



**Design and implementation of a Characterization and
Validation Program based on Textile Sensors and a
Picoscope**

A Degree Thesis

Submitted to the Faculty of the

**Escola Tècnica d'Enginyeria de Telecomunicació de
Barcelona**

Universitat Politècnica de Catalunya

by

Nuria Revuelta Yus

In partial fulfilment

**of the requirements for the degree in Enginyeria de
Tecnologies i Serveis de Telecomunicació**

Advisor: Fernando Seoane i Ramon Bragós

Barcelona, January 2020

Abstract

In a few years, the necessity of being able to know and manage the new discoveries and trends in the field of the communication technologies will stop being an option to start being an obligation.

The reason of this new necessity is that there is no doubt that we are living in an era in which digital technology is transforming societies at an unsuspected speed and, at the same time, makes an unprecedented progress while posing new and profound challenges.

In another hand, the development of the program with the Arduino and the Picoscope is a part that needs to be used in a laboratory in Sweden in the future, so all the process will need to be adjusted to the necessities that the investigation team has.

Resum

En pocs anys, la necessitat de poder conèixer i gestionar els nous descobriments i tendències en el camp de les tecnologies de la comunicació deixarà de ser una opció per començar a ser una obligació.

La raó d'aquesta nova necessitat és que no hi ha dubte que estem vivint en una era en què la tecnologia digital està transformant les societats a una velocitat increïble i, al mateix temps, fa un progrés sense precedents mentre s'enfronten nous i profunds desafiaments.

Per una altre part, el desenvolupament del programa amb l'Arduino i el Picoscope és una part que caldrà utilitzar en un laboratori a Suècia en el futur, per la qual cosa s'haurà d'ajustar tot el procés a les necessitats que tingui l'equip d'investigació.

Resumen

En pocos años, la necesidad de poder conocer y gestionar los nuevos descubrimientos y tendencias en el campo de las tecnologías de la comunicación dejará de ser una opción para comenzar a ser una obligación.

La razón de esta nueva necesidad es que no hay duda de que estamos viviendo en una era en la que la tecnología digital está transformando a las sociedades a una velocidad increíble y, al mismo tiempo, hace un progreso sin precedentes mientras plantea nuevos y profundos retos.

Por otro lado, el desarrollo del programa con el Arduino y el Picoscope es una parte que debe usarse en un laboratorio en Suecia en el futuro, por lo que todo el proceso deberá ajustarse a las necesidades que tenga el equipo de investigación.

Acknowledgements

I want to thank both my tutor Ramon Bragós for all the support in Barcelona and to my tutor Fernando Seoane for giving me advice every time I need it, and to trust me to help him in his project in Sweden.

I also want to thank my family and my sister for supporting me economically and mentally in my stay abroad and to my dear friend Marta Marin, thanks for being my lab mate all this years, without you the degree wouldn't have been the same.

Also I want to thank for the support and the help with a project that was way out of my field in telecommunications to my friends of the university, Silvia, Kike, Moha, Roger, Alberto, Christian... thanks for taking care of me every day during all this 4 years, and finally I also want to thank the new friends I made during my stay in Sweden.

Revision history and approval record

Revision	Date	Purpose
0	08/01/2020	Creació del document
1	19/01/2020	Redacció del document
2	25/01/2020	Redacció del document
3	26/01/2020	Redacció del document

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Nuria Revuelta Yus	revueltanuria.sgr@gmail.com
Ramon Bragós Bardia	rbb@eel.upc.edu
Fernando Seoane Martinez	fernando.seoane@hb.se

Written by:		Reviewed and approved by:	
Date	26/01/2020	Date	26/01/2020
Name	Nuria Revuelta Yus	Name	Ramon Bragós Bardia
Position	Project Author	Position	Project Supervisor

Table of contents

Abstract.....	1
Resum	2
Resumen	3
Acknowledgements	4
Revision history and approval record.....	5
Table of contents	6
List of Figures	7
1. Introduction.....	8
2. Basic background definitions	9
3. Arduino Classes	9
3.1. Introduction	11
3.2. Content	12
3.2.1. Lecture	12
3.2.2. Laboratory	13
4. Characterization	14
4.1. Scope	14
4.2. Design	15
4.3. Implementation	16
5. Validation	19
5.1. Scope	19
5.2. Design	20
5.3. Implementation	21
6. Results	23
7. Cost:	24
8. Conclusions and future development:	25
Bibliography:.....	26
Annex:.....	27

List of Figures

Figure 1. First slide of the PowerPoint	12
Figure 2. Students in the laboratory	13
Figure 3. Characterization Diagram	15
Figure 4. Picoscope device	16
Figure 5. Signal generator option	16
Figure 6. Measured Signal	16
Figure 7. Characterization UI	17
Figure 8. Validation Diagram	20
Figure 9. Validation UI	21
Figure 10. Live Graphics.....	22

1. Introduction

The purpose of this report is to provide an overview of the product I have designed, make sense of the different parts that forms it and explain the whole procedure from the beginning in the Arduino classes development until the final validation done to the sensors that will be created during this process.

In my stay doing the project in Sweden, I worked with the team of my tutor in the University of Borås. This team teaches a class about Smart textiles which consists in giving the students some lectures of different topics related with development of textiles that may have a response to a stimulation, and when they have learned all the concepts needed, as electronics, sensors... , and also have practiced with some exercises of each one of the lectures in the laboratory. Then the students with all this knowledge need to develop a project that can be proposed by the university or by a company. In this year the project was the design of a textile touch sensor for Volvo Company.

For processing the data it was decided that some basic knowledge on Arduino programming will help the students to show if the textile sensor was being touched or not.

To begin with, this project is divided in 3 parts:

First we start with the Basic Arduino classes taught to students in the University of Borås taking the MSc textile engineering program, we will talk about how they developed theoretically in the lectures and practically at the laboratory.

For the next step we will suppose that an external person have developed a textile sensor and needs to view the main characteristics that is has by characterization the signal received in a testing environment. For this I will develop an application in python linked to a Picoscope device.

About the last part of the project, I will suppose that the students of the course have learned the basic knowledge of the lessons and have developed some functional textile sensors that need to be validated, I will develop with an Arduino a validation application to make sure this sensors are valid for their purpose.

With this 3 parts I will be able to make the sensors development and knowledge a bit more able to all kinds of people.

2. **Basic background definition:**

During the project we have used some resources and devices that are not common, so here you will find some background definitions to make sure all the concepts in the project are understood.

- **Arduino** : Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ('shields') or breadboards (For prototyping) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers can be programmed using C and C++ programming languages. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

- **Picoscope**: Is an portable kind of Oscilloscope. PicoScope software enables analysis using FFT, a spectrum analyser, voltage-based triggers, and the ability to save/load waveforms to disk. PicoScope is compatible with Parallel port oscilloscopes and the newer USB oscilloscopes.

All PicoScope models include an integrated function generator or arbitrary waveform generator, triggering, automatic measurements with statistics, a Fast Fourier transform spectrum analysis mode, waveform maths, mask limit testing, and serial decoding for I2C, SPI, UART, CAN, LIN and FlexRay.

- **Python** : Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

- **MVC**: Model–view–controller (usually known as MVC) is a software design pattern commonly used for developing user interfaces which divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. Following the MVC architectural pattern decouples these major components allowing for code reuse and parallel development.

Traditionally used for desktop graphical user interfaces (GUIs), this pattern has become popular for designing web applications. Popular programming languages like JavaScript,

Python, Ruby, PHP, Java, C#, and Swift have MVC frameworks that are used for web or mobile application development straight out of the box.

- **DeepMeasure**: DeepMeasure uses up to 100 million samples to capture the results of every cycle contained in each triggered waveform acquisition.

Results are displayed in a table, with the parameter fields shown in columns and waveform cycles shown in rows.

3. Arduino Classes

The first part of the project was doing Arduino classes for the students of the Master in textile engineering in the following points I will explain how was the process and the results achieved.

All the material and the study plan created for the classes as the laboratory practice, the PowerPoints can be found in the annex of the document.

3.1. Introduction

As I said before in the introduction of the project, the one I was working teaches a class about Smart textiles which consists in giving the students some lectures about topics as sensors, electronics, textile designs and mechanics, so at the end of the semester the student can develop a project.

This year the project was for Volvo company, that is one of the most important ones in Sweden and that is based next to the university in Gothenburg.

So I prepared some lectures which study plan, theory PowerPoint and exercises you will find in the annex of this paper, that may help the students to achieve easier and process the received data in a more technological way. Also I prepared an easy laboratory session with a mini project to start using potentiometers, sensors, led... and start keeping up with all this new instruments.

In the next points you will find the content and the results achieved in this sessions.

3.2. Content

As the necessities for the Volvo project were just to be able to get data from the sensor and once you have it, being able to see the changes in the parameters once you have a functional sensor.

For this I made some material that is divided in the things given in the lectures and in the other hand the laboratory material.

Next I will explain more in detail the information in each one of this parts.

3.2.1. Lecture

For the lecture I made a work plan starting by the board itself and finishing by explaining some more complex coding functions that may be useful. The plan was the following:

First I started with the topic “What is an Arduino?” in which I explained the different parts of it with a diagram and also how to connect it to the computer. The next topic was “Arduino environment” in which I explained how the environment worked, the different parts and how to upload and check the code is in it.

Next I started explaining things about variables, first by explaining what they are and next explaining one by one the different types and putting examples of the most important ones.

To continue, I explained how to set up the code and explained the 3 main things that need to be settled up in the Arduino environment: pins, connexions and random.

To finalize the variables I talked about the loop in Arduino environment, how does it work and what type of information to put on it.

To go more inside the Volvo project I explained most important thinks to know about how to make the “Application to sensors” some of the topics taught are Digital and Analog pins and sensors.

To end I explained some of the most basic programming functions as if-else, for or switch.

For all the students to practice about all this topics, I prepared some exercises about all of the theoretical definitions explained before, this exercises can be found in the annex.



Figure 1

3.2.2. Laboratory

For the laboratory session, I have prepared a more complex code in order to check the knowledge of the students once we put it in practice.

The exercise was about reading the voltage in a potentiometer with the analogic pins in the Arduino once you have this data, show it with a list of led that light progressively with the roll of the potentiometer.

The exercise that they developed had a variation of the code that you will find in the annex of the paper.

With this laboratory exercise they practice:

- Analog pins
- Digital pins
- Led
- Potentiometer
- If-else
- Loop and setup



Figure 2

4. Characterization

The characterization of new sensor prototypes to test and verify the functionalities in health and improving the sensing function.

All the code developed for the Characterization part, I mean the view.py, controller.py and model.py can be found in the annex of the document. Also there is the manual of how to use this part of the program.

4.1. Scope

As I introduced before in previous parts of the document, once we have developed a functional sensor is very useful to be able to see how this behave when we apply different frequencies to it in different environments, so with it we can characterize the signal we receive from the sensor.

For this porpoise we want to use a PicoScope which is an acquisition board plus a software that enables: analysis using FFT, a spectrum analyzer, voltage-based triggers, and the ability to save/load waveforms to disk.

As in the Picoscope we only have the ability to view the voltage signal we receive from the sensor we need to design a new method to see how the impedance of this devices changes.

Some of the main points we need to accomplish with this part of the project are the ones bellow:

- Be able to collect the data of the Picoscope and with a simple voltage divisor, fixed with a resistor of variable value that the user can change.

- One good additional point that may be a plus in the application is having the ability to save all the information we get from the signal in the Picoscope to a CSV file, that we could use later to make a more detailed analysis.

- Plot in an user-friendly application the impedance of the sensor, that you get by sampling the response at different frequencies.

- Use an organized the code, and also try to make easier to change it if necessary.

4.2. Design

For the design I created 3 python scripts, the first one (the model) makes the communication with the Picoscope device, the view creates a python application that shows the user the impedance data collected from the lecture of the Picoscope and finally the last scrip that controls the communication between the device and the view.

The parts of this Characterization process are:

- Script Model
- Script Controller
- Script View
- Application
- Picoscope
- Sensor

This follows the next diagram:

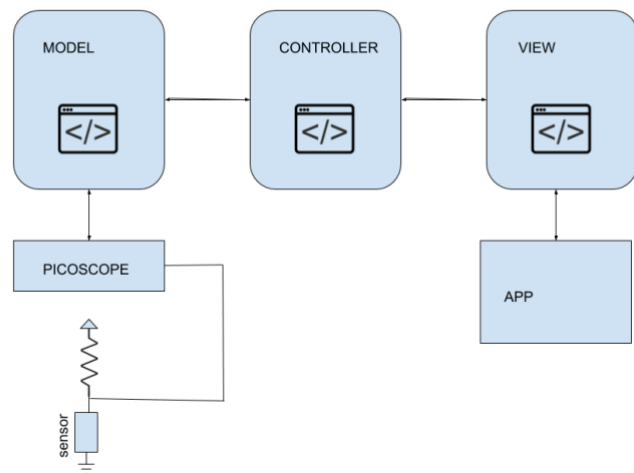


Figure 3

4.3. Implementation

About the way I did the implementation of the characterization part, I will explain in some basic point how the process developed and how we got the results that will be explained next, also in this I will explain the different difficult situations that I found during the process.



Figure 4

Firstly we investigate about the Picoscope and about how to call it and get its data from a signal on it.

To get this we use the terminal commands:

To generate the signal we want in the Picoscope we will use:

```
# picoscope /a Siggen.Frequency.Value=  
# picoscope /a Siggen.Amplitude.Value=
```

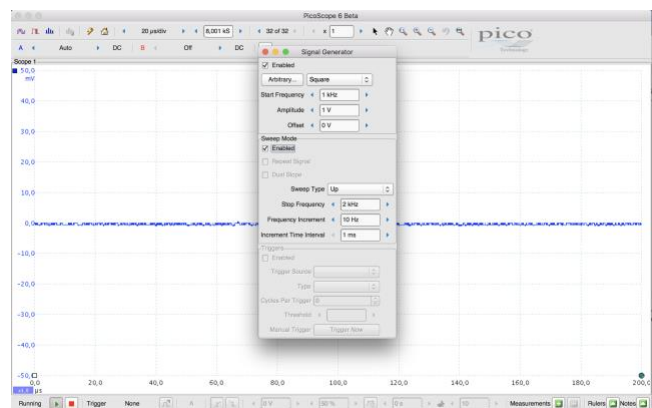


Figure 5

To read the measurements we have in the channel of the oscilloscope we will use the following command in the command line:

```
# picoscope /a Measurements.CSV?
```

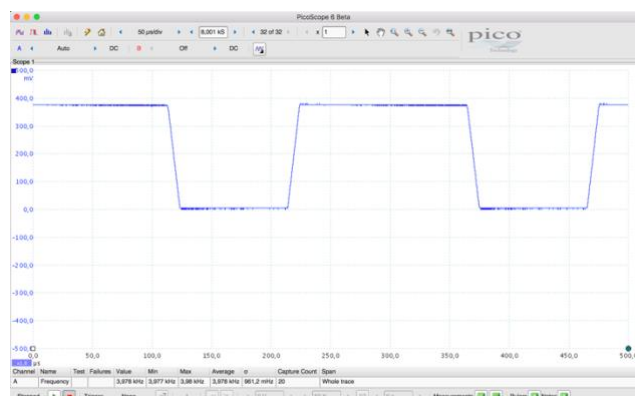


Figure 6

All this commands only work for windows based OS so for the others I decided to create a new method to get the data form inside the Picoscope program with a method of the latest beta version called DeepMesure.

As the oscilloscope measures the voltage, if what we want is to get the exact value of the resistance in every variation of the sensing variables, we will need to construct a voltage divisor, with a resistor that may be changed by the user.

With this we are able to get the data from both channels of the tensor divisor one up on the signal generator and the other on the middle of the divisor and for Windows OS get the data automatically and in Linux and MacOS generate a csv file. This csv file will be collected by the program in the same way as the Windows based code.

Once we have decided this part I started the software investigation, so to make sure that all the information was clear and correct and I decided to get a Model View Controller architecture in the code so both 2 sides communication (with the Application and with the Picoscope device) were well structured.

For the communication with the Picoscope we have use the basic sys python3 functions so we can get the information of the measurements in the terminal. Also for the csv case we use the import csv from python3 to open both csv files needed.

With this, for in the hardware, case we have applied two possible models, one with the sensor up on the voltage divisor, and one with the sensor down. So the user can choose were to put the sensor, and the fixed resistor that will be used to check the impedance of the sensor.

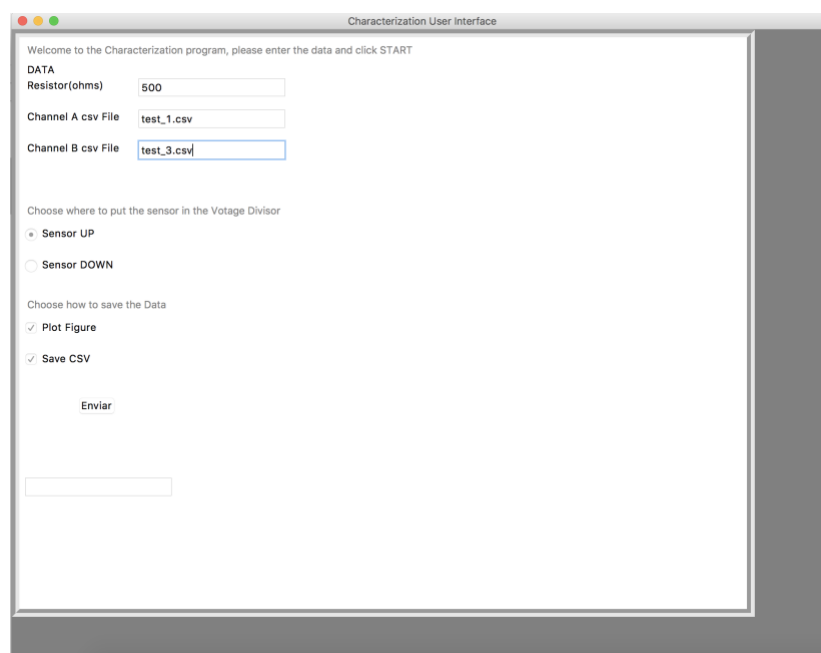


Figure 7

For the view and the User Interface, I used a python module tkinter which is used to create a view python application easy to run and that does not need any extra modules, I think that is good because this program needs to be run by totally external people and all the other view modules that were tried, as PyQt5, that had problems with the Linux OS.

Finally all the I developed a manual for the user to know how to use the application, this manual can be found in the annex of the document.

5. Validation

The validation of new sensor prototypes show the results and say to the user that is working correctly.

All the code developed for the Validation (general.py) part is in the annex of the document.

5.1. Scope

This last part of the project will try to make us validate the sensing functions and sensing development by people with no necessary knowledge of neither software development and electronic circuits. For this purpose we will need to develop an software application linked with an electronic circuit that needs to be able to adapt the sensor to be used by an Arduino UNO board. And show the variation of the parameters in a user-friendly graphic. The most important things that I need to accomplish in this part are the ones listed below:

- Front-end application with the necessary information to show the user the progress of the sensing in function showing in a dashboard the maximums and minimums.
- Front-end application with the possibility to calibrate the sensor inside its maximum and minimum margins and to recalibrate if necessary.
- Making, with the least error possible to adapt a random hand-made sensor (Inductive, resistive...) to be used by an Arduino UNO

5.2. Design

For the design I created 1 python script with all the information, to simplify the content because this was one of the main points of this part, at the end this needs to be gotten by all the students in the smart textiles class. For the view we have done the user interface coded in python3 as in the previous part just to make the program more easy to follow.

Finally for showing the results to the user I prepared a changeable live dashboard that could change as the sensor's parameter is increasing or decreasing.

For the adaptation of the different kinds of sensors (inductive, capacitive, resistive) I decided that at the end as I gave in the Arduino class, the best option was prepare a different code that uploads to the Arduino depending on the kind of sensor we are using adapting it to the use of the python program.

This follows the next diagram:

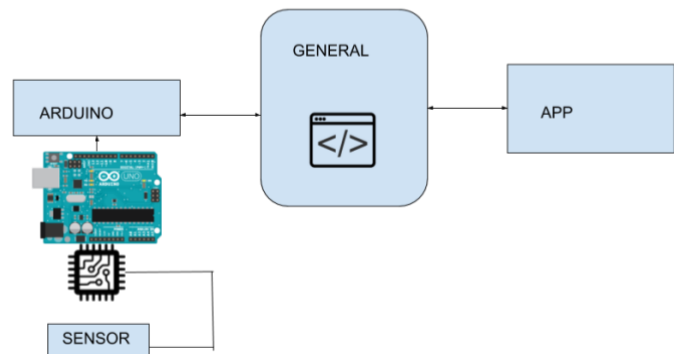


Figure 8

5.3. Implementation

To implement all the validation part we need to make sure everything was as simple as possible to the users to be able to validate the hand-made sensor in each one of its different varieties, either if it is inductive, resistive or a capacitor.

To adapt this the best option is create different codes for the Arduino, in the annex you will find the one for resistive handmade sensors and one for the capacitive sensors, you need to upload it to the Arduino depending what kind you are using.

Once you have uploaded the code to the board, the python general script will read the serial with the information that is being collected and once you have it open, you will need to calibrate it by putting the sensor in its maximum and minimum.

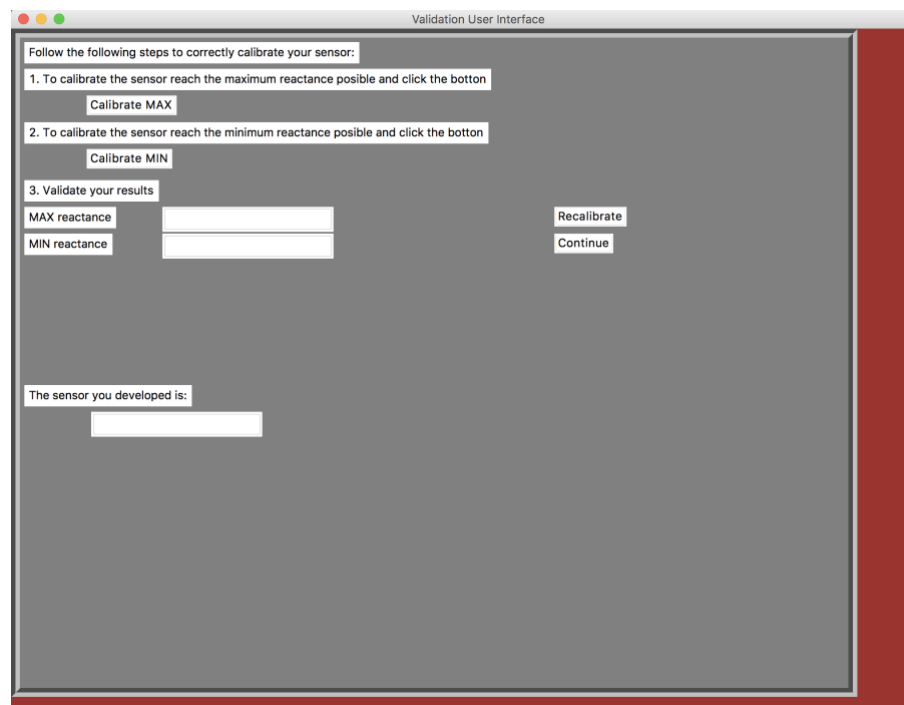


Figure 9

To connect the code in python to the Arduino device, I use the library pyfirmata, and if everything is working well a dashboard done with the matplotlib module will appear and change live, so the students can see perfectly how the values of the sensor that has been created is changing in every moment.

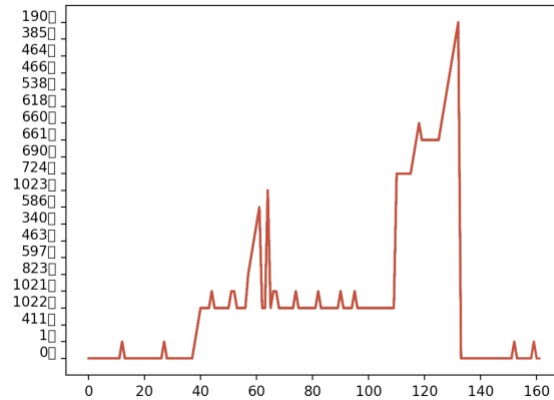


Figure 10

Finally as in the previous part of the project I used the tkinter module to make the view with which the user will work, as I said before it is the easier to work with and is always available.

6. Results

The general valuation of this three parts project has been very good for me, is true that the project was very ambitious but at the end the general result is good.

About the individual results, we can divide them in the 3 parts that my project has:

The results of the classes were very positive they did all the exercises perfectly and also the laboratory project, all the students were divided in groups of two and created a version of the code to made the lightning of the led list.

To end after all the classes and practice they developed correctly the project for Volvo with no major issues.

The results of the characterization were correct in general, all the measurements were done correctly although that we experienced some issues with the Picoscope software as it is only a beta version that is constantly changing with new releases, some of the errors that I found were impossible to be solved because of the bugs in the program itself.

About my code in python, the program did not work correctly for low frequency because when we use DeepMeasure, that is the only option for some OS, the values of the sampling are forced to start high in frequency. Exactly the values for sampling started at 1 kHz, when they should start at 1 Hz.

Finally the results for the Validation program were the expected, we show in the computer the changing value of the parameters in the hand-made sensor. And the students could do this process by their selves.

To conclude I'm satisfied with the results I got in the project at the end, all of this could be used in the future for the students and I think that this knowledge can be exported to the same subject in next semesters.

7. Cost

Material

Component	Cost (€)
Arduino UNO	23
Picoscope	200255
Coaxials wires and converters	65
PC*	500

* Amortization of the product for 4 months considering a shelf life of 3 years

Personal Costs

Employee	Total hours	€/h	Total (€)
Junior Engineer	400 h	9	3600

8. Conclusions and future development:

From the beginning, the whole project was very ambitious. Since the first day, all the team in boras and the students has been working hard and giving their best to make the project succeed. Every single decision has been debated and finally made with internal consensus and the approval of the tutor in the university.

During the project development, we had to change and add several features of the project due to the OS problems, but I managed to do it efficiently. As the work was well organized, I could afford implementing all that new requirements that were coming up by the tutor.

As my tutor was living in Stockholm and only came to the university once every week, sometimes I realized that there was a little lack of communication between us. Nevertheless, having noticed that, we fixed it by making weekly general meetings and extra meetings, when needed.

About the things that may be improved of the project I did, I mean the future developments that might be good to continue with the project if the time wasn't a problem, some of them are the ones that are listed next:

- More beautiful User Interface for the Validation project

- Improvement in the range of measuring the impedance in the Picoscope so it can measure low frequency.

- Easier to run code, as now it only can be run in the terminal, by writing python3 and the name of the script.

Bibliography:

1. Videos to learn how to use Arduino from scratch:
https://www.youtube.com/playlist?list=PLZfay8jtbyJt6gkkOgeeapCS_UrsgfuJA
2. Reference to all the functions:
<https://www.arduino.cc/reference/en/>
3. Tutorials
<https://www.arduino.cc/en/Tutorial/HomePage?from=Main.Tutorials>
4. Useful general content
<https://www.makeuseof.com/tag/learn-electronics-arduino-just-watching-videos/>
5. Information for background definitions
<https://es.wikipedia.org/wiki/Wikipedia:Portada>
6. Doubts and imports
<https://stackoverflow.com/>
7. Picoscope Software
<https://www.picotech.com/downloads>
8. Picoscope Doubts
<https://www.picotech.com/support/>

Annex:

Code of the model.py:

```
import csv
import numpy as np
import matplotlib.pyplot as plt
import view_4
import os

#Cycle No. 0
# Cycle Time (s) 1
# Frequency (Hz) 2
# Low Pulse Width (s) 3
#High Pulse Width (s) 4
# Duty Cycle (High) (%) 5
# Duty Cycle (Low) (%) 6
# Rise Time (s)  Fall Time (s) 7
# Undershoot (%) 8
# Overshoot (%) 9
# Max Voltage (V) 10
# Min Voltage (V) 11
# Voltage Pk-Pk (V) 12
# Start Time (s) 13
# End Time (s) 14

pathResult = "/Caracterization Program/Results CSV"

#Open de CSV file, then read and return the data on it, to call it you need name and path

def get_files(path):
    files = os.listdir(path)
    for filename in files:

        if os.stat(os.path.join(path, filename)).st_size == 0:
```

```

    os.remove(path + filename)

    files.remove(filename)

    if ".csv" not in filename:
        files.remove(filename)

    return files

def get_DataCSV(file_name):
    try:
        with open(pathResult+"/"+file_name, 'r') as f:
            reader = csv.reader(f, delimiter=',')
            head = next(reader)
            data = np.array(list(reader))
            return data
    except:
        view_4.setError(1)

#Gets the necessary data in the file with the numpy library, by returning a frequency and voltage
array
def getMeasurementsCH(file_name):

    data = get_DataCSV(file_name)
    freq = np.array(data).astype(str)
    freq = freq[:,2]
    volt = np.array(data).astype(str)
    volt = volt[:,10]
    nfreq = np.empty(len(freq))
    nvolt = np.empty(len(volt))
    for n in range(0, len(freq)):
        nfreq[n] = freq[n].replace(",",".")
        nfreq[n] = float(nfreq[n])
        nvolt[n] = volt[n].replace(",",".")

```

```

    nvolt[n] = float(nvolt[n])
    return nfreq, nvolt

#With the data of the voltage on both channels calculate  $Z= V_{\text{sensor}}/(V_{\text{resistor}}/R)$  sensor is down
the Voltage divisor
def model_A(file_name, resistor):

    frequency = []
    voltage = []

    for i in len(file_name):
        frequency[i], voltage[i] = getMeasurementsCH(file_name[i])

    N =len(ch1Voltage)

    try:
        ch1Voltage = ch1Voltage - ch2Voltage
        impedance = np.empty(N)
        for n in range(0, N):
            impedance[n] = ch1Voltage[n] / (ch2Voltage[n] / resistor)
        return [ch2Frequency, ch1Voltage, ch2Voltage, impedance]
    except:
        view_4.setError(3)

#With the data of the voltage on both channels calculate  $Z= V_{\text{sensor}}/(V_{\text{resistor}}/R)$  sensor is up the
Voltage divisor
def model_B(file_nameA,file_nameB, resistor):

    ch1Frequency, ch1Voltage = getMeasurementsCH(file_nameA)

```

```

ch2Frequency, ch2Voltage = getMeasurementsCH(file_nameB)
N =len(ch1Voltage)
try:
    for n in range(0, N-1):
        ch1Voltage[n] = ch1Voltage[n] - ch2Voltage[n]
    impedance = np.empty(N)
    for n in range(0, N-1):
        impedance[n] = ch2Voltage[n] / (ch1Voltage[n] / resistor)
    return [ch2Frequency, ch1Voltage, ch2Voltage, impedance]
except:
    view_4.setError(3)

#Create a figure and plots the results
def plotResults(frequencySpace, impedance):
    try:
        plt.plot(frequencySpace, impedance)
        plt.xscale('log')
        if view_4.escala_type() == 1:
            plt.yscale('log')
        plt.show()
    except:
        view_4.setError(2)

#create a CSV file with the results data
def plotResultsCSV(name,frequencySpace, Ch1Voltage, Ch2Voltage, impedance):
    head = "Frequency (Hz), Ch1 Votage, Ch2 Voltage, Impedance"
    np.savetxt(pathResult+"/"+name+".csv", np.transpose([frequencySpace, Ch1Voltage,
Ch2Voltage, impedance]))

```

Code of controller.py:

```
# convertir la UX en py
import view_4
import model_2
import os

#Gets the simulation data on the UI and handles the data
def getSimulationValues():

    [resistor, name] = view_4.getData()

    return [name, float(resistor)]

def crear_estructura():
    os.mkdir('/Caracterization Program/Test CSV',exist_ok=True)
    os.mkdir('/Caracterization Program/Results CSV',exist_ok=True)

#chosse and apply between the 2 models on the circuit
def main_function():
    name = model_2.get_files()
    if view_4.sensor_type()== 2:
        [name_result, resistor] = getSimulationValues()
        frequencySpace, ch1Voltage, ch2Voltage, impedance = model_2.model_A(name,resistor)
    else:
        [name_result, resistor] = getSimulationValues()
        frequencySpace, ch1Voltage, ch2Voltage, impedance = model_2.model_B(name, resistor)
    return [frequencySpace, ch1Voltage, ch2Voltage, impedance]

#Process to do when click start button

def start():

    [frequencySpace, ch1Voltage, ch2Voltage, impedance] = main_function()
```



```

fig.csv =view_4.save_data()

if fig == 1:
    model_2.plotResults(frequencySpace, impedance)

if csv == 1:
    model_2.plotResultsCSV(frequencySpace, ch1Voltage, ch2Voltage, impedance)

#main loop

def main():
    crear_estructura()
    view_4.root.mainloop()

if __name__ == '__main__':
    main()

```

Code of view.py:

```

from tkinter import *
import controler_3

#Config root
root=Tk()
root.title("Characterization User Interface")
root.geometry("1050x850")
root.resizable(1,1)
#root.iconbitmap("alpaca.ico")
root.config(bg="grey")

```

```
#Config frame
frame=Frame()
frame.pack(side="left",anchor="n")
frame.config(bg="white")
frame.config(width="950",height="750")
frame.config(bd="10")
frame.config(relief="groove")

resData=IntVar()
ResultData=StringVar()

ErrorType=StringVar()

Label(frame,text="Welcome to the Characterization program, please enter the data and click
START",fg="grey",font=(14)).place(x=5,y=5)

titleLabel=Label(frame,text="DATA",font=(20))
titleLabel.place(x=5,y=30)

resLabel=Label(frame,text="Resistor(ohms)")
resLabel.place(x=5,y=50)
resEntry=Entry(frame,textvariable=resData)
resEntry.place(x=150,y=50)

ChALabel=Label(frame,text="In case of Save csv name the file:")
ChALabel.place(x=5,y=480)
ChAEntry=Entry(frame,textvariable=ResultData)
ChAEntry.place(x=250,y=480)

ErrorEntry=Entry(frame,textvariable=ErrorType)
ErrorEntry.place(x=5,y=560)
```

```
def getData():

    return [resData.get(), ResultData.get()]

def setError(ErrorValue):
    if ErrorValue == 1:
        ErrorType.set("Error loading the file")
    elif ErrorValue == 2:
        ErrorType.set("Error in showing results")
    elif ErrorValue == 3:
        ErrorType.set("Check the data on csv files make sure dimensions are correct")

#Model of Voltage Divisor

Label(frame,text="Choose where to put the sensor in the Votage
Divisor",fg="grey",font=(14)).place(x=5,y=90)

varOpcion = IntVar()

def sensor_type():
    return varOpcion.get()

Radiobutton(root, text="Sensor UP",variable=varOpcion,value=1).place(x=15,y=130)
Radiobutton(root, text="Sensor DOWN",variable=varOpcion,value=2).place(x=15,y=170)

#Log scale

Label(frame,text="Choose how you want the Y scale",fg="grey",font=(14)).place(x=5,y=200)

varEscala = IntVar()
```

```

def escala_type():
    return varEscala.get()

Radiobutton(root, text="Log",variable=varEscala,value=1).place(x=15,y=250)
Radiobutton(root, text="Escalar",variable=varEscala,value=2).place(x=15,y=290)

#How to save the Data

varFig = IntVar()
varCSV = IntVar()

Label(frame,text="Choose how to save the Data",fg="grey",font=(14)).place(x=5,y=340)

Checkbutton(root, text="Plot Figure",variable=varFig,onvalue=1, offvalue=0).place(x=15,y=395)
Checkbutton(root, text="Save CSV",variable=varCSV,onvalue=1,offvalue=0).place(x=15,y=430)

def save_data():
    return varFig.get(), varCSV.get()

#Send botton with diferent click

botonEnvio = Button(frame, text="Enviar",command=controler_3.start)
botonEnvio.place(x=75,y=660)
botonEnvio.config(cursor="hand2")

```

Code of general.py:

```

import numpy as np
import serial
from tkinter import *

```

```
import matplotlib.pyplot as plt
import sys
sys.setrecursionlimit(10000)

board = serial.Serial('/dev/cu.usbmodem1411',9600)

def calibrate_max(board):
    analog_value = board.readline()
    if analog_value is not None:
        return analog_value;
    else:
        recalibrate(board)

def calibrate_min(board):
    analog_value = board.readline()
    if analog_value is not None:
        return analog_value;
    else:
        recalibrate(board);

def recalibrate(board):
    max = calibrate_max(board);
    min =calibrate_min(board);
    return max, min

def final_validation(max_value, min_value,board):
    if max_value > 0 and min_value >= 0:
        setValid(1)
        dashboard_edit(max_value,min_value,board)
    else:
        setValid(2)
```

```
def dashboard_edit(max, min, board):
    plt.ion()
    plt.xlabel("Time (s)")
    plt.ylabel("Impedance (ohms)")
    y=[]
    plt.ylim([min,max*10])
    while True:
        y.append(board.readline())
        if len(y) <=1000:
            plt.plot(y)
        else:
            plt.plot(y[-10:])
        plt.pause(0.05)

#Config root

root = Tk()
root.title("Validation User Interface")
root.geometry("1050x850")
root.resizable(1,1)
root.config(bg="brown")

#Config frame
frame=Frame()
frame.pack(side="left",anchor="n")
frame.config(bg="grey")
frame.config(width="950",height="750")
frame.config(bd="10")
frame.config(relief="groove")
```

```

MAXrea = StringVar()
MINrea = StringVar()
Valid = StringVar()

def max_process():
    value = calibrate_max(board)
    MAXrea.set(value)

def min_process():
    value = calibrate_min(board)
    MINrea.set(value)

def recalibrate_process():
    max , min =recalibrate(board)
    MAXrea.set(max)
    MINrea.set(min)

def continue_process():
    max= MAXrea.get()
    min = MINrea.get()
    tempmax = re.findall(r'\d+', max)
    tempmin = re.findall(r'\d+', min)

    final_validation(float(tempmax[0]),float(tempmin[0]),board)

Label(frame,text="Follow the following steps to correctly calibrate your
sensor.",font=(14)).place(x=5,y=5)
Label(frame,text="1. To calibrate the sensor reach the maximum reactance posible and click the
botton",font=(14)).place(x=5,y=35)

botonMAX = Button(frame, text="Calibrate MAX",command=max_process)
botonMAX.place(x=75,y=65)

```

```
botonMAX.config(cursor="hand2")

Label(frame,text="2. To calibrate the sensor reach the minimum reactance posible and click the
boton",font=(14)).place(x=5,y=95)

botonMIN = Button(frame, text="Calibrate MIN",command=min_process)
botonMIN.place(x=75,y=125)
botonMIN.config(cursor="hand2")

Label(frame,text="3. Validate your results",font=(14)).place(x=5,y=160)

ReCal = Button(frame, text="Recalibrate",command=recalibrate_process)
ReCal.place(x=600,y=190)
ReCal.config(cursor="hand2")

Cont = Button(frame, text="Continue",command=continue_process)
Cont.place(x=600,y=220)
Cont.config(cursor="hand2")

Label(frame,text="MAX reactance",font=(14)).place(x=5,y=190)
resEntry=Entry(frame,textvariable=MAXrea)
resEntry.place(x=160,y=190)

Label(frame,text="MIN reactance",font=(14)).place(x=5,y=220)
resEntry=Entry(frame,textvariable=MINrea)
resEntry.place(x=160,y=220)

def setValid(var):
    if var == 1:
        Valid.set("Valid for the use")
    else:
        Valid.set("No valid for the use")
```



```
Label(frame,text="The sensor you developed is:",font=(14)).place(x=5,y=390)
resEntry=Entry(frame,textvariable=Valid)
resEntry.place(x=80,y=420)

root.mainloop()
```

Manual Characterization Program:

Manual of how to use the Picoscope- Characterization program

January 2020

The porpoise of this tool is the characterization of new sensor prototypes to test and verify the functionalities in health and improving the sensing function.

Material

Picoscope 6 software -> From version 6.13 to belong

Picoscope Instrument -> PicoScope 3000, 4000, 5000 and 6000 Series instruments

Characterization pack -> With the 3 scripts, view_4.py, model_2.py, controller_3.py

Python 3 IDLE -> The latest python release of python 3 (Python 3.7)

Python imports -> tkinter module, csv module, numpy module and matplotlib.pyplot module

Previous Installation Guide

First of all, in order to run the tool properly, there are some programs that are necessary. In order to make all the programs work, a Linux-like OS has to be used. For this guide in

particular, ubuntu 18.10 . Will be also necessary be a no-root user with sudo privileges and a firewall. In this guide, we will show you step by step how to install them so, please, follow the instructions carefully:

Programs

The following list of programs must be installed:

Python3

The base of the program is coded using python language, in particular is used python 3.7. Therefore its installation is mandatory. Python3 can be installed using Terminal, executing the following command:

```
>>> sudo apt-get install python3.7
```

Tkinter

Tkinter module will be necessary to generate the User interface. It has to be installed according to Python 3 dependencies. To install it, please run Terminal, write the following command and press “Enter” to execute it:

```
>>> pip3 install tkinter
```

CSV

CSV module will be necessary to generate and read new csv files during the run process. It has to be installed according to Python 3 dependencies. To install it, please run Terminal, write the following command and press “Enter” to execute it:

```
>>> pip3 install csv
```

Numpy

Numpy module will be necessary to make operations with vectors and matrix in an efficient way. It has to be installed according to Python 3 dependencies. To install it, please run Terminal, write the following command and press “Enter” to execute it:

```
>>> pip3 install numpy
```

Matplotlib.pyplot

Matplot module will be necessary to show the graphics in the User Interface. It has to be installed according to Python 3 dependencies. To install it, please run Terminal, write the following command and press “Enter” to execute it:

```
>>> pip3 install matplotlib.pyplot
```

Picoscope 6 Software

The Picoscope 6 Software is a computer software for real-time signal acquisition of Pico Technology oscilloscopes. You can download it in the page: <https://www.picotech.com/downloads> but for the latest release in pyhton3 please run Terminal, write the following command and press “Enter” to execute it:

```
>>> sudo bash -c 'echo "deb https://labs.picotech.com/debian/ picoscope main"
>/etc/apt/sources.list.d/picoscope.list'
```

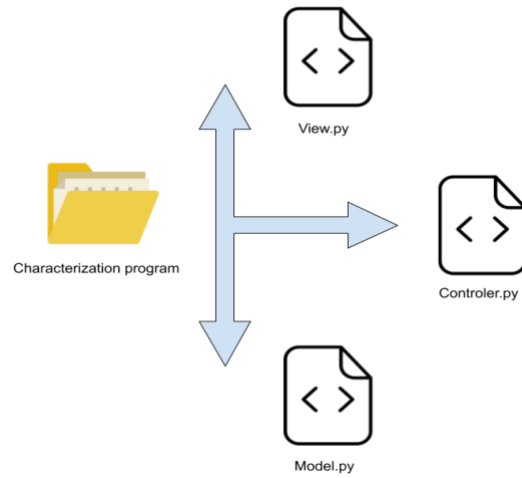
```
>>> wget -O - - https://labs.picotech.com/debian/dists/picoscope/Release.gpg.key | sudo
apt-key add -
```

```
>>> sudo apt-get update
```

```
>>> sudo apt-get install picoscope
```

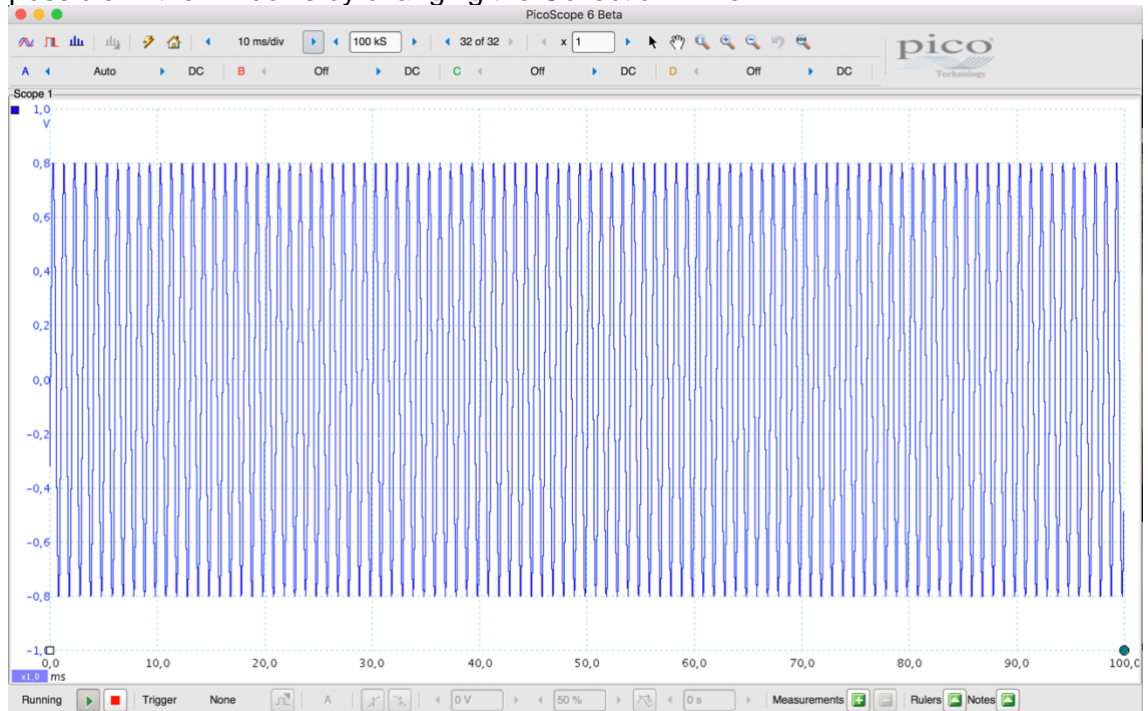
File extraction

In addition to the programs that had to be installed, the scripts that contain the tool itself are necessary. These files can be found inside a folder that has been provided, following the diagram bellow.

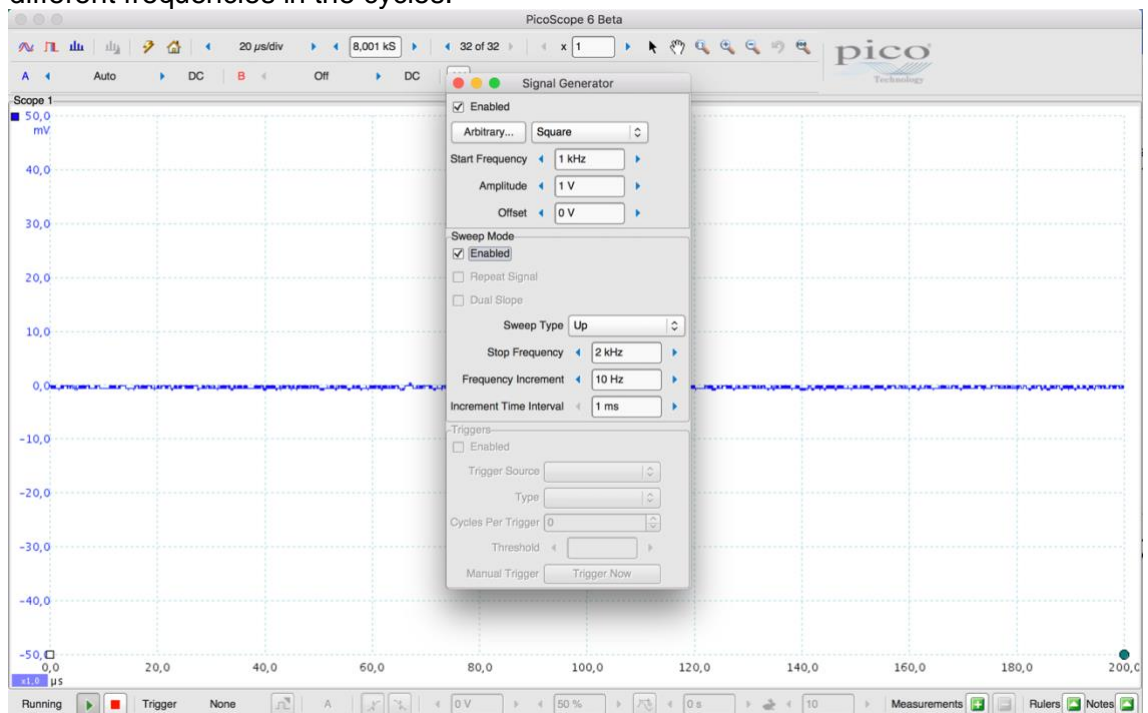


Procedure to run the program

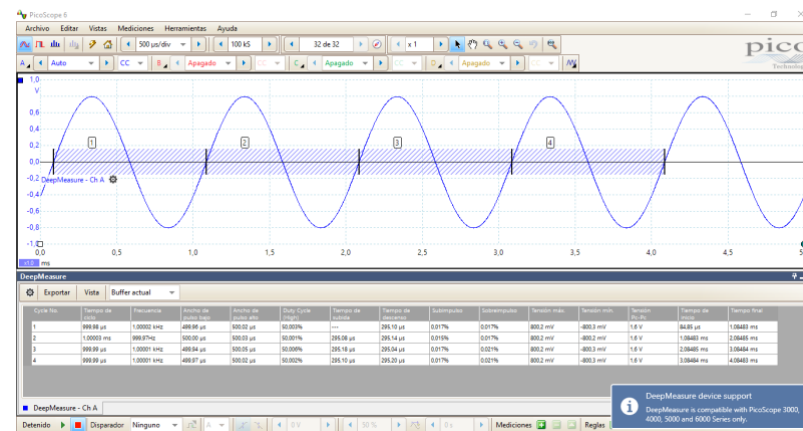
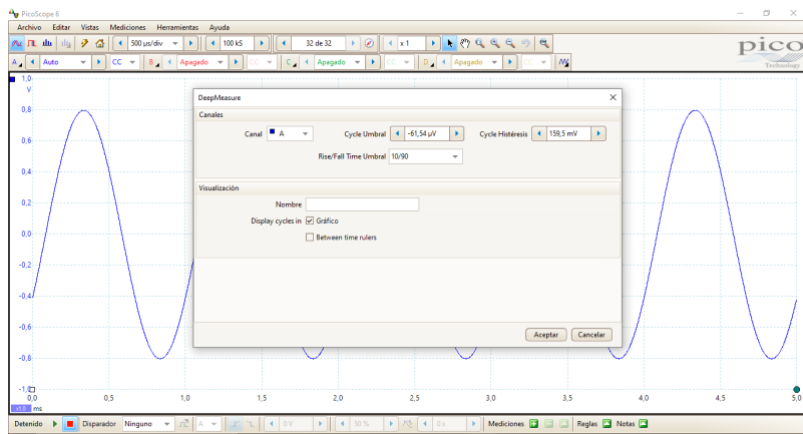
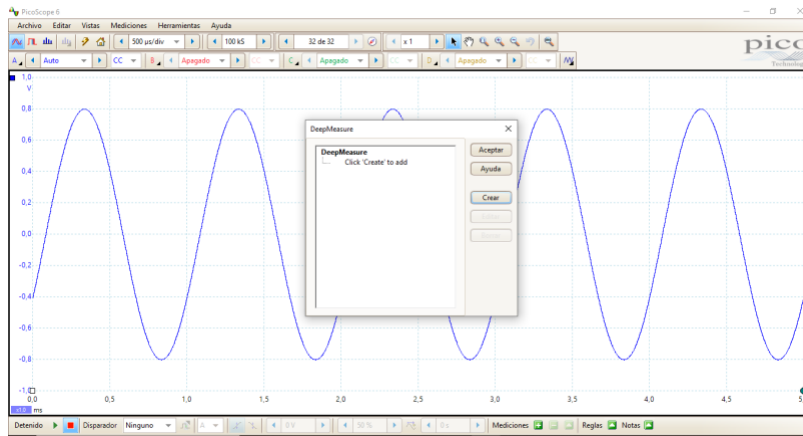
1. Connect the Picoscope device to the computer, make sure you have chosen one of the two different possible models in the circuit adaptation.
2. Open the Picoscope 6 program and set the signal to have as many cycles as possible in the windows by changing the Collection Time

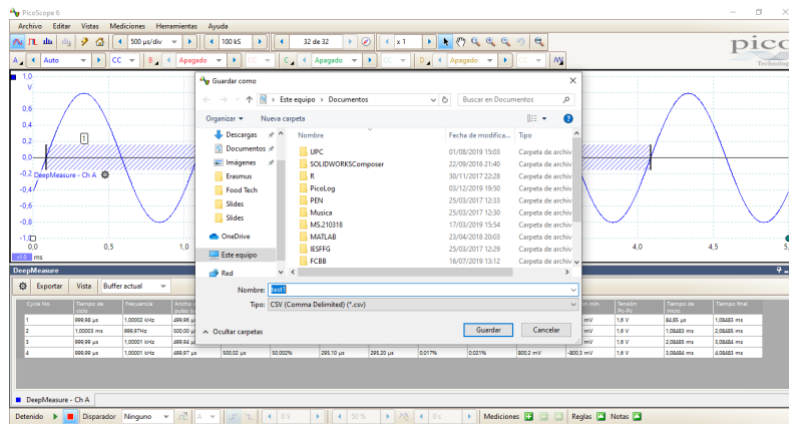


3. Connect the output channel to the signal generator
4. Now change the signal generator to sweep mode so we can see in the window different frequencies in the cycles.



5. Go to the Measurements tab and click to DeepMeasure and create a new measurement and export it as CSV

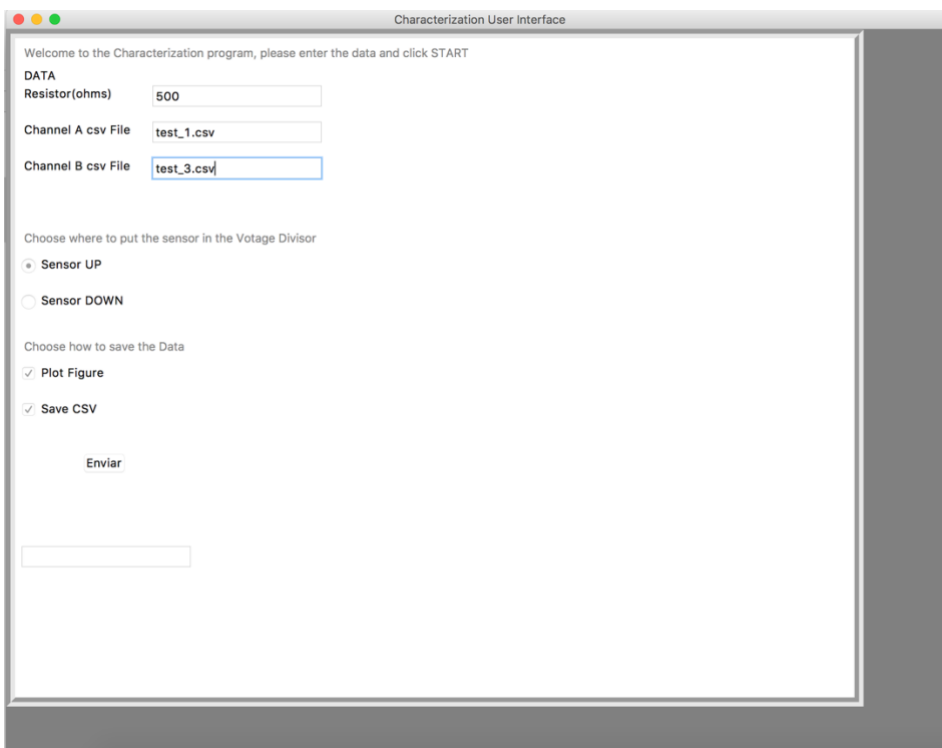




6. Now connect the output channel to the center of the voltage divisor and do the same DeepMeasure process
7. Export a CSV with the new data.
8. Open one terminal and go to the Characterization program folder by using the command

```
>>> cd home/...../Characterization program
```
9. Run the program by writing the next command

```
>>> pyhton3 controller.py
```
10. In the user interface introduce the necessary data:
 - Resistor value in ohms
 - Name and path of the signal generator file
 - Name and path of the output file
 - Select the position of the sensor on the voltage divisor
 - Choose how you want to see the data
11. Click start bottom



Welcome to the Characterization program, please enter the data and click START

DATA

Resistor(ohms)

Channel A csv File

Channel B csv File

Choose where to put the sensor in the Votage Divisor

Sensor UP

Sensor DOWN

Choose how to save the Data

Plot Figure

Save CSV

Class Laboratory:

Example with a Potentiometer as a Resistive sensor

```
//variables
int led1=5;
int led2=6;
int led3=9;
int led4=10;
int led5=11;
int val;

// set-up function
void setup(){
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
  pinMode(led4,OUTPUT);
  pinMode(led5,OUTPUT);
}

// loop function

void loop(){

val=analogRead(A0);

if(val>=0 && val<204) {
  digitalWrite(led1,HIGH);
  digitalWrite(led2,LOW);
  digitalWrite(led3,LOW);
  digitalWrite(led4,LOW);
  digitalWrite(led5,LOW);
}
```

```
if(val>=204 && val<408) {  
    digitalWrite(led1,LOW);  
    digitalWrite(led2,HIGH);  
    digitalWrite(led3,LOW);  
    digitalWrite(led4,LOW);  
    digitalWrite(led5,LOW);  
}
```

```
if(val>=408 && val<612) {  
    digitalWrite(led1,LOW);  
    digitalWrite(led2,LOW);  
    digitalWrite(led3,HIGH);  
    digitalWrite(led4,LOW);  
    digitalWrite(led5,LOW);  
}
```

```
if(val>=612 && val<816) {  
    digitalWrite(led1,LOW);  
    digitalWrite(led2,LOW);  
    digitalWrite(led3,LOW);  
    digitalWrite(led4,HIGH);  
    digitalWrite(led5,LOW);  
}
```

```
if(val>=816 && val<1023) {  
    digitalWrite(led1,LOW);  
    digitalWrite(led2,LOW);  
    digitalWrite(led3,LOW);  
    digitalWrite(led4,LOW);  
    digitalWrite(led5,HIGH);  
}}
```

Arduino Lab instructions

Description or Background

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs either analogic and digital– e.g. *light on a sensor, a finger on a button...*

To learn Arduino a basic understanding of programming is required since the microcontroller needs to be programed in C++. For learning about programing , you need basic computer literacy too, In addition, you also need certain understanding of electronics and understand the functionalities of resistors, diodes, voltages, current etc.

In the appendix you will find links to certain tutorials online about programing

Materials and/or Lab Equipment

- Arduino board
- Arduino coding environment
- Leds
- Potentiometer
- Resistors
- Wire
- Serial

Procedure

1. What is an Arduino?
2. Arduino environment
 - 2.1. Variables
 - 2.1.1. What is a Variable?
 - 2.1.2. Types of Variables in Arduino
 - 2.2. Setup
 - 2.2.1. Pins
 - 2.2.2. Connections
 - 2.2.3. Random
 - 2.3. Loop
3. Application to sensors
 - 3.1. Digital Sensors
 - 3.2. Analog Sensors
4. Basic coding functions
 - 4.1. If – else
 - 4.2. For
 - 4.3. Switch
5. Example with a Potentiometer as a Resistive sensor

Examination: No required.

Appendix – Basic notions of programming

Videos to learn how to use Arduino from scratch:

https://www.youtube.com/playlist?list=PLZfay8jtbyJt6gkkOgeeapCS_UrsgfuJA

Reference to all the functions:

<https://www.arduino.cc/reference/en/>

Tutorials

<https://www.arduino.cc/en/Tutorial/HomePage?from=Main.Tutorials>

Useful general content

<https://www.makeuseof.com/tag/learn-electronics-arduino-just-watching-videos/>

Arduino LAB

Smart Textiles, AT2ST1 HT2019-1

MSc textile engineering program

Tutor: Nuria Revuelta Yus

s194841@student.hb.se

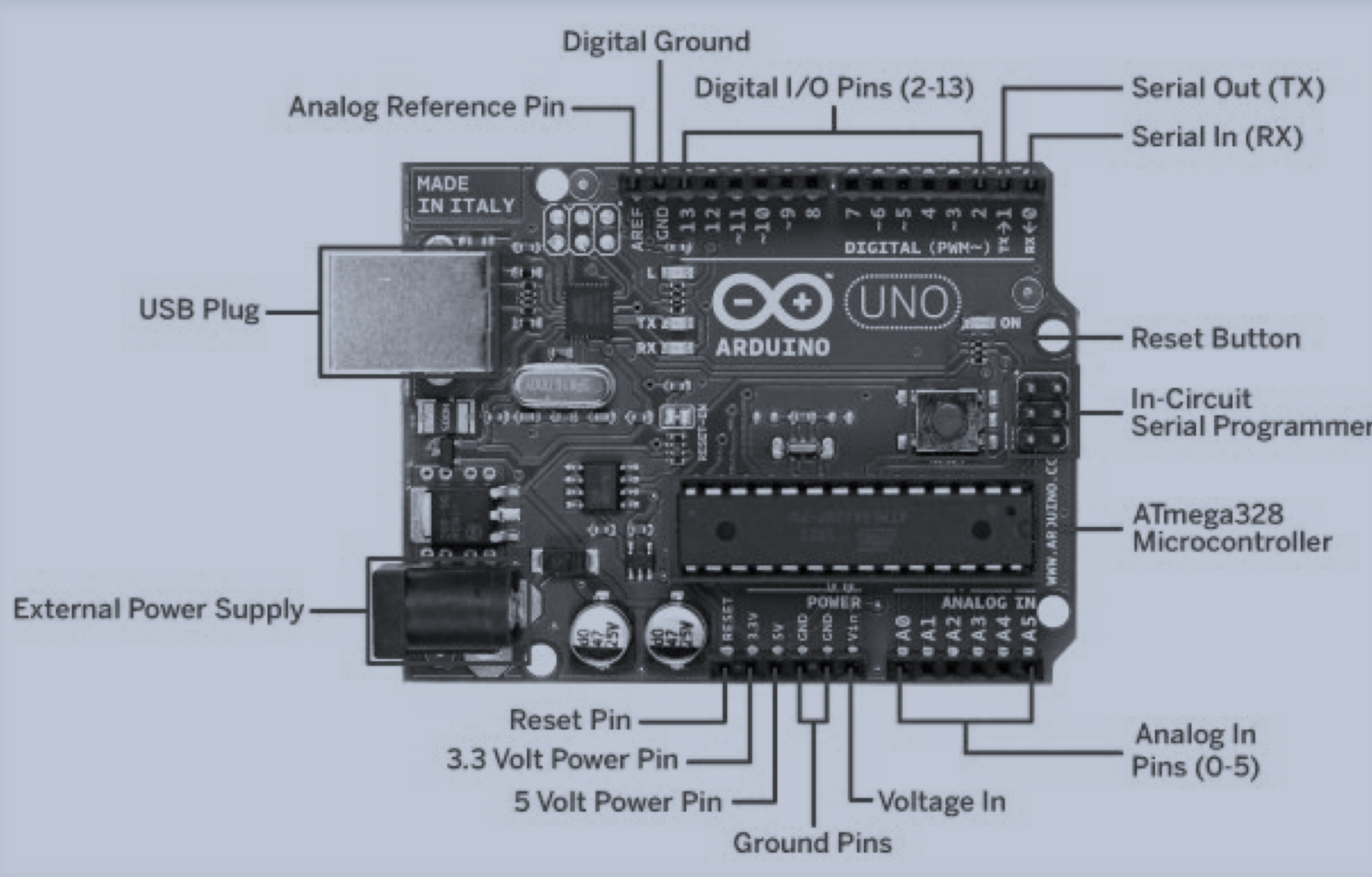
Introduction to Coding

“Programming is the act of instructing computers to carry out tasks”

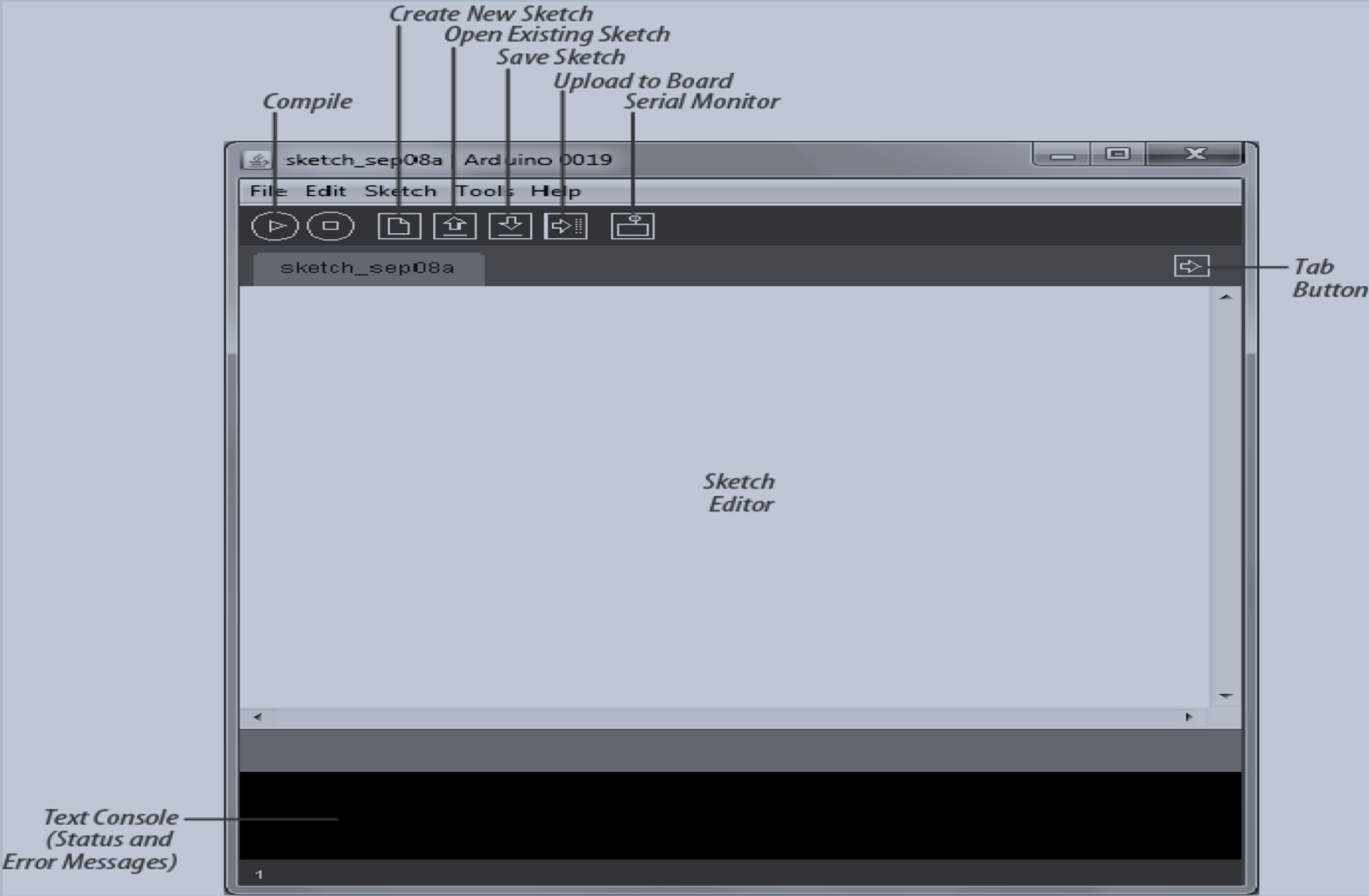
Computer in the definition above is any device that is capable of processing code. This could be smartphones, ATMs, the Raspberry Pi, Servers to name a few.



What is an Arduino?



Arduino enviroment



Variables

In programming, a variable is a value that can change, depending on conditions or on information passed to the program.



Modelo	EEROM disponible
Arduino UNO	1 KB
Arduino Leonardo	1 KB
Arduino Due	No tiene
Arduino Yun	1 KB
Arduino MEGA	4 KB
Arduino Micro	1 KB
Arduino Lilypad	1 KB
Arduino Nano	512 Bytes
Arduino Pro mini	512 Bytes
Arduino Pro	512 Bytes
Arduino Mini	1 KB
Arduino Fio	1 KB

INT

Integers are your primary data-type for number storage.

An int stores a 16-bit (2-byte) value

```
int var = val;
```

FLOAT

Datatype for floating-point numbers, a number that has a decimal point.

```
float var = val;
```

CHAR

A data type used to store a character value. Character literals are written in single quotes,

```
char var = val;
```

EXERCISE 1 : 10 minutes

Set-up

What do we need to set-up:

```
void setup() {...}
```

1. Which pins are going to be used as input and which ones as output:

```
pinMode (A2, OUTPUT) ;  
pinMode (3, OUTPUT) ;  
pinMode (8, INPUT) ;
```

EXERCISE 2 : 5 minutes

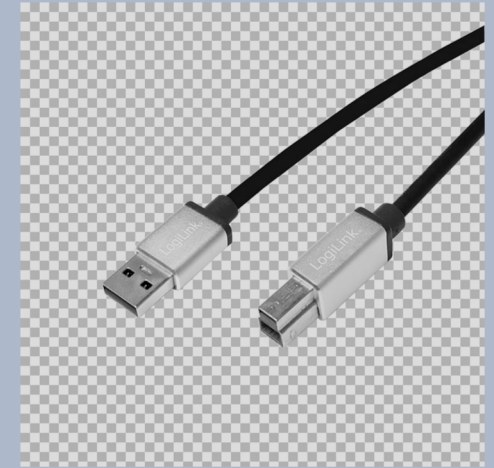
If we are going to want to establish a connection with the computer

```
Serial.begin(9600);
```

EXERCISE 3 : 15 minutes

If we are going to want to use random numbers.

```
randomSeed(0);
```



Loop

In this block you must write all those instructions, orders, primitives, commands or functions necessary for Arduino to function according to our desire.

Actually, this block constitutes an infinite loop.

```
void loop () { ... }
```

EXERCISE 4 : 15 minutes



Application to sensors

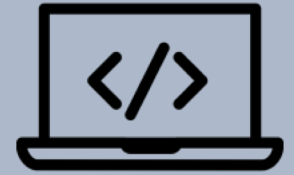
Digital Sensors

```
void loop() {  
  digitalWrite(pinLed, HIGH);  
  delay(t);  
  digitalWrite(pinLed, LOW);  
  delay(t);  
}
```

Analog Sensors

```
void loop() {  
  analogWrite(ledPin, shine);  
  shine = shine + up;  
  if(shine==0 || shine==255) {  
    up = -up;  
  }  
  delay(30);  
}
```

Basic Coding Functions

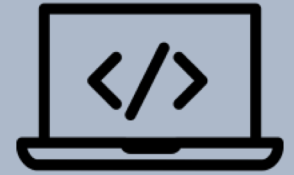


If – else :

```
if (condición1) {...}
else if (condición2) {...}
else if (condición3) {...}
...
else if (condiciónN) {...}
else {...}
```

EXERCISE 5 : 10 minutes

Basic Coding Functions



For :

```
for (int i=1; i<=120; i++) {  
    something(i);  
    delay(50);  
}
```

EXERCISE 6 : 10 minutes

Basic Coding Functions



Switch:

```
switch (val) {  
  case 3: ...;  
  break;  
  case 12: ...;  
  break; ...  
  default: ...;  
}
```

EXERCISE 7 : 10 minutes

Example with a Potentiometer as a
Resistive sensor

EXERCISES

Arduino LAB

Smart Textiles, AT2ST1 HT2019-1
MSc textile engineering program
Tutor: Nuria Revuelta Yus
[\\$194841@student.hb.se](mailto:$194841@student.hb.se)

DOWNLOAD ARDUINO IDE

<https://www.arduino.cc/en/Main/Software>

EXERSICE 1



Code a program that calculates the result of this operation:

$$\text{Result} = (7.34 + 4) * 9.2$$

EXERSICE 2



We are doing some testing on an analog sensor and we decide to use A2 as a Input and A0 as an output, what kind of set-up do we need?

EXERSICE 3



Begin the serial communication in your computer
and see the plot platform

EXERSICE 4



Code a program that chages a number between 1 to 10 every few seconds and view it in your computer with the serial port.

EXERSICE 5



Code a program that detects if a random number is between 0-10, 10-20, 20-30,... up to 50 with if-else

Help:

```
randomNumber = random(1,50);
```

EXERSICE 6



Write a program in Arduino to display the first 10 natural numbers.

View the results in the serial ploter.

EXERSICE 7



Code a program that detects if a random number is between 0-10, 10-20, 20-30,... up to 50 with switch