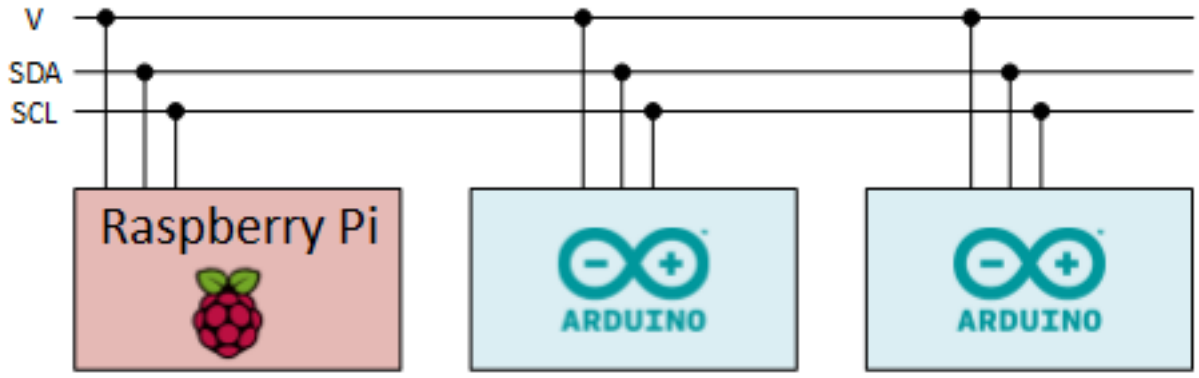
1 | Communication I2C

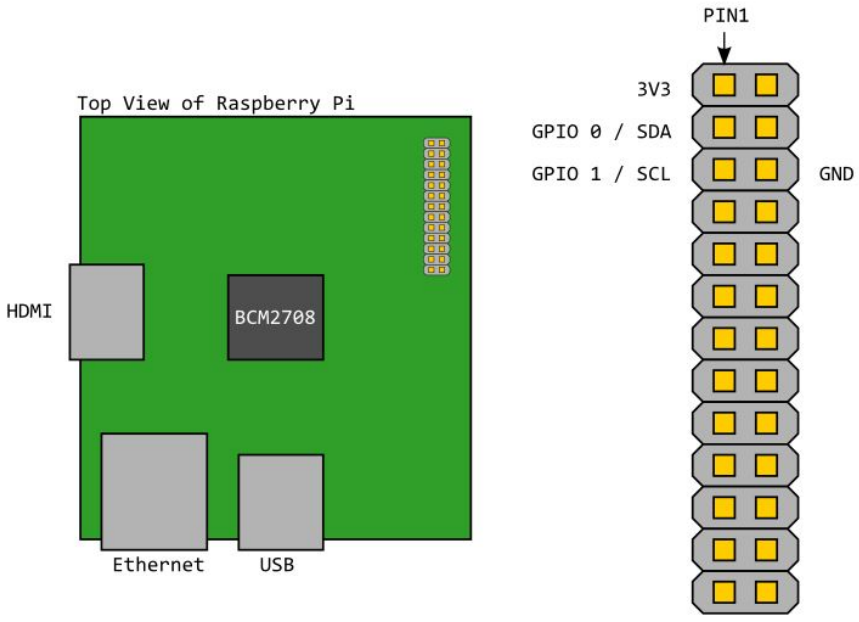
Le protocole I2C va nous permettre de communiquer entre les différents modules du robots. Dans le protocole I2C il y à un *maitre* et un ou plusieurs *esclaves*. Il est important de noter que dans le protocole I2C, c'est uniquement le maitre qui initie une communication avec un esclave.



Dans notre cas, la Raspberry Pi sera le maitre et les arduinos seront les esclaves.

1.1 Configuration Raspberry PI

Sur la Raspberry PI, la pin **SDA** se trouve sur la pin 3 et la pin **SCL** se trouve sur la pin 5.



Pour tester que les arduinos soient bien connectés, on peut utiliser la commande *i2cdetect -y 1* qui va afficher les adresses

des esclaves connectés.

```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- 04 -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

On peut voir qu'il y à un esclave connecté à l'adresse *0x04*.

Pour employer la connexion I2C depuis le code Python, on utilise la librairie **smbus**. Celle-ci nous permet de lire et écrire des *bytes* sur le bus I2C. Ci-dessous, vous pouvez voir un exemple d'un code Python qui envoie un nombre au hasard entre 0 et 180 sur le bus I2C à un esclave connecté à l'adresse *0x04* toute les 2 secondes. L'arduino qui reçoit ces données interprète ces nombres en angles pour commander un servo moteur.

```python
import smbus
import time
import random

bus = smbus.SMBus(1)
adr_arduino1 = 0x4
adr_arduino2 = 0x5
time.sleep(0.1)

while True:
    r = random.randrange(0, 180)
    bus.write_byte(adr_arduino1, r)
    print("Angle: ", r)
    time.sleep(2)
```

1.2 Configuration Arduino

On utilise la librairie *Wire.h* pour utiliser la communication I2C au niveau de l'arduino. Pour communiquer avec la raspberry, il faut correspondre l'adresse avec celle présente dans le code python de la raspberry. Ensuite dans la fonction *setup*, configurer la librairie *Wire.h* avec les fonctions *begin(address)*, *onReceive(fonction de réception)* et *onRequest(fonction d'envoi)*.

Dans un 2ème temps, il faut configurer les fonctions de réception et d'envoi. Pour recevoir une données par I2C avec la librairie *Wire.h*, il suffit d'utiliser la fonction *read()*. Pour envoyer une donnée, il suffit d'utiliser la fonction *write(donnée)*.

Dans l'exemple ci-dessous nous utilisons un servo-moteur avec les données envoyées par la raspberry (un angle).

```arduino
#include <Wire.h>
#include <Servo.h>

#define SLAVE_ADDRESS 0x12
int dataReceived = 0;
Servo monservo;
```

```
void setup() {
    Serial.begin(9600);
    Wire.begin(SLAVE_ADDRESS);
    Wire.onReceive(receiveData);
    Wire.onRequest(sendData);
}

void loop() {
    delay(100);
}

void receiveData(int byteCount){
    while(Wire.available()) {
        dataReceived = Wire.read();
        Serial.print("Donnee recue : ");
        monservo.write(dataReceived);
        Serial.println(dataReceived);
    }
}

void sendData(){
    int envoi = dataReceived + 1;
    Wire.write(envoi);
}
```

1.3 Structurer le code

Dans le code de la Raspberry PI, qui gère le tout, on peut construire une classe par module qui crée une abstraction au dessus des communications I2C. Pour l'exemple du servo moteur, on pourrait avoir une classe `class Servo` qui fournit une méthode `def set_angle(angle)`. Cette méthode s'occuperait de la communication I2C avec l'arduino, ce qui produit une belle abstraction.

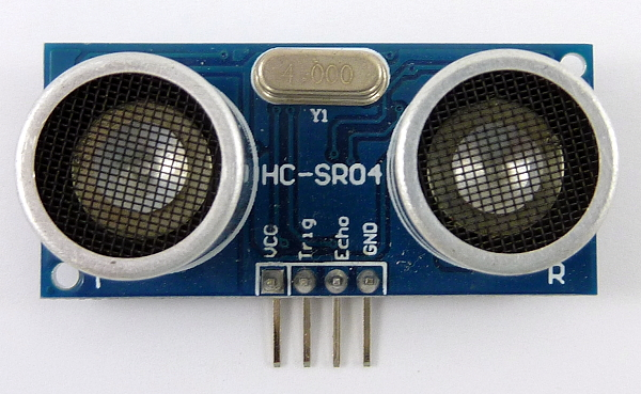
1.4 Sources externes

I2C communication between Raspberry and Arduino

Smbus(documentation library python to communicate i2c) :
- Functions details
- Resume functions

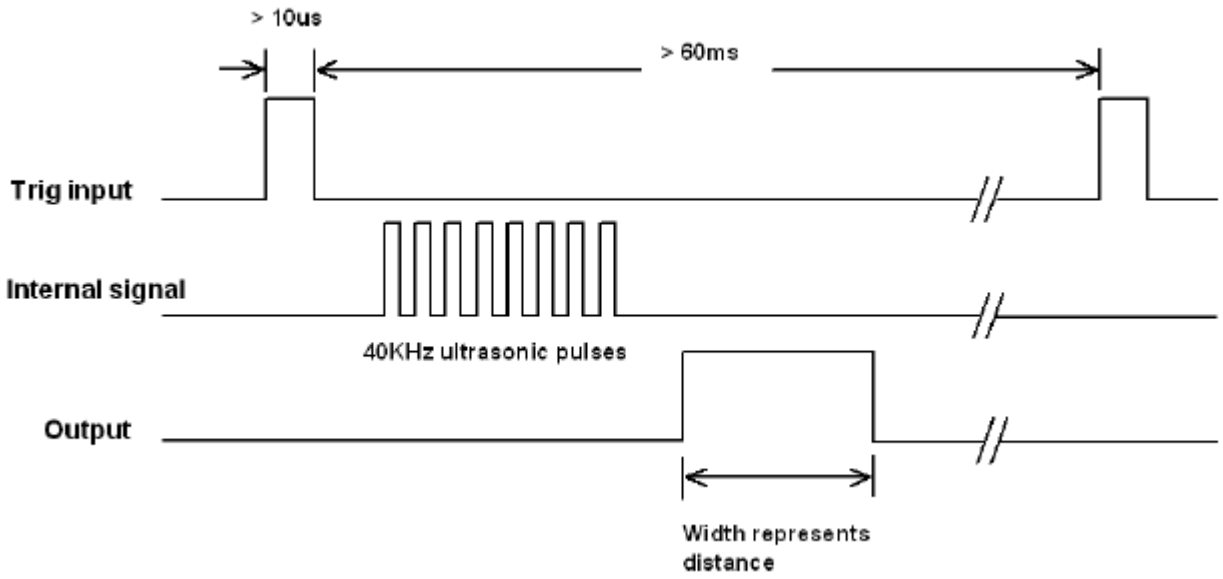
2 | Capteurs

2.1 Ultrason



Le principe de fonctionnement de ce capteur est le suivant :

Une impulsion est envoyée sur la broche du trigger(sortie Arduino). Ce dernier génère un signal de plusieurs kHZ pendant environ 10μs. La pin Echo (entrée analogique Arduino) récupère le signal envoyé et renvoie un signal proportionnel à la distance parcourue par le signal provenant de Trig.



Exemple de code :

```
/* Utilisation du capteur Ultrason HC-SR04 */

// définition des broches utilisées
int trig = 12;
int echo = 11;
```

```
long lecture_echo;
long cm;

void setup()
{
  pinMode(trig, OUTPUT);
  digitalWrite(trig, LOW);
  pinMode(echo, INPUT);
  Serial.begin(9600);
}

void loop()
{
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  lecture_echo = pulseIn(echo, HIGH);
  cm = lecture_echo / 58;
  Serial.print("Distance en cm : ");
  Serial.println(cm);
  delay(1000);

}
```

Note : Pour avoir la distance en cm, il faut diviser la valeur reçue par 58 (voir datasheet HC-SR04).