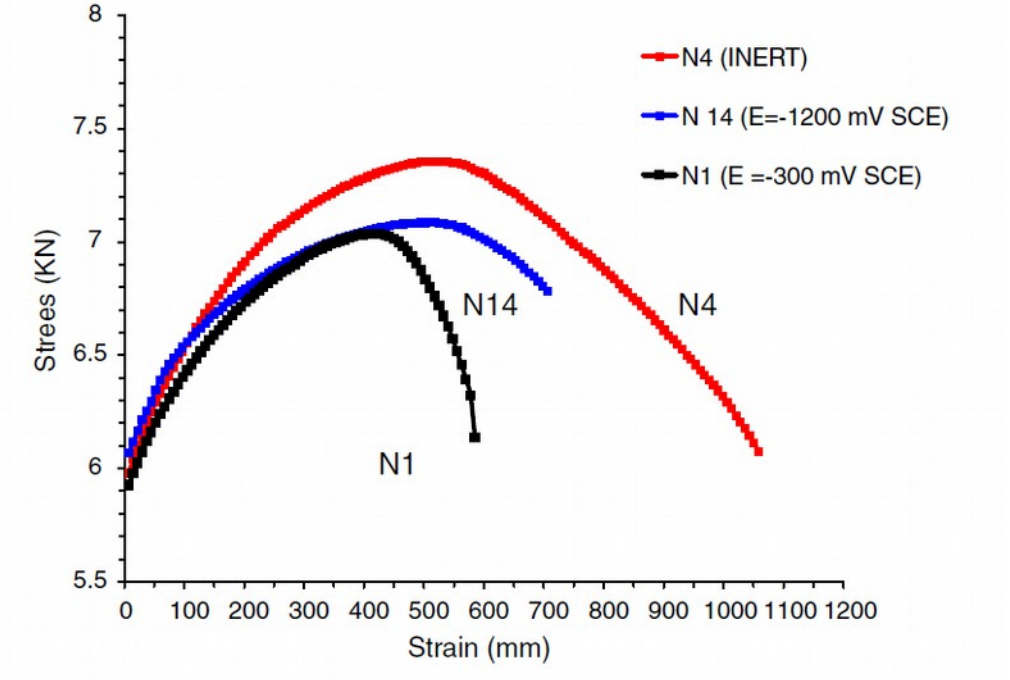
Comment économiser 10 000$ avec un Raspberry Pi

Survol

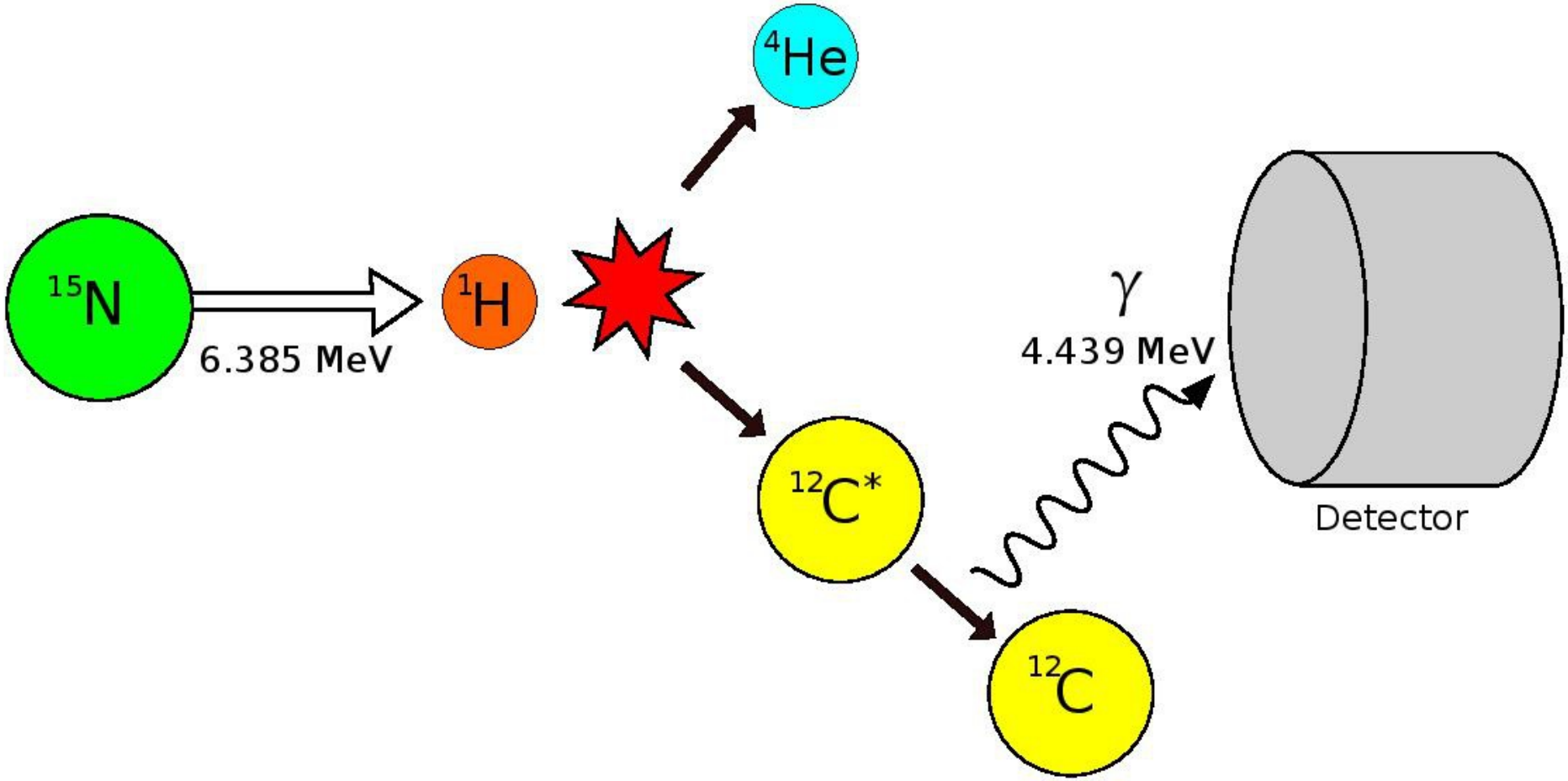
- Physique
  - NRA
  - Acier
  - CRIAQ
- Hardware
  - Arduino
  - RP 3
  - Le reste
- Software
  - Python 2.7
  - Rpi.GPIO
  - Pyserial
  - Matplotlib
  - Numpy
  - Subprocess
  - Multiprocess
  - Threading

La motivation physique

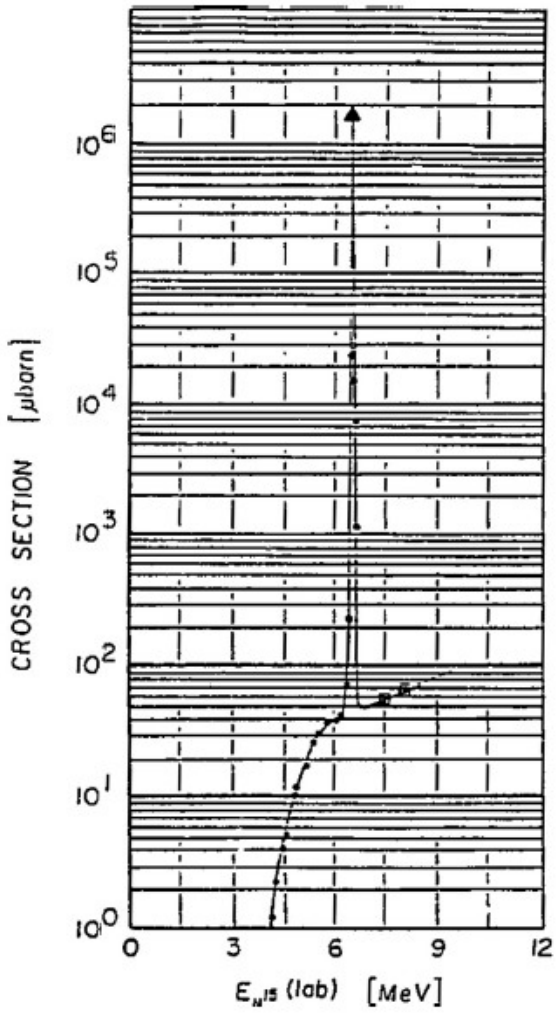
Acier 4340

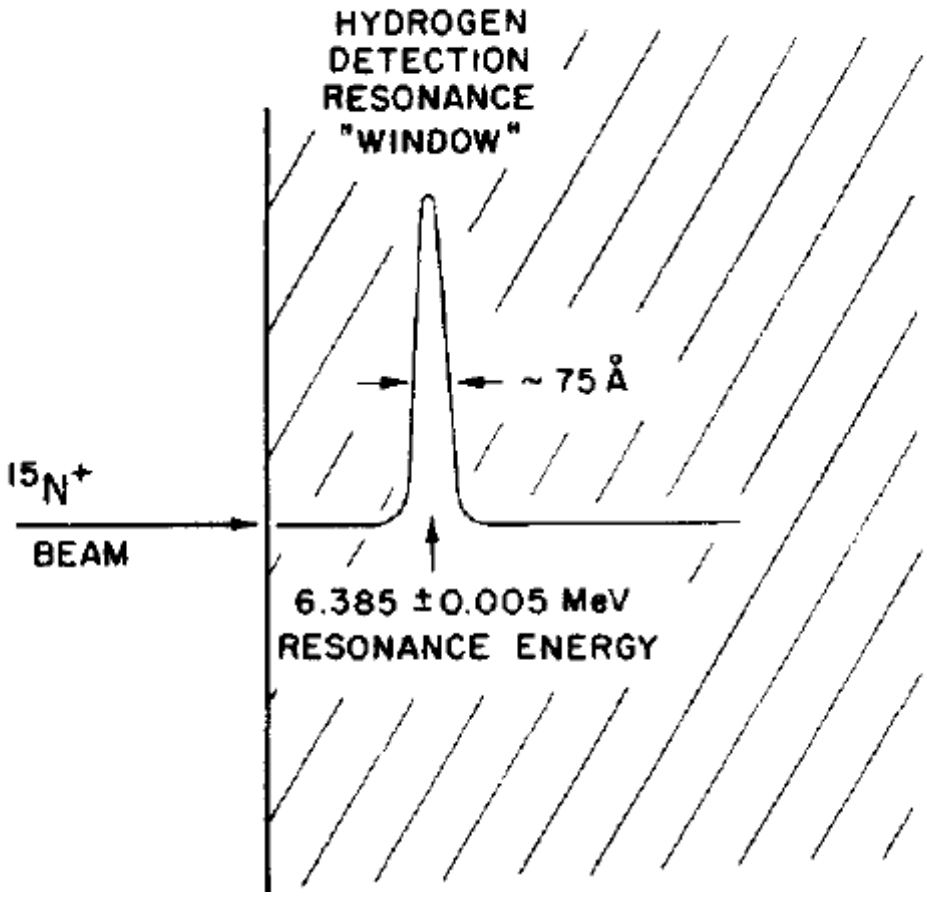


La motivation physique

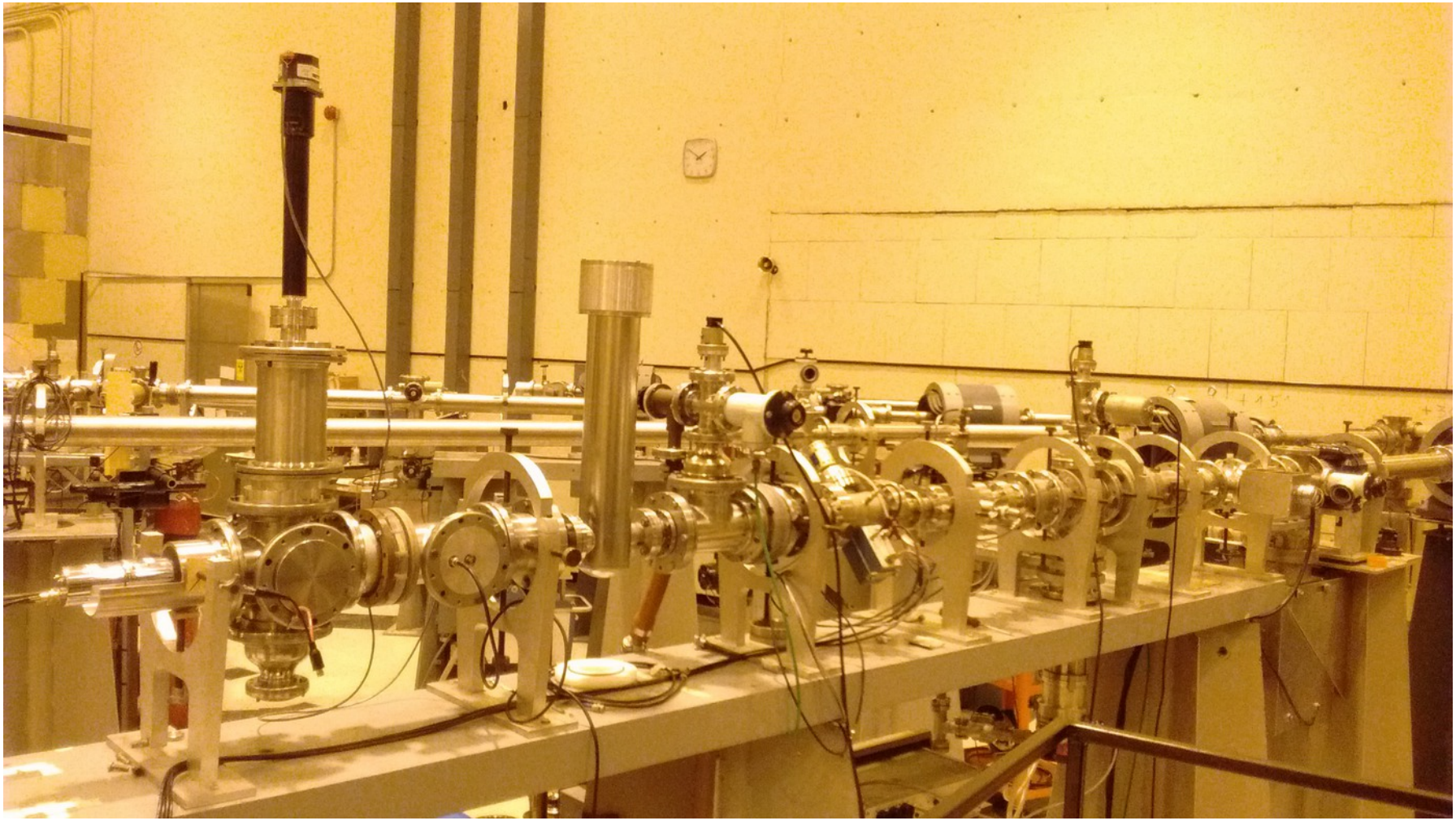


La motivation physique

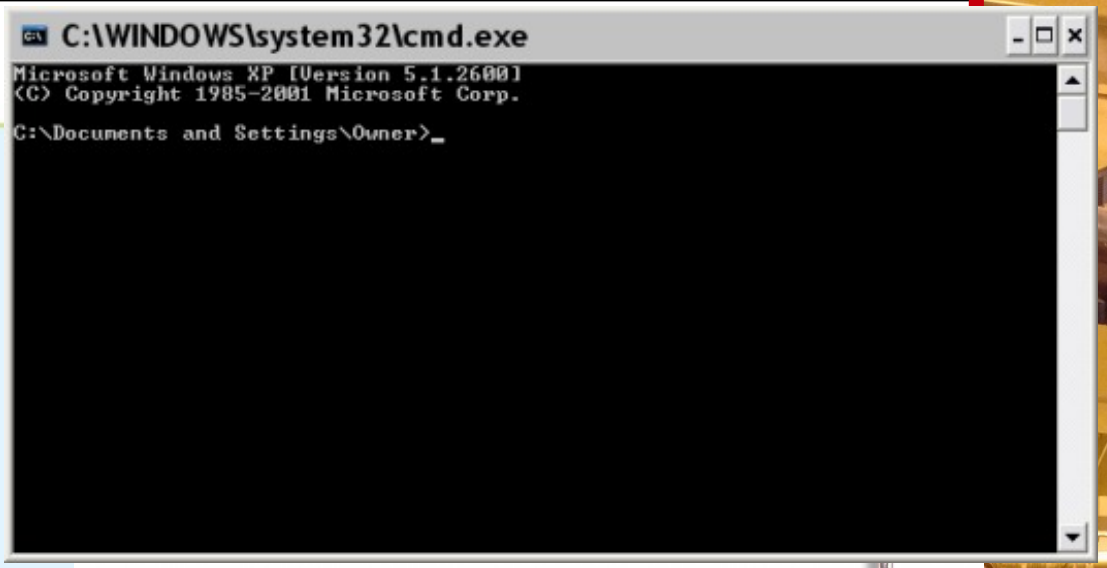




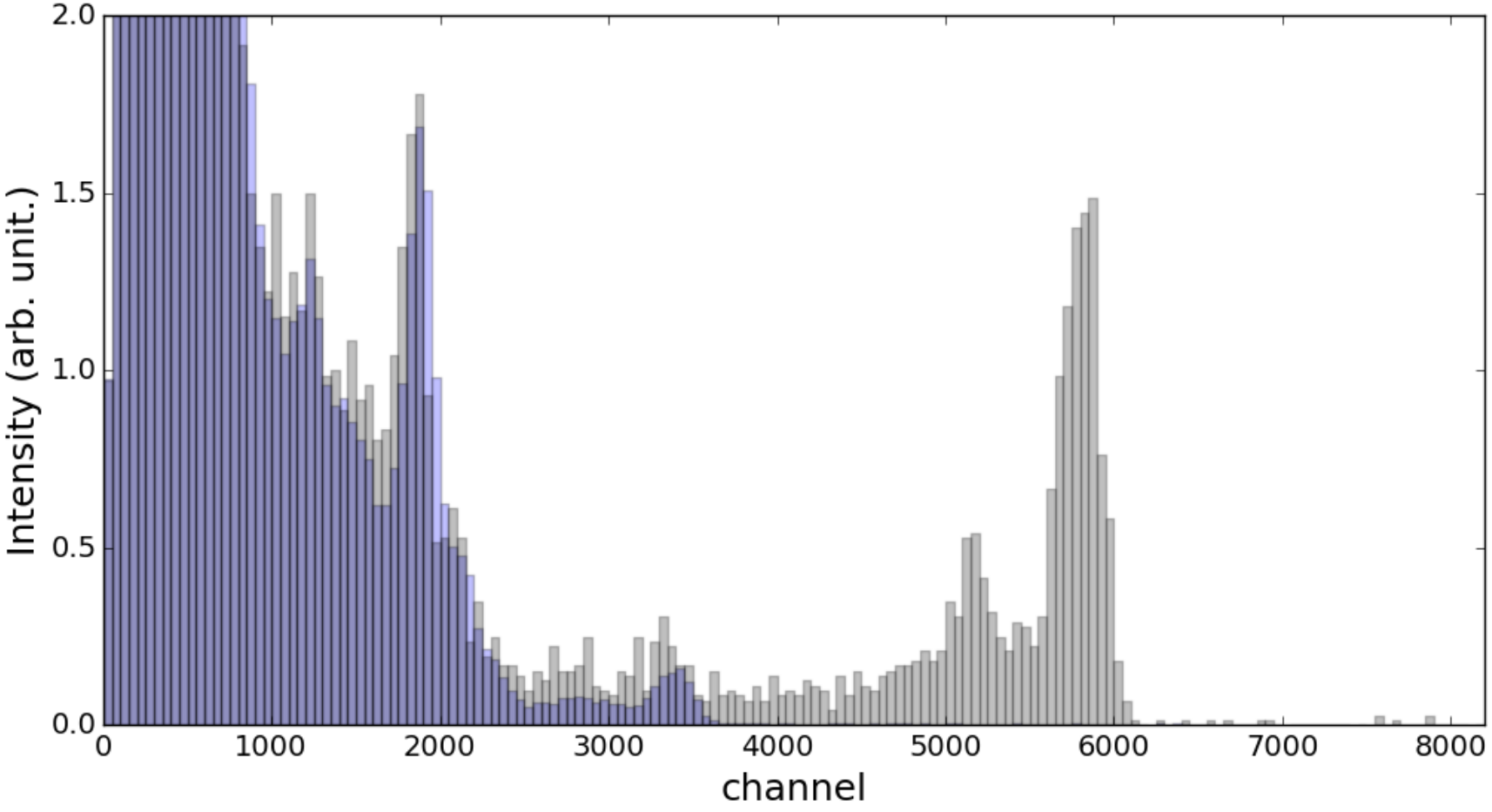
La motivation hardware



La motivation hardware

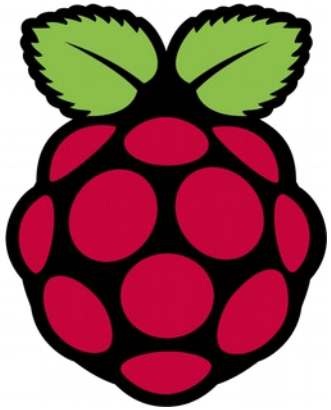


La motivation hardware

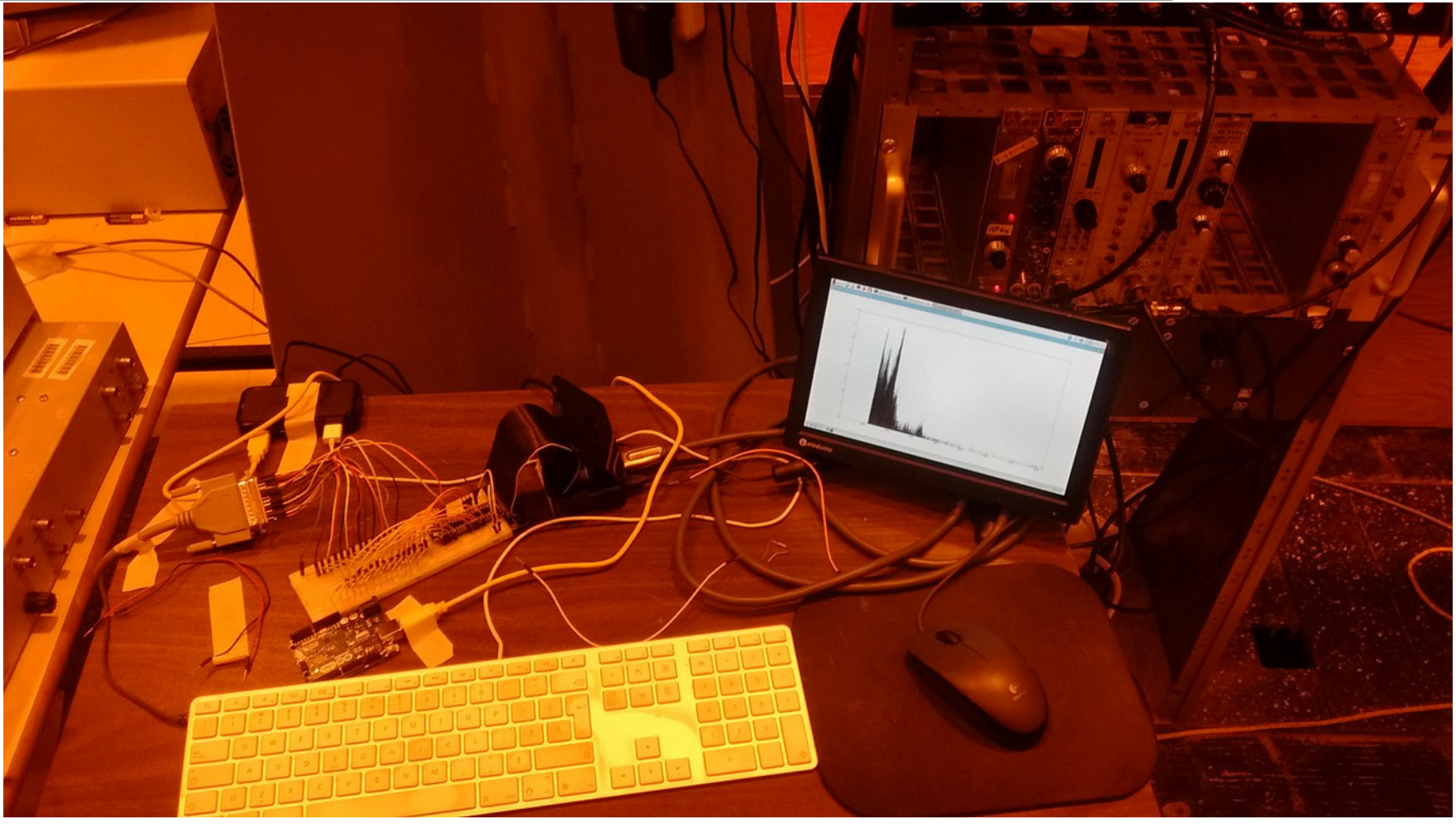


La motivation hardware

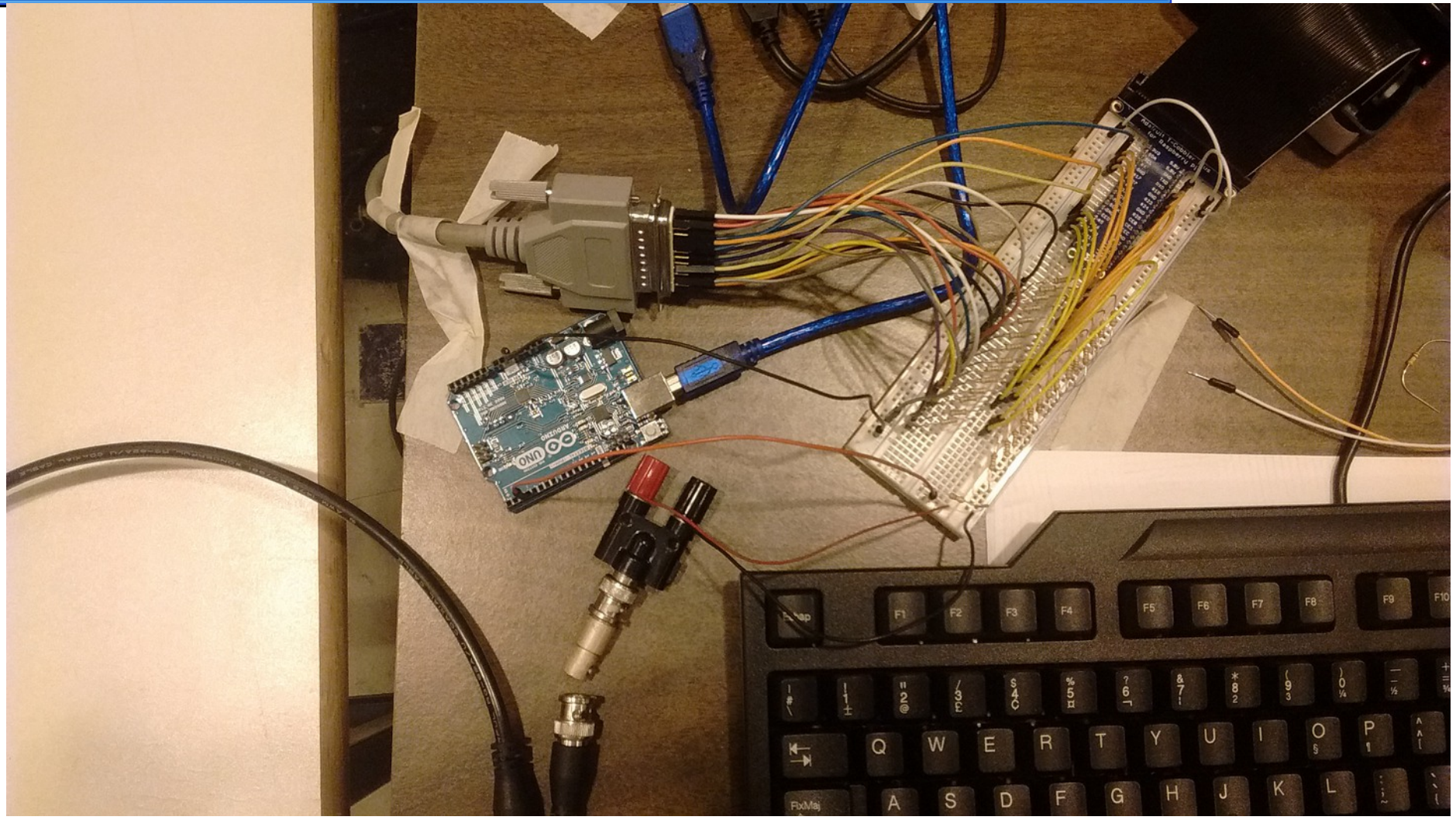
ARDUINO



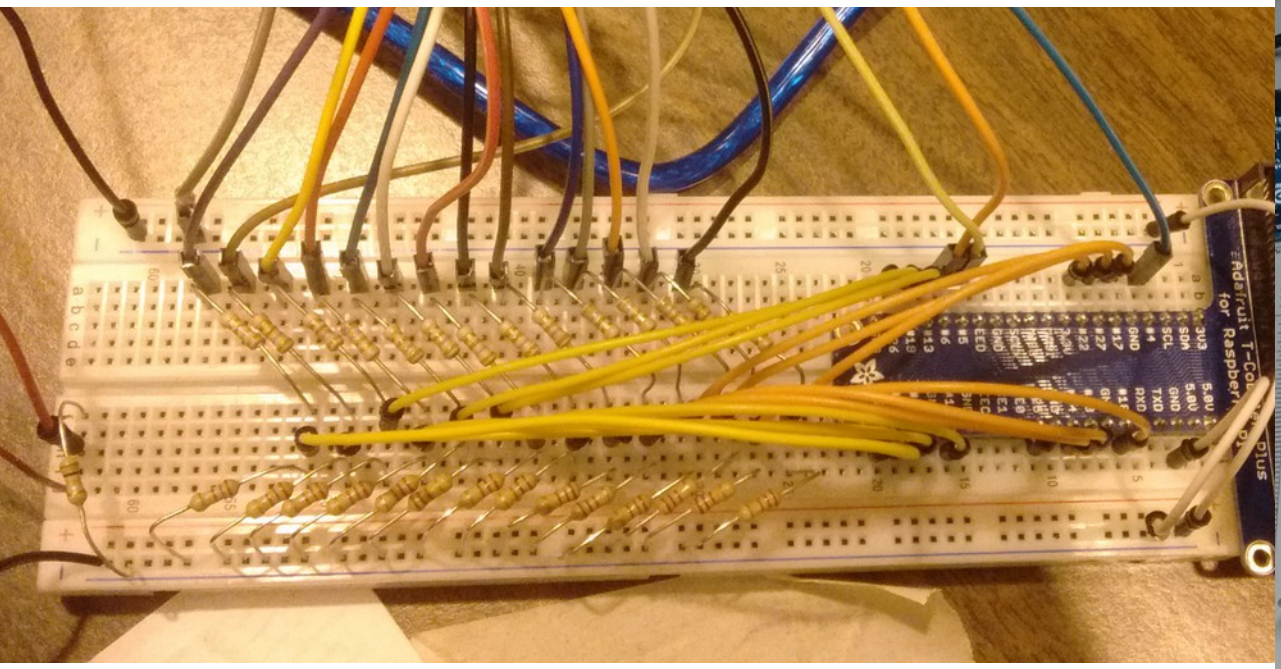
La solution

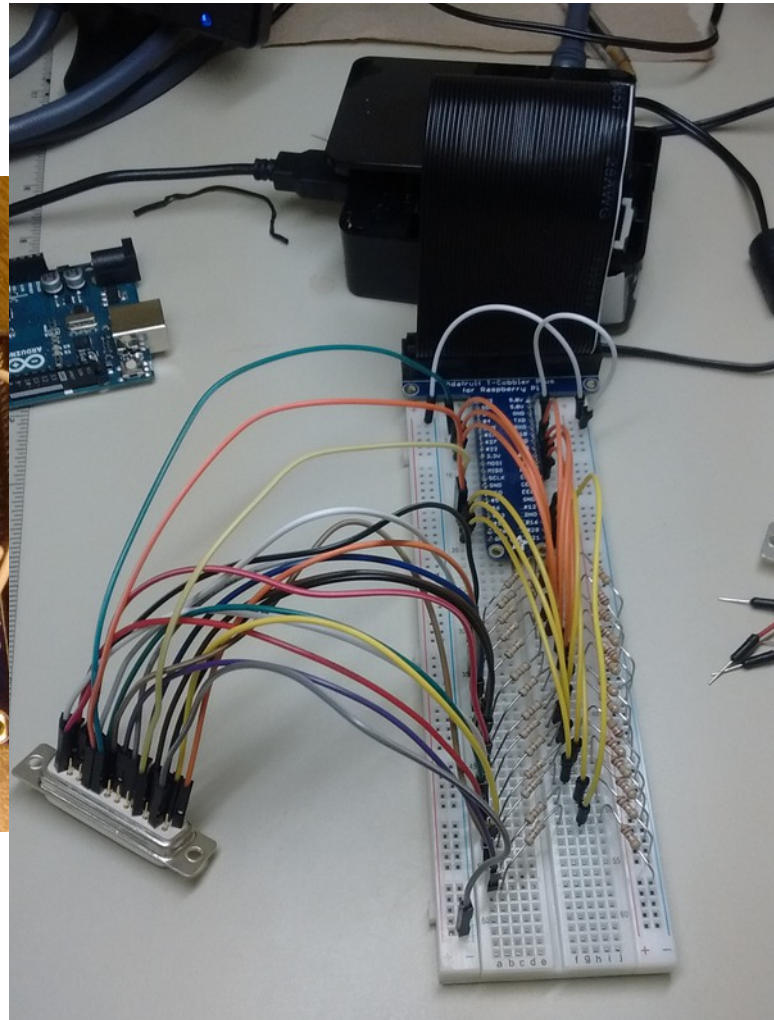


La solution

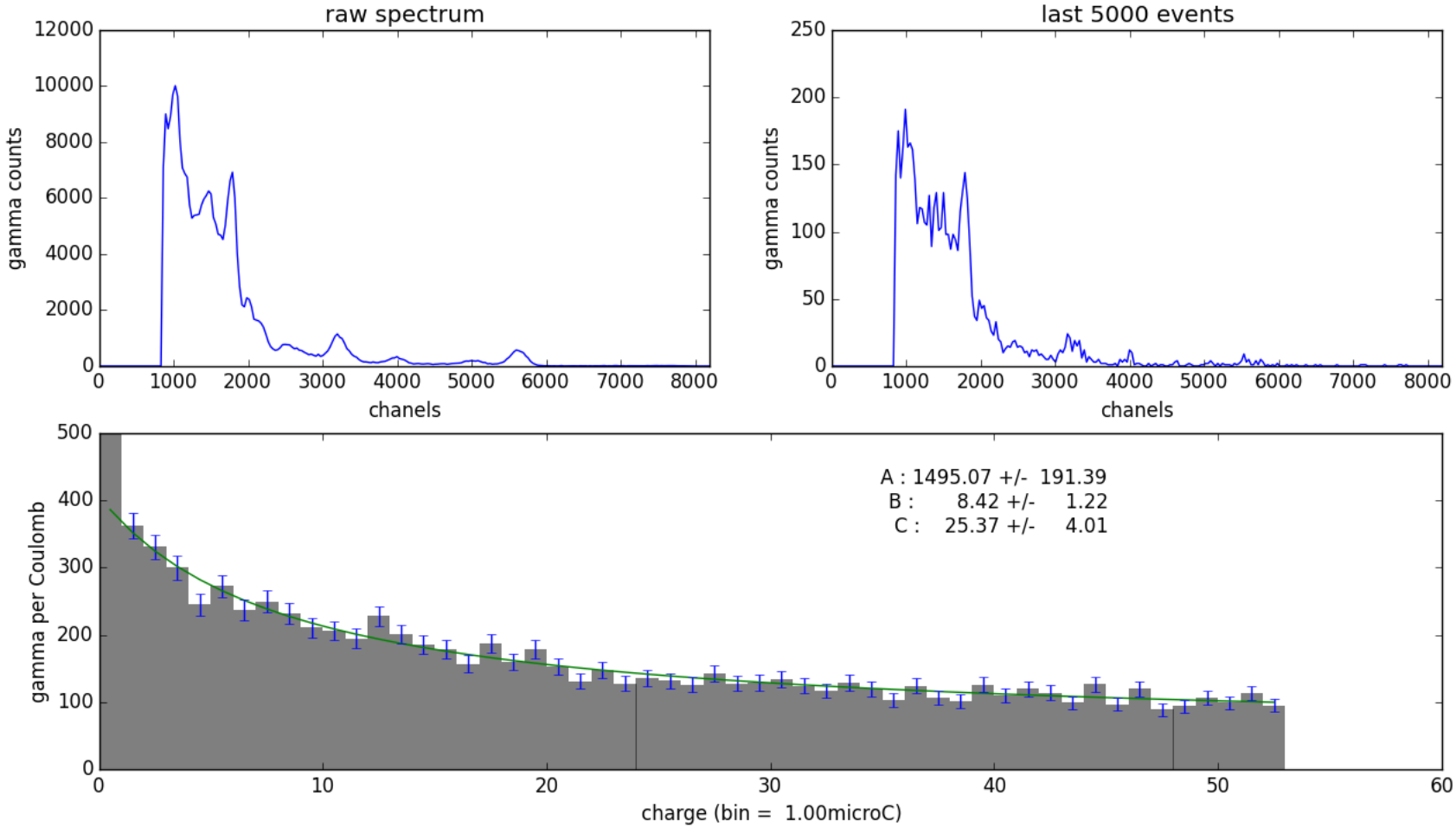


La solution





La solution


raw spectrum
gamma counts
chanels
last 5000 events
gamma counts
chanels
A : 1495.07 +/- 191.39
B : 8.42 +/- 1.22
C : 25.37 +/- 4.01
gamma per Coulomb
charge (bin = 1.00microC)

Le software

```
#*-* coding: utf-8 *-*
##########################################################################################
##########################################################################################
#                                %
# Author  : Alexandre Desilets-Benoit, Ph.D. Phys.
# Institution : Université de Montréal
# Group   : Solid State Physics (Accelerator subgroup with Prof. Sjoerd Roorda)
# Year    : 2016
# Version     : 2.0
# function    : MCA communicating between ORTEC ADC and raspberry pi on NRA set-up
#
##########################################################################################
##########################################################################################


import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use("TKAgg")
import serial as sr
import subprocess
import RPi.GPIO as GPIO
import matplotlib.animation as animation
import multiprocessing
import time
from datetime import datetime
import threading

```

Le software

```
40 try:
41     print "Trying to connect to %s"%'/dev/ttyACM0'
42     arduino = sr.Serial('/dev/ttyACM0',9600)
43     print "Connected to port %s"%'/dev/ttyACM0'
44 except:
45     print "error in connecting to port %s"%'/dev/ttyACM0'
```

Le software

```
try:
    print "Trying to connect to %
    arduino = sr.Serial('/dev/tty
    print "Connected to port %s"%
except:
    print "error in connecting to
```

```
class ArduinoFunc():
    def __init__(self,port,baudrate,delais,charge):
        self.port = port
        self.baudrate = baudrate
        self.delais = delais
        self.charge = charge
        #print "connecting to arduino"
        try:
            print "Trying to connect to %s"%self.port
            self.arduino = sr.Serial(self.port,self.baudrate)
            print "Connected to port %s"%self.port
        except:
            print "error in connecting to port %s"%self.port
        print "Emptying arduino cash"
        self.arduino.write('z')
        print 'wait 5 seconds'
        time.sleep(5)
        test = self.arduino.readline().replace('\n','').replace('\r','')
        while test != 'k':
            print 'empty arduino cache'
            self.arduino.write('z')
            test = self.arduino.readline().replace('\n','').replace('\r','')
        print "Emptied arduino cash\n"
    def Read(self):
        self.arduino.write('k')
        Charge = self.arduino.readline().replace('\n','').replace('\r','')
        while Charge.isdigit() == False:
            Charge = self.arduino.readline().replace('\n','').replace('\r','')
        self.charge.value = int(Charge)
```

Le software

Board	Digital Pins Usable For Interrupts
Uno, Nano, Mini, other 328-based	2, 3
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21
Micro, Leonardo, other 32u4-based	0, 1, 2, 3, 7
Zero	all digital pins, except 4
MKR1000 Rev.1	0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Due	all digital pins
101	all digital pins

```
// Variable pour l'intégrateur de courant
volatile boolean Flag = false;
volatile unsigned long count = 0;

void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);
  pinMode(13, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(2), COUNT, RISING);
}

void loop() {
  if (Serial.available()) {
    int c = Serial.read();
    if (c == 'k') {
      Serial.println(count);
    }
    if (c == 'z') {
      count = 0;
      Serial.println('k');
    }
  }
}

void COUNT() {
  count=count + 1;
}
```

Le software

```
40 try:
41     print "Trying to connect to %s"%'/dev/ttyACM0'
42     arduino = sr.Serial('/dev/ttyACM0',9600)
43     print "Connected to port %s"%'/dev/ttyACM0'
44 except:
45     print "error in connecting to port %s"%'/dev/ttyACM0'
```

```
103 class ArduinoFunc():
104     def __init__(self,port,baudrate,delais,charge):
105         self.port = port
106         self.baudrate = baudrate
107         self.delais = delais
108         self.charge = charge
109         #print "connecting to arduino"
110         try:
111             print "Trying to connect to %s"%self.port
112             self.arduino = sr.Serial(self.port,self.baudrate)
113             print "Connected to port %s"%self.port
114         except:
115             print "error in connecting to port %s"%self.port
116         print "Emptying arduino cash"
117         self.arduino.write('z')
118         print 'wait 5 seconds'
119         time.sleep(5)
120         test = self.arduino.readline().replace('\n','').replace('\r','')
121         while test != 'k':
122             print 'empty arduino cache'
123             self.arduino.write('z')
124             test = self.arduino.readline().replace('\n','').replace('\r','')
125         print "Emptied arduino cash\n"
```

```
249 def SecondProcess(e,charge):
250     i = 0
251     BAUDRATE = 9600
252     PORT = '/dev/ttyACM0'
253     arduino = ArduinoFunc(PORT,BAUDRATE,1e-4,charge)
254     while e.is_set() != True:
255         arduino.Read()
256     print "Close Arduino"
257     return True
```

```
dline().replace('\n','').replace('\r','')
 False:
.readline().replace('\n','').replace('\r','')
harge)
```

Le software

```
48 '''
49 GPIO04 = Start/stop
50 GPIO05 = Data resquest
51 GPIO06 = Data accept
52 GPIO12 = Data ready
53 GPIO13 = 2^0
54 GPIO16 = 2^1
55 GPIO17 = 2^2
56 GPIO18 = 2^3
57 GPIO19 = 2^4
58 GPIO20 = 2^5
59 GPIO21 = 2^6
60 GPIO22 = 2^7
61 GPIO23 = 2^8
62 GPIO24 = 2^9
63 GPIO25 = 2^10
64 GPIO26 = 2^11
65 GPIO27 = 2^12
66 NOTE : sur ADC_B Bin 2^3 ne fonctionne pas
67 NOTE2 : GPIO04 est bel et bien Start/Stop
68 NOTE3 : GPIO05, GPIO06 et GPIO12 semble bien fonctionner
69 NOTE4 : GPIO24 n'est pas 2^9, mais bien 2^3
70 NOTE5 : est-ce que l'erreur est de passer de 0123456789 a 987
71 '''
```

```
72 GPIO.setmode(GPIO.BCM)  #set board mode to Broadcom
73
74 GPIO.setup(4,GPIO.OUT)
75 GPIO.setup(5,GPIO.OUT)
76 GPIO.setup(6,GPIO.OUT)
77
78 GPIO.setup(12,GPIO.IN)
79 GPIO.setup(13,GPIO.IN)
80 GPIO.setup(16,GPIO.IN)
81 GPIO.setup(17,GPIO.IN)
82 GPIO.setup(18,GPIO.IN)
83 GPIO.setup(19,GPIO.IN)
84 GPIO.setup(20,GPIO.IN)
85 GPIO.setup(21,GPIO.IN)
86 GPIO.setup(22,GPIO.IN)
87 GPIO.setup(23,GPIO.IN)
88 GPIO.setup(24,GPIO.IN)
89 GPIO.setup(25,GPIO.IN)
90 GPIO.setup(26,GPIO.IN)
91 GPIO.setup(27,GPIO.IN)
92
93 GPIO.output(4,1) #Ouvert ou ferme??
94 GPIO.output(5,1)
95 GPIO.output(6,1)
```

Le résulta

```
133 class GPIOFunc():
134     def __init__(self,delais,Info):
135         Year,Month,Day,FileName,Hour,Minute,Seconds,SampleName,Position,Energy,Charge,Users,Current,Comment = Info
136         self.delais = delais
137         self.ValeurADC = 0
138         self.fichier = open('RawData/'+Year+'-'+Month+'-'+Day+'/'+FileName,'wa')
139         self.fichier.write('#####################################################'+'\n')
140         self.fichier.write('Date : '+Year+'-'+Month+'-'+Day+'\n')
141         self.fichier.write('Time of launch : '+Hour+':'+Minute+':'+Seconds+'\n')
142         self.fichier.write('Sample : '+SampleName+'\n')
143         self.fichier.write('Position : '+Position+'\n')
144         self.fichier.write('Energy : '+Energy+'\n')
145         self.fichier.write('Charge : '+Charge+'\n')
146         self.fichier.write('Users : '+Users+'\n')
147         self.fichier.write('Initial current : '+Current+'\n')
148         self.fichier.write('Comments : '+Comment+'\n')
149         self.fichier.write('#####################################################'+'\n')
150         self.fichier.write('%s,%s,%s'%('Time','Charge','AnalogValue')+'\n')
151     def Close(self):
152         self.fichier.close()
153     def Request(self):
154         GPIO.output(5,0)
155         time.sleep(self.delais)
156     def Accept(self):
157         GPIO.output(5,1)
158         GPIO.output(6,0)
159         time.sleep(Delais)
160         GPIO.output(6,1)
161     def ReadBin(self,):
162         Pins = [27,26,25,24,23,22,21,20,19,18,17,16,13]
163         try:Data2 = [str(GPIO.input(element)) for element in Pins]#[D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,D10,D11,D12]
164         except:
165             print "Error loading the data"
166             self.ReadBin()
167         DataOut = 8192 - int(''.join(Data2),2)
168         return DataOut
169     def ReadADC(self,Binaries):
170         self.Request()
171         Checking = True
172         while Checking:
173             try:
174                 Valeur = self.ReadBin()
175                 Checking = False
176             except:
177                 print "problems reading output from ADC : Retrying"
178         self.ValeurADC = Valeur
179         self.Accept()
```

Le software

```
153     def Request(self):
154         GPIO.output(5,0)
155         time.sleep(self.delais)
156     def Accept(self):
157         GPIO.output(5,1)
158         GPIO.output(6,0)
159         time.sleep(Delais)
160         GPIO.output(6,1)
161     def ReadBin(self,):
162         Pins = [27,26,25,24,23,22,21,20,19,18,17,16,13]
163         try:Data2 = [str(GPIO.input(element)) for element in Pins]#[D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,D10,D11,D12]
164         except:
165             print "Error loading the data"
166             self.ReadBin()
167         DataOut = 8192 - int(''.join(Data2),2)
168         return DataOut
169     def ReadADC(self,Binaries):
170         self.Request()
171         Checking = True
172         while Checking:
173             try:
174                 Valeur = self.ReadBin()
175                 Checking = False
176             except:
177                 print "problems reading output from ADC : Retrying"
178         self.ValeurADC = Valeur
179         self.Accept()
```

Le software

```
196 def Counting(e,data,Evolution,Last5000,charge,Informations):
197     time.sleep(5)
198     i = 0
199     j = 0
200     DC = 0.5*100000
201     DT = 600
202     Last = np.array([])
203     T0 = time.time()
204     GPIO.output(4,0) #Le buffer se remplir aussi tot que 4 = 0
205     GPIOADC = GPIOFunc(1e-5,Informations)
206     while e.is_set() != True:
207         if GPIO.input(12) == 0:
208             GPIOADC.ReadADC(1)
209             Ti = time.time()
210             Temps = np.round(Ti -T0,6)
211             GPIOADC.fichier.write('%s,%s,%s\n'%(Temps,charge.value,GPIOADC.ValeurADC))
212             if (i%500) == 0 and i!=0:print i,' ',Temps,' ',charge.value,' ',GPIOADC.ValeurADC
213             i+=1
214             data[(GPIOADC.ValeurADC-1)/32] += int(1)
215             if GPIOADC.ValeurADC >= 4000:
216                 if 'BACKG' in Informations[7].upper():
217                     Evolution[int(Temps/DT)] += 1
218                 else:
219                     Evolution[int(charge.value/DC)] += 1
220             Last = np.append(Last,int((GPIOADC.ValeurADC-1)/32))
221             if len(Last) > 5000:
222                 Last = np.delete(Last,0)
223             temp = Tally(Last)
224             Last5000[:] = temp[:]
225         j+=1
226         if i > 2**32:
227             e.set()
228             print "forced program to close!"
229     GPIO.output(4,1) #Ferme le ADC??
230     T1 = time.time()
231     print (T1 - T0)/(j*1.0),'seconds per cycle'
232     print (T1 - T0)/(i*1.0),'seconds per points'
233     GPIOADC.Close()
234     GPIO.cleanup()
235     print "Close GPIO"
236     return True
```

Le software

```
249 def Flaging(event):
250     Input = raw_input('\nType \'stop\' to stop\n\n')
251     while Input.upper() != 'STOP':
252         Input = raw_input('Type \'stop\' to stop\n')
253     print "STOP!"
254     event.set()
255     return True
```

Le software

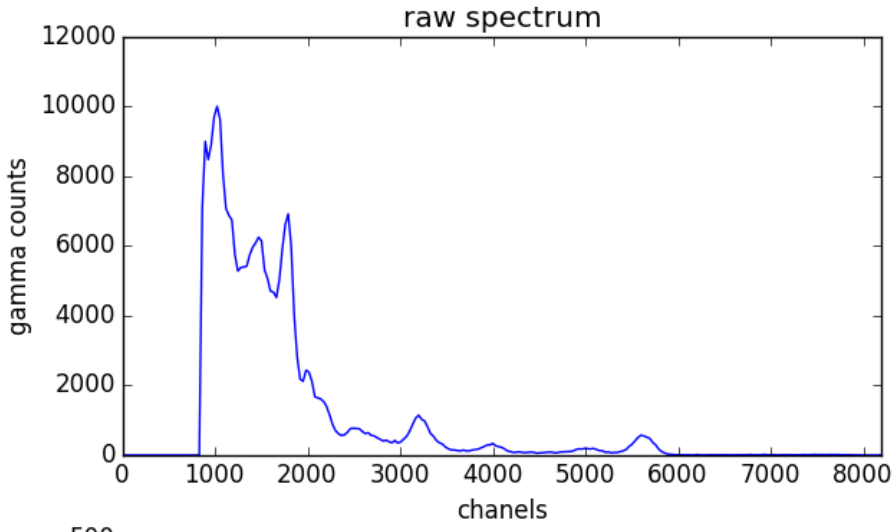
Dans un :

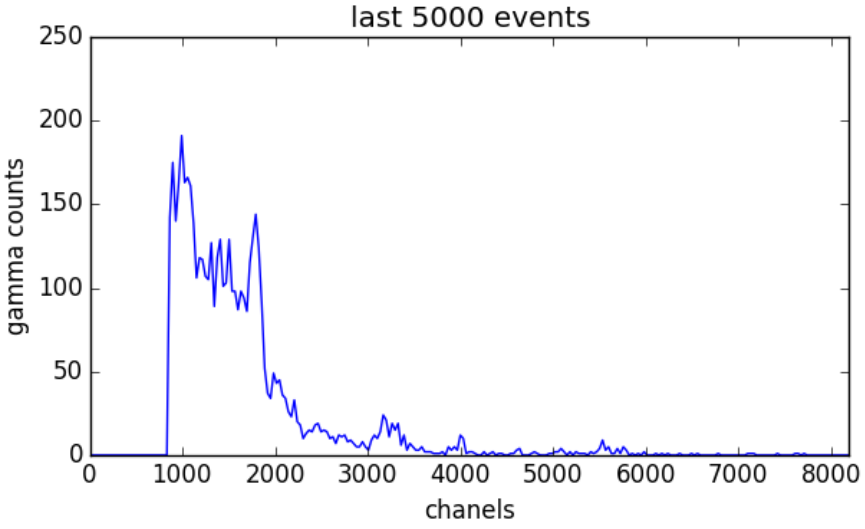
`if __name__ == '__main__':`

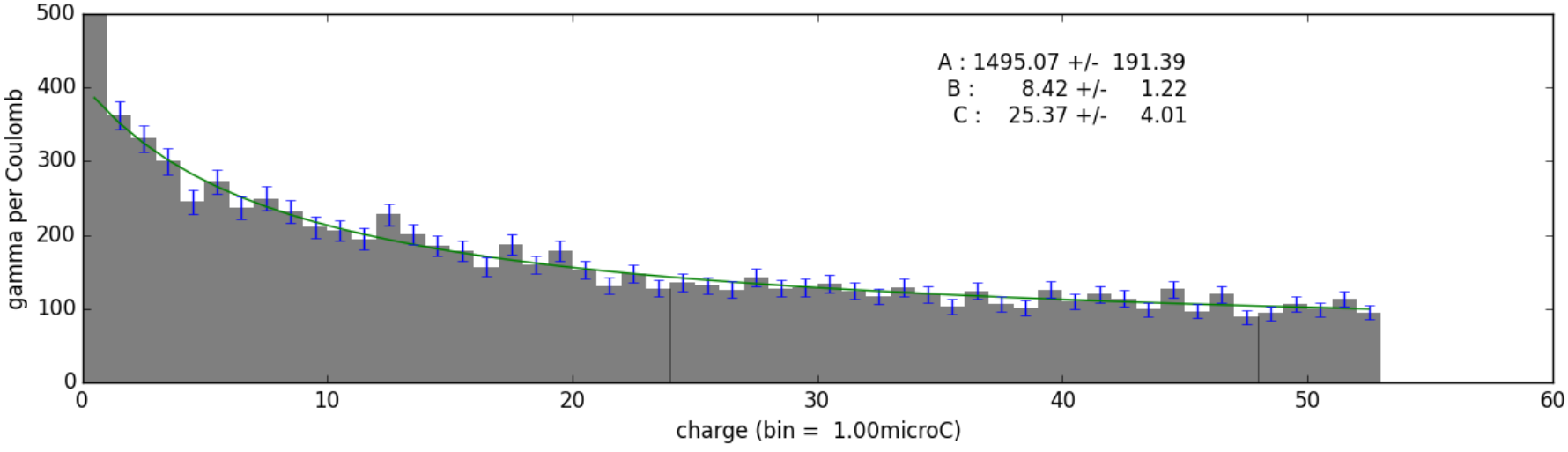
```
458     Informations = GetInfo()
459     ProcessA = multiprocessing.Process(target = Counting,args = (event,dataraw,dataIvsCharge,Last1000,charge,Informations))
460     ProcessB = multiprocessing.Process(target = SecondProcess,args = (event,charge))
461
462     ProcessA.start()
463     ProcessB.start()
464
465     time.sleep(20)
466     threadFlag = threading.Thread(target = Flaging,args = (event,))
467     threadFlag.daemon = True
468     threadFlag.start()
```

```
513         ProcessA.join()
514         ProcessB.join()
515         threadFlag.join()
516 GPIO.output(4,1) #Ferme le ADC??
517 GPIO.cleanup()
518 print "Stoped Processes and threads"
```

Le résultat


raw spectrum
gamma counts
chanels
0
2000
4000
6000
8000
10000
12000
0
1000
2000
3000
4000
5000
6000
7000
8000


last 5000 events
gamma counts
chanels
0
50
100
150
200
250
0
1000
2000
3000
4000
5000
6000
7000
8000


A : 1495.07 +/- 191.39
B : 8.42 +/- 1.22
C : 25.37 +/- 4.01
gamma per Coulomb
charge (bin = 1.00microC)
0
100
200
300
400
500
0
10
20
30
40
50
60

La suite?

Retour sur le survol

- Physique
  - NRA
  - Acier
  - CRIAQ
- Hardware
  - Arduino
  - RP 3
  - Le reste
- Software
  - Python 2.7
  - Rpi.GPIO
  - Pyserial
  - Matplotlib
  - Numpy
  - Subprocess
  - Multiprocess
  - Threading