### Cosas a tener en cuenta:

* Hay que tener instalado python y las librerías serial y time
* Al puerto que conectemos el Arduino, hay que especificarlo en los scripts
* Cerrar el Serial Monitor de Arduino antes de correr el python script.
* Si estamos manejando la bbdd, hay que configurar bien las tablas y credenciales.

### **Leer** datos **de Arduino Serial Port con Python script:**

Codigo Arduino:

void setup() {

Serial.begin(9600);

pinMode(LED\_BUILTIN, OUTPUT);

}

void loop() {

Serial.println("1");

delay(500);

Serial.println("2");

delay(1500);

}

Codigo Python:

import serial

import time

# Replace '/dev/ttyUSB0' with your actual port (for Linux/Mac) or 'COM3', 'COM4' (for Windows)

arduino\_port = 'COM6' # Example for Windows, change to match your port

baud\_rate = 9600 # Must match the baud rate in your Arduino code

# Initialize the serial connection

ser = serial.Serial(arduino\_port, baud\_rate, timeout=1)

# Allow time for the connection to initialize

time.sleep(2)

print("Connected to Arduino port:", arduino\_port)

try:

while True:

if ser.in\_waiting > 0: # Check if there is data waiting in the buffer

line = ser.readline().decode('utf-8').rstrip() # Read the line and decode to utf-8

print(f"Received: {line}")

time.sleep(0.1) # Small delay between reads

except KeyboardInterrupt:

print("Program stopped by user")

finally:

ser.close() # Ensure the serial port is closed when the program ends

#### ¿Cómo saber el puerto que usa el arduino en la raspi?

1. SIN EL ARDUINO CONECTADO, correr en la terminal:

**ls /dev/tty\***

Este comando te mostrará una lista de todos los puertos seriales disponibles. Los más comunes para dispositivos USB son:

* 1. /dev/ttyUSB0
  2. /dev/ttyUSB1
  3. /dev/ttyAMA0 (si se usa el puerto serie GPIO)
  4. /dev/ttyACM0 (puede usarse para algunos dispositivos Arduino)

1. Ahora conectar el arduino y volver a correr el comando **ls /dev/tty\*,** ahora debería aparecer un nuevo dispositivo conectado
2. Verificar dispositivo conectado:

**dmesg | grep tty**

Este comando te mostrará los mensajes del kernel relacionados con la conexión de dispositivos a los puertos seriales. Busca la línea que muestra algo como:



Esto indicará que un dispositivo está conectado al puerto **/dev/ttyUSB0**

### Leer datos desde el Arduino Serial Port con Python script que escriba:

Codigo Arduino:

int ledPin = 13; // Pin donde está conectado el LED

void setup() {

// Iniciar el puerto serie

Serial.begin(9600); // Velocidad de baudios debe coincidir con la configuración en Python

// Configurar el pin del LED como salida

pinMode(ledPin, OUTPUT);

// Apagar el LED inicialmente

digitalWrite(ledPin, LOW);

}

void loop() {

// Verificar si hay datos disponibles en el puerto serie

if (Serial.available() > 0) {

// Leer el valor enviado por el puerto serie

char received = Serial.read();

// Verificar si el valor es '1'

if (received == '1') {

// Encender el LED

digitalWrite(ledPin, HIGH);

Serial.println("LED ON"); // Enviar confirmación por el puerto serie

} else {

// Apagar el LED si se recibe cualquier otro valor

digitalWrite(ledPin, LOW);

Serial.println("LED OFF"); // Enviar confirmación por el puerto serie

}

}

}

Codigo Python:

import serial

import time

# Cambia 'COM3' (en Windows) o '/dev/ttyUSB0' (en Raspberry Pi/Linux) por el puerto correcto

serial\_port = '/dev/ttyUSB0' # O el puerto adecuado donde está conectado tu dispositivo

baud\_rate = 9600 # La velocidad de baudios debe coincidir con la configuración del dispositivo

# Inicializar el puerto serie

ser = serial.Serial(serial\_port, baud\_rate, timeout=1)

# Esperar un poco para asegurarse de que la conexión esté establecida

time.sleep(2)

print(f"Connected to {serial\_port} at {baud\_rate} baud rate")

try:

while True:

# Pedir al usuario que ingrese un mensaje

message = input("Enter a message to send to the serial device: ")

# Enviar el mensaje al dispositivo conectado

ser.write(message.encode('utf-8')) # Convertir el mensaje en bytes antes de enviarlo

ser.write(b'\n') # Enviar un salto de línea (opcional, dependiendo de tu dispositivo)

# Leer la respuesta (si es que el dispositivo envía algo de vuelta)

if ser.in\_waiting > 0:

response = ser.readline().decode('utf-8').rstrip()

print("Received from device:", response)

except KeyboardInterrupt:

print("Program interrupted by the user")

finally:

# Asegurarse de cerrar la conexión al finalizar

ser.close()

print("Serial port closed")

### Leer datos de bbdd y mandarlos x puerto serie

1. InfinityFree protege los datos de Mysql y no permite que accedamos desde python entonces hacemos una magia media rara:
2. Codigo de python:

import requests

import serial

import time

import platform

def fetch\_data\_from\_website():

url = 'https://ciie-lab.42web.io/get\_data.php'

try:

response = requests.get(url)

response.raise\_for\_status() # Levanta excepción si hay un error HTTP

data = response.text.strip().split('\n')

return data

except requests.exceptions.HTTPError as http\_err:

print(f"HTTP error occurred: {http\_err}")

except Exception as err:

print(f"Other error occurred: {err}")

return None

def send\_data\_to\_arduino(data, port):

try:

ser = serial.Serial(port, 9600, timeout=1)

time.sleep(2) # Esperar a que el Arduino se inicialice

for item in data:

if item: # Asegurarse de que el item no esté vacío

ser.write((item + '\n').encode('utf-8'))

print(f"Enviado: {item}")

time.sleep(1) # Esperar 1 segundo entre envíos

ser.close()

except serial.SerialException as e:

print(f"Error con el puerto serie: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

# Ajustar el puerto dependiendo del sistema operativo

if platform.system() == 'Windows':

port = 'COM6'

else:

port = '/dev/ttyUSB0' # Cambia esto según tu configuración en Linux/macOS

# Obtener los datos del sitio web

data = fetch\_data\_from\_website()

# Si hay datos, enviarlos al Arduino

if data:

send\_data\_to\_arduino(data, port)

print("Datos enviados al Arduino exitosamente.")

else:

print("Error: No se pudo obtener datos del sitio web.")