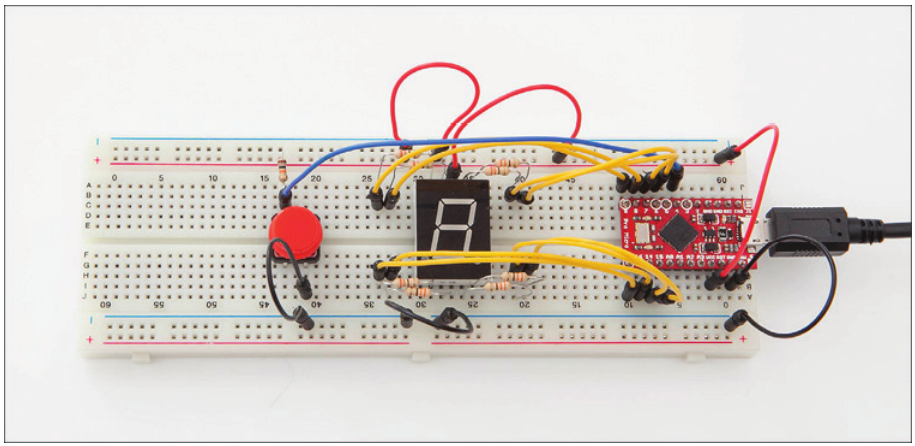

FIGURE 5-16 Sur cette plaque, le bouton est raccordé à un port GPIO via un circuit Arduino et dispose d'une résistance de tirage.

Rédiger le programme du détonateur

Maintenant que tu as raccordé ton bouton, il ne te reste plus qu'à rédiger un programme Python qui te permettra de surveiller l'état de ton bouton à l'aide d'une boucle. Lorsqu'il détectera le signal numérique 0 sur la broche GPIO (autrement dit, lorsque tu appuieras sur le bouton), il lancera un compte à rebours de 5 à 0 sur l'afficheur 7 segments. Il provoquera alors une énorme explosion dans le monde de Minecraft en formant un gigantesque cratère.

1. Crée un nouveau programme en sélectionnant `File>New File`, puis nomme-le `detonateur.py`.
2. Importe les modules dont tu as besoin :

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
import anyio.seg7 as display
```

3. Configure les paramètres GPIO de ton ordinateur :

Sur Raspberry Pi

```
import RPi.GPIO as GPIO
BOUTON = 4
BROCHES_LED = [10,22,25,8,7,9,11,15] # ordre important
```

Sur Arduino

```
import anyio.GPIO as GPIO
```

```
BOUTON = 4
BROCHES_LED = [7,6,14,16,10,8,9,15] # ordre important
```

4. Configure la broche GPIO du bouton en mode entrée et configure aussi les broches GPIO de l'afficheur.

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(BOUTON, GPIO.IN)
ON = False # False = anode commune, True = cathode commune
display.setup(GPIO, BROCHES_LED, ON)
```

5. Connecte-toi au jeu Minecraft :

```
mc = minecraft.Minecraft.create()
```

6. Rédige une fonction pour provoquer les événements suivants : lâcher un bloc de TNT à un endroit précis, déclencher un compte à rebours et former un gros cratère là où le bloc de TNT a été déposé. Les lignes de code suivantes vont poser un bloc de TNT légèrement à côté de ton joueur afin qu'il ne lui tombe pas sur la tête ! Le cratère que tu vas provoquer mesure 20 blocs de longueur (10 à gauche et 10 à droite de Steve). Ses dimensions sont calculées dans la fonction `setBlocks()`, comme ceci :

```
def bombe(x, y, z):
    mc.setBlock(x+1, y, z+1, block.TNT.id)
    for t in range(6):
        display.write(str(5-t))
        time.sleep(1)

mc.postToChat("BOUM !")
mc.setBlocks(x-10, y-5, z-10, x+10, y+10, z+10, block.AIR.id)
```

7. Rédige la boucle de jeu principale de façon à ce qu'elle attende l'activation du détonateur puis qu'elle jette une bombe. Souviens-toi que ton bouton se connecte à la bande de 0 volt lorsque tu appuies dessus et que tu dois donc comparer le port GPIO avec `False` pour qu'il détecte le déclenchement du bouton.

```
try:
    while True:
        time.sleep(0.1)
        if GPIO.input(BOUTON) == False:
            pos = mc.player.getTilePos()
            bombe(pos.x, pos.y, pos.z)
finally:
    GPIO.cleanup()
```

Sauvegarde ton programme et exécute-le. Si tu as un Raspberry Pi, n'oublie pas de saisir l'instruction `sudo python detonateur.py` dans la fenêtre LXTerminal.

Trouve un endroit dans Minecraft et appuie sur le gros bouton rouge de ton détonateur, puis cours t'abriter quelque part pour sauver ta peau ! La figure 5-17 te montre à quoi ressemble l'énorme cratère après l'explosion.

FIGURE 5-17 Un gigantesque cratère formé par une explosion dans Minecraft

La taille du cratère est déterminée par les nombres contenus dans la fonction `setBlocks()`. Mais penses-tu que c'est le meilleur moyen de la définir ? Comment t'y prendrais-tu si tu souhaitais doubler sa taille et combien de lignes de ton script modifierais-tu pour cela ? Tu me prends peut-être pour un fou, mais je suis persuadé que tu peux trouver un moyen de modifier très simplement la taille du cratère.

DÉFI

Crée une explosion du tonnerre en ajoutant des lignes de géorepérage à ton programme, comme dans le chapitre 2, afin de détecter si ton joueur se trouve dans le périmètre de l'explosion.

Utilise une fonction `mc.player.setTilePos()` pour propulser ton joueur dans le ciel au moment de l'explosion, si celui-ci se trouve dans le rayon de la détonation.

DÉFI

Ajoute trois nouveaux boutons à ton circuit, comme tu l'as fait pour ton premier bouton. Rédige des codes supplémentaires pour provoquer des événements différents à l'aide de ces boutons. Par exemple, tu peux publier des messages aléatoires dans le chat Minecraft, téléporter ton joueur dans des cachettes secrètes, poser des blocs de types différents à ton emplacement, ou encore bâtir une maison en face de Steve. Pour cela, tu peux utiliser les numéros de broches GPIO suivants et les attribuer aux trois autres boutons.

BOUTON	RASPBERRY PI	ARDUINO
BOUTON2	14	5
BOUTON3	23	2
BOUTON4	24	3

Laisse ta plaque de montage telle quelle une fois que tu auras terminé le chapitre. Martin utilisera cette même configuration dans le jeu du chapitre 9, et tu l'utiliseras aussi dans le chapitre bonus, que tu peux télécharger sur le site Internet associé (www.editions-eyrolles.com/dl/14292).

Tableau de références

Configurer les broches GPIO

```
import RPi.GPIO as GPIO # RaspberryPi
import anyio.GPIO as GPIO # Arduino
GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.OUT) # sortie
GPIO.setup(5, GPIO.IN) # entrée
```

Interpréter et contrôler des broches GPIO

```
GPIO.output(5, True) # ouvert (3.3V)
GPIO.output(5, False) # fermé (0V)
if GPIO.input(6) == False:
    print("enfoncé")
```

Remettre en état les broches GPIO après utilisation

```
try:
    fais_quelquechose() # rédige le
                        # code ici
finally :
    GPIO.cleanup()
```

Utiliser le module de l'afficheur 7 segments

```
import anyio.seg7 as display
BROCHES_LED = [10,22,25,8,7,9,11,15]
ON = False # anode commune
ON = True # cathode commune
display.setup(GPIO, BROCHES_LED, ON)
```

Afficher des caractères sur l'afficheur 7 segments

```
display.write("3")
display.write("A")
display.write("up")
```

Autres fonctions de l'afficheur 7 segments

```
display.setdp(True) # séparateur
                    # décimal activé
display.setdp(False) # séparateur
                     # décimal
                     # désactivé
display.clear()
display.pattern([1,1,1,1,1,1,1,1])
```

Explorer d'autres pistes de programmation impliquant des circuits électroniques

Dans ce chapitre, tu as relié le monde de Minecraft au monde réel et tu as élargi ton propre horizon en t'initiant à la discipline fascinante de l'électronique. En effet, tu as appris à détecter et à prendre le contrôle d'objets du monde réel. Tu as employé tes nouvelles connaissances pour faire clignoter des LED, faire apparaître des formes sur des afficheurs 7 segments et détecter lorsque des boutons sont actionnés. Mieux encore, tu as réussi à t'échapper des tréfonds du monde de Minecraft. Avec ces nouveaux acquis, tu peux désormais créer tes propres manettes de jeu et périphériques d'affichage !

- Il n'y a pas assez de place dans ce livre pour réaliser d'autres jeux avec des circuits électroniques. Mais dès que j'ai commencé à penser à des idées de jeu, elles fusaient tellement dans mon esprit que je n'ai pas su m'arrêter ! Impossible, par exemple, de te laisser terminer ce livre sans t'enseigner le jeu de l'Ascenseur Minecraft. C'est

pourquoi je l'ai inclus dans le chapitre bonus, que tu peux télécharger sur le site Internet du livre (www.editions-eyrolles.com/dl/14292). Il faut absolument que tu l'essayes ! Quant à moi, je ne compte plus les heures que j'ai passées à faire défiler l'ascenseur à toute vitesse, des bas-fonds du monde de Minecraft jusqu'aux frontières de l'espace. Je me suis même amusé à construire de gigantesques tours pour l'abriter ! Un jour, je construirai la réplique parfaite du gratte-ciel Shard (http://www.the-shard.com) autour de l'ascenseur, mais qui sait, peut-être que tu y parviendras avant moi !

- Un jour, l'un des élèves de notre club Raspberry Pi a eu une formidable idée : pourquoi ne pas concevoir une carte au trésor sur une plaque et l'orner de petites diodes ? Il suffirait de reproduire la carte du monde de Minecraft en dur et de l'orner de LED pour indiquer des défis à relever. Dès que le joueur aurait terminé un défi, la LED concernée s'allumerait pour indiquer sa victoire. J'aimerais tellement voir ce jeu se concrétiser ! J'espère qu'un jour je trouverai une vidéo YouTube le présentant. Seras-tu le premier à relever cet incroyable défi ?
- Un joueur a eu l'idée géniale de connecter un accéléromètre électronique à son ordinateur afin qu'il puisse détecter les mouvements de son bras et les reproduire à l'aide d'un bras mécanique. Regarde ce que ça donne sur cette vidéo : http://www.youtube.com/watch?v=_KGc9vlOrNk. Essaye toi aussi de raccorder un accéléromètre à ton ordinateur et de trouver un moyen d'interagir avec Minecraft en bougeant les bras ! Si tu y parviens, tu pourras présenter ton projet dans un salon d'exposition local et inspirer d'autres personnes à s'initier au monde fascinant de la robotique dans Minecraft !

Niveau terminé : ça y est, tu as brisé les barrières qui séparaient le monde virtuel de Minecraft du monde réel ! Tu es maintenant l'un des pionniers de cette discipline de pointe que représente l'électronique dans Minecraft. Ton expérience de jeu n'a désormais plus aucune limite !