Microcontroladores II

Juan Esteban Giraldo Hoyos
Ingeniero Electrónico
Magíster en Gestión de Ciencia, Tecnología e Innovación

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Anteriormente trabajamos

com x

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Hoy trabajaremos

com x

Escuela de Ingenierías

IUSH Institución Universitaria SALAZAR Y HERRERA

SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

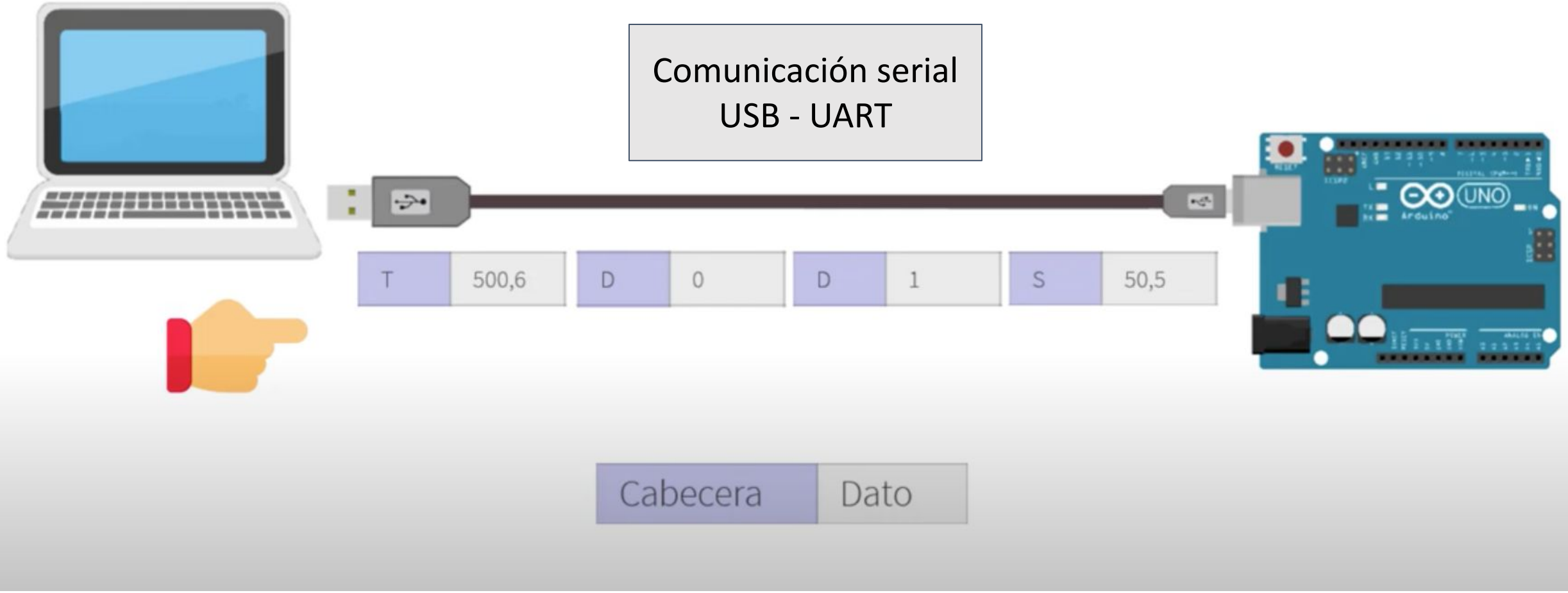
Requerimientos:

1) Instalar las siguientes librerías de python y validar que se tengan ya instaladas las 2 primeras de la lista: `pip install matplotlib`

- pyserial (Manejar la comunicación serial)
- tkinter (Manejar la interfaz gráfica)
- matplotlib (Graficar plano x y)

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Comunicación con Arduino



Microcontroladores II - Interfaz gráfica usando python

Pasos:

1) Abrir Visual Code y empezar a editar un nuevo archivo (seleccionando lenguaje Python)
2) Agregar las librerías de python

```
import tkinter as tk
import tkinter.ttk as ttk
import serial.tools.list_ports
import matplotlib
matplotlib.use("TKAgg")
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from random import randrange
import time
```

Microcontroladores II - Interfaz gráfica usando python

Pasos:

3) Definir variables globales

```
#Global variables
wide_graph = 100 #define el ancho de la gráfica con la cantidad de datos
global isOpen
isOpen = False
dataList = []
```

4) Crear las siguientes funciones para establecer y retirar la comunicación serial

```
#Functions
def serial_ports():
    if not serial.tools.list_ports.comports():
        return "NO PORTS"
    else:
        global ports
        ports = serial.tools.list_ports.comports()
        return ports

def connect():
    global serial_com
    if(cb.get()=="NO PUERTOS"):
        print("No se puede conectar")
    else:
        serial_com = serial.Serial(str(ports[cb.current()][0]),9600)
        global isOpen
        isOpen = True
        port_connect_button["state"] = "disabled"
        port_disconnect_button["state"] = "normal"

def disconnect():
    global serial_com
    serial_com.close()
    global isOpen
    isOpen = False
    port_connect_button["state"] = "normal"
    port_disconnect_button["state"] = "disabled"
```

Microcontroladores II - Interfaz gráfica usando python

Pasos:

5) Crear las siguientes funciones para el control de los leds.

```
def led_off():
    global serial
    global isOpen
    if isOpen:
        serial_com.write(b"D0\n")

def led_on():
    global serial_com
    global isOpen
    if isOpen:
        serial_com.write(b"D1\n")

def slider_changed(event):
    data_slider.configure(text=get_current_value()+"%")
    global isOpen
    if isOpen:
        data = "S"+get_current_value()+"\n"
        serial_com.write(bytes(data,'utf-8'))
        print("Slider: "+data)

def get_current_value():
    return '{:.1f}'.format(slider.get())
```

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

Pasos:

6) Agregar la gráfica XY y configurar sus ejes.

```
#Graph setup
fig = plt.figure(figsize=(6,3))
fig.set_facecolor("#17161b")
ax = plt.axes()
ax.set_facecolor("#17161b")
ax.title.set_color('white')
ax.xaxis.label.set_color('white')
ax.yaxis.label.set_color('white')
ax.grid(alpha=0.1)
ax.tick_params(axis='x', colors='white')
ax.tick_params(axis='y', colors='white')

#Buffer for data input
x = [None]*wide_graph
y = [0]*wide_graph
for i in range(0,wide_graph,1):
    x[i] = i
ln, = plt.plot(x, y, '-', color="orange")

plt.axis([0, wide_graph, 0, 1100])
plt.ylabel('Valor lectura')
plt.title('Serial COM Data')
```

IUSH Institución Universitaria SALAZAR Y HERRERA

ABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

Pasos:

7) Crear la función que recibe periodicamente los datos enviados desde el arduino con la información del potenciómetro y actualiza la vista de la gráfica.

```python
def update(frame):
    if isOpen:
        global serial_com
        global y
        global x
        data = serial_com.readline().decode('ascii')
        print("Recibido: ", data)
        data = data.rstrip()
        data = data.replace("T","")
        y.append(float(data))

        #pop para evitar desbordamiento del buffer
        if len(y) > wide_graph:
            y.pop(0)

        y = y[-wide_graph:]
        x = x[-wide_graph:]

        ln.set_data(x, y)
        return ln

def on_hover(event):
    event.widget.configure(bg="#020A90")

def on_default(event):
    event.widget.configure(bg="#00044A")
```

IUSH
Institución Universitaria
SALAZAR Y HERRERA
ABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

Pasos:

8) Crear la ventana de la interfaz de usuario

```
#main
root = tk.Tk()
root.title("Arduino Serial Data")
root.geometry("700x500")
root.configure(bg="#17161b")
root.resizable(0,0)

frame = tk.Frame(root)
frame.pack()
frame.configure(bg="#17161b")

canvas = FigureCanvasTkAgg(fig, master = root)
canvas.draw()

#Putting canvas in the Tkinder window
canvas.get_tk_widget().pack()

#Matplotlib Toolbar
toolbar = NavigationToolbar2Tk(canvas,root)
toolbar.update()

#Toolbar in tkinder window
canvas.get_tk_widget().pack()
```

IUSH
Institución Universitaria
SALAZAR Y HERRERA
ABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

Pasos:

9) Agregar a la vista el selector y los botones de la comunicación serial

```
#Serial port
serial_info_frame = tk.LabelFrame(frame, text="Conexion Serial",bg="#17161b",fg="#fff")
serial_info_frame.grid(row=0, column=0)

cb = ttk.Combobox(serial_info_frame, value=serial_ports())
cb.current(0)
cb.grid(row=0, column=1)

port_connect_button= tk.Button(serial_info_frame, text="CONECTAR", bd=1, fg="#fff", bg="#00044A", activebackground="#020A90", relief="flat", command=connect)
port_connect_button.grid(row=0,column=2)
port_connect_button.bind('<Enter>', on_hover)
port_connect_button.bind('<Leave>', on_default)

port_disconnect_button= tk.Button(serial_info_frame, text="DESCONECTAR", bd=1, fg="#fff", bg="#00044A", activebackground="#020A90", relief="flat", command=disconnect)
port_disconnect_button.grid(row=0,column=3)
port_disconnect_button["state"] = "disabled"
port_disconnect_button.bind('<Enter>', on_hover)
port_disconnect_button.bind('<Leave>', on_default)

for widget in serial_info_frame.winfo_children():
    widget.grid_configure(padx=10, pady=5)
```

Microcontroladores II - Interfaz gráfica usando python

Pasos:

10) Agregar a la vista los botones de control del led 1 y el slider para controlar el led 2

```
#Send data
send_info_frame = tk.LabelFrame(frame, text="Envio de Datos", bg="#17161b", fg="#fff", highlightcolor="#00044A")
send_info_frame.grid(row=1, column=0, padx=10, pady=10)

#ON button
on1_button = tk.Button(send_info_frame, text="ON", bd=1, fg="#00044A", activebackground = "#020A90", relief="flat", command=led_on)
on1_button.grid(row=0, column=0)
on1_button.bind('<Enter>', on_hover)
on1_button.bind('<Leave>', on_default)

#OFF button
off1_button = tk.Button(send_info_frame, text="OFF", bd=1, fg="#00044A", activebackground = "#020A90", relief="flat", command=led_off)
off1_button.grid(row=0, column=1)
off1_button.bind('<Enter>', on_hover)
off1_button.bind('<Leave>', on_default)

#Slider
current_value = tk.DoubleVar()
style = ttk.Style()
style.configure("TScale", background="#17161b")
slider = ttk.Scale(
    send_info_frame,
    from_=0,
    to=100,
    orient='horizontal',
    variable=current_value,
    command=slider_changed
)
slider.grid(row=0,column=2)
data_slider = tk.Label(send_info_frame, text="0%", bg="#17161b", fg="#fff")
data_slider.grid(row=0,column=3)

for widget in send_info_frame.winfo_children():
    widget.grid_configure(padx=10, pady=5)
```

IUSH
Institución Universitaria
SALAZAR Y HERRERA
ABER Y EL HACER

Microcontroladores II - Interfaz gráfica usando python

Pasos:

11) Agregar una animación que llama cada 500ms la función update() para actualizar la gráfica y el inicio del loop

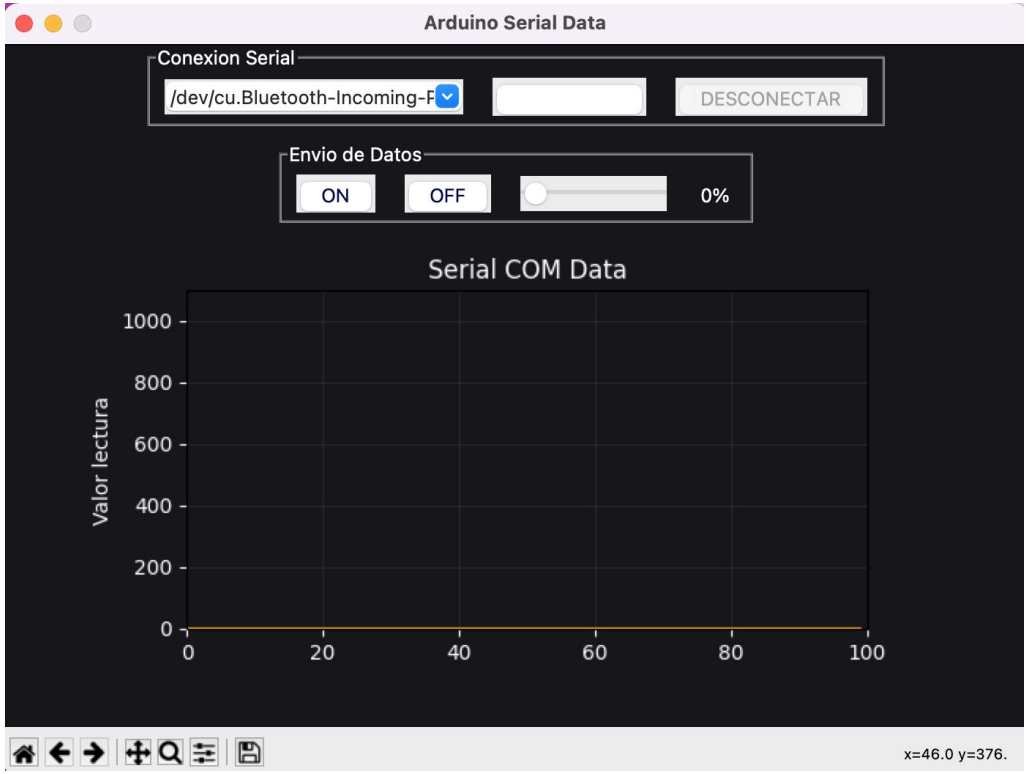
```
animation = FuncAnimation(fig,update,interval=500, cache_frame_data=False)
root.mainloop()
```

Microcontroladores II - Interfaz gráfica usando python

Pasos:

12) Guardamos el programa con un nombre como: “comunicacion_bidireccional.py”

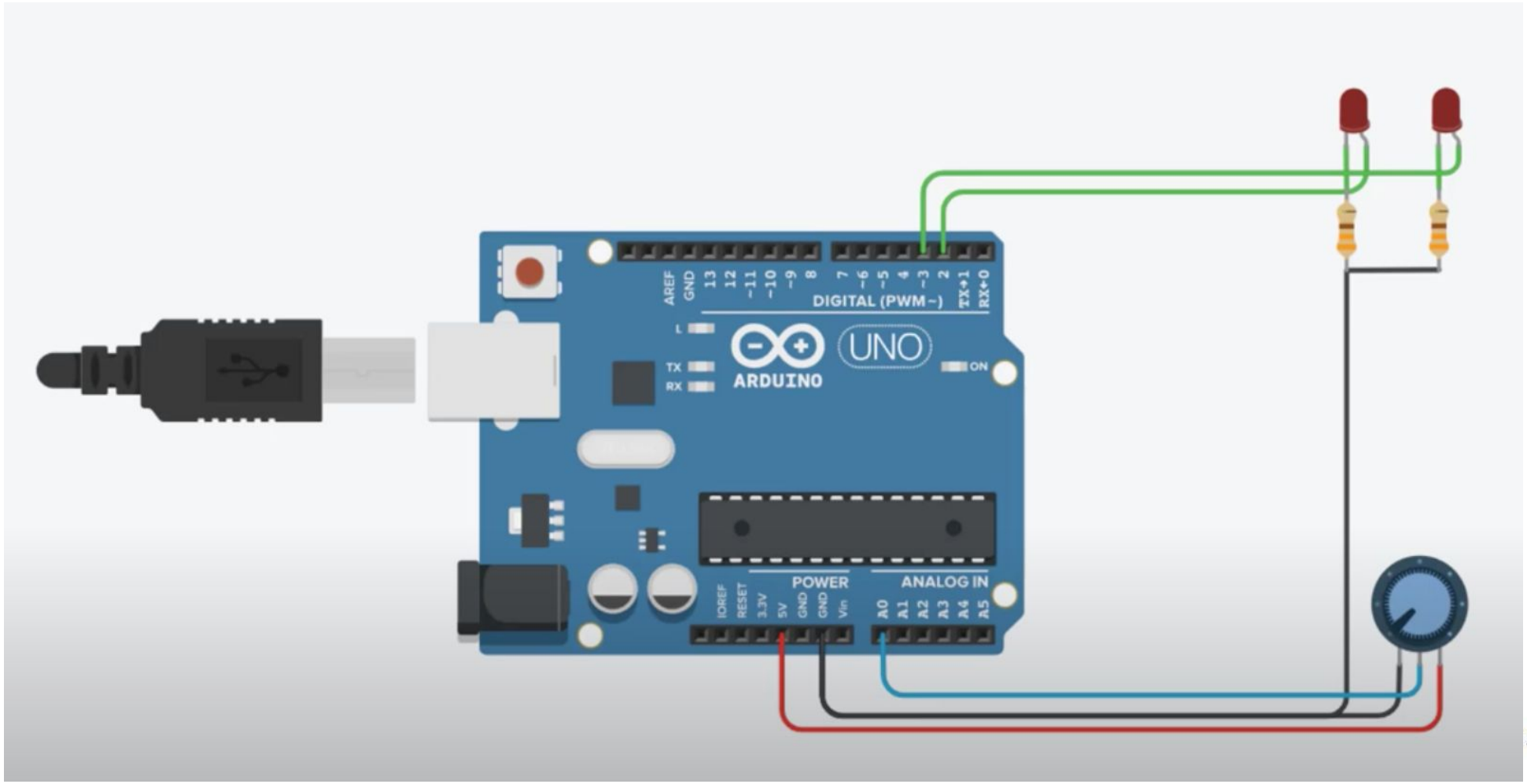
13) Vamos a la ubicación del archivo y ejecutamos python comunicacion_bidireccional.py para depurar el programa y ver la interfaz creada



Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Firmware Arduino

MONTAJE: 2 Leds, 1 potenciómetro, 1 cable de comunicación serial



IUSH
Institución Universitaria
SALAZAR Y HERRERA
ABER Y EL HACER

Microcontroladores II - Firmware Arduino

comunicacionBidireccional

```
#include <TimerOne.h>

//Definicion de pines
#define analogPin A0
#define led1 2
#define led2 3

//Definicion de variables globales
int val = 0;
char dato;
String serialData;

void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  Serial.begin(9600); //Inicializando comunicacion serial
  digitalWrite(led1,LOW); //Inicializando en bajo led 1
  analogWrite(led2,0); //Inicializando en bajo led 2

  // Configuramos Timer1 para generar una interrupción cada 1 segundos
  Timer1.initialize(1000000); // Intervalo de 1 segundos en microsegundos
  Timer1.attachInterrupt(analogData); // Asociamos la función de interrupción al temporizador
}

void loop() {
}
```

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER

Microcontroladores II - Firmware Arduino

```
void analogData(){
  val = analogRead(analogPin);
  Serial.print("T");
  Serial.println(val*1.0);
}

void serialEvent(){
  dato = (char)Serial.read();
  switch(dato){

    case 'D':
      serialData = Serial.readStringUntil('\n');
      digitalWrite(led1,serialData.toInt());
      serialData = "";
    break;

    case 'S':
      serialData = Serial.readStringUntil('\n');
      analogWrite(led2,serialData.toInt());
      serialData = "";
    break;

  }
}
```

PRÁCTICA #3 10%

Implementar la interfaz de usuario propuesta en esta clase y agregar un botón a la interfaz que permite actualizar el listado de los puertos de comunicación (COMs) disponibles en el selector (Ver esa parte desde programa creado en la clase pasada) para controlar el encendido y apagado de un led, dimerizar un segundo led y gráficar el estado de la señal entregado por un potenciometro.

Modificar la interfaz creada de usuario para cambiar los colores y personalizar aspectos como el tamaño de la ventana, el nombre del programa, etc

Bibliografía

- https://pythonbasics.org/tkinter-button/
- https://www.youtube.com/watch?v=egldIB_--Ho

Escuela de Ingenierías
IUSH Institución Universitaria SALAZAR Y HERRERA
SER PERSONA DESDE EL SABER Y EL HACER