

Data Science

Integração



+ Python



Minicurso:
Integração Arduino – Python:
Um estudo de caso na
Engenharia de Materiais

Eng. Afrânio Costa, MSc
(6Sigma Excellence)

Data: 21/04/21 (quarta-feira)
Horário: 08h00 às 12h00

Laboratório de Soldagem e Inspeção
UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Afrânio Costa

Ago/2021

6Sigma Excellence



<https://www.linkedin.com/in/afraniofilho/>



6Sigma Excellence

<https://www.youtube.com/channel/UCPh3wY2LGydXYLAVxELU-8A>



<https://github.com/afraniofilho>



acostaf3@ford.com

afraniofilho@hotmail.com

Agenda

- Introdução ao ARDUINO
- Comunicação Serial pelo Arduino
- Introdução ao Python
- Biblioteca Python pyserial
- Integração Arduino-Python (Gráficos usando a Matplotlib e drawnow)
- Aplicação Medição da Oxigenação (vídeo)
- Q&A

Arduino



Arduino Nano 33 IoT



Raspberry Pi Pico



SMARTSAM D21

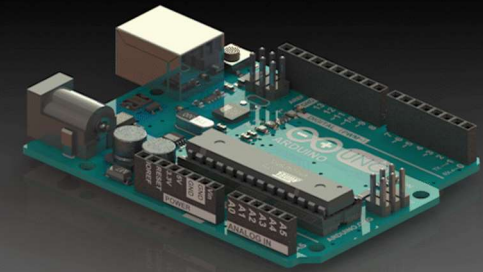
<https://store.arduino.cc/usa/nano-33-iot>



RP2040

<https://www.raspberrypi.org/documentation/rp2040/getting-started/>

Arduino UNO



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



ATmega328P

<https://www.microchip.com/en-us/product/ATmega328P>

Um microcontrolador é praticamente um computador em um chip.

O microcontrolador contém todos os itens necessários tais como processador, memória ROM, memória RAM, periféricos de entrada / saída, Conversor Analógico/Digital, etc.

Devices

OLED LCD



Bluetooth



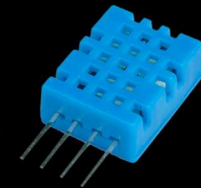
Ultrasonic
HC-SR04



Joystick



Micro Servo 9g SG90



Temperature and
Humidity DHT11



LCD 16x2 5V



Accelerometer and
Gyroscope MPU-6050



WiFi
ESP8266



Sound

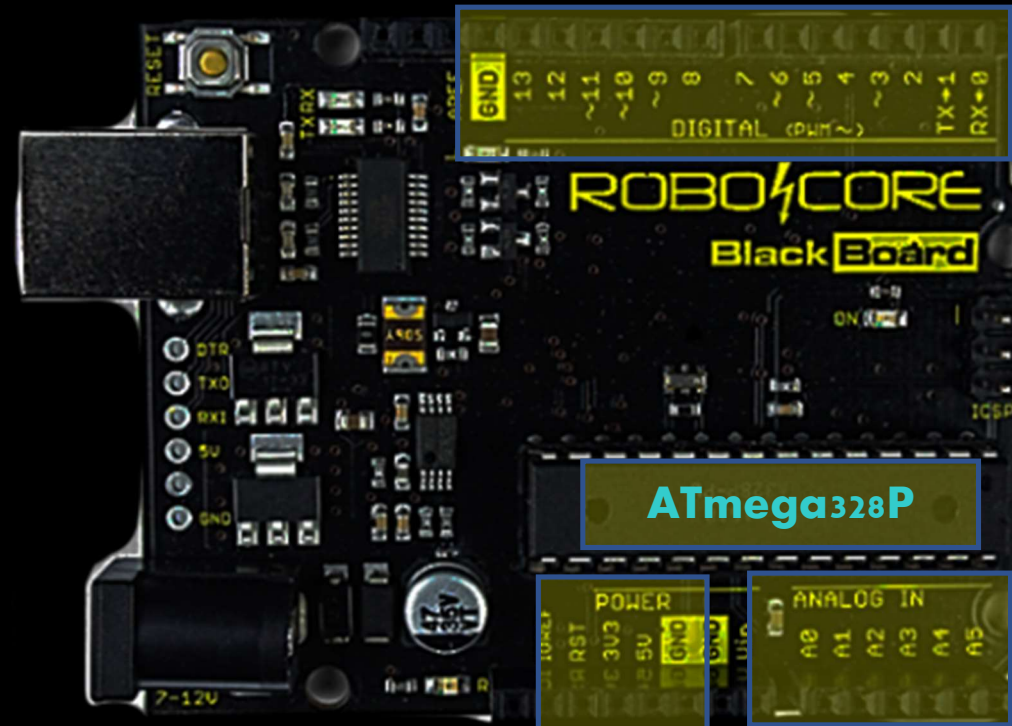
Arduino

ARDUINO UNO



PORTAS DIGITAIS

PORTA USB



ATmega328P

ALIMENTAÇÃO
EXTERNA

PORTAS ENERGIA

PORTAS ANALÓGICAS

Arduino

AMBIENTE DE PROGRAMAÇÃO

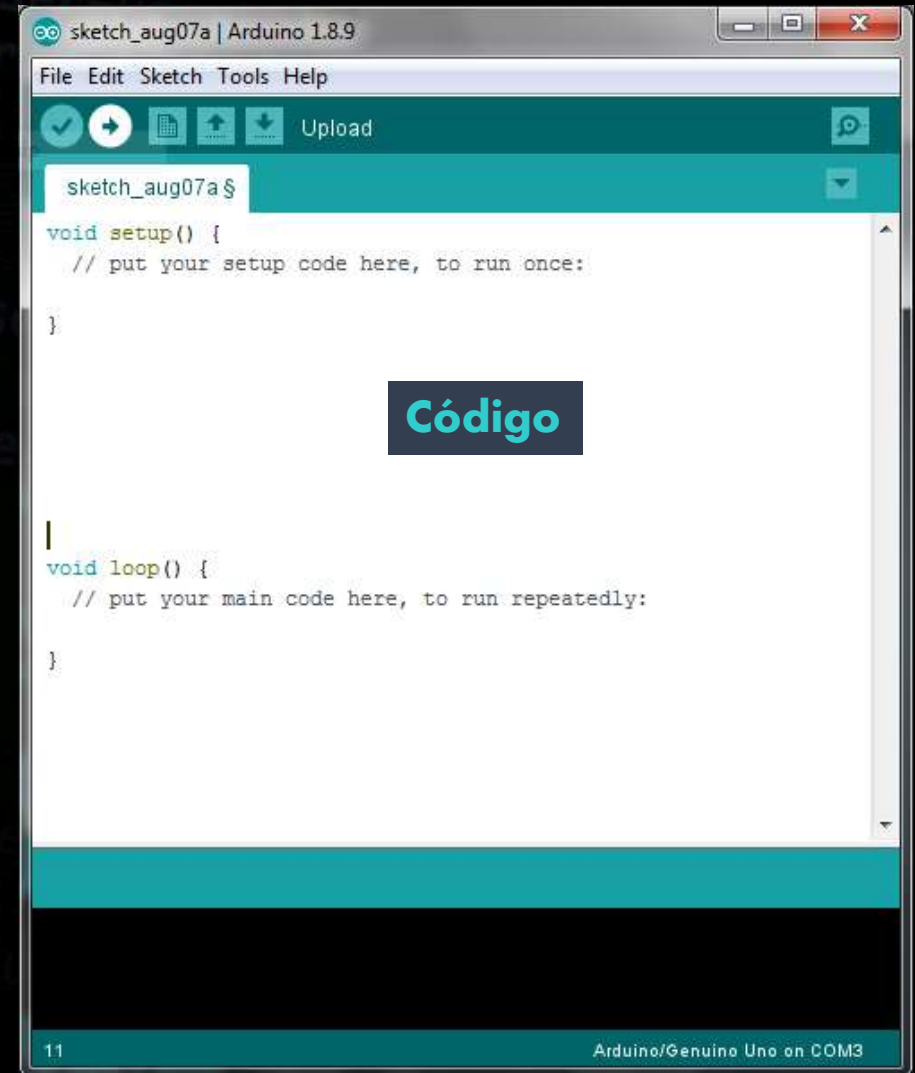
IDE - Integrated Development Environment

LINGUAGEM C++ (com pequenas modificações)

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

ORIENTADA A OBJETO (C++, Java, Python)

O LED é um objeto
A COR é uma propriedade
do objeto LED que posso
mudar via código



Arduino

AMBIENTE DE PROGRAMAÇÃO

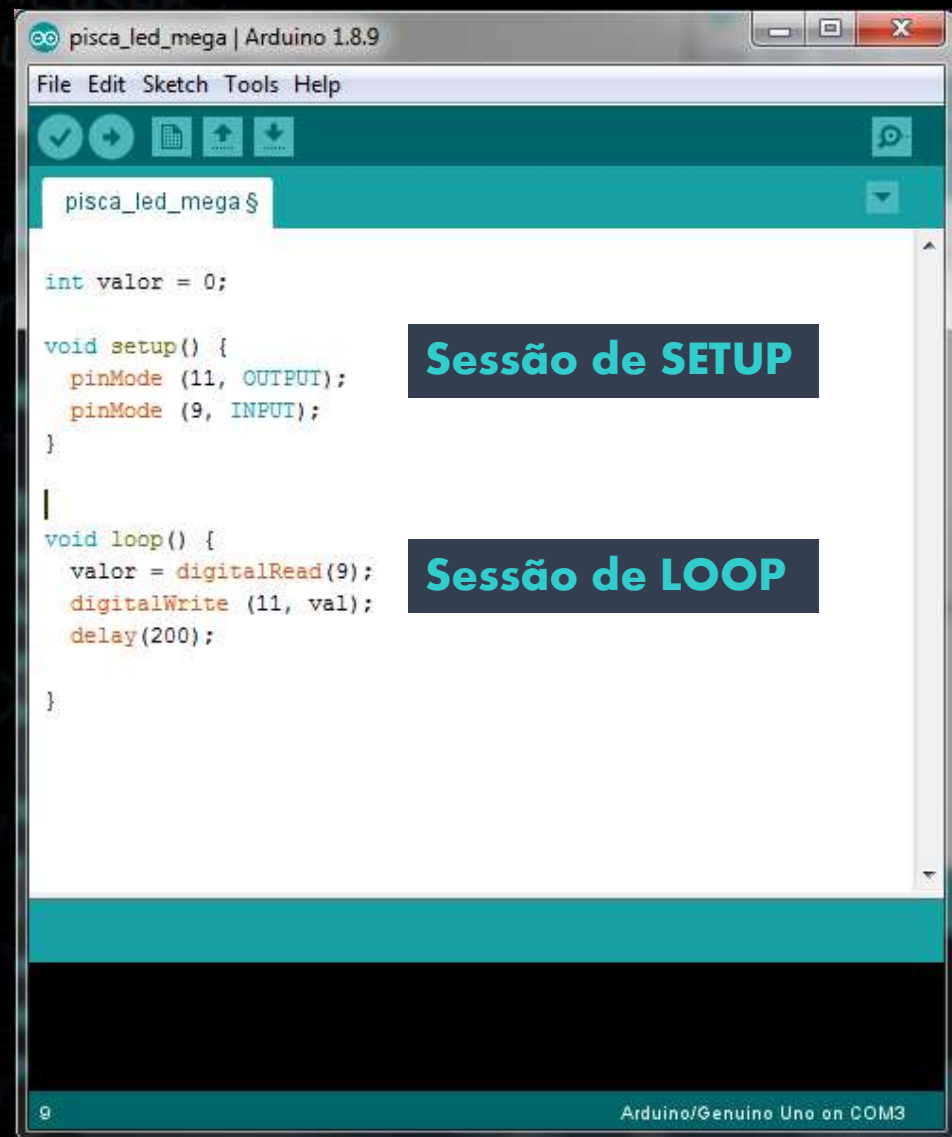
SETUP:

Será executado uma vez ao iniciar o código.

Configurações iniciais dos pinos, variáveis, portas, etc.

LOOP

Código do programa que será executado em loop (até que um comando de parada ocorrer)



The screenshot shows the Arduino IDE interface with a sketch named 'pisca_led_mega'. The code is written in C++ and includes a `setup()` function and a `loop()` function. The `setup()` function configures pin 11 as an output and pin 9 as an input. The `loop()` function reads the value from pin 9 and writes it to pin 11, with a 200ms delay between each iteration.

```
int valor = 0;

void setup() {
  pinMode (11, OUTPUT);
  pinMode (9, INPUT);
}

void loop() {
  valor = digitalRead(9);
  digitalWrite (11, val);
  delay(200);
}
```

Two callouts are present on the right side of the code editor:

- Sessão de SETUP**: Points to the `void setup()` function.
- Sessão de LOOP**: Points to the `void loop()` function.

The status bar at the bottom indicates '9' and 'Arduino/Genuino Uno on COM3'.

Arduino

SETUP:

Configurar o pino 13 como pino de saída

LOOP

Escrever no pino 13 HIGH (sinal 1)

Esperar 1000 ms

Escrever no pino 13 LOW (sinal 0)

Esperar 1000 ms

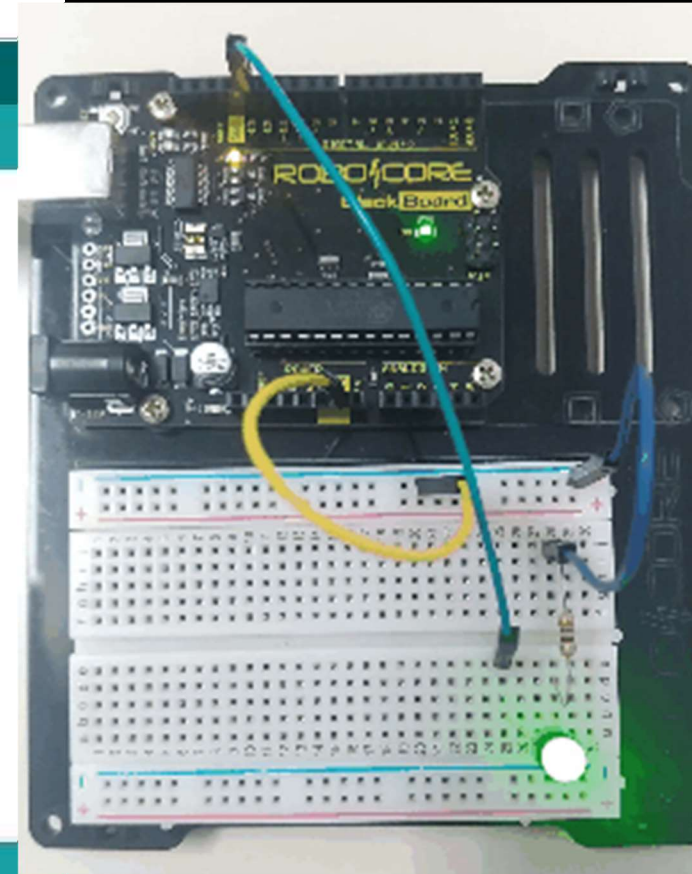
```
simple_blink | Arduino 1.8.10
File Edit Sketch Tools Help

simple_blink

void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13, HIGH);
  delay(1000)
  digitalWrite(13, LOW);
  delay(1000);
}

Done Saving.
```

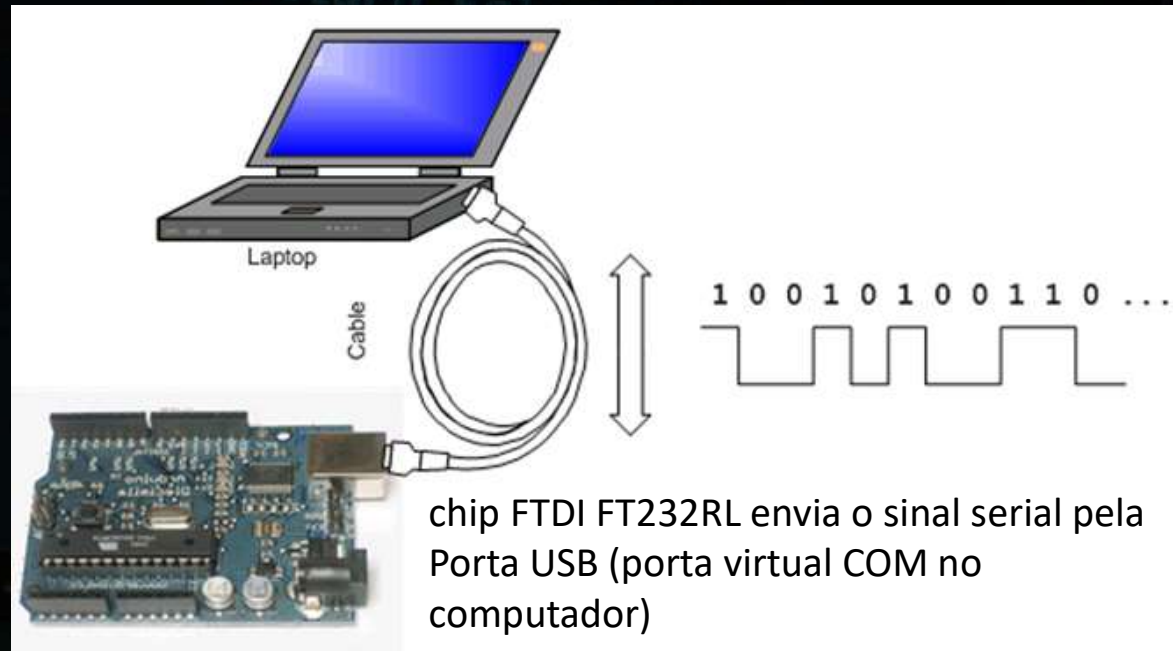


Arduino

COMUNICAÇÃO SERIAL



Usada para comunicação entre uma placa Arduino e um computador ou outros dispositivos. Todas as placas Arduino possuem pelo menos uma porta serial (também conhecida como UART ou USART).



Apesar do Arduino ser gravado via porta USB, a comunicação é serial.

O ATmega328P permite comunicação serial no padrão UART TTL (5 V), que está disponível nos pinos digitais 0 (RX) e 1 (TX).

Arduino



COMUNICAÇÃO SERIAL

A biblioteca `Serial()` do [Wiring](#) tem todas as funções necessárias para se implementar comunicação serial entre o Arduino e o PC.

Velocidades típicas de comunicação são: 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 e 115200 **BPS**.

SETUP:

Inicia a porta serial com a velocidade de 9600 BPS

LOOP

Escreve na porta serial a mensagem

Apenas uma aplicação pode utilizar a porta serial por vez.

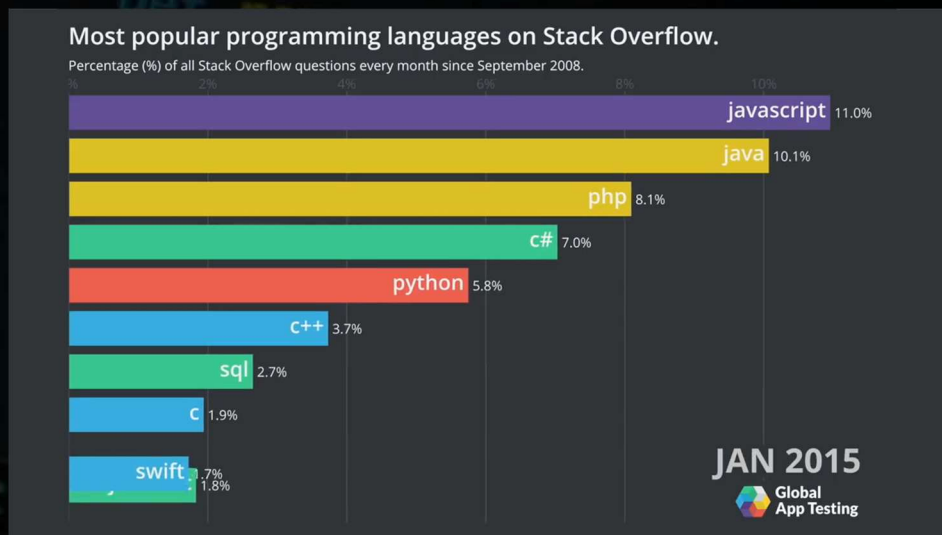


Com_serial.ino

```
1 void setup() {
2     // put your setup code here, to run once:
3     Serial.begin(9600);
4 }
5
6
7 void loop() {
8     // put your main code here, to run repeatedly:
9     Serial.print("Esta me ouvindo");
10    Serial.write(63);
11    Serial.print("\n");
12 }
13 }
```

Python

- Linguagem de programação orientada a objeto
- Lançada por Guido van Rossum em 1991
- Modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation
- O nome Python teve a sua origem no grupo humorístico britânico Monty Python

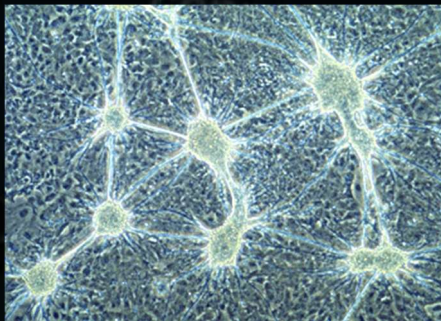
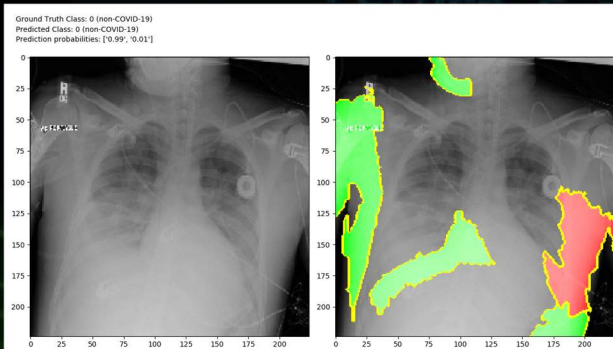


Python



“- Sistema Financeiro / Seguros ” : “ prever inadimplência, detectar fraudes, analisar tendência de mercado, liberar um crédito, etc. ”,

“Medicina” : “ analisar dados e sinais de um paciente para diagnosticar uma doença ou prever um problema de saúde”
- Sistema de Detecção da COVID através das imagens de Raio-X,



“Redes Neurais Artificiais (DeepLearning) ” : “Algoritmo utilizado no ML para extrair informações de dados brutos e representá-los através de um modelo matemático.”

Python

BIBLIOTECAS pySerial

Extensão de porta serial para Python (Win32, OSX, Linux, BSD, Jython, IronPython)

<https://pypi.org/project/pyserial/>

[Help](#)[Sponsors](#)[Log in](#)[Register](#)

pyserial 3.5

`pip install pyserial`

Released: Nov 23, 2020

https://pythonhosted.org/pyserial/pyserial_api.html#classes

Python

BIBLIOTECAS pySerial

- Mesma interface para todas as plataformas suportadas
- Configuração da porta através do Python
- Arquivos do pacote funcionam tipo API com "read" e "write" ("readline" etc. também são suportados).
- Arquivos do pacote são 100% Python
- A porta é configurada para transmissão binária. Isso torna esse módulo universalmente útil.

Compatibilidade:

- Python 2.7 ou mais atuais, incluindo Python 3.4 e mais atuais.

Instalação:

- pip install pyserial

Python

BIBLIOTECAS pySerial

https://pythonhosted.org/pyserial/pyserial_api.html#classes

```
class serial.Serial
```

```
__init__(port=None, baudrate=9600, bytesize=EIGHTBITS, parity=PARITY_NONE, stopbits=STOPBITS_ONE, timeout=None, xonxoff=False, rtscts=False, write_timeout=None, dsrdtr=False, inter_byte_timeout=None)
```

Parameters:

- **port** – Device name or None.
- **baudrate** (*int*) – Baud rate such as 9600 or 115200 etc.
- **bytesize** – Number of data bits. Possible values: FIVEBITS, SIXBITS, SEVENBITS, EIGHTBITS
- **parity** – Enable parity checking. Possible values: PARITY_NONE, PARITY_EVEN, PARITY_ODD, PARITY_MARK, PARITY_SPACE
- **stopbits** – Number of stop bits. Possible values: STOPBITS_ONE, STOPBITS_ONE_POINT_FIVE, STOPBITS_TWO
- **timeout** (*float*) – Set a read timeout value.
- **xonxoff** (*bool*) – Enable software flow control.
- **rtscts** (*bool*) – Enable hardware (RTS/CTS) flow control.
- **dsrdtr** (*bool*) – Enable hardware (DSR/DTR) flow control.
- **write_timeout** (*float*) – Set a write timeout value.
- **inter_byte_timeout** (*float*) – Inter-character timeout, None to disable (default).

Raises:

- **ValueError** – Will be raised when parameter are out of range, e.g. baud rate, data bits.
- **SerialException** – In case the device can not be found or can not be configured.

Python

BIBLIOTECAS pySerial

Ler o dado da porta serial que foi enviado pelo Arduino



SETUP:

Inicia a porta serial com a velocidade de 9600 BPS

LOOP

Escreve na porta serial a mensagem, o caractere "?" (tabela ASCII 63) e o código de nova linha "\n"

Com_serial.ino

```
1 void setup() {
2     // put your setup code here, to run once:
3     Serial.begin(9600);
4 }
5
6
7 void loop() {
8     // put your main code here, to run repeatedly:
9     Serial.print("Esta me ouvindo");
10    Serial.write(63);
11    Serial.print("\n");
12 }
13 }
```

Python

BIBLIOTECAS pySerial

Ler o dado da porta serial que foi enviado pelo Arduino



Importar a biblioteca serial

Criar um objeto `arduinoData` na porta `COM10` com velocidade `9600 BPS`

Loop enquanto a porta for legível (`while`)
Lê uma linha escrita na porta serial
(espera pelo código `"\n"` no final)

Mostra a mensagem e os dados da porta

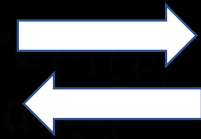
Comunica arduino serial python.py

```
1  #Importar bibliotecas Serial
2  import serial
3
4  # cria um objeto serial (arduinoData) na porta de comunicação
5  arduinoData = serial.Serial(port='com10', baudrate = 9600)
6
7  # Loop para Ler a porta Serial
8  while (arduinoData.readable()): ## verifica se a porta está
9      arduinoString = arduinoData.readline() #Ler uma linha da
10     print ('Valor lido: {} '.format(arduinoString)) ## Most
11     print ('Status Porta: {} '.format(arduinoData.isOpen()))
12     print ('Device conectado: {} '.format(arduinoData.name))
13     print ('Velocidade: {}'.format(arduinoData.baudrate))
```

Python

GRÁFICOS COM O PYTHON

Ler o dado da voltagem e mostrar num gráfico no Python



matplotlib
Version 3.4.1

matplotlib.pyplot

NumPy

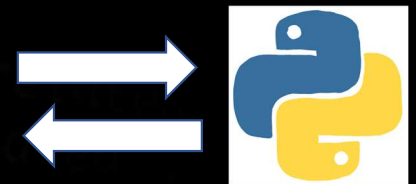


drawnow 0.72.5

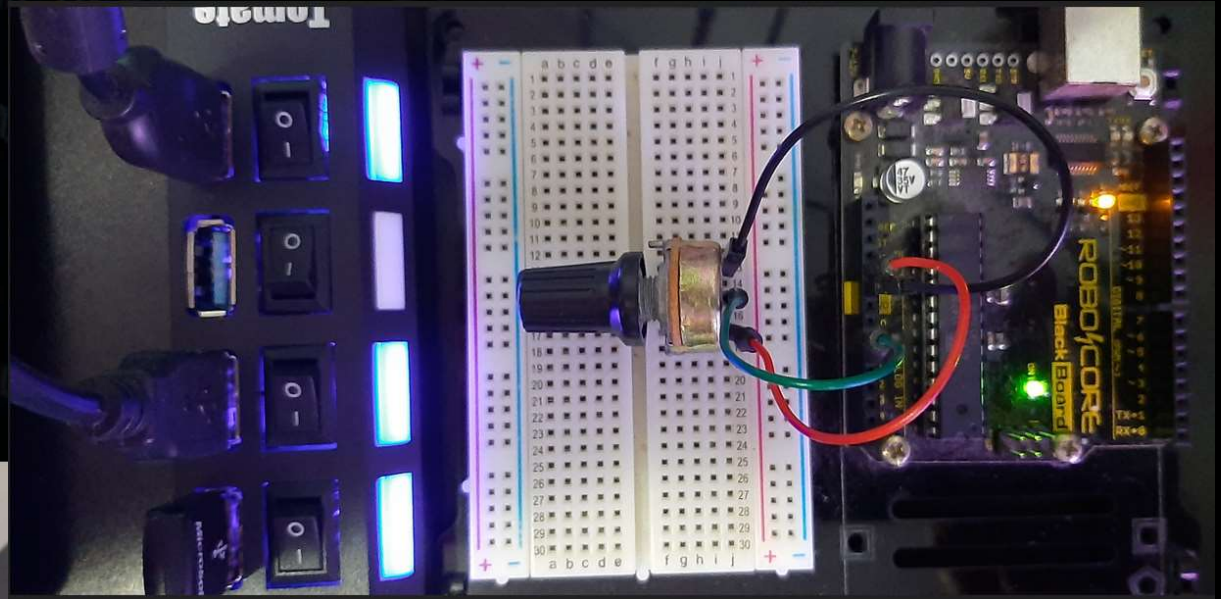
Python

GRÁFICOS COM O PYTHON

Ler o dado da voltagem e mostrar num gráfico no Python



```
serial potenciometro.py > ...  
1  #Importar bibliotecas  
2  
3  import serial  
4  import numpy as np  
5  import matplotlib.pyplot as plt  
6  from drawnow import *  
7  import time
```

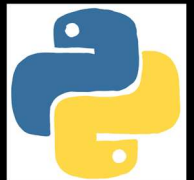
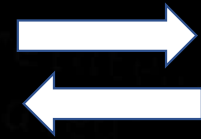


https://www.amazon.com.br/Hub-USB-portas-Indicador-Bot%C3%A3o/dp/B07S52472Q/ref=sr_1_1?mk_pt_BR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=Hub+USB+3.0+com+4+portas+com+Led+Indicador+e+Bot%C3%A3o+On%2FOff+-+Suporta+HD+at%C3%A9+1TB&qid=1629377984&s=electronics&sr=1-1

Python

GRÁFICOS COM O PYTHON

Ler o dado da voltagem e mostrar num gráfico no Python



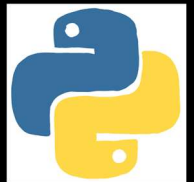
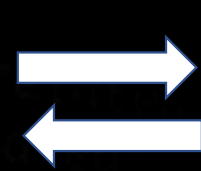
```
8 Volt = [] # cria uma lista vazia para guardar os valores de voltagem
9
10 arduinoData = serial.Serial(port='com10', baudrate = 9600) # cria um objeto serial (arduinoData)
11 plt.ion() #Configura o matplotlib no modo interativo para atualizar automatico
```

models.py

Python

GRÁFICOS COM O PYTHON

Ler o dado da voltagem e mostrar num gráfico no Python



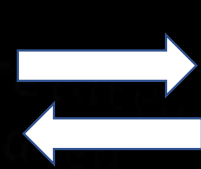
```
14 def makeFig(): # Criar uma função para o gráfico
15     plt.ylim(0, 7)
16     plt.title('Voltagem')
17     plt.grid(True)
18     plt.ylabel('V')
19     plt.plot(Volt, 'ro-')
```

models.py

Python

GRÁFICOS COM O PYTHON

Ler o dado da voltagem e mostrar num gráfico no Python



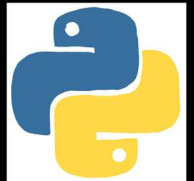
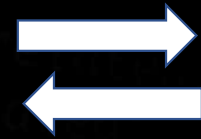
```
21 #Função para checar se o valor lido pode ser convertido a float
22 def check_float(potential_float):
23     try:
24         float(potential_float)
25         return True
26     except ValueError:
27         return False
```

```
connection = self.filter(from_user=user)
if not connection:
    connection = self.filter(from_user=
connection.delete()
models.py
```

Python

GRÁFICOS COM O PYTHON

Ler o dado da voltagem e mostrar num gráfico no Python

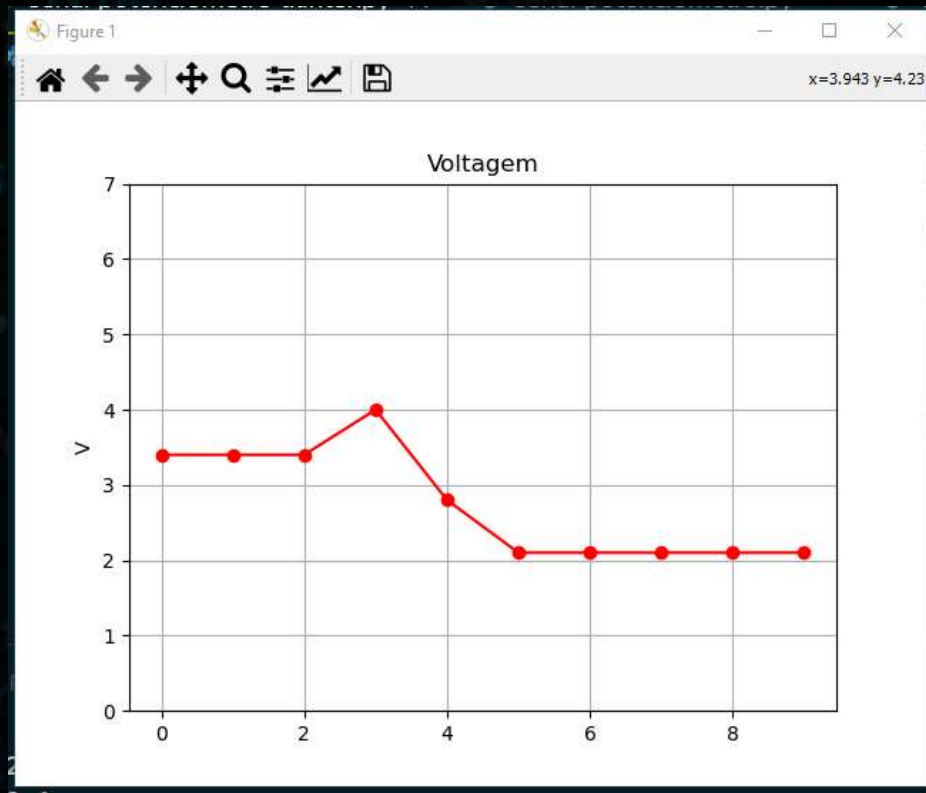
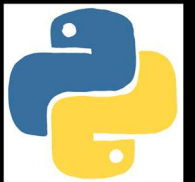
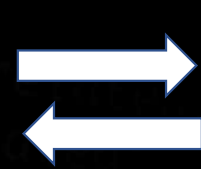


```
31 while True:
32     arduinoString = arduinoData.readline() #Ler uma linha da porta serial
33     dataArray = arduinoString[:3] #Ler os 3 primeiros caracteres (valor da temperatura)
34     if check_float(dataArray): # verifica se o valor lido pode ser convertido para float
35         voltagem = float(dataArray) # converte em float e guarda em temperatura
36         Volt.append(voltagem) # cria um array adicionando as leituras de Temp
37         drawnow(makeFig) # chama a função do gráfico
38         arduinoData.reset_input_buffer()
39         print(voltagem) # mostra valor da temperatura no terminal
40     time.sleep(1)
```


Python

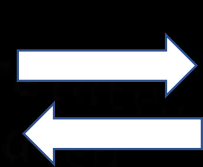
GRÁFICOS COM O PYTHON

Ler o dado da voltagem e mostrar num gráfico no Python



Python

USANDO A BIBLIOTECA TKINTER



Mostrar uma caixa de mensagem quando o valor de Voltagem atingir 5.0V

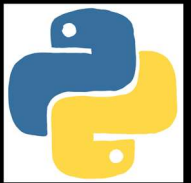
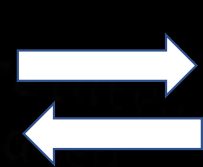
```
9  import tkinter
10 from tkinter import messagebox
```

```
12 root = tkinter.Tk()
13 root.withdraw()
```

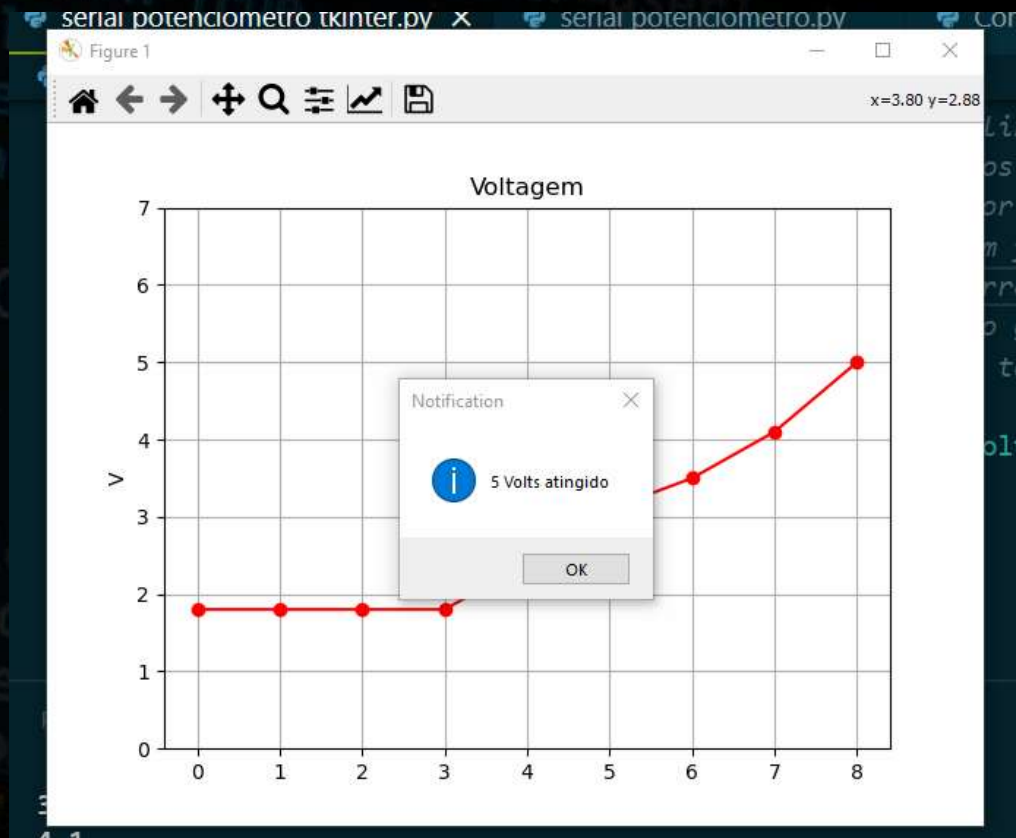
```
46     if voltagem == 5.0:
47         messagebox.showinfo("Notification", "5 Volts atingido")
48         root.update()
49         time.sleep(2)
50         arduinoData.reset_input_buffer()
```

Python

USANDO A BIBLIOTECA TKINTER

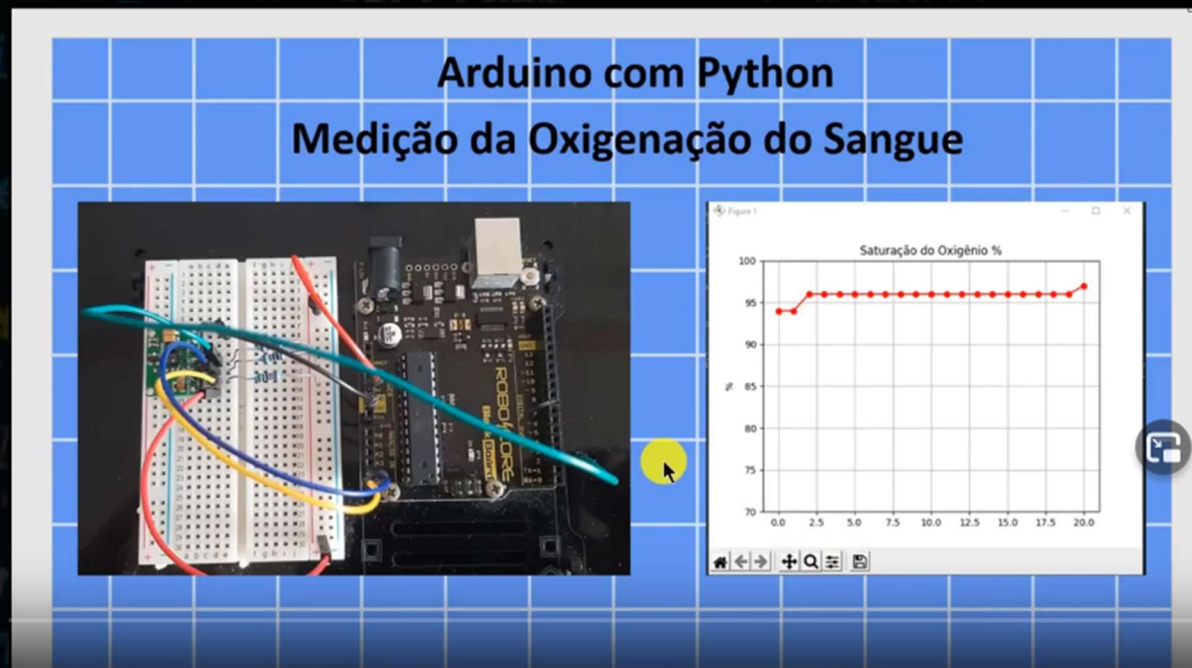
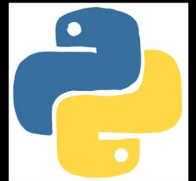
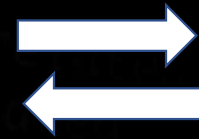


Mostrar uma caixa de mensagem quando o valor de Voltagem atingir 5.0V



Python

MEDIÇÃO DA OXIGENAÇÃO DO SANGUE



6 Sigma Excellence

<https://www.youtube.com/watch?v=Y2w79RZnliM>

Referências



**Welcome to
The Workshop!**



<https://www.youtube.com/channel/UCzml9bXoEM0itbcE96CB03w>



Andreas Spiess ✓

318K subscribers

<https://www.youtube.com/c/AndreasSpiess>



Paul McWhorter ✓

211K subscribers

<https://www.youtube.com/user/mcwhorpi>



sentdex ✓

1.03M subscribers

<https://www.youtube.com/user/sentdex>

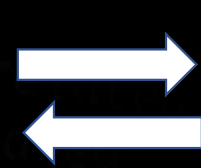


Corey Schafer ✓

753K subscribers

<https://www.youtube.com/user/schafer5>

Referências



Brincando com Ideias
359K subscribers

https://www.youtube.com/channel/UCcGk83PAQ5aGR7IVID_cBaw



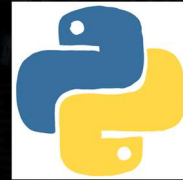
Electrolab
143K subscribers

<https://www.youtube.com/user/eecid>



Curso em Vídeo ✓
1.38M subscribers

<https://www.youtube.com/watch?v=S9uPNppGsGo>



OBRIGADO!
Q&A