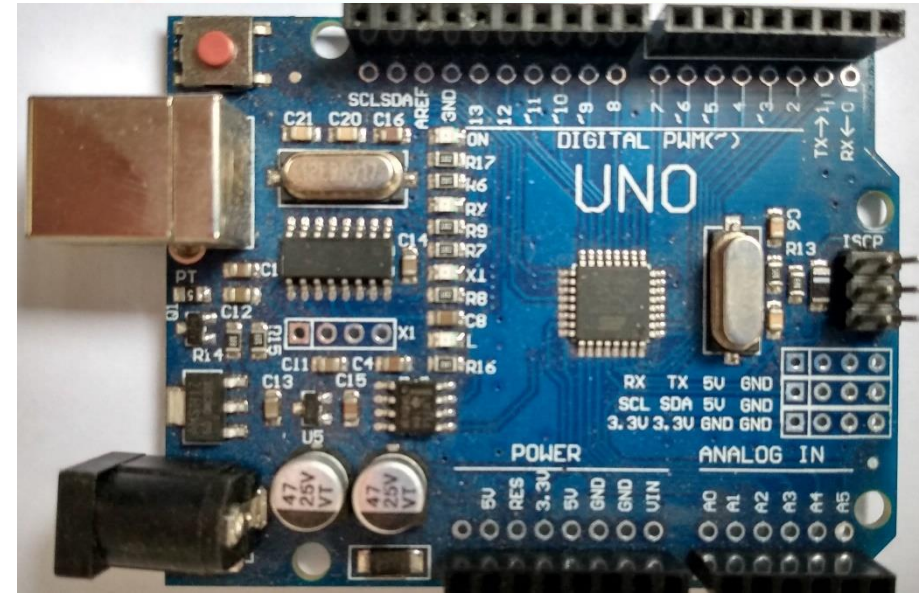




LABORATORIO 3 - DF - FCEyN - UBA
1er cuatrimestre de 2022

Arduino es una plataforma de creación de electrónica de software y código abierto, la cual está basada en una placa con todos los elementos necesarios para **conectar periféricos a las entradas y salidas de un microcontrolador**, y que puede ser programada en software libre, flexible y fácil de utilizar: [Arduino IDE \(Entorno de Desarrollo Integrado\)](#)

El proyecto **nació en 2003**, impulsado por **estudiantes** del Instituto de Diseño Interactivo de Ivrea, **Italia**, con el fin de facilitar el acceso y uso de electrónica y programación.



```
analogRead_slow | Arduino 1.8.13
File Edit Sketch Tools Help
analogRead_slow $
int bufVal = 75; //Virtual DC Offset for better oscilloscope visibility

void setup() {
  Serial.begin(115200);
}
void loop() {
  Serial.write(analogRead(A0)+bufVal);
}

Done Saving.
11 Arduino Uno on COM4
```

Arduino UNO

Technical specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

- Placa basada en un microcontrolador ATMEL (ATmega328 o ATmega328P):
 - + 6 canales analógicos de entrada
 - + 14 canales digitales (alto: 5 V, bajo: 0 V)
 - + AVR (Advanced Virtual Risk)
 - + 32 registros de 8 bits
 - + ADC x aproximaciones sucesivas
- Resolución: 10 bits
- Muestreo: = ajustable
Rango: 0 a +5 V (ajustable a 1.1 V)
- Permite calibraciones externas
- Comunicación: serie (vía USB)

Arduino UNO

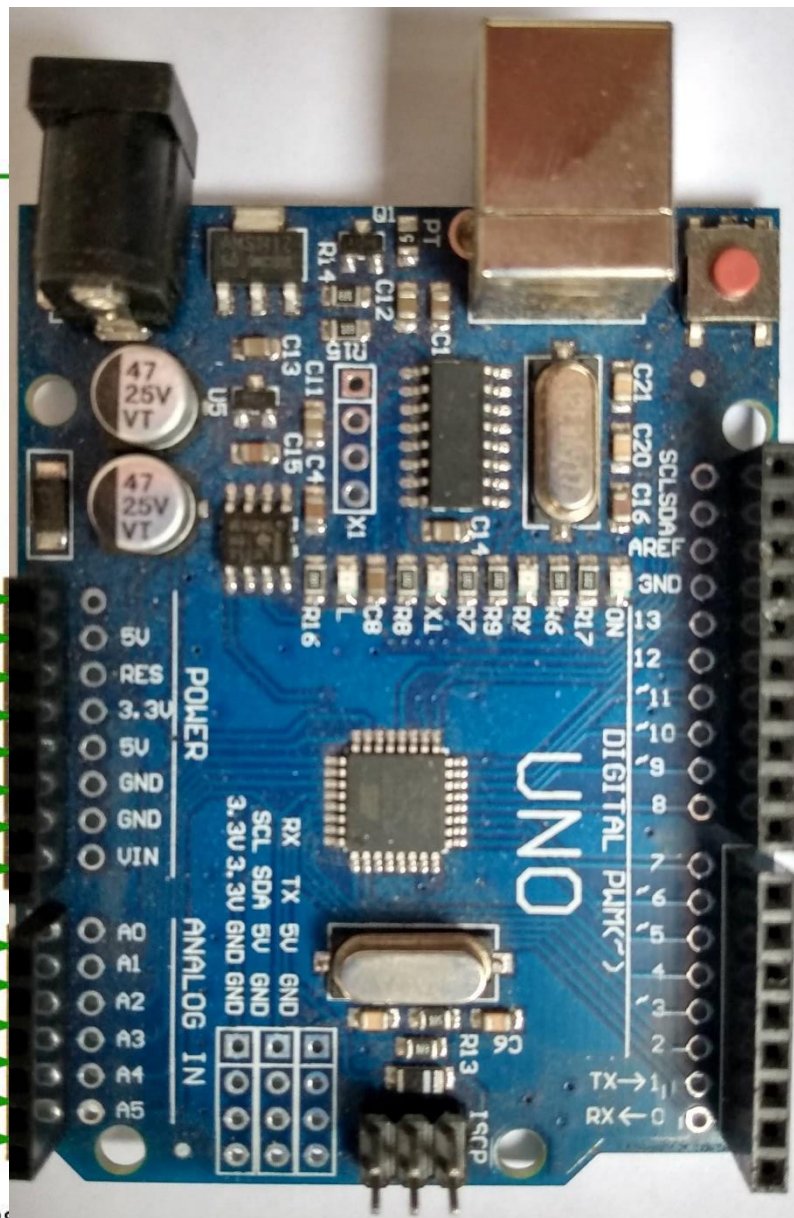
Alimentación (entrada)
Fuente externa

7 to 12VDC input
2.1mm x 5.5mm
Male center positive

Alimentación (salida)

Alimentación (entrada)
x ej. pila

Entradas analógicas (ADC)
0 V < Vin < 5 V



ATmega16U2
microcontroller IC/USB controller

USB-B port
to computer

Reset button

ICSP for
USB interface

(I2C) SCL - Serial clock

(I2C) SDA - Serial data

Pin-13 LED

(SPI) SCK - Serial clock

(SPI) MISO - Master-in, slave-out

(SPI) MOSI - Master-out, slave-in

(SPI) SS - Slave select

Note: Pins denoted with "~"
are PWM supported

Interrupt 1

Interrupt 2

TXD

RXD

Not connected

I/O Reference voltage

Reset

3.3V Output

5V Output

Ground

Ground

Input voltage

Analog pin 0

Analog pin 1

Analog pin 2

Analog pin 3

(I2C) SDA

(I2C) SCL

ATmega328P
microcontroller IC

ICSP for
ATmega328P

VCC

MOSI

GND

RESET

SCK

MISO

Comunicación Serie (USB)

+ potencia
+ lenta
Comunicación con otros periféricos
+ rápida
- potencia

Pulse Width Modulation

Interrupciones externas
Comunicación Serie

14 Entradas/Salidas digitales
6 salidas PWM
(5V - 20 mA)

Para programar otros microcontroladores →

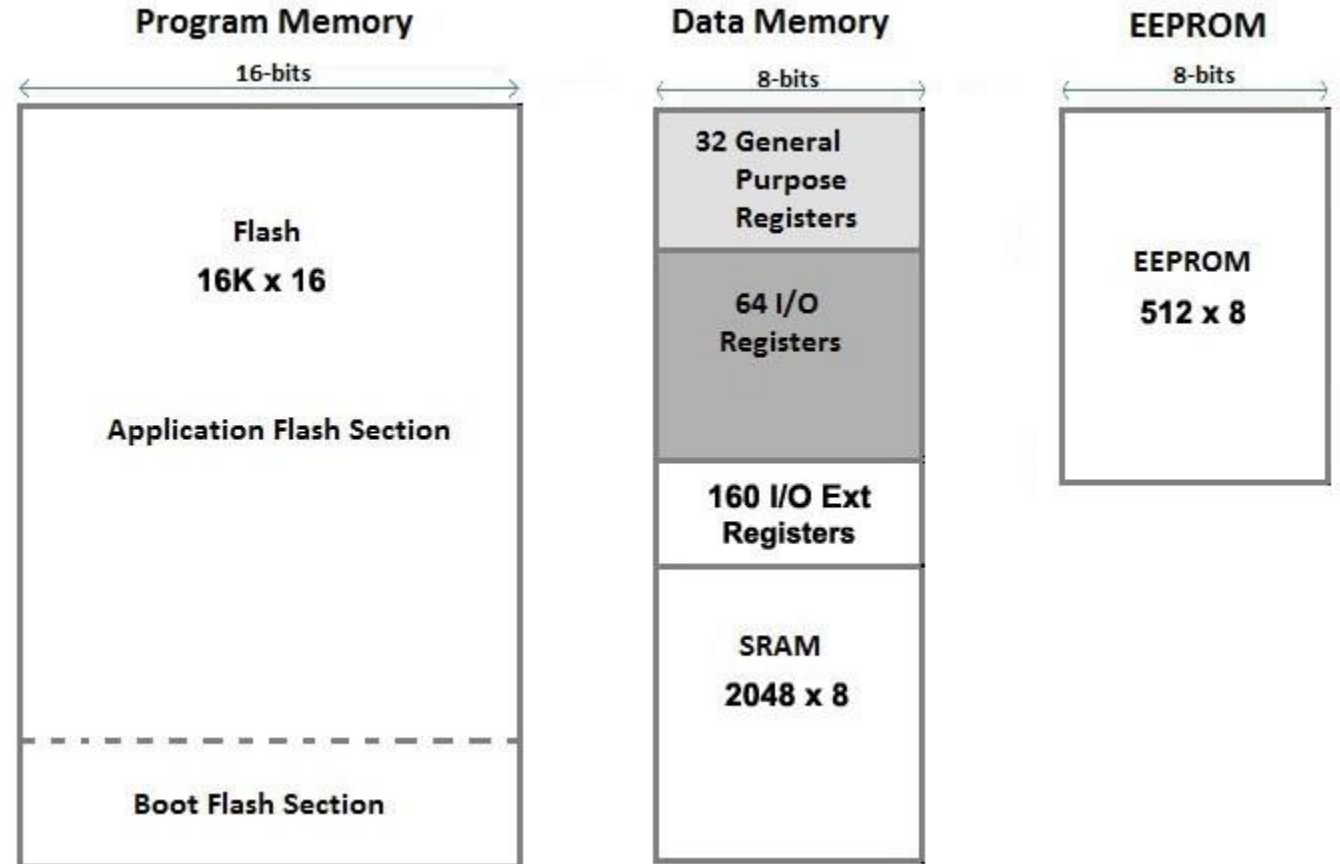
Arduino UNO

Memoria Del ATmega328

La **memoria flash** tiene una capacidad de 32 kB. Tiene una dirección de 15 bits. Es una memoria programable de solo lectura (ROM). Es memoria no volátil. **El programa** se almacena en esta memoria.

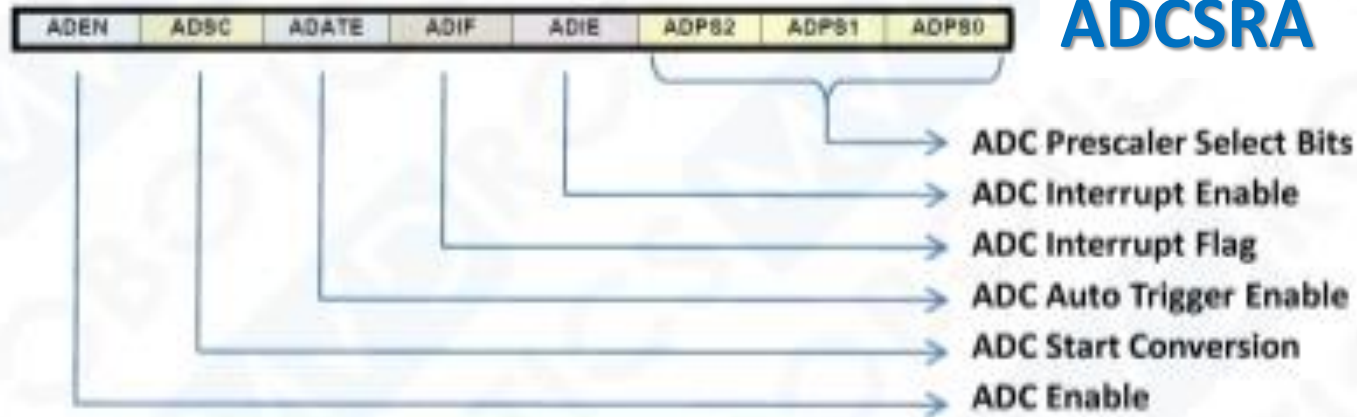
SRAM significa memoria estática de acceso aleatorio. Es una memoria volátil, es decir, los datos se eliminarán después de quitar la fuente de alimentación. **Las variables del programa** se almacenan en esta memoria.

EEPROM significa memoria de solo lectura programable y borrable eléctricamente. Tiene datos a largo plazo. **Datos de configuración** se almacenan en esta memoria.



Registros del ADC (ATmega168/328P)

ADCSRA – ADC Control and Status Register A



ADCSRA

`ADCSRA = (ADCSRA & 0xf8) | 0x04;`

AND

Hexa

`1111 1000 OR 0000 0100`

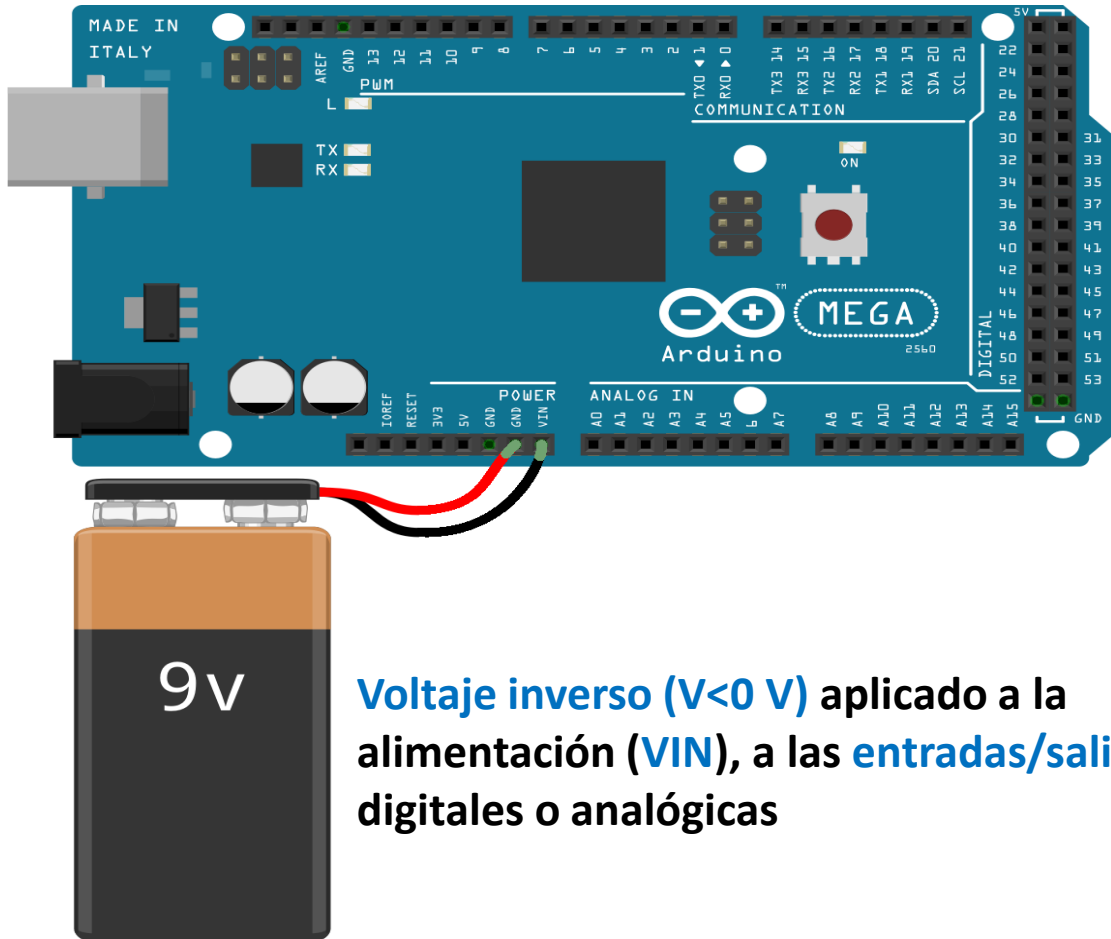
`1111 1100`

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

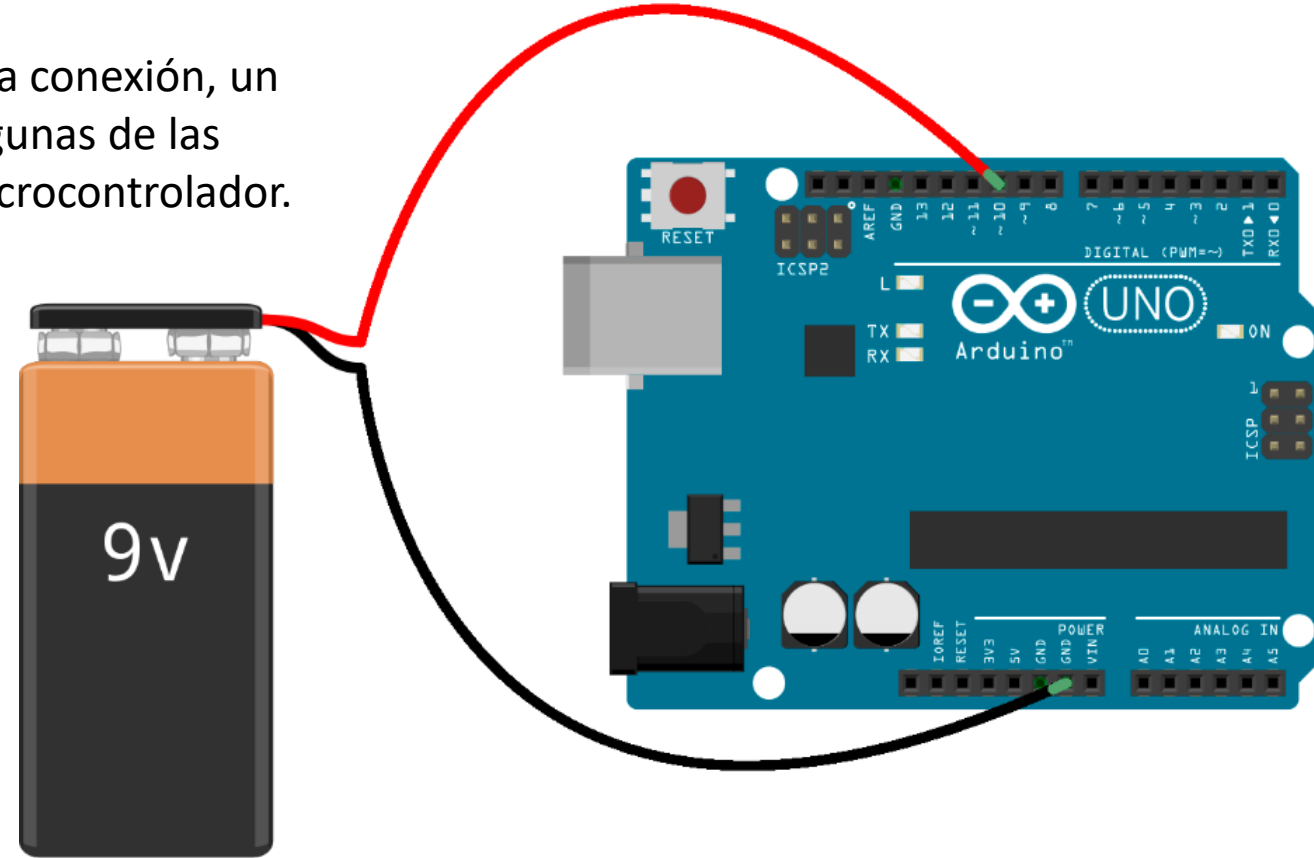
ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Cómo destruir un Arduino...o todo lo que hay que evitar!!!

Existen muchas formas de destruir un Arduino. Una mala conexión, un sobrevoltaje o un exceso de corriente son solamente algunas de las principales razones que llevan a la destrucción de un microcontrolador.



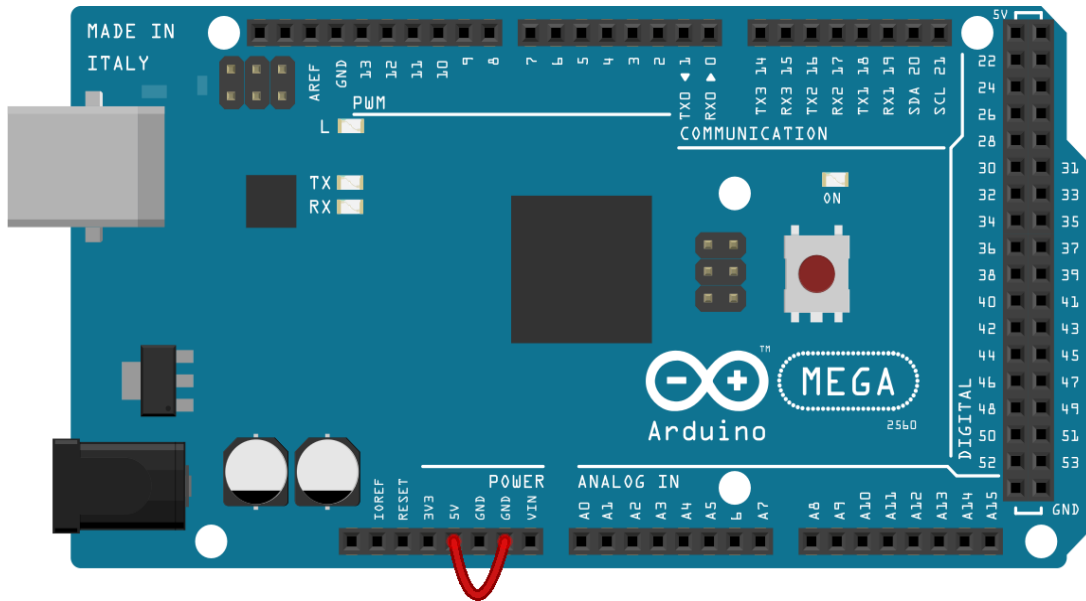
Voltaje inverso ($V < 0\text{ V}$) aplicado a la alimentación (VIN), a las **entradas/salidas digitales** o analógicas



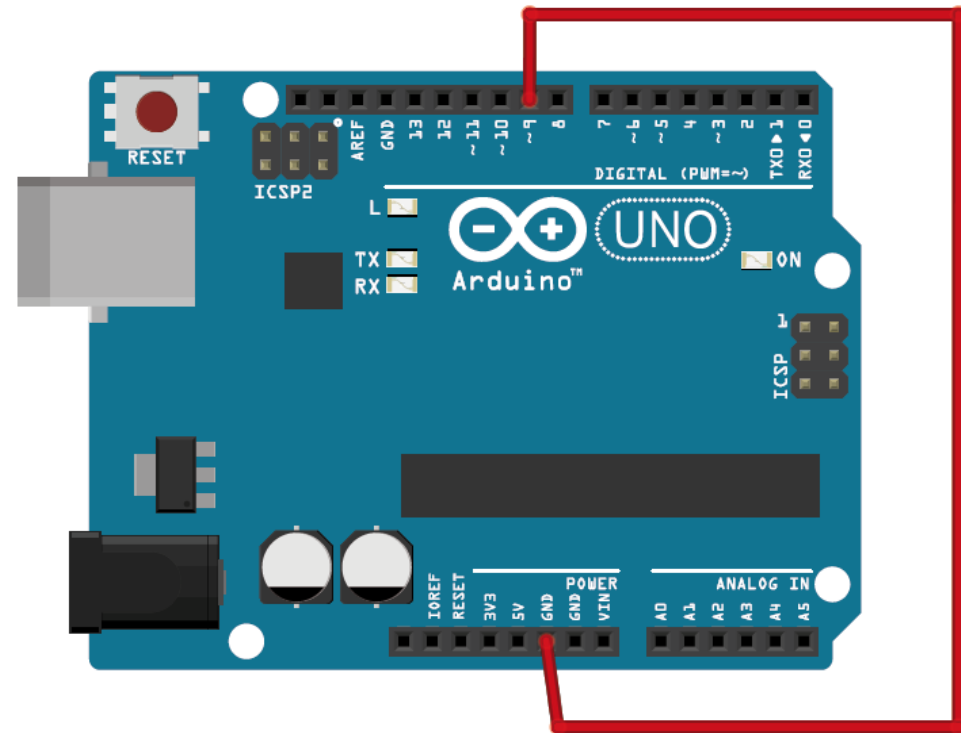
Sobrevoltaje ($V > 5\text{ V}$) aplicado a un pin **analógico/digital** o a un pin de **alimentación (VIN)**

Cómo destruir un Arduino...o todo lo que hay que evitar!!!

Existen muchas formas de destruir un Arduino. Una mala conexión, un sobrevoltaje o un exceso de corriente son solamente algunas de las principales razones que llevan a la destrucción de un microcontrolador.



Cortocircuito en placa!



Sobrecorriente en pin digital (> 20 mA)
Para evitarla, usar siempre una $R \geq 220$ ohm para este tipo de conexión

Programar Arduino UNO

Programa de placa Arduino



Serial Plot

Instalación / Programación

Guía de instalación para Windows y otros

<https://www.arduino.cc/en/Guide/Windows>

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

(seguir las instrucciones)

Guía para la instalación específica de UNO

<https://www.arduino.cc/en/Guide/ArduinoUno>

Guía para la programación (en Español)

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

Adicionales

Aprendiendo Arduino (Curso)

<https://aprendiendoarduino.wordpress.com/>

Algo sobre microcontroladores

<https://www.newbiehack.com/MicrocontrollersADC10Bits.aspx>

<https://hetpro-store.com/TUTORIALES/adc-del-atmega8/>

Programar Arduino UNO

The image shows the Arduino IDE interface with the 'Herramientas' menu open. The menu items are:

- Auto Formato (Ctrl+T)
- Archivo de programa.
- Reparar codificación & Recargar.
- Administrar Bibliotecas... (Ctrl+Mayús+I)
- Monitor Serie (Ctrl+Mayús+M)
- Serial Plotter (Ctrl+Mayús+L)
- WiFi101 / WiFiNINA Firmware Updater
- Placa: "Arduino Uno" (selected)
- Puerto
- Obtén información de la placa
- Programador: "AVR ISP"
- Quemar Bootloader

The 'Placa: "Arduino Uno"' option is expanded, showing a list of boards:

- Gestor de tarjetas...
- Arduino ARM (32-bits) Boards >
- Arduino AVR Boards > (selected)
- Arduino Yún
- Arduino Uno (selected)
- Arduino Duemilanove or Diecimila
- Arduino Nano
- Arduino Mega or Mega 2560
- Arduino Mega ADK
- Arduino Leonardo
- Arduino Leonardo ETH
- Arduino Micro
- Arduino Esplora
- Arduino Mini
- Arduino Ethernet
- Arduino Fio
- Arduino BT
- LilyPad Arduino USB
- LilyPad Arduino
- Arduino Pro or Pro Mini
- Arduino NG or older
- Arduino Robot Control
- Arduino Robot Motor
- Arduino Gemma
- Adafruit Circuit Playground
- Arduino Yún Mini
- Arduino Industrial 101
- Linino One
- Arduino Uno WiFi

The IDE window title is 'sketch_sep03a Arduino 1.8.13'. The code editor shows the following code:

```
void setup()
// put your

}

void loop() {
// put your

}
```

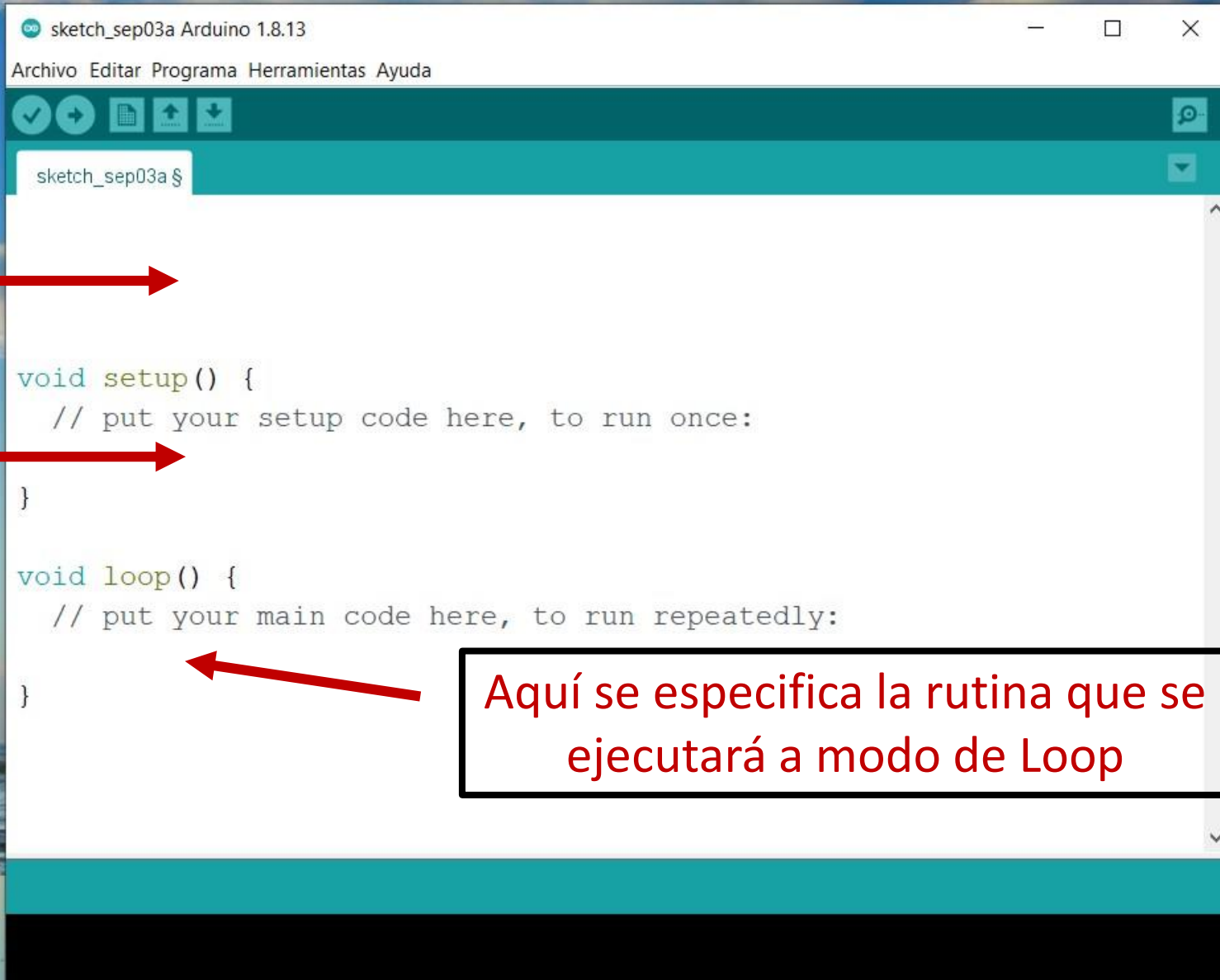
The Windows taskbar at the bottom shows the search bar with the text 'Escribe aquí para buscar', the system tray with the time '20:35' and date '3/9/2020', and the status 'Arduino Uno en COM4'.

Programar Arduino UNO

Programación en C:

Aquí se declaran las variables
(globales) y su tipo

Aquí se determinan algunos
parámetros ligados al
microcontrolador, los pines y
las comunicaciones



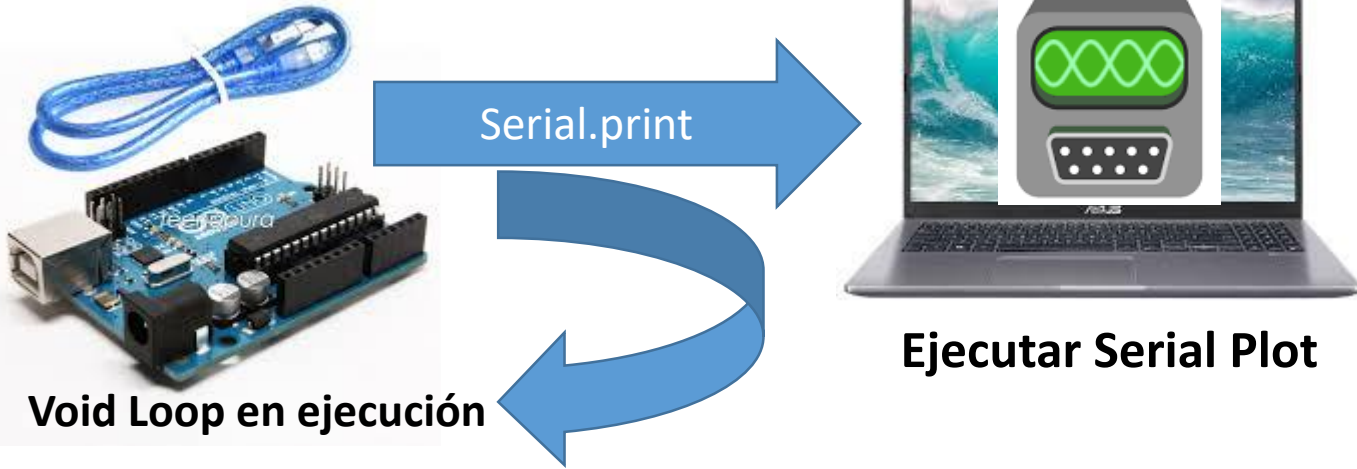
```
sketch_sep03a Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda
sketch_sep03a §

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Aquí se especifica la rutina que se
ejecutará a modo de Loop

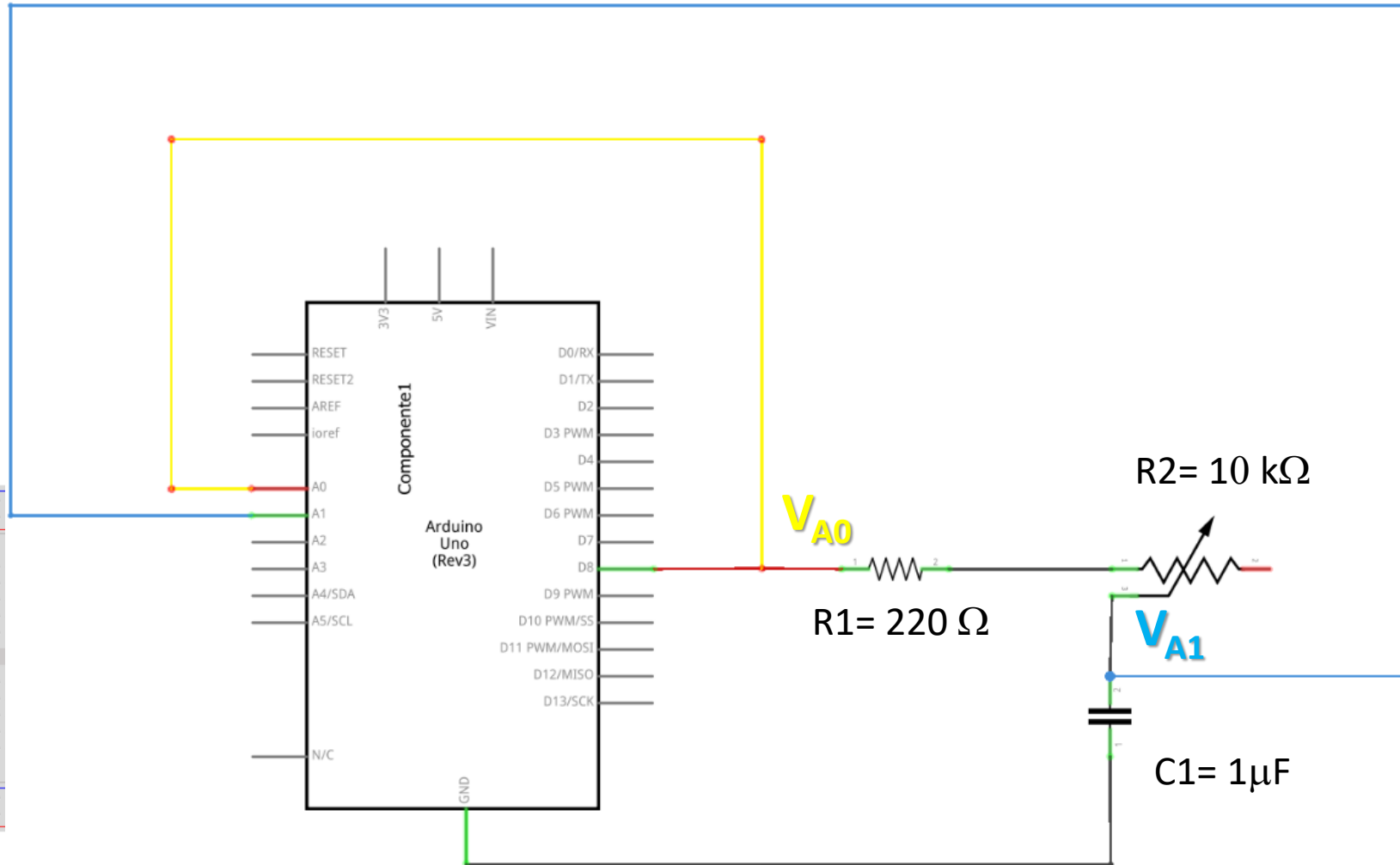
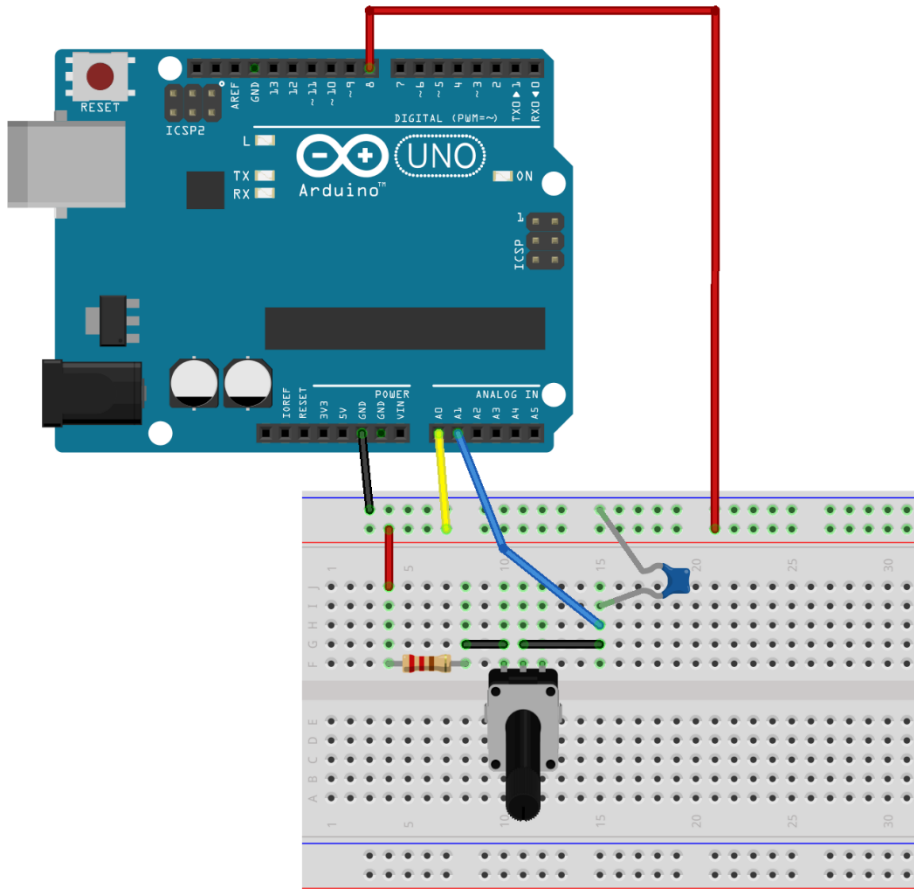
Transferir mediciones de Arduino UNO



- 1) Dejar ejecutándose el IDE de Arduino
- 2) Chequear los datos adquiridos en el Monitor Serie
- 3) Ejecutar el Serial Plot (cerrar previamente el Monitor Serie o el Serial Plotter del IDE (si los tuvieran abiertos))
- 4) Igualar el “Baud Rate” en el “Port” y Setear “Data Format” en ASCII si hiciera falta. Clickear OPEN
- 5) Visualizar los datos
- 6) Grabar en archivo en “Record”: seleccionar nombre archivo y luego clickear para iniciar y para finalizar sobre RECORD → LISTO! Los datos están en un archivo tipo CSV

```
for (int i=0; i<130; i++){
    Serial.print(tiempo[i]/1000000.00000,5); //
    tiempo en segundos con 5 dígitos de resolución
    Serial.print(',');
    V1 = canal1[i]*1.1/1023;
    Serial.print(V1,3) ;
    Serial.print(',');
    V2 = canal2[i]*1.1/1023;
    Serial.print(V2,3);
    Serial.print(',');
    Serial.print(Amplitud1,3);
    Serial.print(',');
    Serial.print(Amplitud2,3);
    Serial.print(',');
    Serial.print(frec1,1);
    Serial.print(',');
    Serial.print(Amplitud2/Amplitud1,3);
    Serial.print(',');
    Serial.println(defasaje,1);
}
delay(10000); // espera de 10 s para reiniciar las mediciones
```

El circuito RC serie: estudio experimental



El circuito RC serie: estudio experimental

Transitorio_RC_Lab3 Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

```

/* LABORATORIO 3 - DF - FCEyN - UBA
 * Este programa permite medir los voltajes de las entradas
 * analógicas A0 y A1 en función del tiempo
 * Enciende a alto (5 V) una salida digital (pin 8) para medir el transitorio de un cir
 * RC y LC y mide 60 veces la evolución de estos 2 voltajes. Luego aplica un valor bajo
 * (0 V) y repite las 60 mediciones. Finalmente envía por el puerto serie las
 * 120 mediciones del tiempo y de ambos voltajes
 */

int V0[160], V1[160];
long tiempo[160];
int espera=0; // Cambiar para ajustar la ventana de datos a la frecuencia medida // 200
/* CAMBIAR "espera" según la frecuencia de la señal a medir: espera=0--> 1 medición cada
 * Para freq< 1000 Hz // 100 161-450 Hz // 300 74-160 Hz // 1000 26-73 Hz // 3000 9
 * Para freq> 1000 Hz fijar espera=0 y cambiar el prescaler de 04 a 03 como se indica má
 */

void setup() {

Serial.begin(57600); // velocidad de transferencia de la interfaz serie
pinMode(8, OUTPUT); // fija al pin digital 8 como Salida

/* CAMBIAR "0x04" o "0x03" según la frecuencia de la señal a medir
 * USAR "0x04" para frecuencias < 1000 Hz y regular con "espera" para frecuencias bajas
 * USAR "0x03" para frecuencias >= 1000 Hz con espera=0
 */

ADCSRA = (ADCSRA & 0xf8) | 0x03; // clear prescaler | set prescaler to /16
ADCSRA |= 1<<ADEN; // make sure ADC is enabled

```

Se miden 80 tríos de datos (t, VA0, VA1) para el semi-ciclo de carga (Pin 8 "HIGH") y otros 80 tríos de datos para el semi-ciclo de descarga (Pin 8 "LOW")

Recomendación para ajustar el valor de "espera":
espera=0 → mide 80 datos (t, VA0, VA1) en 4 ms

Espera=n μs de espera entre cada terna de medición
→ 80 x n μs. Si n=200 → (80X0.2 + 4)ms = 20 ms para los 80 datos (semi-ciclo)

Guardado.

El Sketch usa 4258 bytes (13%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 1512 bytes (73%) de la memoria dinámica, dejando 536 bytes para las variables locales. El máximo es 2048 bytes.

79

Escribire aquí para buscar

Escritorio 20:04 24/9/2020

Arduino Uno en COM4

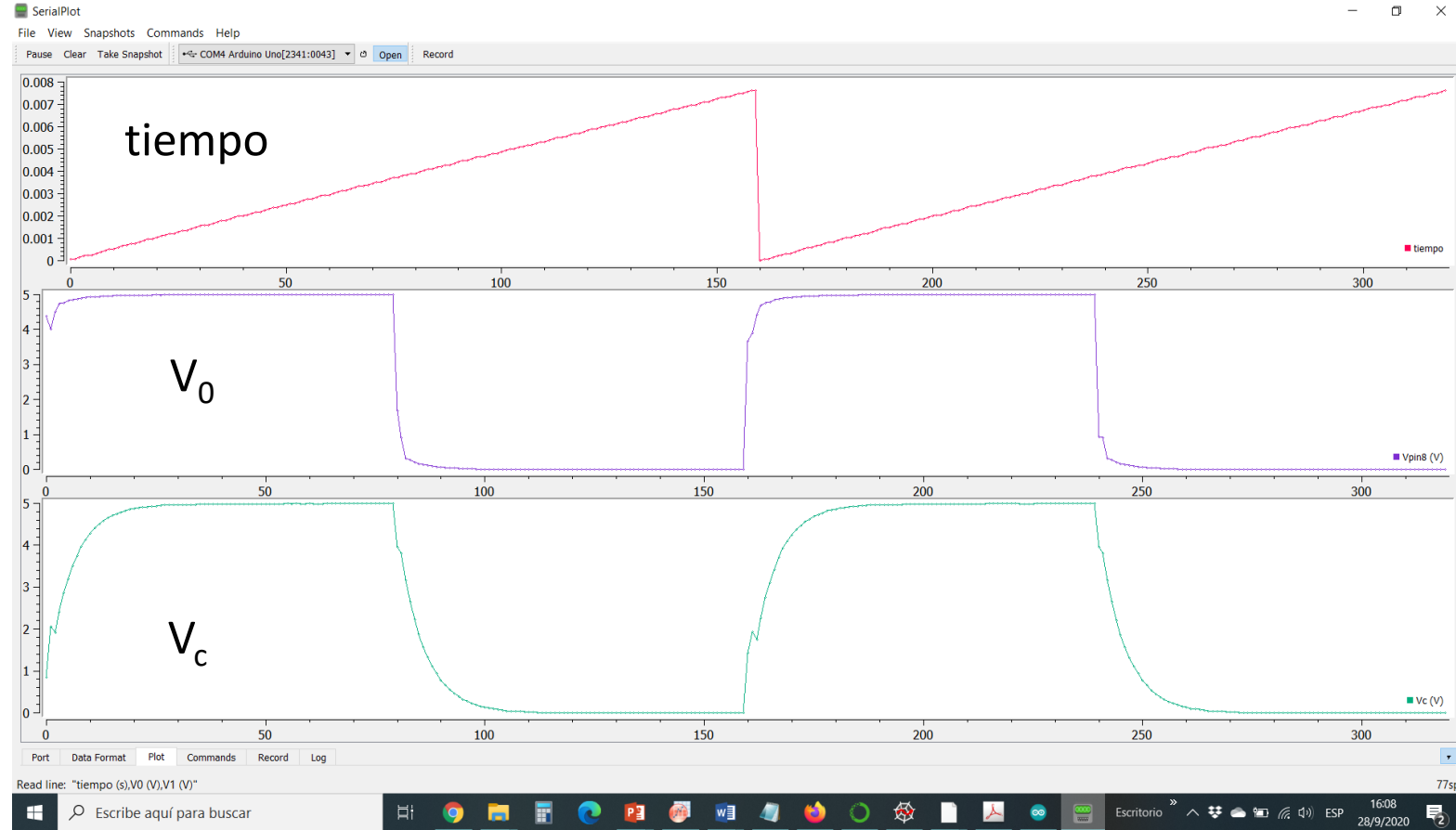
El circuito RC serie: estudio experimental

Tiempo de medición **OK**
respecto del tiempo
característico

Se observa la saturación
de $V_{A1}(t)$

Es decir $T_{med} > 4 \tau$

→ No cambiar “espera”



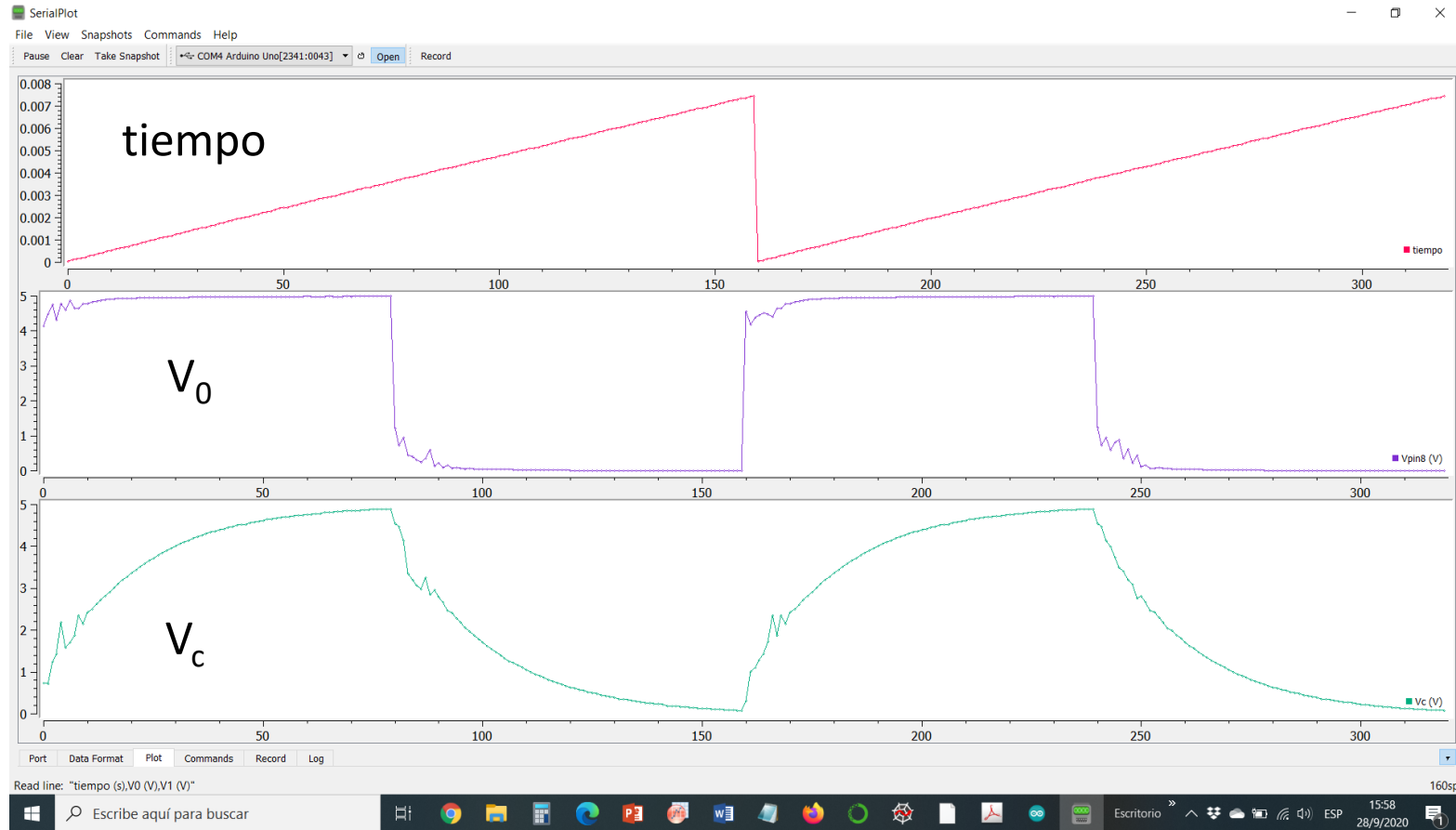
El circuito RC serie: estudio experimental

Tiempo de medición **corto**
respecto del tiempo
característico

No se observa la saturación
de $V_{A1}(t)$

Es decir $T_{med} < 4 \tau$

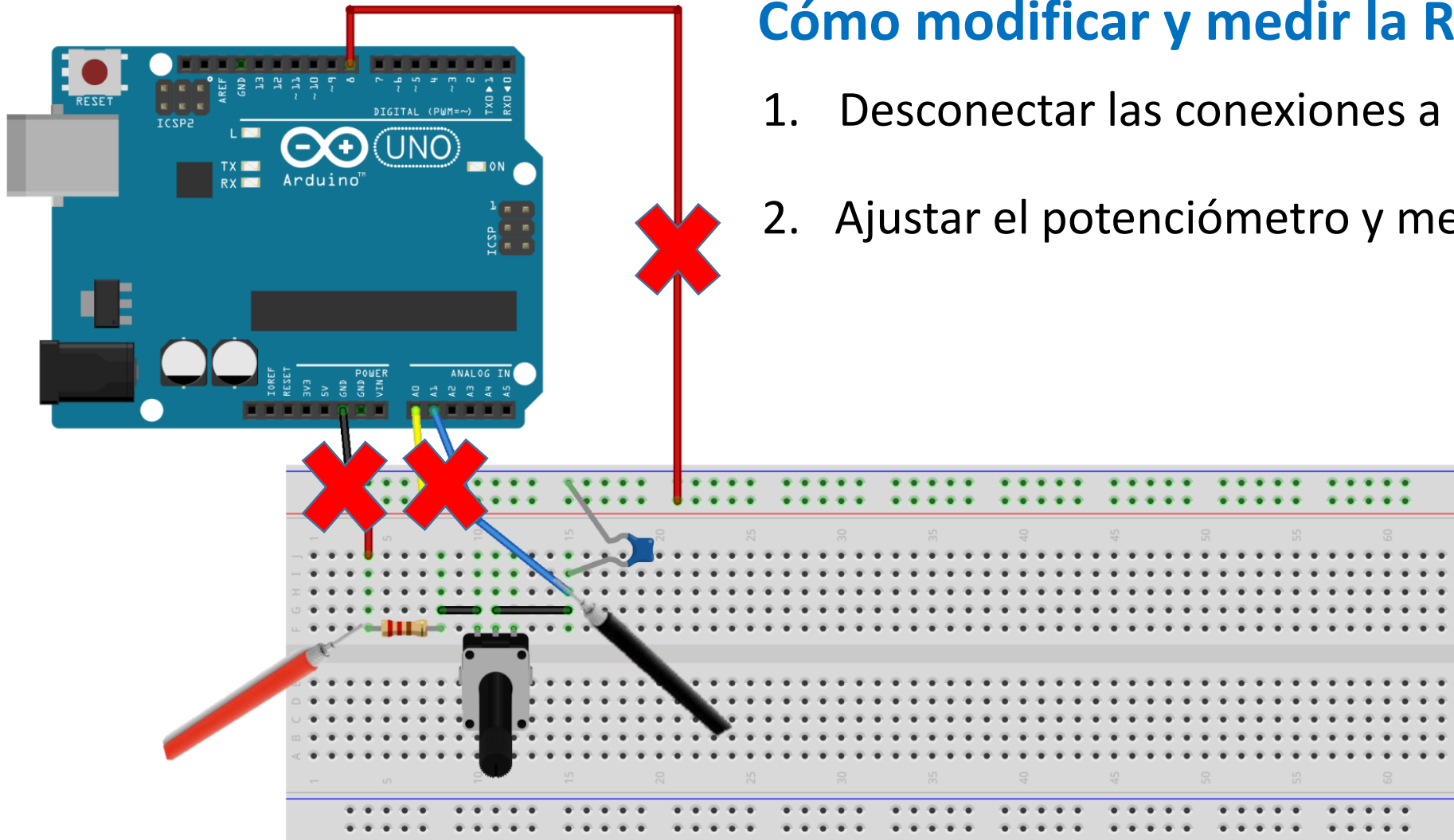
→ Aumentar “espera”



El circuito RC serie: estudio experimental

Cómo modificar y medir la R total?

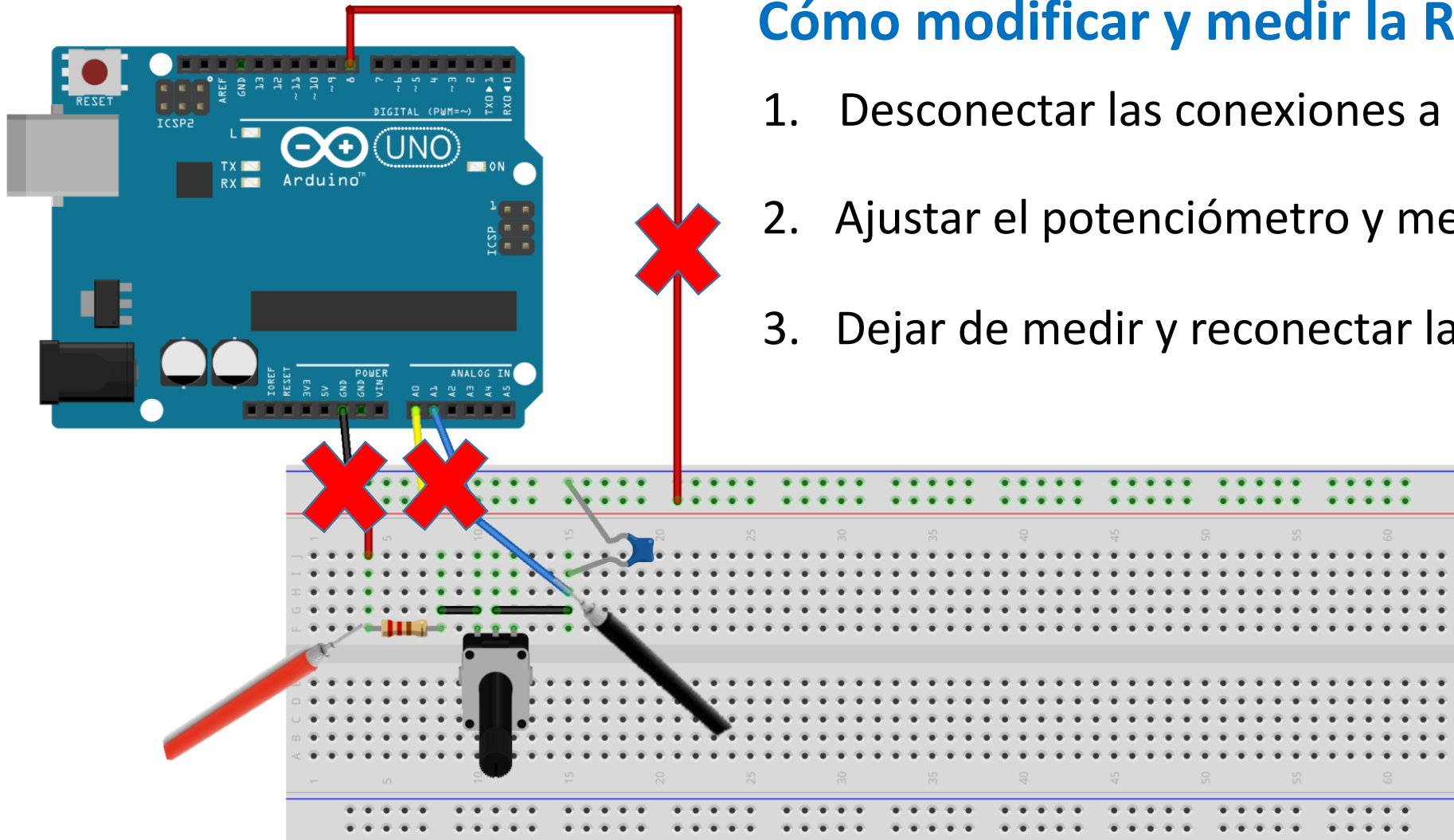
1. Desconectar las conexiones a Arduino
2. Ajustar el potenciómetro y medir la R total



El circuito RC serie: estudio experimental

Cómo modificar y medir la R total?

1. Desconectar las conexiones a Arduino
2. Ajustar el potenciómetro y medir la R total
3. Dejar de medir y reconectar las conexiones a Arduino



El circuito RC serie con Arduino: un desafío a la sistematicidad!

Pasos:

1. Abrir el IDE de ARDUINO (Transitorio_RC_Lab3.ino) y compilarlo (cargarlo en Arduino)
2. Abrir el SerialPlot: setear plot y fijar nombre de los archivos de salida
3. Fijar R2 y medir R1+R2 con el multímetro
4. Conectar los pines de Arduino
5. Monitorear la señal con el SerialPlot
6. Si se observa la saturación de Vc seguir con **7** / caso contrario → ir a **9**
7. Grabar (prop. de indexación automática) archivo con 3-4 ciclos de carga y descarga
8. Volver a **3**

Pasos':

9. “Cerrar el OPEN” del SerialPlot
10. Cambiar el valor de la variable “espera” en el IDE de ARDUINO (Transitorio_RC_Lab3.ino)
11. Compilar este IDE (cargarlo en Arduino)
12. “Abrir el OPEN” el SerialPlot
13. Monitorear la señal con el SerialPlot
14. Si se observa la saturación seguir con **7** / caso contrario volver a **9**

Scripts de Python para graficar y analizar los datos

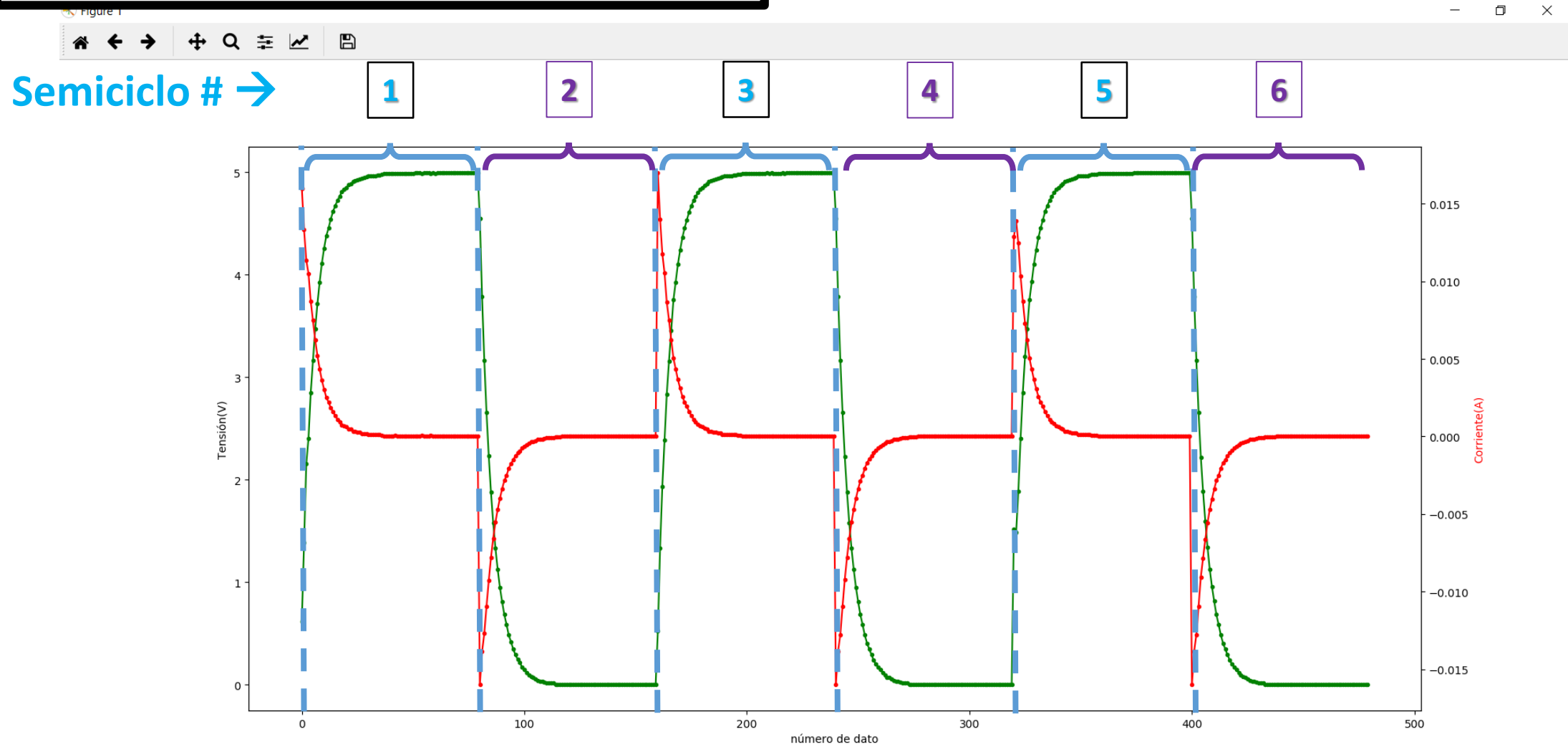
1) `RCTransitorio.py` → ajusta la $V_c(t)$ y la $i(t)$ con las expresiones que figuran en “enfoque teórico”. Se extrae el valor del tiempo tau para cada R elegida

```
23 #Elijo el directorio donde se encuentran los archivos que voy a analiza
24 os.chdir(r'C:\Users\Charly\NubeDF\Arduino\Data\Transitorios')
25
26 print("nombre del archivo completo con terminación .txt incluida")
27 file = input()
28 print("Valor de la R (ohm)")
29 Resist = float(input())
30
31 #Importo los datos del archivo
32 data = np.loadtxt(file,dtype=float,delimiter = ',',skiprows= 1)
33
34
35
36 x1i = data[:,0] #columna de tiempo en segundos
37 y1i = data[:,2] #caída de tensión sobre el capacitor
38 v2i= (data[:,1]-data[:,2])/Resist #corriente del circuito
39
40
41
42
43 #vdiff=
44 #const
45 numdat
```

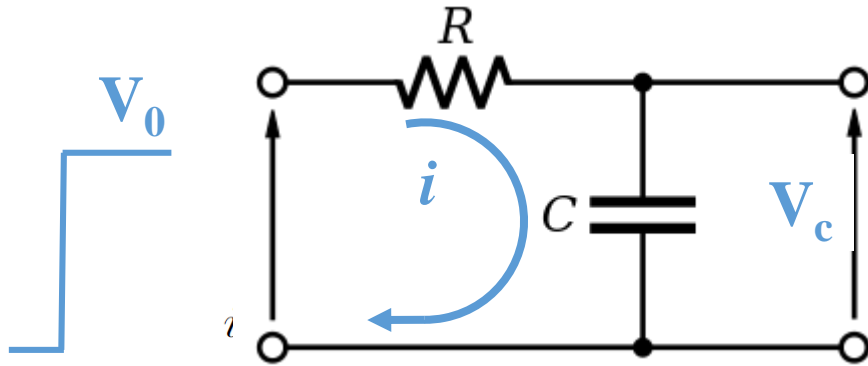
```
57 ax2.set_ylabel('corriente(A)', color='r')
58
59 #%%
60 print("Seleccionar el semi-ciclo de carga o descarga")
61 print("Por ej: 1-3-5 para carga o 2-4-6 para descarga")
62 print("(si es que tienen 3 ciclos completos medidos)")
63 ciclo = int(input())
64
65 #Limitar los datos al rango de interés
66
67 datoinicial1=(ciclo-1)*80
68 datofinal1=datoinicial1+79
69
70
71 x1=x1i[datoinicial1:datofinal1]
72 y1=y1i[datoinicial1:datofinal1]
73 y2=y2i[datoinicial1:datofinal1]
74 x1corr=x1i[datoinicial1+5:datofinal1]
75 y2corr=y2i[datoinicial1+5:datofinal1]
76
77 #Graficamos los datos en la zona elegida en función
78 fig=plt.figure()
79
80
81 y1x.set_xlabel('tiempo(s)',
82 y2x=y1x.twinx()
```

Se graban los valores de tau vs R en el archivo: “`SalidaRCTransitorio.txt`”

Cómo elegir los “semi-ciclos”



Análisis del “semi-ciclo” elegido

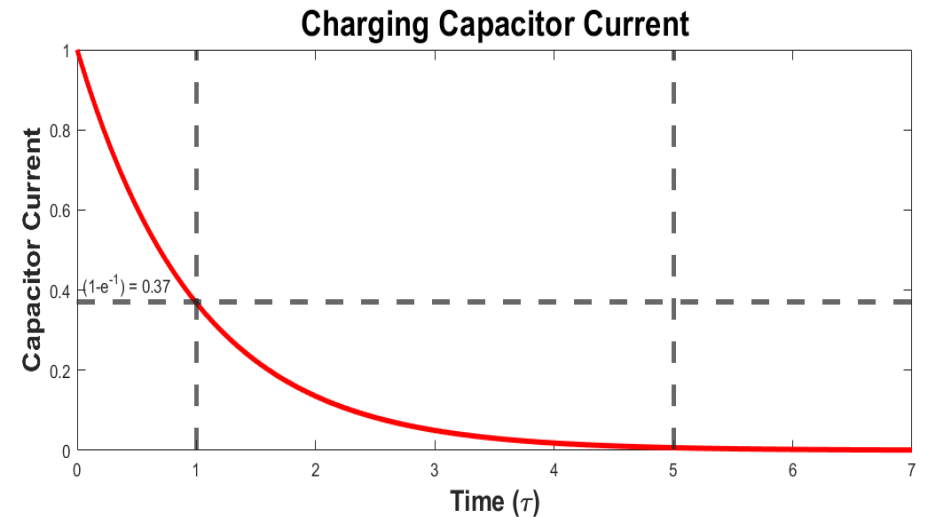
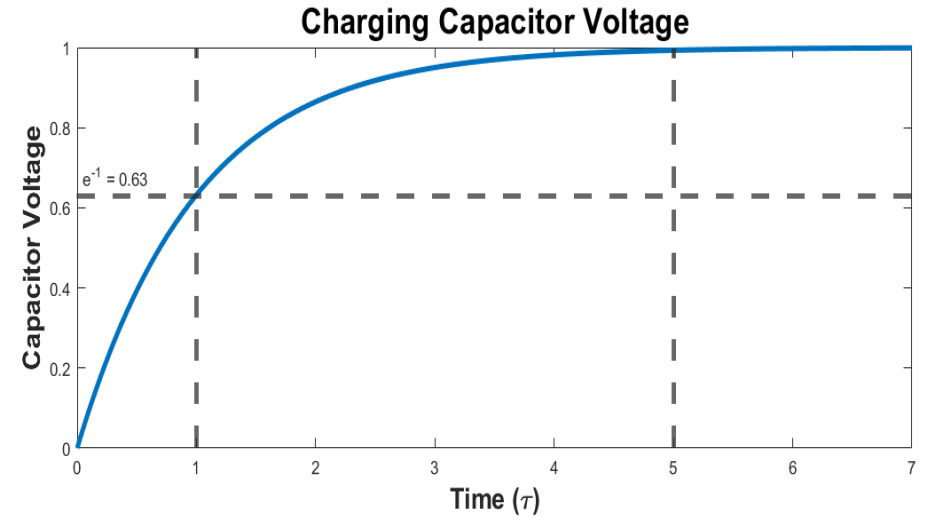


$$i(t) = \frac{V_0 - V_C(0)}{R} e^{-t/RC}$$

$$V_C(t) = V_C(0) + [V_0 - V_C(0)] (1 - e^{-t/RC})$$

Carga: $V_C(0)=0$ $V_C = V_0(1 - e^{-t/\tau})$, $i(t) = \frac{V_0}{R}(1 - e^{-t/\tau})$

Descarga: $V_0=0$
 $V_C(0)=V_0$ $V_C = V_0 e^{-t/\tau}$, $i(t) = \frac{-V_0}{R} e^{-t/\tau}$



Scripts de Python para graficar y analizar los datos

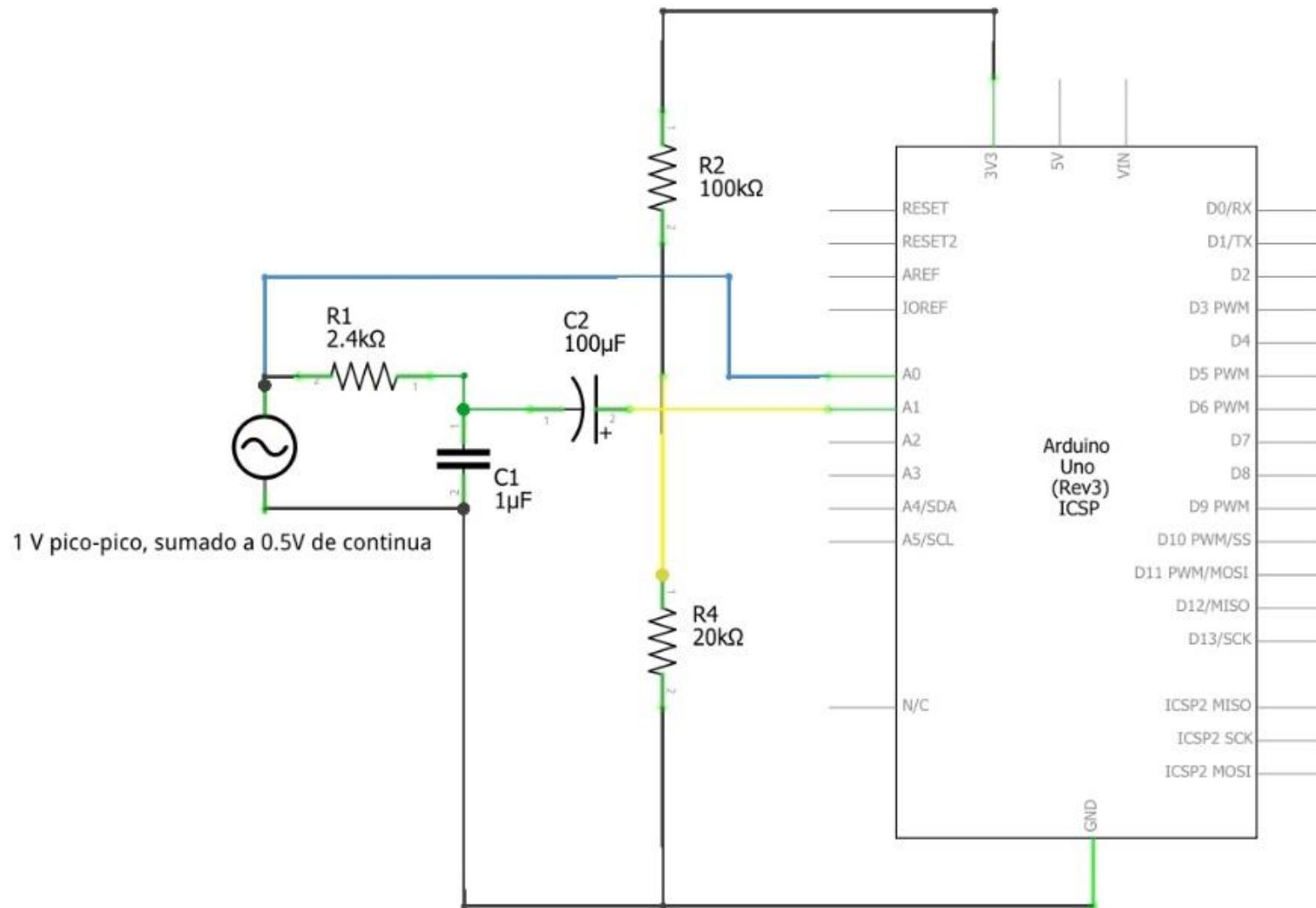
2) Ajuste RC.py: ajusta los datos obtenidos de tau vs R → Extrae el valor de C

```
Spyder (Python 3.7)
Archivo Editar Buscar Código fuente
C:\Users\Charly\NubeDF\PythonProgs\Cursos\La
Derivadas y rho de T.py dataSaver
11 import statsmodel
12 import os
13 from IPython import get_ipython
14
15 """
16 #selecciono si las figuras aparecen en la terminal (inline) o en ventana emergente (qt5)
17 #get_ipython().run_line_magic('matplotlib', 'inline')
18 get_ipython().run_line_magic('matplotlib', 'qt5')
19
20 #Elijo el directorio donde se encuentran los archivos que voy a analizar
21 os.chdir(r'C:\Users\Charly\NubeDF\Arduino\Data\Transitorios')
22
23 #Leo el archivo txt tiene que estar en la misma carpeta que el programa
24
25 file = 'SalidaRCtransitorio.txt'
26 #en el archivo que leo las columnas están separadas por coma, la primera fila tiene el título ent
27 Misdatos1=[]
28 Misdatos = np.loadtxt(file1, delimiter=",", skiprows=1)
29 #print(Misdatos1)
30 #Ordeno los datos con la tensión de entrada
31 Misdatos1=Misdatos[np.argsort(Misdatos[:, 2])]
32
33 """
34 #leer los datos desde un archivo csv delimitado por tabs
35 print("nombre del archivo completo con terminación .csv incluida")
36 file = input()
37 Misdatos1=np.genfromtxt(file, delimiter='\t', skip_header=1)
38 """
39
40 #Datos ordenados col0 : Vfuerza, col1: Vresistencia, Col2. Corriente en mA
41 y0= Misdatos1[:,0] #valores de tau
42 errorry0 = Misdatos1[:,1] #incertidumbre de tau
43 x0= Misdatos1[:,2] #valores de r
44 errorrx0=Misdatos1[:,3]/10 #valores de incertidumbre de r
45
46 plt.ion()
47
```

```
C:\Users\Charly\NubeDF\PythonProgs\Cursos\Lab3\Ajuste RC.py
Derivadas y rho de T.py dataSaver.py dataPlotter.py Ploteando_varias_figs.py FeTeSe_filtrados.py RCtransitorio_Ch.py Ajuste RC.py
54 plt.xlabel('Resistencia (ohm)');
55 plt.ylabel('Tau (s)');
56 plt.show()
57
58
59 print('Chequeamos que el comportamiento de Tau vs R sea lineal antes de
60
61
62 """
63 #Ajuste por cuadrados mínimos ponderado con incertidumbres en y
64 w=1/errorry0
65 X = sm.add_constant(x0)
66 wls_model = sm.WLS(y0,X, weights=w)
67 results = wls_model.fit()
68 o,C=results.params
69 #intervalo de confianza para ordenada al origen y pendiente
70 oint,Cint=results.conf_int(alpha=0.05)
71 deltaC=abs((Cint[1]-Cint[0])/2)
72 deltao= abs((oint[1]-oint[0])/2)
73
74 print("C=(", C," +/- ",deltaC," ) F")
75 print("ordenada al origen=(", o," +/- ",deltao," ) V")
76
77 #calculo las bandas de confianza y predicción
78 from statsmodels.stats.outliers_influence import summary_table
79 from statsmodels.sandbox.regression.predstd import wls_prediction_std
80
81 st, data, ss2 = summary_table(results, alpha=0.05)
82
```

FILTRO PASA-BAJOS

$$\omega_0 = \frac{1}{RC} \approx 416 \text{ rad/s} \rightarrow F_o^{(PB)} \approx 66 \text{ Hz} // F_o^{(PA)} \approx 0.03 \text{ Hz}$$



Software: medición / análisis

Doscanales_frec_ampli_fase.ino / SerialPlot

```

Doscanales_frec_ampli_fase Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda

Doscanales_frec_ampli_fase
int espera=0; // Cambiar para ajustar la ventana de datos a la frecuencia
/* CAMBIAR "espera" según la frecuencia de la señal a medir: espera=0--> 1
 * 0 para frec=451-1000 Hz // 100 161-450 Hz // 300 74-160 Hz // 1000 26-7
 * Para frec > 1000 Hz fijar espera=0 y cambiar el prescaler de 04 a 03 como
 */

int canal1[130];
int canal2[130];
long tiempo[130]; // para que no resetee cada 32 ms (aprox)!
float V1;
float V2;

float maxV, minV, maxV11, minV11, maxV12, minV12, maxV21, minV21, maxV22,
int maxIndice, minIndice, maxIndice11, maxIndice12, minIndice11, minIndice
int maxIndice21, maxIndice22, minIndice21, minIndice22, Flagmax2, Flagmin2
float Amplitud1, Amplitud2, defasaje, frec1, frec2;
int Flag;
int ventana=4; // fija la ventana local (# de puntos) para determinar los

Doscanales_frec_ampli_fase Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda

Doscanales_frec_ampli_fase
* LOS Labels indican el orden en el que se envían los
* datos medidos y estimados (separados x ",")
* Sacamos los "labels" para poder usar el SerialPlotter.py
Serial.print("tiempo (s)");
Serial.print(',');
Serial.print("V1 (V)");
Serial.print(',');
Serial.print("V2 (V)");
Serial.print(',');
Serial.print("Amplitud1 (V)");
Serial.print(',');
Serial.print("Amplitud2 (V)");
Serial.print(',');
Serial.print("frec1 (Hz)");
Serial.print(',');
Serial.print("Transferencia");
Serial.print(',');
Serial.println("defasaje (°)");

```

Monitorear la **frecuencia** estimada (SerialPlot)
Modificar 'espera' si la registra mal y descartar la medición donde eso pasó...

Software: medición / análisis

**Python: 1) Analisis 2 señales con amplitud V2 –
2) Ajuste filtros PB.py**

Puntos de Control

1. Obtener señales V_{in} y V_{out} para cada filtro a distintas frecuencias (en el rango relevante!).
2. Obtener los diagramas de Bode correspondientes.
3. Ajustar mediante Python \rightarrow obtener la frecuencia de corte / discutir valores y diferencias obtenidos

