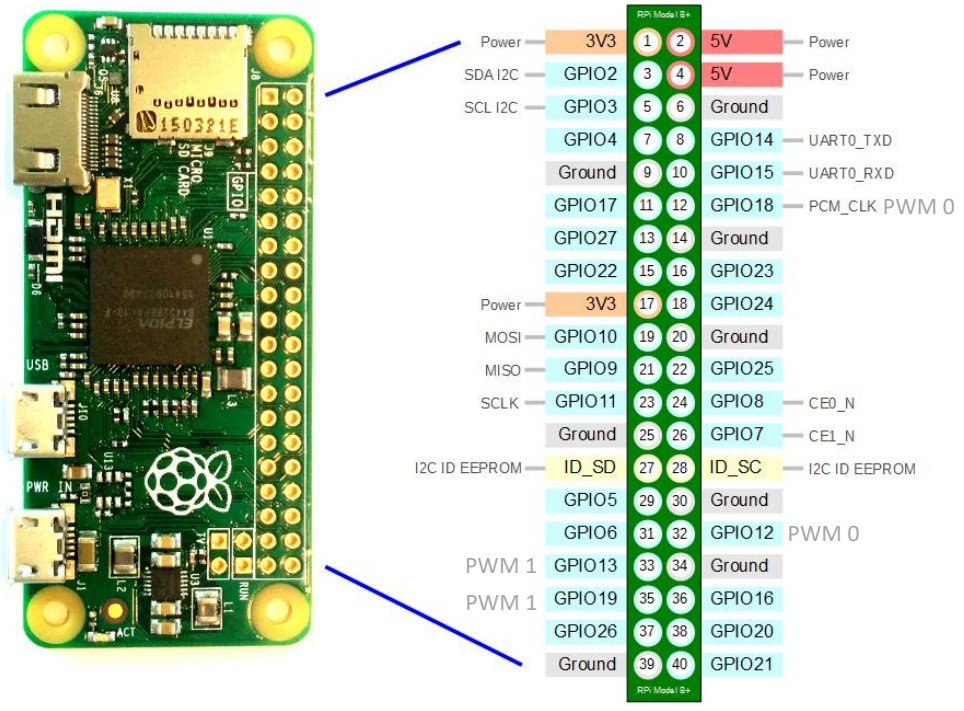
esieabot
Contrôle des GPIO sur Raspberry Pi

Que sont les GPIO ?

Les ports GPIO (General Purpose Input/Output) sont littéralement des ports à usage général d'entrée/sortie. Ce sont des ports généralement sous forme de broche qui sont présents sur des microcontrôleurs et ordinateurs de développement. Comme leur nom l'indique, ils peuvent être utilisés pour de multiples fonctions. Vous en avez déjà probablement manipulé sur Arduino. Vous aviez alors à votre disposition une série de ports pouvant servir d'entrée/sortie logique ou alors d'entrée analogique. Chaque carte de développement a des GPIO différents. Par exemple le Raspberry Pi possède plus de ports qu'un Arduino Uno mais ne possède pas d'entrée analogique.

Les GPIO sur un Raspberry Pi

Votre esieabot est doté d'un ordinateur Raspberry Pi Zero WH. Voici un schéma expliquant la fonction et la numérotation des broches :



Les ports « Power » correspondent à l'alimentation, 3,3V ou 5V. Les ports « Ground » correspondent à la masse. Les ports « GPIO » sont nos fameux GPIO. A noter que certains ports ont des fonctions spéciales (i2c, uart, SDI, etc) que nous n'allons pas utiliser aujourd'hui. A noter également que vous pourrez trouver sur internet des schémas avec une numérotation différente. Chaque numérotation correspond à une bibliothèque de programmation différente. Faites attention à ne pas vous mélanger les pinceaux. Il existe 2 canaux de PWM matériel : PWM 0 et PWM1. Le PWM matériel est géré par un autre composant que le processeur principal et est extrêmement précis. Attention, on ne peut pas utiliser le même canal (les GPIO 13 et 19 par exemple) pour envoyer 2 signaux différents.

Utiliser les GPIO en C

Comme évoqué précédemment, il existe une multitude de bibliothèques de programmation permettant de piloter les GPIO du Raspberry Pi. Nous allons travailler aujourd'hui avec la bibliothèque pigpio. Elle permet de contrôler le Raspberry Pi à travers des programmes écrits en C, en Python, ou même depuis un navigateur internet à travers le protocole réseau HTTP. Nous allons bien-sûr travailler aujourd'hui en C.

La syntaxe et la logique sont très similaires à celles de la bibliothèque Arduino. Il faut tout d'abord activer un GPIO, en choisissant de l'activer en mode INPUT ou en mode OUTPUT. Il est ensuite possible d'y lire ou d'y écriture des valeurs logiques (HIGH ou LOW).

On définit les pins utilisés

On initialise pigpio

On active en mode sortie les pins

On envoie un signal sur les pins

On attend 5s

On coupe le signal sur les pins

On désactive pigpio

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pigpio.h>

#define PIN_ENABLE_LEFT 24
#define PIN_FORWARD_LEFT 23

int main() {
        if (gpioInitialise() < 0) {
                // pigpio initialisation failed.
                printf("Can't initialize pigpio\n");
                return -1;
        } else {
                // pigpio [redacted] okay.
                gpioSetMode(PIN_ENABLE_LEFT, PI_OUTPUT);
                gpioSetMode(PIN_FORWARD_LEFT, PI_OUTPUT);
                gpioWrite(PIN_ENABLE_LEFT, PI_HIGH);
                gpioWrite(PIN_FORWARD_LEFT, PI_HIGH);
                sleep(5);
                gpioWrite(PIN_FORWARD_LEFT, PI_LOW);
                gpioWrite(PIN_ENABLE_LEFT, PI_LOW);
                gpioTerminate();
                return 0;
        }
}
```

Voici un exemple très basique d'un code en C qui permet d'activer le moteur gauche et de le faire tourner pendant 5 secondes. Les numéros des broches 24 et 23 correspondent au schéma précédent. Pour faire le lien entre la numérotation des GPIO et les commandes faites aux moteurs, vous devez vous référer à la fiche technique de votre pont en H.

Vous remarquerez que la syntaxe générale est comparable à celle d'un programme pour Arduino :

- pinMode() devient gpioSetMode()
- digitalWrite() devient gpioWrite()
- digitalRead() devient gpioRead()
- HIGH et LOW deviennent PI_HIGH et PI_LOW
- etc.

Toutes les syntaxes sont disponibles sur cette page de documentation : https://abyz.me.uk/rpi/pigpio/cif.html.

Pour compiler du code avec la bibliothèque pigpio, il faut naturellement rajouter à gcc l'option « lpigpio ». Par exemple : « `gcc -Wall test.c -o test -lpigpio` ». On pourra ensuite lancer l'exécutable « `sudo ./test` ». A noter que le « sudo » est nécessaire, en effet il faut les droits super-utilisateur pour pouvoir piloter les GPIO.

Vous remarquerez qu'à la fin de l'exécution une fonction « `gpioTerminate()` » est appelée. Cette fonction permet de libérer les GPIO utilisés. En effet, comme le Raspberry Pi est un ordinateur, d'autres processus pourraient avoir besoin par la suite d'utiliser les GPIO sans devoir redémarrer le système. C'est pour cela qu'il faut toujours appeler cette fonction à la fin de votre programme. Si votre programme plante avant l'exécution de la fonction « `gpioTerminate()` », vous pourriez avoir besoin de redémarrer votre esieabot.

Pour l'utilisation de périphériques évolués tels que les servomoteurs, des fonctions existent déjà dans la bibliothèque pigpio. Vous les trouverez sur la documentation officielle.