Raspberry Pi

Python et le port GPIO

Par Alexandre GALODE

Date de publication : 16 mai 2015

TOUT PUBLIC

Il est possible de trouver de multiples utilisations du Raspberry Pi sur Internet : serveur, passerelle multimédia, PC d'appoint… Tous ces projets mettent en œuvre le mini PC lui-même.

Cependant, au-delà de l'aspect mini PC embarqué, et de son OS Linux, cette cible possède un atout important : son port d'entrées-sorties, ou GPIO. Et ce dernier est souvent mal connu ou non exploité.

Commentez

I - Introduction

Source:http://www.clker.com/clipart-raspberry-pi.html

Ce tutoriel vise à vous fournir l'essentiel des informations nécessaires afin de vous permettre de créer vos propres projets, tout en exploitant au mieux le port GPIO.

Le port GPIO (General Purpose Input/Output, ou Entrées/Sorties pour un usage générique) mettra à votre disposition de nombreuses entrées-sorties, une alimentation, un port SPI, un port I2C, et un port série.

Après une rapide présentation du Raspberry Pi, de ses différentes versions existantes, et des caractéristiques liées, nous nous pencherons sur le fonctionnement et la programmation du port GPIO.

Enfin, nous finirons par une mise en pratique de la théorie que nous aurons vue précédemment.

Bien que nous essayons de rester le plus clair possible, il se peut que certains concepts électroniques soient néanmoins évoqués. Je vous renvoie alors vers votre moteur de recherche préféré afin de mieux les appréhender.

II - Le Raspberry Pi

II-A - Présentation rapide

Le Raspberry Pi est une cible embarquée de type mini PC à moins de 50 €.

Le but initial du projet était de permettre aux étudiants de s'équiper du même matériel qu'en cours, et ce à moindres frais.

Après plusieurs années de développement, et plusieurs versions, le projet a bien évolué. Malgré son faible coût, la carte dispose d'un panel d'entrées-sorties honorable et se prête aussi bien à des applications statiques qu'embarquées.

II-B - Les systèmes d'exploitation disponibles

Bien que consistant en une cible embarquée, le Raspberry Pi dispose néanmoins d'un OS officiel appréciable, et non des moindres, puisqu'il s'agit tout simplement d'un Linux, entre autres une Debian. Cet OS a été surnommé **Raspbian**.

La couche graphique est, par défaut, totalement désactivée, mais est très simple à lancer.

La communauté étant très active sur notre plateforme adorée, il est également possible d'y retrouver, entre autres, une Fedora, ou encore XBMC/Kodi, ou OSMX.

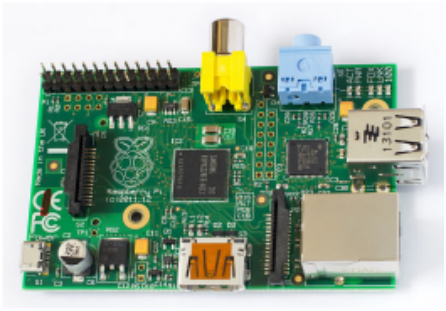
Parmi les OS annoncés dans l'avenir proche, nous pourrons citer **le prochain Microsoft Windows**, ainsi que Firefox OS.

II-C - Modèles et caractéristiques

Nous nous pencherons ici sur les modèles les plus couramment diffusés et ferons l'impasse sur les tout premiers modèles, ainsi que sur la gamme professionnelle, au format DIMM.

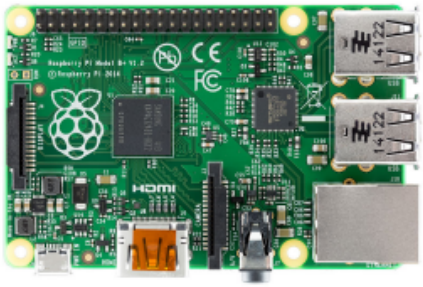
Modèle A

Modèle B

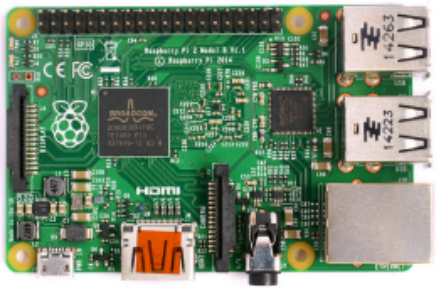


Modèle A+

Modèle B+



Modèle B2



Source:http://fr.wikipedia.org/wiki/Raspberry_Pi

Modèle	**A**	**A+**	**B**	**B+**	**B2**
CPU	Monocœur ARM 700 MHz				Quadricœur ARM 900 MHz
GPU	Décodeur vidéo Broadcam VideoCore IV				
RAM	256 MO		512 MO		1 GO
USB	1 * USB2.0		2 * USB2.0	4 * USB2.0	
Audio/vidéo	Jack 3.5, composite et HDMI	HDMI et jack audio/vidéo	Jack 3.5, composite et HDMI	HDMI et jack audio/vidéo	
Ethernet	0	10/100	0	10/100	
Entrées/sorties	GPIO 26 pts	GPIO 40 pts	GPIO 26 pts	GPIO 40 pts	
OS	Officiel : Raspbian				

	Tiers : Fedora, XBMC/Kodi, OSMC			
Stockage	SD	Micro SD	SD	Micro SD
Dimensions	86*54*17	65*54*17	86*54*17	
Poids	45g	23g	45g	
Consommation	1.5W	1W	3.5W	3W

L'alimentation, quelle que soit la carte, s'effectue par un connecteur micro#USB. Les premières générations pouvaient être alimentées par un port USB, via un HUB. Ce n'est plus le cas des nouvelles versions (A+/B+/B2).

Ainsi, comme vous pouvez le constater, il existe toujours la possibilité de trouver un modèle plus adapté à son projet que les autres.

II-D - Le port GPIO

Les GPIO ne possèdent aucune protection. Il ne faut donc pas oublier d'utiliser des buffers ou des optocoupleurs si vous avez des doutes quant à votre montage.

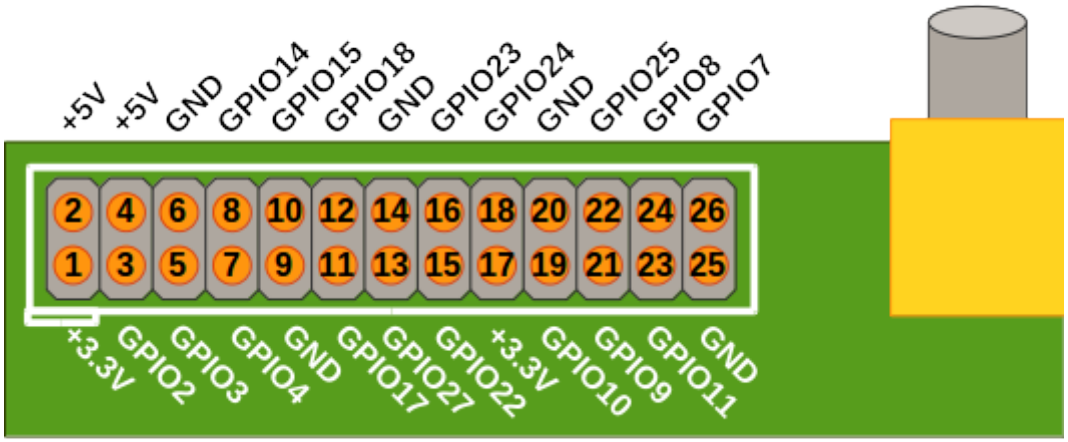
Les modèles A et B possédaient un port GPIO à 26 points. Les modèles A+, B+ et B2 possèdent un port GPIO à 40 points, correspondant aux 26 points des modèles A/B plus 14 points supplémentaires. Un circuit conçu pour les versions A/B est donc compatible A+/B+/B2.

Pour la suite nous présenterons les deux versions à chaque fois.

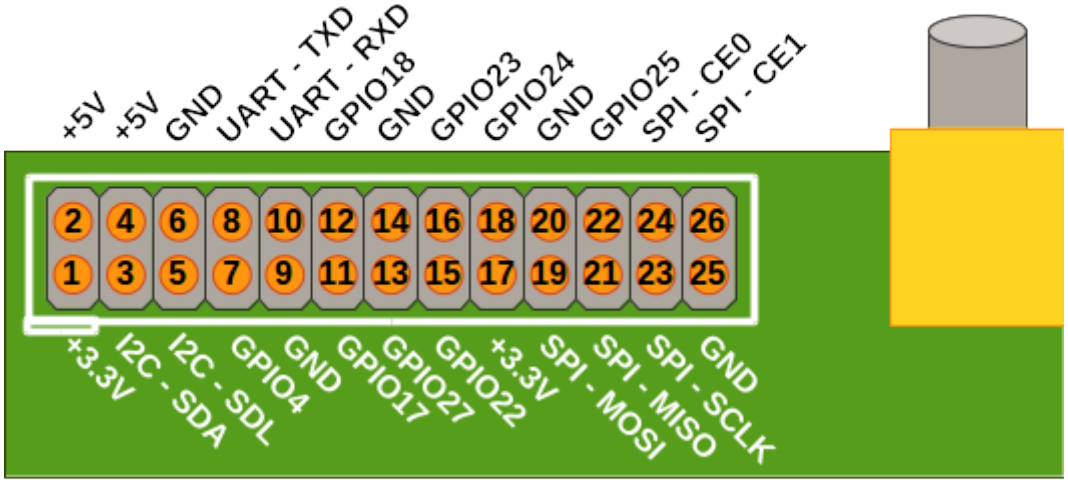
Concernant les modèles A et B, il y a eu plusieurs révisions. Le câblage du GPIO a été modifié entre ces différentes versions. Le GPIO des versions A et B que nous verrons ici correspond à la dernière révision, qui est également la plus répandue : la 2.1.

Les entrées-sorties sont de type CMOS, et fonctionnent donc en +3.3V. N'essayez surtout pas de brancher des circuits TTL (+5V) sur les entrées-sorties, sans adaptation. Vous risqueriez de détruire votre Raspberry Pi.

II-D-1 - Raspberry Pi - modèles A et B

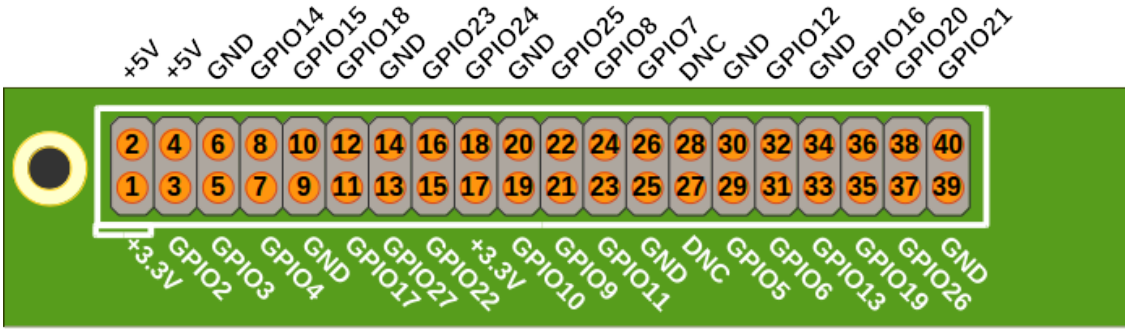


Fonctionnalités normales du port GPIO

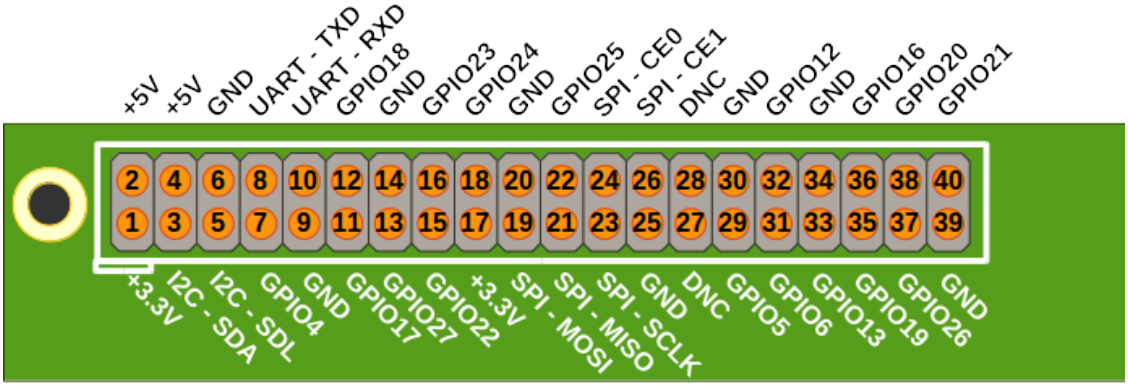


Fonctionnalités étendues du port GPIO

II-D-2 - Raspberry Pi - modèles A+, B+, et B2



Fonctionnalités normales du port GPIO



Fonctionnalités étendues du port GPIO

L'ensemble de ces fonctionnalités peuvent être activées une à une selon vos besoins.

Dans le cas du SPI, on remarquera qu'il y a deux « Chip Enable ». Nativement, le Raspberry Pi est donc capable de gérer deux SPI en parallèle.

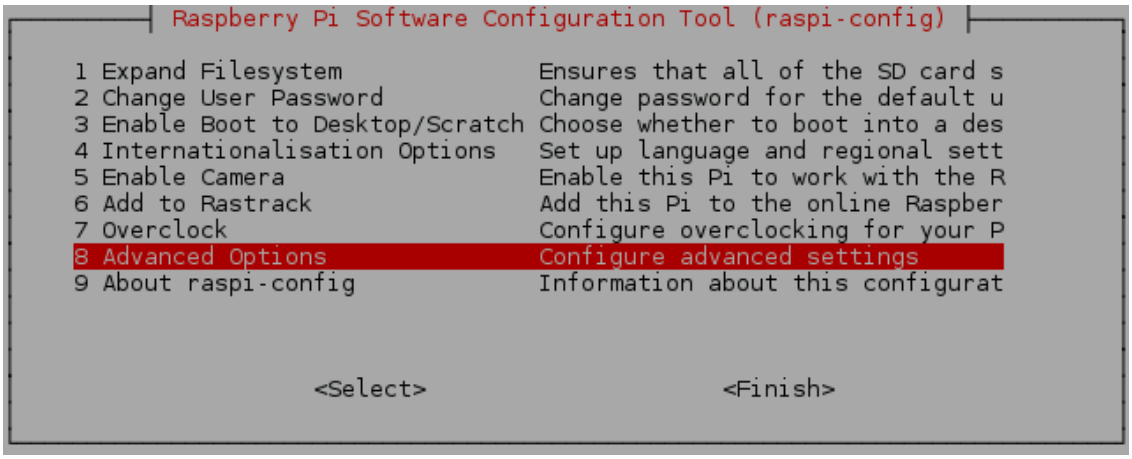
II-D-3 - Activation de l'I2C

La méthode que nous allons voir est la méthode standardisée à partir des versions « + ». Elle est donc compatible modèles A+, B+, B2.

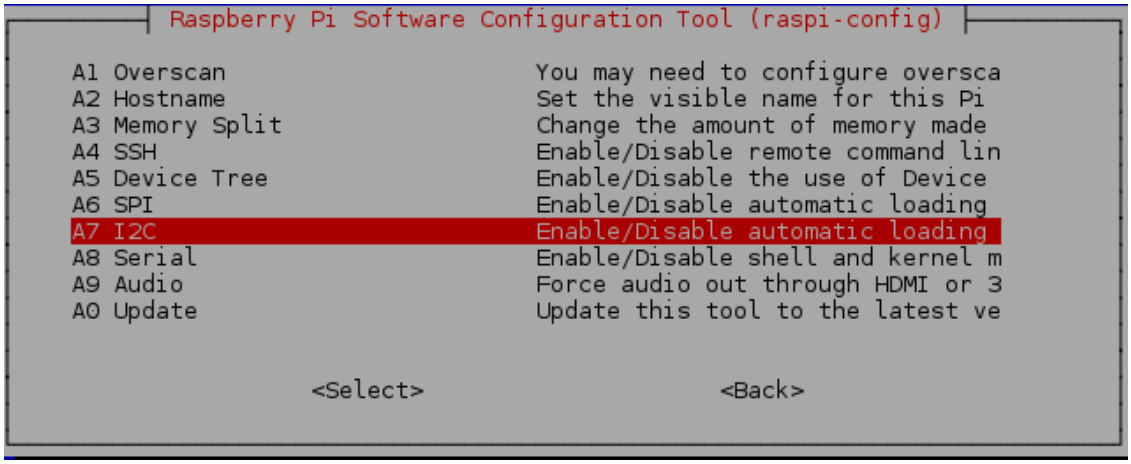
- Ouvrez un terminal, depuis une interface graphique, et/ou lancez la commande suivante :

```
sudo raspi-config
```

- Cela lance l'interface de configuration du Raspberry Pi, sélectionnez « Advanced Options » puis validez :



- Sur le nouvel écran, sélectionnez « I2C » puis validez :



- Répondez « Yes » à toutes les questions qui suivent.
- Rebootez votre machine.

```
sudo reboot
```

- Testez que l'I2C est bien reconnu avec la commande suivante (elle doit vous renvoyer une ligne d'information) :

```
lsmod | grep i2c_
```

Voilà, vous disposez de l'I2C activé par défaut sur votre Raspberry Pi.

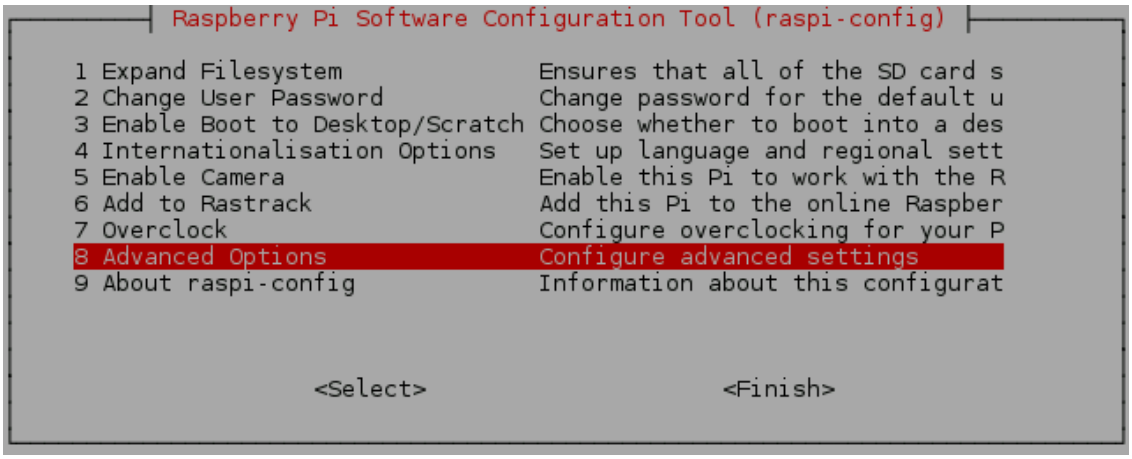
II-D-4 - Activation du SPI

La méthode que nous allons voir est la méthode standardisée à partir des versions « + ». Elle est donc compatible modèles A+, B+, B2.

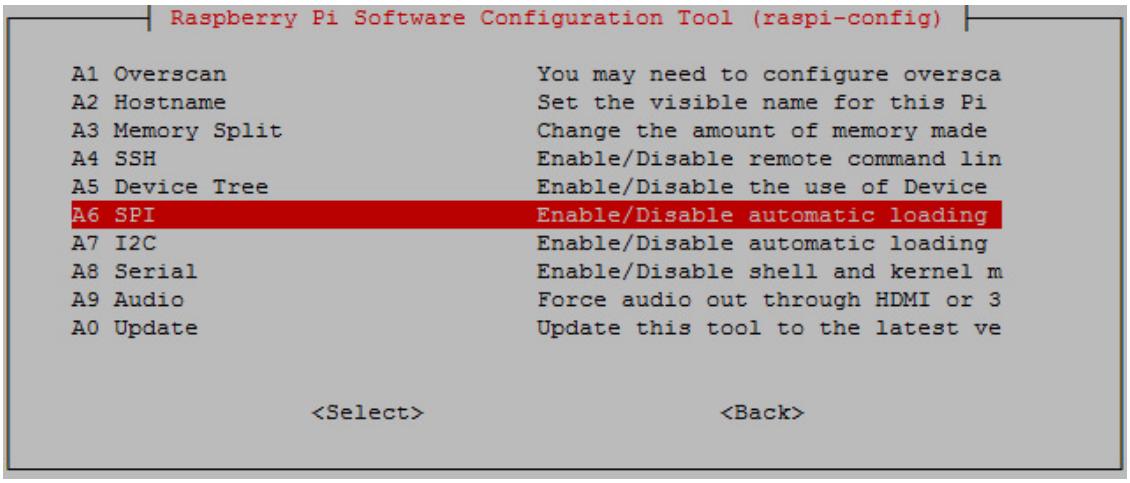
- Ouvrez un terminal, depuis une interface graphique, et/ou lancez la commande suivante :

```
sudo raspi-config
```

- Cela lance l'interface de configuration du Raspberry Pi, sélectionnez « Advanced Options » puis validez :



- Sur le nouvel écran, sélectionnez « SPI » puis validez :



- Répondez « Yes » à toutes les questions qui suivent.
- Rebootez votre machine.

```
sudo reboot
```

- Testez que le SPI est bien reconnu avec la commande suivante (elle doit vous renvoyer une ligne d'information) :

```
lsmod | grep spi_
```

Voilà, vous disposez du SPI activé par défaut sur votre Raspberry Pi.

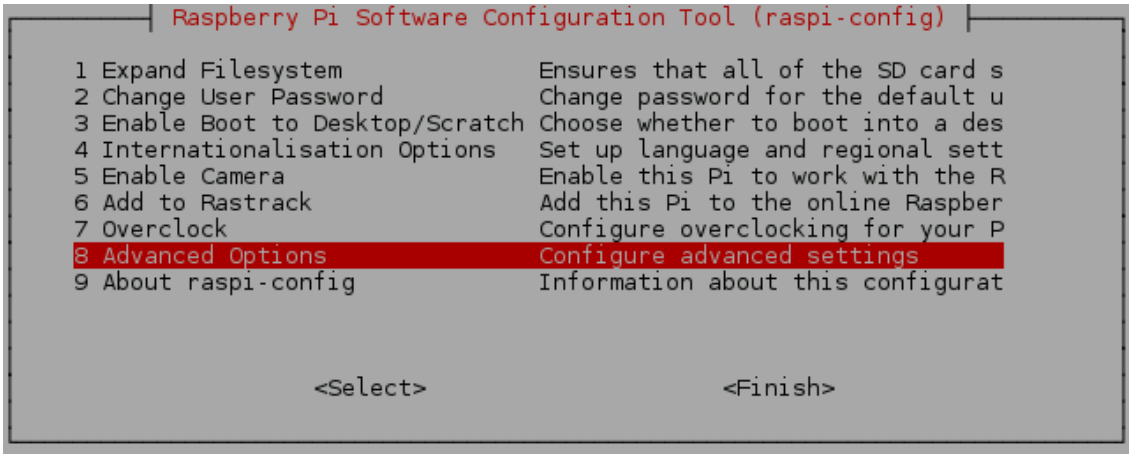
II-D-5 - Activation de l'UART et des autres périphériques

Je pense que vous avez dès à présent compris comment activer ou non les différentes options de votre Raspberry Pi.

- Ouvrez un terminal, depuis une interface graphique, et/ou lancez la commande suivante :

```
sudo raspi-config
```

- Cela lance l'interface de configuration du Raspberry Pi, sélectionnez « Advanced Options » puis validez :



- Sur le nouvel écran, sélectionnez l'option que vous voulez paramétrer, ici « serial » puis validez.

- Répondez « Yes » à toutes les questions qui suivent.
- Rebootez votre machine :

```
sudo reboot
```

Bien entendu, il vous faudra à chaque fois une bibliothèque adaptée pour pouvoir utiliser votre périphérique.

III - Python et le port GPIO

Pour la suite, nous partirons sur l'hypothèse que nous développons sur le dernier modèle à savoir un Raspberry Pi B2.

III-A - Les bibliothèques disponibles et recommandées

Il existe de nombreuses bibliothèques dédiées au Raspberry Pi. Elles sont stockées, pour la plupart, sur Pypi.

Celle que nous étudierons ici est la bibliothèque historique : RPi.GPIO.

Pour l'installer, il suffit donc d'un simple pip install :

```
pip install RPi.GPIO
```

Si vous ne possédez pas encore pip, il vous suffit de charger get-pip.py, puis de le lancer via la commande ci-après. Une connexion Internet vous sera nécessaire. La première ligne de commande est pour ceux qui exécuteraient le tout sans interface graphique.

```
1. wget https://bootstrap.pypa.io/get-pip.py
2. python get-pip.py
```

Bien entendu, n'oubliez pas d'utiliser un « sudo » si nécessaire.

Pour rappel, Python est sensible à la casse. Attention donc à l'import de RPi.GPIO. RPi s'écrit avec un i minuscule, et tout le reste est en majuscules.

Le port GPIO n'est accessible qu'en mode root. N'oubliez donc pas de lancer Python avec la commande sudo, sinon cela ne fonctionnera pas.

```
sudo python mon_script.py
```

III-B - La bibliothèque GPIO

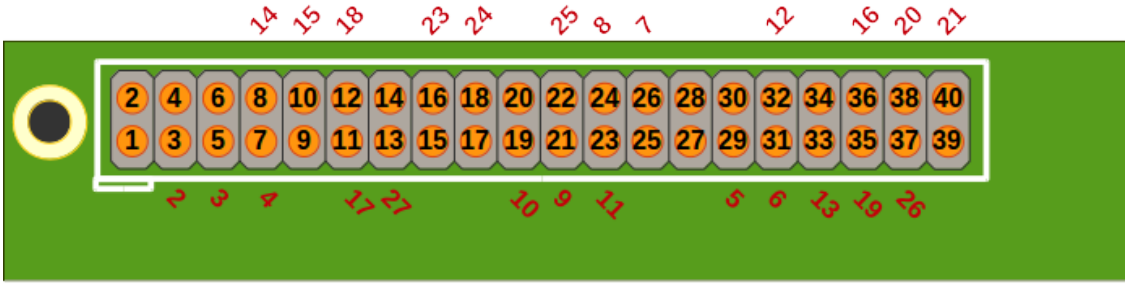
```
import RPi.GPIO as GPIO
```

Concernant l'état des entrées-sorties (E/S), le module RPi.GPIO accepte des variables dédiées, des entiers ou des booléens.

Ainsi, l'état haut peut valoir au choix **GPIO.HIGH**, **1** ou encore **True**. De même, l'état bas peut valoir **GPIO.LOW**, **0** ou **False**.

III-B-1 - Configuration générale

Le Raspberry Pi autorise deux numérotations : celle de la sérigraphie du connecteur de la carte (**GPIO.BOARD**), ou la numérotation électronique de la puce (**GPIO.BCM**). À vous de choisir celle que vous voulez.



En noir la numérotation GPIO.BOARD. En rouge, la numérotation GPIO.BCM

```
1. GPIO.setmode(GPIO.BOARD)
2. GPIO.setmode(GPIO.BCM)
3.
4. # On peut aussi demander la configuration etablie
5. configuration = GPIO.getmode()
```

III-B-2 - Configuration des entrées-sorties

Afin d'initialiser une E/S, il suffit de préciser son numéro, fonction de la configuration précédemment choisie, son paramétrage (entrée ou sortie) et éventuellement son état initial (pour une sortie).

```
1. GPIO.setup(12, GPIO.IN)                    # broche 12 est une entree numerique
2. GPIO.setup(12, GPIO.OUT)                   # broche 12 est une sortie numerique
3. GPIO.setup(12, GPIO.OUT, initial=GPIO.HIGH)# broche 12 est une sortie initialement a l'etat haut
```

III-B-3 - Lire une entrée numérique

Afin de lire l'état d'une entrée, il suffit de préciser le numéro de l'E/S.

```
GPIO.input(12)
```

III-B-4 - Changer l'état d'une sortie numérique

Pour modifier l'état d'une sortie, il faut indiquer le numéro de la sortie, ainsi que l'état désiré.

```
GPIO.output(12, GPIO.LOW)
```

Un *toggle* (inversement d'état) s'effectue en inversant l'état lu.

```
GPIO.output(12, not GPIO.input(12))
```

III-B-5 - Connaître la configuration d'une entrée-sortie numérique

On peut interroger l'E/S afin de connaître son état de configuration. Les valeurs renvoyées sont alors **GPIO.INPUT**, **GPIO.OUTPUT**, **GPIO.SPI**, **GPIO.I2C**, **GPIO.HARD_PWM**, **GPIO.SERIAL** ou **GPIO.UNKNOWN**.

```
1. state = GPIO.gpio_function(pin)
2. print(state)
```

III-B-6 - Remettre à zéro une entrée-sortie numérique

À la fin de tout programme, il est conseillé d'effectuer une purge des ressources afin de les remettre dans l'état où le programme les a trouvées. On n'impactera ainsi que les E/S utilisées.

```
GPIO.cleanup()
```

III-B-7 - La PWM

La PWM pour *Pulse Width Modulation* consiste en un signal carré dont on fait varier le rapport cyclique, en d'autres termes, dont on fait varier la durée de l'état haut, par rapport à l'état bas.

Source : http://commons.wikimedia.org/wiki/File:PWM_duty_cycle_with_label.gif

La PWM fonctionne comme un objet, en Python et sur Raspberry Pi. Il faut donc commencer par créer une instance PWM, en déclarant le canal (*channel*) utilisé, ainsi que la fréquence désirée. En cours de route, vous pourrez changer aussi bien le rapport cyclique, que la fréquence.

Bien que n'ayant pas été notée sur le brochage du connecteur du GPIO, pour des raisons de clarté, la PWM est disponible sur GPIO18.

```
1. p = GPIO.PWM(channel, frequence)
2. p.start(rapport_cyclique) #ici, rapport_cyclique vaut entre 0.0 et 100.0
3. p.ChangeFrequency(nouvelle_frequence)
4. p.ChangeDutyCycle(nouveau_rapport_cyclique)
5. p.stop()
```

III-B-8 - Pull up et pull down

Afin d'éviter de laisser flottante toute entrée, il est possible de connecter des résistances de pull-up ou de pull-down, au choix, en interne.

Pour information, une résistance de pull-up ou de pull-down a pour but d'éviter de laisser une entrée ou une sortie dans un état incertain, en forçant une connexion à la masse ou à un potentiel donné.

```
1. GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_UP)
2. GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

III-B-9 - Interruption et détection de front

La détection de front est quelque chose d'important, et liée aux interruptions. Il existe trois façons de procéder.

La première consiste à bloquer l'exécution du programme jusqu'à ce que l'événement se produise.

```
GPIO.wait_for_edge(channel, GPIO.RISING)
```

Le second paramètre peut ici valoir **GPIO.RISING**, **GPIO.FALLING** ou **GPIO.BOTH**.

La seconde solution est liée à l'utilisation d'une boucle. On ajoute des événements à écouter sur un canal, et au sein de la boucle while, on teste si quelque chose s'est produit sur un des canaux écoutés.

```
1. GPIO.add_event_detect(channel, GPIO.RISING)
2. While True:
3.     if GPIO.event_detected(channel):
4.         print("Bouton appuye")
```

Enfin, la dernière solution est celle qui est la plus proche du concept d'interruption sur microcontrôleur. Pour ce faire, en cas d'événement, un thread parallèle est lancé en cas d'interruption. Trois valeurs sont possibles : front montant (**GPIO.RISING**), front descendant (**GPIO.FALLING**) ou bien les deux (**GPIO.BOTH**)).

```
1. Def my_callback(channel):
2.     print("un evenement s'est produit")
3.
4. #ici on ajoute une tempo de 75 ms pour eviter l'effet rebond
5. GPIO.add_event_detect(channel, GPIO.BOTH, callback=my_callback, bouncetime=75)
6. #votre programme ici
```

Enfin, au besoin, vous pouvez supprimer les interruptions sur un canal très simplement.

```
GPIO.remove_event_detect(channel)
```

III-B-10 - Le nécessaire pour l'I2C

Nous avons précédemment vu comment paramétrer le Raspberry Pi pour activer l'I2C. Nous allons voir ici comment installer le nécessaire pour piloter l'I2C en Python.

La bibliothèque utilisée s'appelle « smbus-cffi » et est disponible sur Pypi, via pip. Elle requiert des dépendances.

Voici la liste des commandes à exécuter :

```
1. sudo apt-get install build-essential libi2c-dev python-dev
2. sudo apt-get install libffi-dev i2c-tools
3. sudo pip install smbus-cffi
```

III-B-11 - Le nécessaire pour le SPI

Nous avons précédemment vu comment paramétrer le Raspberry Pi pour activer le SPI. Nous allons voir ici comment installer le nécessaire pour piloter le SPI en Python.

La bibliothèque utilisée s'appelle « spidev ». Elle est disponible via pip, mais requiert cependant des dépendances.

Voici donc la liste des commandes à exécuter pour installer l'ensemble :

```
1. sudo apt-get install python2.7-dev
2. sudo apt-get install python3-dev
3. sudo apt-get libevent-dev
4. sudo pip install spidev
```

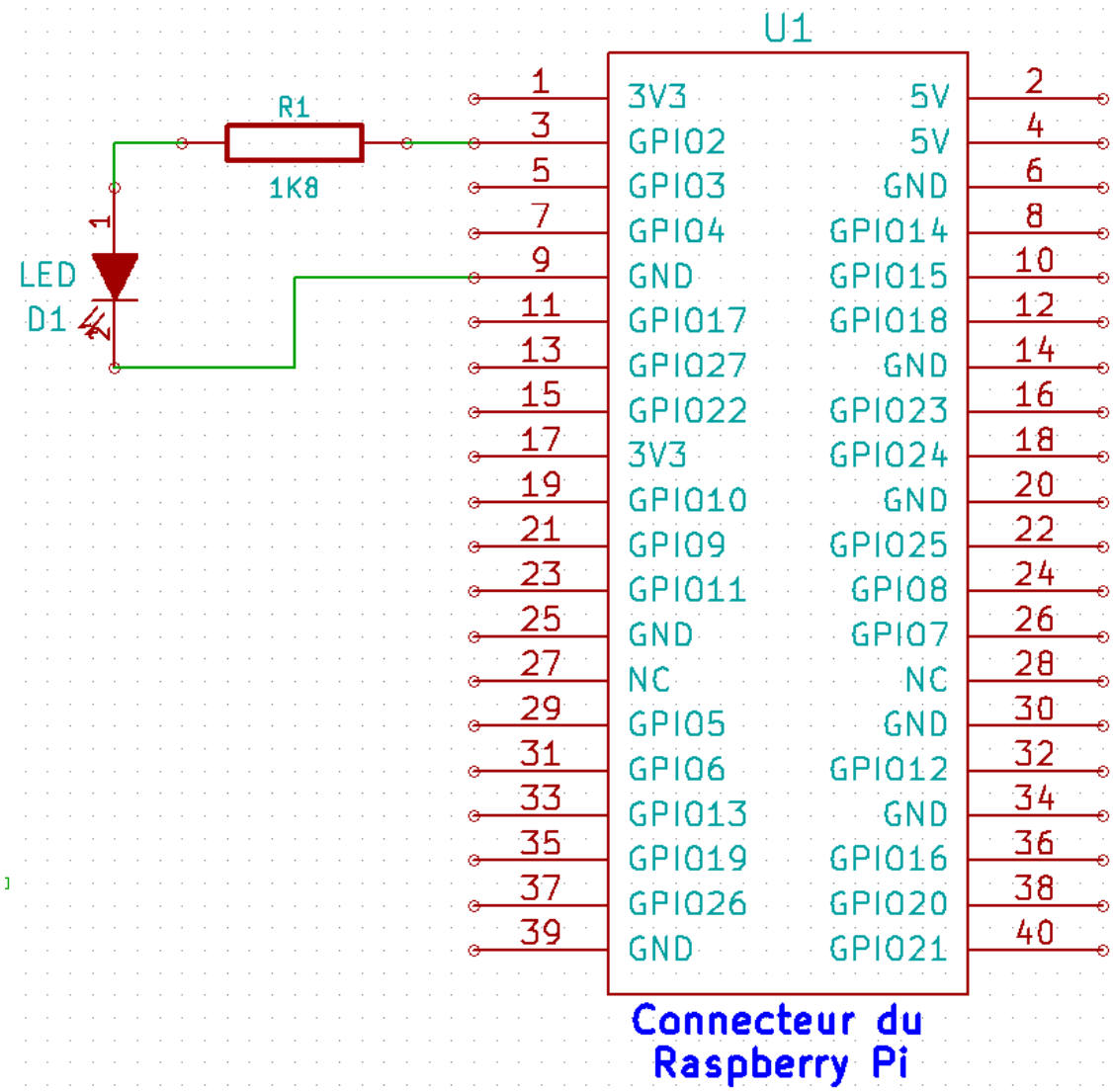
III-C - Mise en pratique

Maintenant que nous avons vu toute la partie théorique, je vous propose un peu de mise en pratique, avec quelques exemples concrets.

III-C-1 - Gestion d'une LED

Nous allons ici piloter notre LED de deux façons. Tout d'abord, nous la ferons simplement clignoter. Basique, mais incontournable. Par la suite, nous ferons varier son intensité via la PWM.

La LED sera branchée sur la broche (*pin* en anglais) 3 du GPIO (GPIO2).



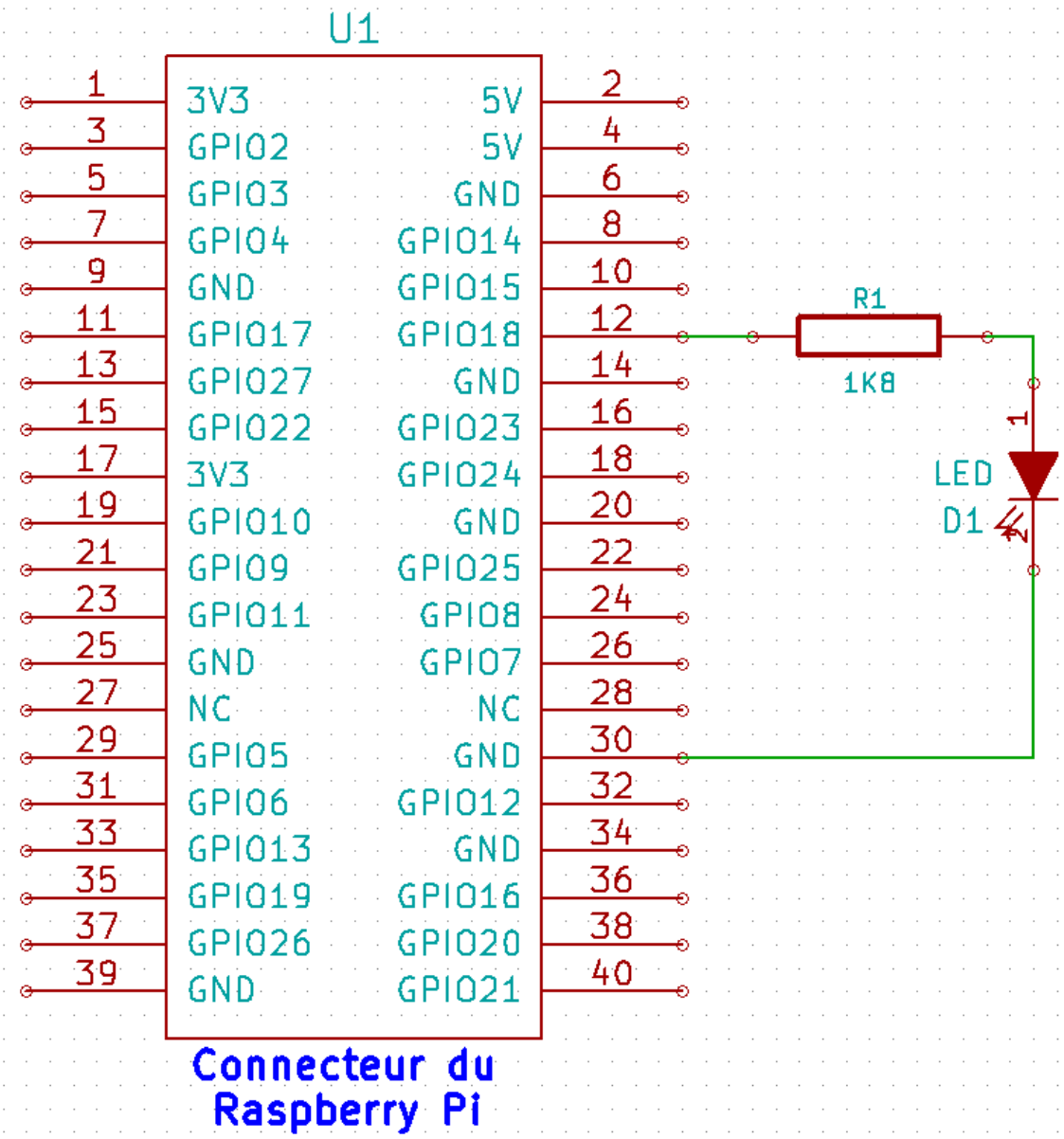
Pour le PWM nous utiliserons cette dernière de manière à modifier la quantité d'énergie transmise. Pour cela, nous choisirons une fréquence de 200 Hz, et le rapport cyclique évoluera dans le temps.

Blink.py

```
1. # Import des modules
2. import RPi.GPIO as GPIO
3. import time
4.
5. # Initialisation de la numerotation et des E/S
6. GPIO.setmode(GPIO.BOARD)
7. GPIO.setup(3, GPIO.OUT, initial = GPIO.HIGH)
8.
9. # On fait clignoter la LED
10. while True:
11.     GPIO.output(3, not GPIO.input(3))
12.     time.sleep(0.5)
```

Cliquer sur ce lien pour lancer l'animation

Bien entendu, pour éclaircir votre code, vous avez la possibilité d'utiliser des variables intermédiaires. Par exemple, dans le code que nous venons de voir, vous auriez pu remplacer le « 3 » par une variable « led ».



Pour la PWM, nous devrons brancher notre LED sur la GPIO18 (broche 12), qui est la seule à proposer cette fonctionnalité.

Pwm.py

```
1. # Import des modules
2. import RPi.GPIO
3. import time
4.
5. # Initialisation de la numerotation et des E/S
6. GPIO.setmode(GPIO.BOARD)
7. GPIO.setup(12, GPIO.OUT, initial = GPIO.LOW)
8.
9. Rapport = 10.0
10. sens = True
11.
12. p = GPIO.PWM(12, 200)
13. p.start(rapport) #ici, rapport_cyclique vaut entre 0.0 et 100.0
14.
15. # On fait varier l'intensite de la LED
16. while True:
17.     if sens and rapport < 100.0:
18.         rapport += 10.0
19.     elif sens and rapport >= 100.0:
```

Pwm.py
```
20.         sens = False
21.     elif not sens and rapport > 10.0:
22.         rapport -= 10.0
23.     elif rapport == 10.0:
24.         sens = True
25.     p.ChangeDutyCycle(rapport)
26.     time.sleep(0.25)
```

Cliquer sur ce lien pour lancer l'animation

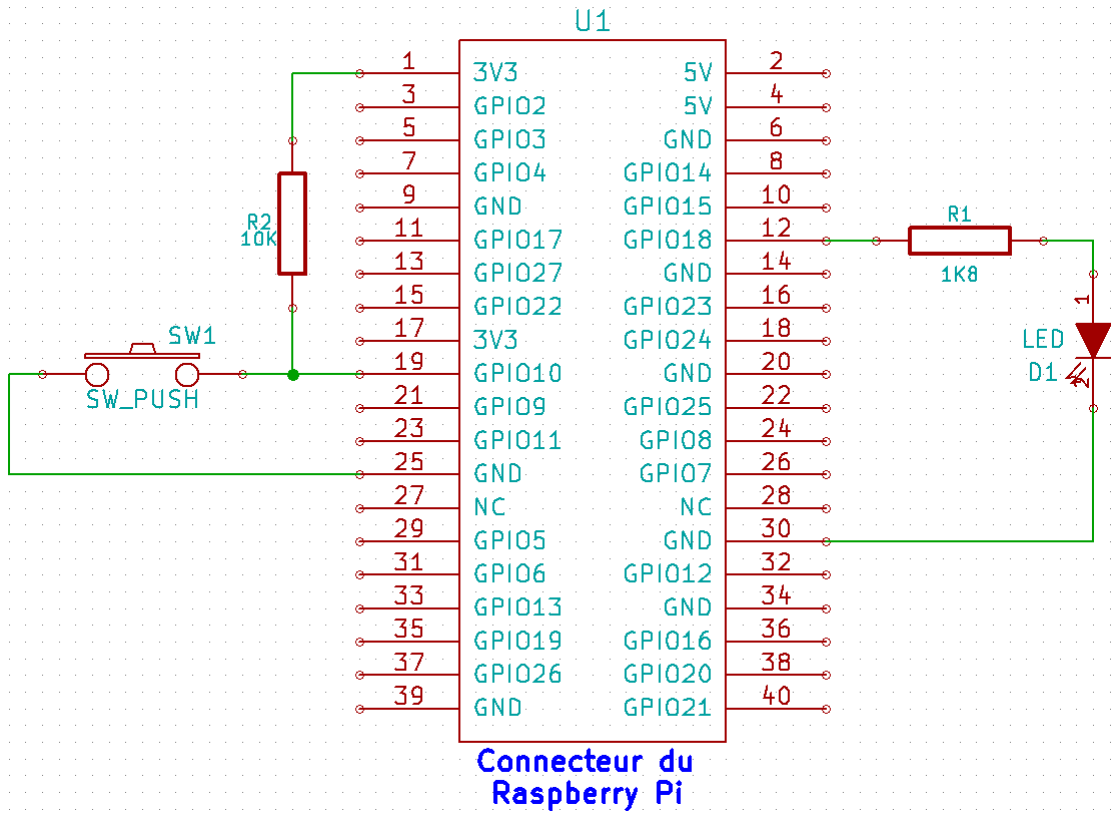
III-C-2 - Lecture d'un bouton

Lorsque nous désirons lire l'état d'un bouton, nous avons deux solutions : venir lire l'état à intervalle régulier, ou bien programmer une interruption et vaquer au reste du programme.

Une fois le bouton appuyé, nous attendrons le relâchement tout simplement.

Les deux pattes de notre bouton seront branchées sur une masse et la broche 19 du GPIO (GPIO10). Une résistance de pull-up sera également branchée entre une des pattes de +3,3V et la broche 19.

Enfin, notre LED sera branchée cette fois entre une masse et la broche 12 (GPIO18).



Button.py
```
1. # Import des modules
2. import RPi.GPIO
3. import time
4.
5. # Initialisation de la numerotation et des E/S
6. GPIO.setmode(GPIO.BOARD)
7. GPIO.setup(12, GPIO.OUT, initial=GPIO.LOW)
8. GPIO.setup(19, GPIO.IN)
9.
```

Button.py
```
10. # Si on detecte un appui sur le bouton, on allume la LED
11. # et on attend que le bouton soit relache
12. while True:
13.     state = GPIO.input(19)
14.     if not state:
15.         # on a appuye sur le bouton connecte sur la broche 19
16.         GPIO.output(12, GPIO.HIGH)
17.         while not state:
18.             state = GPIO.input(19)
19.             time.sleep(0.02)  # Pause pour ne pas saturer le processeur
20.         GPIO.output(12, GPIO.LOW)
21.     time.sleep(0.02)  # Pause pour ne pas saturer le processeur
```

Cliquer sur ce lien pour lancer l'animation

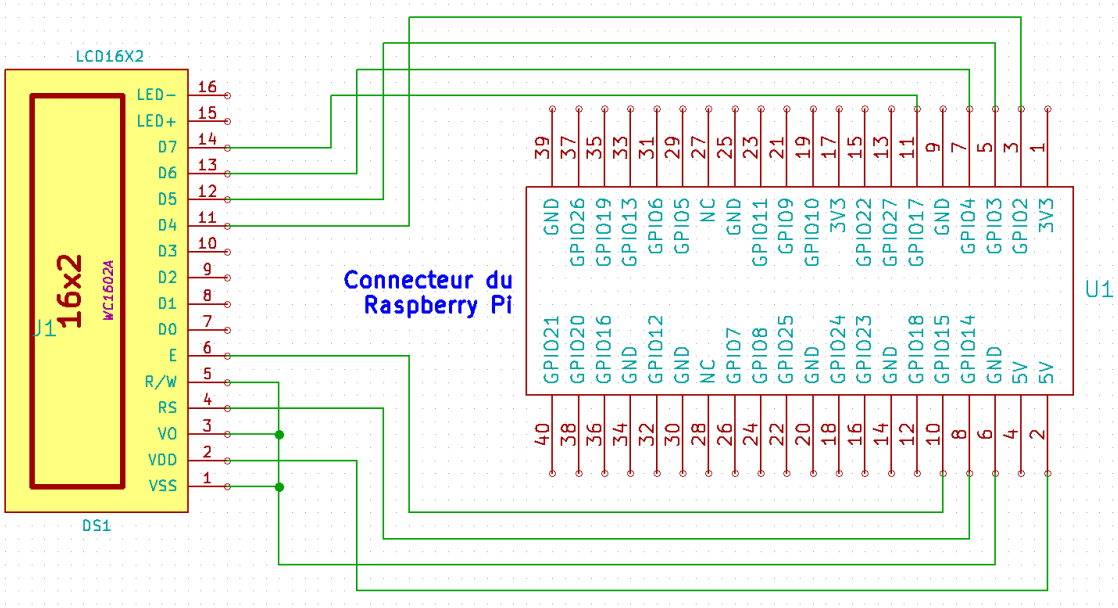
III-C-3 - Pilotage d'un afficheur LCD

Pour cette partie, nous utiliserons un afficheur LCD à base de HD44780, piloté non pas sur 8 bits, mais en mode 4 bits.

Malgré le fait que l'écran ne fonctionne qu'en 5V, nous pouvons tout de même le brancher en direct sur le Raspberry, car nous ne recevons aucune donnée de l'écran, nous ne faisons que lui envoyer des ordres. De fait, aucun souci d'incompatibilité.

Notre écran sera alimenté par du +5V. Le signal RS sera connecté à la broche 8 (GPIO14), enable sur la broche 10 (GPIO15), et D4 à D7 sur les broches 3,5,7 et 11 (GPIO2, GPIO3, GPIO4, GPIO17).

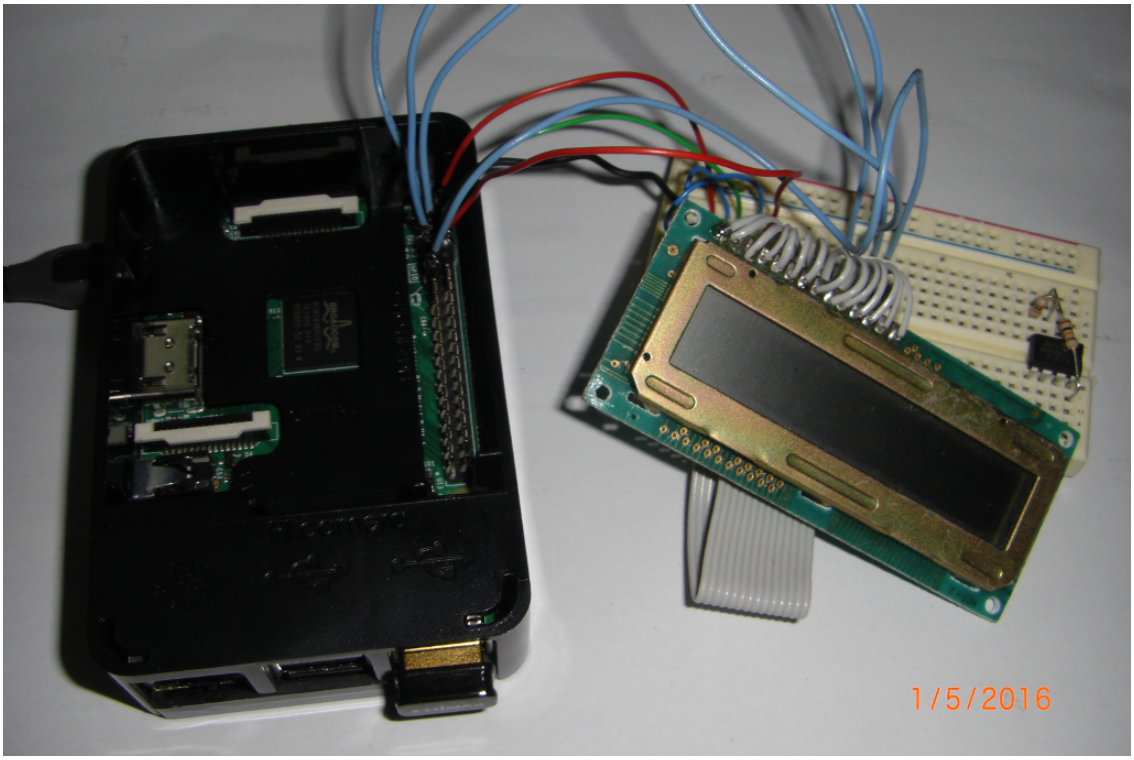
Les afficheurs LCD possèdent une mémoire pour créer des caractères spéciaux. On peut écrire et lire dans cette mémoire. Si au sein de votre projet, vous devez à un moment donné lire cette mémoire, il est indispensable d'adapter les signaux, a minima avec une diode Zener, au mieux avec un optocoupleur.

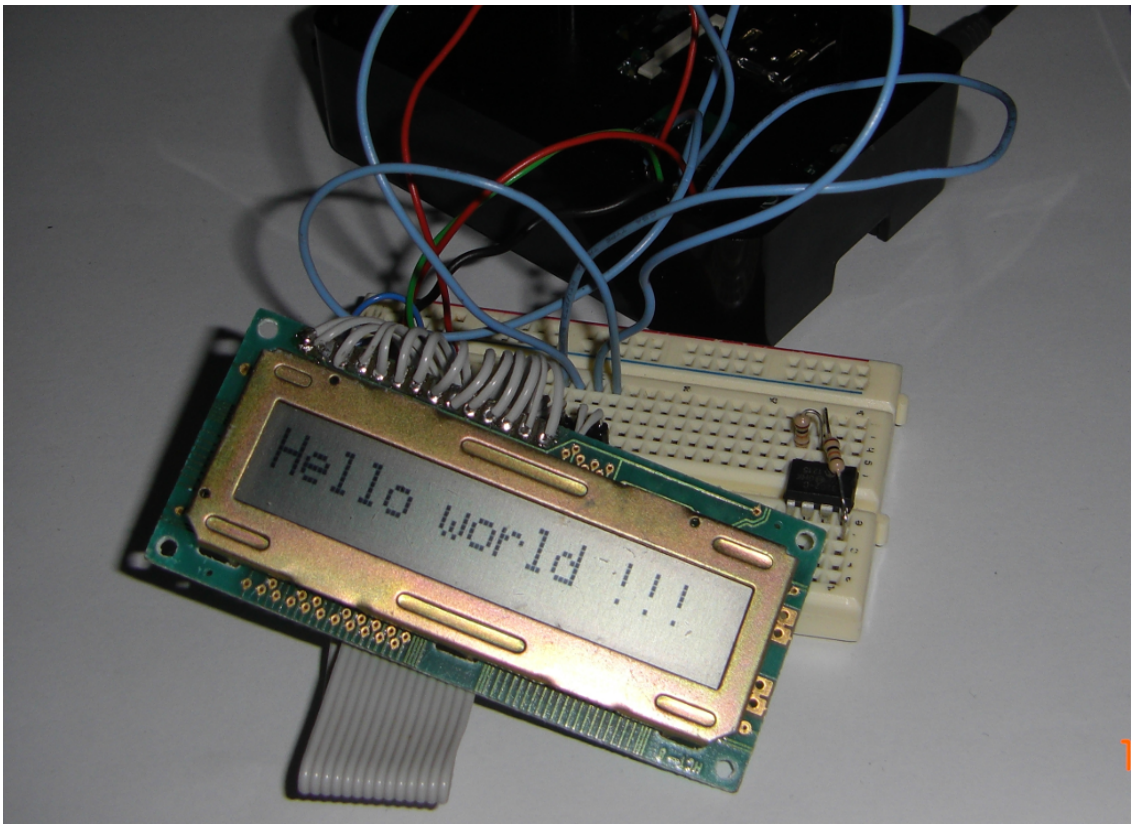


Plutôt que de « réinventer la roue », nous allons utiliser une bibliothèque fournie par Adafruit : **Adafruit_CharLCD.py**

Lcd.py
```
1. import RPi.GPIO
2. import adafruit_charlcd
3.
4. lcd = Adafruit_CharLCD.Adafruit_CharLCD(pin_rs=14, pin_e=15, pins_db=[2,3,4,17])
5. lcd.clear()
```

Lcd.py
```
6. lcd.message("Hello world !!!")
```

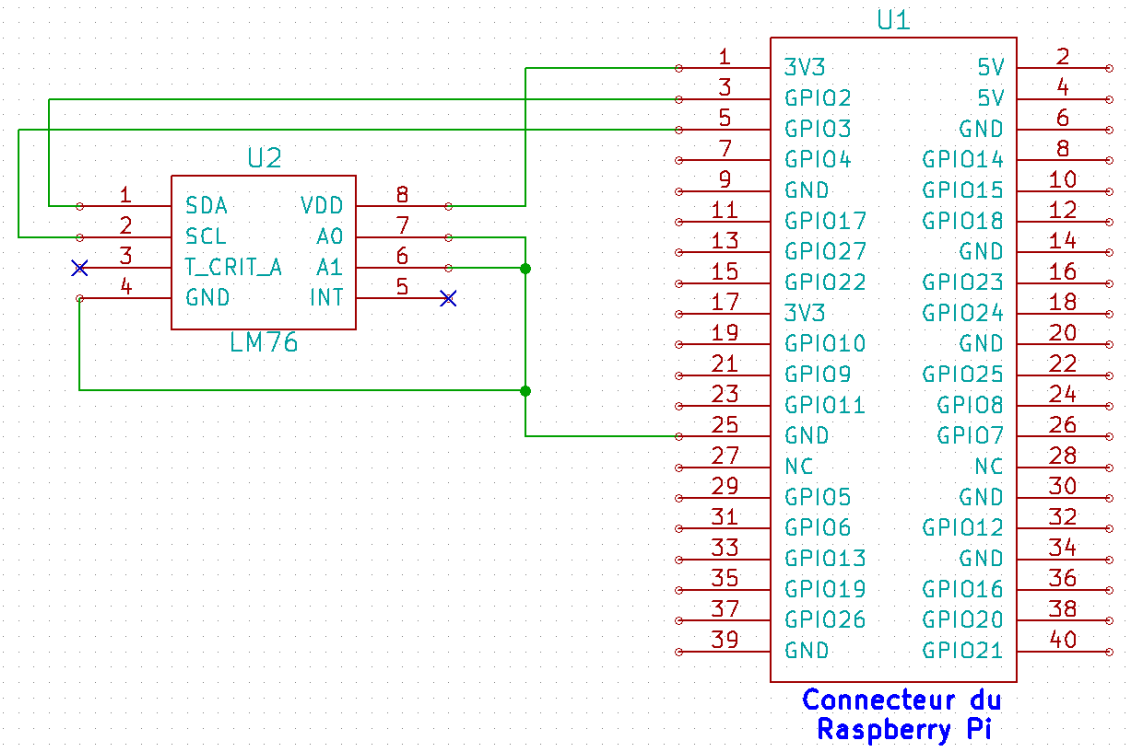




III-C-4 - Lire un capteur de température I2C

Cet exercice n'est pas le mien, mais celui de M. Pascal YON, professeur à l'IUT GEII de l'université de Rennes 1.

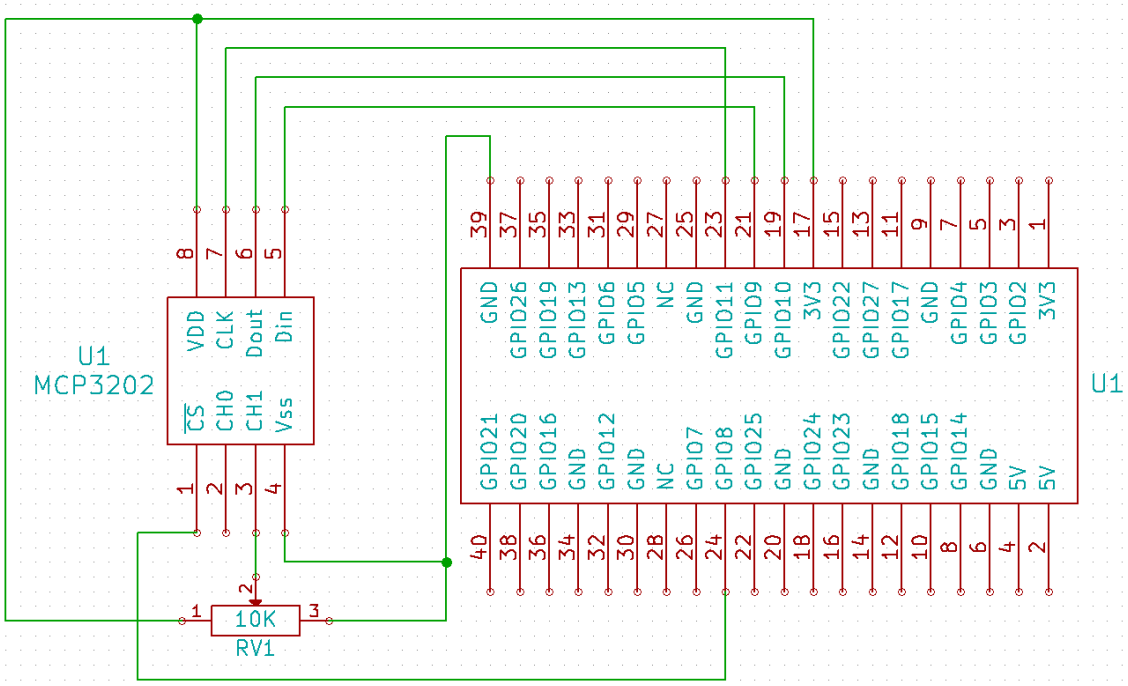
Nous allons ici utiliser un capteur LM76, puis aller lire le contenu d'un registre afin de connaître la température.




```
1. import time
2. from smbus import SMBus
3. bus= SMBus(1)    # 1 indique qu'il faut utiliser le port /dev/i2c-1
4.
5. while True:
6.      # Le composant porte l'adresse 0x48 (A0 et A1 relies à GND)
7.      # On va lire plusieurs octets a partir du registre 0
8.      data = bus.read_i2c_block_data(0x48, 0)
9.
10.      tempMSB = data[0]
11.      tempLSB = data[1]
12.
13.      temperature=(((tempMSB << 8) + tempLSB) >>7) * 0.5
14.      if temperature > 128 :  # test si la temperature est negative
15.          # complement a 1 de la temperature
16.          temperature = (((((tempMSB << 8) + tempLSB) >>7 )* 0.5) -256)
17.
18.      print(temperature)
19.      fichier = open ('fichier_anne_marie','a')
20.      fichier.write (str(temperature))
21.      fichier.write (',  ')
22.      fichier.close ()
23.
24.      time.sleep(2)
```

III-C-5 - Pilotage d'un périphérique SPI

Nous allons ici utiliser un circuit SPI, plus précisément un MCP3202, qui est un convertisseur Analogique/Numérique 12 bits.



Spi.py

```
1. from __future__ import division
2. import spidev
3.
4. def lire_analog(puce_spi = 0, entree_analog = 1):
5.     liaison = spidev.SpiDev(0, puce_spi)
6.     liaison.max_speed_hz = 300000 # en Hertz
7.
8.     # Initialisation des parametres de lecture
9.     # (cf datasheet pour les curieux)
10.     if entree_analog == 0:
11.         value = 128
12.     else:
13.         value = 160
14.     to_send = [value, 0]
15.
16.     # Lecture
17.     rd_octets = liaison.xfer2(to_send)
18.
19.     # La reponse arrive sur deux octets
20.     msb = rd_octets[0]
21.     lsb = rd_octets[1]
22.     value = (msb << 8) + lsb
23.     calcul = 2 * (value * 3.3) / 4096.0
24.     return calcul
25.
26. if __name__ == '__main__':
27.     print(lire_analog())
```

IV - Allons plus loin

Maintenant que vous avez vu comment créer vos programmes et vous interfacer aisément avec l'extérieur, avec des montages externes simples, certains d'entre vous voudraient peut-être aller plus loin, en se créant leurs propres cartes maison.

Si tel était le cas, je vous conseille de vous orienter vers **http://www.kicad-pcb.org/display/KICAD/KiCad+EDA+Software+Suite** pour la conception, et **https://oshpark.com/** pour la fabrication.

KICAD est un logiciel open source de conception électronique, très diffusé maintenant et disposant d'une grande communauté. Il tourne sous Linux, Mac, Windows.

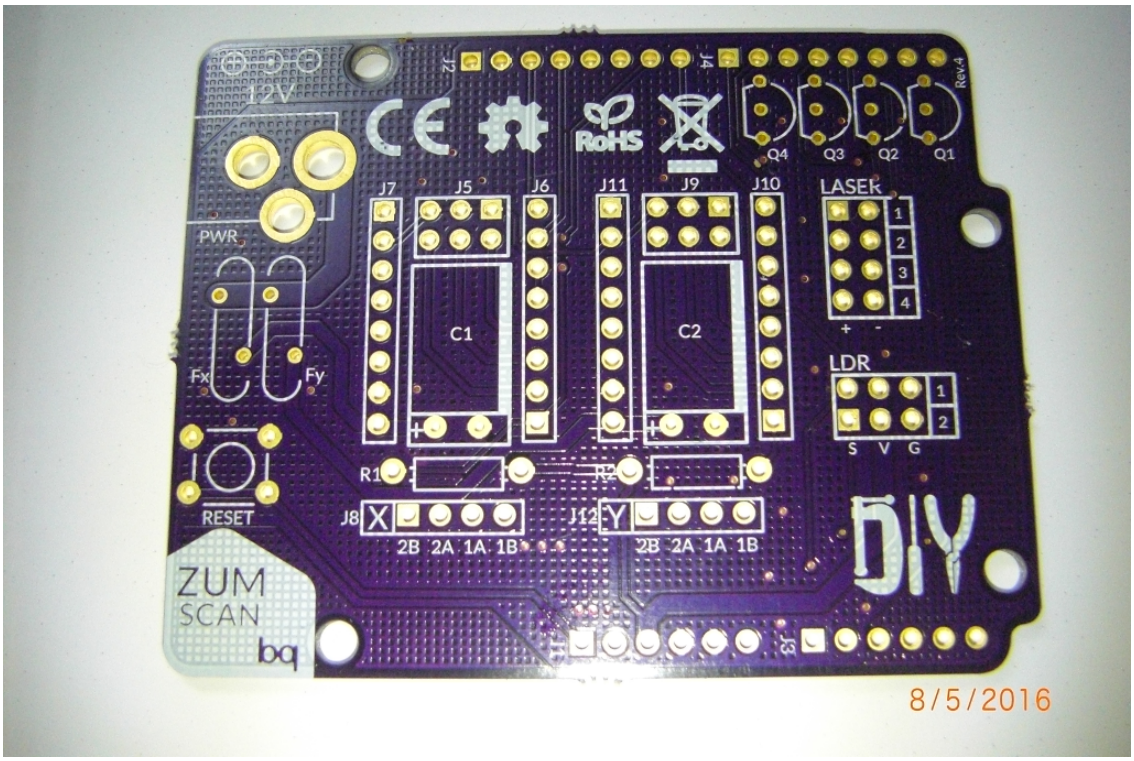
Vous trouverez de nombreux tutoriels, et aides, sur divers sites et forums.

L'intégralité des schémas de ce tutoriel a été effectuée avec KICAD. Notez que nous aurions également pu nous orienter vers un projet type **http://fritzing.org/home/**, lequel permet d'effectuer des schémas plus concrets pour Arduino et Raspberry Pi.

OSH PARK est une association américaine à but non lucratif, ayant un partenariat avec une société de fabrication de circuits imprimés, moyennant un partenariat donnant-donnant : l'association ramène du chiffre d'affaires, et permet d'optimiser les chutes des professionnels, et en contrepartie l'association bénéficie de tarifs défiant toute concurrence.

Ainsi, une carte Arduino (la carte seule) revient à 11 €, le tout en connectique dorée et sérigraphiée.

Les frais de port sont gratuits à travers le monde entier. Les cartes sont vendues par multiples de trois, et le vernis est systématiquement violet.



Une carte au format Arduino de chez OSH Park(Carte du scanner 3D Open SourceBQ Ciclop)

V - Conclusion

Comme nous venons de le voir, le Raspberry Pi est un mini PC dont les possibilités décuplent dès lors que nous prenons en compte son port GPIO.

Liaison concrète entre le monde de l'électronique et celui de l'informatique, le port GPIO est un outil fort pratique, et simple à interfacer grâce à Python.

Il fait ainsi du Raspberry Pi une cible idéale à embarquer et programmer en Python afin de réaliser :

- de la robotique (robot suiveur, bras motorisé...) ;
- de la domotique (station météo, pilotage de volets roulants, centrale d'alarme...) ;

- ou tout autre projet que, il y a encore peu d'années, vous n'auriez pas pensé pouvoir réaliser vous-même.

J'espère que ce tutoriel vous aura permis de mieux l'appréhender et vous aura ouvert de nouvelles perspectives avec votre framboise numérique.

VI - Remerciements

Merci aux personnes suivantes pour leur aide :

- **f-leb**
- **zoom61**
- **ClaudeLELOUP**