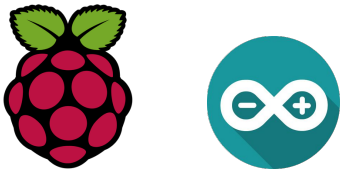
CHARLES CARNUS
LYCÉE TECHNIQUE
ENSEIGNEMENT SUPÉRIEUR



Informatique embarquée

Raspberry – Arduino

Prof : **Kamal Boudjelaba**

2 octobre 2023

B T S S N

LTP Charles Carnus
Avenue de Saint Pierre - Bourran - 12000 RODEZ, France

IR–EC

Table des matières

1. Installer l'IDE Arduino pour Raspberry Pi (Linux ARM)

1. Télécharger l'IDE Arduino pour processeur ARM à partir de ce lien.
2. Ouvrir un Terminal
   - Aller dans le répertoire Download `$ cd ~/Download`
   - Déplacer le fichier compressé à la racine ou au dossier Documents (`$ mv nomFichier /home` ou `$ mv example.txt  /Documents`). Ensuite, décompresser l'archive avec la commande `tar xvf`
     `$ tar xvf arduino-1.8.16-linuxaarch64.tar.xz`
   - Lancer le script d'installation
     ```
     $ cd arduino-1.8.16
     $ ./install.sh
     ```

   Ou :
   ```
    $ cd Downloads/
   $ tar -xf arduino-1.8.16-linuxaarch64.tar.xz
   $ sudo mv arduino-1.8.16 /opt
   $ sudo /opt/arduino-1.8.16/install.sh
   ```

Si tout s'est bien passé, un nouveau raccourci est ajouté vers l'IDE Arduino dans le menu Développement.

Exercice :

- Écrire un programme simple pour vérifier le bon fonctionnement de l'IDE Arduino.
- Après avoir branché la carte Arduino au port USB de la carte Raspberry, ne pas oublier de choisir le nom du port usb occupé par l'Arduino. Dans le terminal, taper :
  `$ ls /dev/tty*`

Normalement, le nom doit ressembler à **ttyUSB0**

2. Communication entre une carte Raspberry Pi et une carte Arduino

Code Arduino

Brancher la carte Arduino à un ordinateur.
Le code à téléverser sur l'Arduino pour tester la connexion permet d'écrire une lettre dans la console série de l'Arduino depuis le Raspberry Pi. L'Arduino va écouter ce port série, et dès que quelque chose y est écrite, on la recopie.

Programme Arduino — BTS SN : KB

```
1
2  void setup()
3  {
4    Serial.begin(9600);
5  }
6  void loop()
7  {
8    while (Serial.available())  {
9      message = Serial.read()-'0';  // on soustrait le caractère 0, qui vaut 48 en ASCII
10     Serial.println(message);
11   }
12   delay(10);
13 }
```

Brancher la carte Arduino au port USB de la carte Raspberry.
Ouvrir le moniteur série.

Code Python (carte Raspberry)

- Installer pyserial pour communiquer avec le port USB en utilisant un script Python (s'il n'est pas installé)
  `$ sudo apt-get install python-serial`
- Identifier le port USB utilisé (pour être certain de communiquer avec le bon port)
  `$ ls /dev/tty*`
- Code Python :
  - `$ cd /home/pi`
  - `$ nano test.py`
  - Copier/coller le code ci-dessous
  - Ctrl+X puis Y pour sauvegarder
  - Puis lancer la commande `$ python test.py`

Programme Python — BTS SN : KB

```
1 import sys      # pour la gestion des paramètres
2 import serial   # bibliothèque permettant la communication série
3 #On écrit dans l'interface /dev/ttyUSB0 à 9600 bauds
4 ser = serial.Serial('/dev/ttyUSB0', 9600)
5 #On écrit dans la console série de l'Arduino la lettre passée en paramètre lors
6 #de l'execution de la commande :
7 ser.write(sys.argv[1])
```

2.1 Communication unidirectionnelle de l'Arduino vers le Raspberry Pi

Côté Arduino, c'est très simple : il suffit juste de lui demander d'écrire sur la sortie série Serial le message que l'on souhaite transmettre. Le code ci-dessous envoie un message toutes les deux secondes de type : Message i, "i" s'incrémentant à chaque envoi.
Ne pas brancher encore l'Arduino au Raspberry Pi.

Programme Arduino — BTS SN : KB

```
1
2  int i=0;
3
4  void setup() {
5    Serial.begin(9600);
6    Serial.println("Un message va être envoyé toutes les 2 secondes dès maintenant !");
7  }
8
9  void loop() {
10   Serial.println("Message #" + String(i));
11   delay(2000);
12   i++;
13 }
```

Côté Raspberry, on va écrire un programme en Python qui va utiliser la fonction serial.Serial. Cependant, il faut vérifier le port utilisé par l'Arduino sur le Raspberry Pi. Il suffit d'entrer la commande suivante :

```
$ ls /dev/tty*
```

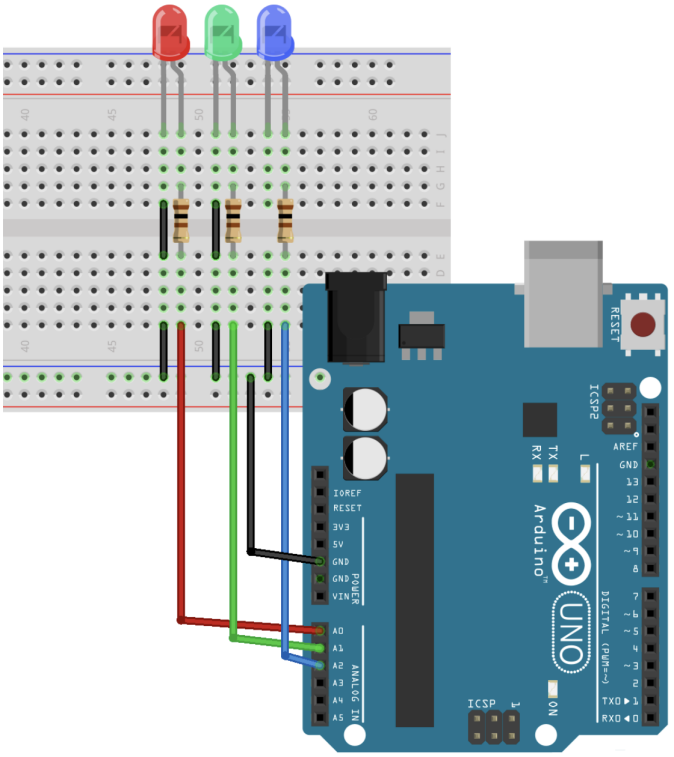
Le programme qui sera exécuté sur le Raspberry :

Programme Python — BTS SN : KB

```
1  # Import de la librairie serial
2  import serial
3
4  # Ouverture du port série avec :
5  # '/dev/ttyXXXX' : definition du port d'écoute (remplacer 'X' par le bon nom)
6  # 9600 : vitesse de communication
7  serialArduino = serial.Serial('/dev/ttyXXXX', 9600)
8
9  # Ecriture de chaque message reçu
10 while True :
11     print(serialArduino.readline())
```

2.2 Communication unidirectionnelle du Raspberry Pi vers l'Arduino

Pour cette communication, on va utiliser un exemple plus concret : Contrôler des LEDs connectées à l'Arduino à partir du Raspberry. On commence par brancher les LEDs à l'Arduino, en suivant ce schéma (par exemple) :



Programme Arduino qui va recevoir les ordres du Raspberry Pi :

Programme Arduino — BTS SN : KB

```
1
2  #define Rouge A0
3  #define Vert A1
4  #define Bleu A2
5
6  int message = 0;
7  int couleurs[] = {Rouge, Vert, Bleu};
8
9  void setup() {
10   Serial.begin(9600);
11   pinMode(Rouge, OUTPUT);
12   pinMode(Vert, OUTPUT);
13   pinMode(Bleu, OUTPUT);
14 }
15
16 void loop() {
17   if (Serial.available()) {
18     message = Serial.read() - '0';
19     if (message > 3) {
20       digitalWrite(couleurs[message-4], LOW);
21     }
22     else {
23       digitalWrite(couleurs[message-1], HIGH);
24     }
25   }
26 }
```

Code d'envoi des ordres par le Raspberry Pi :

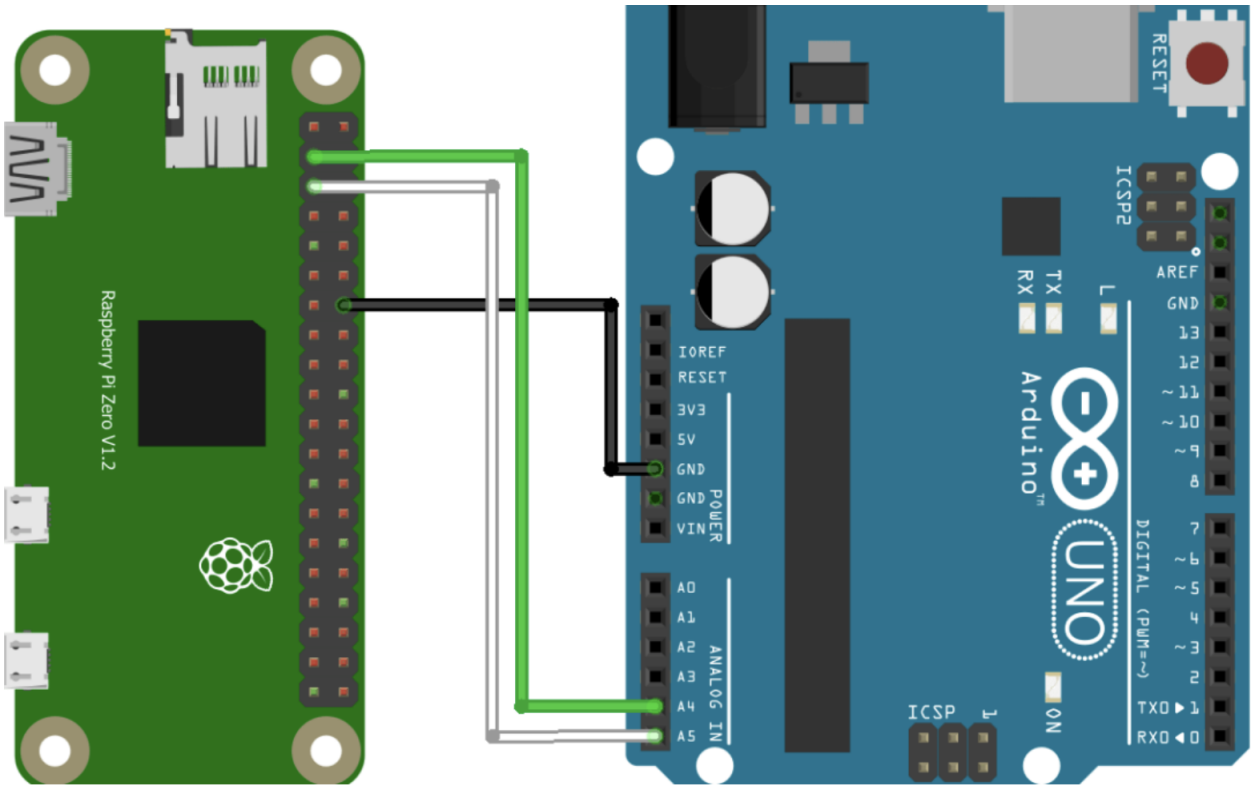
Programme Python — BTS SN : KB

```python
import serial

ser = serial.Serial('/dev/ttyXXXX', 9600)
print("CTRL + C pour arrêter")

while True :
  led = int(input('Quelle LED souhaitez-vous allumer / eteindre ? (Rouge=1 / Vert=2 / Bleu
    =3)'))
  action = input("Souhaitez-vous allumer ou Eteindre la LED ? (Allumer=1 / Eteindre=0)")
  if (action == 1) :
    ser.write(str(led))
  else :
    msg = led+3
    ser.write(str(msg)
```

3. Communication I2C entre Raspberry Pi et Arduino



Pour établir la communication I2C entre Raspberry Pi et Arduino, il faut relier physiquement le bus qui utilise 3 broches. Une communication I2C est défini par un bus de deux fils (parfois appelé TWI, Two Wire Interface) et une adresse. Les broches utilisées par la communication I2C sont généralement fixé pour chaque appareil. L'une sur laquelle sont envoyées les données (SDA Serial Data Line) et sur l'autre l'horloge de synchronisation (SCL Serial Clock Line). Les masses des deux cartes doivent être reliées pour établir une référence commune de potentiel.

- SDA BCM2(RPI) $\longleftrightarrow$ SDA A4(Arduino)
- SCL BCM3(RPI) $\longleftrightarrow$ SCL A5(Arduino)
- GND (RPI) $\longleftrightarrow$ GND(Arduino)

Configuration du Raspberry PI

Pour utiliser l'interface I2C du Raspberry Pi, celle-ci doit être activée dans le menu de configuration. Pour cela, entrer la commande suivante dans un terminal :

```
$ sudo raspi-config
```

Dans le Menu, sélectionner 5 – Interfacing Options puis P5 I2C et valider.

Un fois le branchement fait, on peut vérifier les appareils branchés sur le bus en tapant dans le terminal la commande :

```
$ i2cdetect -y 1
```

Le Raspberry Pi retourne la liste des adresses détectées sur le bus.
Installer la librairie **smbus2** qui permet de gérer la communication I2C côté Raspberry Pi (Python).

```
$ pip3 install smbus2
```

Programme Python (Maître) BTS SN : KB

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#i2cdetect -y 1

import sys
import smbus2 as smbus#,smbus2
import time

# Adresses "Esclave"
I2C_SLAVE_ADDRESS = 11 #0x0b ou 11
I2C_SLAVE2_ADDRESS = 12
I2C_SLAVE3_ADDRESS = 13

# Fonction de conversion de caractères en octets.
def ConvertStringsToBytes(src):
  converted = []
  for b in src:
    converted.append(ord(b))
  return converted

def main(args):
    # Crée le bus I2C
    I2Cbus = smbus.SMBus(1)
    with smbus.SMBus(1) as I2Cbus:
        slaveSelect = input("Which Arduino (1-3): ")
        cmd = input("Entrer la commande : ")

        if slaveSelect == "1":
            slaveAddress = I2C_SLAVE_ADDRESS
        elif slaveSelect == "2":
            slaveAddress = I2C_SLAVE2_ADDRESS
        elif slaveSelect == "3":
            slaveAddress = I2C_SLAVE3_ADDRESS
        else:
            # Arrêt s'il y a une erreur
            print(slaveSelect== "1")
            print(type(slaveSelect))
            print("no slave selected")
            quit()
        BytesToSend = ConvertStringsToBytes(cmd)
        print("Envoi " + str(slaveAddress) + " le " + str(cmd) + " commande.")
        print(BytesToSend )
        I2Cbus.write_i2c_block_data(slaveAddress, 0x00, BytesToSend)
        time.sleep(0.5)

        while True:
            try:
                data=I2Cbus.read_i2c_block_data(slaveAddress,0x00,16)
                print("rreçu de l'esclave:")
                print(data)
            except:
                print("Erreur d'E/S distante ")
                time.sleep(0.5)
    return 0

if __name__ == '__main__':
    try:
        main(sys.argv)
    except KeyboardInterrupt:
        print("programme arrêté manuellement")
    input()
```

Le Raspberry Pi envoie la commande "slave" à l'Arduino, puis réceptionne un tableau de données provenant de l'Arduino.

Programme Arduino (Esclave) BTS SN : KB

```
#include <Wire.h>
# define I2C_SLAVE_ADDRESS 11 // 12 pour l'esclave 2 et ainsi de suite
#define PAYLOAD_SIZE 2
void setup()
{
Wire.begin(I2C_SLAVE_ADDRESS);
Serial.begin(9600);
Serial.println("----------------------------------------Esclave 1");
delay(1000);
Wire.onRequest(requestEvents);
Wire.onReceive(receiveEvents);
}
void loop(){}
int n = 0;
void requestEvents()
{
Serial.println(F("---> demande reçue"));
Serial.print(F("Envoi de valeur : "));
Serial.println(n);
Wire.write(n);
}
void receiveEvents(int numBytes)
{
Serial.println(F("---> réception d'évènements"));
n = Wire.read();
Serial.print(numBytes);
Serial.println(F("octets reçus "));
Serial.print(F("valeur reçue : "));
Serial.println(n);
}
```

L'Arduino réceptionne la commande "slave" puis envoie deux valeurs mises à jour dès qu'il reçoit une requête du Raspberry Pi.