N° 08 - DEC 2012

Venez sur Kickstarter
http://kck.st/TvkdvG
pour le MagPi papier!

The MagPi™

Un magazine pour les utilisateurs de Raspberry Pi

Attraper le Père Noël grâce à la domotique

Gagnez un Raspberry Pi avec 512 Mo

Le MagPi vous souhaite un Joyeux Noël

Dans ce numéro...

- Skutter
- Nanpy
- Jauge Pi
- Pibow
- CESIL Pi
- C++
- Ada
- MySQL
- Le repaire du Python

http://www.themagpi.com

The MagPi™

Bienvenue dans cette huitième édition du magazine MagPi,

C'est Noël ! Dans ce numéro, nous espérons vous inciter à essayer quelques projets festifs après vous être goinfré à ras-bord avec le repas de Noël.

Dans l'édition de ce mois-ci, nous vous faisons découvrir un projet simple de domotique vous permettant de contrôler l'éclairage et des appareils dans votre maison grâce à la puissance du Pi ! Juste à temps pour attraper M. Noël ! Nous poursuivons votre projet Skutter avec un article de Morphy sur l'ajout de roues à votre base. Gordon nous enseigne comment allumer un arbre de Noël, nous en apprenons davantage sur l'utilisation du Pi pour contrôler un Arduino et Ben explique comment contrôler des servos reliés au Pi via Internet ! Si ce n'est pas assez, vos anciennes séries préférées sont là ainsi qu'une introduction au SQL.

Comme toujours, nous avons quelques cadeaux à vous faire gagner dans notre magazine mensuel. Le MagPi voudrait encore une fois dire un grand merci à PC Supplies qui s'est surpassé ce mois-ci en proposant comme prix un Raspberry Pi 512 Mo !

En plus de tout cela, nous avons quelques nouvelles excitantes ce mois-ci pour vous. À partir du 1er décembre, nous, au MagPi, sommes excités d'être en mesure d'offrir à nos lecteurs la possibilité d'avoir une version imprimée de tous les huit numéros du magazine ! C'est quelque chose que nos lecteurs nous demandent en permanence. Tous les huit numéros seront magnifiquement enveloppés dans un classeur MagPi édition limitée, ce qui en fera un superbe cadeau pour vous-même ou vos proches de tout âge. Pour plus d'informations sur ce sujet, merci d'aller sur www.kickstarter.com/projects/themagpi/the-magpi-magazine-from-virtual-to-reality

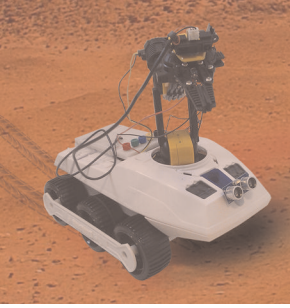
Au nom de toute l'équipe, encore merci pour tout votre soutien. Nous vous souhaitons un fabuleux Noël et nous nous reverrons pour la Nouvelle Année (1er février). Bien que nous n'ayons pas réussi à le placer dans ce numéro, vous serez peut-être intéressé par www.xmas4all.co.uk , site à partir duquel vous serez en mesure de contrôler leurs lumières de Noël pilotées par un Raspberry Pi !

Ash Stone
Rédacteur en chef du MagPi

Rédaction du MagPi
Ash Stone - *Rédactrice en chef/Administratrice*
Chris "tzj" Stagg - *Rédacteur/Photo/Maquette*
Colin Deady - *Rédacteur/Maquette*
Jason "Jaseman" Davies - *Site web/Maquette*
Matt "0the0judge0" - *Site web/Administrateur*
Meltwater - *Photographe/Maquette/Admin.*
Aaron Shaw - *Maquette/Graphisme*
Ian McAlpine - *Maquette/Graphisme*
Lix - *Maquette/Graphisme*
Sam Marshall - *Maquette/Graphisme*
W. H. Bell - *Maquette*

Rédacteurs invités
Bodge N Hackitt - *Rédacteur*
Geoff Johnson - *Rédacteur*
Andrea Stagi - *Rédacteur*
Ben Schaefer - *Rédacteur*
Gordon Henderson - *Rédacteur*
Alex Kerr - *Rédacteur*
Luke Guest - *Rédacteur*
Richard Wenner - *Rédacteur*

Sommaire



Ajout d'une base motorisée

Partie 2

DIFFICULTÉ : ÉLEVÉE

Un simple "pont en H" de commutation

Dans le dernier article, nous avons vu quelques moyens physiques pour ajouter des moteurs à un robot et rechercher comment adapter certains jouets électroniques motorisés pour s'en servir éventuellement comme bases.

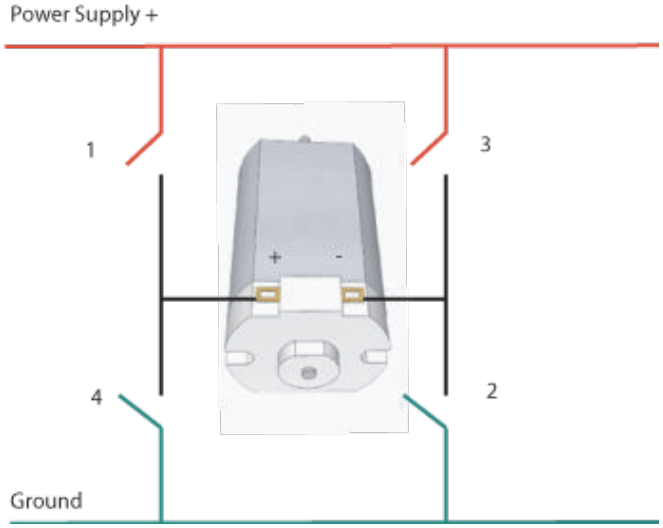
Dans cet article, je vais commencer à vous expliquer comment construire votre propre module de contrôle pour moteur électronique CC et écrire un programme de contrôle rudimentaire pour celui-ci.

Nous allons démarrer en réexaminant le moteur CC classique que nous avons abordé dans un article précédent.

Pour faire en sorte que le moteur tourne dans un sens, nous appliquons une source d'alimentation entre les bornes + et - du moteur et pour le faire aller en sens inverse, nous échangeons simplement les bornes d'alimentation.

Le module de contrôle du moteur que nous allons créer devra être un circuit capable d'échanger les bornes d'alimentation de manière électronique. Cela peut être réalisé avec un "circuit de pont en H".

Ce schéma montre une version simplifiée d'un tel circuit. Fermer les commutateurs 1 et 2 relie effectivement la sortie positive de



l'alimentation à la borne + du moteur et la masse à la borne - et le moteur se met à tourner. Sinon, fermer les commutateurs 3 et 4 connecte la masse à la borne + et le positif à la borne - et le moteur tourne à l'envers.

Il existe des situations potentiellement dangereuses lorsque les commutateurs 1 et 4 ou 3 et 2 sont fermés. Cela créerait un court-circuit entre l'alimentation + et la masse ce qui peut être très problématique, pour le moins qu'on puisse dire. Toutes les précautions doivent être prises lors de la vérification du circuit pour s'assurer que cette situation ne peut jamais se produire.

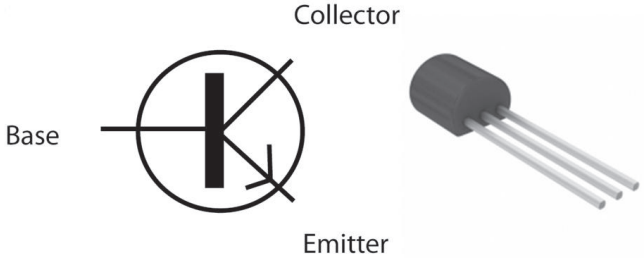
En réalité, nous ne pouvons pas avoir quatre commutateurs ouverts/fermés comme ça dans la mesure où nous devons contrôler le circuit en utilisant le GPIO du Raspberry Pi.

Il y a des solutions électroniques à cela. Une des possibilités consiste à utiliser des relais électromagnétiques pour fermer ces "commutateurs", cependant le Raspberry Pi n'est pas capable de fournir assez de puissance depuis le GPIO pour activer directement un tel relais sans devoir placer entre les deux quelque chose comme un transistor. Cela nous conduit à la deuxième possibilité qui est de simplement utiliser quelques transistors comme commutateurs.

Des transistors comme commutateurs

Le transistor est sans doute l'invention électronique la plus importante jamais créée. Son développement est à l'origine de tout, depuis les lecteurs de musique portables jusqu'au processeur utilisé dans le Raspberry Pi.

Nous allons nous intéresser aux transistors



de type NPN. Ce composant dispose de trois pattes appelées base, collecteur et émetteur.

Connecter une alimentation entre le collecteur et l'émetteur permet au transistor d'être employé en tant que commutateur. Sans une connexion à la base, sa "résistance interne" est très élevée et le commutateur est "ouvert".

Si nous appliquons un courant à la base du transistor alors la résistance interne va baisser d'une quantité correspondante et davantage de courant va circuler du collecteur vers l'émetteur.

Le transistor est capable de voir sa résistance interne varier très rapidement, des dizaines de milliers de fois par seconde. (C'est cette caractéristique qui leur permet d'être utilisés comme amplificateurs).

La quantité dont la résistance interne du transistor est affectée par le courant est définie par un rapport appelé "gain en courant" et il y est fait référence par "h_{FE}".

Dans notre cas, nous souhaitons fournir un courant à la base de manière à rendre la résistance interne égale à zéro - à l'instar d'un commutateur fermé. On parle de "transistor saturé" et il existe une équation qui nous dit quel courant nous devons appliquer à la base pour que cela arrive :

$$I_B = I_C / h_{FE}$$

où I_C est le courant appliqué au collecteur et I_B celui de la base. Afin de trouver quel est ce courant, il est nécessaire de mesurer celui qui est pris par le moteur. Cela signifie qu'une expérience est requise !

Pour cela vous aurez besoin de votre base motorisée (dans mon cas il s'agit du Big Trak modifié), une alimentation (quelques piles) et un multimètre.

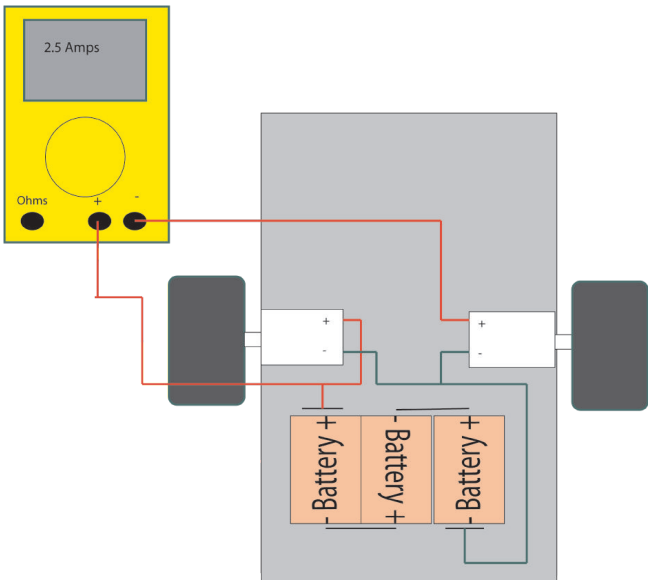
Si vous n'avez pas de multimètre pour l'instant, c'est un outil essentiel pour tous ceux qui souhaitent se lancer dans l'électronique et qui vous permet de prendre une large gamme de mesures, parmi lesquelles tension, intensité, résistance, capacité et h_{FE}. Maplins en vend un pour £7.99 (CODE : N20AX).

Il est possible de trouver de bons multimètres pour moins de dix livres chez divers commerçants.

Les moteurs CC tirent différentes quantités de courant selon les différentes conditions. Si un moteur est en "roues libres", alors il ne prendra comparativement que peu de courant.

Par contre, un moteur au "point mort" (un moteur que l'on empêche de tourner) va tirer un courant extrêmement élevé. Plus il est difficile de faire tourner un moteur, plus il prendra de courant. Dans notre cas, nous voulons mesurer le courant que les moteurs utilisent quand notre base robotique roule par terre. Une façon d'obtenir avec précision cette mesure est de faire avancer la base sur le sol et de mesurer le courant pris à ce moment. Voici la méthode que j'ai utilisée avec mon Big Trak :

Connectez le multimètre en série entre les piles/alimentation et un des moteurs du Big Trak.



Le second moteur doit aussi être branché sur l'alimentation et actif sinon il n'y aura qu'un seul moteur pour essayer de faire avancer le Big Trak entier et cela aura pour résultat une mesure imprécise. Cependant, nous n'avons besoin de mesurer le courant utilisé que sur un seul des deux moteurs identiques.

Ajoutez au Big Trak un certain poids de manière à ce que le poids total soit approximativement celui du robot terminé. Pour le Skutter, cela inclut l'ajout du bras robot.

Complétez le circuit entre les piles et le moteur, en ajoutant le multimètre en série comme indiqué. Pendant que le Big Trak roule sur le sol, prenez une mesure du courant consommé. Sous la charge prévue pour le Skutter, en utilisant cette méthode, un des deux moteurs du Big Trak devrait consommer un courant de 2,5A.

ATTENTION : lorsque le moteur a calé, la mesure du courant consommé a montré qu'il était d'environ 20A.

Choix du bon transistor

Chaque transistor doit recevoir du courant sur le collecteur ; ce courant est le même que celui consommé par les moteurs. C'était

2,5 A pour le Big Trak (mais il sera évidemment différent si vous avez basé votre robot sur d'autres moteurs). Nous devons maintenant trouver des transistors qui vont se comporter comme des commutateurs quand ils laisseront passer ce courant.

Voyons de nouveau l'équation de saturation du transistor :

$$I_B = I_C / h_{FE}$$

Deux des valeurs de cette équation sont désormais connues. Nous avons 2,5 A pour I_C et nous savons que chaque broche GPIO du Raspberry Pi est capable de délivrer au maximum 15 mA (soit 0,015 A). Il faut trouver une valeur minimale de h_{FE} (gain en courant) en utilisant ces deux valeurs ; cela peut être obtenu en réarrangeant l'équation ainsi :

$$h_{FE} = I_C / I_B$$

Pour qu'un transistor puisse faire office de commutateur pour un moteur du Big Trak en utilisant le GPIO comme déclencheur, cela doit être :

$$h_{FE} = 2.5 / 0.015$$

Par conséquent, il est nécessaire d'avoir un transistor dont le gain est d'au moins 167. De plus, il doit pouvoir supporter 2,5 A. Pour des périodes de durée très courte, le circuit que nous concevons doit également être en mesure de gérer jusqu'à 20 A car les moteurs peuvent nécessiter une telle quantité lorsqu'ils sont bloqués.

Un transistor pouvant correspondre à ces critères est le BCU81 (RS Components 217-8199). Ce dernier peut accepter un courant continu de 3 A et possède un gain en courant minimum (h_{FE}) de 210.

Un circuit de contrôle de moteur à transistors

Le gain en courant du transistor BCU81 est 210. Cela implique que nous n'avons besoin de fournir que 12 mA depuis le GPIO :

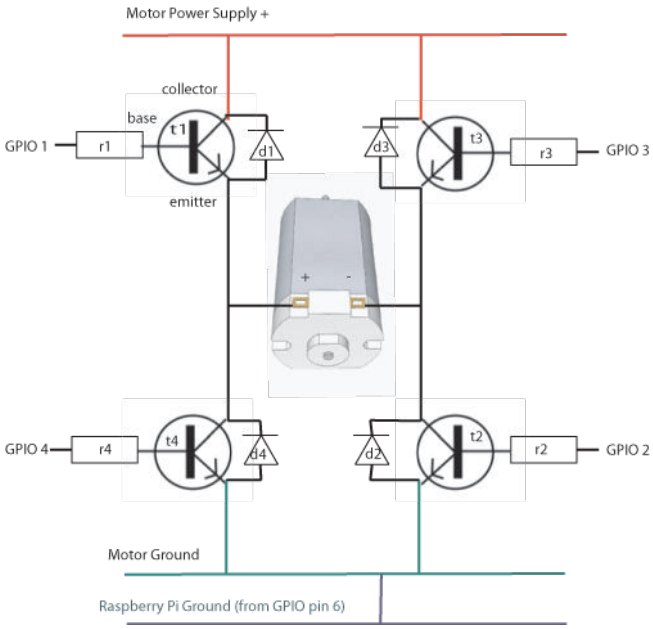
$$I_B = I_C / h_{FE}$$
$$2.5 / 210 = 0.012$$

Nous pouvons utiliser une résistance limitant le courant en série avec les sorties GPIO pour s'assurer que nous ne prendrons que 12 mA. Pour calculer la valeur de cette résistance il faut employer la loi d'Ohm : Résistance = Tension / Intensité.

Le GPIO utilise 3,3 volts (désignés par 3v3) si bien que nous pouvons nous servir de cette donnée pour déterminer les valeurs nécessaires à nos résistances de limitation de courant : 3,3 / 0,012 = 275 Ohms. Habituellement nous devrions tenir compte d'une chute de tension dans cette équation, mais comme un transistor ne voit pas de chute de tension significative lorsqu'il est saturé, ce n'est pas utile ici, et ce calcul dans notre cas reste très simple.

Une valeur de résistance communément disponible est 270 Ohms. Utiliser cette valeur au lieu des 275 Ohms n'a qu'un effet négligeable sur le courant du GPIO, et par conséquent, pour des raisons de simplicité, il est acceptable d'utiliser cette résistance.

Il s'agit alors d'un circuit pour un contrôleur de moteurs basé sur un pont en H opérationnel pour un Raspberry Pi (il y a



quand même un avertissement de sécurité le concernant, prenez-en connaissance en fin d'article).

Quatre diodes ont été ajoutées ; elles sont orientées dans la direction opposée au flux du courant. Un moteur CC va aussi générer du courant électrique, en particulier au moment de son extinction. Cela est désigné par le terme "courant de retour " et peut être suffisamment élevé pour endommager les transistors. Placer ces diodes à contre-sens permet à ce courant d'être évacué en toute sécurité.

Vous pouvez utiliser ce circuit si vous possédez le même modèle de Big Trak, ou alors, si votre base est équipée de moteurs différents, adaptez les calculs décrits précédemment afin de trouver les transistors et résistances appropriées.

Un aspect important de ce circuit est que la masse du Raspberry Pi est partagée avec celle de l'alimentation du moteur. Avoir une "masse commune" comme dans ce cas signifie que nous pouvons avoir deux périphériques distincts, alimentés par deux alimentations séparées et qui fonctionneront néanmoins ensemble sur le même circuit.

Enfin, nous pouvons voir que pour faire tourner les moteurs, il faut activer GPIO 1 et GPIO 2. Pour les faire tourner en sens inverse, nous devons activer GPIO 3 et GPIO 4. Les articles du MagPi "Sous contrôle" contiennent les informations nécessaires concernant la façon de programmer cela et si vous réalisez ce circuit, alors vous pourriez envisager de concevoir ce programme comme défi pour le mois prochain !

Si, comme moi, vous fabriquez un robot différentiel équipé de deux moteurs, vous aurez besoin de deux de ces circuits, un pour chaque moteur. Cela implique qu'au total vous utiliserez jusqu'à 8 de vos GPIO, ce qui ne laisse pas beaucoup d'entrées ou de sorties de libres pour le reste du robot.

Avertissement de sécurité :

Il peut arriver à certains moments que les deux moteurs soient dirigés, quatre broches du GPIO vont alors envoyer 12 mA chacune. Cela représente un total de 48 mA. Le courant maximum total que le connecteur GPIO entier puisse fournir à un moment donné est de 51 mA.

Alimenter un pont en H directement depuis les broches GPIO atteint quasiment les limites absolues pour agir en terme de sécurité. Afin de pouvoir utiliser toute autre sortie, il serait d'abord nécessaire de mettre toutes les broches GPIO du pont en H à l'état bas pour permettre au courant d'être envoyé ailleurs.

Pour conclure, le courant que le GPIO est capable de fournir étant relativement bas, nous devons chercher un transistor avec un hFE élevé et pouvant également supporter 3 ampères. De tels transistors sont des solutions réellement onéreuses !

À venir..... ?

La prochaine fois, je décrirai un moyen d'augmenter considérablement le nombre de GPIO de votre Raspberry Pi grâce au bus I2C et à un circuit bon marché appelé MCP23008. Cette méthode résoudra le problème de la limitation du courant et permettra à votre robot de bénéficier d'encore plus d'entrées et de sorties tout en rendant possible l'utilisation dans notre circuit de transistors beaucoup moins chers et de h_{FE} plus bas.

BasicGPIOHBridgeControl.py :

```
#! /usr/bin/python
import time
import RPi.GPIO as GPIO

#----------------------------IMPORTANT ------------------------------
#IF GPIO 0, 1 est "1" THEN GPIO 2, 3 doit être "0" ELSE transistor
en court-circuit !
#--------------------------------------------------------------------

GPIO.setup(1, GPIO.OUT)
GPIO.setup(2, GPIO.OUT)
GPIO.setup(3, GPIO.OUT)
GPIO.setup(4, GPIO.OUT)

#Fait tourner le moteur en avant pendant 3 secondes
GPIO.output(1, True)
GPIO.output(2, True)
time.sleep(3)
#Arrête le moteur
GPIO.output(1, False)
GPIO.output(2, False)
#Fait tourner le moteur à l'envers pendant 3 secondes
GPIO.output(3, True)
GPIO.output(4, True)
time.sleep(3)
#Arrête le moteur
GPIO.output(3, False)
GPIO.output(4, False)
```

REMARQUE : Si le robot utilise deux moteurs, le programme aurait besoin d'un jeu de 4 GPIO supplémentaires pour contrôler un second pont en H.

Article de Bodge N Hackitt

Automatisez votre maison avec un Raspberry Pi... et prenez le Père Noël en flagrant délit !

Cet article décrit un moyen facile à construire, économique, et par-dessus tout, sûr, pour contrôler avec un Raspberry Pi les appareils reliés au secteur. Rien dans ce projet ne conduit à aller vers des tensions dangereuses. La soudure se limite juste à quelques points et pour la partie logicielle, le code source peut être téléchargé.

L'histoire

J'avais acheté quelques prises télé-commandées sur http://www.amazon.co.uk (recherchez Status remote control socket). [Ed : J'ai vu quelque chose de très ressemblant dans Home Depot] Je les ai essayées avec quelques appareils au

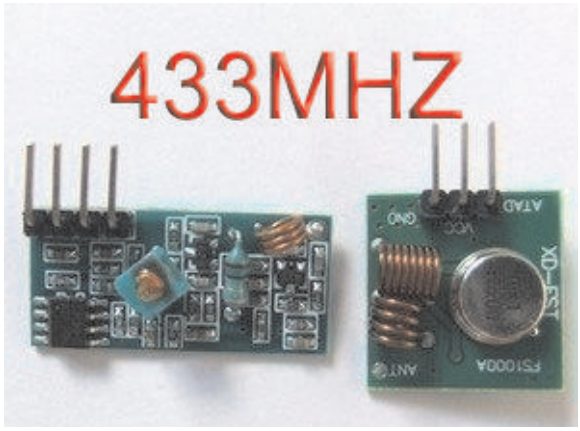
hasard, rangées dans un placard puis les ai oubliées... mais le Raspberry Pi m'a donné l'inspiration pour faire quelque chose d'utile avec.

J'ai décidé d'utiliser le Raspberry Pi pour remplacer la télécommande et comme il est programmable, nous pouvons en étendre les capacités. Que diriez-vous d'allumer ou éteindre les guirlandes du sapin de Noël à une heure déterminée, d'utiliser votre smartphone pour allumer votre bouilloire dès que vous arrivez à la maison ou de faire clignoter votre lampe de chevet quand le Père Noël marchera sur le tapis à capteur de pression placé devant la cheminée la veille de Noël ?

Décodage des codes distants

La première étape consistait à identifier les signaux utilisés par les prises, par conséquent je me suis dit que je pourrais essayer de les imiter avec le Raspberry Pi. La communication entre les prises et leur commande est basée sur la radio ce qui permet un fonctionnement même sans être dans la ligne de mire. Cela permet également de contrôler les prises placées dans différentes pièces depuis un seul endroit.

Comme l'autocollant au dos des prises indique 433,92 MHz, j'ai cherché "récepteur 433 MHz" sur eBay afin de trouver un

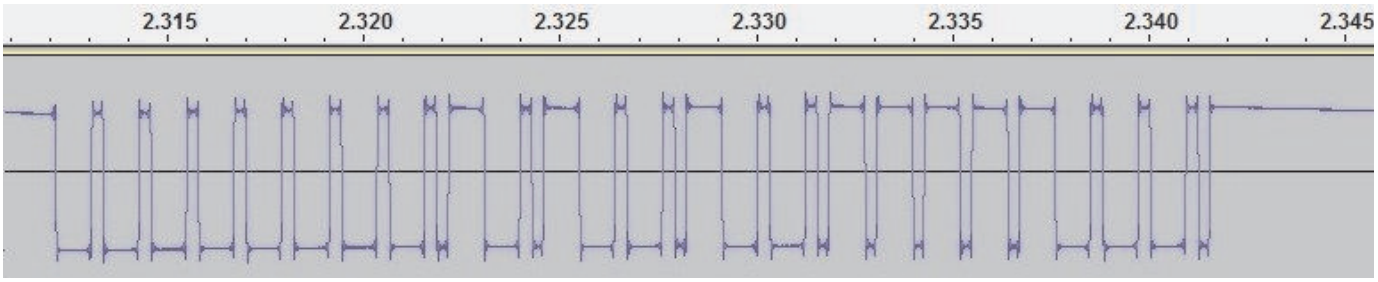


récepteur approprié. J'ai ainsi trouvé un émetteur et un récepteur pour projets Arduino à seulement 1,60 £ (1,99 $), emballage et frais de port compris depuis la Chine !

J'ai dû souder un câble d'antenne de 7" (17 cm) sur le récepteur et l'émetteur. La longueur de l'antenne représente 1/4 de la longueur d'onde 433,92 MHz. Vous pouvez utiliser une calculatrice en ligne comme http://www.csgnetwork.com/freqwavelengthcalc.html.

Pour lire le code émis par la télécommande, j'ai branché le module récepteur sur la prise microphone d'un ordinateur. J'ai utilisé une alimentation 5V séparée, mais il y a une sortie 5V sur le connecteur GPIO du Raspberry Pi qui peut servir à cet usage. La sortie du module récepteur est du 5V numérique, ce qui ne convient pas à une connexion directe sur la prise audio de l'ordinateur. J'ai donc branché la sortie du module à la prise microphone de mon portable via une résistance de 1M Ohm. J'ai assemblé le circuit sur une platine, mais vous pouvez utiliser l'autre extrémité du câble de lecteur de disquettes cité dans la section Matériel.

Audacity (http://audacity.sourceforge.com) est un excellent gratuiciel pour examiner des signaux de ce genre. Une fois satisfait par le niveau d'enregistrement que je trouvais relativement correct, j'ai démarré l'enregistrement et appuyé sur l'un des boutons de la télécommande. Ayant arrêté l'enregistrement, j'ai été en mesure d'agrandir



la région dans laquelle se trouvait le signal de la commande. Le signal se répétait encore et encore jusqu'à ce que le bouton soit relâché. Les impulsions étroites semblaient durer environ 0,25 ms tandis que les larges duraient 3 fois plus longtemps.

J'ai créé cette chaîne binaire à partir de la forme d'onde. Chaque bit représente 0,25 ms avec 1 = impulsion haute et 0 = impulsion basse. Pour faciliter la lecture, les impulsions ont été séparées par des tirets.

```
11111-000-1-000-1-000-1-000-1-000-1-000-
1-000-1-000-1-0-111-000-1-0-111-000-1-
000-1-0-111-000-1-000-1-0-111-0-111-0-
111-0-111-0-111-000-1-000-1-000-1-0-
111111
```

Pour le projet, l'heure est venue d'agir ou de s'arrêter. Si vous êtes arrivés à ce point et qu'il vous est impossible de trouver un motif qui se répète en apparaissant lorsque vous appuyez sur les boutons de la télécommande, cela signifie que vos prises peuvent ne pas utiliser de simples signaux AM sur lesquels repose ce projet.

Envoi d'un signal

Pour pouvoir exploiter les données capturées, j'ai branché l'émetteur sur le Raspberry Pi. Comme il est prévu de le laisser connecté, il est par conséquent alimenté par la broche +5V du GPIO. J'ai relié GPIO 7 depuis le Raspberry Pi à la broche de données et la masse de l'émetteur à celle du connecteur GPIO du Raspberry. Les signaux 3,3V sont suffisants pour piloter l'émetteur, quoique je n'ai découvert ça qu'en faisant le test.

Linux étant un système d'exploitation multi-tâches, quelque chose peut très bien utiliser le CPU au mauvais moment, c'est pourquoi mon programme envoie la séquence de bits à 10 reprises dans l'espoir qu'au moins une soit transmise correctement.

Quand j'ai lancé mon logiciel sur mon Raspberry Pi pour la première fois, j'ai capturé le signal avec Audacity. J'ai pu voir que la surface de la forme d'onde était correcte mais à l'envers. Inutile de dire que la prise ne réagissait pas. J'ai donc inversé tous les bits de mon flux de sortie et relancé le code de test. Cette fois la prise s'est allumée ! Il ne me restait plus qu'à décoder les autres boutons.

Matériel

Le connecteur pour le GPIO et le câble que j'ai utilisés ont été faits à partir de câbles pour lecteur de disquettes de vieux PC. Ils sont plus larges que le connecteur GPIO du Raspberry Pi, mais vous pouvez placer la prise sur les broches du GPIO en vous calant sur un bord. Bien sûr, ça ne sera pas possible si votre Raspberry Pi est dans un boîtier.

Trois fils seulement sont nécessaires pour le GPIO : +5V, GND et GPIO 7 (CE1) pour la broche que j'utilise pour contrôler l'émetteur.

Un tout petit morceau de plaque d'essai est placé à l'autre extrémité de la nappe. Soudé

sur cette plaque, se trouve le connecteur de l'émetteur, fabriqué à partir d'un support de circuit intégré qui a été découpé.

Logiciel

J'ai suivi le code GPIO de l'exemple trouvé sur http://www.elinux.org/Rpi_Low-level_peripherals. Comme celui-ci accède directement à la mémoire, il est nécessaire de l'exécuter en tant qu'utilisateur root. Pour me faciliter la vie, je me suis logué en root pour le tester et le développer. Toutes les commandes données ici impliquent que vous ferez de même.

Tout se passe en ligne de commande, donc, à moins que votre système ne soit configuré pour démarrer directement l'interface graphique, il suffira de rester simplement en mode texte.

Pour créer un répertoire dans lequel stocker le code source et le fichier exécutable pendant la durée du développement, tapez les commandes suivantes :

```
$ mkdir gpio
$ cd gpio
```

Mon code source peut être téléchargé depuis http://www.hoagieshouse.com. Suivez simplement les liens et enregistrez le fichier dans le répertoire que vous venez juste de créer. Il prend 2 paramètres, le canal et on ou off. Vous devrez éditer le code pour remplacer mes codes de télécommande par les vôtres. Pour éditer le code, tapez :

```
$ nano switch.cpp
```

Soyez attentif à conserver les guillemets autour des codes dans le source. Après les avoir modifiés, quittez en appuyant sur <CTRL>-<X>, répondez Y à la demande d'enregistrement du fichier et acceptez le nom de fichier switch.cpp. Pour construire le fichier exécutable, tapez :

```
$ g++ -o switch switch.cpp
```

Testez-le avec vos prises en lançant la commande :

```
$ ./switch 1 on
```

Si cela fonctionne, vous souhaiterez sans doute pouvoir le lancer avec n'importe quel utilisateur. Utilisez les commandes suivantes pour le faire :

```
$ chmod +s switch
$ mv switch /usr/bin/
```

Pour planifier l'allumage et l'extinction des prises à des moments précis, vous pouvez utiliser quelque chose appelée des tâches cron. Tapez simplement :

```
$ crontab –e
```

Vous serez alors en mesure d'éditer un fichier qui contrôle les tâches planifiées. Le format est décrit dans le fichier, mais pour faire un test, ajoutez ces lignes en fin de fichier :

```
0 *  * * * switch 1 on
10 *  * * * switch 1 off
```

Cela va allumer la prise 1 pendant dix minutes au début de chaque heure.

Jusque-là, ce n'est pas très convivial. Une interface web serait beaucoup mieux. La mienne permet d'allumer ou éteindre les 4 canaux, mais je ne suis pas allé assez loin pour y ajouter la planification. Premièrement, installez mini-httpd pour agir en tant que serveur web. Pour cela, tapez :

```
$ apt-get install mini-httpd
```

Les fichiers de l'interface web peuvent être téléchargés depuis http://www.hoagieshouse.com. Dans le fichier zip se trouvent le fichier de configuration pour mini-httpd et le répertoire /var/www dans lequel j'ai placé les pages web et les programmes cgi. Aucune modification ne devrait être nécessaire, mais il faut les placer aux bons emplacements. Le fichier HTML utilise une requête AJAX pour lancer le script CGI avec les paramètres de canal et on/off. Ce script récupère simplement les paramètres à partir de la requête et appelle le programme switch avec.

Conclusion

Maintenant que vous avez les bases logicielles pour contrôler à distance plusieurs appareils branchés sur secteur avec votre Raspberry Pi, je vous laisse comme exercice d'imaginer quels autres déclencheurs pourraient être reliés au GPIO pour allumer ou éteindre des appareils. Oh, si vous attrapez le Père Noël, dites-lui s'il vous plaît que j'aimerais l'ensemble Noël du MagPi imprimé trouvé sur htt://www.kickstarter.com/projects/themagpi/the-magpi-magazine-from-virtual-to-reality.

Liste des composants Maplin
VY48C - TX/RX 433MHz £9.99
M1M - Résistance 1M Ohm £0.29
DG41U - Câble pour lecteur de disquettes £3.99

Article de Geoff Johnson

CONCOURS DE DÉCEMBRE

The MagPi et PC Supplies Limited sont fiers de vous annoncer un prix très spécial pour le gagnant du concours de ce mois.

Le prix de ce mois est le nouveau Modèle B 512 Mio du Raspberry Pi, une alimentation 1A 5V et un boîtier de PCSL !

Les 2e et 3e gagnants recevront chacun un boîtier Raspberry Pi de PCSL.

Pour avoir une chance de participer au concours de ce mois, visitez : **http://www.pcslshop.com/info/magpi**

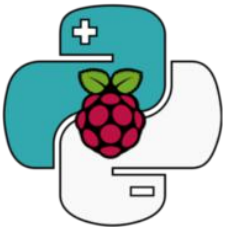
La date de clôture est le 20 décembre 2012. Les gagnants seront avertis dans le magazine du mois prochain et par courriel. Bonne chance !

Découvrez la gamme d'accessoires Raspberry Pi de PCSL sur **http://www.pcslshop.com**

Les gagnants du mois dernier !

Les 5 gagnants du boîtier coloré Raspberry de PCSL sont **Dave Heneghan (Chorley, GB)**, **Dean Hutchison (Glasgow, GB)**, **Dave Carney (Hartlepool, GB)**, **Nigel Laudat (Liverpool, GB)** et **Peter Locastro (Derby, GB)**.

Félicitations. PCSL vous enverra prochainement par courriel la procédure à suivre pour récupérer tous ces fantastiques cadeaux !



Contrôlez votre carte Arduino avec un Raspberry Pi et Python

La puissance du Raspberry et la simplicité de l'Arduino grâce à Python et une simple bibliothèque : Nanpy.

Introduction

Une carte Arduino peut communiquer avec le Raspberry Pi via une liaison par connexion USB. Une interface série virtuelle est créée et utilisée comme une interface normale, avec des lectures/écritures vers le fichier de périphérique série. Pour commencer, connectez votre carte Arduino et tapez :

```
$dmesg | tail
[..]usb 1-1.2: Manufacturer: Arduino[..]
[..]cdc_acm 1-1.2:1.0: ttyACM0: USB ACM device[..]
```

Mon périphérique de carte Arduino Uno est /dev/tty/ACM0 et son pilote est cdc_acm. Les anciennes cartes avec un circuit USB-série sont accessibles grâce à /dev/ttyUSB* :

```
$ls -l /dev/ttyACM*
crw-rw---T 1 root dialout 166, 0 Nov  5 00:09 /dev/ttyACM0
```

Il faut maintenant ajouter l'utilisateur au groupe "dialout" pour lui donner les droits d'accès en lecture/écriture, puis fermer et rouvrir la session pour que la modification prenne effet :

```
$sudo usermod -a -G dialout UTILISATEUR
```

C'est important car Nanpy travaille par l'intermédiaire du fichier de périphérique. Nanpy est un projet open source distribué sous les termes de la licence MIT, et se compose d'une partie serveur (enregistrée sur l'Arduino pour attendre les commandes sur le port série) et d'une bibliothèque 100% Python. Celle-ci permet de communiquer avec l'Arduino branché en USB en utilisant des classes et des méthodes très similaires à celles du framework Arduino. En arrière-plan, quand vous créez/supprimez un objet ou appelez des méthodes avec Python, Nanpy communique via l'USB avec la partie serveur et lui demande de créer/supprimer l'objet correspondant ou d'appeler les méthodes sur l'Arduino : vous pouvez instancier autant d'objets que vous le souhaitez sans vous préoccuper de désallocation et il est également possible de travailler dans un contexte multithreadé. Nanpy a pour objectif de simplifier la vie des développeurs en leur apportant des outils simples, clairs et rapides pour créer des prototypes et des scripts qui interagissent avec l'Arduino, économisant ainsi beaucoup de temps. Pour installer Nanpy, lisez le fichier README. Il faut installer Arduino sur votre portable ou votre Raspberry Pi afin de construire le micrologiciel :

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

Nanpy est actuellement en plein développement et il n'a été testé que sur cartes Uno. Vous pouvez le récupérer depuis la page Pypi (**http://pypi.python.org/pypi/nanpy**) ou Github (**https://github.com/nanpy**).

Voyons maintenant Nanpy à l'œuvre et essayons d'allumer une DEL située sur la 13e broche de l'Arduino :

```
from nanpy import Arduino
Arduino.pinMode(13, Arduino.OUTPUT)
Arduino.digitalWrite(13, Arduino.HIGH)
```

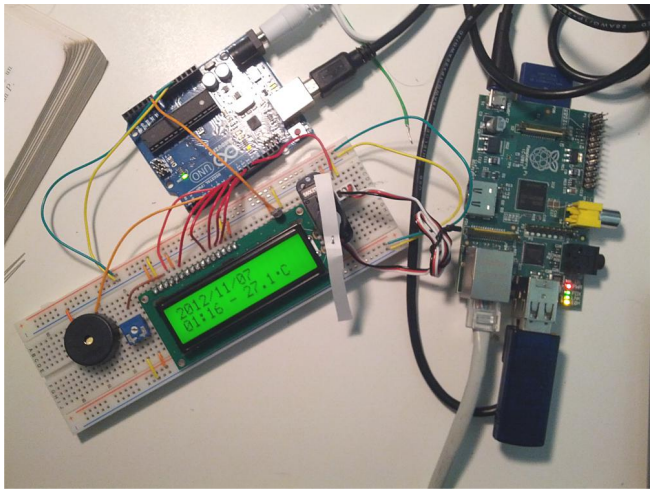
Arduino fournit toutes les principales fonctions, temporisations, lecture et écriture analogique/numérique. Aucune fonction de configuration ou de boucle, uniquement des objets et des appels de méthodes. En réalité, Nanpy prend en charge toutes les principales méthodes Arduino - LCD, Tone, Stepper et autres bibliothèques. Voyons à présent comment utiliser notre écran ACL en mode texte 16x2 sur les broches 6, 7, 8, 9, 10, 11 en écrivant un script Bonjour tout le monde :

```
from nanpy import Lcd
lcd = Lcd([7, 8, 9, 10, 11, 12],[16, 2])
lcd.printString("Bonjour tout le monde !")
```

Juste un mot d'avertissement : Le Raspberry Pi peut ne pas fournir assez de puissance pour alimenter votre Arduino, vous aurez donc peut-être besoin de le brancher sur une source d'alimentation externe.

Le monde extérieur

Je vais maintenant vous montrer comment faire communiquer l'Arduino avec le monde extérieur grâce au Raspberry Pi. Pour que vous compreniez, nous allons fabriquer une horloge moderne, capable de mesurer la température extérieure, avec une alarme initialisée via bluetooth (avec un appareil Android dans ce cas) et une actualisation de la date et de l'heure via un serveur ntp...



Le projet avec ses instructions, une application Android et les composants nécessaires se trouvent ici : **https://github.com/nanpy/eggsamples/tree/master/synclock**. Pour illustrer le fonctionnement de Nanpy dans un contexte multithreadé, ce programme crée un fil d'exécution (thread) pour chaque fonctionnalité, et les affiche tous sur le même ACL. Dans cet article, je ne montre que la partie à l'intérieur des boucles "while True" de chaque méthode "run", par conséquent, je vous recommande de le suivre avec le code source. Commençons par le fil principal, TimeThread, qui lit l'heure depuis notre serveur ntp chaque seconde puis la stocke dans une variable globale, milltime :

```
response = ntplib.NTPClient().request(
      'europe.pool.ntp.org',
      version=3)
milltime = int(response.tx_time)
```

Pour afficher la date et l'heure sur l'ACL, créez un second fil d'exécution, ShowTimeThread :

```
...
self.servo = Servo(12)
...
dt = datetime.fromtimestamp(milltime)
lcd.printString(dt.strftime('%Y/%m/%d'),
      0, 0)
lcd.printString(dt.strftime('%H:%M'),
      0, 1)
self.servo.write(90 + (30 * self.c))
self.c *= -1
```

Chaque seconde, nous récupérons la variable globale milltime, la transformons dans un format lisible puis affichons la date et l'heure sur l'ACL. Comme vous pouvez le constater, printString peut être appelée en spécifiant la position (colonne, ligne) où vous souhaitez voir apparaître la chaîne sur l'ACL. Ensuite, nous actionnons le servo-moteur comme une horloge chaque seconde. La température peut être mise à jour dans un autre fil. Lisons la valeur de notre capteur de température depuis la broche analogique 0 et affichons-la sur l'ACL près de l'heure, toutes les 60 secondes :

```
temp = ((Arduino.analogRead(0) / 1024.0)
      * 5.0 - 0.5) * 100
lcd.printString("- %0.1f\xDFC" % temp,
      6, 1)
```

Voyons à présent comment communiquer avec un téléphone Android pour définir l'alarme avec le bluetooth. J'ai l'ai associé au Raspberry Pi avant de démarrer grâce au guide : **http://wiki.debian.org/BluetoothUser**. Rappelez-vous d'installer python-bluez aussi. Nous utiliserons AlarmClock, une classe thread-safe, pour enregistrer et récupérer la valeur de l'alarme sur disque (voir code). Nous pouvons alors lancer notre communication bluetooth dans un fil AlarmClockThread :

```
...Connexion et init Bluetooth...
cli_sock, cli_info = srv_sock.accept()
cli_sock.send("%d:%d:%d", ck.getAlarm())
try:
    while True:
        data = cli_sock.recv(3)
        if len(data) == 0: break
        ck.setAlarm(ord(data[0]),
                    ord(data[1]),
                    ord(data[2]))
except IOError:
    pass
```

Notre Raspberry Pi se comporte en serveur, attendant une connexion bluetooth : après que cela soit arrivé, il envoie l'heure d'alarme à notre appareil et attend une nouvelle valeur à enregistrer. Dans TimeThread, nous comparons l'heure actuelle et la valeur de l'alarme : si elles sont égales, nous pouvons démarrer un autre fil, PlayAlarmThread, qui joue la note Do pendant 250 ms, à 5 reprises, en utilisant un objet à travers un haut-parleur contrôlé par la 4e broche numérique. Debout !

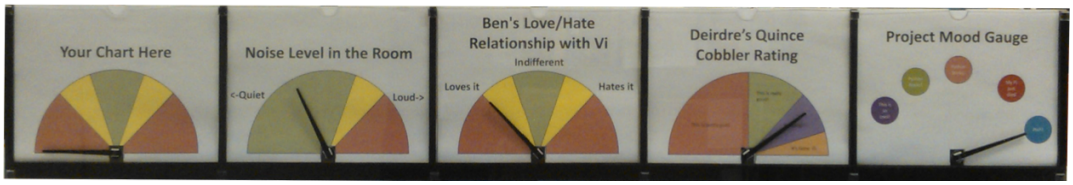
Commencez à imaginer vos propres projets avec Nanpy, par exemple en ressuscitant votre vieille voiture radiocommandée : youtu.be/NI4PDfVMdgM

Article de Andrea Stagi

Jauge Pi

DIFFICULTÉ : Facile-Moyenne

Ce projet sympa montre comment contrôler des servomoteurs par Internet en utilisant un Raspberry Pi.

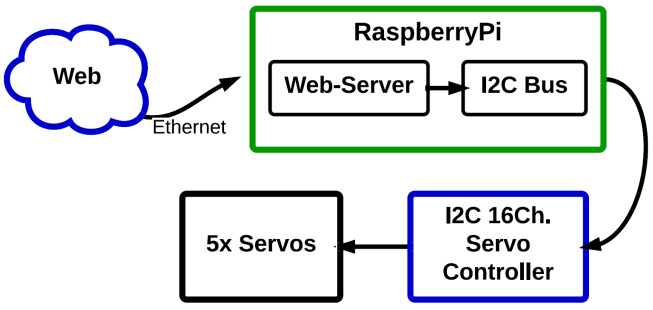


En résumé

Contrôler du matériel branché sur le Pi est vraiment sympa. Le faire depuis une autre ville ou un autre pays est génial.

Nous allons piloter cinq servos, chacun contrôlant une aiguille sur un graphique représentant toute donnée que vous souhaitez afficher, en modulant l'arrière-plan. Nous avons utilisé PHP pour créer une page web mise à disposition par le Pi. PHP fait les appels système via la ligne de commande qui appelle un script Python. À son tour, celui-ci contrôle les mouvements des servos via un bus I²C ; il renvoie aussi leurs positions en lisant la valeur depuis un registre disponible sur le contrôleur d'Adafruit 16 canaux (http://www.adafruit.com/products/815). Il est fourni avec une bibliothèque qui s'occupe des opérations de bas niveau. Son tutoriel sera nécessaire pour la configuration initiale et le téléchargement des bibliothèques. Le code et les modèles sont fournis dans un fichier d'aide présent sur un dépôt Git. Ce projet peut être étendu pour gérer jusqu'à 16 servos.

Le code est conçu pour la nouvelle Debian Wheezy sur un Pi Type B Rev1. Une Rev2 peut aussi être employée avec quelques modifications de la bibliothèque.



Liste du matériel

Voici la liste des éléments nécessaires à la réalisation de ce projet :

Liste du matériel		
Élément	*Qté*	*Remarques*
Servomoteur	5	Rotation à 180°
Alimentation 4-6V	1	≈ 100 mA par servo
Contrôleur de servo 16 canaux	1	I²C
Dispositif de montage	1	Nous l'avons fait maison

Fiche technique du contrôleur de servo Adafruit :
http://www.adafruit.com/datasheets/PCA9685.pdf

Branchement du matériel

Pour des raisons de sécurité, éteignez votre Pi et débranchez l'alimentation avant d'effectuer les branchements.

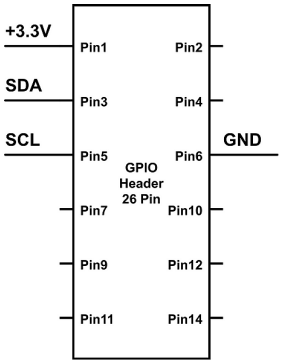
```
$ sudo shutdown -h now
```

Il faut tout d'abord connecter les servos. La plupart sont fournis avec des connecteurs adaptés préinstallés. Branchez-les sur le contrôleur de servo en vous assurant bien que les couleurs correspondent à la sérigraphie. Nous avons utilisé les canaux 1-5 :

Noir = Masse
Rouge = V+
Jaune = Signal

Le Pi ne peut pas fournir assez de courant pour alimenter les servos. Par conséquent, une alimentation externe est nécessaire. Nous avons utilisé le bloc mural (adaptateur CA) d'un vieux chargeur +5VCC de téléphone portable que nous avions sous la main. Utilisez les bornes du contrôleur de servo pour faire les connexions du V+ et de la masse.

Pour finir, il faut relier le Pi au contrôleur de servo. Cela nécessite de faire quatre connexions à partir des broches GPIO du Pi vers celles du contrôleur : 3,3V, GND, SDA et SCL.

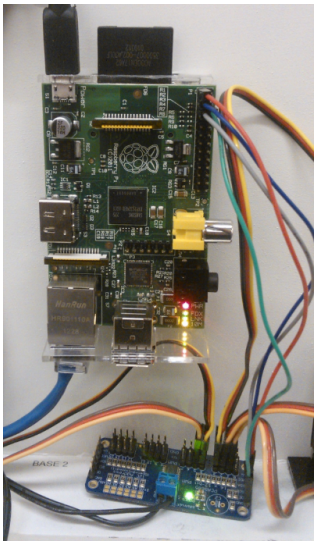


Vérifiez TOUTES vos connexions plusieurs fois AVANT la mise sous tension.

Danger: Les broches Vcc et V+ sont proches sur le contrôleur, ne les confondez pas ou vous aurez, comme nous, un Pi moribond !

Branchez votre bloc d'alimentation sur secteur et allumez votre Pi.

Si vous avez tout branché correctement vous ne devriez pas voir ou sentir de fumée bleue.



Téléchargement logiciel et utilitaire

Bien que facultative, la mise à jour de votre Pi est une bonne idée, commencez avec :

```
$ sudo apt-get update && sudo apt-get upgrade
```

Enregistrez les fichiers dans votre répertoire personnel :

```
$ sudo apt-get install git
$ git clone https://github.com/Thebanjodude/PiGauge
```

Décommentez toutes les lignes de ce fichier :

```
$ sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Ajoutez le périphérique I²C au noyau. Redémarrez votre Pi puis ajoutez-vous vous-même au groupe I²C :

```
$ sudo modprobe i2c-dev
$ sudo usermod -aG i2c votrenomdutilisateur
```

Installation d'Apache et PHP

```
$ sudo apt-get install apache2 php5 libapache2-mod-php5
```

Pour trouver l'IP du Pi (ex : 192.168.1.10) :

```
$ ip addr
inet: ip.de.votre.pi
```

Allez sur http://ip.de.votre.pi et vous devriez voir la page "It Works!".

Créez un lien du projet PiGauge vers la racine www :

```
$ cd /var/www
$ sudo ln -s /home/pi/PiGauge
```

Ajoutez apache au groupe I²C pour lui donner l'accès au bus I²C. Relancez-le ensuite :

```
$ sudo adduser www-data i2c
$ sudo /etc/init.d/apache2 restart
```

Depuis votre répertoire personnel :

```
$ sudo cp ./Adafruit-Raspberry-Pi-Python-Code/Adafruit_PWM_Servo_Driver/Adafruit_I2C.py /usr/local/lib/python2.7/site-packages/

$ sudo cp ./Adafruit-Raspberry-Pi-Python-Code/Adafruit_PWM_Servo_Driver/Adafruit_PWM_Servo_Driver.py /usr/local/lib/python2.7/site-packages/
```

Vous devriez être prêt à y aller, rendez-vous sur http://ip.de.votre.pi/PiGauge/ et essayez !

Lecture des positions des servos

Dans cet extrait de code nous allons ajouter deux octets non signés depuis le bus I²C pour obtenir la position d'un servo.

Suite sur la page suivante...

```
def print_position(chart_num):
  chart_pos = (chip.readU8(8 +    chart_num * 4)
+(chip.readU8(9 +          chart_num * 4) << 8))
  print chart_pos
```

Dans le tableau 3 de la fiche technique du PCA9685, il est possible de voir que les positions sont conservées tous les 4e et 5e registres à partir des registres 12 et 13. Pour obtenir la position d'un servo, il faudra ajouter ensemble les contenus des deux registres. Pourtant, en les ajoutant ainsi le résultat obtenu sera faux ! Pourquoi ? Deux registres 8 bits sont "collés" pour faire un registre de 16 bits. Cela s'appelle concaténation. En d'autres termes, le premier registre contient les bits 0-7 et le second les bits 8-15. Pour les ajouter correctement il faudra décaler tous les bits du second registre de 8 bits vers la gauche (>>8). Après, le nombre de 16 bits sera obtenu par addition. Ce qui est pratique avec les registres, c'est que l'électronique ne se préoccupe pas de ce qu'il y a dedans. C'est à vous, programmeur, qu'il appartient totalement d'en interpréter le contenu.

Utilisation des servos

Le projet entier tourne autour de la manipulation des servomoteurs. Les lignes de code suivantes sont sans doute les plus critiques. Nous avons défini une fonction appelée move_servos(). Celle-ci prend deux paramètres : sur quel numéro de graphique voulez-vous agir et où voulez-vous le positionner. pwm.setPWM() est fournie par la bibliothèque Adafruit.

```
def move_servos(chart_num, chart_pos):
  pwm.setPWM(chart_num,0,chart_pos)
  time.sleep(0.1)
```

chart_pos est un nombre entre 170 et 608 mais peut légèrement varier d'un servo à un autre. Ces nombres se réfèrent à la durée en largeur d'impulsion (cherchez contrôle de servo si ça vous intéresse). Pour rendre le logiciel plus intuitif, nous avons utilisé une fonction de transfert pour avoir des nombres de 0 à 100, puis nous sommes allés un peu plus loin. Comme les servos ne sont pas strictement linéaires, nous avons pris des points de données, nommés servo_data, et codé une régression linéaire (un mot pompeux pour dire droite de meilleur ajustement) pour compenser les non-linéarités des servos. La fonction de régression linéaire retourne les variables xfer_m et xfer_b utilisées plus bas.

```
def transfer(chart_percent):
  return int(xfer_m * chart_percent + xfer_b)

def inverse_transfer(chart_pos):
  return int(round((chart_pos - xfer_b) / xfer_m))
```

Test logiciel

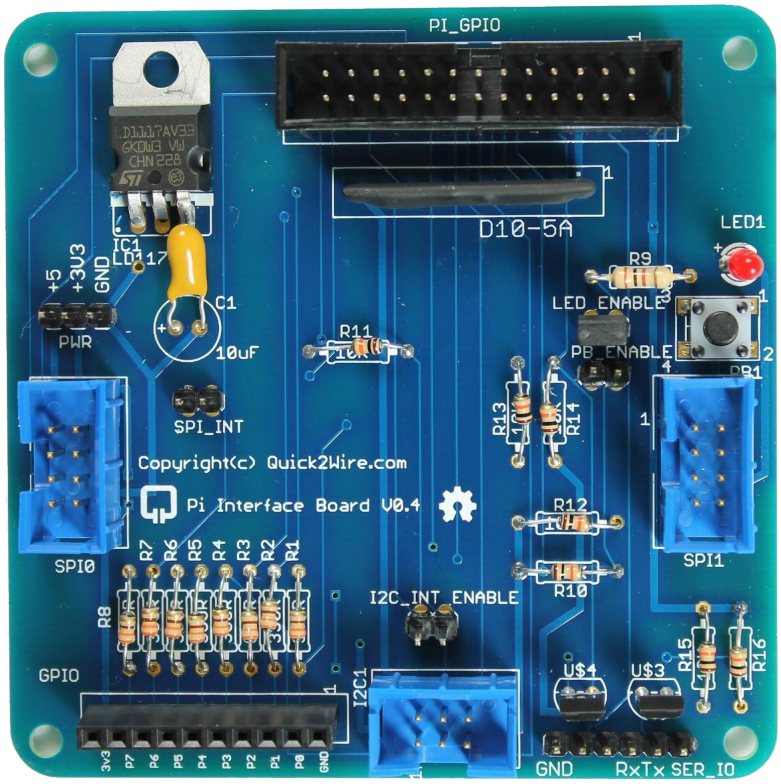
Nous sommes de fervents convaincus de la méthode de développement logiciel Agile et de l'innovation incrémentale. Nous n'avons pas déployé l'utilisation de Scrum ni de suivi pour ce petit projet mais nous avons fait quelques tests unitaires légers ; ils sont sur le dépôt si vous vous sentez d'humeur curieuse.

Remerciements

Des remerciements particuliers à Scott Ehlers pour m'avoir patiemment apporté de nouvelles compétences UNIX et PHP et à Tanda Headrick pour la construction de l'affichage mécanique. Un grand merci à National Technical Systems (NTS) pour avoir commandité le projet en nous accordant une petite récréation pour construire un projet d'affichage d'état. Suivez les liens vers NTS pour plus d'informations sur ce que nous faisons quand nous ne sommes pas en train de jouer avec un Raspberry Pi.

Article de Ben Schaefer

Quick2Wire



Safe and simple connection to your Raspberry Pi

Hardware

- Interface board
- I²C Port Extender
- Analogue board

Software

- For GPIO, I²C, SPI
- Device Libraries
- Examples

Support

- Articles
- Tutorials
- Forum

Find out more at quick2wire.com

Quick2Wire

A1

KICKSTARTER

Le Magazine MagPi - Du Virtuel au Réel http://kck.st/TvkdvG

Transposer le magazine MagPi, le meilleur et le seul magazine pour les passionnés du Raspberry Pi, du domaine du virtuel à celui du réel.

C'est une opportunité unique pour démarrer rapidement (kick-start), que d'étendre la disponibilité du magazine MagPi depuis le domaine du virtuel à celui du réel, et nous offrons à nos donateurs quelques récompenses très intéressantes. Le MagPi est le seul magazine au monde dédié aux utilisateurs de Raspberry Pi, l'ordinateur à 35$.

Le magazine MagPi a été publié en ligne chaque mois depuis mai 2012. Chaque numéro de 32 pages est intégralement créé avec le Raspberry Pi par des passionnés. Notre site, www.TheMagPi.com, reçoit 100000 visites par mois.

Nous avons toujours souhaité rendre le magazine disponible sous forme imprimée, au point d'avoir fait un tirage de test avec le numéro 6. Nous avons obtenu pour résultat un grand nombre de demandes pour que tous les anciens numéros soient imprimés. Cependant, l'impression professionnelle d'un magazine implique des coûts initiaux importants et il est nécessaire d'imprimer un nombre d'exemplaires conséquent afin d'obtenir des tarifs qui nous décident à le faire.

Vos promesses vont faire passer le MagPi du virtuel au réel. S'il y a un intérêt manifeste, alors cela nous apportera la confiance pour continuer à produire des copies imprimées du MagPi quand nous lancerons le Volume 2 l'année prochaine.

Au lieu de rendre disponible chaque ancien numéro individuellement, nous avons estimé qu'il serait plus intéressant et plus efficace financièrement de réaliser le Volume 1 du MagPi sous la forme d'un seul coffret exceptionnel "bumper pack". Ce volume 1 sera composé de tous les 8 numéros parus en 2012, y compris celui de décembre. Vous obtiendrez en supplément une reliure exclusive pour conserver l'ensemble de ces numéros et dont la présentation a été conçue spécialement pour ce projet Kickstarter.

Avec 32 pages par magazine, cela représente 256 pages en couleurs de pur contenu.

L'objectif de financement n'est pas le but réel. Nous avons besoin d'au minimum 250 promesses pour le magazine avant de pouvoir lancer un tirage convenable qui présente un intérêt pour nous, et un prix intéressant pour vous.

Où va aller l'argent ? Ici au MagPi nous sommes tous des bénévoles - nous ne faisons certainement pas ça pour l'argent. Nous tirons notre inspiration de la fondation Raspberry Pi et mettons en œuvre nos talents, notre enthousiasme ainsi qu'une grande partie de notre temps personnel pour contribuer à l'ensemble de la communauté et favoriser davantage l'éducation de toute personne utilisant l'ordinateur Raspberry Pi.

Cependant, il existe certains coûts administratifs inhérents à la production d'un magazine de grande qualité. Imprimer des magazines n'est évidemment pas gratuit et de surcroît des coûts relatifs à l'hébergement du site web et à l'utilisation de la bande passante sont également présents. Alors que la plus grande partie du montant des promesses sera employée à imprimer les magazines et fournir les récompenses, il y aura un petit excédent tiré de chaque promesse. Il sera utilisé pour financer les coûts récurrents liés à la production du MagPi et nous permettre d'explorer d'autres horizons pour accroître la disponibilité du magazine et intégrer d'autres types de contenu.

Chaque promesse donnera également lieu à une donation à la fondation Raspberry Pi.

MAGPI BUMPER PACK : C'est lui. Il comporte les exemplaires imprimés du Volume 1 (2012) avec le MagPi composé de tous les 8 numéros. Vous recevez également la reliure MagPi en édition limitée pour conserver vos revues plus des autocollants MagPi. **£25**

GAMBLE PACK : Composé du MAGPI BUMPER PACK plus un boîtier Raspberry Pi "Gamble". Ce qui est drôle (gamble=pari) c'est que vous ne savez pas de quelle couleur sera le boîtier que vous allez recevoir. **£30**

BOGO PACK : Achetez un MAGPI BUMPER PACK et donnez en un en cadeau. **£50**

MAG+PI KIT : Comporte le MAGPI BUMPER PACK et un Raspberry Pi 512 Mio avec son boîtier. **£65**

STARTUP KIT : Vous donne le MAG+PI KIT plus tout ce dont vous avez besoin pour démarrer et faire tourner votre Raspberry Pi. **£99**

SIGNATURE EDITION : En quantités très limitées, contient le MAGPI BUMPER PACK plus un exemplaire du livre "Raspberry Pi User Guide" dédicacé par Eben et Liz Upton. **£100**

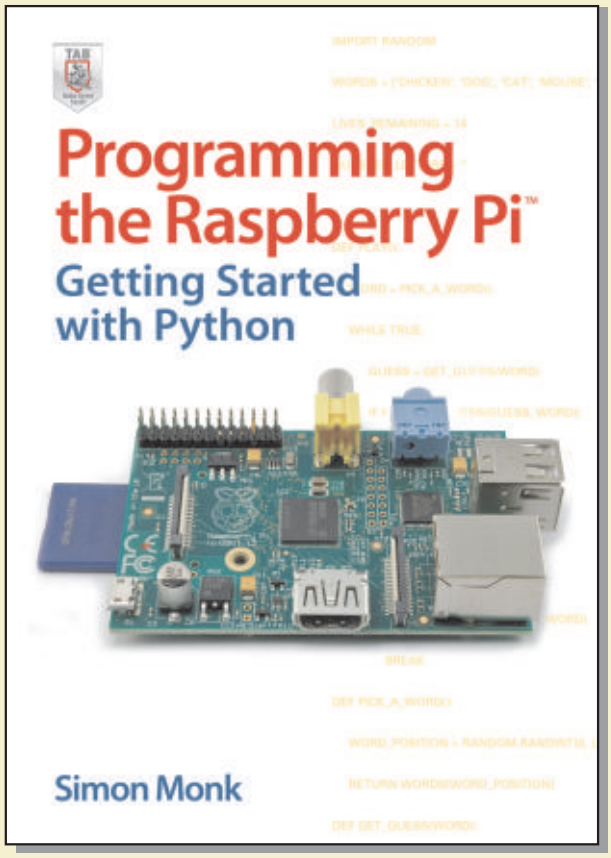
ULTIMATE KIT : C'est le fin du fin. Vous avez le STARTUP KIT et PLUS d'accessoires. **£145**

SIGNATURE KIT : C'est L'expérience Raspberry Pi. Cette édition très limitée vous donne l'ULTIMATE KIT plus la SIGNATURE EDITION. **£200**

Programmer le Raspberry Pi: Débuter avec Python

Dans un nouveau livre de www.raspberrypibook.com, Simon Monk couvre les bases de Python jusqu'à l'utilisation approfondie du GPIO

Ayant acheté un Raspberry Pi, il y a des chances pour que vous soyez intéressé par l'apprentissage de la programmation de votre nouveau gadget. Le livre "Programming the Raspberry Pi: Getting Started with Python" de Simon Monk, guide le lecteur à travers le processus d'apprentissage de Python avec le Raspberry Pi.

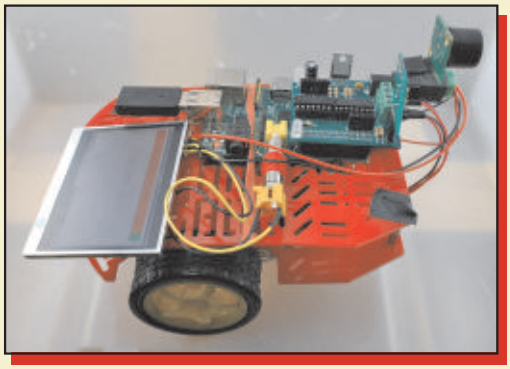


Le livre est accessible aux nouveaux venus en programmation et mène le lecteur à travers les bases de Python, avant de s'orienter vers des sujets plus complexes comme l'utilisation des bibliothèques Tkinter et Pygame ou encore la programmation du connecteur GPIO.

L'approche est très pratique. Les concepts de programmation sont développés dans des programmes d'exemple, construits sur des bases simples de la même manière que si vous écriviez un programme depuis le début.

Trois chapitres du livre sont consacrés exclusivement à la programmation et à l'utilisation du connecteur GPIO. Différentes techniques, outils et produits de prototypage sont étudiés et détaillés, parmi lesquels Gertboard, PiFace, Pi Cobbler et la RaspiRobotBoard.

Deux des chapitres sur le matériel contiennent des instructions pas-à-pas pour construire et programmer des projets matériels utilisant le connecteur GPIO. Le premier projet est un simple affichage DEL à 7 segments qui affiche l'heure système du Raspberry Pi. Le second est un rover qui utilise le kit rover à bas prix Magician Chassis, avec une carte d'interface RaspiRobotBoard.



L'ensemble du code source est disponible en téléchargement depuis le site web du livre.

Le livre est disponible dans la plupart des grandes librairies depuis la fin novembre 2012 et le site web (www.raspberrypibook.com) propose davantage d'informations.

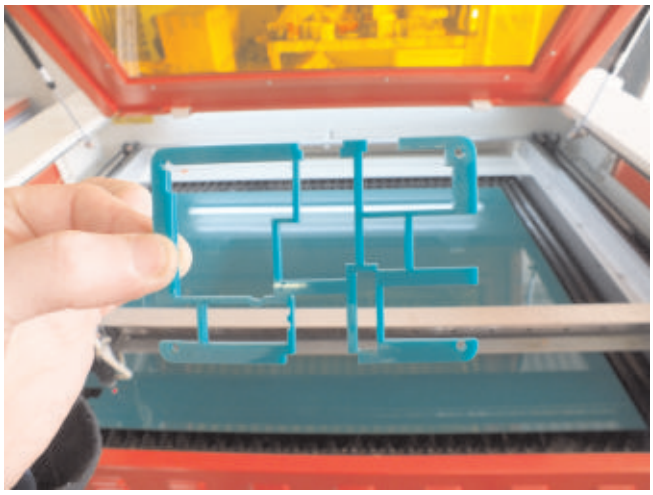
L'interview de Pibow

Les gars de Pibow nous donnent un aperçu du fonctionnement interne de leur usine, la façon dont le Pibow les a changés, les projets qu'ils ont en réserve pour nous et même des trucs pour démarrer sa propre entreprise.

Rencontrer les gars de Pibow était pour moi une expérience intéressante et comme ils étaient assez proches de moi j'ai pensé qu'il était logique d'aller leur rendre visite et de les interviewer, et je suis content de l'avoir fait !

Q : À quoi attribuez-vous le succès du Pibow et comment cela vous a-t-il affecté ?

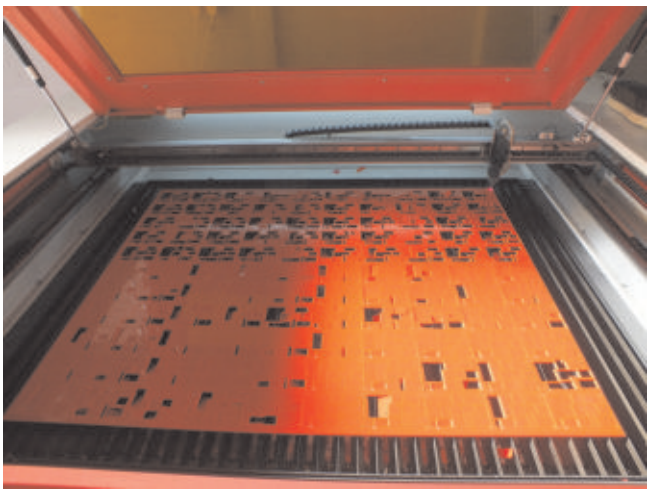
"Pibow était tellement différent des autres boîtiers qu'il s'est fait vraiment remarqué. Évidemment, le fait d'être mis en évidence sur la page d'accueil de Raspberry.org a été un coup décisif pour nous et a généré l'intérêt initial pour le Pibow, à partir de quoi nous devions juste nous assurer de pouvoir les expédier aussi vite que possible. C'était génial d'envoyer des lots importants d'unités, le jour suivant il y aura toujours des quantités de tweets excités et joyeux de la part de gens dont le Pibow vient juste d'arriver."



Un exemple de partie découpée au laser

Q : Y a-t-il quelque chose que vous auriez fait différemment ?

"Si nous avions su le succès que nous allions obtenir, nous aurions acheté les découpeuses à laser supplémentaires plus tôt, ce qui aurait rendu les choses beaucoup plus faciles et rapides. Elles sont chères, cependant nous avons toujours voulu être sûrs d'en avoir besoin sur le long terme."

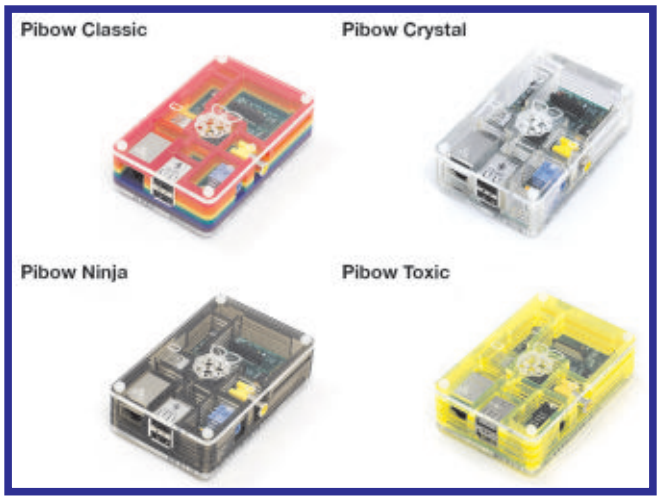


Q : Quels sont vos projets pour le futur ? Est-ce-qu'il y aura des variations de couleur comme une édition spéciale "Piwasp" (noir et jaune) ou "Americana style" (rouge, blanc et bleu), pourrait-il y avoir des boîtiers avec des roues ?

"Là maintenant, nous sommes concentrés sur Picade (http://www.kickstarter.com/projects/pimoroni/picade-the-arcade-cabinet-kit-for-your-raspberry-p) que nous avons dévoilé il y a deux semaines sur KickStarter. Il y a aussi certaines choses à venir dans un avenir très proche pour pibow - dont des nouvelles couleurs !"

[D'accord, la suggestion concernant des boîtiers avec des roues était donc un peu farfelue mais vue la façon dont ces gars progressent, qui sait ?]

Jon et Paul de Pimoroni



PiBows en attente d'expédition

Q : Pour conclure, d'après votre expérience, quel conseil donneriez-vous à des entrepreneurs en herbe ?

"Connais ton truc. Internet est rempli d'informations de qualité si vous les cherchez. Alors lancez-vous et acceptez toute l'aide que vos amis ou votre famille peuvent vous apporter :-)

Nous avons travaillé vraiment très très dur au cours des quatre derniers mois pour envoyer tous les Pibows que les gens ont commandés. Nous avons fait marcher les lasers pendant presque 16 heures par jour, chaque jour, y compris les week-ends, pendant cette durée. Nous étions prêts à mettre nos vies entre parenthèses pour être sûrs de pouvoir faire les livraisons aussi vite que possible.

Les amis et la famille ont été incroyables et nous ont offert leur aide à l'atelier et ils nous ont également apporté un certain support financier pour nous faire tenir et continuer. Nous ne pourrons jamais les remercier assez pour tout ce qu'ils ont fait pour nous aider."

Ils ressemblent à de grands boîtiers. Avec un peu de chance, nous verrons bientôt le kit Picade entrer en production lui aussi.

PICADE

Article de Chris Stagg

VOUS LE SAVIEZ?

Paul de Pimoroni est responsable de la conception du logo Raspberry Pi !

Le boîtier PiBow est fabriqué en acrylique.

Pimoroni possède désormais 3 découpeuses à laser, appelées Bert, Ernie et Cookie Monster ! *[Ed : ça peut être long avant que Oscar, Big Bird et The Count n'apparaissent.]*

Apprenez à créer un sapin de Noël style 70's avec CESIL et un Raspberry Pi

CESIL - acronyme signifiant Computer Education in Schools Instructional Language - a été conçu dans les années 70 comme tentative de faire découvrir le monde de la programmation informatique à la jeune génération. Sans ordinateur dans les écoles, les élèves devaient écrire les programmes sur papier et les envoyer à leur centre d'informatique local. Les résultats leur étaient retournés par la poste une semaine plus tard !

CESIL est un langage d'assemblage très simplifié, avec une base applicative très limitée, il est facile de l'apprendre et d'écrire des programmes simples avec. Comme ce n'est pas le propre de CESIL d'être vraiment passionnant, j'en ai donc écrit un interpréteur en BASIC, et ajouté un sapin de Noël avec des lumières féériques programmables ! Le sapin a 4 rangées de 8 lampes. Imaginez-le comme une grille de 8 de large et de 4 de haut.

Un programme CESIL est essentiellement constitué de trois colonnes de texte. La première colonne (qui peut être vide) est l'étiquette. C'est un emplacement dans le programme vers lequel il est possible de "sauter" depuis un autre endroit. La colonne du milieu contient l'opérateur - il s'agit de l'instruction à exécuter, et la dernière colonne est l'opérande - c'est-à-dire la donnée que l'instruction va utiliser. Cette donnée peut être le nom d'une étiquette s'il s'agit d'une instruction de saut, elle peut être un nombre ou faire référence à un emplacement mémoire nommé, ou une variable.

Mes ajouts à la machine CESIL consistent en deux registres supplémentaires (trois au total) afin de conserver les positions des rangées et colonnes des lampes, une instruction de couleur pour définir la couleur d'une lampe ainsi qu'une fonctionnalité de sous-programme. Le programme peut atteindre 256 lignes et contenir jusqu'à 256 variables.

La meilleure façon d'expliquer son fonctionnement consiste à étudier un programme réel. Celui-ci lit un nombre depuis le clavier et affiche une table de multiplication :

```
# mtable:
# Générateur de table de multiplication
  line
  print "Générateur de table de
multiplication"
  line
  print "Quelle table"
  in
  store table
  load 1
  store index # Index fois ....

loop: load index
  out
  print " FOIS "
  load table
  out
  print " = "
  mul index # Table était dans
l'accumulateur
  out
  line
  load index # Ajoute 1 à l'index
  add 1
  store index
  sub 11 # Soustrait 11 pour compter de
1 à 10
  jineg loop # Si <0 alors sauter à
l'étiquette loop
  halt
```

Les lignes vides sont autorisées et les commentaires commencent par le symbole #. Quasiment tout devrait pouvoir se passer d'explication, mais avec seulement un accumulateur, absolument tout doit être transféré depuis ou vers la mémoire - l'instruction "store table" enregistre l'accumulateur dans une variable appelée "table".

Les instructions standard de CESIL sont :

LOAD – Transfère le nombre vers l'accumulateur

STORE – Transfère l'accumulateur vers une variable nommée

JUMP – Saute à l'étiquette spécifiée

JINEG – Saute si l'accumulateur est négatif

JIZERO – Saute si l'accumulateur est nul

ADD – Ajoute une valeur à l'accumulateur

SUB – Soustrait de l'accumulateur

MUL – Multiplie l'accumulateur par une valeur

DIV – Divise l'accumulateur par une valeur

HALT – Termine le programme

IN – Lit un nombre depuis le clavier

OUT – Affiche l'accumulateur en tant que nombre

PRINT – Affiche une chaîne littérale (entre "guillemets")

LINE – Affiche une nouvelle ligne

Extensions :

JSR – Saute à un sous-programme

RET – Retour depuis un sous-programme (vers la ligne qui suit la dernière instruction JSR)

Extensions relatives au sapin de Noël :

TREE – Construit un nouvel arbre de Noël

ROW – Transfère l'accumulateur vers le registre Row

COL – Transfère l'accumulateur vers le registre Column

COLOUR – Utilise la valeur de couleur stockée dans l'accumulateur pour fixer la couleur de la lampe désignée par les registres Row et Column

WAIT – Fait une pause du nombre donné en centièmes de seconde (100 cs)

Notez que vous devez exécuter une instruction WAIT pour que les changements de couleur des lumières soient effectifs. Cela signifie que vous pouvez fixer en une seule fois beaucoup de lumières, et ensuite quand vous exécuterez le WAIT (même une instruction WAIT 0), elles seront toutes modifiées en même temps.

Il y a 16 couleurs standard :

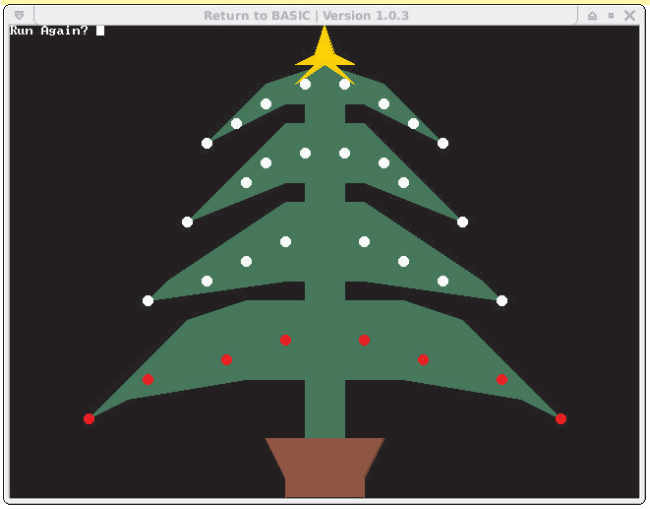
0:Éteint, 1:Bleu marine, 2:Vert, 3:Bleuâtre, 4:Marron, 5:Violet, 6:Olive, 7:Argent, 8:Gris, 9:Bleu, 10:Verdâtre, 11:Turquoise, 12:Rouge, 13:Rose, 14:Jaune, 15:Blanc

Notre sapin de Noël possède 4 rangées de 8 lampes. La rangée 0 est en bas, et la colonne 0 est du côté gauche.

L'extrait de programme qui suit va remplir la rangée du bas avec des lampes rouges :

```
# Exemple de programme pour allumer les
# lampes de la rangée du bas en ROUGE
  tree # Crée un arbre !
  load 0
  row # Rangée 0 - Bas
  load 7 # Compte de 7 à zéro
loop:
  store col-count
  col
  load 12 # Rouge
  colour
  load col-count
  sub 1
  jineg done # Saute si négatif
  jump loop

done: wait 1 # Actualise les lumières
  halt
```



Pour commencer, il faut télécharger l'interpréteur BASIC RTB depuis https://projects.drogon.net/return-to-basic/. Ensuite, l'interpréteur CESIL et les démos peuvent être installés en utilisant :

```
  cd
  git clone git://git.drogon.net/cesil
  cd cesil
  rtb
  load cesil
  run
```

Ce que j'aimerais voir, ce sont des personnes partageant des exemples, aussi, merci de les poster sur les forums, de me les envoyer par courriel (projects@drogon.net) et en janvier je regarderai ceux que j'aurai trouvés et j'enverrai une carte à échelle Raspberry gratuite à celui que j'estimerai être le plus original, ou astucieux...

Article de Gordon Henderson

Bienvenue au C++ caché!

La dernière fois nous avons débuté avec le C++ et écrit nos tous premiers programmes. Aujourd'hui, nous allons poursuivre, et nous vous montrerons également comment exécuter vos programmes ainsi que quelques autres types de variables.

Compiler et exécuter des programmes

Après avoir écrit votre programme (vous pouvez utiliser n'importe quel éditeur de texte, comme nano ou Geany), enregistrez-le en tant que fichier .cpp, puis ouvrez une nouvelle fenêtre de terminal et tapez la ligne suivante :

```
g++ [nom].cpp -o [nom_du_programme]
```

Remplacez [nom] par le nom du fichier et [nom_du_programme] par le nom que vous souhaitez donner au programme à créer.

Si vous obtenez une erreur selon laquelle le fichier est introuvable, vous devrez utiliser la commande cd pour vous placer dans le répertoire où le fichier .cpp a été enregistré, et ensuite relancer la commande. Si l'erreur indique que g++ n'a pas été trouvé (cela ne devrait pas être le cas sur les nouvelles images), tapez sudo apt-get install build-essential pour installer g++.

Une fois la compilation terminée (cela peut prendre beaucoup de temps pour les gros programmes), tapez simplement :

```
./[nom_du_programme]
```

Où [nom_du_programme] doit être identique à ce que vous avez utilisé lors de la compilation. Le programme doit alors s'exécuter, et quand il sera terminé, vous serez renvoyé sur le terminal comme avec tout autre programme en ligne de commande.

Plus de types de variables

La dernière fois nous avons vu le type de variable int qui nous permet de stocker des nombres entiers. Bien sûr, nous pouvons avoir besoin d'enregistrer autres choses, si bien qu'il existe différents types de variable. Ceux-ci sont présentés dans le tableau suivant :

Nom	*Ce qu'il stocke*	*Exemple*
int	Un nombre entier.	42
float	Un nombre décimal avec 6 chiffres après la virgule.	3.141592
double	Un nombre décimal avec 10 chiffres après la virgule.	3.1415926535
char	Un caractère unique.	c
string	Une chaîne de caractères.	Bonjour, comment ça va ?
bool	Vrai ou faux.	VRAI

Pour créer une variable, indiquez le type désiré, suivi du nom que vous voulez lui donner,

```
#include <string>
#include <iostream>
using namespace std;

int main()
{
    // Création de quelques variables numériques :
    int nombreEntier = 5;
    float nombreDecimal = 5.5;

    // Création de quelques variables littérales :
    // Remarquez comment les guillemets simples sont utilisés
    // pour les caractères, et les guillemets doubles
    // pour les chaînes.
    string salutation = "Bonjour à tous ";
    char ponctuation = '!';
    // Création d'une variable booléenne :
    bool estVrai = false;

    // Affichage de la somme de nos variables :
    cout << nombreEntier + nombreDecimal << endl;
    cout << salutation + ponctuation << endl;
    cout << estVrai << endl;

        return 0;
}
```

Regardez ce qui se passe quand des variables de types différents sont additionnées. Celles de type numérique donnent le résultat escompté, 5 + 5,5 est égal à 10,5 et c'est bien ce que nous obtenons. Quand nous ajoutons des variables littérales, les lettres sont ajoutées aux autres. Ainsi, nous avons notre chaîne, "Bonjour à tous ", et nous y ajoutons le point d'exclamation, si bien que nous avons finalement "Bonjour à tous !". C'est pourquoi les types de données sont importants. Si les nombres avaient été enregistrés en tant que chaînes, l'addition aurait ajouté 5.5 après 5, et le résultat aurait été "55.5". C'est une bonne idée également de donner des noms judicieux aux variables. Ces noms peuvent contenir des lettres, des nombres et des caractères de soulignement (underscore), mais ne peuvent pas commencer par un nombre. Il existe aussi certains mots-clés qu'il n'est pas possible d'utiliser car ils sont réservés pour le langage.

Notez également que estVrai renvoie 0 au lieu de "false". Un booléen vaut fondamentalement 0 pour faux, ou 1 pour vrai, comme en binaire, et c'est pourquoi le programme l'affiche ainsi. La fois précédente nous avons utilisé cin pour permettre à l'utilisateur de saisir une valeur et de l'enregistrer dans une variable. Essayez de faire ça avec le code ci-dessus. Le signe = peut aussi être utilisé à tout moment pour modifier ce qui est enregistré dans une variable (c'est la raison pour laquelle elles sont appelées variables). La valeur d'une variable peut être affectée à une autre variable, par exemple :

```
int i = 4;
int j = i;
```

Par contre, il n'est pas possible de l'utiliser pour affecter une valeur à une chaîne :

```
int i = 4;
string s = i;
```

car la classe string ne possède aucune fonction membre qui permettrait cette affectation. L'affectation d'un nombre à une chaîne peut être réalisée grâce à un flux de chaîne de caractères (stringstream), cela sera abordé lors d'un prochain tutoriel.

Article d'Alex Kerr

De petits pas dans...

LE GRAND MONDE

Ada, un langage pour tout le monde

Par Luke A. Guest

Introduction

Faisant suite au numéro 6, nous allons continuer à apprendre les bases du langage Ada.

Types numériques (suite)

Les types entiers peuvent avoir une valeur négative, comme -10, -55, etc. Les types naturels ne peuvent accepter des valeurs qu'à partir de 0 (donc, pas de nombre négatif) et les types positifs ne peuvent prendre que des valeurs à partir de 1 (pas de valeur négative ni de zéro).

Tous ces types peuvent être utilisés ensemble dans des expressions mathématiques, mais il faut s'assurer de ne pas provoquer de débordement selon le type qui leur est affecté, sinon une erreur pourra se produire. Par exemple, si vous définissez deux variables X : Natural := 1; et Y : Integer := 2; et ensuite Y est soustrait de X puis réaffecté à X (c.-à-d. X := X - Y), cela provoquera une erreur car le résultat vaut -1 ce qui est en dehors des limites autorisées pour les types Natural.

Types booléens

Les types booléens possèdent deux valeurs, True (vrai) et False (faux), il n'y a rien d'autre qui puisse être attribué à une variable de type Boolean.

Décisions simples

Tous les langages ont la notion de valeurs booléennes car toutes les conditions en programmation reposent sur le fait qu'une chose est soit vraie soit fausse.

Tapez le code du listing 1 pour voir comment nous pouvons prendre des décisions en Ada en utilisant le type

```
1     with Ada.Text_IO;
2     use Ada.Text_IO;
3
4     procedure Decisions is
5        Is_Defective : Boolean := False;
6     begin
7        if Is_Defective = True then
8           Put_Line ("Defective");
9        else
10          Put_Line ("Not defective");
11       end if;
12    end Decisions;
```

Listing 1: decisions.adb

Ligne 5 : Nous définissons une variable booléenne, Is_Defective et lui donnons la valeur False.

Ligne 7 : Nous testons pour savoir si **Is_Defective = True**. Le = signifie que ce qui est à gauche est égal ou identique à ce qui est à droite. La partie entre le **if** (si) et le **then** (alors) est appelée une expression. Nous aurions aussi pu simplement mettre if **Is_Defective then** pour dire la même chose.

De même, **if Is_Defective = False** then et **if not Is_Defective then** sont similaires.

Ligne 8 : Tout ce qu'il y a entre les mots-clés **then** (alors) et **else** (sinon) est exécuté quand la condition est vraie.

Ligne 10 : Tout ce qu'il y a entre les mots-clés **else** (sinon) et **end if** (fin si) est exécuté quand la condition est fausse.

Ligne 11 : Tous les blocs **if** doivent se clôturer avec **end if**; même s'il n'y a pas de **else**.

booléen.

Littéraux

Nous avons déjà vu quelques littéraux, même si nous ne savons pas de quoi il s'agit. Un littéral est un morceau de données dans le code source comme le nombre 125 ou la chaîne "Bonjour, de la part de Ada." dans le listing 1 de la partie 1 ; ce sont des littéraux numériques et chaînes, respectivement.

En Ada également, il existe des caractères littéraux, tels que 'C'. Une chaîne est constituée de caractères, de telle sorte que des caractères peuvent être ajoutés à des chaînes en utilisant l'opérateur &, comme dans le listing 2 de la partie 1, par exemple "Hello " & 'G' est identique à "Hello G". Il est possible de créer des variables de type Character comme cela a été fait précédemment avec Integer, Natural et Positive.

Pour le type booléen, tel qu'il a été indiqué plus haut, les valeurs True et False sont des littéraux booléens.

Plus d'attributs

Nous avons déjà fait connaissance avec l'attribut 'Image, mais nous allons maintenant aborder trois autres attributs de type entier qui peuvent se révéler très utiles. Chaque type d'entier possède un intervalle de valeur acceptable, cet intervalle étant défini par les attributs 'First (premier) et 'Last (dernier), il est également possible d'y accéder en utilisant l'attribut 'Range, qui concrètement retourne 'First .. 'Last où les deux points (tréma) signifient "c'est un intervalle de valeurs."

Si vous essayez d'affecter une valeur en dehors de cette plage à une variable d'un type, cela provoquera une erreur et votre programme ne fonctionnera pas correctement.

Une autre utilisation de 'First consiste à affecter une valeur initiale à une variable. Dans le listing 2, la valeur initiale affectée à une variable vient après le symbole := qui est lui-même situé après le nom du type.

Fonctionnalités sympas : les attributs

Beaucoup d'entités du langage possèdent des attributs qui peuvent fournir des informations au programmeur. Pour y accéder, il faut utiliser l'apostrophe ('). Dans nos exemples, nos avons vu les attributs des 3 types d'entiers : 'Image (), 'First, 'Last et 'Range.

Exercices

1. Utilisez l'attribut Integer'Image pour afficher les valeurs retournées par les attributs Integer'First et Integer'Last.
2. Refaites l'exercice 1 avec les types Positive et Natural.
3. Essayez de remplacer les valeurs initiales avec l'attribut 'First.

UN PEU D'HISTOIRE

Dans les années 70, le Ministère de la Défense des États-Unis (DoD) décida qu'un langage de programmation devrait être utilisé pour l'ensemble de ses projets. Plusieurs équipes à travers le monde envoyèrent des propositions pour un nouveau langage, chacune nommée d'après une couleur : Rouge, Verte, Bleue et Jaune. Parmi toutes, c'est l'équipe verte dirigée par Jean Ichbiah de CII Honeywell Bull en France qui gagna.

La première version du langage vert fut adoptée par les équipes de développement du DoD et il a été dit de l'utiliser pour tous les projets ultérieurs.

Le langage vert fut ensuite renommé Ada d'après le nom de la fille de lord Byron, Augusta Ada (comtesse Lovelace), qui a travaillé aux côtés de Charles Babbage et qui est considérée comme la première programmeuse du monde en raison du travail sur la Machine Analytique de ce dernier.

La première version du langage Ada sortit vers 1983, sous le nom Ada 83, la version suivante parut au cours des années 90, initialement appelée Ada9X et renommée plus tard Ada95, la suivante fut Ada 2005 et cette année fut distribuée une autre révision appelée Ada 2012.

Le compilateur GNAT est toujours en cours de développement pour pouvoir prendre en charge les fonctionnalités du nouvel Ada 2012, donc malheureusement, certaines d'entre-elles ne fonctionneront pas si vous les essayez. Pour nos programmes, nous pouvons assurément utiliser Ada95 mais aussi la plupart des fonctionnalités d'Ada 2005.

CAMP D'ENTRAINEMENT BASE DE DONNÉES

Il semble y avoir beaucoup de polémique à propos du langage le plus approprié pour l'apprentissage de l'informatique, et l'un des plus rarement proposés est SQL - Structured Query Language.

Il s'agit de la norme internationale pour manipuler des bases de données et j'affirme qu'il est plus simple à apprendre que beaucoup de code. C'est ce qui est connu comme un "langage de haut niveau", c.-à-d. très proche d'une forme tronquée de l'anglais. SQL couvre une large gamme d'applications dans le monde, notamment la gestion de sites web intelligents !

Une fois que vous aurez commencé à jouer avec, vous comprendrez le rythme, ou la syntaxe, pour vous exprimer en SQL au point où vous pourrez presque deviner une commande.

Suivez ces étapes sur votre propre Pi ; elles expliquent comment faire fonctionner MySQL et obtenir une plateforme solide sur laquelle vous pourrez développer vos compétences en bases de données - au-delà de la portée de cet article.

Installation de MySQL

Nous allons utiliser une version Open Source de SQL appelée "MySQL", mais elle n'est pas présente dans la distribution Raspbian standard ; il faudra donc la localiser, la télécharger et l'installer sur votre Pi. Heureusement, ce processus complexe est contrôlé par une seule commande et est lancé dans un écran texte ou fenêtre de terminal :

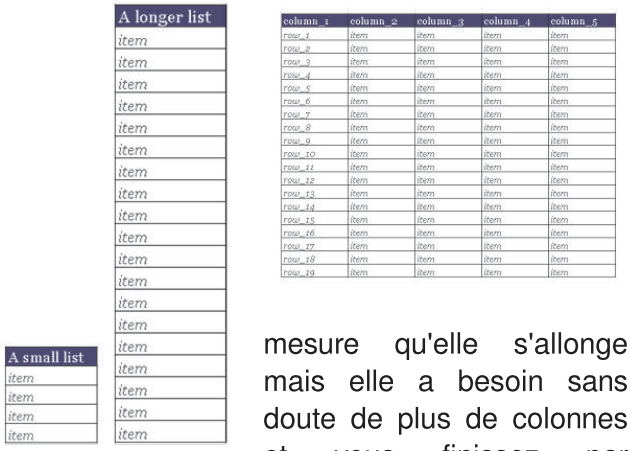
```
pr0mpt > apt-get install mysql-server
```

Votre Pi se met en route et commence le téléchargement. Il peut y avoir des moments au cours du processus pendant lesquels il semble ne rien se passer mais soyez patient. Enfin, un écran bleu s'affichera afin de demander un mot de passe pour MySQL - deux fois pour confirmation.

Bases de données

La base de données la plus simple est une liste.

Une liste devient de plus en plus utile au fur et à



mesure qu'elle s'allonge mais elle a besoin sans doute de plus de colonnes et vous finissez par conséquent avec une table.

Une table possède des colonnes et des lignes - dans le jargon des bases de données, elles sont appelées champs et enregistrements. Une grande base de données peut contenir beaucoup de tables - et ainsi de suite.

Il y a trois fonctions de base pour la manipulation de données dans une table : ajout, suppression et modification.

Donc, en ce qui concerne les données que nous conservons, nous pouvons **ajouter** un nouvel enregistrement, **supprimer** un ancien enregistrement ou **modifier** le contenu d'un enregistrement existant - c'est tout ! (DML - Data Manipulation Language).

Quant à la façon dont sont gardées les données, nous pouvons **ajouter, supprimer** et **modifier** des tables ainsi qu'**ajouter, supprimer** et **modifier** des champs - c'est tout ! Maîtrisez ces commandes et vous êtes bien parti pour contrôler des bases de données. (DDL - Data Definition Language).

Une fois les données en place elles sont inutiles à moins que nous ne FASSIONS quelque chose avec et cela se résume aux deux fonctions de base **tri** et **recherche**.

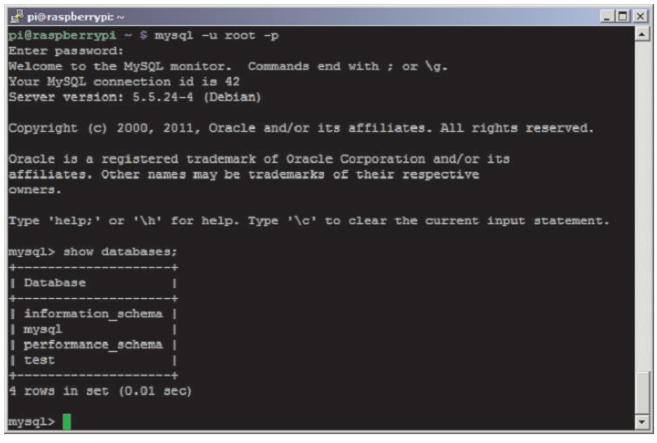
Données

Nous avons réussi à résumer des bases de données rudimentaires à quelques fonctions mais avant de procéder en SQL il est nécessaire d'avoir un aperçu final des données elles-mêmes.

Si jamais vous avez été amené à remplir un formulaire, qu'il s'agisse d'un document imprimé ou sur un site web, vous êtes sûrement déjà tombé sur une question peu claire ou dont la réponse est trop longue pour tenir dans l'espace disponible. Vous pouvez griffonner des notes sur un formulaire papier ou écrire plus petit pour que vos réponses ne débordent pas, mais ce n'est pas possible sur ordinateur car tout est défini plus strictement. Il en est ainsi avec les "types de données". MySQL ne reconnait que trois types de données : texte, nombre et date. Pour que les données soient stockées de manière efficace, il est nécessaire de leur allouer une quantité appropriée d'espace de stockage sur l'ordinateur.

Accès à MySQL

La commande `mysql -u root -p` permet d'entrer dans MySQL. À partir de là vous avez quitté Raspbian et vous devez taper des instructions du langage de requête structurée SQL. Bien que SQL soit une norme internationale, il existe des variantes selon les implémentations et MySQL en est une version particulièrement polie. Il faut lui dire s'il vous plaît à la fin de chaque instruction sinon rien ne se passe.



Le raccourci pour s'il vous plaît est le point-virgule. **Souvenez-vous - pas de ; pas d'action.**

Affichage des bases de données

Pour afficher les bases de données existantes, tapez simplement :

```
pr0mpt > show databases;
```

Si vous avez oublié vos bonnes manières, vous pouvez entrer le ; sur une ligne séparée. Les quatre bases de données par défaut vont s'afficher. Avant de commencer à changer quelque chose, prenons confiance en nos capacités en regardant alentour, c.-à-d. :

```
pr0mpt > show tables in
information_schema;
```

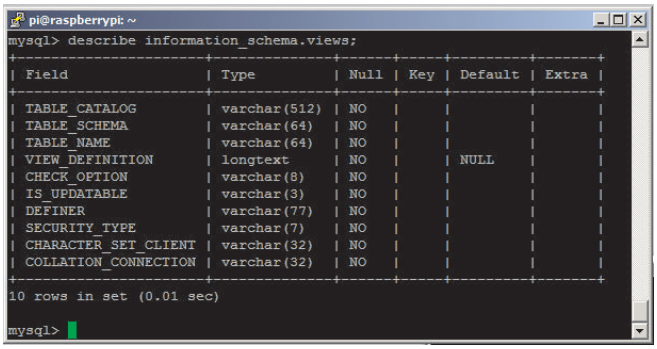
Cela affiche la liste des tables de la base de données information_schema. Les colonnes de la table peuvent être affichées de trois manières. Toutes produisent le même résultat :

```
pr0mpt > show columns in
information_schema.views;
pr0mpt > show fields in
information_schema.views;
pr0mpt > describe
information_schema.views;
```

Remarquez que le format générique de la commande précise la table et la base de données séparées par un point c'est-à-dire :

```
pr0mpt> show fields in
<databasename>.<tablename>
```

Travaillez là-dessus, essayez ces exemples et testez vos propres options pour tirer le meilleur de cet article. Un affichage type de sortie ressemble à ceci :



Ici vous pouvez voir les colonnes - appelées champs - et d'autres détails qui seront expliqués brièvement.

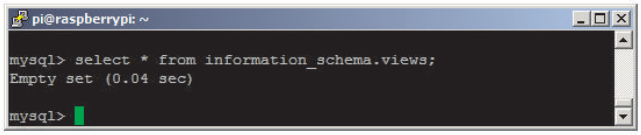
Affichage des données

Il est possible d'aller plus loin en choisissant de voir toutes les données de toutes les tables. L'étoile * est utilisée comme "joker" et signifie "tout", pour tout sélectionner dans une table entrez donc :

```
pr0mpt > select * from
information_schema.views;
```

Suite page suivante...

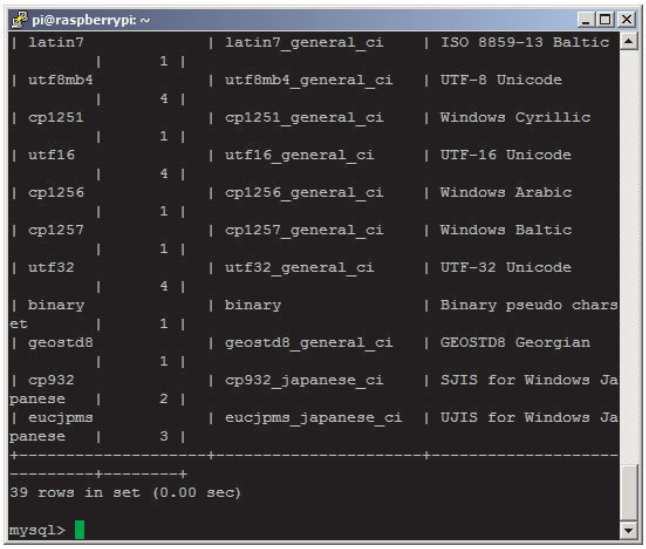
La sortie produite est extrême - c.-à-d. rien, un "ensemble vide". Notez également la durée ici - le Pi a pris quatre centièmes de seconde pour générer cette sortie.



L'autre option est représentée par :

```
pr0mpt > select * from
information_schema.CHARACTER_SETS;
```

Le résultat semble désordonné, et ça l'est, mais il devrait ressembler à quelque chose comme ça :



Tous les caractères |+ et - sont en fait les bordures de lignes d'un tableau brut qui ressemble à ceci :

```
+---------+---------+---------+----------+---------+----------+
| col_1 | col_2 | col_3 | col_4 | col_5 | col_6 |
+---------+---------+---------+----------+---------+----------+
| data | data | data | data | data | data |
| data | data | data | data | data | data |
+---------+---------+---------+----------+---------+----------+
```

Création d'une base de données

Maintenant le plus dur. MySQL peut se révéler très compliqué. Créons une base de données appelée magpi en tapant :

```
pr0mpt > create database magpi;
```

Vérifions que la base de données a bien été créée :

```
pr0mpt > show databases;
```

Il faut maintenant dire au système d'utiliser la nouvelle base de données magpi, alors tapez :

```
pr0mpt > use magpi;
```

Création d'une table

Une base de données peut contenir beaucoup de tables, comme nous l'avons vu. Les tables n'ont besoin d'être définies qu'une seule fois mais il est nécessaire de spécifier les types et tailles des données qu'il est prévu de stocker.
Une instruction générique ressemble à ceci :

```
pr0mpt > create table <tablename>
(
<col1><col1_data-type>,
<col2><col2_data-type>,
<col3><col3_data-type>,
( ( ( ,
);
```

Un exemple plus utile :

```
pr0mpt > create table tableoiseau
(
id int NOT NULL AUTO_INCREMENT,
first_name char(25),
town char(30) default ( Cardiff( ,
dob date,
birds int,
record timestamp default now(),
primary key (id)
);
```

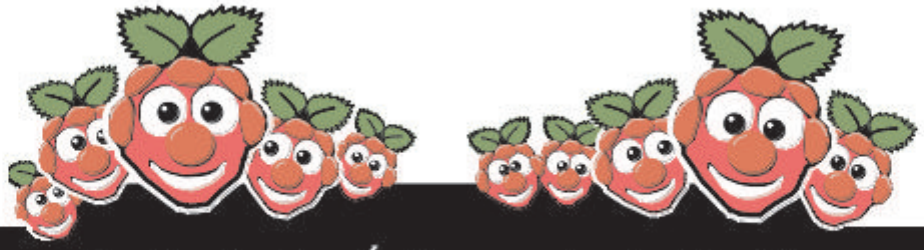
Vous pouvez à présent utiliser `describe tableoiseau;` pour retrouver la description désormais familière :

```
+------------+-------------+------+-----+-------------------+----------------+
| Field      | Type        | Null | Key | Default           | Extra          |
+------------+-------------+------+-----+-------------------+----------------+
| id         | int(11)     | NO   | PRI | NULL              | auto_increment |
| first_name | varchar(25) | YES  |     | NULL              |                |
| town       | char(30)    | YES  |     | Cardiff           |                |
| dob        | date        | YES  |     | NULL              |                |
| birds      | int         | YES  |     | NULL              |                |
| record     | timestamp   | NO   |     | CURRENT_TIMESTAMP |                |
+------------+-------------+------+-----+-------------------+----------------+
6 rows in set (0.01 sec)
```

La prochaine fois nous verrons comment insérer des données dans des tables pré-remplies et afficher les données en utilisant les tris, les recherches et les expressions logiques.

Article de Richard Wenner

Cet article est basé sur http://pr0mpt.me - Une vidéo libre de droits.



Le Guide des Événements du MagPi

Vous voulez savoir tout ce qui concerne le Raspberry Pi dans votre région ?
Alors cette nouvelle section du MagPi est pour vous ! Nous avons pour objectif de lister tous les événements Raspberry Jam dans votre région en vous fournissant un calendrier RPi pour le mois à venir.

Organisez-vous un événement Raspberry Pi ? Voulez-vous le promouvoir ?
Contactez-nous par courriel à : editor@themagpi.com

Bloominglabs Raspberry Pi Meetup

Quand : Premier Mardi de Chaque Mois @ 19h00
Où : Bloomington, Indiana, USA

Les réunions ont lieu le premier mardi de chaque mois de 19h00 jusqu'à 21h00, tout le monde est bienvenu. Plus de détails disponibles sur http://bloominglabs.org

Manchester Raspberry Jamboree

Quand : Samedi 9 mars 2013 @ 10h00
Où : Manchester Central Conference Venue, M2 3GX, UK

La réunion se déroulera de 10h00 à 16h00 et le nombre de places est limité.
Les billets et plus d'informations sont disponibles sur http://raspberryjamboree.eventbrite.com

Norwich Raspberry Pi User & DEV Group

Quand : Samedi 8 décembre 2012 @ 12h00
Où : House Cafe, 52 St. Benedicts Street, Norwich, UK

La réunion se déroulera de 12h00 à 17h00. Plus de détails disponibles sur http://norwichrpi.org

CERN Tarte au Framboise

Quand : Samedi 19 janvier 2013 @ 09h30
Où : Microcosm, CERN, 1211 Genève, Suisse

Cet événement se tiendra au mondialement célèbre CERN. Les portes ouvriront à 9h30 et la conférence durera jusqu'à 16h30. Davantage d'informations sont disponibles sur http://cern-raspberrypi.eventbrite.fr

Python peut lancer des sous-processus qui fonctionnent séparément. L'utilisation de cette approche rend possible la création de n'importe quel nombre de gadgets de bureau.

Nous allons créer deux gadgets : un lecteur de flux RSS rudimentaire et un programme qui télécharge l'image du site Astronomy Picture of the Day.

Lancez `magpi_widgets.py`. Ceci lancera les gadgets `widget_image.py` et `widget_rss.py`. Un unique CTRL+C dans le terminal va envoyer un signal de terminaison à chaque sous-processus. Ce code a besoin de modules Python supplémentaires, ces derniers pouvant être installés depuis le terminal :

```
sudo apt-get install python-setuptools
sudo easy_install-2.7 pip
sudo pip install feedparser # analyseur RSS
sudo pip install beautifulsoup4 # analyseur HTML
sudo apt-get install python-imaging-tk # manipulation d'images
```

magpi_widgets.py :

```
# Gadgets Python utilisant pygame et sous-processus
# Par ColinD - 02 novembre 2012

import subprocess, os, signal, Tkinter, time

# lance les sous-processus des gadgets - n'hésitez pas à en ajouter plus
ici !
pImg = subprocess.Popen(["python","widget_image.py"],stdin=subprocess.PIPE)
pRss = subprocess.Popen(["python","widget_rss.py"],stdin=subprocess.PIPE)

# envoie la largeur de l'écran aux sous-processus
r = Tkinter.Tk()
width = r.winfo_screenwidth()
pImg.stdin.write(str(width)+"\n")
pRss.stdin.write(str(width)+"\n")

# Exécution jusqu'à ce que les sous-processus soient tués par un seul CTRL-
C
try:
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    os.kill(pImg.pid, signal.SIGKILL)
    os.kill(pRss.pid, signal.SIGKILL)
```

widget_image.py :

```
import urllib, Image, pygame, os, sys, time
from bs4 import BeautifulSoup

# lit l'entrée standard depuis le processus parent et calcule la position
du gadget à l'écran
baseXPos = int(sys.stdin.readline()) - 200 - 10
os.environ['SDL_VIDEO_WINDOW_POS'] = str(baseXPos) + "," + str(30)

# affiche une fenêtre sans bordure pour contenir l'image redimensionnée
windowSurface = pygame.display.set_mode((200,200), pygame.NOFRAME)

while True:
    try:
        soup = BeautifulSoup(
                urllib.urlopen('http://apod.nasa.gov/apod/astropix.html'))
        # L'image d'APOD possède un tag ce qui nous permet d'utiliser find
au lieu de findAll
        imgTag = soup.find('img')
```

VERSION PYTHON : 2.7.3rc2
VERSION PYGAME : 1.9.2a0
O.S. : Debian 7

TESTED!

```
        imgUrl = imgTag['src']

    imgName = os.path.basename(imgUrl)

    # ne pas retélécharger l'image si elle existe déjà
    if not os.path.exists(imgName):
        urllib.urlretrieve("http://apod.nasa.gov/apod/"+imgUrl,imgName)

    # télécharge, redimensionne et enregistre l'image du gadget
    imgOriginal = Image.open(imgName)
    imgResized = imgOriginal.resize((200, 200), Image.NEAREST)
    imgResized.save(imgName)

     imgLoad = pygame.image.load(imgName)
    windowSurface.blit(imgLoad, (0,0))
    pygame.display.update()
# si une exception se produit, ignorer le téléchargement pour cette fois
except (IOError, TypeError, RuntimeError):
    print "Erreur de téléchargement, nouvel essai ultérieurement"

# pause de 8 heures car nous ne voulons pas inonder le serveur !
time.sleep(28800)
```

widget_rss.py :

```
import pygame, os, sys, feedparser, time
pygame.init()

# lit l'entrée standard à partir du processus parent et calcule la position
du gadget à l'écran
baseXPos = int(sys.stdin.readline()) - 300 - 10
os.environ['SDL_VIDEO_WINDOW_POS'] = str(baseXPos) + "," + str(430)

# création de la fenêtre Pygame et remplissage de l'arrière-plan avec des
blocs de couleur
screen = pygame.display.set_mode((300,150))
pygame.draw.rect(screen,(80,140,80),(0,0,300,50))
pygame.draw.rect(screen,(80,80,80),(0,50,300,50))
pygame.draw.rect(screen,(160,160,160),(0,100,300,50))

# définit la police pour afficher le texte RSS
font = pygame.font.SysFont('dejavuserif', 10, True)

while True:
    myFeed = feedparser.parse('http://www.raspberrypi.org/feed')

    # donne au titre de la fenêtre le nom du blog
    pygame.display.set_caption(myFeed['feed']['title']+" RSS")

    # récupère les articles depuis le flux RSS et affiche le texte
    for i in range(0, 3):
        layerText = pygame.Surface(screen.get_size())
        outputText = (myFeed['items'][i].title, myFeed['items'][i].updated,
                      myFeed['items'][i].link, myFeed['items'][i].description)
          # efface la surface à chaque boucle en la remplissant avec des
pixels transparents
        layerText.set_colorkey((0,0,0))
        layerText.fill((0,0,0))

        j = 0
        for line in outputText:
            j = layerText.get_rect().y + j + 5
            text = font.render(line.rstrip('\n'), 0, (255,255,255))
            textpos = text.get_rect()
            textpos.x = layerText.get_rect().x + 5
            textpos.y = textpos.y+j
            layerText.blit(text, textpos)
            screen.blit(layerText, (0, 50*i))
            pygame.display.flip()
            j = j +5
    # pause pendant une heure, ne pas inonder le serveur !
    time.sleep(3600)
```

Suggestions d'améliorations

L'affichage des gadgets n'est rafraîchi que lorsqu'il faut vérifier si un nouveau contenu doit être téléchargé, ce qui cause l'apparition d'un blanc si une autre fenêtre est glissée par dessus. Essayez d'utiliser datetime pour déterminer quand télécharger un nouveau contenu tandis que la mise à jour de l'affichage à l'écran se fera séparément chaque seconde.

L'année de The MagPi

Pour commencer, j'ai proposé l'idée du magazine The MagPi en mars dernier sur les forums Raspberry Pi après avoir lu quantité de messages affichés par des nouveaux en programmation qui souhaitaient apprendre ; ceux-ci ne savaient pas comment démarrer ou ignoraient même s'ils avaient assez d'expérience pour utiliser le Raspberry Pi.

À cette époque, il n'existait aucune documentation sur la façon d'utiliser ce petit ordinateur astucieux et la plupart des tutoriels et des projets visaient ceux qui avaient une certaine expérience dans l'utilisation de Linux ou qui avaient des connaissances en programmation. J'ai senti que cela pouvait éloigner beaucoup de débutants de l'apprentissage de la programmation ou les empêcher de prendre conscience de la grande variété de projets qui peuvent être réalisables avec le Pi sans beaucoup d'effort.

Il me semblait qu'il manquait quelque chose à la communauté Raspberry Pi pour cibler les programmeurs les plus expérimentés : un point de rencontre central, mis à part des forums, qui pourrait permettre à certains d'expliquer en détails aux autres comment reproduire leurs projets, montrer des photos et des vidéos d'expériences, et répondre aux questions de la communauté dans le but de venir en aide à ceux qui sont dans des situations similaires.

Issue du milieu médical avec une expérience de la publication d'articles dans des journaux, je me suis dit que la meilleure façon pour que la communauté puisse partager son expérience était de créer un endroit centralisé auquel chacun pouvait trouver ou partager des informations. Un média comme un journal en ligne constitué d'articles rédigés par des pairs paraissait correspondre à tous ces critères tout en se conformant à la philosophie éducative de la fondation.

Au cours des 8 derniers mois, le MagPi ainsi que l'équipe ont tous deux évolué. Nous sommes maintenant une société Limited et nous travaillons en étroite collaboration avec la fondation Raspberry Pi.

Nous encourageons les utilisateurs à

nous faire parvenir leurs projets et nous les incluons dans le brouillon du numéro à paraître. Les lecteurs consultent l'ébauche du prochain magazine et peuvent apporter des suggestions pour corriger ou clarifier ce qui a été écrit, ou des trucs et astuces qui pourraient être bénéfiques.

Nous réalisons pour le moment un magazine mensuel de 32 pages. Le contenu est composé d'articles couvrant une large variété de thèmes relatifs au Raspberry Pi, parmi lesquels la programmation, la robotique, la domotique, l'électronique et des cas pratiques pour n'en nommer que quelques-uns.

Nous sommes très reconnaissants envers la fondation Raspberry Pi et nos lecteurs pour tout le soutien qu'ils nous apportent. Nous avons été submergés par la façon dont nous avons été positivement reçus, avec des retours continus pour en témoigner, et la parution d'articles importants consacrés au magazine, comme ceux du Wall Street Journal ou de Rory Cellan-Jones de la BBC !

Nous avons grandi en taille pour devenir une équipe, avec des rédacteurs et des contributeurs du monde entier, de toutes les tranches d'âge et de toutes les professions, depuis les éducateurs jusqu'aux médecins, du technicien à l'étudiant. Pour rendre compte de l'intérêt mondial suscité par le MagPi, ce dernier a été traduit en français et en allemand, et des traductions en chinois et en espagnol sont en cours.

Nous sommes super enthousiasmés par nos plans de 2013 pour le MagPi, parmi lesquels notre projet Kickstarter qui donnera à nos lecteurs la chance d'avoir l'intégralité des huit numéros dans un classeur à édition limitée. Nous espérons que vous continuerez à nous soutenir pour la Nouvelle Année.

Ash Stone
Rédactrice en chef du MagPi

The MagPi™

editor@themagpi.com