Principles of Measurement & Instrumentation I

Laboratory

PHYS417

Lab Report

Abdullah Burkan BEREKETOĞLU

Experiment 3- Data Acquisition

Submitted By: Abdullah Burkan BEREKETOĞLU

Submitted to: Deniz Orhun BOZTEMUR & Enver BULUR

Student Id: 2355170

Experiment Date: 10.12.2019

Submission Date: 21.12.2019

OBJECTIVES

Here we wanted to learn interface usage by Oscilloscope and Python, visualize and gather data by pyvisa library. Also, there was the part that we used Pyserial library and Arduino.

INTRODUCTION

In the experiments, as one know, many things can be the cause of the error, also some of these are systematic errors. There can hysteresis happen (not going on the same path while loading and dropping.). Calibration might be done wrongly. Random changes in temperature, luminescence, and also noise can occur. Also one can say there is human error. Even things that we think we can precisely measure has many errors due to human error. To minimize this error we use instruments and interfaces of hardware and make machines manipulate the system rather than an imprecise human being. If this data gathering is made in computer or as one can say in a digital form. Then it can be said that this has a less human error, and it is called DAQ(Data Acquisition). In this experiment, we also did use Data Acquisition techniques, by connecting Arduino to PC by USB and collecting data. Also by the oscilloscope to PC.

NumPy & SciPy

SciPy can be said as the complementary library to NumPy and both are mathematical function libraries of python. One can think them as Scientific and Numeric Python.

Matplotlib

Python plotting library

PyVISA

Package that helped us to control measurement devices, therefore we could get datas.

PySerial

Package that helped us to communicate with Arduino also control it by using python.

EQUIPMENT

- LAPTOP (PC)
- Oscilloscope
- Arduino UNO
- Thermistor (not ours it was too tiny)

PROCEDURE & CALCULATIONS

1-

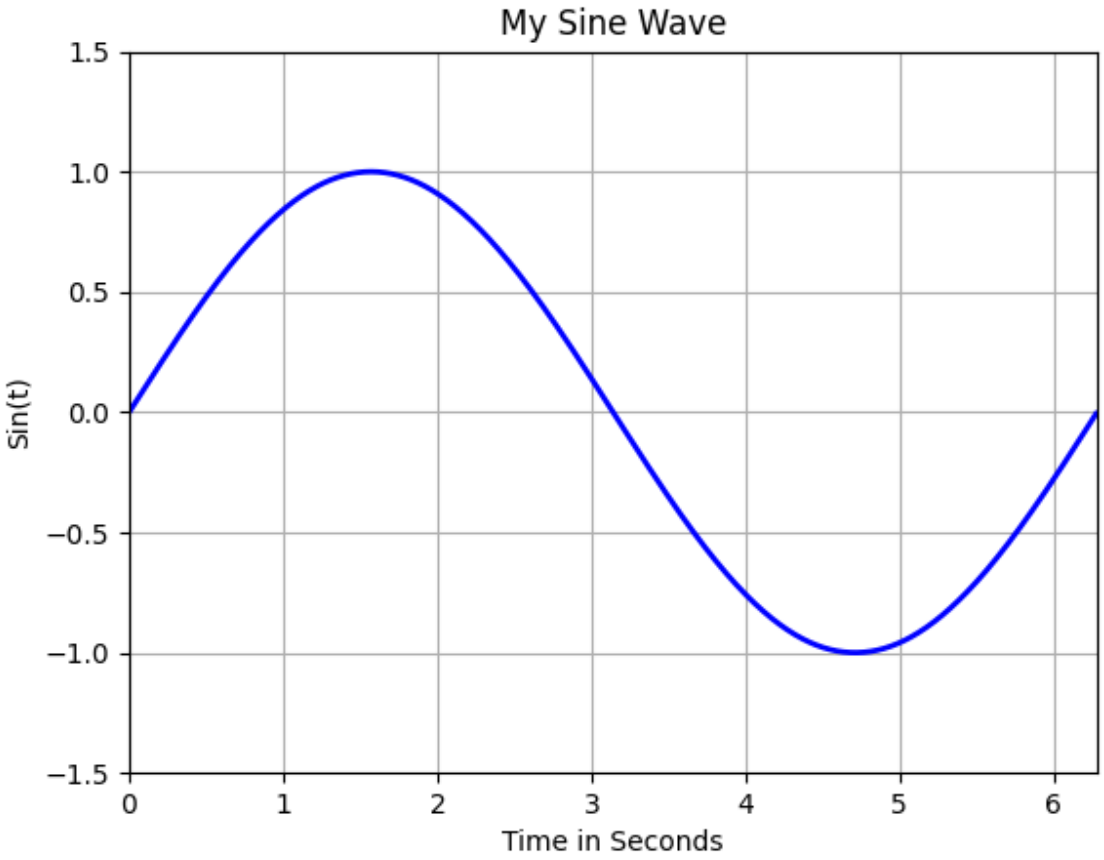

Figure 1. Sine Wave Graph

```python
import numpy as np
import matplotlib.pyplot as plt

x = []
y = []


for i in np.arange(0,2*np.pi, 2*np.pi/1000):
    x.append(i)
    y.append(np.sin(i))

plt.plot(x,y,"b-",linewidth= 2)
plt.grid(True)
plt.axis([0,2*np.pi,-1.5,1.5])
plt.title("My Sine Wave")
plt.xlabel("Time in Seconds")
plt.ylabel("Sin(t)")
plt.show()
```

Here is the code of Sine graph above.

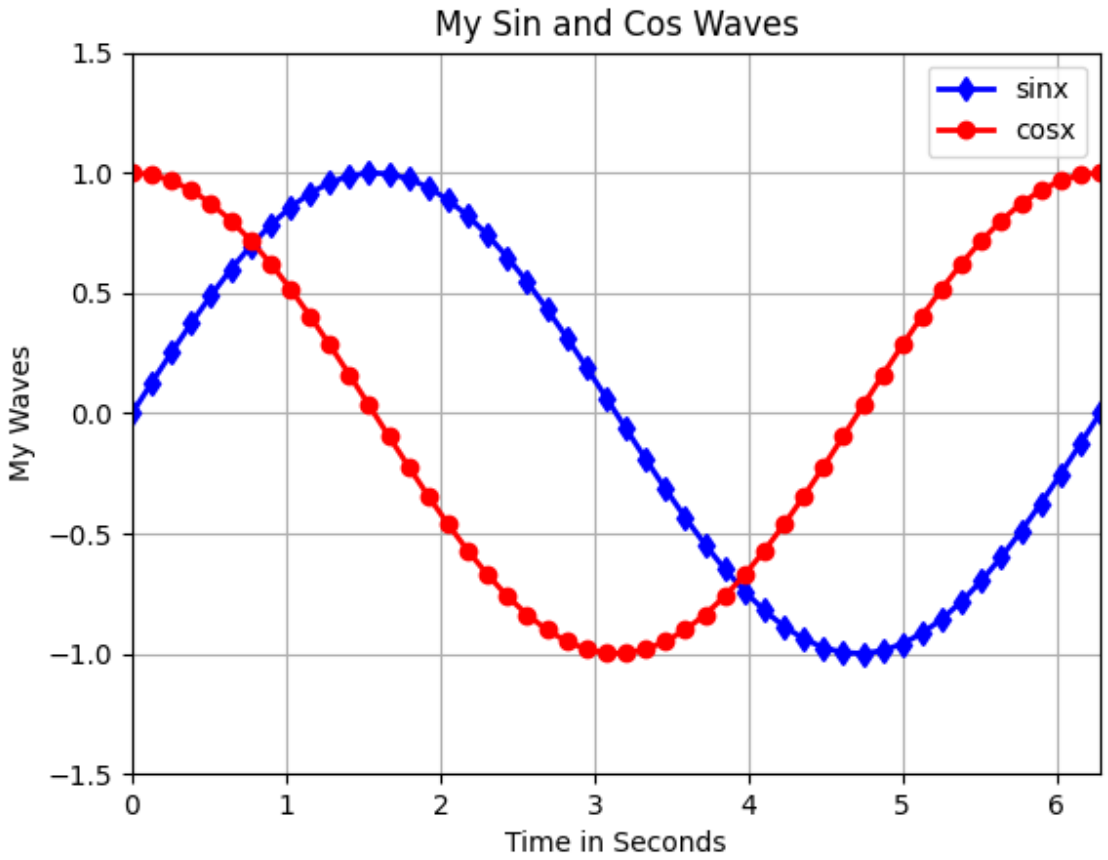

Figure 2. Graph for Sine and Cosine Wave Graph

```
import numpy as np # import numpy library
import matplotlib.pyplot as plt # import matplotlib



x=np.linspace(0,2*np.pi,50) # Create your x array
y=np.sin(x) # Create y array
z=np.cos(x) # create z array
plt.plot(x,y,'b-d',linewidth=2, label='sinx') # plot y
plt.plot(x,z,'r-o',linewidth=2, label='cosx') # plot z
plt.grid(True) # display background grid
plt.axis([0,2*np.pi,-1.5,1.5]) # set range on axis
plt.title('My Sin and Cos Waves') # chart title
plt.xlabel('Time in Seconds') # label x axis
plt.ylabel('My Waves') # label y axis
plt.legend() # show legend
plt.show() # show the plot

```

2-

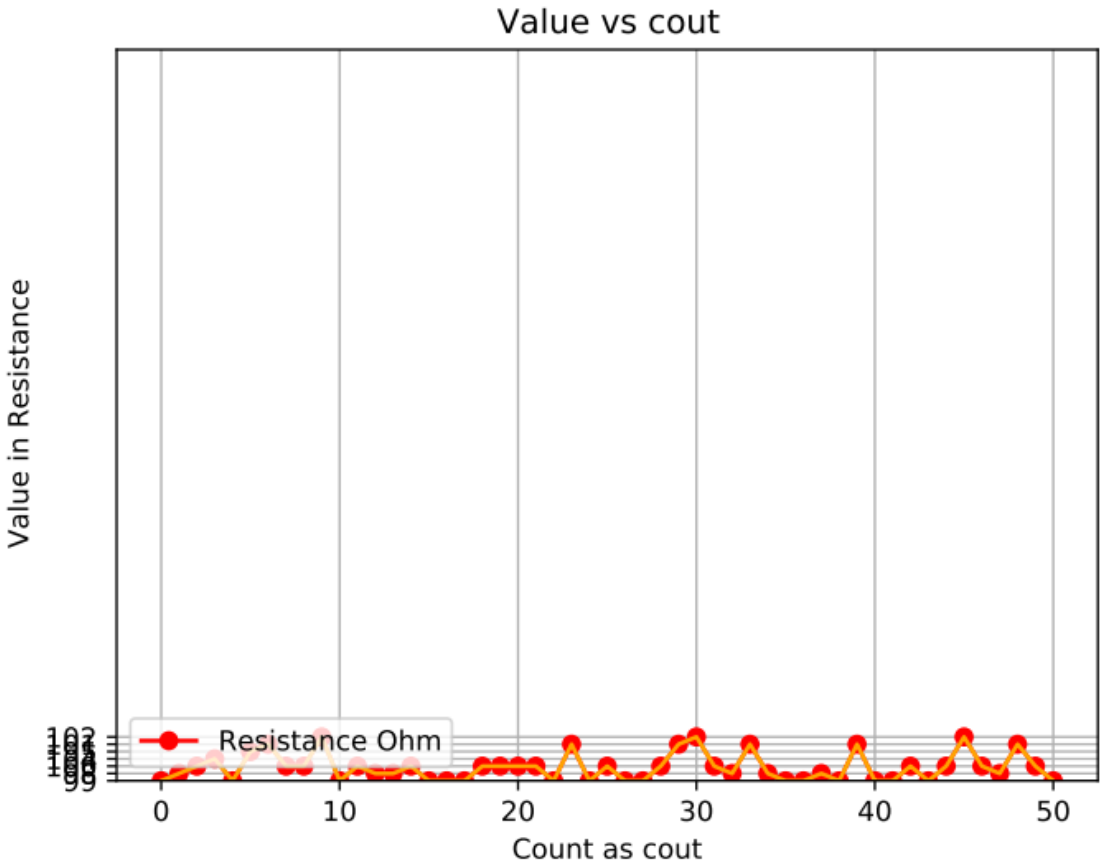

Figure 3 - Graph of Live data

```
import time
import serial
import matplotlib.pyplot as plt
from drawnow import *

f = open("volt_time.txt", "x")
"""
def makeFig():                                          # Create a function that makes our desired plot
    plt.ylim(0, 100)                                    # Set y limit as 0 to 500
    plt.xlabel('Time')
    plt.title('Volts vs time')                        # Write the title
    plt.grid(True)                                      # Turn the grid on
    plt.ylabel('Volts')                               # Set y's label
    plt.plot(value_array, 'ro-', label='Volts')     # plot the resistance
    plt.legend(loc='lower left')                        # plot the legend
"""
cout = 0
cout_as_array =[]
value_array = []
ser = serial.Serial('COM3', 9600)

while True:
    value = ser.readline().decode().strip("\r\n")
    print(value)
    f.write(value)

    time.sleep(0.1)
    value_array.append(value)
    cout_as_array.append(cout)
    cout += 1
    #drawnow(makeFig)
    if (cout == 51):
        f.close()
        break
"""
plt.plot(cout_as_array,value_array, color= "orange")
plt.title("Value vs cout")
plt.ylabel("Value in Volts")
plt.xlabel("Count as cout")

plt.gcf().savefig("Volts vs count.pdf")
"""
```

Code includes some other programs inside, in which some I forgot to take the picture.

3-

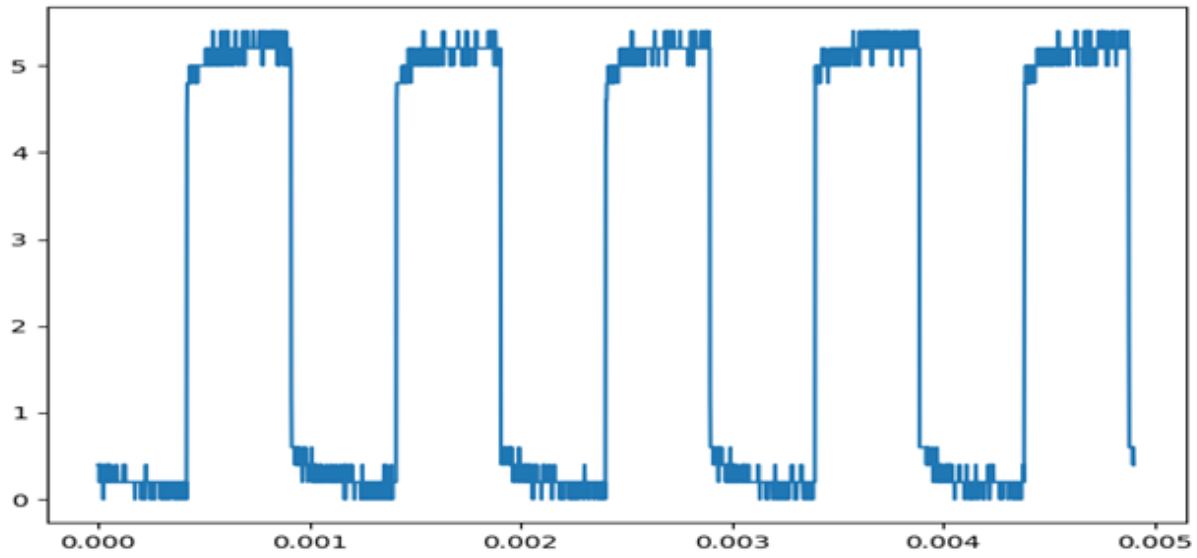

Figure 4- Oscilloscope graph.

```python
import pyvisa as visa
import numpy as np
import pylab
from struct import unpack
rm = visa.ResourceManager()
print(rm.list_resources())


inst = rm.open_resource('USB0::0x0699::0x0368::C027902::INSTR')
print(inst.query("*IDN?"))

inst.write('Data:SOU CH1')
inst.write('DATA:WIDTH 1')
inst.write('DATA:ENC RPB')
"""
inst.write('Data:SOU CH2')
inst.write('DATA:WIDTH 2')
inst.write('DATA:ENC RPB')
"""
ymult = float(inst.query('WFMPRE:YMULT?'))
yzero = float(inst.query('WFMPRE:YZERO?'))
yoff = float(inst.query('WFMPRE:YOFF?'))
xincr = float(inst.query('WFMPRE:XINCR?'))

inst.write('CURVE?')
data= inst.read_raw()
headerlen = 2+int(data[1])
header = data[:headerlen]
ADC_wave = data[headerlen:-1]
ADC_wave=np.array(unpack('%sB' % len(ADC_wave),ADC_wave))
Volts = (ADC_wave - yoff)* ymult + yzero
Time = np.arange(0, xincr* len(Volts),xincr)
pylab.plot(Time,Volts)
pylab.show()
```

4-

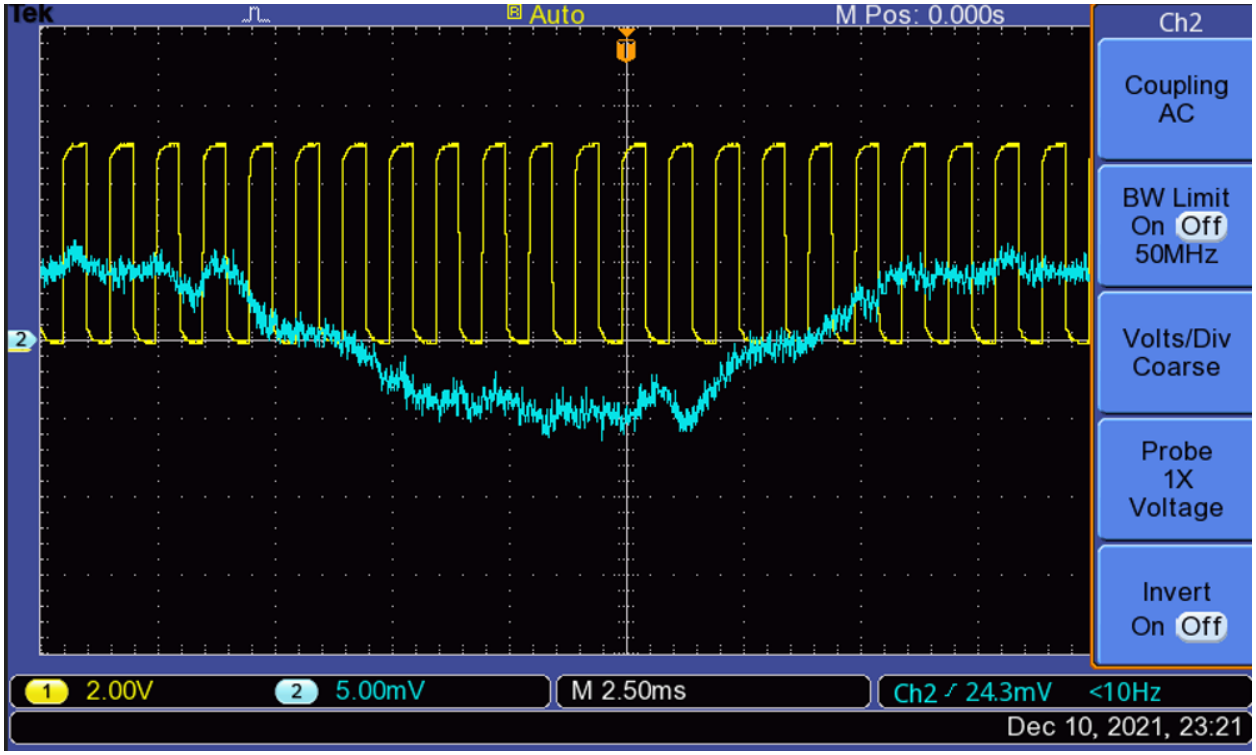

Figure 5. Screen Save of two channel oscilloscope waves.

```python
import visa

rm = visa.ResourceManager()
print(rm.list_resources())
inst = rm.open_resource('USB0::0x0699::0x0368::C027902::INSTR')
inst.query('*IDN?')

inst.write('SAVE:IMAG:FILEF PNG')
inst.write('HARDCOPY START')
data = inst.read_raw()

file1 = open('70khz 0.0001uf.bmp', 'wb')
file1.write(data)
file1.close()

print("Image is ready")
```

5- In the code photo of part 3 I actually did conver arduino code of Serial print into python code too so many parts collided to each other. I forgot to save this one and directly used the python one.

```
 // the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

6- Even this part is actually in the part 3 code which I will again put in this one.
I only took the drawnow graph.

```
import time
import serial
import matplotlib.pyplot as plt
from drawnow import *

f = open("volt_time.txt", "x")
"""
def makeFig():                                          # Create a function that makes our desired plot
    plt.ylim(0, 100)                                    # Set y limit as 0 to 500
    plt.xlabel('Time')
    plt.title('Volts vs time')                        # Write the title
    plt.grid(True)                                      # Turn the grid on
    plt.ylabel('Volts')                               # Set y's label
    plt.plot(value_array, 'ro-', label='Volts')     # plot the resistance
    plt.legend(loc='lower left')                        # plot the legend
"""
cout = 0
cout_as_array =[]
value_array = []
ser = serial.Serial('COM3', 9600)

while True:
    value = ser.readline().decode().strip("\r\n")
    print(value)
    f.write(value)

    time.sleep(0.1)
    value_array.append(value)
    cout_as_array.append(cout)
    cout += 1
    #drawnow(makeFig)
    if (cout == 51):
        f.close()
        break
"""
plt.plot(cout_as_array,value_array, color= "orange")
plt.title("Value vs cout")
plt.ylabel("Value in Volts")
plt.xlabel("Count as cout")

plt.gcf().savefig("Volts vs count.pdf")
"""
```

7- here is the text file values for part 7

```
0.50    0    0.49    150    0.50    301    0.49    451    0.51    602    0.49    752    0.48    903    0.49    1053    0.49    1204    0.50    1354    0.50    1505
0.50    1656    0.49    1807    0.49    1957    0.50    2108    0.51    2258    0.50    2409    0.50    2560    0.48    2710    0.49    2862    0.49    3012    0.50
3163    0.49    3313    0.49    3464    0.50    3614    0.49    3765    0.50    3915    0.50    4067    0.50    4217    0.50    4368    0.50    4518    0.50    4669
0.50    4819    0.50    4970    0.49    5121    0.50    5272    0.50    5423    0.49    5573    0.49    5724    0.50    5874    0.49    6025    0.50    6175    0.50
6327    0.50    6477    0.49    6628    0.49    6778    0.50    6929    0.49    7079    0.49    7230    0.50    7380    0.49    7532
```

Figure 6. Volt_time.txt

8-

sketch_dec10a | Arduino 1.8.18 (Windows Store 1.8.55.0)

File Edit Sketch Tools Help

sketch_dec10a

```
#define LED 13
void setup() {
    pinMode(LED, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    if (Serial.available()) {
        char serialListener = Serial.read();
        if (serialListener == 'H') {
          Serial.println("LED HIGH");
            digitalWrite(LED, HIGH);
        }
        else if (serialListener == 'L') {
            digitalWrite(LED, LOW);
            Serial.println("LED LOW");
        }
    }
}
```

Figure 7. Code on Arduino

```
import serial

import time

arduino = serial.Serial('COM3',9600)

def onOffFunction():

        command = input ("Type something: (on/off/ exit)");

        if command =="on":

        print("LED on")
```

```
            time.sleep(1)
            arduino.write('H'.encode())
            onOffFunction()
            elif command =="off":
            print("LED off")
            time.sleep(1)
            arduino.write('L'.encode())
            onOffFunction()
            #when input is "off", it connects with arduino and use encoded L to not blink LED.
            elif command == "exit":
            print("See You!")
            time.sleep(1)
            arduino.close()
            #when input is "exit", it closes the code.
            else:
            print("Try Again!")
            onOffFunction()
            time.sleep(2)
onOffFunction()
```

DISCUSSION & COMMENT

This lab was hard for me, even though I had some background in programming, I never used such libraries, hence struggled a lot. The pre-report was hard for me too, because I assumed that I needed to memorize all the functions that are written as pre report homework. Thanks to Deniz hoca, I and other could manage to solve the problems with ease other than the query problem in the 2 channel call from the oscilloscope. I believe I still do need practice in the case.

REFERENCES

Programming Example: SDS Oscilloscope screen image capture using Python over LAN - Siglent (siglentna.com)
Sources of Error in Science Experiments (sciencenotes.org)
Remote Oscilloscope Operation with Python and VISA (mindchasers.com)
Sending Data From Arduino to Python Via USB : 4 Steps - Instructables