Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

L1: Almost real-time monitoring

Autors:

Filbà Gallego, Marc
Pérez Gutierrez, Ferran

Professor:

Mart Colom, Pau

1 de març del 2016

Índex

1 Real TODO list **2**

2 Adquisició de les dades **3**
2.1 Funcionament codi Arduino 3

3 Monitorització de la dades **6**
3.1 Funcionament del codi python 6
3.2 Resultat final del mostreig per pantalla 8

4 Annex **9**
4.1 Codi Arduino 9
4.2 Codi Python 12

1 Real TODO list

(Almost) Real-time monitoring:

Enable the system (Arduino and MPU) to plot in the PC the 3 accelerations and 3 angular velocities in the International System of Units (m/s2 and /sec) in sliding windows over time. Plot also the temperature (C)."
Program the monitoring in such a way that the user may be able to:

- chose the sensibility/accuracy of the measures by specifying the scale range of 250, 500, 1000, and 2000 /sec for the gyro, and the scale range of 2g, 4g, 8g, and 16g for the accelerometer.

- the data acquisition frequency (e.g. from 1Hz to 200Hz 1s to 5ms).

2 Adquisició de les dades

2.1 Funcionament codi Arduino

Fem una breu explicació de com adquirim les dades l'Arduino.
Primer de tot, quan s'inicialitza el sistema tenim les següents opcions a configurar (*línies 38-57*):

- **Tecla 1**: Es mostra el menú per terminal.
- **Tecla 2**: S'inicia l'adquisició de les dades.
- **Tecla 3**: S'atura l'adquisició de les dades.
- **Tecla 4 a la 7**: Podem escollir el rang de precisió d'adquisició de la acceleració amb una precisió que oscil·la entre els següents valors: ***2g***, ***4g***, ***8g*** i ***16g***
- **Tecla 8 a la 11**: Podem escollir el rang de precisió d'adquisició de la velocitat angular amb una precisió que oscil·la entre els següents valors: ***250***, ***500***, ***1000*** i ***2000*** o/sec.
- **Tecla 12 a la 15**: Increment i decrement de la freqüència de l'adquisició de dades en 5ms o 50ms.

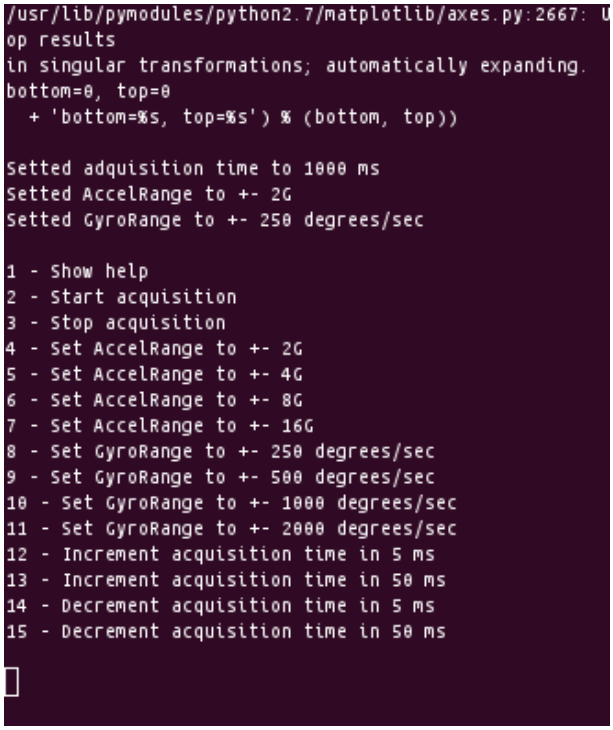

Figura 1: Captura del menú

Primer és recomanable escollir les opcions de de rang de precisió i velocitat d'adquisició.
Si no s'escull res per defecte, l'acceleració estarà en un rang de precisió 2g; la velocitat angular tindrà un rang de precisió 250 o/sec i s'adquiriran les dades a una freqüència d'un segon.
Quan es vulgui es pot modificar qualsevol propietat amb el menú anteriorment esmentat.
En el moment d'adquirir les dades, es transformaran en unitats dels sistema internacional (*línies 187-198*).
En el cas de l'acceleració, són transformades en m/s2. En el cas de la velocitat angular són transformades en o/sec. Finalment, la temperatura es transformada a ºC.

Les dades s'envien separades per comes, com podem veure a continuació:

$$x - axisaccel, y - axisaccel, z - axisaccel, x - axisgyro, y - axisgyro, z - axisgyro, temperature$$

Posteriorment el codi python llegeix les dades i les tracta. Ho explicarem en detall en el següent apartat.
A continuació deixem unes quantes captures:

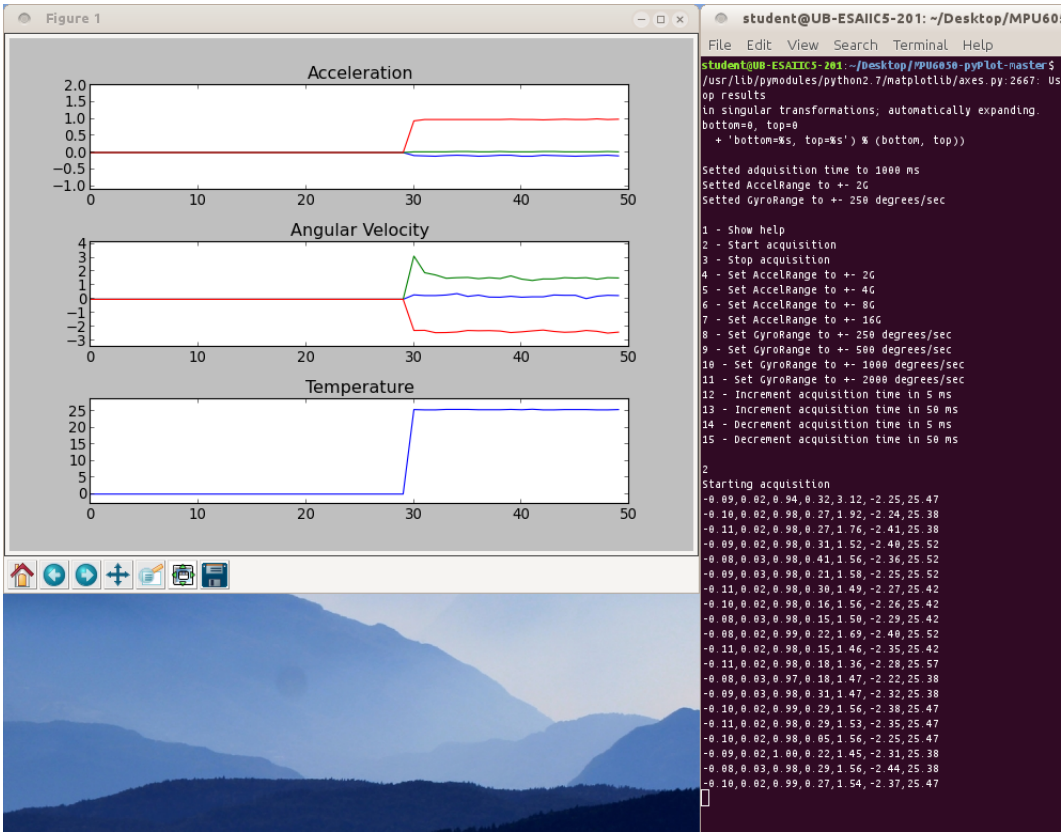

Figura 2: Iniciem l'adquisició de dades

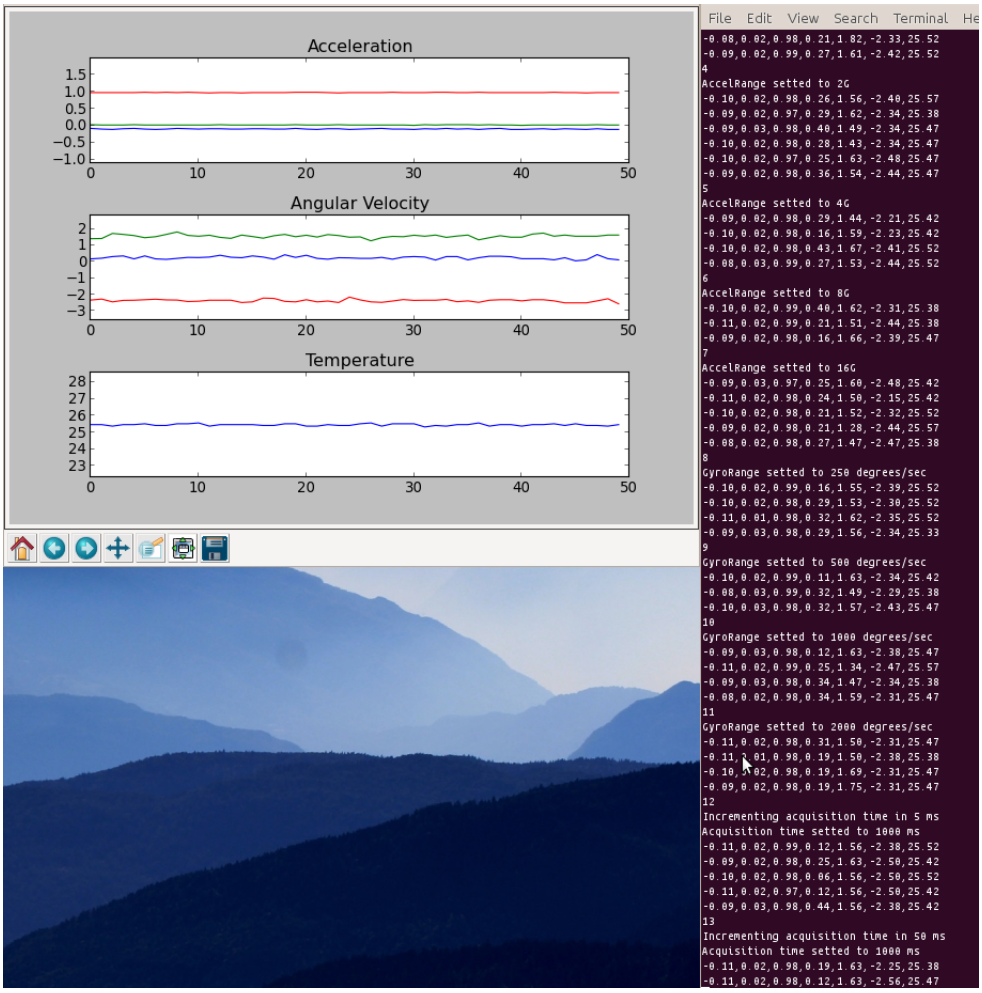

Figura 3: Canviem l'escala

3 Monitorització de la dades

3.1 Funcionament del codi python

Inicialitzem el port serie (per defecte és "/dev/ttyACM0")(línia 8)

```
serialPort = '/dev/ttyACM0'
baudRate = 9600
```

Inicialitzem el plot (*línia 13*).

```
plt.ion() # set plot to animated
```

Declarem arrays de 50 elements per les acceleracions, les velocitats angulars i per la temperatura. (*línies 14 - 24*).

```
xAccelData = [0] * 50
yAccelData = [0] * 50
zAccelData = [0] * 50

xAnVelData = [0] * 50
yAnVelData = [0] * 50
zAnVelData = [0] * 50

tempData = [0] * 50
```

Iniciem els "plotsï assignem els arrays que es mostraran en cada gràfica (*línies 27-42*).

```
plt.figure(1).subplots_adjust(hspace = .5)

plt.subplot(311).set_title("Acceleration")
xAccel, = plt.plot(xAccelData)
yAccel, = plt.plot(yAccelData)
zAccel, = plt.plot(zAccelData)

plt.subplot(312).set_title("Angular Velocity")
xAnVel, = plt.plot(xAnVelData)
yAnVel, = plt.plot(yAnVelData)
zAnVel, = plt.plot(zAnVelData)

plt.subplot(313).set_title("Temperature")
temp, = plt.plot(tempData)

plt.ylim([0,0])
```

Dins del *while True* (*línia 43*) tenim la part on recollim les dades adquirides per l'Arduino UNO i les rebem com s'ha descrit en la secció anterior. Per tant, fem "split"de les dades i les col·loquem als arrays corresponents (*línies 47-64*).

```
plt.figure(1).subplots_adjust(hspace = .5)

plt.subplot(311).set_title("Acceleration")
xAccel, = plt.plot(xAccelData)
yAccel, = plt.plot(yAccelData)
zAccel, = plt.plot(zAccelData)

plt.subplot(312).set_title("Angular Velocity")
xAnVel, = plt.plot(xAnVelData)
yAnVel, = plt.plot(yAnVelData)
zAnVel, = plt.plot(zAnVelData)

plt.subplot(313).set_title("Temperature")
temp, = plt.plot(tempData)

plt.ylim([0,0])
```

Re-calculem els màxims i mínims dels plots que es mostraran per pantalla (*línies 81-90*).

```
        min (min (xAccelData), min (yAccelData), min (zAccelData)) - 1, \
        max (max (xAccelData), max (yAccelData), max (zAccelData)) + 1])

    plt.subplot(312)
    plt.ylim([\
        min (min (xAnVelData), min (yAnVelData), min (zAnVelData)) - 1, \
        max (max (xAnVelData), max (yAnVelData), max (zAnVelData)) + 1])
```

```
8
9       plt.subplot(313)
10      plt.ylim([min (tempData)-3,max (tempData)+3])
```

Finalment eliminem el primer element dels arrays (*línies 92-100*).

```
1       del xAccelData [0]
2       del yAccelData [0]
3       del zAccelData [0]
4
5       del xAnVelData [0]
6       del yAnVelData [0]
7       del zAnVelData [0]
8
9       del tempData [0]
```

3.2 Resultat final del mostreig per pantalla

Un cop s'executem el programa python, com a **super-usuari**, es mostren tres gràfiques:

1. La gràfica superior on es mostren les tres acceleracions.
2. La gràfica central es mostra les tres velocitats angulars.
3. La gràfica inferior mostra la temperatura.

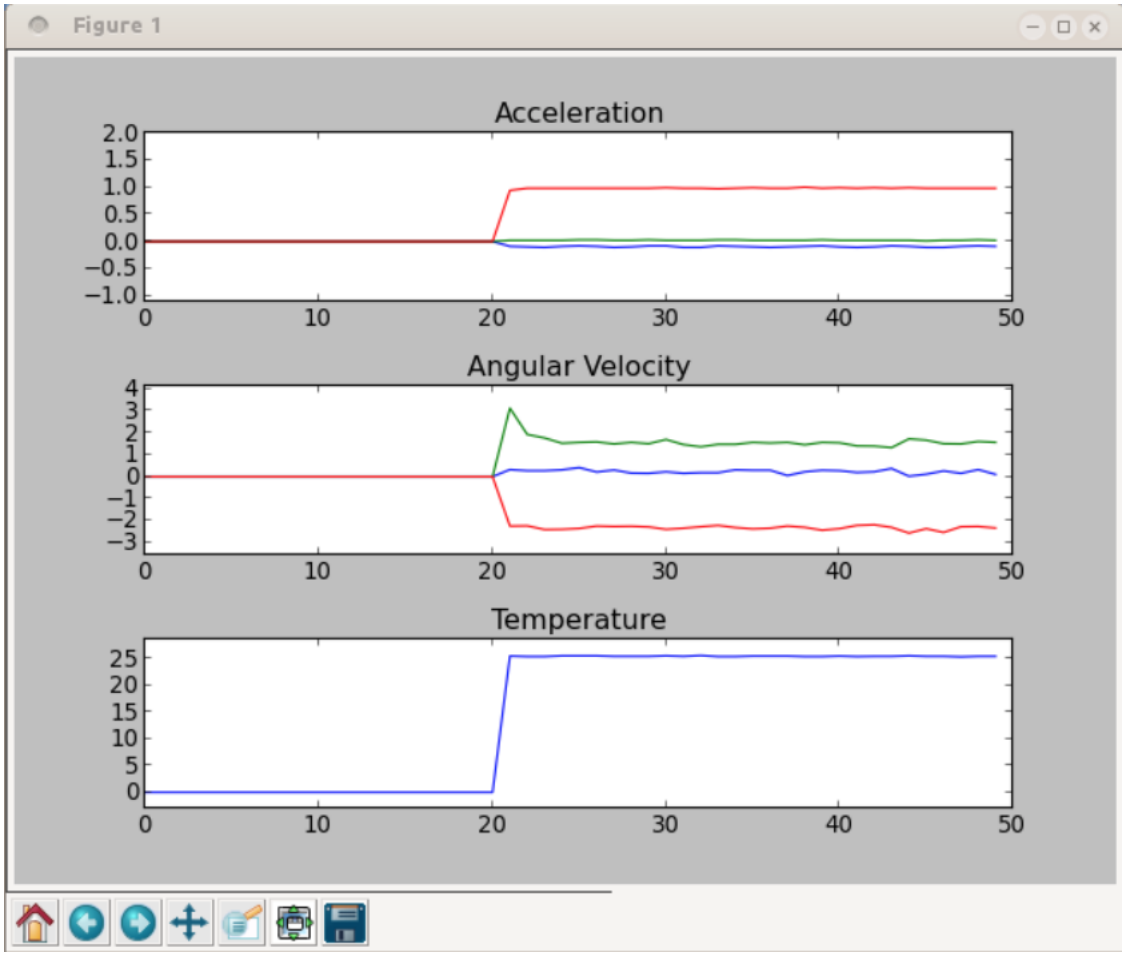

Figura 4: Captura de les tres gràfiques

Cada gràfica es re-calcula quan s'obté un màxim o un mínim superior a anteriors adquisicions, com s'ha explicat anteriorment.

El nostre codi el podreu trobar en el repositori següent:
https://github.com/marcfilba/MPU6050-pyPlot

4 Annex

4.1 Codi Arduino

```
#include "I2Cdev.h"
#include "MPU6050.h"

/*
 * #define MPU6050_ACCEL_FS_2          0x00
 * #define MPU6050_ACCEL_FS_4          0x01
 * #define MPU6050_ACCEL_FS_8          0x02
 * #define MPU6050_ACCEL_FS_16         0x03
 *
 * #define MPU6050_GYRO_FS_250         0x00
 * #define MPU6050_GYRO_FS_500         0x01
 * #define MPU6050_GYRO_FS_1000        0x02
 * #define MPU6050_GYRO_FS_2000        0x03
 */

#define MPU6050_ACCEL_FS_2_EQUIV    16384
#define MPU6050_ACCEL_FS_4_EQUIV    8192
#define MPU6050_ACCEL_FS_8_EQUIV    4096
#define MPU6050_ACCEL_FS_16_EQUIV   2048

#define MPU6050_GYRO_FS_250_EQUIV   131
#define MPU6050_GYRO_FS_500_EQUIV   65,5
#define MPU6050_GYRO_FS_1000_EQUIV  32,8
#define MPU6050_GYRO_FS_2000_EQUIV  16,4

// class default I2C address is 0x68
MPU6050 accelgyro;

String inString = "";

int acquisitionTime = 1000;

int equivAccel = MPU6050_ACCEL_FS_2_EQUIV;
int equivGyro = MPU6050_GYRO_FS_250_EQUIV;

bool acquisitionActivated = false;

void showHelp (){
  Serial.println ("1 - Show help");
  Serial.println ("2 - Start acquisition");
  Serial.println ("3 - Stop acquisition");

  Serial.println ("4 - Set AccelRange to +- 2G");
  Serial.println ("5 - Set AccelRange to +- 4G");
  Serial.println ("6 - Set AccelRange to +- 8G");
  Serial.println ("7 - Set AccelRange to +- 16G");

  Serial.println ("8 - Set GyroRange to +- 250 degrees/sec");
  Serial.println ("9 - Set GyroRange to +- 500 degrees/sec");
  Serial.println ("10 - Set GyroRange to +- 1000 degrees/sec");
  Serial.println ("11 - Set GyroRange to +- 2000 degrees/sec");

  Serial.println ("12 - Increment acquisition time in 5 ms");
  Serial.println ("13 - Increment acquisition time in 50 ms");
  Serial.println ("14 - Decrement acquisition time in 5 ms");
  Serial.println ("15 - Decrement acquisition time in 50 ms\n");
}

int getOption (){
  int option = -1;
  while (Serial.available() > 0){
    int inChar = Serial.read();
    if (isDigit(inChar))  inString += (char)inChar;

    if (inChar == '\n') {
      option = inString.toInt();
      inString = "";
    }
  }
```

```
  return option;
}

void handleOption (){
  switch (getOption ()){
    case -1: break;
    case 1:{
      showHelp();
    break;
    }
    case 2:{
      acquisitionActivated = true;
      Serial.println("Starting acquisition");
      break;
     }
     case 3:{
      acquisitionActivated = false;
      Serial.println("Stoping acquisition");  ;
      break;
     }
    case 4:{
      accelgyro.setFullScaleAccelRange(MPU6050_ACCEL_FS_2);
      Serial.println("AccelRange setted to 2G");
      equivAccel = MPU6050_ACCEL_FS_2_EQUIV;
      break;
     }
    case 5:{
      accelgyro.setFullScaleAccelRange(MPU6050_ACCEL_FS_4);
      Serial.println("AccelRange setted to 4G");
      equivAccel = MPU6050_ACCEL_FS_4_EQUIV;
      break;
    }
    case 6:{ accelgyro.setFullScaleAccelRange(MPU6050_ACCEL_FS_8);
      Serial.println("AccelRange setted to 8G");
      equivAccel = MPU6050_ACCEL_FS_8_EQUIV;
      break;
    }
    case 7:{
      accelgyro.setFullScaleAccelRange(MPU6050_ACCEL_FS_16);
      Serial.println("AccelRange setted to 16G");
      equivAccel = MPU6050_ACCEL_FS_16_EQUIV;
      break;
    }
    case 8:{
      accelgyro.setFullScaleGyroRange(MPU6050_GYRO_FS_250);
      Serial.println("GyroRange setted to 250 degrees/sec");
      equivGyro = MPU6050_GYRO_FS_250_EQUIV;
      break;
    }
    case 9:{
      accelgyro.setFullScaleGyroRange(MPU6050_GYRO_FS_500);
      Serial.println("GyroRange setted to 500 degrees/sec");
      equivGyro = MPU6050_GYRO_FS_500_EQUIV;
      break;
    }
    case 10:{
      accelgyro.setFullScaleGyroRange(MPU6050_GYRO_FS_1000);
      Serial.println("GyroRange setted to 1000 degrees/sec");
      equivGyro = MPU6050_GYRO_FS_1000_EQUIV;
      break;
    }
    case 11:{
      accelgyro.setFullScaleGyroRange(MPU6050_GYRO_FS_2000);
      Serial.println("GyroRange setted to 2000 degrees/sec");
      equivGyro = MPU6050_GYRO_FS_2000_EQUIV;
      break;
    }
    case 12:{
      Serial.println ("Incrementing acquisition time in 5 ms");
      acquisitionTime = min (acquisitionTime + 5, 1000);
      Serial.print ("Acquisition time setted to "); Serial.print (acquisitionTime);
    Serial.println (" ms");
      break;
```

```
142     }
143     case 13:{
144       Serial.println ("Incrementing acquisition time in 50 ms");
145       acquisitionTime = min (acquisitionTime + 50, 1000);
146       Serial.print ("Acquisition time setted to "); Serial.print (acquisitionTime);
      Serial.println (" ms");
147       break;
148     }
149     case 14:{
150       Serial.println("Decrementing acquisition time in 5 ms");
151       acquisitionTime = max (acquisitionTime - 5, 5);
152       Serial.print ("Acquisition time setted to "); Serial.print (acquisitionTime);
      Serial.println (" ms");
153       break;
154     }
155     case 15:{
156       Serial.println("Decrementing acquisition time in 50 ms");
157       acquisitionTime = max (acquisitionTime - 50, 5);
158       Serial.print ("Acquisition time setted to "); Serial.print (acquisitionTime);
      Serial.println (" mz");
159       break;
160     }
161     default:{
162       Serial.println ("Option not implemented");
163       break;
164     }
165   }
166 }
167 void setup(){
168
169   #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
170     Wire.begin();
171   #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
172     Fastwire::setup(400, true);
173   #endif
174
175   Serial.begin(9600);
176
177   accelgyro.initialize();
178   Serial.println ("Setted adquisition time to 1000 ms");
179   Serial.println ("Setted AccelRange to +- 2G");
180   Serial.println ("Setted GyroRange to +- 250 degrees/sec\n");
181
182   showHelp ();
183 }
184
185 void loop(){
186
187   if (acquisitionActivated){
188     Serial.print(float (accelgyro.getAccelerationX()) / equivAccel) * 9.8;  Serial.print
      (",");
189     Serial.print(float (accelgyro.getAccelerationY()) / equivAccel) * 9.8;  Serial.print
      (",");
190     Serial.print(float (accelgyro.getAccelerationZ()) / equivAccel) * 9.8;  Serial.print
      (",");
191
192     Serial.print(float (accelgyro.getRotationX()) / equivGyro);   Serial.print (",");
193     Serial.print(float (accelgyro.getRotationY()) / equivGyro);   Serial.print (",");
194     Serial.print(float (accelgyro.getRotationZ()) / equivGyro);   Serial.print (",");
195
196     Serial.println(accelgyro.getTemperature() / 340.00 + 36.53);
197     delay(acquisitionTime);
198   }
199
200   handleOption ();
201 }
```

4.2 Codi Python

```
import serial
import select
import numpy as np

from sys import stdin
from matplotlib import pyplot as plt

serialPort = '/dev/ttyACM0'
baudRate = 9600

ser = serial.Serial(serialPort, baudRate)

plt.ion() # set plot to animated

xAccelData = [0] * 50
yAccelData = [0] * 50
zAccelData = [0] * 50

xAnVelData = [0] * 50
yAnVelData = [0] * 50
zAnVelData = [0] * 50

tempData = [0] * 50

ax1 = plt.axes()

plt.figure(1).subplots_adjust(hspace = .5)

plt.subplot(311).set_title("Acceleration")
xAccel, = plt.plot(xAccelData)
yAccel, = plt.plot(yAccelData)
zAccel, = plt.plot(zAccelData)

plt.subplot(312).set_title("Angular Velocity")
xAnVel, = plt.plot(xAnVelData)
yAnVel, = plt.plot(yAnVelData)
zAnVel, = plt.plot(zAnVelData)

plt.subplot(313).set_title("Temperature")
temp, = plt.plot(tempData)

plt.ylim([0,0])


rawData = 'AccelRange'
rawDataSplitted = []

read = False

print ''

while True:

  if select.select([ser,],[],[],0.0)[0]:
    read = True
    rawData = ser.readline().rstrip() # llegim accel, gyro i temp
    rawDataSplitted = rawData.split (",")

    #if len (rawDataSplitted) != 7:
    print rawData
  else:
    read = False

  while select.select([stdin,],[],[],0.0)[0]:
    ser.write(stdin.readline())

  if len(rawDataSplitted) == 7 and read:

    xAccelData.append(float (str(rawDataSplitted[0])))
    yAccelData.append(float (str(rawDataSplitted[1])))
    zAccelData.append(float (str(rawDataSplitted[2])))
```

```
72
73      xAnVelData.append(float (str(rawDataSplitted[3])))
74      yAnVelData.append(float (str(rawDataSplitted[4])))
75      zAnVelData.append(float (str(rawDataSplitted[5])))
76
77      tempData.append (float (str(rawDataSplitted[6])))
78
79      plt.subplot(311)
80      plt.ylim([\
81          min (min (xAccelData), min (yAccelData), min (zAccelData)) - 1, \
82          max (max (xAccelData), max (yAccelData), max (zAccelData)) + 1])
83
84      plt.subplot(312)
85      plt.ylim([\
86          min (min (xAnVelData), min (yAnVelData), min (zAnVelData)) - 1, \
87          max (max (xAnVelData), max (yAnVelData), max (zAnVelData)) + 1])
88
89      plt.subplot(313)
90      plt.ylim([min (tempData)-3,max (tempData)+3])
91
92      del xAccelData [0]
93      del yAccelData [0]
94      del zAccelData [0]
95
96      del xAnVelData [0]
97      del yAnVelData [0]
98      del zAnVelData [0]
99
100     del tempData [0]
101
102     xAccel.set_xdata(np.arange(len(xAccelData)))
103     xAccel.set_ydata(xAccelData)
104
105     yAccel.set_xdata(np.arange(len(yAccelData)))
106     yAccel.set_ydata(yAccelData)
107
108     zAccel.set_xdata(np.arange(len(zAccelData)))
109     zAccel.set_ydata(zAccelData)
110
111
112     xAnVel.set_xdata(np.arange(len(xAnVelData)))
113     xAnVel.set_ydata(xAnVelData)
114
115     yAnVel.set_xdata(np.arange(len(yAnVelData)))
116     yAnVel.set_ydata(yAnVelData)
117
118     zAnVel.set_xdata(np.arange(len(zAnVelData)))
119     zAnVel.set_ydata(zAnVelData)
120
121     temp.set_xdata(np.arange(len(tempData)))
122     temp.set_ydata(tempData)
123
124     plt.draw()
```