Communication Arduino-Python via le protocole de communication "Firmata"

Nous avons vu qu'avec la liaison série, il était possible d'interagir avec un Arduino à l'aide d'un programme en Python. Cependant, en fonction des circuits électroniques, des capteurs utilisés et des actions que l'on souhaite réaliser, il faut téléverser dans la mémoire de l'Arduino, un programme adapté et bien souvent modifier le programme en Python associé pour la réception ou l'envoi des données.

Pour pallier à ce problème, il existe un protocole de communication, appelé, **"Firmata"**, et basé sur deux programmes :

- un programme en Python "donneur d'ordres", sur l'ordinateur, pour pouvoir envoyer des ordres à l'Arduino ou recevoir des données via le port USB.
- un programme "pilote", sur le microcontrôleur, qui comme son nom l'indique, pilotera les matériels connectés en réponse aux ordres reçus.

Pour que le programme Python "donneur d'ordres" fonctionne, le chargement, dans la mémoire de l'Arduino, du code "pilote" doit être fait avant son lancement, à l'aide du logiciel "**IDE ARDUINO**".
Mais une fois le téléversement du programme "pilote" effectué, seul le programme en Python sera modifié en fonction des actions que l'on souhaite réaliser. En effet, avec le protocole de communication **"Firmata"**, la lecture et l'écriture sur les entrées et sorties de l'Arduino se font directement à partir du programme en Python, ce qui facilite l'exploitation des données des capteurs.

Suivant les circuits étudiés ou les capteurs utilisés, on utilisera deux protocoles de communication avec l'Arduino différents, "**Firmata Standard**" ou "**Firmata Express** ".

1. le protocole de communication "Firmata Standard"

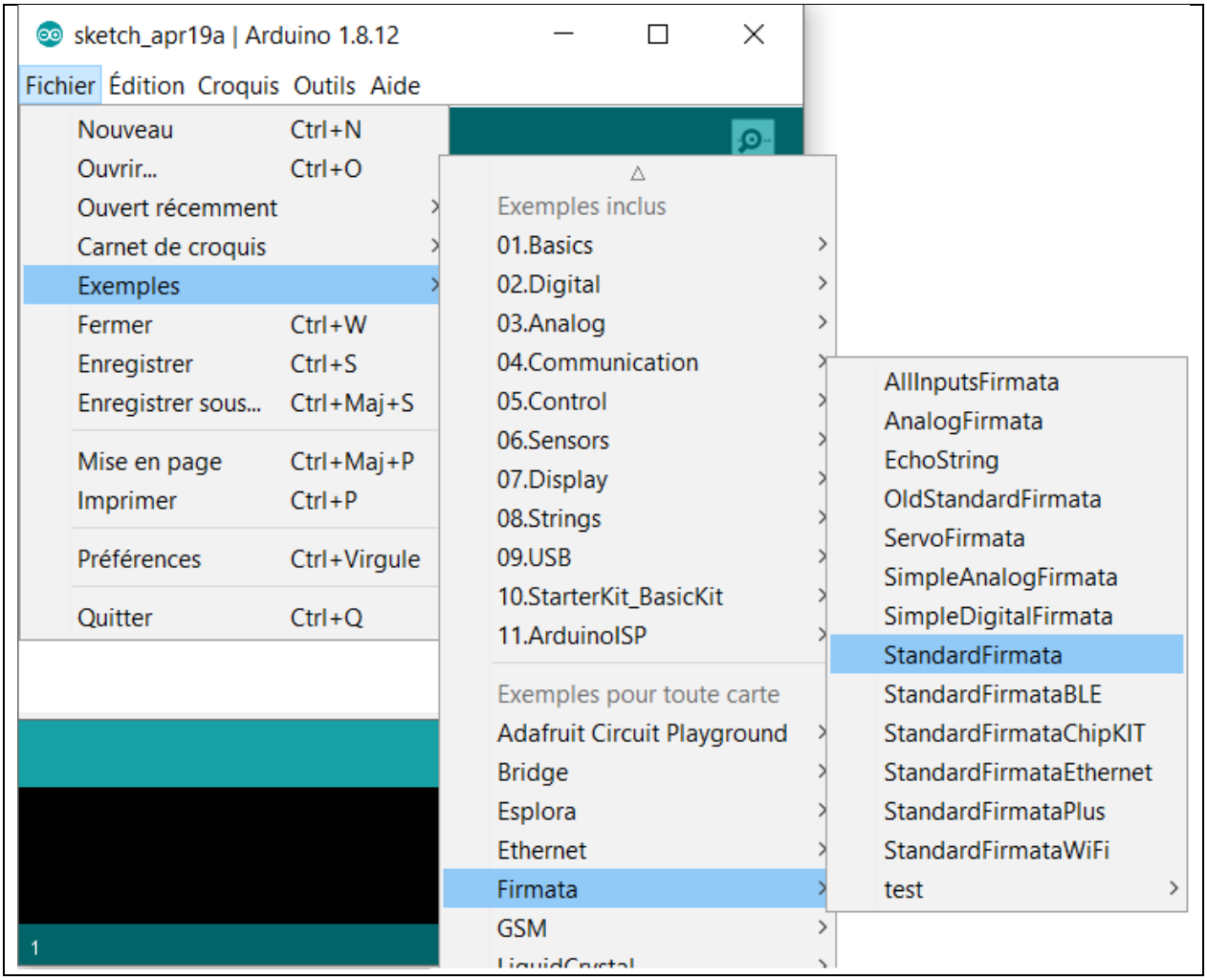
. Chargement du code "Firmata Standard" dans la mémoire de l'Arduino :

– Brancher l'Arduino via un port USB,

- Afin de charger la librairie "**Firmata standard**" sur l'ARDUINO, il faut lancer le logiciel "**IDE ARDUINO**", puis sélectionner :

Fichier > Exemples > Firmata > Standard Firmata,

- puis cliquer sur "**téléverser**".



. <u>Installation de la bibliothèque PyFirmata dans Python:</u>

Pour faire fonctionner, un programme en Python qui contrôle l'Arduino via le protocole de communication "**Firmata standard**", Python doit disposer de la bibliothèque "**PyFirmata**". Celle-ci peut être installée via **"pip"**, à l'aide de la ligne de commande :

pip install pyfirmata

Pour utiliser la bibliothèque **"pyfirmata"** dans un programme python, il faut importer le module "**pyfirmata**", à l'aide de l'instruction : **import pyfirmata**

La connexion avec le microcontrôleur, via le port série, est réalisée avec la méthode **"Arduino"** du module **"pyfirmata"** en précisant le port COM sur lequel l'Arduino est connecté :

board = pyfirmata.Arduino(Port COM)

Une fois la connexion établie, il est possible d'interroger ou de modifier les entrées et sorties numériques ou analogiques de l'Arduino.

1.1 Gestion des sorties numériques

Pour modifier l'état d'une sortie numérique (l'équivalent d'un "**digitalWrite**" en langage Arduino), la syntaxe est la suivante :

board.digital[numéro_de_pin].write(0_ou_1)

où :

- **"board"** est l'objet créé lors de l'appel de la méthode "**Arduino**" du module "**pyfirmata**",
- "**numéro_de_pin**" est le numéro de la sortie du microcontrôleur dont on veut modifier l'état,
- "**0_ou_1**" est le niveau logique souhaité.

Le plus simple est de créer une fonction qui sera appelée dans nos programmes en Python pour modifier l'état d'une sortie numérique :

```
def DigitalWrite(board,pin,val):
  board.digital[pin].write(val)
```

Ainsi, l'instruction pour mettre la sortie numérique "**9**", de l'objet "**board**", à l'état haut est :

DigitalWrite(board, 9, 1)

Remarque :

Contrairement à un programme Arduino, avec Pyfirmata, il n'est pas nécessaire de déclarer au préalable une broche en sortie numérique avant de l'utiliser.

Exemple :

Avec notre circuit support des exemples, nous allons reprendre le programme Arduino qui demande à l'utilisateur de saisir une instruction :

- si l'instruction est "ON", la DEL rouge s'allume,
- et si c'est "OFF ", la DEL rouge s'éteint.

Et l'adapter en langage Python en utilisant la bibliothèque "**pyfirmata**" (programme "**LedRVBDigitalOutput.py**" dans le dossier "Codes/PY/ FirmataStandard") :

```
# Importations des librairies et définition de fonctions

from ConnectToArduino import *

def DigitalWrite(board,pin,val):
    board.digital[pin].write(val)


# Déclaration des constantes et variables

PinLedR = 11


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)

print("Connexion à l'arduino établie - Appuyez sur Ctrl-C pour quitter\n")


# Boucle principale du programme

print("Pour allumer la DEL rouge, envoyez: ON");
print("Pour éteindre la DEL rouge, envoyez: OFF\n");

while True:
    try:
        saisie = str(input(""))
        if saisie=="ON" : DigitalWrite(board,PinLedR,1)
        elif saisie=="OFF" : DigitalWrite(board,PinLedR,0)

    except KeyboardInterrupt:
        DigitalWrite(board,PinLedR,0)
        board.exit()
        sys.exit(0)
```

<u>Déroulement du programme</u> :

- <u>Importation des librairies et définition de fonctions</u> :

. Le module **”ConnectToArduino.py”,** contenant les fonctions de connexion à l’arduino via le protocole **”Firmata standard”**, est importé.
L’importation de ce module importe également les bibliothèques :

- **"pyfirmata"** pour contrôler l’Arduino,
- **"serial.tools.list_ports"** pour déterminer la liste des ports COM disponibles,
- **"sys"** pour sortir du programme.

. Fonction **"DigitalWrite"** pour modifier l’état logique d’une broche numérique.

- Déclaration des constantes et variables :

. **PinLedR = 11** (constante correspondant au n° de la broche sur laquelle la DEL rouge est connectée)

- Connexion à l'Arduino :

. Appel de la fonction du module "**ConnectToArduino.py**" de sélection du port COM :

PortComArduino = SelectPortCOM()

Le nombre de port COM disponible est alors déterminé :

PortsCOM = list(serial.tools.list_ports.comports()

→ si nombre de port COM = 0 : message d'erreur,

→ si nombre de port COM = 1 : sélection de ce port COM pour la connexion,

→ si nombre de port COM > 1 : L'utilisateur doit sélectionner le bon port COM.

```
def SelectPortCOM():

    Nport=[]
    PortsCOM = list(serial.tools.list_ports.comports())
    for port_numero, description, address in PortsCOM:
        Nport.append(port_numero)


    if len(PortsCOM)== 0:
        print("Aucune carte Arduino n'a été détectée!")
        saisie = ""
        while saisie != "q":
            saisie = str(input("Entrez 'q' pour quitter: "))
        sys.exit()

    elif len(PortsCOM)== 1:
        PortComArduino = Nport[0]

    else:
        print("Liste des ports COM disponibles:\n")
        for i in range(len(PortsCOM)):
            print(i+1, ": ", PortsCOM[i])
        ChoixPort=False
        while ChoixPort==False:
            Choix = input("\nVeuillez indiquer le numéro du port de la carte Arduino:")
            try:
                Choix = int(Choix)
                assert Choix >= 1 and Choix <= len(PortsCOM)
                ChoixPort = True
            except AssertionError:
                print("Le numéro indiqué n'est pas entre 1 et", len(PortsCOM) ,"!")
                ChoixPort = False
            except:
                print("Vous n'avez pas saisi un numéro entre 1 et", len(PortsCOM) ,"!")
        PortComArduino = Nport[Choix-1]

    return PortComArduino
```

. Tentative d'ouverture du port COM sélectionné et de connexion à l'Arduino via le protocole **"Firmata standard"** avec la fonction "**OpenPortCom**" du module "**ConnectToArduino.py**":

board = OpenPortCom(PortComArduino)

→ message d'erreur et le programme est arrêté si l'ouverture du port COM sélectionné ou la création de l'objet "board" a échoué.

Un échec lors de la connexion se produit si le port COM indiqué ne correspond pas à un port Arduino ou si le protocole **"Firmata standard"** n'a pas été téléversé dans la mémoire de l'Arduino.

```
def OpenPortCom(PortCom):

    try:
        board = pyfirmata.Arduino(PortCom)
        testConnect = board.get_firmata_version()

        assert testConnect != None

    except AssertionError:
        AffichMessageErreur(PortCom)

    except:
        AffichMessageErreur(PortCom)

    else:
        return board


def AffichMessageErreur(PortCom):

    print("Un problème s'est produit à l'ouverture du port.\n"
        "Vérifiez que le port utilisé par la carte Arduino est bien "+ PortCom + ",\n" +
        "et que le protocole de communication FIRMATA STANDARD a bien été téléversé.\n")
    saisie = ""
    while saisie != "q":
        saisie = str(input("Entrez 'q' pour quitter: "))
    sys.exit()
```

- <u>Boucle principale du programme (boucle "while True")</u> :

. Si la connexion à l'Arduino est réussie, attente de la saisie de l'instruction :

saisie = str(input(""))

. La DEL rouge est allumée ou éteinte suivant l'instruction saisie :

if saisie=="ON" : DigitalWrite(board,PinLedR,1)

elif saisie=="OFF" : DigitalWrite(board,PinLedR,0)

- <u>Fin du programme en appuyant sur Ctrl-C</u> :

→ La DEL rouge est éteinte et le port COM est fermé.

1.2 Gestion des entrées numériques

Pour lire l'état logique d'une broche numérique (par exemple, la broche N°5), il faut la déclarer au préalable en entrée avec la commande suivante :

pin5 = board.get_pin('d:5:i')

La syntaxe est "**d**" pour digital, "**5**" est le numéro de la broche, "**i**" pour input et **"board"** est l'objet créé lors de l'appel de la méthode "Arduino" du module "pyfirmata".

On peut définir une fonction déclarant plus facilement une broche en entrée numérique :

```
def DigitalInput(board,pin):
    DigitalInputPin=board.get_pin('d:'+ str(pin) +':i')
    return DigitalInputPin
```

La syntaxe pour déclarer la broche N°5 en entrée numérique est alors plus simple :

pin5 = DigitalInput(board,5)

Ensuite on pourra lire l'état logique de la broche au moyen de cette instruction :

valeur = pin5.read()

qui retourne "**1**" lorsque l'entrée est à **5 V**, et "**0**" lorsqu'elle est à **0 V**.

Attention :

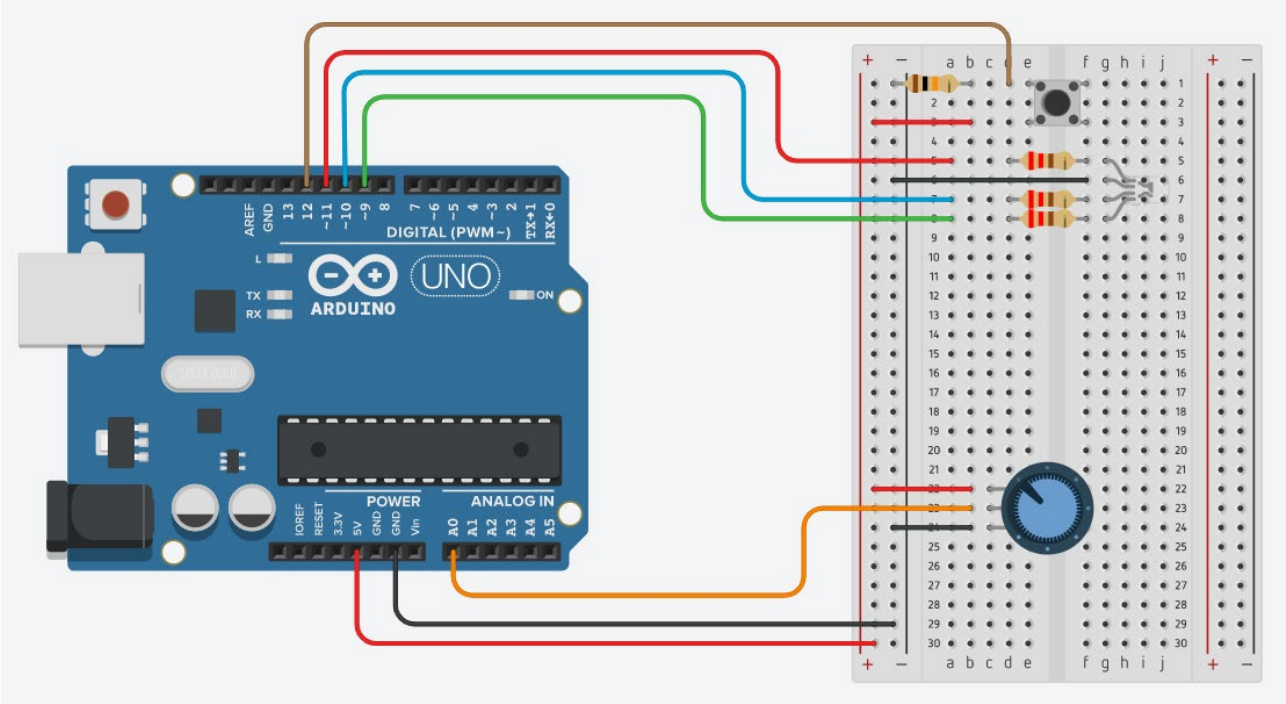
. Les données lues transitent bien-sûr par la liaison série, et pour éviter qu'un trop grand nombre de mesures n'encombrent la communication série entre l'Arduino et l'ordinateur, Il faut aussi utiliser un itérateur prévu dans pyFirmata :

```
it = pyfirmata.util.Iterator(board)
it.start()
```

. L'itérateur doit est lancé après la connexion à l'Arduino.

Exemple :

Nous allons ajouter un bouton poussoir à notre circuit support des exemples :



La sortie du bouton poussoir est reliée à la borne 12 de l'Arduino. La broche 12 sera donc à l'état logique haut (5 V) lorsque le bouton sera appuyé, et à l'état logique bas (0 V) lorsque le bouton sera relâché.

Le programme suivant (programme "LedRVBDigitalInput.py" dans le dossier "Codes/PY/FirmataStandard") permet d'allumer la DEL rouge quand le bouton poussoir est appuyé et de l'éteindre quand celui-ci est relâché :

```
# Importations des librairies et définition de fonctions

from ConnectToArduino import *
import time

def DigitalInput(board,pin):
    DigitalInputPin=board.get_pin('d:'+ str(pin) +':i')
    return DigitalInputPin

def DigitalWrite(board,pin,val):
    board.digital[pin].write(val)

def Iterateur(board):
    it = pyfirmata.util.Iterator(board)
    it.start()
    return it


# Déclaration des constantes et variables

PinLedR = 11
PinButton = 12
ValButton = 0
```

```
# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)

ArduinoIterateur = Iterateur(board)
InputPin = DigitalInput(board, PinButton)
time.sleep (0.5)
print("Connexion à l'arduino établie - Appuyez sur Ctrl-C pour quitter\n")


# Boucle principale du programme

print("Appuyez sur le bouton poussoir pour allumer la DEL.")

while True:
    try:
        ValButton = InputPin.read()
        if ValButton == 1:
            DigitalWrite(board,PinLedR,1)
        else:
            DigitalWrite(board,PinLedR,0)

    except KeyboardInterrupt:
        DigitalWrite(board,PinLedR,0)
        board.exit()
        sys.exit(0)
```

Déroulement du programme :

- Importation des librairies et définition de fonctions :

. Le module **"ConnectToArduino.py",** contenant les fonctions de connexion à l'arduino via le protocole **"Firmata standard"**,
. La bibliothèque **"time"** pour la gestion des temps de pause,
. Fonction **"DigitalInput"** pour déclarer une entrée numérique,
. Fonction **"DigitalWrite"** pour modifier l'état logique d'une broche numérique,
. Fonction **"Iterateur"** pour lancer l'itérateur de pyFirmata.

- Déclaration des constantes et variables :

. **PinLedR = 11** (constante correspondant au n° de la broche sur laquelle la DEL rouge est connectée)
. **PinButton = 12** (cst correspondant au n° de la broche sur laquelle le bouton poussoir est connecté)
. **ValButton = 0** (variable pour stocker la valeur de l'état logique de la broche du bouton poussoir)

- Connexion à l'Arduino (Idem programme précédent)

→Tentative d'ouverture du port COM sélectionné et connexion à l'Arduino:

PortComArduino = SelectPortCOM()

board = OpenPortCom(PortComArduino)

→ Si la connexion à l'Arduino est réussie:

- Lancement de l'itérateur : **ArduinoIterateur = Iterateur(board)**,

- Déclaration de la broche du bouton poussoir en entrée digitale : **InputPin = DigitalInput(board, PinButton)**,

- Attente de 500 ms pour le lancement de l'itérateur.

- <u>Boucle principale du programme (boucle "while True")</u> :

. lecture de l'état logique de la broche du bouton poussoir :

ValButton = InputPin.read()

. La DEL rouge est allumée ou éteinte suivant la valeur de ValButton :

```
if ValButton == 1:
    DigitalWrite(board,PinLedR,1)
else:
    DigitalWrite(board,PinLedR,0)
```

- <u>Fin du programme en appuyant sur Ctrl-C</u>

→ La DEL rouge est éteinte et le port COM est fermé.

1.3 Gestion des sorties analogiques

Pour utiliser une broche de l'Arduino en sortie analogique (mode PWM), il faut au préalable la déclarer en créant un objet "**pinPWM**" avec la commande suivante :

pinPWM = board.get_pin('d:numéro_de_pin:p')

La syntaxe est "**d**" pour digital, "**numéro_de_pin**" est le numéro de la broche (pour rappel, les broches supportant le PWM sont les broches 3, 5, 6, 9, 10, 11 sur l'Arduino Uno), "**p**" pour PWM et **"board"** est l'objet créé lors de l'appel de la méthode "Arduino" du module "pyfirmata".

On peut définir une fonction déclarant plus facilement une broche en sortie analogique :

```
def AnalogOutput(board,pin):
    AnalogOutputPin=board.get_pin('d:'+ str(pin) +':p')
    return AnalogOutputPin
```

La syntaxe pour déclarer, par exemple, la broche N°11 en sortie analogique est alors plus simple:

pinPWM = AnalogOutput(board,11)

La tension de la broche N°11 déclarée en sortie analogique est réglable en modifiant le rapport cyclique du signal PWM entre **0** (0 %) et **1** (100%) :

board.digital[11].write(rapport cyclique)

→ si rapport cyclique = 0, alors la tension est de 0 V
→ si rapport cyclique = 1, alors la tension est de 5 V
→ si rapport cyclique = 0.5, alors la tension est de 2,5 V

On peut également définir une fonction pour modifier le rapport cyclique d'une broche déclarée en sortie analogique :

```
def AnalogWrite(board,pin,val):
    board.digital[pin].write(val)
```

Ainsi, l'instruction pour appliquer une tension de 2,5V sur la sortie "11" déclarée en sortie analogique, de l'objet "board", est :

AnalogWrite(board, 11, 0.5)

Exemple :

Nous allons utiliser les propriétés des sorties analogiques de l'Arduino pour régler la luminosité de la DEL rouge de notre circuit entre 0 et 100%. Le programme ("LedRVBAnalogOutput.py" dans le dossier "Codes/PY/FirmataStandard") va demander à l'utilisateur de saisir un nombre entier entre 0 et 100 de façon à régler la tension de la broche sur laquelle la DEL est connectée entre 0 et 5 V.

```
# Importations des librairies et définition de fonctions

from ConnectToArduino import *

def AnalogOutput(board,pin):
    AnalogOutputPin=board.get_pin('d:'+ str(pin) +':p')
    return AnalogOutputPin

def AnalogWrite(board,pin,val):
    board.digital[pin].write(val)


# Déclaration des constantes et variables

PinLedR = 11
RapportCyclique = 0.0


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)

PinPWM = AnalogOutput(board, PinLedR)

print("Connexion à l'arduino établie - Appuyez sur Ctrl-C pour quitter\n")


# Boucle principale du programme

print("Veuillez saisir la luminosite de la DEL rouge (nombre entier entre 0 et 100):")

while True:
    try:
        ChoixRP=False
        while ChoixRP==False:
            saisie = input("")
            try:
                saisie = int(saisie)
                assert saisie >= 0 and saisie <= 100
                ChoixRP = True
            except AssertionError:
                print("Le nombre indiqué n'est pas entre 0 et 100 !")
                ChoixRP = False
            except:
                print("Vous n'avez pas saisi un nombre entre 0 et 100 !")

        RapportCyclique = float(saisie/100)
        AnalogWrite(board,PinLedR,RapportCyclique)

    except KeyboardInterrupt:
        AnalogWrite(board,PinLedR,0)
        board.exit()
        sys.exit(0)
```

Déroulement du programme :

- Importation des librairies et définition de fonctions :

. Le module **"ConnectToArduino.py",** contenant les fonctions de connexion à l'arduino via le protocole **"Firmata standard"**,
. Fonction **"AnalogOutput"** pour déclarer une sortie analogique,
. Fonction **"AnalogWrite"** pour modifier la tension d'une broche analogique.

- Déclaration des constantes et variables :

. **PinLedR = 11** (constante correspondant au n° de la broche sur laquelle la DEL rouge est connectée)
. **RapportCyclique = 0.0** (variable pour stocker la valeur du rapport cyclique du signal PWM)
. **ChoixRP =False** (variable booléenne indiquant si la saisie du rapport cyclique a été réalisée)
. **PortComArduino** (port COM sur lequel l'Arduino est connecté)

- Connexion à l'Arduino (Idem programme précédent) :

. Tentative d'ouverture du port COM sélectionné et connexion à l'Arduino:

PortComArduino = SelectPortCOM()

board = OpenPortCom(PortComArduino)

. si la connexion à l'Arduino est réussie:

→ Déclaration de la broche de la DEL rouge en sortie analogique :
PinPWM = AnalogOutput(board, PinLedR)

- Boucle principale du programme (boucle "while True") :

. Demande de saisie de la valeur de la luminosité souhaitée entre 0 et 100 (un test est effectué sur la valeur saisie):

saisie = input("")

. Calcul du rapport cyclique entre 0 et 1 :

RapportCyclique = float(saisie/100)

. La luminosité de la DEL rouge est réglée à l'aide du rapport cyclique calculé :

AnalogWrite(board,PinLedR,RapportCyclique)

- Fin du programme en appuyant sur Ctrl-C :

→ La DEL rouge est éteinte et le port COM est fermé.

1.4 Gestion des entrées analogiques

Pour lire la valeur de la tension d'une entrée analogique (par exemple, la broche A0), il faut la déclarer au préalable en entrée analogique avec la commande suivante :

pinA0 = board.get_pin('a:0:i')

La syntaxe est "**a**" pour analogique, "**0**" est le numéro de la broche A0, "**i**" pour input et **"board"** est l'objet créé lors de l'appel de la méthode "Arduino" du module "pyfirmata".

On peut définir une fonction déclarant plus facilement une broche en entrée analogique :

```
def AnalogInput(board,pin):
   AnalogInputPin=board.get_pin('a:'+ str(pin) +':i')
   return AnalogInputPin
```

La syntaxe pour déclarer la broche A0 en entrée analogique est alors plus simple :

pinA0 = AnalogInput(board,0)

Ensuite, on utilise la fonction "**read()**" pour lire la valeur de la tension de la broche :

pinA0.read()

Attention :

. La fonction "**read()**" pour une broche déclarée en entrée analogique retourne une valeur décimale entre 0 et 1 (alors qu'en langage Arduino, "analogRead()" retourne un entier entre 0 et 1023).

. Il faut également utiliser un itérateur afin d'éviter de saturer la communication série entre l'Arduino et l'ordinateur hôte.

Exemple :

Notre circuit, support des exemples, dispose d'un potentiomètre dont le "point milieu" est relié à la broche A0 de l'Arduino. Suivant la position du "point milieu", la tension appliquée à la broche A0 varie entre 0 et 5 V. On peut donc utiliser le potentiomètre pour régler la luminosité de la DEL rouge, par exemple.

Le programme ("LedRVBAnalogInput.py" dans le dossier "Codes/PY/FirmataStandard") suivant, lit la valeur de la broche A0 et règle la luminosité de la DEL rouge à cette valeur :

```
# Importations des librairies et définition de fonctions

from PyFirmataDef import *
from ConnectToArduino import *
import time


# Déclaration des constantes et variables

PinLedR = 11
PinPot = 0
ValPot = 0


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)

InputPin = AnalogInput(board, PinPot)
PinPWM = AnalogOutput(board, PinLedR)

ArduinoIterateur = Iterateur(board)
time.sleep(0.5)

print("Connexion à l'arduino établie - Appuyez sur Ctrl-C pour quitter\n")


# Boucle principale du programme

while True:
    try:
        ValPot = InputPin.read()
        AnalogWrite(board,PinLedR,ValPot)

    except KeyboardInterrupt:
        AnalogWrite(board,PinLedR,0)
        board.exit()
        sys.exit(0)
```

Déroulement du programme :

- Importation des librairies et définition de fonctions :

. Le module **"ConnectToArduino.py",** contenant les fonctions de connexion à l'arduino via le protocole **"Firmata standard"**,

. Toutes les fonctions utiles à l'utilisation de "**PyFirmata**" (fonction de déclaration des entrées et sorties, d'itérateur, d'écritures…) que nous avons définies jusqu'à présent ont été regroupées dans un fichier Python, nommé "**PyFirmataDef.Py**" que l'on peut importer dans tous les programmes, à condition que le fichier des fonctions soit dans le même dossier que le fichier du programme, avec l'instruction :

from PyFirmataDef import *

```
import pyfirmata


def DigitalInput(board,pin):
    DigitalInputPin=board.get_pin('d:'+ str(pin) +':i')
    return DigitalInputPin

def AnalogOutput(board,pin):
    AnalogOutputPin=board.get_pin('d:'+ str(pin) +':p')
    return AnalogOutputPin

def AnalogInput(board,pin):
    AnalogInputPin = board.get_pin('a:'+ str(pin) +':i')
    return AnalogInputPin

def DigitalWrite(board,pin,val):
    board.digital[pin].write(val)

def AnalogWrite(board,pin,val):
    board.digital[pin].write(val)

def Iterateur(board):
    it = pyfirmata.util.Iterator(board)
    it.start()
    return it
```

. La bibliothèque **"time"** pour la gestion des temps de pause.

- <u>Déclaration des constantes et variables</u> :

. **PinLedR = 11** (constante correspondant au n° de la broche sur laquelle la DEL rouge est connectée)

. **PinPot = 0** (constante correspondant au n° de la broche A0 sur laquelle le potentiomètre est connecté)

. **ValPot = 0** (variable pour stocker la valeur de l'entrée analogique A0)

. **PortComArduino** (port COM sur lequel l'Arduino est connecté)

- <u>Connexion à l'Arduino</u> (Idem programme précédent)

. Détection du port COM , tentative d'ouverture du port COM sélectionné et connexion à l'Arduino:

PortComArduino = SelectPortCOM()

board = OpenPortCom(PortComArduino)

. Si la connexion à l'Arduino est réussie:

→ Déclaration de la broche de la DEL rouge en sortie analogique :
PinPWM = AnalogOutput(board, PinLedR)

→ Déclaration de la broche du potentiomètre en entrée analogique :
InputPin = AnalogInput(board, PinPot))

→ Lancement de l'itérateur : **ArduinoIterateur = Iterateur(board)**

- Boucle principale du programme (boucle "while True") :

. Lecture de la valeur (nombre décimal flottant entre 0 et 1) de la broche du potentiomètre :

ValPot = InputPin.read()

. La luminosité de la DEL rouge est réglée à cette valeur :

AnalogWrite(board,PinLedR, ValPot)

- Fin du programme en appuyant sur Ctrl-C

→ La DEL rouge est éteinte et le port COM est fermé.