Table of Contents

Contrôle d'un GPIO en sortie

Un des grands intérêts du Raspberry pi est qu'il dispose de GPIO utilisables comme entrée ou sortie afin de lire des capteurs ou commandes des systèmes. Nous nous verrons ici comment contrôler les GPIO du Raspberry pi configurés en mode sorte, à travers un exemple concret ou nous commanderons une LED. Il est également possible d'accéder à la liste des tutoriels sur le Raspberry pi pour voir d'autres utilisations.

Prérequis

Pour ce tutoriel, nous partirons d'un Raspberry Pi (A, A+, B, B+, peu importe) déjà configuré, avec Raspbian. Il est possible de faire les mêmes choses avec d'autres systèmes, mais ce tutoriel repose sur Raspbian. Du point de vue matériel, nous aurons besoin du Raspberry pi fonctionnel, d'une breadboard, d'une LED quelconque (dans cet exemple, rouge), d'une résistance adaptée et de quelques câbles pour breadboard, des “jumper wire”. La résistance dépendra de la LED, mais pour une LED rouge classique commandée depuis du 3.3V, il nous faut théoriquement 68Ohms. En pratique, nous prendrons au moins cette valeur, mais 100 ou 200Ohms feront facilement l'affaire. Plus la résistance sera élevée, moins la led éclairera fort. Si la résistance est trop basse, on peut la brûler.

Montage

Il serait possible de simplifier ce montage, en se passant de la breadboard, mais pour simplifier, j'en utilise une ici. Le principe est le suivant : la masse est la broche P1-06, c'est à dire la sixième broche du connecteur P1, en haut à droite. En pratique, c'est la troisième broche de la colonne de droite, en partant du haut. On connecte le + de la LED à la broche GPIO P1-07, choisi arbitrairement parmi ceux disponibles (on peut donc en prendre un autre), qui correspond au GPIO7 du Raspberry Pi. De la même manière, il est possible d'ajouter une seconde LED connectée à un autre GPIO par le +, le - à une autre résistance, et la résistance connectée à la masse. Voici le schéma de montage fait avec Fritzing, un logiciel libre multi-plateformes:

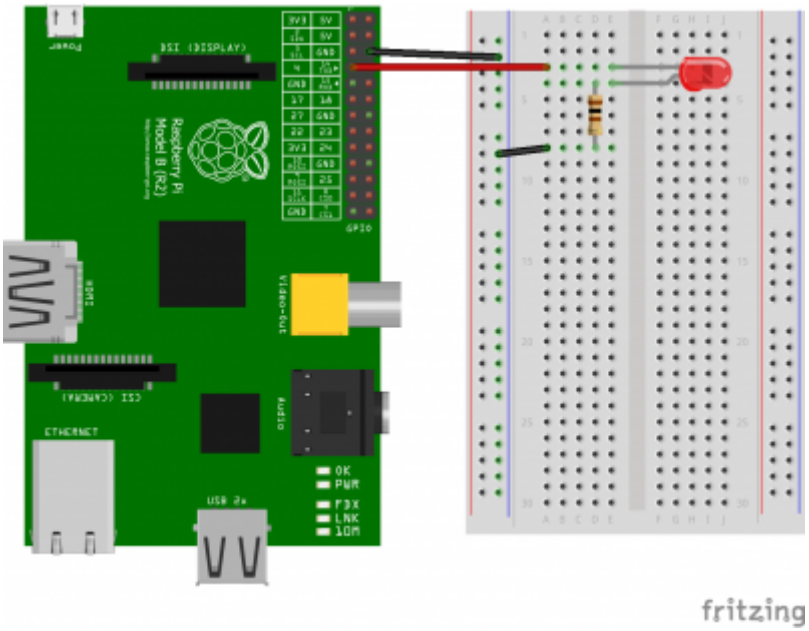

schéma de montage d'une LED contrôlée par le Raspberry pi en PDF

Programmation en C

Maintenant que notre montage est réalisé, nous pouvons programmer la LED. Ici, nous ferons clignoter celle ci, mais il est possible d'adapter le programme comme souhaité. Pour pouvoir faire cela en langage C, nous nous appuierons sur la librairie WiringPi[1)], publiée sous licence GNU LGPLv3, et utilisable pour le C, le C++ et divers autre langages. Cette excellente librairie est pensée pour reproduire les fonctionnalités offertes par l'environnement de programmation Arduino sur le Raspberry pi. Pour l'installer, rien de plus simple, il suffit de se connecter au Raspberry pi en SSH, par exemple (ou bien même directement avec un clavier et un écran), puis de lancer les commandes suivantes :

```
sudo apt-get install git-core
git clone git://git.drogon.net/wiringPi
cd wiringPi
git pull origin
cd wiringPi
./build
```

Une fois la compilation terminée, il est alors possible d'utiliser WiringPi.

Nous pourrons alors creer un fichier blink.c, contenant le code suivant :

blink.c

```
#include <stdio.h>
#include <wiringPi.h>
int main(void)
{
int pin =7;
if(wiringPiSetup()==-1)
{
```

```
  return 0;
}
pinMode(pin,OUTPUT);//on indique que le GPIO7 est en mode sortie
while(1)
{
  digitalWrite(pin,1);//la valeur est définie à HIGH (3.3v)
  delay(500); //on attend 500ms
  digitalWrite(pin,0);//la valeur est définie à LOW (0V)
  delay(500);
}
  return 0;
}
```

Il ne reste plus qu'à compiler, en utilisant GCC[2], avec l'argument -lwiringPi :

```
gcc blink.c -o blink -lwiringPi
```

On peut alors exécuter le programme par la commande suivante :

```
sudo ./blink
```

Le sudo permet de l'exécuter comme si on était root, car il faut des droits root pour accéder aux GPIO, à moins de modifier les droits d'accès de ceux ci. A l'exécution du programme, votre LED devrait se mettre à clignoter. Vous pouvez en ajouter d'autres, en définissant les GPIO selon la table des GPIO du Raspberry Pi par l'auteur de WiringPi[3]. Vous pouvez maintenant changer la vitesse de clignotement, mais aussi faire un programme qui allume la led, et un autre qui l'éteint. Il devient alors possible de lancer ces programmes depuis d'autres selon des conditions précises, par exemple lorsque l'occupation CPU dépasse un certain seuil, dès que le réseau est détecté, dès que le ping est trop élevé, etc...

Code en Python

Il est possible de se passer de WiringPi et de ne pas utiliser le C, ce que je conseillerais à quelqu'un ayant peu d'expérience en programmation. Une autre solution pour cela est d'utiliser le langage python, qui est interprété et donc ne nécessite pas de compilation. Voici l'équivalent du code précédent, qui fait clignoter notre LED, mais en python :

blink.py

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)
while true:
  GPIO.output(7, False)
  time.sleep(0.5)
  GPIO.output(7, True)
```

```
    time.sleep(0.5)
```

Pour exécuter le programme, il suffira de saisir la commande suivante dans un terminal :

```
sudo python blink.py
```

Conclusions

Nous savons maintenant commander un GPIO pour allumer et éteindre une LED via les GPIO. On peut contrôler d'autres dispositifs qu'une LED, mais il convient de prendre en compte certaines limitations. En effet, les GPIO ne sont pas pensés pour alimenter des dispositif puissants. Une ou deux LED ne poseront pas de problème, mais il ne faut pas alimenter un moteur depuis le 3.3V. Si l'on souhaite commander une charge plus importante depuis un GPIO, il faudra utiliser un transistor : commander un dispositif puissant depuis un GPIO via un transistor. On pourra également utiliser un relais pour commander un circuit de grande puissance ou en courant alternatif. Dans tous les cas, il convient de ne pas consommer plus de 50mA sur le 3.3V et l'ensemble des GPIO en sortie.

La prochaine étape à envisager sera d'utiliser le GPIO en lecture, par exemple pour lire la valeur d'un bouton poussoir.

Enfin, si l'on souhaite lire des valeurs analogiques plutôt que numériques comme c'est le cas ici, il nous faudra utiliser un convertisseur analogique-numérique. On pourra par exemple utiliser un MCP3008 pour ajouter 8 entrées analogiques au Raspberry pi et ainsi lire la valeur d'un potentiomètre ou d'un capteur analogique.

1)

page officielle de la librairie wiringpi : http://wiringpi.com/

2)

page wikipedia sur GCC : GNU_Compiler_Collection

3)

table des GPIO dans Wiring Pi, sur le site officiel : https://projects.drogon.net/raspberry-pi/wiringpi/pins/