

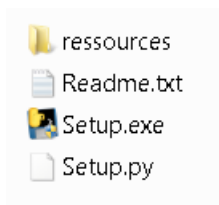
Programmer une carte Arduino™ Uno en Python avec PythonArduinoPackage

PythonArduinoPackage contient Python et Firmata qui est un protocole permettant la communication entre un ordinateur et un microcontrôleur.

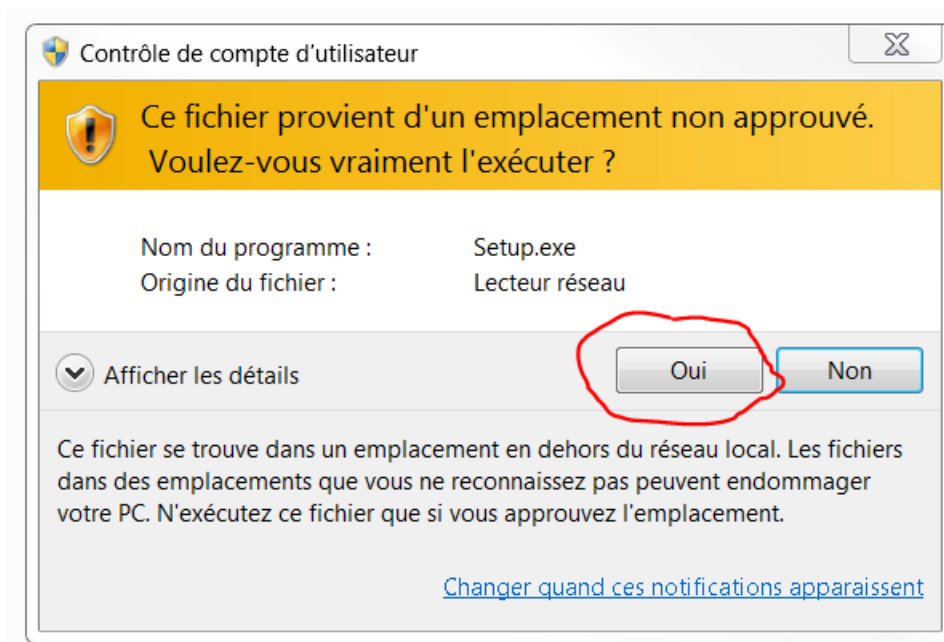
Une fois ce protocole téléversé dans la carte, elle devient contrôlable via les instructions en langage Python de la bibliothèque **pyFirmata2Ext**

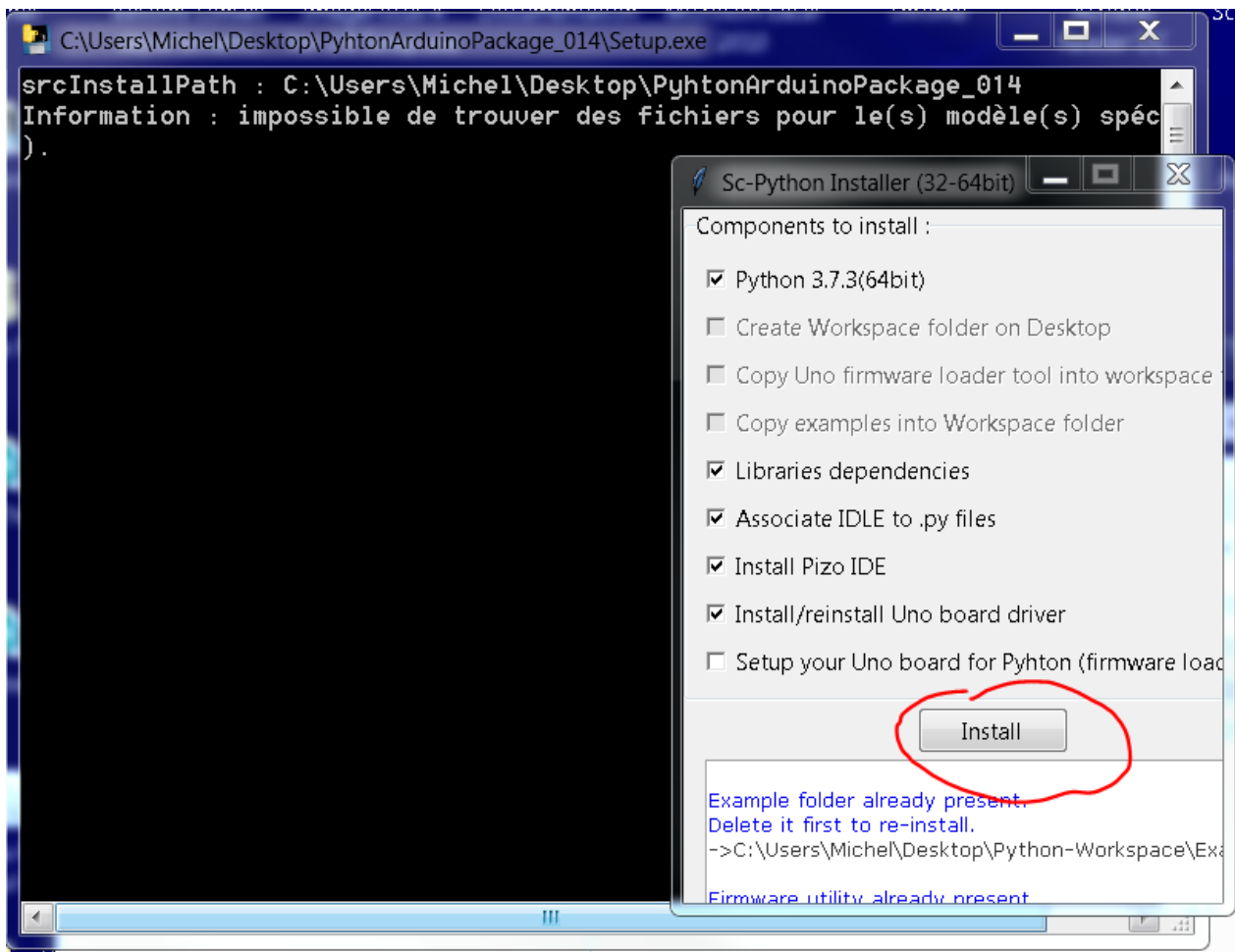
A – Installation du package de logiciels PythonArduinoPackage sur votre ordinateur

Placer le dossier PythonArduinoPackage_014 sur le bureau de votre ordinateur



Lancer l'exécutable « Setup.exe »





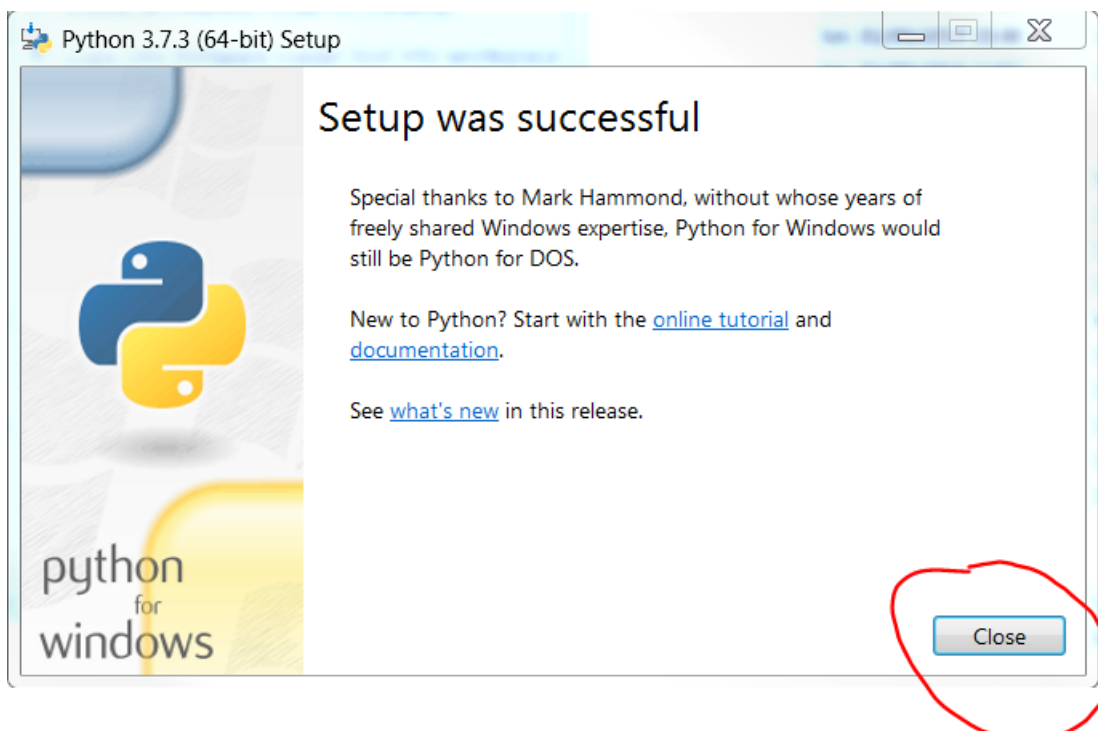
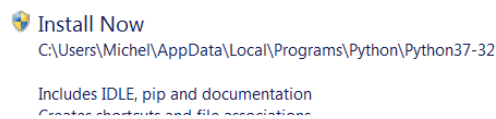
Lancer l'installation des composants proposés. Si vous aviez des versions antérieures, il convient de les désinstaller.

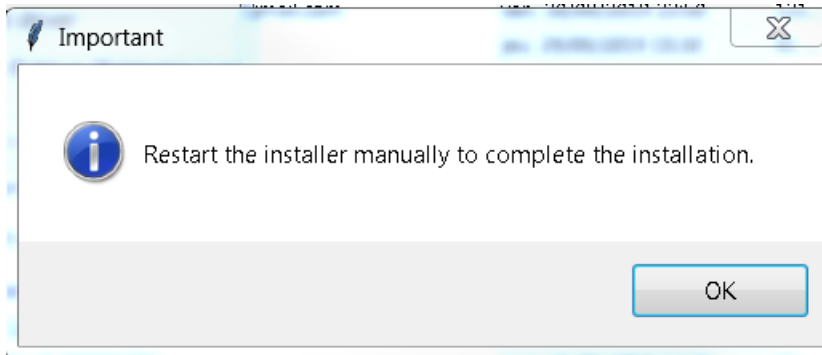
Noter que cette version de Python n'est pas compatible avec les versions de Windows XP et antérieures.

Attention, à ce stade, il est important de cocher la case « Add Python 3.7 to PATH » avant de lancer l'installation

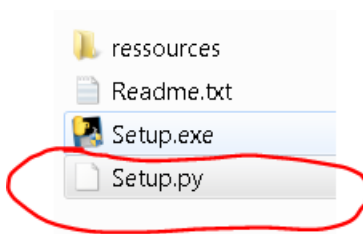


Une fois la case cochée, Installer Python en cliquant sur « Install Now »



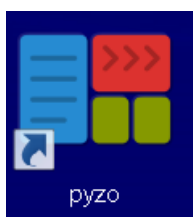


La fenêtre ci-dessus demande de lancer « Setup.py » pour compléter l'installation :



Accepter toutes les demandes d'installation.

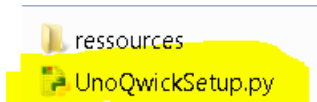
L'éditeur choisi pour programmer en Python installé est **Pyzo**, le raccourci ci-dessous est installé sur votre bureau :



B – Configuration du microcontrôleur Plug'Uino® Uno

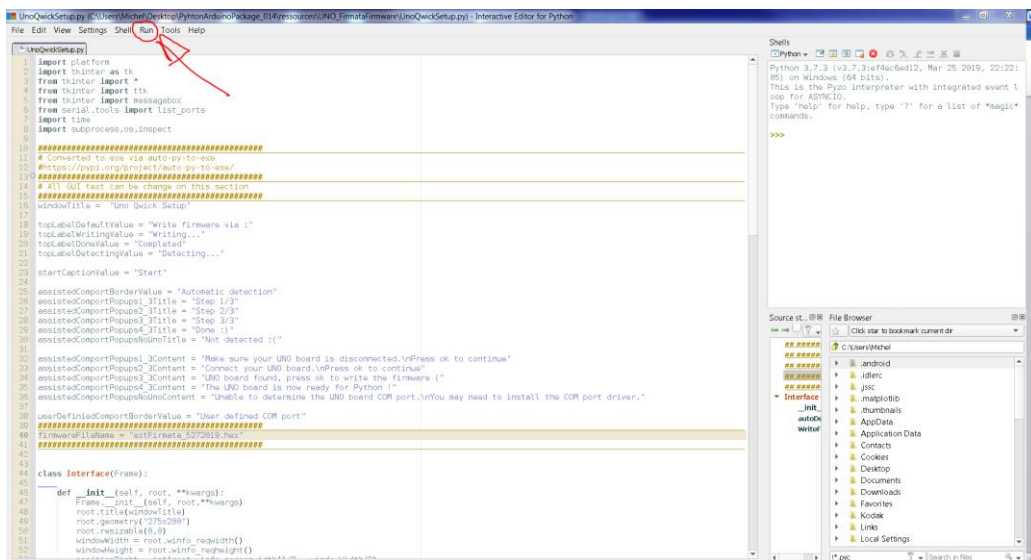
1- Plug'Uino® Uno doit être branché à l'ordinateur au moyen d'un câble USB.

Exécuter le programme UnoQuickSetup.py. (UnoFWLoader.exe) disponible dans le dossier ressources/UNO_Firmware

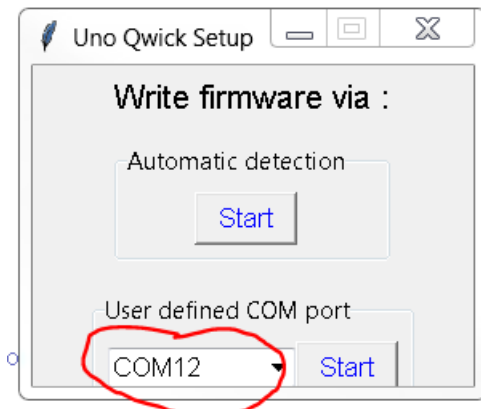


Double cliquer sur le fichier UnoQuickSetup.py

Le programme Python d'installation du Firmware s'ouvre dans l'éditeur Pyzo

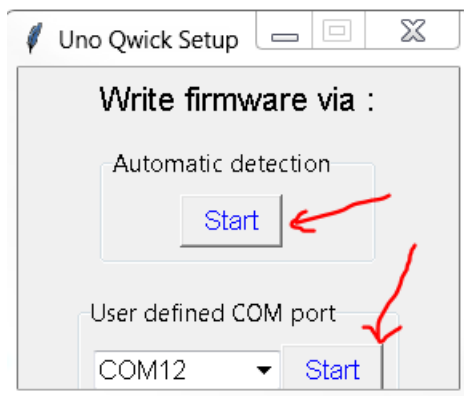


Lancer le programme en cliquant dans le menu « Run » sur « Run file as script »



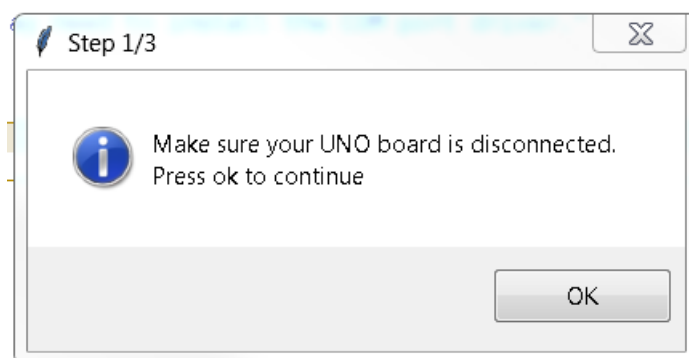
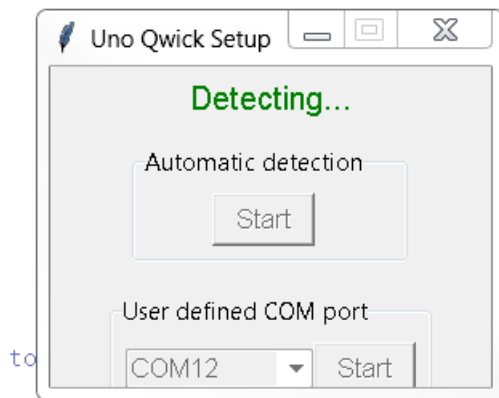
Le port de communication s'affiche, ici COM12, ce qui montre que le microcontrôleur communique bien avec votre ordinateur par la liaison série.

Cliquer sur un des 2 boutons « Start » au choix (les 2 fonctionnent):

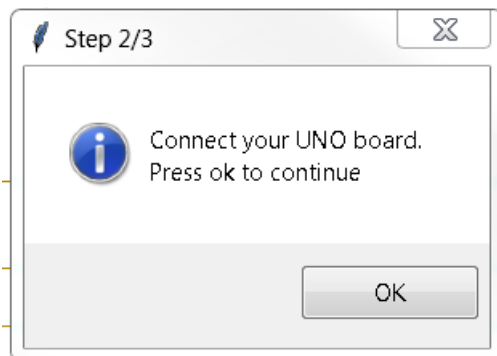


- 2- Suivre les instructions de l'assistant pour charger le firmware modifié de Firmata pour Arduino.
 Cette version modifiée de Firmata implémente les modules Plug'Uino® spécifiques : LED programmable RVB, Matrice de LED, Télémètre à US etc...

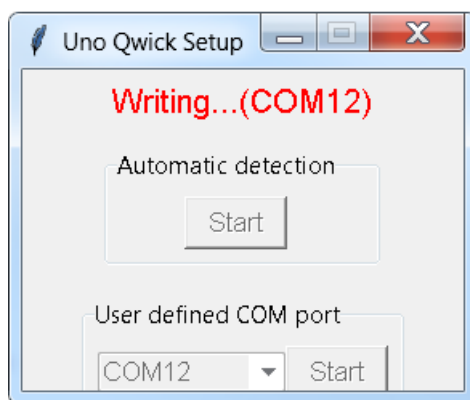
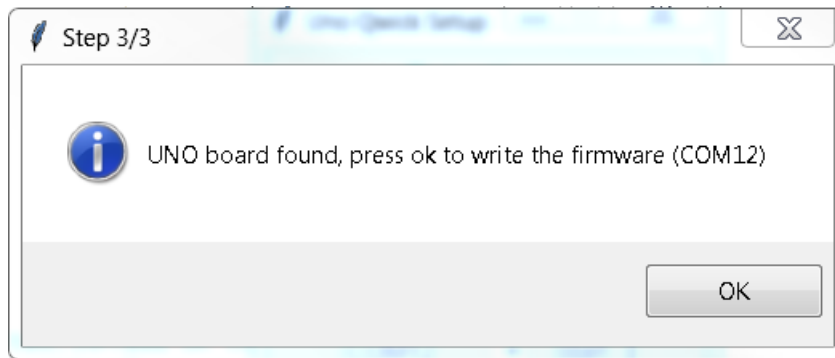
A l'étape 1/3, assurez-vous que le câble USB est bien débranché



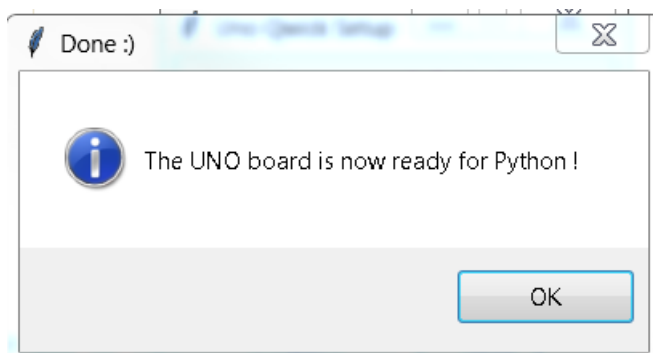
A l'étape 2/3, rebrancher le câble et cliquer sur OK.



A l'étape 3/3, cliquer sur OK pour écrire le Firmware dans le microcontrôleur :



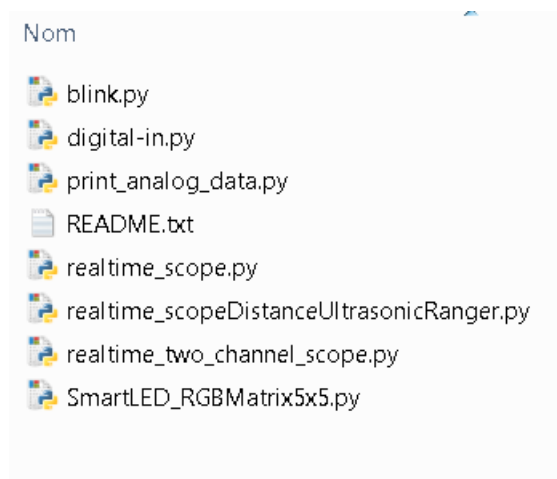
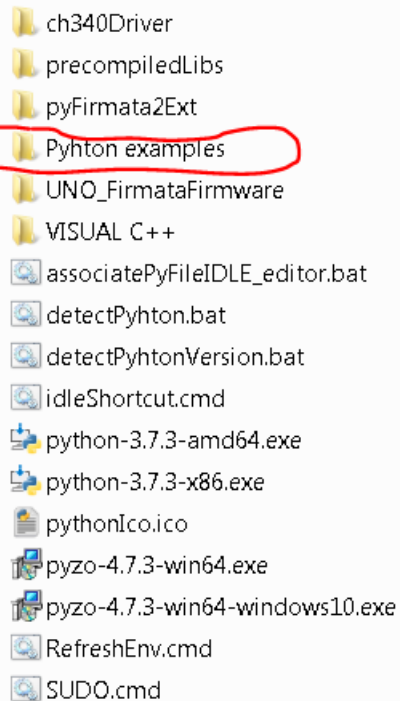
La fenêtre ci-dessous indique que le microcontrôleur est prêt à être programmé en langage Python :



Le microcontrôleur Plug'Uino® Uno est maintenant prêt à être asservi via Python.

C – Utiliser un des exemples fournis dans le dossier « Python examples » :

Dans le dossier ressources, ouvrir le dossier Python exemples.



1- Exécuter le programme *realtime_scopeDistanceUltrasonicRanger.py*

Vous pouvez, par exemple, exécuter le programme *realtime_scopeDistanceUltrasonicRanger.py* fourni dans le dossier « Exemples ».

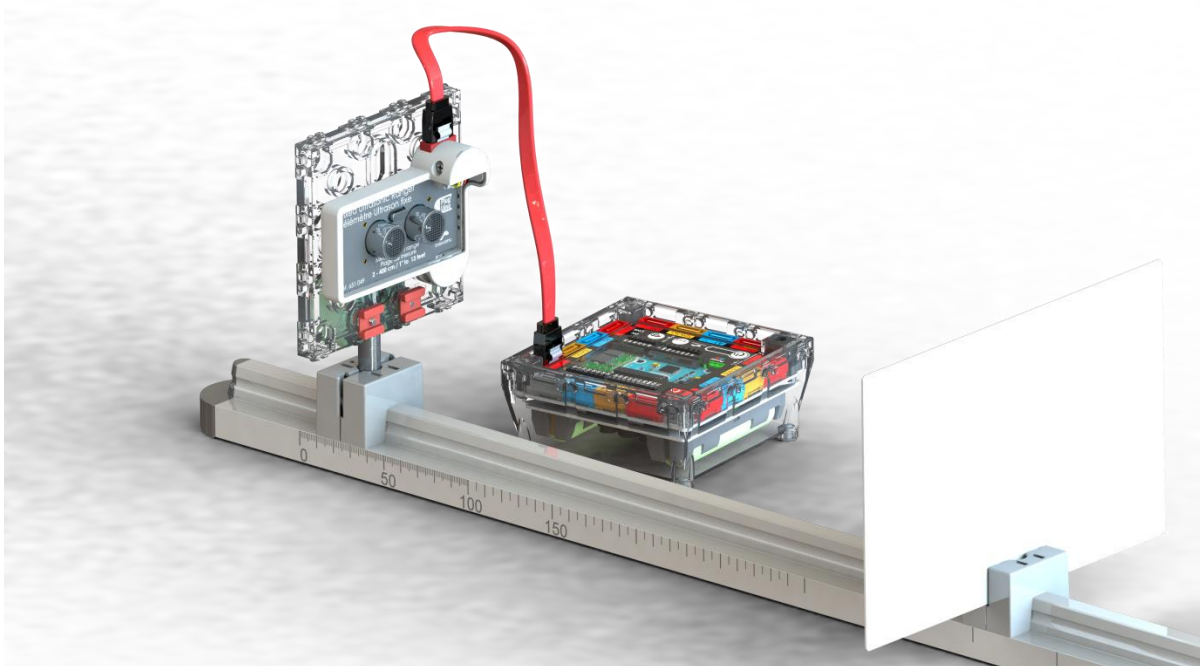
Ce programme permet de tracer un graphe en temps réel de la mesure de la distance à l'aide d'un capteur télémètre à Ultra Son.

Ce programme permet de tracer un graphe de l'évolution de la distance entre un écran et le télémètre à ultrasons Plug'Uino®.

Il convient de brancher préalablement le capteur télémètre à US Plug'Uino® à l'aide d'un câble SATA :

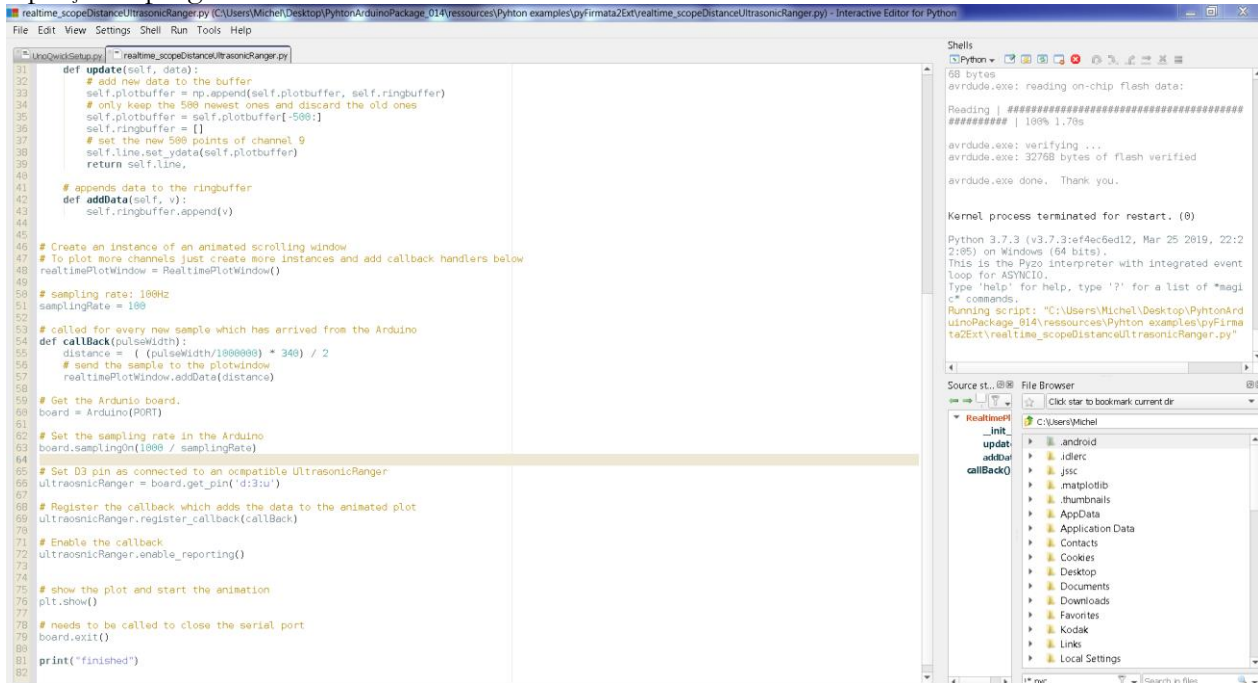


sur l'entrée définie dans le programme `realtime_scopeDistanceUltrasonicRanger.py`, ici sur l'entrée D3 (définie et modifiable à la ligne 63 du programme).



Ouvrir le programme (Run file as a script) `realtime_scopeDistanceUltrasonicRanger.p` à l'aide de l'éditeur Pyzo :

Aperçu du programme :



```

31 def update(self, data):
32     # add new data to the buffer
33     self.plotbuffer = np.append(self.plotbuffer, self.ringbuffer)
34     # only keep the 500 newest ones and discard the old ones
35     self.plotbuffer = self.plotbuffer[-500:]
36     self.ringbuffer = []
37     # set the new 500 points of channel 9
38     self.line.set_ydata(self.plotbuffer)
39     return self.line,
40
41 # appends data to the ringbuffer
42 def addData(self, v):
43     self.ringbuffer.append(v)
44
45 # Create an instance of an animated scrolling window
46 # To plot more channels just create more instances and add callback handlers below
47 realtimePlotWindow = RealtimePlotWindow()
48
49 # sampling rate: 100Hz
50 samplingRate = 100
51
52 # called for every new sample which has arrived from the Arduino
53 def callback(pulseWidth):
54     distance = ( pulseWidth/1000000 * 340 ) / 2
55     # send the sample to the plotwindow
56     realtimePlotWindow.addData(distance)
57
58 # Get the Arduino board.
59 board = Arduino(PORT)
60
61 # Set the sampling rate in the Arduino
62 board.samplingOn(1000 / samplingRate)
63
64 # Set D3 pin as connected to an compatible UltrasonicRanger
65 ultrasonicRanger = board.get_pin('d:3:u')
66
67 # Register the callback which adds the data to the animated plot
68 ultrasonicRanger.register_callback(callback)
69
70 # Enable the callback
71 ultrasonicRanger.enable_reporting()
72
73 # show the plot and start the animation
74 plt.show()
75
76 # needs to be called to close the serial port
77 board.exit()
78
79 print("finished")
80
81
82

```

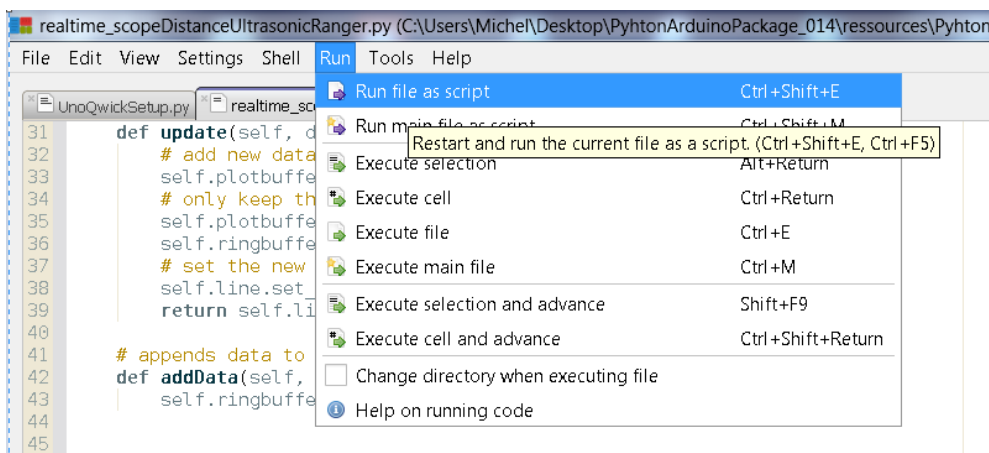
Chercher dans les lignes de ce programme, l'instruction qui définit l'entrée où doit être connecté le télémètre à US : ici D3

```

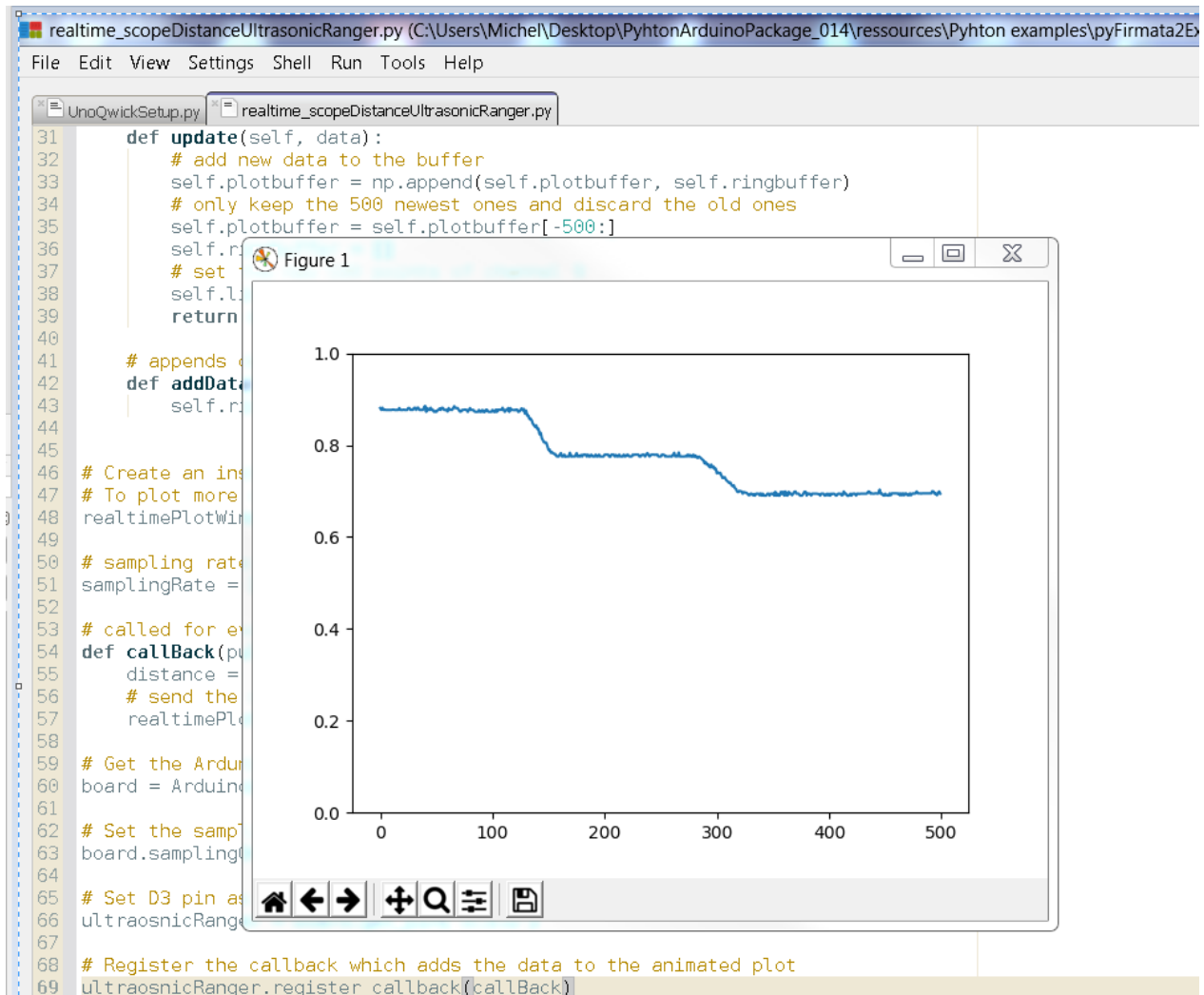
64
65 # Set D3 pin as connected to an compatible UltrasonicRanger
66 ultrasonicRanger = board.get_pin('d:3:u')
67

```

Lancer le programme : Menu **Run**

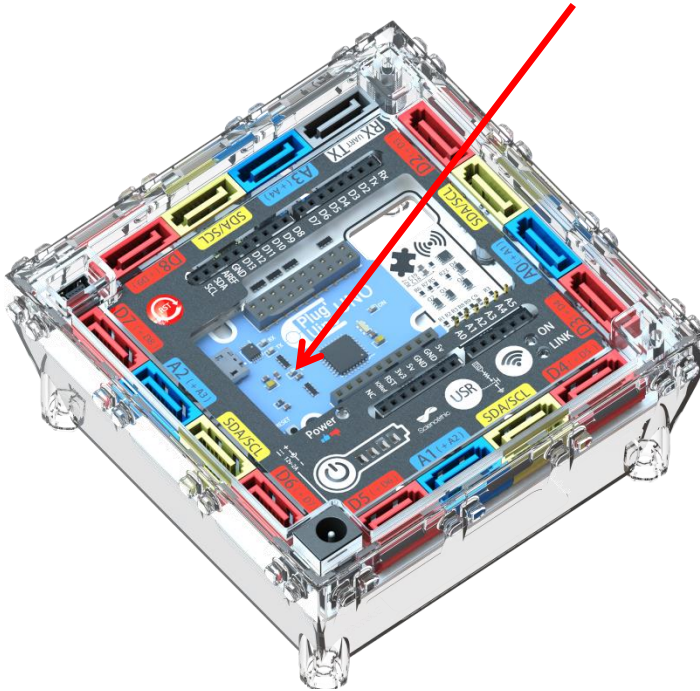


Au bout de quelques secondes, une fenêtre s'affiche et vous pouvez tracer les variations de la distance entre le capteur télémètre US et l'écran qui réfléchit les US.



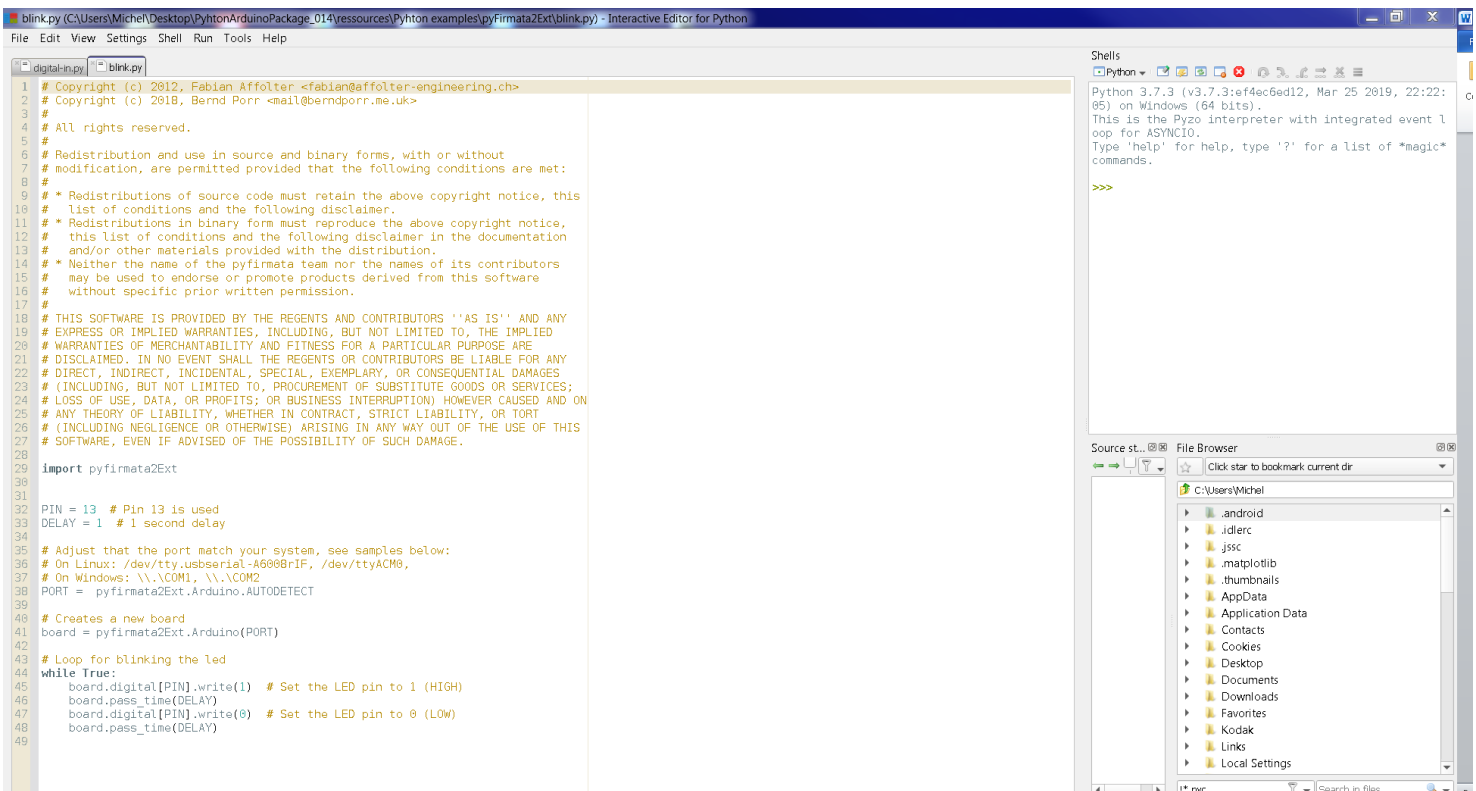
1- Exécuter le programme `blink.py`

Ce programme permet de faire clignoter la LED orange sur la carte Plug'Uino® Uno



Attention, ce programme fait appel à la nouvelle librairie `PyFirmata2Ext`.

Selon la version de votre package, il convient de corriger dans les lignes de code où apparaît le texte « `pyfirmata2` », le remplacer par « `pyfirmata2Ext` », ici lignes 38 et 41



```

1  # Copyright (c) 2012, Fabian Affolter <fabian@affolter-engineering.ch>
2  # Copyright (c) 2018, Bernd Porr <mail@berndporr.me.uk>
3  #
4  # All rights reserved.
5  #
6  # Redistribution and use in source and binary forms, with or without
7  # modification, are permitted provided that the following conditions are met:
8  #
9  # * Redistributions of source code must retain the above copyright notice, this
10 # list of conditions and the following disclaimer.
11 # * Redistributions in binary form must reproduce the above copyright notice,
12 # this list of conditions and the following disclaimer in the documentation
13 # and/or other materials provided with the distribution.
14 # * Neither the name of the pyfirmata team nor the names of its contributors
15 # may be used to endorse or promote products derived from this software
16 # without specific prior written permission.
17 #
18 # THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS 'AS IS' AND ANY
19 # EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
20 # WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
21 # DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY
22 # DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
23 # (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
24 # LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
25 # ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
26 # (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
27 # SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
28
29 import pyfirmata2Ext
30
31
32 PIN = 13 # Pin 13 is used
33 DELAY = 1 # 1 second delay
34
35 # Adjust that the port match your system, see samples below:
36 # On Linux: /dev/tty.usbserial-A680B1F, /dev/ttyACM8,
37 # On Windows: \\.\COM1, \\.\COM2
38 PORT = pyfirmata2Ext.Arduino.AUTODETECT
39
40 # Creates a new board
41 board = pyfirmata2Ext.Arduino(PORT)
42
43 # Loop for blinking the led
44 while True:
45     board.digital[PIN].write(1) # Set the LED pin to 1 (HIGH)
46     board.pass_time(DELAY)
47     board.digital[PIN].write(0) # Set the LED pin to 0 (LOW)
48     board.pass_time(DELAY)
49

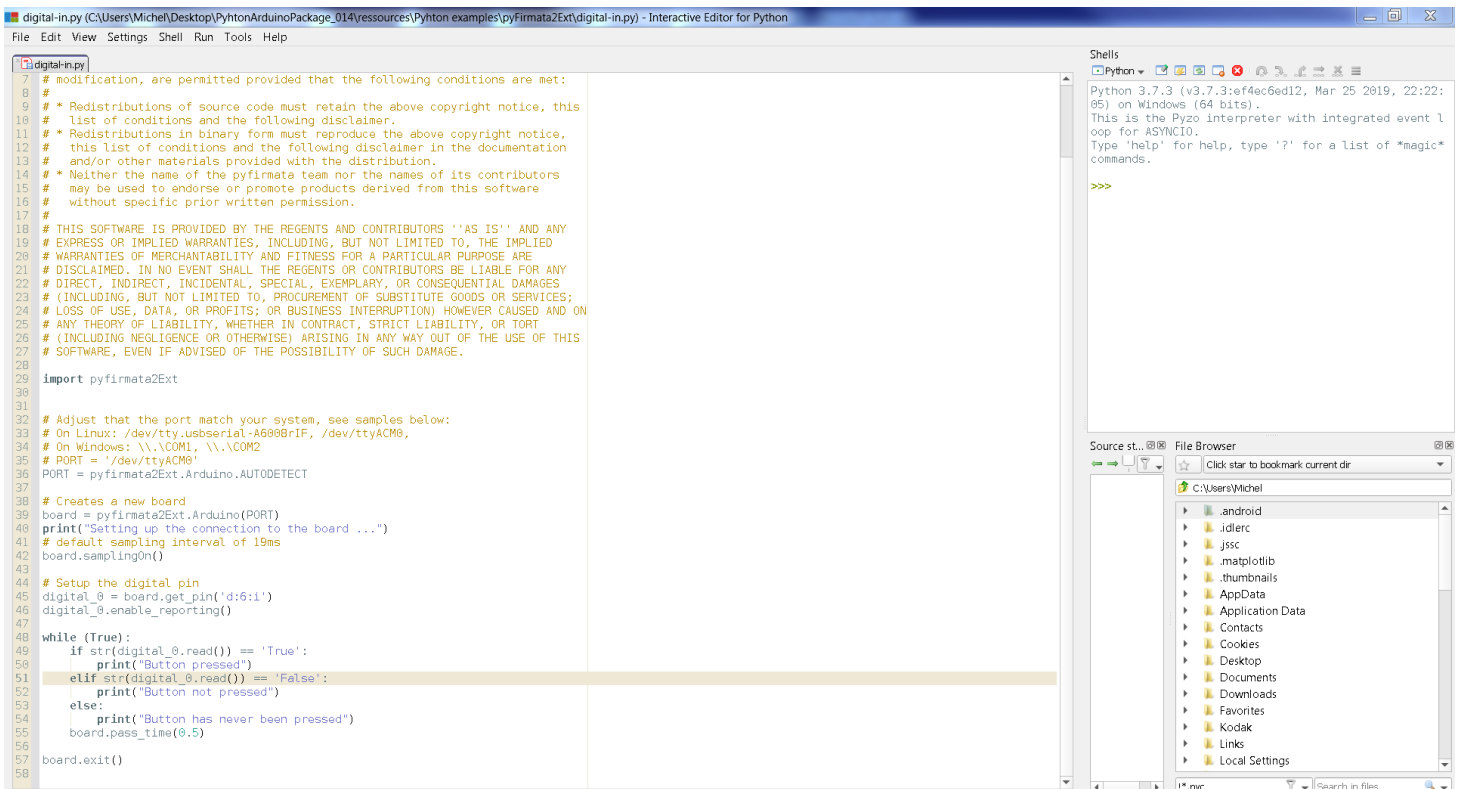
```

2- Exécuter le programme **digital-in.py**

Ce programme permet de tester l'entrée D0 et d'indiquer si le capteur numérique branché sur cette entrée (Bouton poussoir par exemple) est activé (Button pressed) ou non (Button not pressed)

Attention, ce programme fait appel à la nouvelle librairie PyFirmata2Ext.

Selon la version de votre package, il convient de corriger dans les lignes de code où apparaît le texte « pyfirmata2 », le remplacer par « pyfirmata2Ext », ici lignes 36 et 39.



```

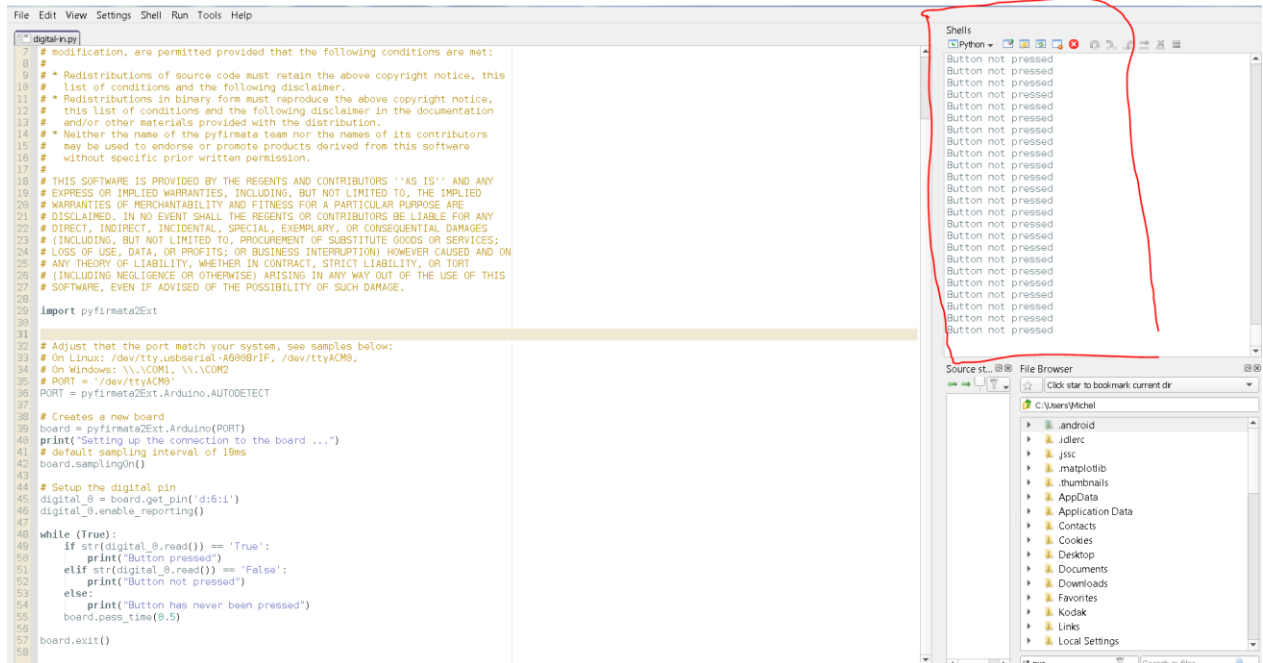
1 # modification, are permitted provided that the following conditions are met:
2 #
3 # * Redistributions of source code must retain the above copyright notice, this
4 #   list of conditions and the following disclaimer.
5 # * Redistributions in binary form must reproduce the above copyright notice,
6 #   this list of conditions and the following disclaimer in the documentation
7 #   and/or other materials provided with the distribution.
8 # * Neither the name of the pyfirmata team nor the names of its contributors
9 #   may be used to endorse or promote products derived from this software
10 #   without specific prior written permission.
11 #
12 # THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS 'AS IS' AND ANY
13 # EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
14 # WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
15 # DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY
16 # DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
17 # (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
18 # LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
19 # ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
20 # (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
21 # SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
22
23 import pyfirmata2Ext
24
25 # Adjust that the port match your system, see samples below:
26 # On Linux: /dev/tty.usbserial-A600B8rIF, /dev/ttyACM0,
27 # On Windows: \\.\COM1, \\.\COM2
28 # PORT = '/dev/ttyACM0'
29 PORT = pyfirmata2Ext.Arduino.AUTODETECT
30
31 # Creates a new board
32 board = pyfirmata2Ext.Arduino(PORT)
33 print("Setting up the connection to the board ...")
34 # default sampling interval of 19ms
35 board.samplingOn()
36
37 # Setup the digital pin
38 digital_0 = board.get_pin('d:6:i')
39 digital_0.enable_reporting()
40
41 while (True):
42     if str(digital_0.read()) == 'True':
43         print("Button pressed")
44     elif str(digital_0.read()) == 'False':
45         print("Button not pressed")
46     else:
47         print("Button has never been pressed")
48     board.pass_time(0.5)
49
50 board.exit()

```

Brancher un bouton poussoir Plug'Uino® sur l'entrée D6 :



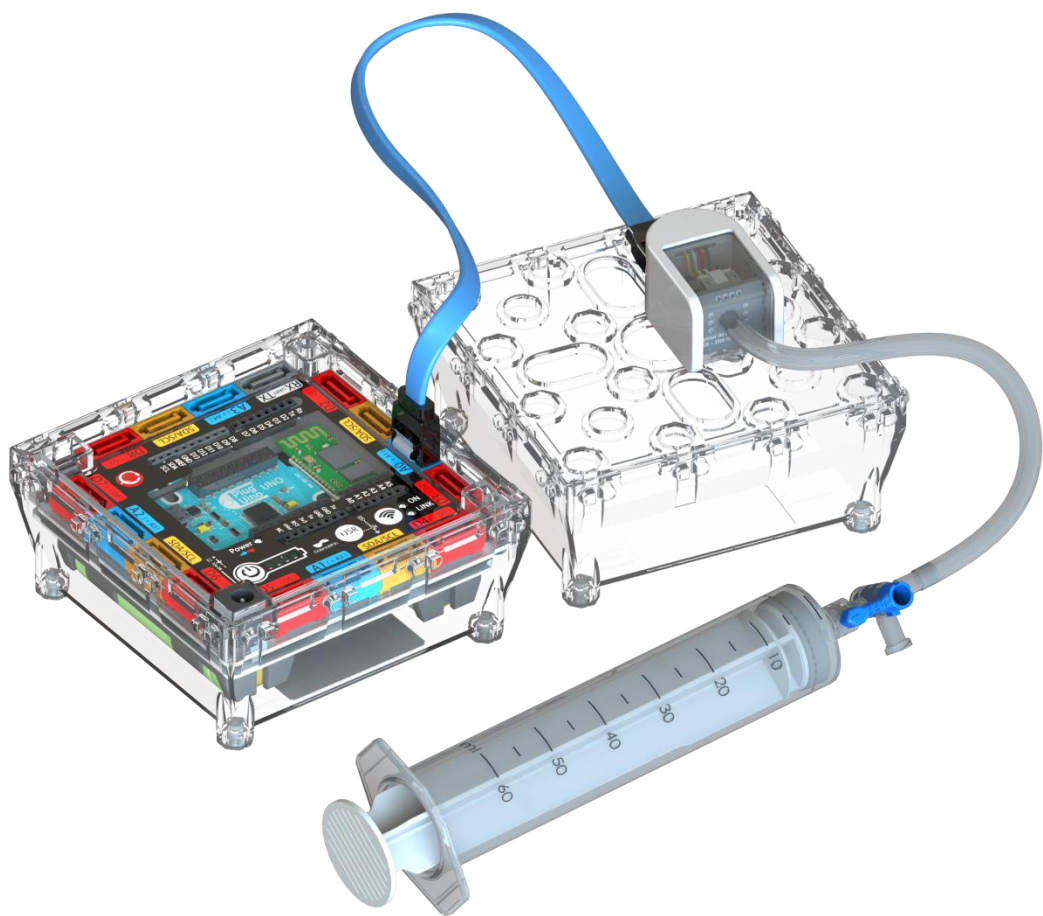
Le programme s'exécute dans le Shell de Pyzo à droite de votre écran :



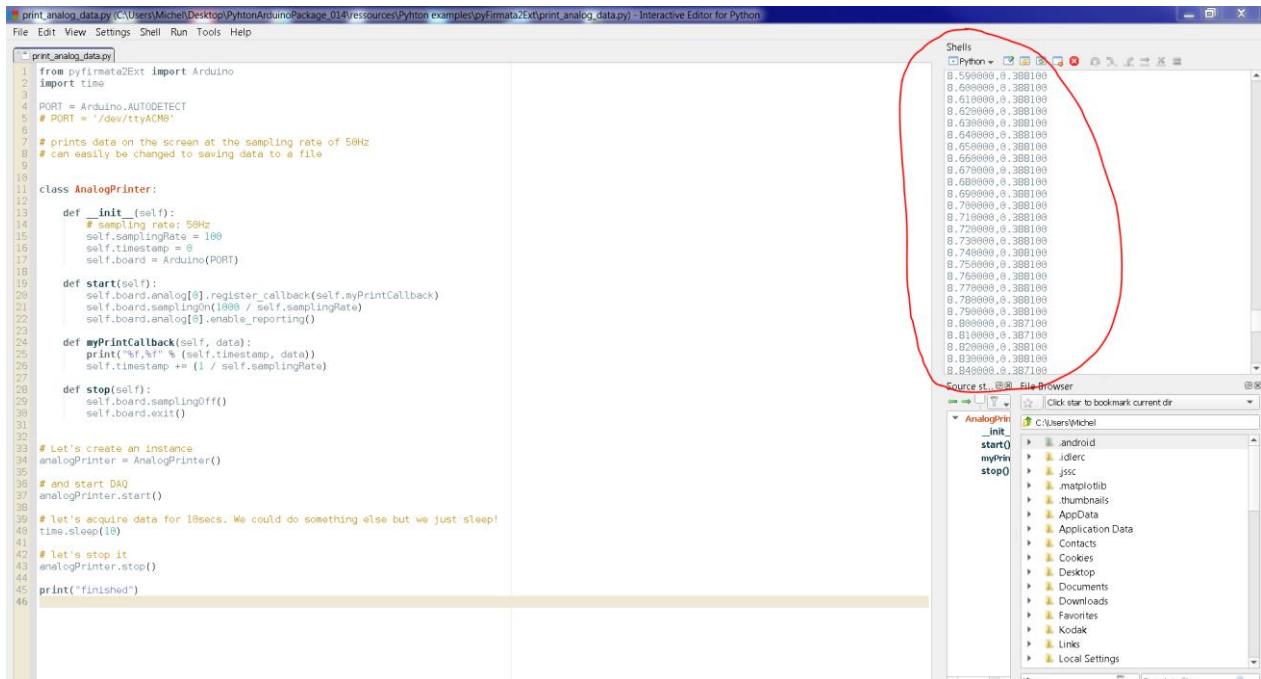
3- Exécuter le programme `print_analog_data.py`

Ce programme permet de lire et d'écrire des données analogiques au rythme d'acquisition de 50 Hz sur l'entrée A0 pendant 10 secondes (la valeur affichée par Python évolue de 0 à 1 pour une tension analogique en entrée du CAN qui évolue de 0 à 5 V) .

Brancher par exemple le capteur Pression -1000/+2000 hPa, équipé d'une seringue sur l'entrée A0 et enregistrer l'évolution de la pression sur 10 secondes.



Les valeurs obtenues suite à l'exécution du programme apparaissent dans le shell à droite et peuvent ensuite être copiées-collées dans Excel pour un traitement.



```

1 from pyfirmata2 import Arduino
2 import time
3
4 PORT = Arduino.AUTODETECT
5 # PORT = '/dev/ttyACM0'
6
7 # prints data on the screen at the sampling rate of 50Hz
8 # can easily be changed to saving data to a file
9
10
11 class AnalogPrinter:
12
13     def __init__(self):
14         # sampling rate: 50Hz
15         self.samplingRate = 100
16         self.timestamp = 0
17         self.board = Arduino(PORT)
18
19     def start(self):
20         self.board.analog[0].register_callback(self.myPrintCallback)
21         self.board.samplingOn(1000 / self.samplingRate)
22         self.board.analog[0].enable_reporting()
23
24     def myPrintCallback(self, data):
25         print("%f,%f" % (self.timestamp, data))
26         self.timestamp += (1 / self.samplingRate)
27
28     def stop(self):
29         self.board.samplingOff()
30         self.board.exit()
31
32
33 # Let's create an instance
34 analogPrinter = AnalogPrinter()
35
36 # and start DAQ
37 analogPrinter.start()
38
39 # let's acquire data for 10secs. We could do something else but we just sleep!
40 time.sleep(10)
41
42 # let's stop it
43 analogPrinter.stop()
44
45 print("finished")
46

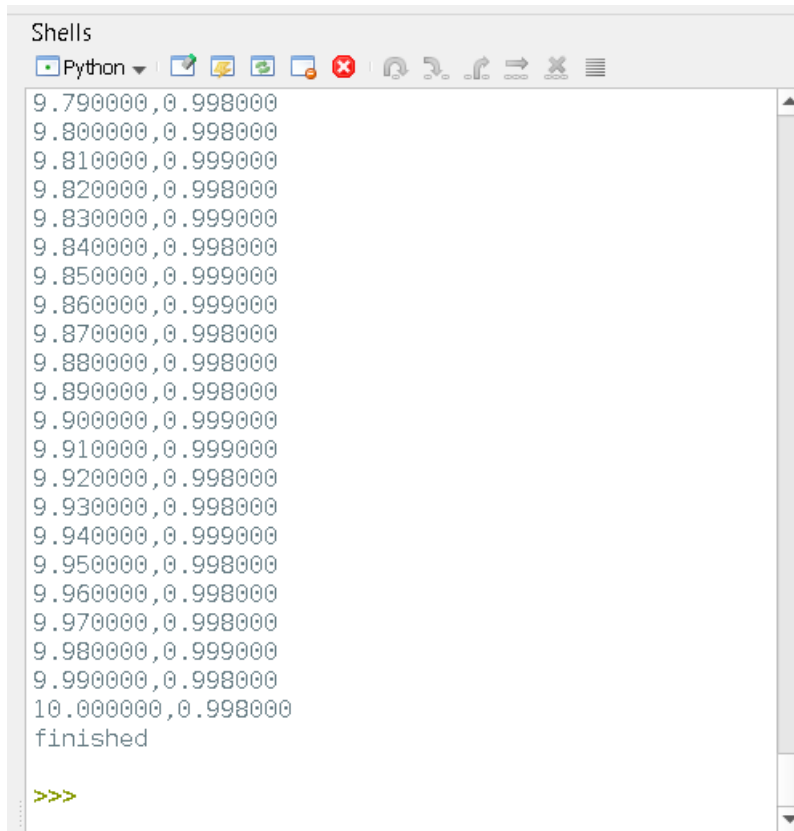
```

The shell on the right displays the following output (circled in red):

```

8.599000,0.388100
8.609000,0.388100
8.619000,0.388100
8.629000,0.388100
8.639000,0.388100
8.649000,0.388100
8.659000,0.388100
8.669000,0.388100
8.679000,0.388100
8.689000,0.388100
8.699000,0.388100
8.709000,0.388100
8.719000,0.388100
8.729000,0.388100
8.739000,0.388100
8.749000,0.388100
8.759000,0.388100
8.769000,0.388100
8.779000,0.388100
8.789000,0.388100
8.799000,0.388100
8.809000,0.387100
8.819000,0.387100
8.829000,0.388100
8.839000,0.388100
8.849000,0.387100

```



```

Shells
Python
9.790000,0.998000
9.800000,0.998000
9.810000,0.999000
9.820000,0.998000
9.830000,0.999000
9.840000,0.998000
9.850000,0.999000
9.860000,0.999000
9.870000,0.998000
9.880000,0.998000
9.890000,0.998000
9.900000,0.999000
9.910000,0.999000
9.920000,0.998000
9.930000,0.998000
9.940000,0.999000
9.950000,0.998000
9.960000,0.998000
9.970000,0.998000
9.980000,0.999000
9.990000,0.998000
10.000000,0.998000
finished
>>>

```