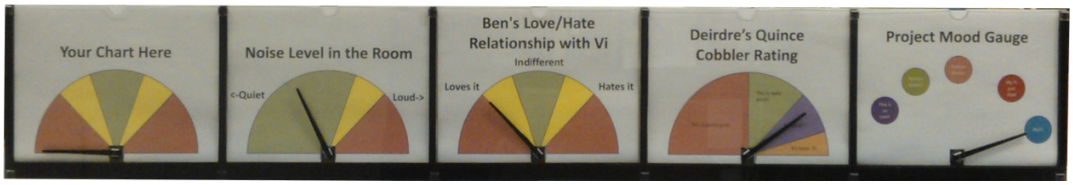
Jauge Pi

DIFFICULTÉ : Facile-Moyenne

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

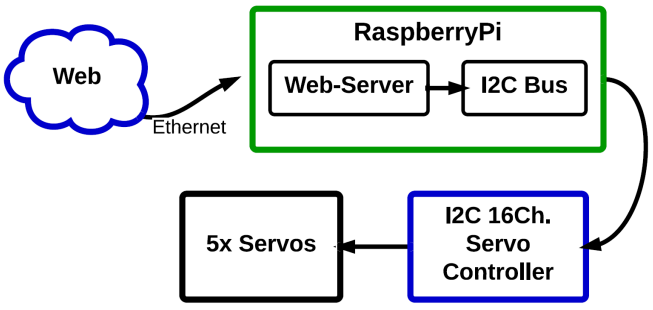


En résumé

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.

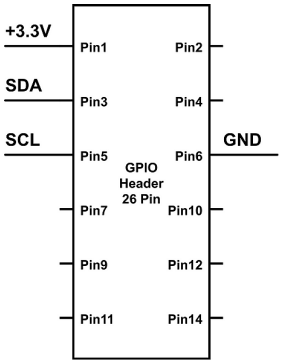
```
$ sudo shutdown -h now
```

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.

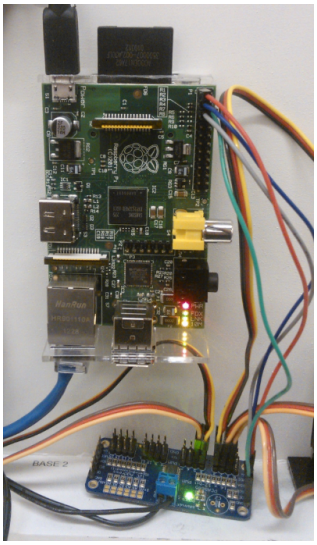


Vérifiez TOUTES vos connexions plusieurs fois AVANT la mise sous tension.

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
```

Mettez en commentaires les lignes du 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