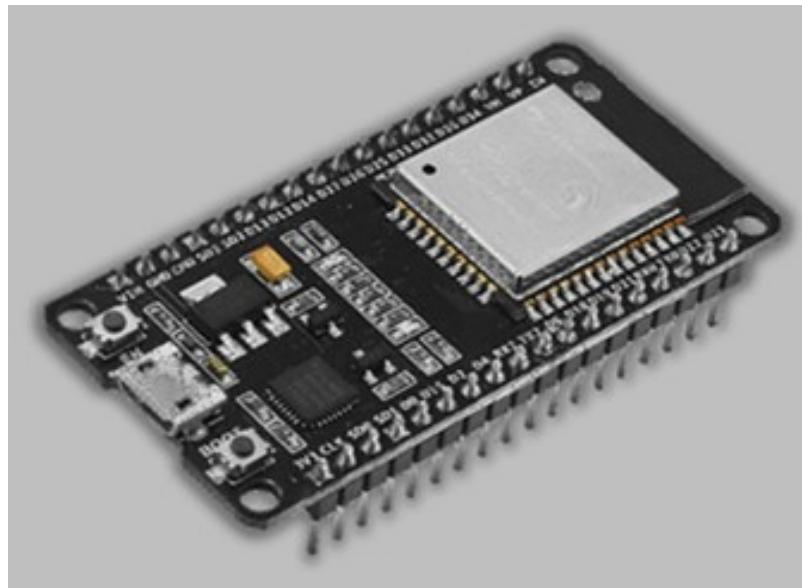


# ESP 32/ESP8266

## Microcontrolador, WiFi e Bluetooth

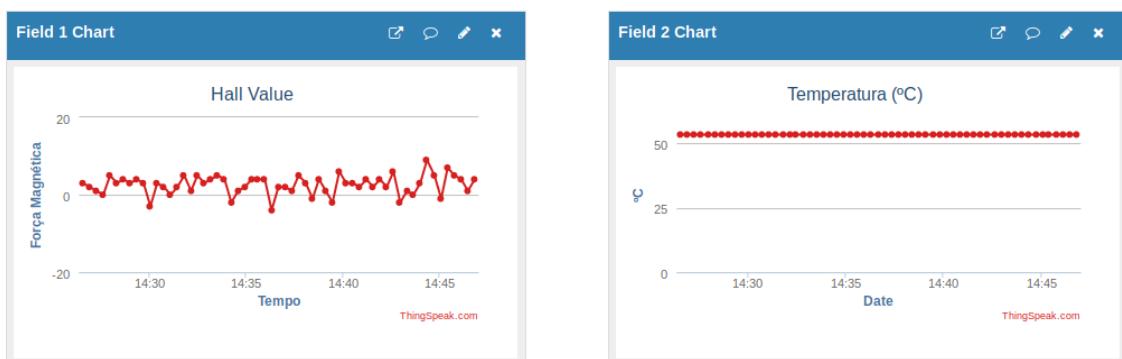


### Channel Stats

Created: [about 3 hours ago](#)

Last entry: [less than a minute ago](#)

Entries: 261



# Índice

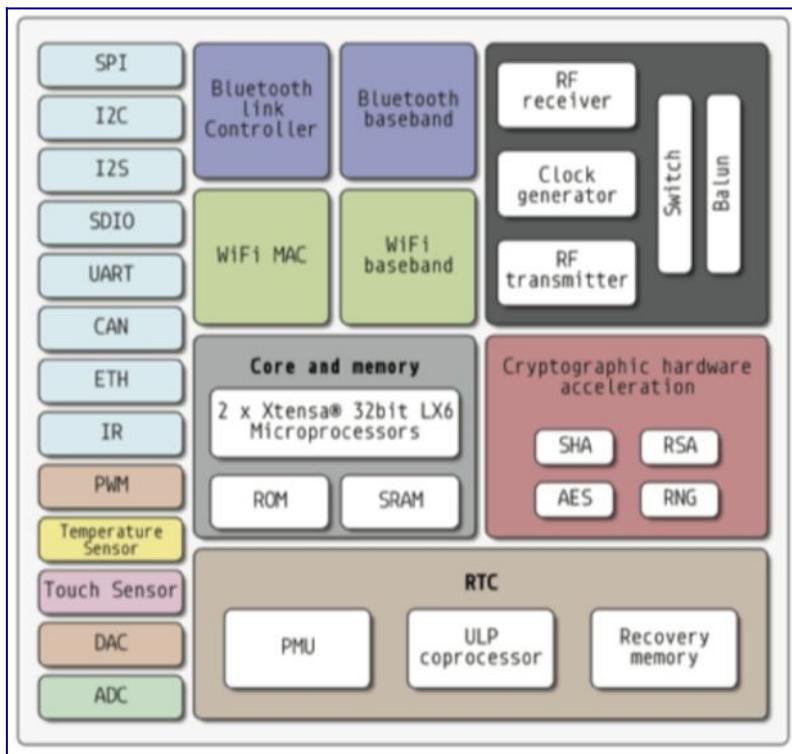
Introduction to ESP32 Board in Arduino IDE on Ubuntu Linux.....	1
Pinout.....	4
1 Programação através de Arduino IDE.....	6
Installing ESP32 Board in Arduino IDE (UBUNTU Linux).....	6
Installing ESP32 Board in Arduino IDE (Windows Install).....	9
Instalação de SPIFFS (File System).....	16
Exemplos de Programas (Arduino IDE).....	18
Exemplo 0: Led interno.....	18
Exemplo 1: Como obter o Mac address.....	18
Exemplo 1: Efeito de Hall.....	19
Exemplo 2: Efeito Capacitivo (Touch Sensor).....	22
Exemplo 3: Entrada Analógica.....	24
Exemplo 4: Entrada Analógica + Saída PWD.....	26
Exemplo 5: Sending and Receiving Data From a Local Web Page.....	28
Exemplo 6: ESP32 as HTTP Server using WiFi Access Point (AP) mode.....	32
Accessing the Web Server in AP mode.....	34
ThingSpeak.....	37
Exemplo 7: ESP ligado à aplicação de Telemóvel Blynk com data e hora dadas pela internet....	41
Como configurar Blynk e programar (resumo) - <a href="https://docs.blynk.cc/">https://docs.blynk.cc/</a> .....	42
Programa ESP32.....	43
Exemplo 8: Como ligar esp32 à rede eduroam.....	48
Exemplo 9: Enviar texto e comandos para LED's através de formulários HTML (como estação STA).....	49
Exemplo 10: Enviar texto e comandos para LED's através de formulários HTML (como Acess Point).....	61
Exemplo 11: Controlar LED e servo com webserver.....	72
2 Programação através de MicroPython.....	75
Como instalar o programa Thonny.....	75
Como usar o Thonny para programar MicroPython em ESP-32.....	76
1. Como enviar o firmware para o ESP-32.....	76
2. Como ver os ficheiros existentes no ESP-32.....	79
Comunicação:.....	80
Exemplo 1: Como criar funções e importá-la de outro ficheiro.....	81
Exemplo 2: Read Hall Effect Sensor and write to file.....	82
Escrever num ficheiro.....	83
Exemplo 3: Comando de um led através de AP (WebServer).....	84
css e html separado.....	87
Exemplo 3: DHT11.....	92
Exemplo 4: Comando de 4 relés.....	94
Exemplo 5: Web server on/off.....	98
7. Transmitir temperatura e humidade para o Thingspeak.....	103
main.py.....	103
config.py.....	104
6. node red mqqt.....	105
3 Programação através de Node-Red (rascunho).....	106
4 Plataformas IOT e processamento de dados (rascunho).....	110
Como instalar Blynk Server.....	110

ESP8266.....	113
Boards Manager.....	113
Prerequisites.....	113
Instructions.....	113
Using git version.....	113
Prerequisites.....	113
Instructions.....	113
Exemplos de Programas.....	115
Exemplo 1: Potenciômetro.....	115
Exemplo 2: Geolocalização - <a href="http://ip-api.com/">http://ip-api.com/</a> .....	116
Exemplo 3: Gravação de dados no local em cartão microSD.....	116
Exemplo 4: DHT11, NTPClient, WebServer (Lab212).....	119
Montagem.....	125
Código.....	125
Bonecos.....	130
Referências.....	131

# Introduction to ESP32 Board in Arduino IDE on Ubuntu Linux

ESP32 is highly-integrated with in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. ESP32 adds priceless functionality and versatility to your applications with minimal Printed Circuit Board (PCB) requirements. ESP32 can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces. Most of the **ESP32** framework is implemented. Most noticeable is the missing **analogWrite**. While analogWrite is on its way, there are a few other extra options that you can use:

## The ESP32 Main Characteristics



The ESP32 is an under US\$10 board with great advantages over similar IoT boards in the market.

This board has a dual processed microprocessor that helps a lot, because when one processor is handle communication, the other one is in charge of I/O control, for example. This feature will prevent some issues that happen with ESP8266, where the sole CPU needs stop controlling I/Os when handle with Comm. Besides, the ESP32 has integrated WIFI, BLUETOOTH, DAC, several ADC (not only one as the ESP8266), capacitive touch sensors, etc (give a look at above block diagram). And the good news is that Power Consumption is almost the same as ESP8266.

Bellow a chart that can show us its main characteristics, and differences when compared with ESP8266:

Specifications	ESP8266	ESP32
MCU	Xtensa® Single-Core 32-bit L106	Xtensa® Dual-Core 32-bit LX6 600 DMIPS
802.11 b/g/n Wi-Fi	Yes, HT20	Yes, HT40
Bluetooth	None	Bluetooth 4.2 and below
Typical Frequency	80 MHz	160 MHz
SRAM	160 kBytes	512 kBytes
Flash	SPI Flash , up to 16 MBytes	SPI Flash , up to 16 MBytes
GPIO	17	36
Hardware / Software PWM	None / 8 Channels	1 / 16 Channels
SPI / I2C / I2S / UART	2/1/2/2	4/2/2/2
ADC	10-bit	12-bit
CAN	None	1
Ethernet MAC Interface	None	1
Touch Sensor	None	Yes
Temperature Sensor	None	Yes
Working Temperature	-40°C – 125°C	-40°C – 125°C

Let's point its main properties for more details:

### Key Features:

- 240 MHz dual-core Tensilica LX6 microcontroller with 600 DMIPS
- Integrated 520 KB SRAM
- Integrated 802.11 b/g/n HT40 Wi-Fi transceiver, baseband, stack and LwIP
- Integrated dual mode Bluetooth (classic and BLE)
- 16 MB flash, memory-mapped to the CPU code space
- 2.3V to 3.6V operating voltage
- -40°C to +125°C operating temperature
- Onboard PCB antenna / IPEX connector for external antenna

### Sensors:

- Ultra-low noise analog amplifier
- Hall sensor
- 10x capacitive touch interfaces
- 32 kHz crystal oscillator

### 34 x GPIO:

- 3 x UARTs, including hardware flow control
- 3 x SPI
- 2 x I2S
- 18 x ADC input channels
- 2 x DAC
- 2 x I2C
- PWM/timer input/output available on every GPIO pin
- OpenOCD debug interface with 32 kB TRAX buffer
- SDIO master/slave 50 MHz
- Supports external SPI flash up to 16 MB
- SD-card interface support

### **Security Related:**

- WEP, WPA/WPA2 PSK/Enterprise
- Hardware accelerated encryption: AES/SHA2/Elliptical Curve Cryptography/RSA-4096

### **Performance:**

- Supports sniffer, Station, SoftAP and Wi-Fi direct mode
- Max data rate of 150 Mbps@11n HT40, 72 Mbps@11n HT20, 54 Mbps@11g, and 11 Mbps@11b
- Maximum transmit power of 19.5 dBm@11b, 16.5 dBm@11g, 15.5 dBm@11n
- Minimum receiver sensitivity of -97 dBm
- 135 Mbps UDP sustained throughput
- 5 µA power consumption in Deep-sleep

O ESP pode ser programado nas seguintes linguagens:

- LUA
- [Arduino IDE](#)
- [MicroPython](#)
- NodeJS (JavaScript) / Node-RED

Zerynth IDE (Python): <https://docs.zerynth.com/latest/official/core.zerynth.docs/installationguide/docs/index.html>

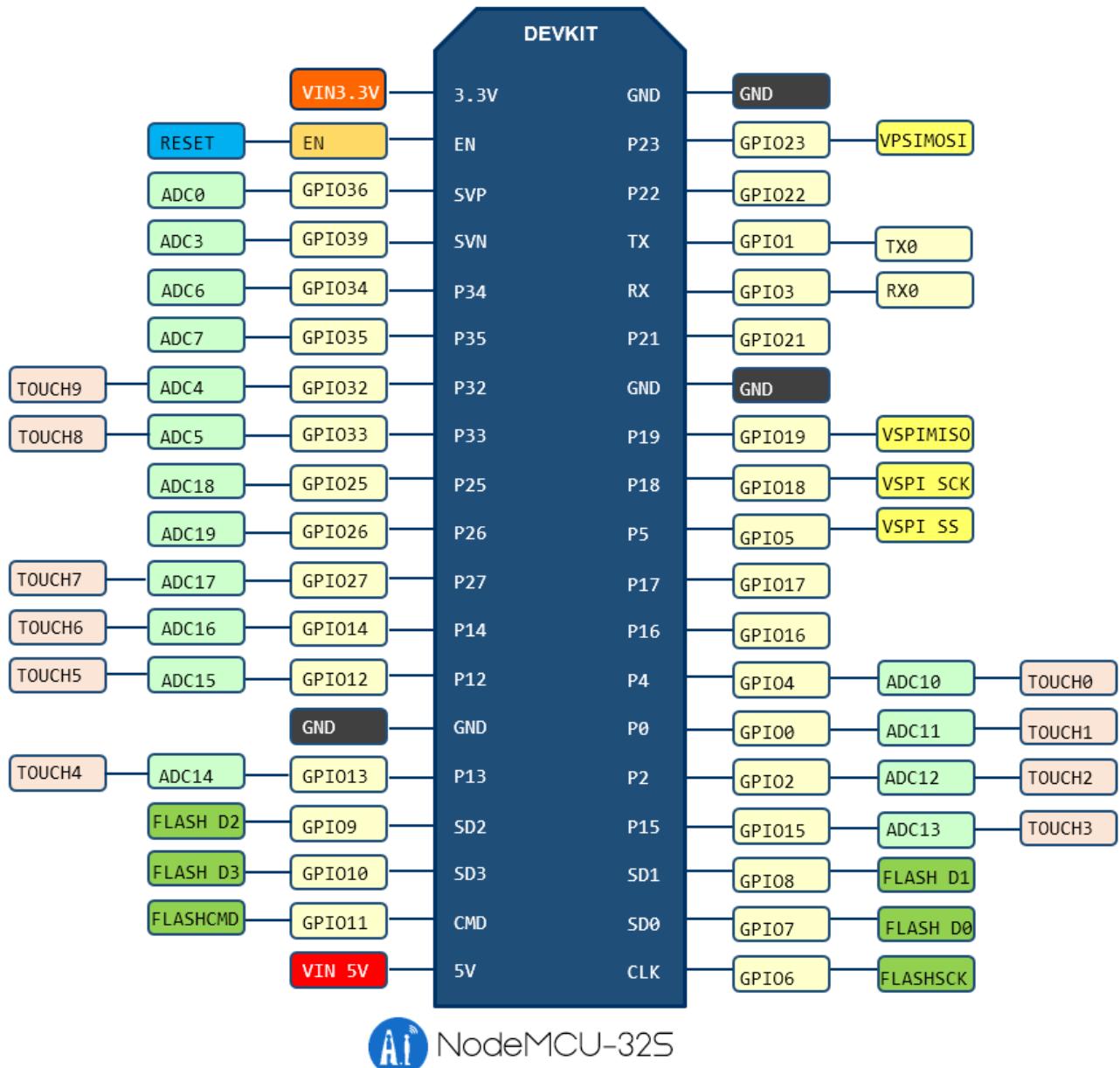
Chilipepr (LUA): <http://chilipeppr.com/esp32>

Micropython: <https://randomnerdtutorials.com/getting-started-micropython-esp32-esp8266/>

# Pinout

## PIN DEFINITION

www.ai-thinker.com



NodeMCU-32S

Paǵina Oficial: [https://wiki.ai-thinker.com/esp32/boards/nodemcu\\_32s](https://wiki.ai-thinker.com/esp32/boards/nodemcu_32s)

Especificações do Produto e Desenhos:

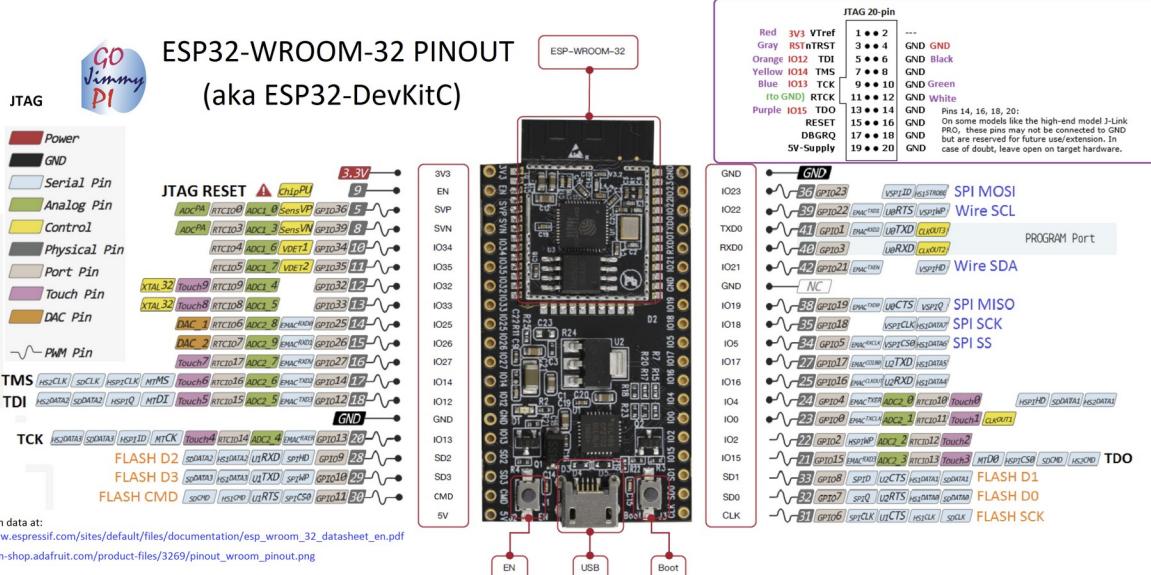
[https://wiki.ai-thinker.com/\\_media/esp32/docs/nodemcu-32s\\_product\\_specification.pdf](https://wiki.ai-thinker.com/_media/esp32/docs/nodemcu-32s_product_specification.pdf)

Firmware: [ai-thinker\\_nodemcu-32s\\_dio\\_32mbit\\_v1.0\\_20161101.7z](https://ai-thinker.com/nodemcu-32s_dio_32mbit_v1.0_20161101.7z)

Hardware: [nodemcu\\_32s\\_hardware\\_resources.7z](https://ai-thinker.com/nodemcu-32s_hardware_resources.7z)

NodeMCU: <https://github.com/nodemcu/nodemcu-firmware/tree/dev-esp32>

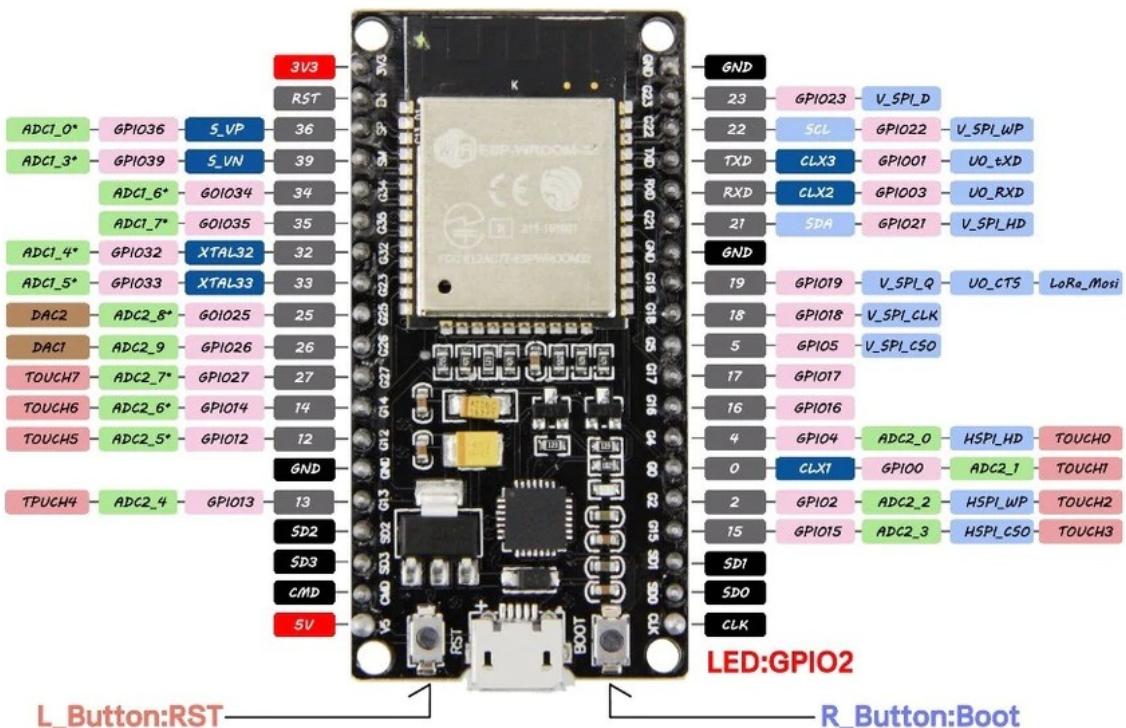
Arduino: <https://github.com/espressif/arduino-esp32>



Adapted from data at:

[http://www.espressif.com/sites/default/files/documentation/esp\\_wroom\\_32\\_datasheet\\_en.pdf](http://www.espressif.com/sites/default/files/documentation/esp_wroom_32_datasheet_en.pdf)

[https://cdn-shop.adafruit.com/product-files/3269/pinout\\_wroom\\_pinout.png](https://cdn-shop.adafruit.com/product-files/3269/pinout_wroom_pinout.png)



**ESP32 Rev1 Development  
WiFi+Bluetooth Board  
Pin Diagram**

# 1 Programação através de Arduino IDE

## Installing ESP32 Board in Arduino IDE (UBUNTU Linux)

In this tutorial we are installing ESP32 board in **Arduino IDE**. Its not same as [installing ESP8266](#). **ESP32** is dual core CPU in arduino IDE it will not support all the functions but its enough to make many projects. Para instalar o ESP8266 siga este link, que é semelhante ao que faremos com o ESP-32:

<https://arduino-esp8266.readthedocs.io/en/latest/installing.html>

### Step 1: Instalar última versão do arduino IDE

```
sudo apt-get update  
sudo apt-get install ubuntu-make
```

Allow non root user to use tty0 (USB to Serial converter) serial communication with ESP32

```
sudo usermod -a -G dialout $USER
```

**Faça o Logout do utilizador.**

**Step 1.2:** Install git. By default its installed with ubuntu linux installation

```
sudo apt-get install git
```

**Step 1.3:** Get repository for **get-pip.py**, is a bootstrapping script that enables users to install **pip**, setup tools, and wheel in Python environments that don't already **have** them.

```
wget https://bootstrap.pypa.io/get-pip.py
```

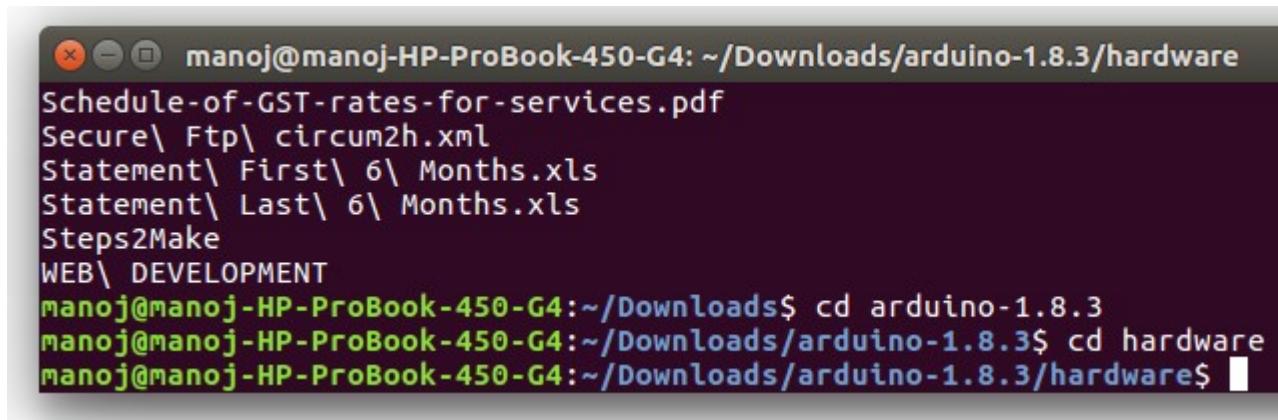
**Step 1.4:** Run python get-pip.py

```
sudo python get-pip.py
```

**Step 1.5:** Install **pySerial** is a Python API module to access the serial port. **pySerial** provides a uniform API across multiple operating systems, including Windows, Linux, and BSD.

```
sudo pip3 install pyserial
```

**Step 1.6:** On terminal go to you **Arduino** installation directory using **cd command**



```
manoj@manoj-HP-ProBook-450-G4: ~/Downloads/arduino-1.8.3/hardware  
Schedule-of-GST-rates-for-services.pdf  
Secure\ Ftp\ circum2h.xml  
Statement\ First\ 6\ Months.xls  
Statement\ Last\ 6\ Months.xls  
Steps2Make  
WEB\ DEVELOPMENT  
manoj@manoj-HP-ProBook-450-G4:~/Downloads$ cd arduino-1.8.3  
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3$ cd hardware  
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3/hardware$ █
```

**Step 1.7:** Create directory **espressif** in **arduino-1.8.3/hardware/** folder

```
mkdir -p ~/Arduino/hardware/espressif
```

**Step 1.8:** Change directory to **espressif**

```
cd ~/Arduino/hardware/espressif
```

**Step 1.9:** Clone **esp32 core** from git to **esperssif** directory

```
git clone https://github.com/espressif/arduino-esp32.git esp32
```

**Step 1.10:** After cloning you will find **esp32** directory

```
cd esp32
```

**Step 1.11:** Update submodules of ESP32

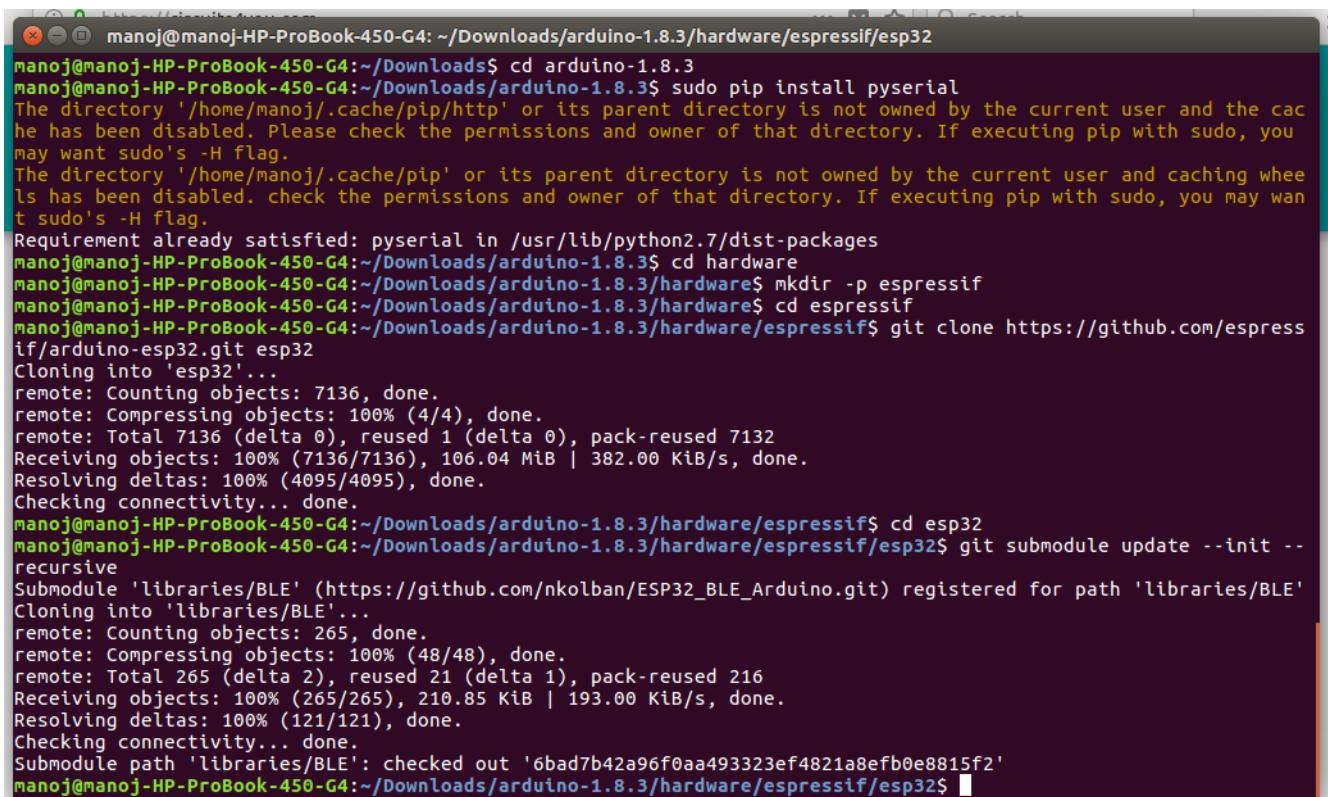
```
git submodule update --init --recursive
```

**Step 1.12:** Run following final commands

```
cd tools
```

```
python get.py
```

Here is my installation Screen shot



```
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3/hardware/espressif/esp32$ cd arduino-1.8.3
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3$ sudo pip install pyserial
The directory '/home/manoj/.cache/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
The directory '/home/manoj/.cache/pip' or its parent directory is not owned by the current user and caching wheels has been disabled. Check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
Requirement already satisfied: pyserial in /usr/lib/python2.7/dist-packages
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3$ cd hardware
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3/hardware$ mkdir -p espressif
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3/hardware$ cd espressif
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3/hardware/espressif$ git clone https://github.com/espressif/arduino-esp32.git esp32
Cloning into 'esp32'...
remote: Counting objects: 7136, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 7136 (delta 0), reused 1 (delta 0), pack-reused 7132
Receiving objects: 100% (7136/7136), 106.04 MiB | 382.00 KiB/s, done.
Resolving deltas: 100% (4095/4095), done.
Checking connectivity... done.
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3/hardware/espressif$ cd esp32
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3/hardware/espressif/esp32$ git submodule update --init --recursive
Submodule 'libraries/BLE' (https://github.com/nkolban/ESP32_BLE_Arduino.git) registered for path 'libraries/BLE'
Cloning into 'libraries/BLE'...
remote: Counting objects: 265, done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 265 (delta 2), reused 21 (delta 1), pack-reused 216
Receiving objects: 100% (265/265), 210.85 KiB | 193.00 KiB/s, done.
Resolving deltas: 100% (121/121), done.
Checking connectivity... done.
Submodule path 'libraries/BLE': checked out '6bad7b42a96f0aa493323ef4821a8efb0e8815f2'
manoj@manoj-HP-ProBook-450-G4:~/Downloads/arduino-1.8.3/hardware/espressif/esp32$
```

## Step 2: Restart Your arduino IDE

See in boards you will find ESP32 board. Go to Tools (Ferramentas), Placa and choose in this case NodeMCU-32S.

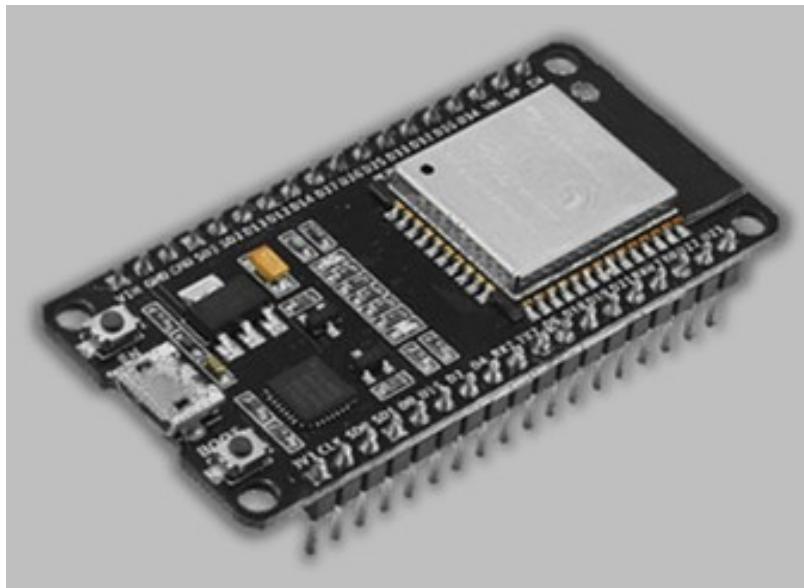
Temos um ESP-32 WROOM DEV Module. O ESP8266 é um AI Thinker com 4mb de ram.



Change Upload Speed to 115200 and search for the right Porta to make the connection.

# Installing ESP32 Board in Arduino IDE (Windows Install)

## Driver conversor USB-Série para ESP32 e ESP8266



Após a ligação do NodeMCU-32S no computador ele será identificado e a busca por drivers de instalação vai iniciar. Por padrão, o Windows vai procurar pelos drivers na internet e se ele encontrar a instalação vai ocorrer de forma automática.

Caso o Windows não consiga encontrar os drivers para instalação da placa, então terá que instalar manualmente.

Faça download do driver no link abaixo:

[Driver CP2102 para Windows](#)

Após o download faça a instalação dos drivers. Terminado a instalação, seu NodeMCU-32S já está pronto para ser usado.

A IDE do Arduino (Programa do Arduino) é uma ferramenta multiplataforma escrita em Java derivada dos projetos Processing e Wiring. Inclui um editor de código com recursos de realce de sintaxe, parênteses correspondentes e identação automática, sendo capaz de compilar e carregar programas para a placa com um único clique. Por isso não há a necessidade de editar Makefiles ou correr programas em ambientes de linha de comando.

Através da IDE do Arduino é possível programar (utilizando a linguagem do Arduino) para o NodeMCU-32S. Dessa forma, será possível trabalhar num ambiente mais simples, que é bastante conhecido por quem já programa para Arduino ou C / C++.

Antes de efetuar a instalação da IDE é necessário fazer a instalação do Java. Caso já possua o Java instalado, verifique se o mesmo está atualizado.

Para download do Java, basta aceder <https://www.java.com/> e fazer o download do instalador.

Feito a instalação ou atualização do Java, já é possível fazer a instalação da IDE do Arduino.

Para fazer o download da versão atualizada da IDE do Arduino para Windows basta aceder <https://www.arduino.cc/en/Main/Software> e efetuar a instalação.

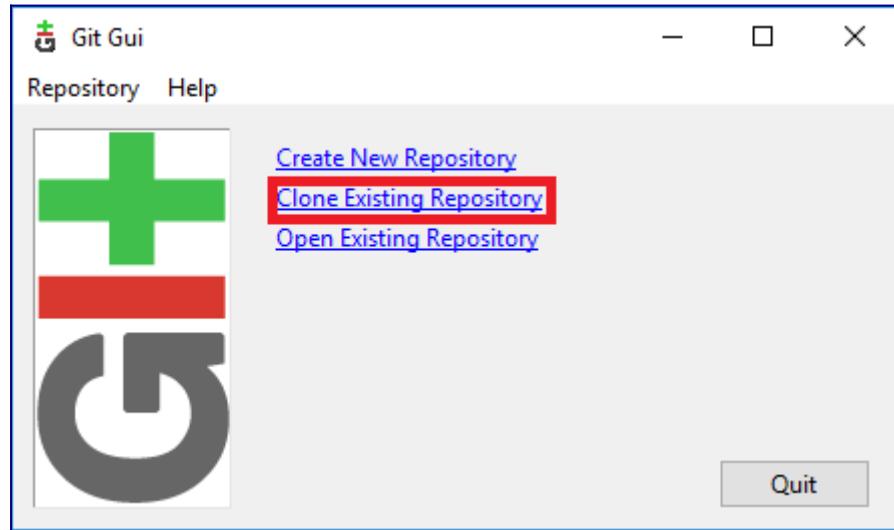
Após a instalação da IDE, será necessário preparamos os Ficheiros necessários para que a IDE do Arduino possa reconhecer o NodeMCU-32S.

Faça o download do Git:

<https://git-scm.com/download/win>

Após o download, faça a instalação do programa. Terminado a instalação, acesse o caminho “C:\Program Files\Git\cmd” e execute “git-gui”.

Com o programa aberto, clique em “Clone Existing Repository”:



Na janela que abrir, insira as informações da seguinte forma:

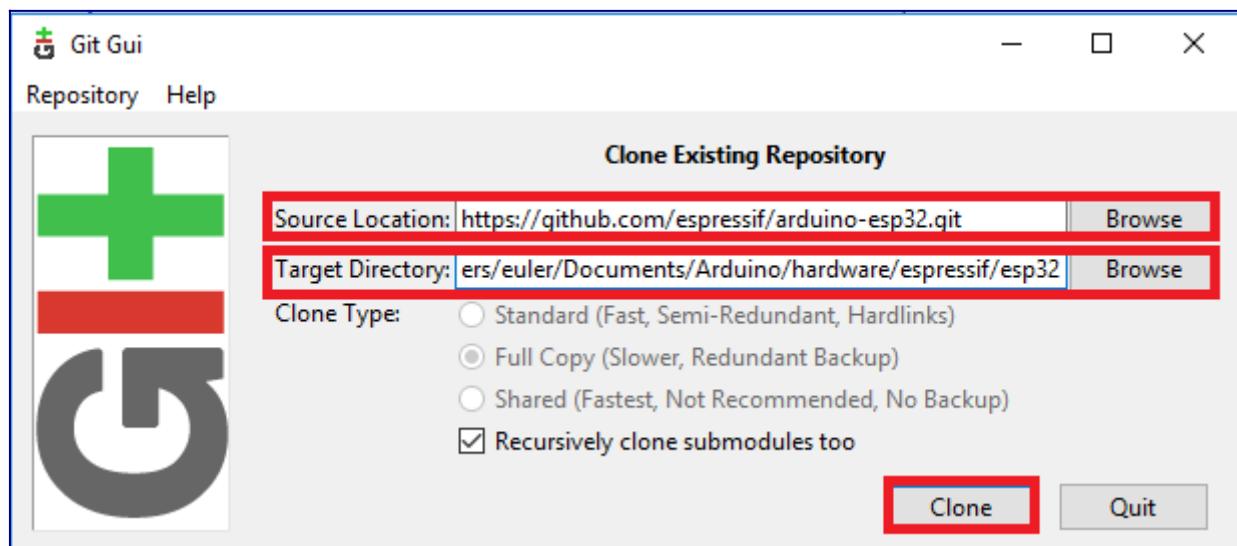
Source Location:

<https://github.com/espressif/arduino-esp32.git>

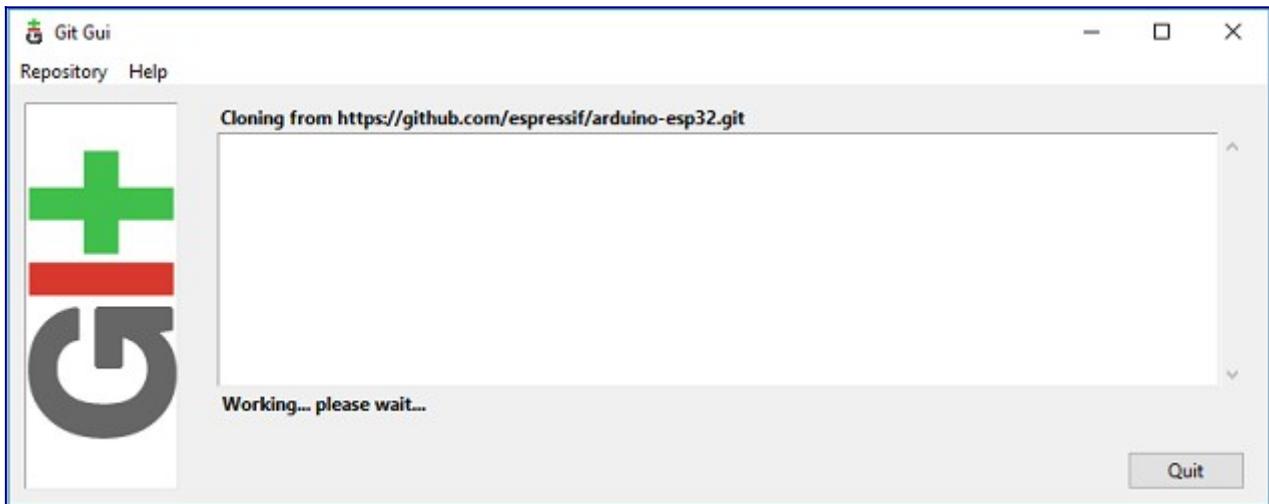
Target Directory:

C:/Users/[Utilizador-PC]/Documents/Arduino/hardware/espressif/esp32

**OBS: Em “Target Directory:” apague o [USUARIO-PC] e insira o nome de utilizador do seu computador para que o caminho possa ser válido.**

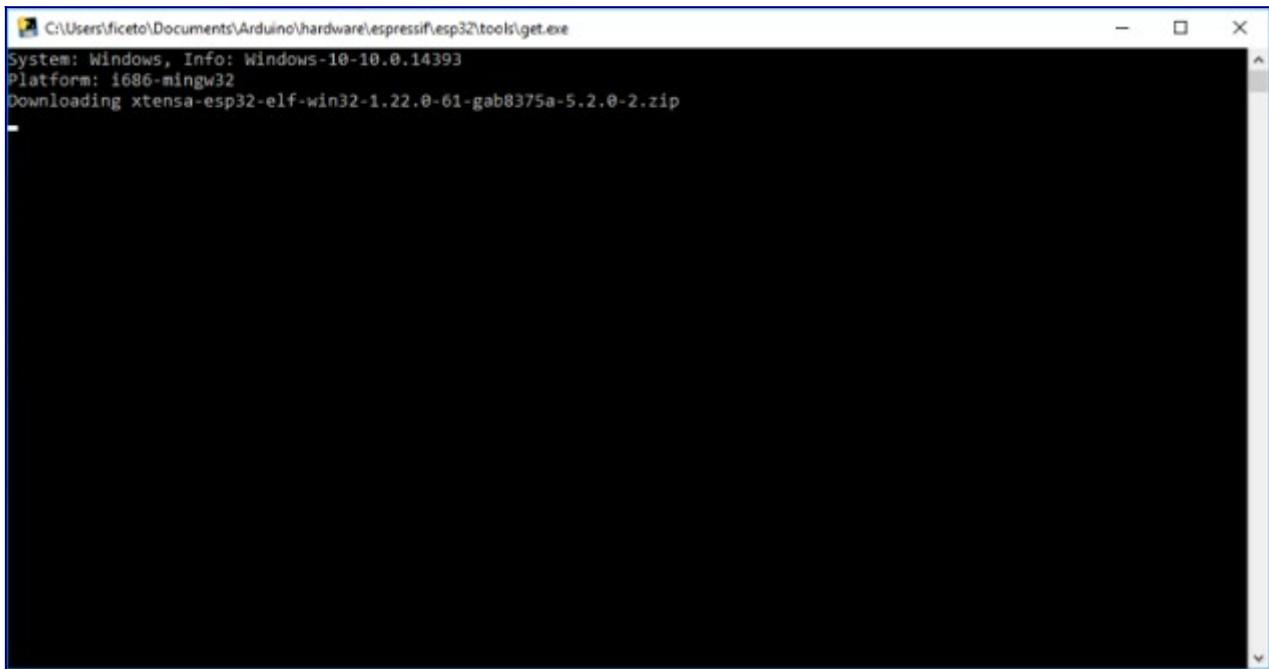


Após inserir as informações, clique em “Clone” para que o processo de cópia do repositório seja realizado:



Este processo é um pouco demorado, portanto paciência.

Finalizado a clonagem do repositório, acesse o caminho “C:/Users/[ USUARIO-PC]/Documents/Arduino/hardware/espressif/esp32/tools” (**lembre-se de mudar [USUARIO-PC] para o nome de utilizador do seu computador**) e execute o “get.exe” e aguarde:



Este processo também é demorado.

Terminado, conecte o NodeMCU-32S ao computador e abra a IDE do Arduino.

Com a IDE aberta, será mostrado algumas opções na barra de ferramentas da mesma. Abaixo é feito um apanhado geral (**das opções relevantes**) do menu Ficheiro e Ferramentas:

### 1. Ficheiro

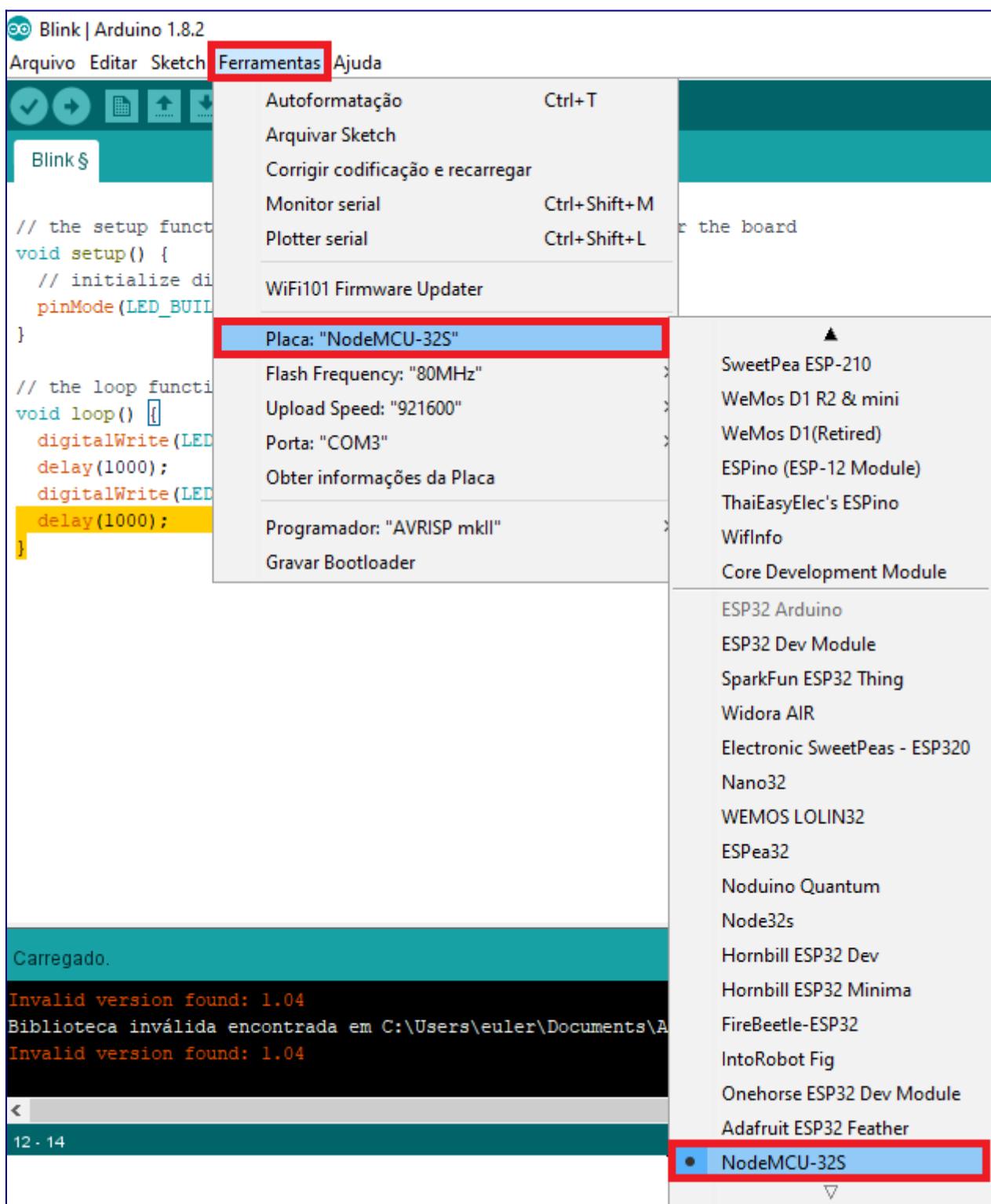
- **Novo:** abre uma nova instância da IDE.
- **Abrir:** carrega na IDE projetos guardados no computador.
- **Exemplos:** disponibiliza diversos exemplos que são separados por pastas e bibliotecas. Os exemplos estão em submenus e para abrir algum deles, basta selecionar e clicar.
- **Fechar:** fecha a instância atual.

- **Guardar:** guarda as últimas modificações do seu projeto.
- **Guardar como:** se o projeto já foi guardado, mas deseja Guardar noutro local, essa é a opção indicada.
- **Preferências:** configurações gerais da IDE.
- **Sair:** encerra definitivamente a IDE.

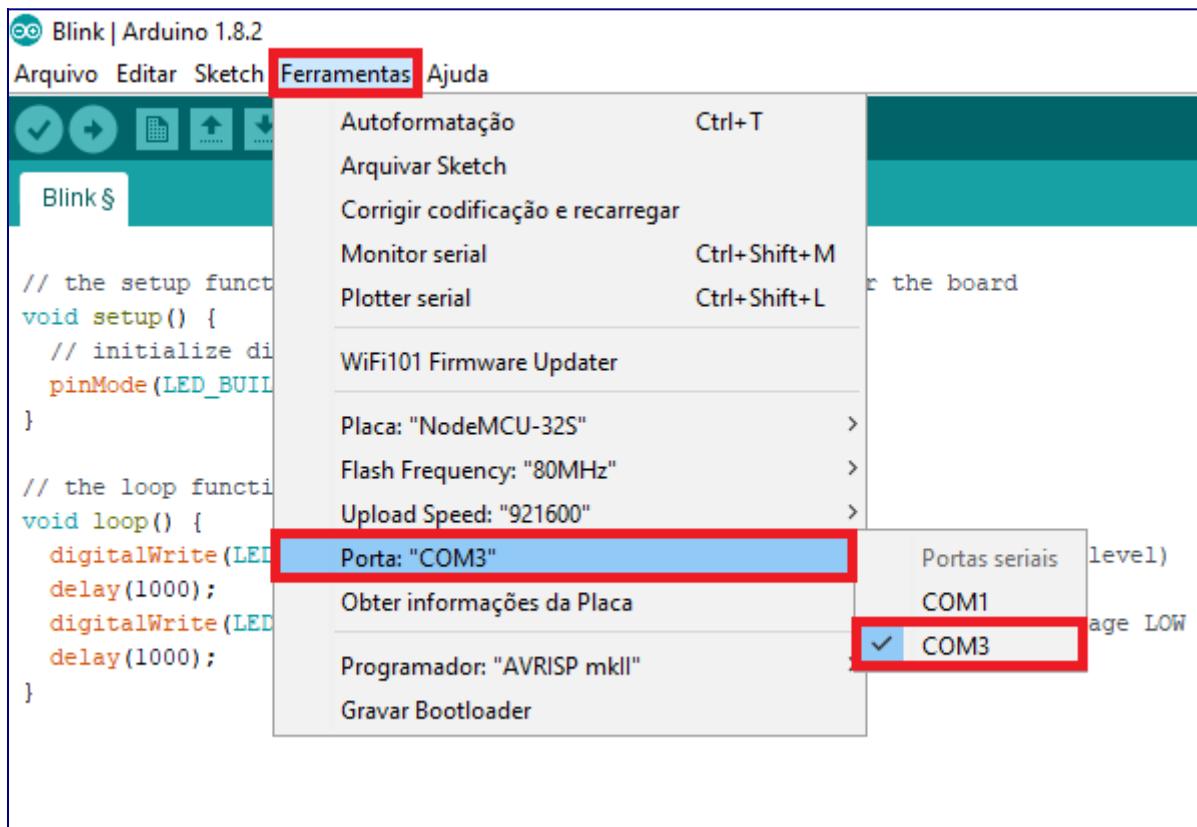
## 2. Ferramentas

- **Monitor série:** executa o terminal série que auxilia no recolha e envio de dados para a placa sem a necessidade de recorrer a uma ferramenta externa.
- **Placa:** possibilita selecionar o modelo da placa que está ligado ao computador.
- **Porta:** possibilita selecionar a porta COM em que a placa está recebendo/enviando informações.

No menu “Ferramentas” selecione a opção “Placas”, role a lista até o final e selecione “NodeMCU-32S”:



Ainda no menu “Ferramentas”, selecione a opção porta e marque a porta COM em que sua placa foi alocada:



Veja que a minha placa foi alocada na COM3, porém, o seu NodeMCU-32S pode ter sido alocado em uma COM de outro valor. Caso não saiba em qual porta COM sua placa foi alocada, basta retornar no menu Iniciar do Windows, aceder a opção Dispositivos e Impressoras e verificar a porta em que seu NodeMCU-32S está ligado e retornar na IDE e selecionar a porta COM.

Feito essas configurações, sua IDE está pronta para enviar os códigos ao NodeMCU-32S.

Para um teste rápido, na IDE clique no menu “Ficheiro”, selecione a opção “Exemplos”, em seguida “01.Basics” e selecione “Blink”. Uma nova janela da IDE vai abrir. Faça a conferência das informações para garantir que nada foi alterado na placa e na porta. Se atente ao rodapé da IDE que mostra a placa e a porta COM que está configurada na IDE:



Agora, clique no botão de upload do código para o NodeMCU-32S e aguarde o código ser carregado na placa.

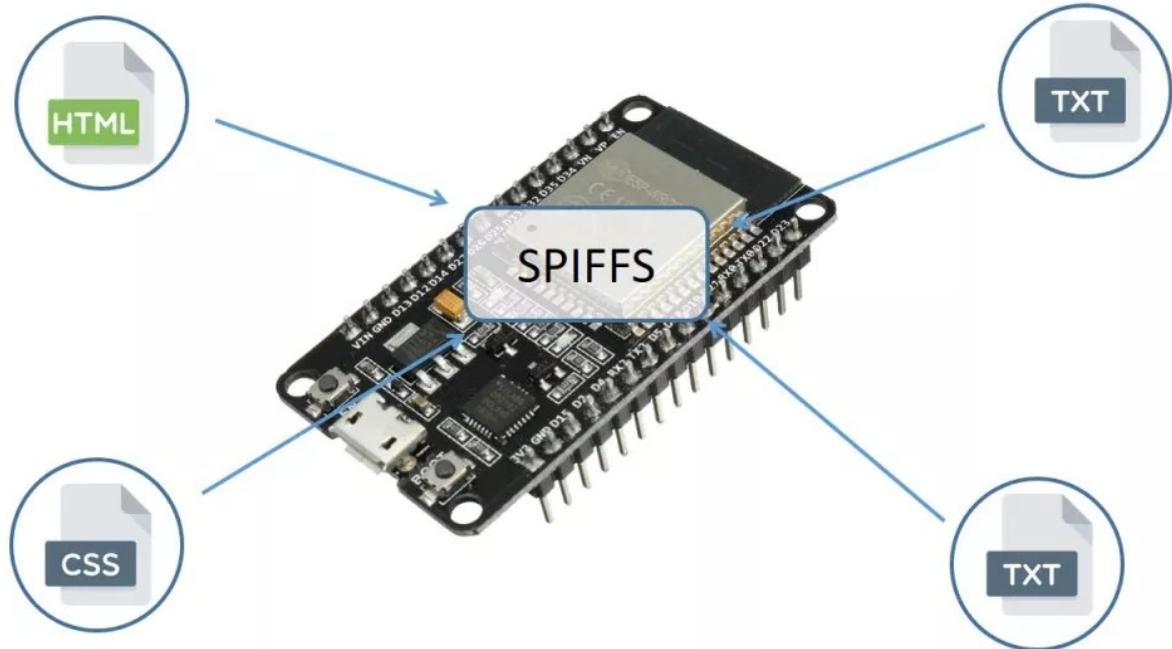
Terminado o carregamento, o LED da placa irá piscar com intervalo de 1 segundo:

Sempre que abrir a IDE, faça a conferência das informações para garantir que nada foi alterado. Geralmente, feito a configurações a primeira vez ela vai se manter sem necessidade de fazer alteração.

Reforçando: sem executar os passos de definição de placa e porta, caso você escreva o código ou utilize algum exemplo de código e tente carregar no NodeMCU-32S, o carregamento do código para a placa não será bem sucedido e um erro será mostrado na IDE.

# Instalação de SPIFFS (File System)

<https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/>



Fazer o download dos seguintes ficheiros para a pasta:

ESP-32

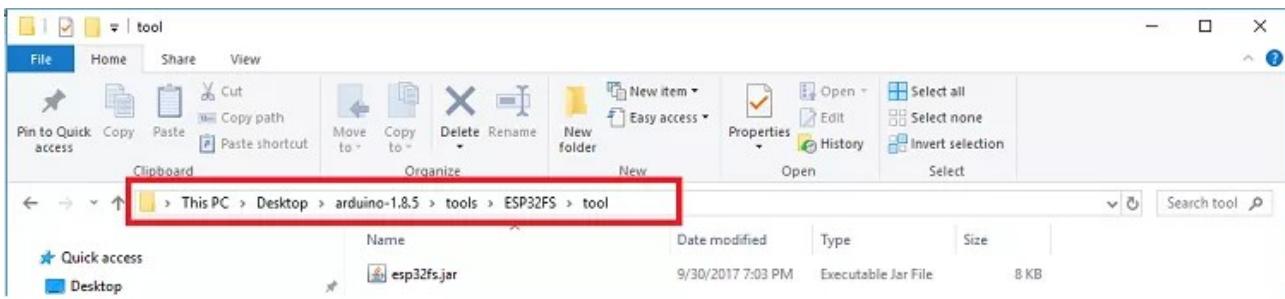
<https://github.com/me-no-dev/arduino-esp32fs-plugin/releases/>

ESP-8266

<https://github.com/esp8266/arduino-esp8266fs-plugin/releases>

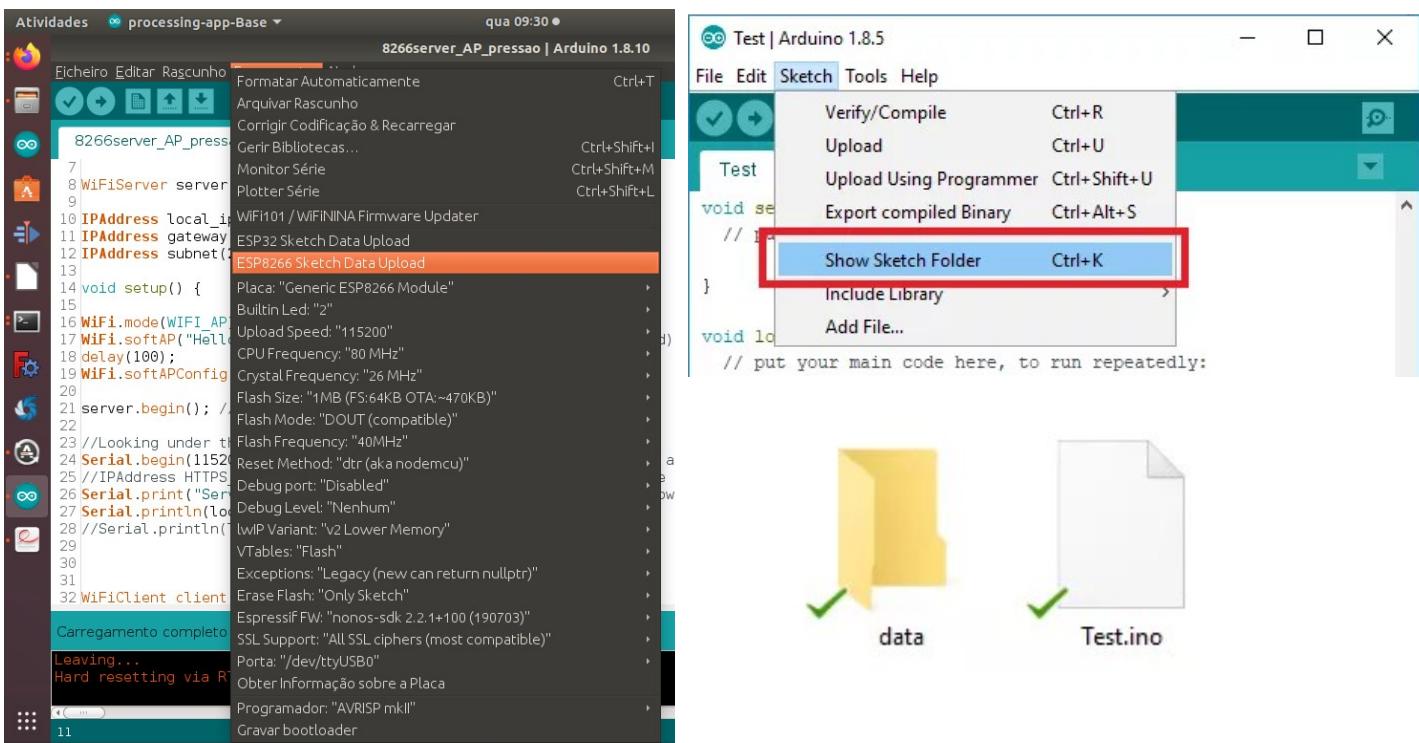
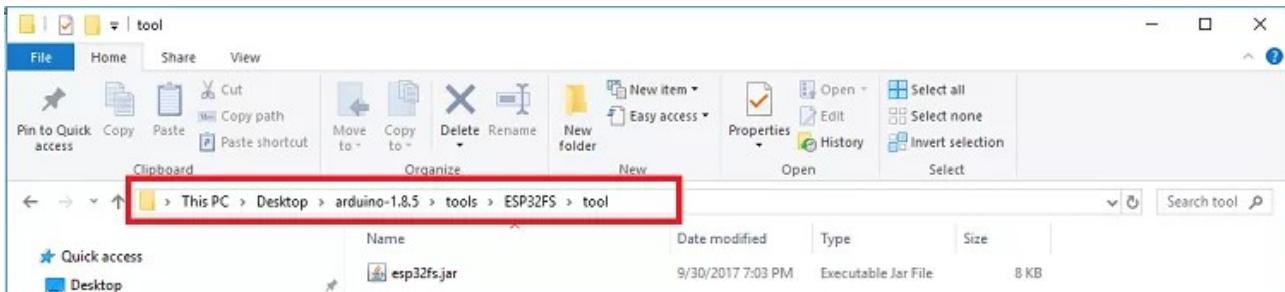
Descomprimir os ficheiros para a pasta Tools na em Arduino IDE:

📁 > arduino-1.8.5 > tools



Reinicie o arduino IDE.

Depois verifique se o plugin está bem instalado Tools → ESP32 Sketch Data Upload:



Colocar os ficheiros na pasta “Show Sketch Folder” e criar a pasta data, que é onde se coloca os ficheiros que ficaram na SpiFFS.

**Note:** in some ESP32 development boards you need to keep the ESP32 on-board “**BOOT**” button pressed while it’s uploading the files. When you see the “Connecting .....\_\_\_\_\_.....” message, you need to press the ESP32 on-board “**BOOT**” button.

# **Exemplos de Programas (Arduino IDE)**

## **Exemplo 0: Led interno**

```
int LED_BUILTIN = 2;  
void setup() {  
pinMode (LED_BUILTIN, OUTPUT);  
}  
void loop() {  
digitalWrite(LED_BUILTIN, HIGH);  
delay(1000);  
digitalWrite(LED_BUILTIN, LOW);  
delay(1000);  
}
```

## **Exemplo 1: Como obter o Mac adress**

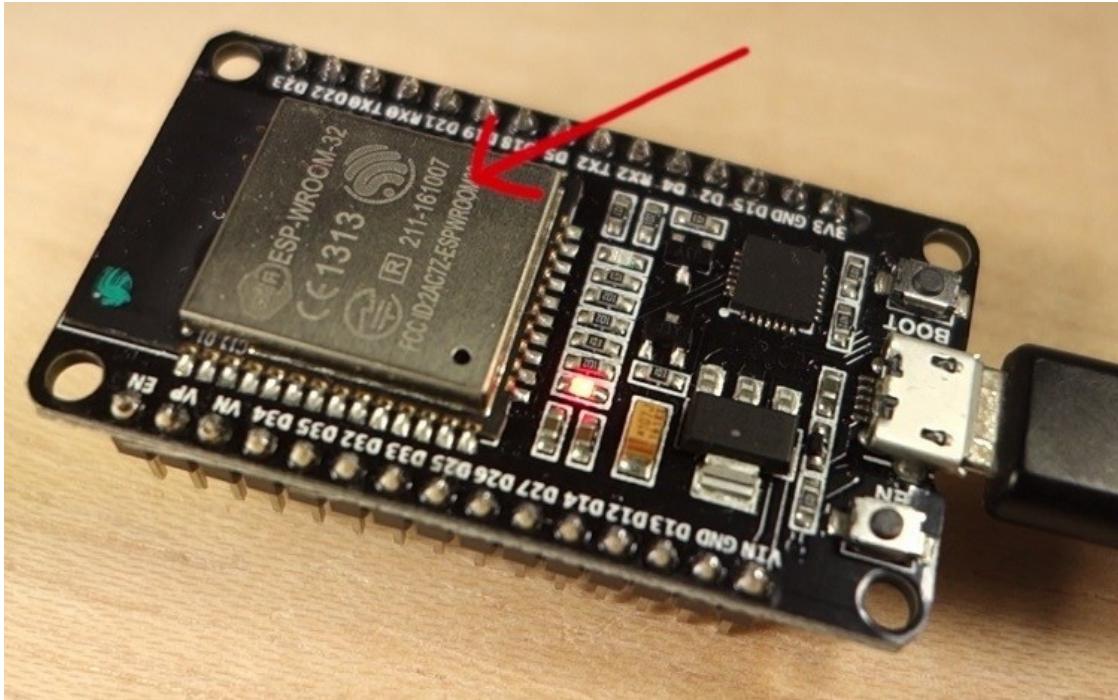
```
16:01:58.032 -> ESP Board MAC Address: B4:E6:2D:F5:F4:69  
16:01:59.029 -> teste
```

```
#ifdef ESP32  
    #include <WiFi.h>  
#else  
    #include <ESP8266WiFi.h>  
#endif  
  
void setup(){  
Serial.begin(115200);  
Serial.println();  
Serial.print("ESP Board MAC Address: ");  
Serial.println(WiFi.macAddress());  
delay(1000);  
Serial.println("teste");  
}  
  
void loop(){  
}
```

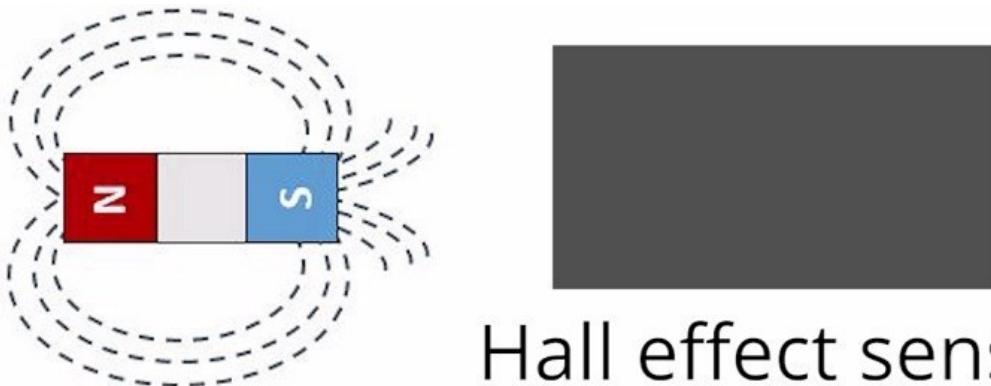
## Exemplo 1: Efeito de Hall

### The ESP32 Hall Effect Sensor

The ESP32 features a built-in hall effect sensor located behind the metal lid of the ESP32 chip as shown in the following figure.



A hall effect sensor can detect variations in the magnetic field in its surroundings. The greater the magnetic field, the greater the sensor's output voltage.



Hall effect sensor

The hall effect sensor can be combined with a threshold detection to act as a switch, for example. Additionally, hall effect sensors are mainly used to:

- Detect proximity;
- Calculate positioning;
- Count the number of revolutions of a wheel;
- Detect a door closing;
- And much more.

[Read Hall Effect Sensor – Arduino IDE](#)

Reading the hall effect sensor measurements with the ESP32 using the Arduino IDE is as simple as using the `hallRead()` function.

In your Arduino IDE, go to **File > Examples > ESP32 > HallSensor** sketch:

```
// Simple sketch to access the internal hall effect detector on the esp32.  
// values can be quite low.  
// Brian Degger / @sctv  
  
int val = 0;  
  
void setup() {  
    Serial.begin(9600);  
}  
  
// put your main code here, to run repeatedly  
void loop() {  
    // read hall effect sensor value  
    val = hallRead();  
    // print the results to the serial monitor  
    Serial.println(val);  
    delay(1000);  
}
```

[Hall\\_Effect\\_Sensor.ino](#) → [hello@ruisantos.me](mailto:hello@ruisantos.me) curso ESP-32

This example simply reads the hall sensor measurements and displays them on the Serial monitor.

```
val = hallRead();  
Serial.println(val);
```

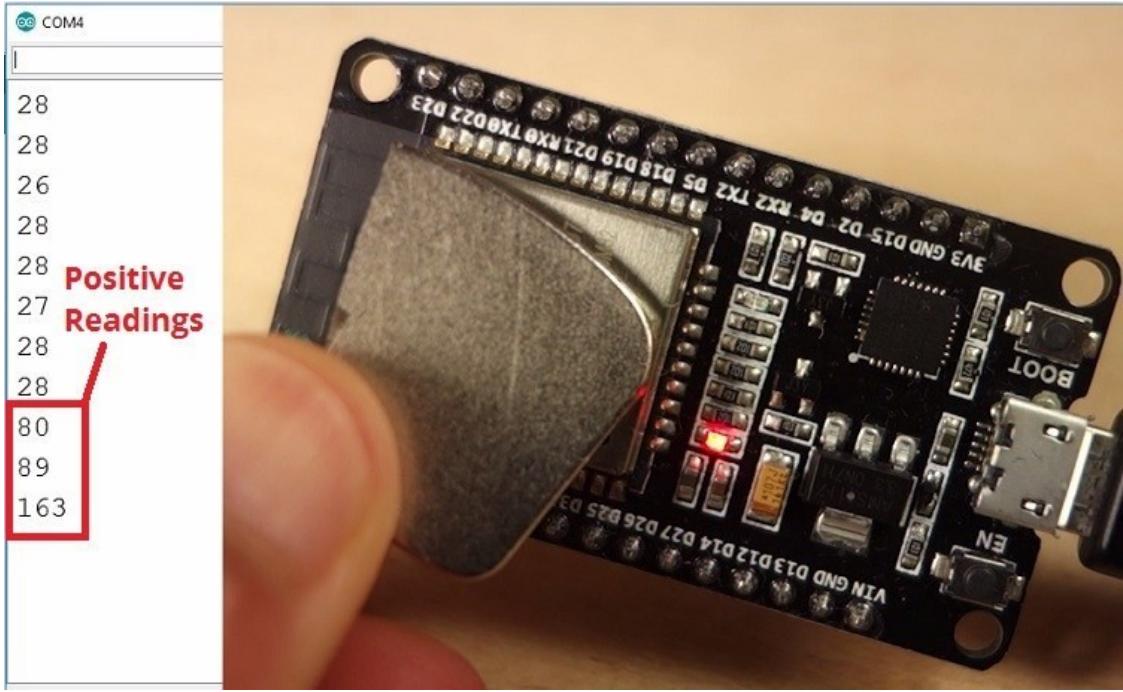
Add a delay of one second in the loop, so that you can actually read the values.

```
delay(1000);
```

Upload the code to your ESP32 board:

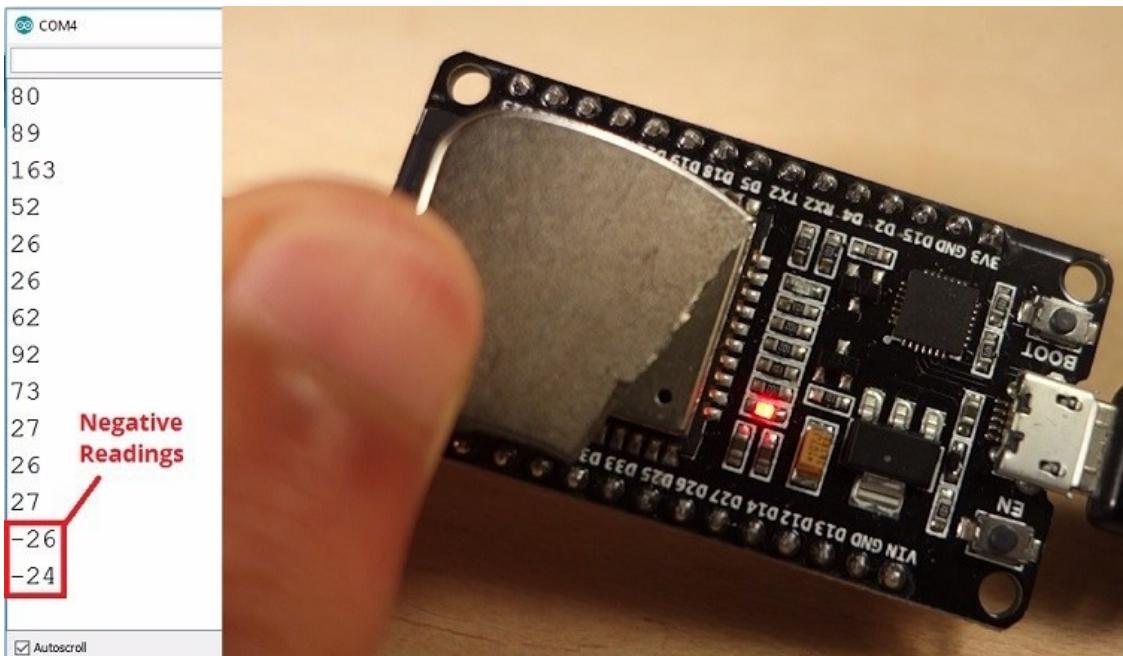
## Demonstration

Once the upload is finished, open the Serial Monitor at a baud rate of 9600. Approximate a magnet to the ESP32 hall sensor and see the values increasing...



Or decreasing depending on the magnet pole that is facing the sensor:

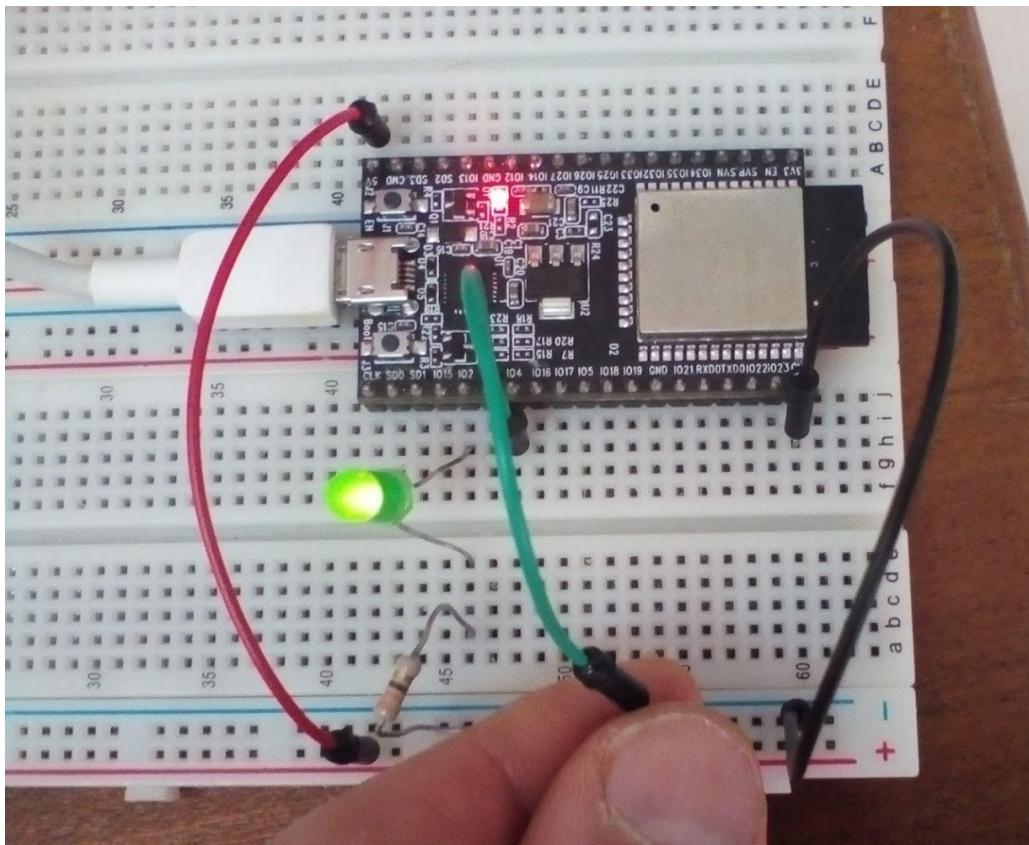
The closer the magnet is to the sensor, the greater the absolute values are.



## Exemplo 2: Efeito Capacitivo (Touch Sensor)

```
*****  
* ESP32 Touch Test and LED Ctrl  
* Touch pin ==> Touch0 is T0 which is on GPIO 4 (D4).  
* LED pin  ==> D2  
*  
* MJRoBot.org 6Sept17  
*****  
  
#define TOUTCH_PIN T0 // ESP32 Pin D4  
#define LED_PIN 2  
int touch_value = 100;  
  
void setup()  
{  
    Serial.begin(115200);  
    delay(1000); // give me time to bring up serial monitor  
    Serial.println("ESP32 Touch Test");  
    pinMode(LED_PIN, OUTPUT);  
    digitalWrite (LED_PIN, LOW);  
}  
  
void loop()  
{  
    touch_value = touchRead(TOUTCH_PIN);  
    Serial.println(touch_value); // get value using T0  
    if (touch_value < 50)  
    {  
        digitalWrite (LED_PIN, HIGH);  
    }  
    else  
    {  
        digitalWrite (LED_PIN, LOW);  
    }  
    delay(1000);  
}
```

[touch.ino Mjrovai](#)



```
/dev/ttyUSB0
Enviar
7
7
56
54
53
53
53
54
26
37
9
7
6
6
55
 Avanço automático de linha  Mostrar marca de tempo
Nova linha 115200 baud Limpar saída
```

Em vazio: >50 e led desligado

Em contacto: <10 e led ligado

## Exemplo 3: Entrada Analógica

GPIO ADC Channel

- GPIO 0 ==> ADC2\_CH1
- GPIO 2 ==> ADC2\_CH2
- GPIO 4 ==> ADC2\_CH0
- GPIO 12 => ADC2\_CH5
- GPIO 13 => ADC2\_CH4
- GPIO 14 => ADC2\_CH6
- GPIO 15 => ADC2\_CH3
- GPIO 25 => ADC2\_CH8
- GPIO 26 => ADC2\_CH9
- GPIO 27 => ADC2\_CH7
- GPIO 32 => ADC1\_CH4
- GPIO 33 => ADC1\_CH5
- GPIO 34 => ADC1\_CH6
- GPIO 35 => ADC1\_CH7
- GPIO 36 => ADC1\_CH0
- GPIO 37 => ADC1\_CH1
- GPIO 38 => ADC1\_CH2
- GPIO 39 => ADC1\_CH3

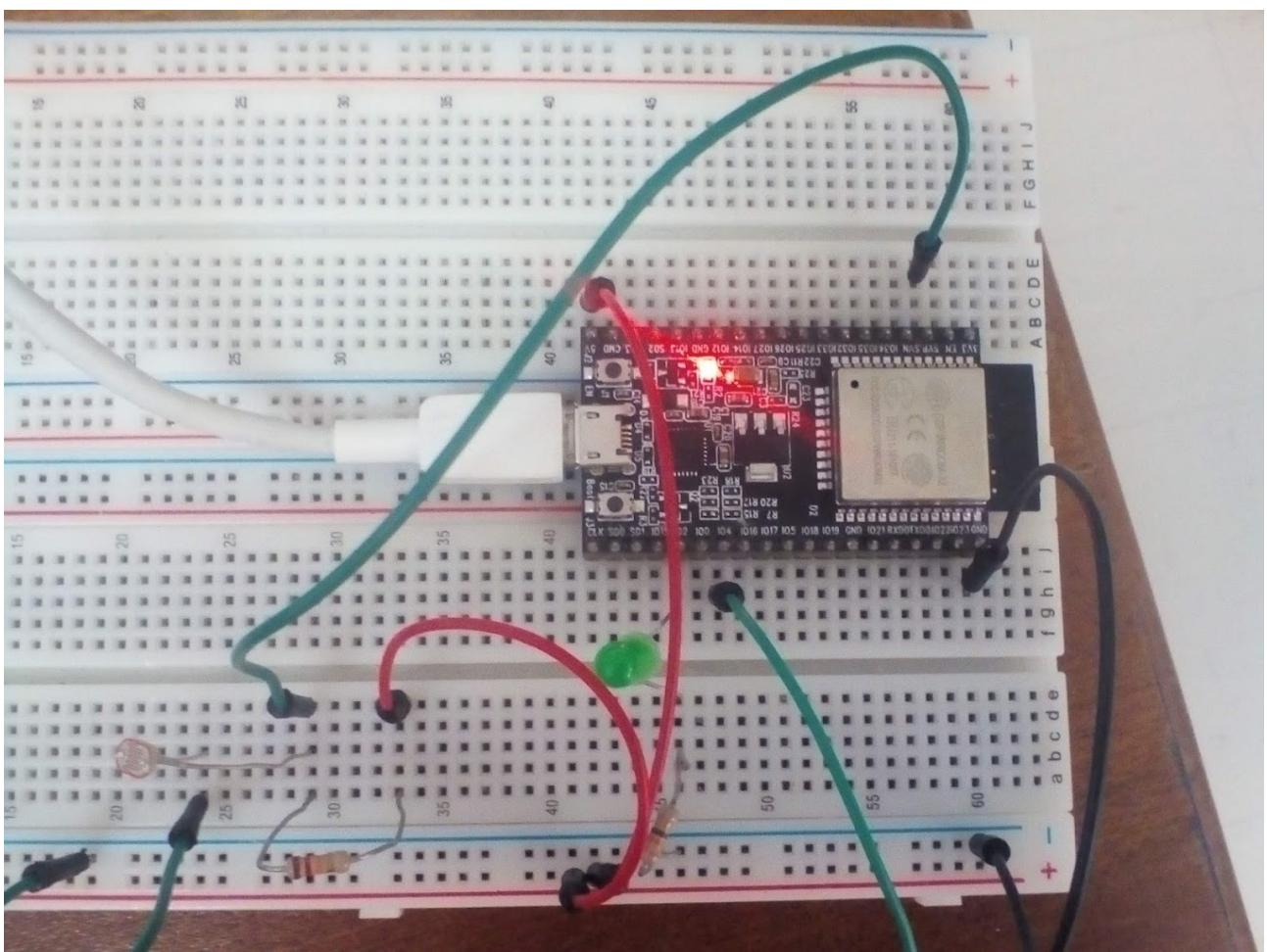
```
*****  
* ESP32 Analog Input Test  
* Analog Input: ADC_1_0 pin ==> GPIO36 (VP).  
*  
* MJRoBot.org 6Sept17  
*****  
//Analog Input  
#define ANALOG_PIN_0 36  
int analog_value = 0;  
  
void setup()  
{  
    Serial.begin(115200);  
    delay(1000); // give me time to bring up serial monitor  
    Serial.println("ESP32 Analog IN Test");  
}  
  
void loop()  
{  
    analog_value = analogRead(ANALOG_PIN_0);  
    Serial.println(analog_value);  
    delay(500);  
}
```

RTD

Com a RTD é possível ir dos 0 (luz total – telemóvel em cima sensor) aos 4095 (escuridão total – dedo no sensor)

```
/dev/ttyUSB0
Enviar
1309
1323
1344
1291
1276
1280
1296
1324
1314
1296
1307
2365
2545
2815
2851

Avanço automático de linha  Mostrar marca de tempo
Nova linha 115200 baud Limpar saída
```



[analog.ino](#)

## Exemplo 4: Entrada Analógica + Saída PWD

Use a mesma montagem anterior.

```
*****  
* ESP32 Analog Input Test  
* Analog Input: ADC_1_0 pin ==> GPIO36 (VP).  
*  
* MJRoBot.org 6Sept17 / Paulo Glavao 28Jun19  
*****/
```

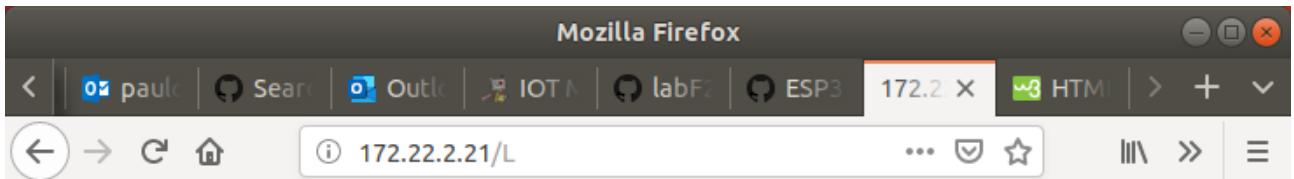
```
//Analog Input  
  
#define ANALOG_PIN_0 36  
  
int analog_value = 0;  
  
  
// PMW LED  
  
#define LED_PIN 2  
  
int freq = 5000;  
int ledChannel = 0;  
int resolution = 8;  
int dutyCycle = 0;  
  
  
void setup()  
{  
    Serial.begin(115200);  
    delay(1000); // give me time to bring up serial monitor  
    Serial.println("ESP32 Analog IN/OUT Test");  
  
  
    ledcSetup(ledChannel, freq, resolution);  
    ledcAttachPin(LED_PIN, ledChannel);  
    ledcWrite(ledChannel, dutyCycle);
```

```
}

void loop()
{
analog_value = analogRead(ANALOG_PIN_0);
Serial.print("LDR: ");
Serial.println(analog_value);
dutyCycle = map(analog_value, 0, 4095, 0, 255);
ledcWrite(ledChannel, dutyCycle);
Serial.print("PWD: ");
Serial.println(dutyCycle);
delay(500);
}
```

[analog\\_PWM.ino](#)

## Exemplo 5: Sending and Receiving Data From a Local Web Page



### Receiving Data from ESP32 webserver on 172.22.2.X

Analog Data (LDR): 1115

Click [here](#) to turn the LED on.  
Click [here](#) to turn the LED off.

```
/* WiFi parameters and credentials */

#include <WiFi.h>
/*const char* ssid    = "YOUR SSID";
const char* password = "YOUR PASSWORD";
*/
const char* ssid    = "EACI";
const char* password = "SMD@EACI";

WiFiServer server(80);

/* DHT Temperature and Humidity Sensor */
/*#include "DHT.h"
#define DHTPIN 23
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
float localHum = 0;
float localTemp = 0;
*/

/* LED */
#define LED_PIN 2

//Analog Input
#define ANALOG_PIN_0 36
int analog_value = 0;

void setup()
```

```

{
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  delay(10);

  connectWiFi();

// dht.begin();
}

int value = 0;

void loop()
{
  analog_value = analogRead(ANALOG_PIN_0);
  //getDHT();
  WiFiLocalWebPageCtrl();
}

/******************
* Get indoor Temp/Hum data
******************/

/*void getDHT()
{
  float tempIni = localTemp;
  float humIni = localHum;
  localTemp = dht.readTemperature();
  localHum = dht.readHumidity();
  if (isnan(localHum) || isnan(localTemp)) // Check if any reads failed and exit early (to try again).
  {
    localTemp = tempIni;
    localHum = humIni;
    return;
  }
}

*/
/******************
* Send and receive data from Local Page
******************/

void WiFiLocalWebPageCtrl(void)
{
  WiFiClient client = server.available(); // listen for incoming clients
  //client = server.available();
  if (client) { // if you get a client,
    Serial.println("New Client."); // print a message out the serial port
    String currentLine = ""; // make a String to hold incoming data from the client
    while (client.connected()) { // loop while the client's connected
      if (client.available()) { // if there's bytes to read from the client,
        char c = client.read(); // read a byte, then
        Serial.write(c); // print it out the serial monitor
        if (c == '\n') { // if the byte is a newline character

```

```

// if the current line is blank, you got two newline characters in a row.
// that's the end of the client HTTP request, so send a response:
if (currentLine.length() == 0) {
    // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
    // and a content-type so the client knows what's coming, then a blank line:
    client.println("HTTP/1.1 200 OK");
    client.println("Content-type:text/html");
    client.println();

    // the content of the HTTP response follows the header:
    //WiFiLocalWebPageCtrl();
/*
    client.print("Temperature now is: ");
    client.print(localTemp);
    client.print(" °C<br>");
    client.print("Humidity now is:   ");
    client.print(localHum);
    client.print(" % <br>");
    client.print("<br>");*/
}

client.print("<h1>Receiving Data from ESP32 webserver on 172.22.2.X</h1><hr><br>");

client.print("Analog Data (LDR):   ");
client.print(analog_value);
client.print("<br>");
client.print("<br>");

client.print("Click <a href=\"/H\">here</a> to turn the LED on.<br>");
client.print("Click <a href=\"/L\">here</a> to turn the LED off.<br>");

// The HTTP response ends with another blank line:
client.println();
// break out of the while loop:
break;
} else { // if you got a newline, then clear currentLine:
    currentLine = "";
}
} else if (c != '\r') { // if you got anything else but a carriage return character,
    currentLine += c; // add it to the end of the currentLine
}

// Check to see if the client request was "GET /H" or "GET /L":
if (currentLine.endsWith("GET /H")) {
    digitalWrite(LED_PIN, HIGH); // GET /H turns the LED on
}
if (currentLine.endsWith("GET /L")) {
    digitalWrite(LED_PIN, LOW); // GET /L turns the LED off
}
}

// close the connection:
client.stop();

```

```

        Serial.println("Client Disconnected.");
    }
}

*****
* Connecting to a WiFi network
*****
void connectWiFi(void)
{
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());

    server.begin();
}

```

Original:

[ESP32 WiFi Server Sending Receiving Data.ino](#)

Transformado para a rede da escola: [webserver\\_html.ino](#)

ver endereço web dado pelo router através da porta série:

```

/dev/ttyUSB0
40078000,len:8424
ho 0 tail 12 room 4
load:0x40080400,len:5868
entry 0x4008069c

Connecting to EACI
..
WiFi connected.
IP address:
172.22.2.21

```

Avanço automático de linha  Mostrar marca de tempo       Nova linha      115200 baud      Limpar saída

```

const char* ssid    = "EACI";
const char* password = "SMD@EACI";

```

## Exemplo 6: ESP32 as HTTP Server using WiFi Access Point (AP) mode

As the heading suggests, this example demonstrates how to turn the ESP32 into an access point (AP), and serve up web pages to any connected client. To start with, plug your ESP32 into your computer and Try the sketch out; and then we will dissect it in some detail.

```
#include <WiFi.h>
#include <WebServer.h>

/* Put your SSID & Password */
const char* ssid = "ESP32"; // Enter SSID here
const char* password = "12345678"; //Enter Password here

/* Put IP Address details */
IPAddress local_ip(192,168,1,1);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);

WebServer server(80);

uint8_t LED1pin = 2;
bool LED1status = LOW;

uint8_t LED2pin = 5;
bool LED2status = LOW;

//Analog Input
#define ANALOG_PIN_0 36
int analog_value = 0;

void setup() {
    Serial.begin(115200);
    pinMode(LED1pin, OUTPUT);
    pinMode(LED2pin, OUTPUT);

    WiFi.softAP(ssid, password);
    WiFi.softAPConfig(local_ip, gateway, subnet);
    delay(100);

    server.on("/", handle_OnConnect);
    server.on("/led1on", handle_led1on);
    server.on("/led1off", handle_led1off);
    server.on("/led2on", handle_led2on);
    server.on("/led2off", handle_led2off);
    server.onNotFound(handle_NotFound);

    server.begin();
    Serial.println("HTTP server started");
}

void loop() {
    server.handleClient();
    if(LED1status)
        {digitalWrite(LED1pin, HIGH);}
    else
        {digitalWrite(LED1pin, LOW);}

    if(LED2status)
```

```

    {digitalWrite(LED2pin, HIGH);}
else
{digitalWrite(LED2pin, LOW);}
analog_value = analogRead(ANALOG_PIN_0);
}

void handle_OnConnect() {
    LED1status = LOW;
    LED2status = LOW;
    Serial.println("GPIO4 Status: OFF | GPIO5 Status: OFF");
    server.send(200, "text/html", SendHTML(LED1status,LED2status));
}

void handle_led1on() {
    LED1status = HIGH;
    Serial.println("GPIO4 Status: ON");
    server.send(200, "text/html", SendHTML(true,LED2status));
}

void handle_led1off() {
    LED1status = LOW;
    Serial.println("GPIO4 Status: OFF");
    server.send(200, "text/html", SendHTML(false,LED2status));
}

void handle_led2on() {
    LED2status = HIGH;
    Serial.println("GPIO5 Status: ON");
    server.send(200, "text/html", SendHTML(LED1status,true));
}

void handle_led2off() {
    LED2status = LOW;
    Serial.println("GPIO5 Status: OFF");
    server.send(200, "text/html", SendHTML(LED1status,false));
}

void handle_NotFound(){
    server.send(404, "text/plain", "Not found");
}

String SendHTML(uint8_t led1stat,uint8_t led2stat){
    String ptr = "<!DOCTYPE html> <html>\n";
    ptr += "<meta charset=UTF-8 />";
    ptr += "<meta http-equiv=""refresh"" content=""2"">";
    ptr += "<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
    ptr += "<title>LED Control</title>\n";
    ptr += "<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
    ptr += "body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;margin-bottom: 50px;}\n";
    ptr += ".button {display: block;width: 80px;background-color: #3498db;border: none;color: white;padding: 13px 30px;text-decoration: none;font-size: 25px;margin: 0px auto 35px;cursor: pointer;border-radius: 4px;}\n";
    ptr += ".button-on {background-color: #3498db;}\n";
    ptr += ".button-on:active {background-color: #2980b9;}\n";
    ptr += ".button-off {background-color: #34495e;}\n";
    ptr += ".button-off:active {background-color: #2c3e50;}\n";
    ptr += "p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
    ptr += "</style>\n";
    ptr += "</head>\n";
    ptr += "<body>\n";
    ptr += "<h1>ESP32 Web Server</h1>\n";
}

```

```

ptr += "<h3>Using Access Point(AP) Mode</h3>\n";

if(led1stat)
{ptr += "<p>LED1 Status: ON</p><a class=\"button button-off\" href=\"/led1off\">OFF</a>\n";
}
else
{ptr += "<p>LED1 Status: OFF</p><a class=\"button button-on\" href=\"/led1on\">ON</a>\n";
}

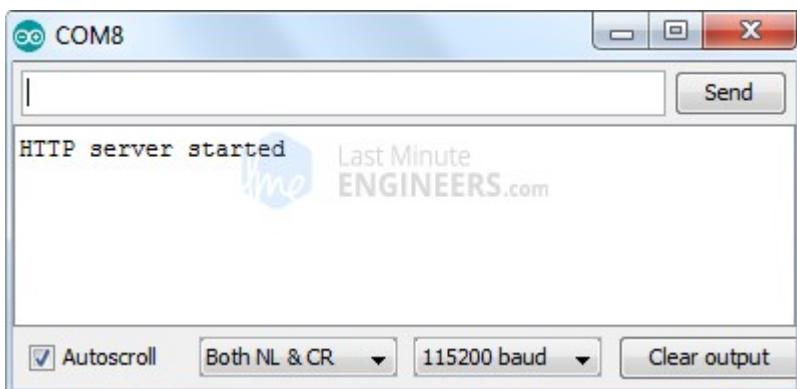
if(led2stat)
{ptr += "<p>LED2 Status: ON</p><a class=\"button button-off\" href=\"/led2off\">OFF</a>\n";
}
else
{ptr += "<p>LED2 Status: OFF</p><a class=\"button button-on\" href=\"/led2on\">ON</a>\n";
}

ptr += "<p>Leitura Analógica: ";
ptr +=analogRead(ANALOG_PIN_0);
ptr += "</p>";
ptr += "</body>\n";
ptr += "</html>\n";
return ptr;
}

```

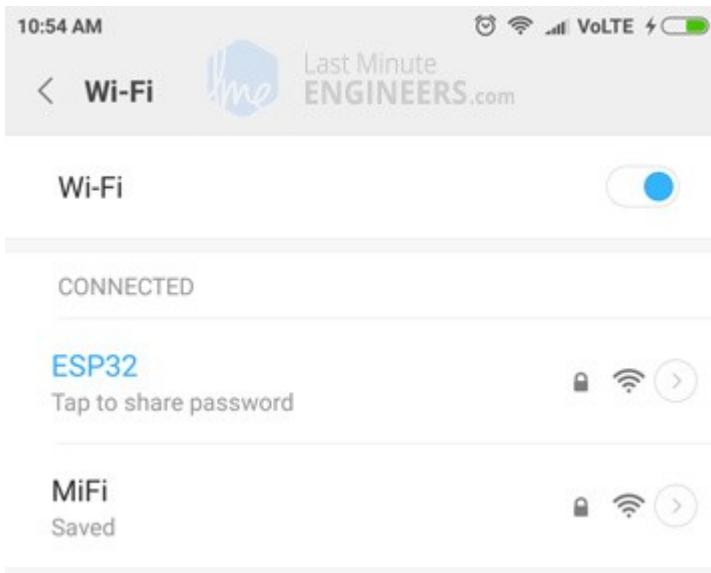
## Accessing the Web Server in AP mode

After uploading the sketch, open the Serial Monitor at a baud rate of 115200. And press the RESET button on ESP32. If everything is OK, it will show **HTTP server started** message.



Next, find any device that you can connect to a WiFi network – phone, laptop, etc. And look for a network called **ESP32**. Join the network with password **123456789**.

[simple web server](#)



After connecting to your ESP32 AP network, load up a browser and point it to 192.168.1.1. The ESP32 should serve up a web page showing current status of LEDs and two buttons to control them. If you take a look at the serial monitor at the same time, you can see status of ESP32's GPIO pins.



## ESP32 Web Server

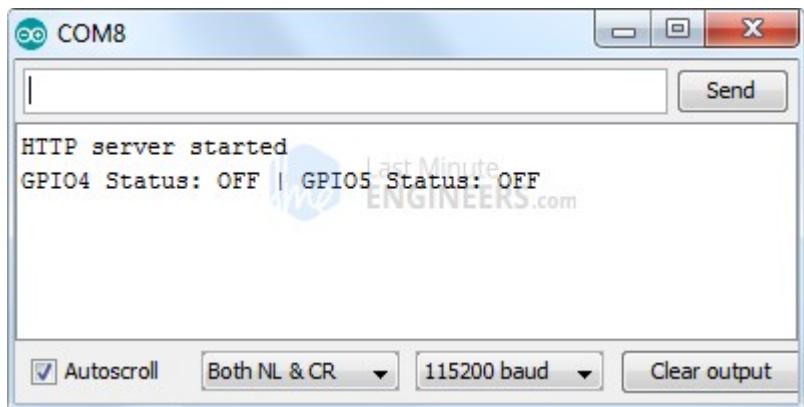
### Using Access Point(AP) Mode

LED1 Status: OFF

ON

LED2 Status: OFF

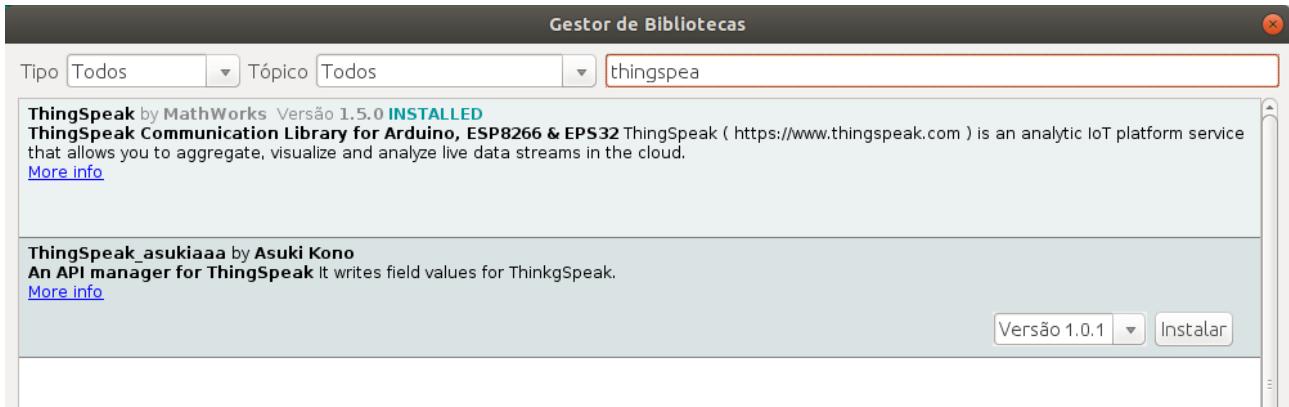
ON



## ThingSpeak

Análise de dados em web. Possibilidade de exportar dados para Matlab.

Instalar antes a biblioteca ThinkSpeak.

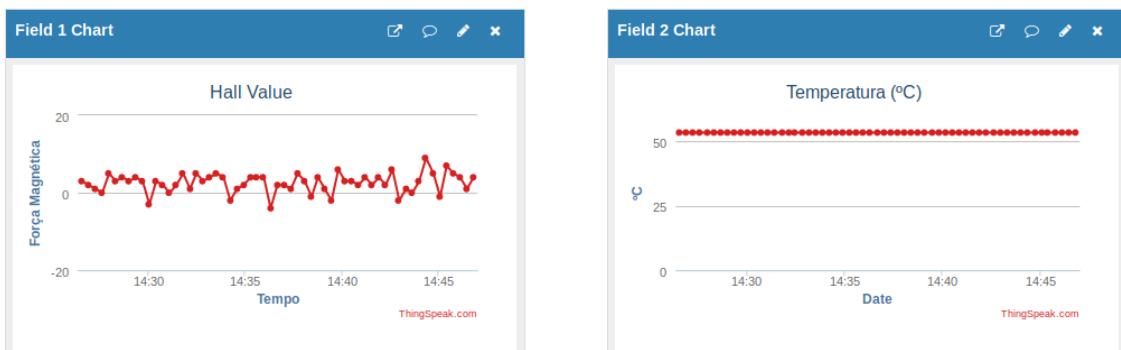


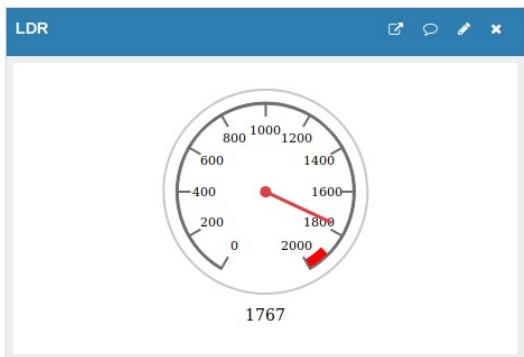
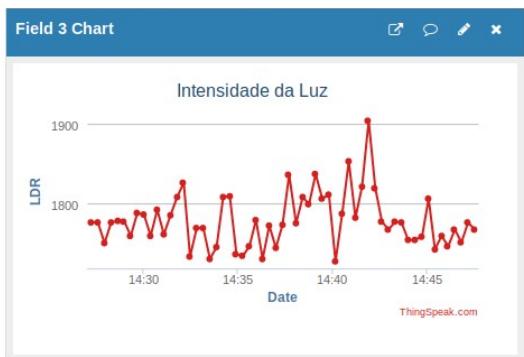
### Channel Stats

Created: [about 3 hours ago](#)

Last entry: [less than a minute ago](#)

Entries: 261





```

#include "ThingSpeak.h"
#include <WiFi.h>

#ifndef __cplusplus
extern "C" {
#endif
uint8_t temprature_sens_read();
#ifndef __cplusplus
}
#endif
uint8_t temprature_sens_read();

#define ANALOG_PIN_0 36
int analog_value = 0;

char ssid[] = "EACI"; // your network SSID (name)
char pass[] = "SMD@EACI"; // your network password
int keyIndex = 0; // your network key Index number (needed only for WEP)
WiFiClient client;

unsigned long myChannelNumber = 815527; //update
const char * myWriteAPIKey = "2IS0VFSE7U31WKZ6"; //update

void setup()
{
Serial.begin(115200); //Initialize serial

delay(10);
WiFi.mode(WIFI_STA);
ThingSpeak.begin(client); // Initialize ThingSpeak
}

void loop() {

int h = 0;
float t =0;

h = hallRead();
t = ((temprature_sens_read()-32)/1.8);           //changing temperature parameter to celsius

analog_value = analogRead(ANALOG_PIN_0);
Serial.println(analog_value);

// Connect or reconnect to WiFi

```

```

if(WiFi.status() != WL_CONNECTED){
Serial.print("Attempting to connect to SSID: ");
//Serial.println(SECRET_SSID);
while(WiFi.status() != WL_CONNECTED){
WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open or WEP
network
Serial.print(".");
delay(5000);
}
Serial.println("\nConnected.");
}

// Write to ThingSpeak. There are up to 8 fields in a channel, allowing you to store up to 8 different
// pieces of information in a channel.
ThingSpeak.setField(1, h);
ThingSpeak.setField(2, t);
ThingSpeak.setField(3, analog_value);

// write to the ThingSpeak channel
int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
if(x == 200){
Serial.println("Channel update successful.");
}
else{
Serial.println("Problem updating channel. HTTP error code " + String(x));
}

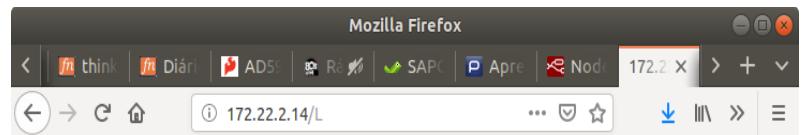
delay(20000); // Wait 20 seconds to update the channel again
}

```

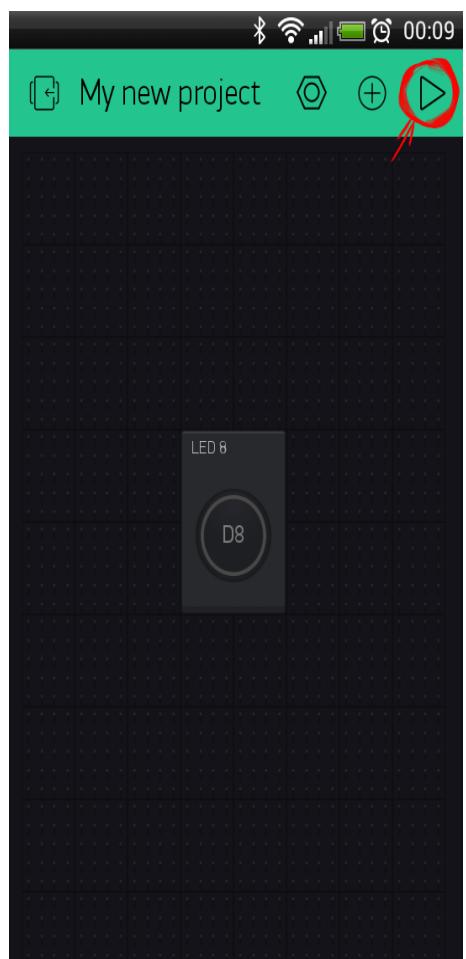
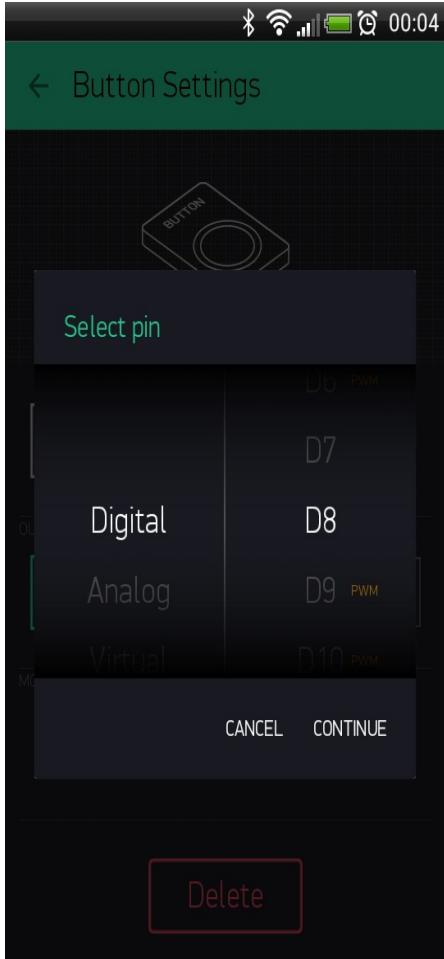
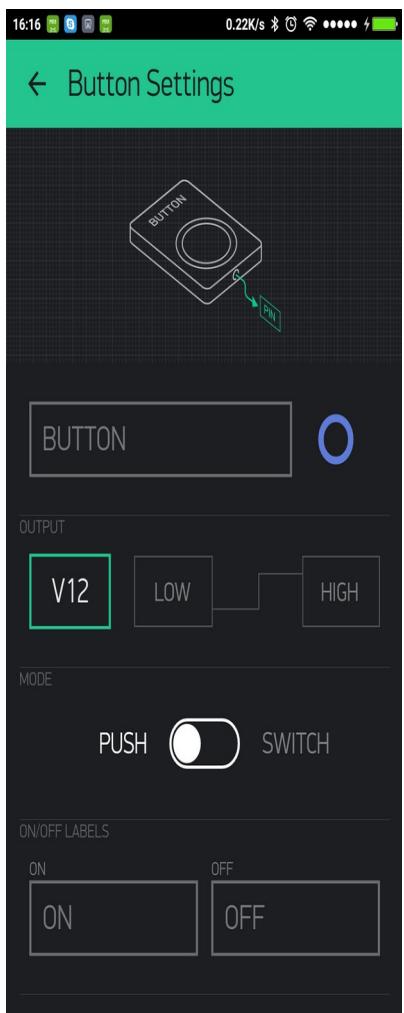
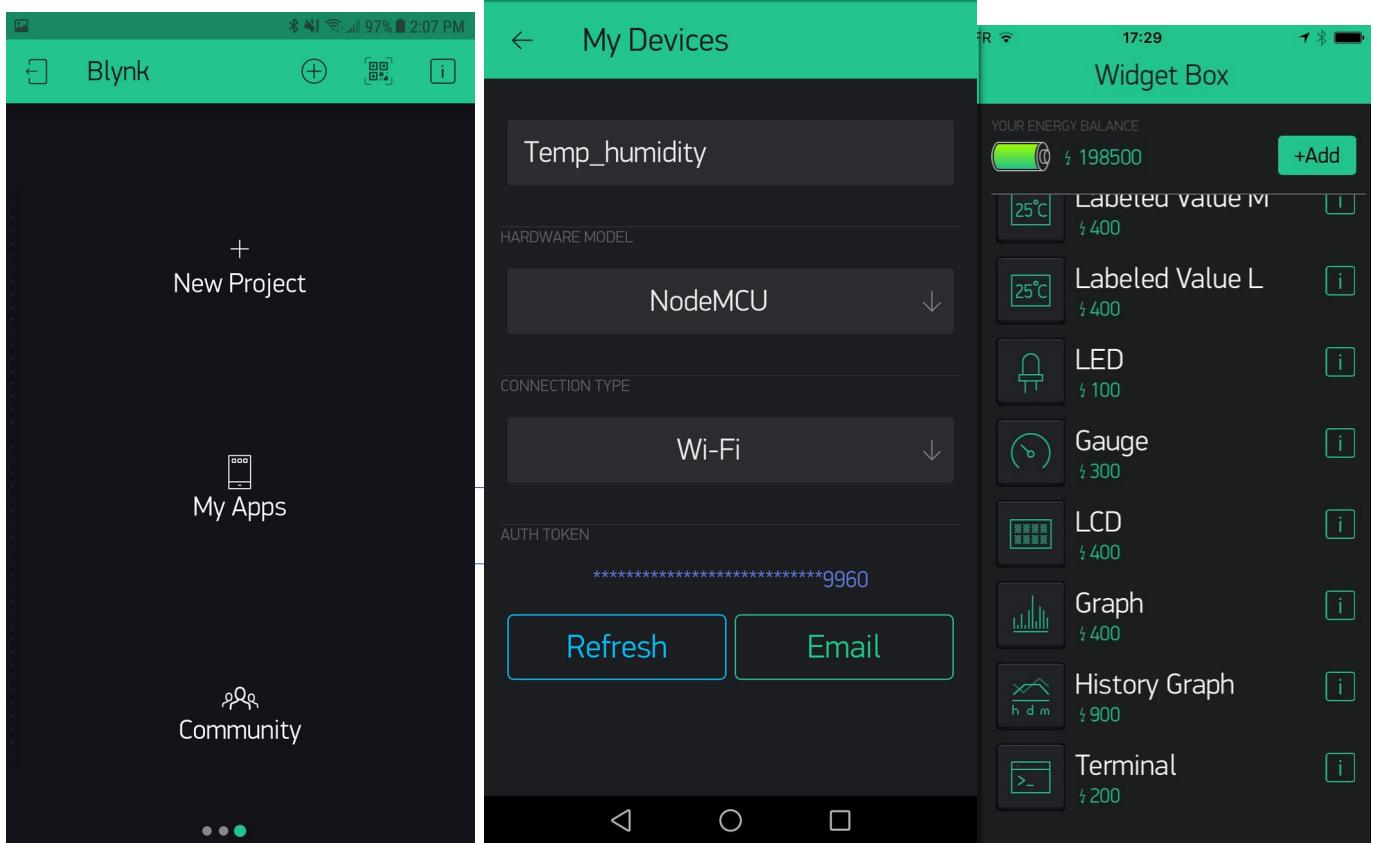
The screenshot shows a terminal window titled '/dev/ttyUSB0'. The window displays a series of messages indicating successful channel updates. Each message starts with 'Channel update successful.' followed by a numerical value (1634, 1654, 1643, 1647, 1627, 1675). The terminal interface includes a scroll bar on the right, a text input field at the bottom left, and buttons for 'Enviar' (Send), 'Nova linha' (New Line), '115200 baud', and 'Limpar saída' (Clear Output).

[thingSpeak.ino](#)

## Exemplo 7: ESP ligado à aplicação de Telemóvel Blynk com data e hora dadas pela internet



## Como configurar Blynk e programar (resumo) - <https://docs.blynk.cc/>



O blynk tem a vantagem (além de ser opensource) de se **pode ser instalado localmente** por exemplo num Raspberry PI, o que permite a sua utilização de forma mais rápida, segura e sem limitações.

Os dados do gráfico Superchart podem ser exportados para .csv.

## Programa ESP32

```
/* WiFi parameters and credentials */
#include <WiFi.h>
#define BLYNK_PRINT Serial
#include <BlynkSimpleEsp32.h> // Blynk-ESP32
#include <WiFiClient.h> // Wi-Fi client
/*const char* ssid    = "YOUR SSID";
const char* password = "YOUR PASSWORD";
*/
#include <NTPClient.h>
#include <WiFiUdp.h>
// https://github.com/taranais/NTPClient instalar esta biblioteca
// fazendo desta forma https://www.robocontrol.net/tutoriais/adicionando-bibliotecas-na-ide-arduino.html

const char* ssid    = "EACI";
const char* password = "SMD@EACI";

//----- Token de Autenticação -----
char auth[] = " vSW44z6xNbpmkVa6y2jO7o58IB2dxzIi"; // Coloque aqui o código de Token

WiFiServer server(80);

// Define NTP Client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

/* DHT Temperature and Humidity Sensor */
/*#include "DHT.h"
#define DHTPIN 23
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
float localHum = 0;
float localTemp = 0;
*/

/* LED */
#define LED_PIN 2

//Analog Input
#define ANALOG_PIN_0 36
int analog_value = 0;

// Variables to save date and time
```

```

String formattedDate;
String dayStamp;
String timeStamp;

BlynkTimer timer;
void sendSensor()
{
    analog_value = analogRead(ANALOG_PIN_0); // Armazena os valores de leitura do sensor de luz

    Blynk.virtualWrite(V1, LED_PIN); //No widget botão colocar no output GP2
    Blynk.virtualWrite(V2, analog_value); // Valor de LDR porta virtual V2
    Blynk.virtualWrite(V3, timeStamp); // Colocação da hora na 1ª linha do widget LCD
    Blynk.virtualWrite(V4, dayStamp); // Colocação do dia na 2ª linha do widget LCD
}

void setup()
{
    Serial.begin(115200);
    pinMode(LED_PIN, OUTPUT);
    delay(10);

    connectWiFi();

    Blynk.begin(auth, ssid, password); // TOKEN+REDE+SENHA
    timer.setInterval(1000L, sendSensor);

    // dht.begin();

    // Initialize a NTPClient to get time
    timeClient.begin();
    // Set offset time in seconds to adjust for your timezone, for example:
    // GMT +1 = 3600
    // GMT +8 = 28800
    // GMT -1 = -3600
    // GMT 0 = 0
    timeClient.setTimeOffset(3600);
}

int value = 0;

void data_hora() {
    while(!timeClient.update()) {
        timeClient.forceUpdate();
    }
    // The formattedDate comes with the following format:
    // 2018-05-28T16:00:13Z
    // We need to extract date and time
    formattedDate = timeClient.getFormattedDate();
    Serial.println(formattedDate);
}

```

```

// Extract date
int splitT = formattedDate.indexOf("T");
dayStamp = formattedDate.substring(0, splitT);
Serial.print("DATE: ");
Serial.println(dayStamp);
// Extract time
timeStamp = formattedDate.substring(splitT+1, formattedDate.length()-1);
Serial.print("HOUR: ");
Serial.println(timeStamp);
delay(1000);
}

void loop()
{
analog_value = analogRead(ANALOG_PIN_0);
//getDHT();
WiFiLocalWebPageCtrl();

timer.run();
Blynk.run();
data_hora();

}

/******************
* Get indoor Temp/Hum data
******************/

/*void getDHT()
{
float tempIni = localTemp;
float humIni = localHum;
localTemp = dht.readTemperature();
localHum = dht.readHumidity();
if (isnan(localHum) || isnan(localTemp)) // Check if any reads failed and exit early (to try again).
{
    localTemp = tempIni;
    localHum = humIni;
    return;
}
}

*/
/******************
* Send and receive data from Local Page
******************/

void WiFiLocalWebPageCtrl(void)
{
WiFiClient client = server.available(); // listen for incoming clients
//client = server.available();
if (client) { // if you get a client,
    Serial.println("New Client."); // print a message out the serial port
    String currentLine = ""; // make a String to hold incoming data from the client
    while (client.connected()) { // loop while the client's connected
        if (client.available()) { // if there's bytes to read from the client,
            char c = client.read(); // read a byte, then
            Serial.write(c); // print it out the serial monitor
            if (c == '\n') { // if the byte is a newline character

```

```

// if the current line is blank, you got two newline characters in a row.
// that's the end of the client HTTP request, so send a response:
if (currentLine.length() == 0) {
    // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
    // and a content-type so the client knows what's coming, then a blank line:
    client.println("HTTP/1.1 200 OK");
    client.println("Content-type:text/html");
    client.println();

    // the content of the HTTP response follows the header:
    //WiFiLocalWebPageCtrl();
/*
    client.print("Temperature now is: ");
    client.print(localTemp);
    client.print(" °C<br>");
    client.print("Humidity now is:   ");
    client.print(localHum);
    client.print(" % <br>");
    client.print("<br>");*/
}

client.print("<h1>Receiving Data from ESP32 webserver on 172.22.2.X</h1><hr><br>");

client.print("Analog Data (LDR):   ");
client.print(analog_value);
client.print("<br>");
client.print("<br>");

client.print("Click <a href=\"/H\">here</a> to turn the LED on.<br>");
client.print("Click <a href=\"/L\">here</a> to turn the LED off.<br>");
client.print("<br>");
client.print("Hora:   ");
client.print(timeStamp);
client.print("<br>");
client.print("Dia:   ");
client.print(dayStamp);
client.print("<br>");

// The HTTP response ends with another blank line:
client.println();
// break out of the while loop:
break;
} else { // if you got a newline, then clear currentLine:
    currentLine = "";
}
} else if (c != '\r') { // if you got anything else but a carriage return character,
    currentLine += c; // add it to the end of the currentLine
}

// Check to see if the client request was "GET /H" or "GET /L":
if (currentLine.endsWith("GET /H")) {
    digitalWrite(LED_PIN, HIGH); // GET /H turns the LED on
}
if (currentLine.endsWith("GET /L")) {
    digitalWrite(LED_PIN, LOW); // GET /L turns the LED off
}
}
}

```

```

// close the connection:
client.stop();
Serial.println("Client Disconnected.");
}

*******/

* Connecting to a WiFi network
****

void connectWiFi(void)
{
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());

    server.begin();
}

```

[blynk\\_time\\_date.ino](#)

## Exemplo 8: Como ligar esp32 à rede eduroam

```
/*-----*/
/*|WORKING EXAMPLE FOR HASSELT UNIVERSITY          */
/*|Only connection to eduroam network            */
/*|https://www.uhasselt.be/eduroam               */
/*|SKETCH PROVIDED BY: Thijs Vandenvyrt        */
/*-----*/
#include <WiFi.h> //Wifi library
#include "esp_wpa2.h" //wpa2 library for connections to Enterprise networks
#define EAP_IDENTITY "paulo.galvao@estsetubal.ips.pt" //for student
//#define EAP_IDENTITY "id@uhasselt.be" //for researchers, scientists, teachers
#define EAP_PASSWORD "asua"
const char* ssid = "eduroam"; // Eduroam SSID
void setup() {
    byte error = 0;
    Serial.begin(115200);
    delay(10);
    Serial.println("Connecting to: ");
    Serial.println(ssid);
    WiFi.disconnect(true); //disconnect from wifi to set new wifi connection
    error += esp_wifi_sta_wpa2_ent_set_identity((uint8_t *)EAP_IDENTITY, strlen(EAP_IDENTITY));
    error += esp_wifi_sta_wpa2_ent_set_username((uint8_t *)EAP_IDENTITY, strlen(EAP_IDENTITY));
    error += esp_wifi_sta_wpa2_ent_set_password((uint8_t *)EAP_PASSWORD, strlen(EAP_PASSWORD));
    //Following times, it ran fine with just this line (connects very fast).
    if (error != 0) {
        Serial.println("Error setting WPA properties.");
    }
    WiFi.enableSTA(true);

    esp_wpa2_config_t config = WPA2_CONFIG_INIT_DEFAULT();
    if (esp_wifi_sta_wpa2_ent_enable(&config) != ESP_OK) {
        Serial.println("WPA2 Settings Not OK");
    }
}

WiFi.begin(ssid); //connect to Eduroam function
WiFi.setHostname("RandomHostname"); //set Hostname for your device - not neccesary
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address set: ");
Serial.println(WiFi.localIP()); //print LAN IP
}
void loop(){
if (WiFi.status() != WL_CONNECTED) {
    WiFi.begin(ssid);
}
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
}
```

[Hasselt\\_university\\_upload\\_second.ino](#)

## Exemplo 9: Enviar texto e comandos para LED's através de formulários HTML (como estação STA)

<https://github.com/labF212/ESP32/blob/master/formulario.ino>



Pré-requisitos:

To build the asynchronous web server, you need to install these libraries.

- **ESP32:** install the [ESPAsyncWebServer](#) and the [AsyncTCP](#) libraries.
- **ESP8266:** install the [ESPAsyncWebServer](#) and the [ESPAsyncTCP](#) libraries.

These libraries aren't available to install through the Arduino Library Manager, so you need to copy the library files to the Arduino Installation folder. Alternatively, in your Arduino IDE, you can go to **Sketch > Include Library > Add .zip Library** and select the libraries you've just downloaded.

```

#include <Arduino.h>
#ifndef ESP32
  #include <WiFi.h>
  #include <AsyncTCP.h>
#else
  #include <ESP8266WiFi.h>
  #include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

AsyncWebServer server(80);

// REPLACE WITH YOUR NETWORK CREDENTIALS
const char* ssid    = "EACI";
const char* password = "SMD@EACI";

const char* PARAM_INPUT_1 = "input1";
const char* PARAM_INPUT_2 = "input2";
const char* PARAM_INPUT_3 = "input3";
const char* PARAM_INPUT_4 = "input4";
const char* PARAM_INPUT_5 = "input5";
const char* PARAM_INPUT_6 = "input6";
const char* PARAM_INPUT_7 = "input7";
const char* PARAM_INPUT_8 = "input8";
const char* PARAM_INPUT_9 = "input9";
const char* PARAM_INPUT_10 = "input10";

String palavra;
char letra;

/* LED */
#define LED_PIN1 15
#define LED_PIN2 16
#define LED_PIN3 17
#define LED_PIN4 18
#define LED_PIN5 19

// HTML web page to handle 3 input fields (input1, input2, input3)
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
  <head>
    <meta charset="UTF-8"> <!--portuguese chars-->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Comando de Robôs por ESP32</title>
  </head>

  <body>

    <h1>Palavra a enviar ao Robô</h1>

```

```
<form action="/get">
  Escreva a palavra (apenas com 10 caracteres): <input type="text" name="input1">
  <input type="submit" value="Enviar">
</form><br>

<div style="background-color:lightblue"><center>
<h1>Jogo do Galo</h1>
<table>
  <!--Linha 1-->
  <tr><td>
    <form action="/get">
      <!--input4:<input type="submit" value="Submit"-->
      <!--input4: <input type="text" name="input4"-->
      <input type="submit" name="input2" value="Posição 1" />
    </form></td>
    <td>
      </&emsp;> <!--4 spaces-->
    </td>
    <td>
      <form action="/get">
        <input type="submit" name="input3" value="Posição 2" />
      </td>
      </form>
      <td>
        </&emsp;> <!--4 spaces-->
      </td>
      <td>
        <form action="/get">
          <input type="submit" name="input4" value="Posição 3" />
        </td>
        </form>
        <!--Linha 2-->
      </tr>
    </table>
    <table>
      <tr><td>
        <form action="/get">
          <input type="submit" name="input5" value="Posição 4" />
        </form></td>
        <td>
          </&emsp;> <!--4 spaces-->
        </td>
        <td>
          <form action="/get">
            <input type="submit" name="input6" value="Posição 5" />
          </td>
          <td>
            </&emsp;> <!--4 spaces-->
          </td>
          <td>
            <form action="/get">
              <input type="submit" name="input7" value="Posição 6" />
            </td>
          </td>
        </table>
      </div>
```

```

</td>
</tr></table>
<!--Linha 3-->
</tr>
</table>
<tr><td>
<form action="/get">
<input type="submit" name="input8" value="Posição 7" />
</form></td>
<td>
</&emsp> <!--4 spaces-->
</td>
<td>
<form action="/get">
<input type="submit" name="input9" value="Posição 8" />
</td>
<td>
</&emsp> <!--4 spaces-->
</td>
<td>
<form action="/get">
<input type="submit" name="input10" value="Posição 9" />
</td>
</tr></table>

</br>
</div>
<p id="demo">Hi.</p>
<script>
document.getElementById("demo").innerHTML="Programado por: Fernando Costeira e Paulo
Galvão";
</script>
<p><a href="https://github.com/labF212/ESP32">Obter código</a></p>

</body></html>)rawliteral';

void notFound(AsyncWebServerRequest *request) {
    request->send(404, "text/plain", "Not found");
}

void setup() {
    Serial.begin(115200);

    pinMode(LED_PIN1, OUTPUT);
    pinMode(LED_PIN2, OUTPUT);
    pinMode(LED_PIN3, OUTPUT);
    pinMode(LED_PIN4, OUTPUT);
    pinMode(LED_PIN5, OUTPUT);
}

```

```

delay(10);

// connectWiFi();

WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
if (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("WiFi Failed!");
    return;
}
Serial.println();
Serial.print("IP Address: ");
Serial.println(WiFi.localIP());

// Send web page with input fields to client
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html);
});

// Jogo da Palavra
// Send a GET request to <ESP_IP>/get?input1=<inputMessage>
server.on("/get", HTTP_GET, [] (AsyncWebServerRequest *request) {
    String inputMessage;
    String inputParam;
    // GET input1 value on <ESP_IP>/get?input1=<inputMessage>
    //Recebe a Palavra
    if (request->hasParam(PARAM_INPUT_1)) {
        inputMessage = request->getParam(PARAM_INPUT_1)->value();
        //inputParam = PARAM_INPUT_1;
        Serial.println("Letra");
        //Serial.println(inputParam);

        inputMessage.toLowerCase();
        Serial.println(inputMessage);

        //palavra+ = inputParam;
        for (int i = 0; i < 10; i++) {
        //Serial.println(palavra[i]);

        char letra = inputMessage[i];
        Serial.println(letra);

        if(letra == 'a'){
            digitalWrite(LED_PIN1, LOW);
            digitalWrite(LED_PIN2, HIGH);
            digitalWrite(LED_PIN3, HIGH);
            digitalWrite(LED_PIN4, HIGH);
            digitalWrite(LED_PIN5, HIGH);
        }
    }
}

```

```
delay(400);
}

if(letra == 'b'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'c'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'd'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'e'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'f'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'g'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, LOW);
```

```
digitalWrite(LED_PIN4, HIGH);
digitalWrite(LED_PIN5, HIGH);
delay(400);
}

if(letra == 'h'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'i'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'j'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'k'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'l'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'm'){
    digitalWrite(LED_PIN1, LOW);
```

```
digitalWrite(LED_PIN2, HIGH);
digitalWrite(LED_PIN3, LOW);
digitalWrite(LED_PIN4, LOW);
digitalWrite(LED_PIN5, HIGH);
delay(400);
}

if(letra == 'n'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'o'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'p'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 'q'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 'r'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}
```

```
if(letra == 's'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 't'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2,HIGH);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 'u'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 'v'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 'x'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 'y'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}
```

```

    }

if(letra == 'z'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

}

// JOGO DO GALO

// GET input2 value on <ESP_IP>/get?input2=<inputMessage>
// Recebe o numero da posição 1 da jogada do jogo do galo
else if (request->hasParam(PARAM_INPUT_2)) {
    digitalWrite(LED_PIN1, LOW);          // GET turns the LED off
    digitalWrite(LED_PIN2, HIGH);         // GET turns the LED off
    digitalWrite(LED_PIN3, HIGH);         // GET turns the LED on
    digitalWrite(LED_PIN4, HIGH);         // GET turns the LED on
    delay(2000);
}

// GET input3 value on <ESP_IP>/get?input3=<inputMessage>
// Recebe o numero da posição 2 da jogada do jogo do galo
else if (request->hasParam(PARAM_INPUT_3)) {
    digitalWrite(LED_PIN1, HIGH);         // GET turns the LED off
    digitalWrite(LED_PIN2, LOW);          // GET turns the LED off
    digitalWrite(LED_PIN3, HIGH);          // GET turns the LED on
    digitalWrite(LED_PIN4, HIGH);          // GET turns the LED on
    delay(2000);
}

// Recebe o numero da posição 3 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_4)) {
    digitalWrite(LED_PIN1, LOW);          // GET turns the LED off
    digitalWrite(LED_PIN2, LOW);          // GET turns the LED off
    digitalWrite(LED_PIN3, HIGH);          // GET turns the LED on
    digitalWrite(LED_PIN4, HIGH);          // GET turns the LED on
    delay(2000);
}

// Recebe o numero da posição 4 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_5)) {
    digitalWrite(LED_PIN1, HIGH);         // GET /H turns the LED on
    digitalWrite(LED_PIN2, HIGH);         // GET /H turns the LED on
    digitalWrite(LED_PIN3, LOW);          // GET /H turns the LED on
    digitalWrite(LED_PIN4, HIGH);          // GET /H turns the LED on
    delay(2000);
}

// Recebe o numero da posição 5 da jogada do jogo do galo

```

```

else if (request ->hasParam(PARAM_INPUT_6)) {
    digitalWrite(LED_PIN1, LOW);           // GET /H turns the LED on
    digitalWrite(LED_PIN2, HIGH);          // GET /H turns the LED on
    digitalWrite(LED_PIN3, LOW);          // GET /H turns the LED on
    digitalWrite(LED_PIN4, HIGH);          // GET /H turns the LED on
    delay(2000);
}

// Recebe o numero da posição 6 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_7)) {
    digitalWrite(LED_PIN1, HIGH);          // GET /H turns the LED off
    digitalWrite(LED_PIN2, LOW);           // GET /H turns the LED off
    digitalWrite(LED_PIN3, LOW);           // GET /H turns the LED off
    digitalWrite(LED_PIN4, HIGH);          // GET /H turns the LED off
    delay(2000);
}

// Recebe o numero da posição 7 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_8)) {
    digitalWrite(LED_PIN1, LOW);           // GET /H turns the LED off
    digitalWrite(LED_PIN2, LOW);           // GET /H turns the LED off
    digitalWrite(LED_PIN3, LOW);           // GET /H turns the LED off
    digitalWrite(LED_PIN4, HIGH);          // GET /H turns the LED off
    delay(2000);
}

// Recebe o numero da posição 8 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_9)) {
    digitalWrite(LED_PIN1, HIGH);          // GET /H turns the LED off
    digitalWrite(LED_PIN2, HIGH);          // GET /H turns the LED off
    digitalWrite(LED_PIN3, HIGH);          // GET /H turns the LED off
    digitalWrite(LED_PIN4, LOW);           // GET /H turns the LED off
    delay(2000);
}

// Recebe o numero da posição 9 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_10)) {
    digitalWrite(LED_PIN1, LOW);           // GET /H turns the LED off
    digitalWrite(LED_PIN2, HIGH);          // GET /H turns the LED off
    digitalWrite(LED_PIN3, HIGH);          // GET /H turns the LED off
    digitalWrite(LED_PIN4, LOW);           // GET /H turns the LED off
    delay(2000);
}

else {
    inputMessage = "No message sent";
    inputParam = "none";
}

//Usar linhas seguintes para verificar envio de mensagens
/*Serial.println(inputMessage);
request->send(200, "text/html", "HTTP GET request sent to your ESP on input field (" +
    + inputParam + ") with value: " + inputMessage +
    "<br><a href=\"^\">Return to Home Page</a>");*/
}

);

```

```
server.onNotFound(notFound);
server.begin();
}
```

```
void loop() {
```

```
}
```

## **Exemplo 10: Enviar texto e comandos para LED's através de formulários HTML (como Acess Point)**

```
#http://onlineshouter.com/use-esp8266-wifi-modes-station-access-point/
#include <Arduino.h>
#ifdef ESP32
  #include <WiFi.h>
  #include <AsyncTCP.h>
#else
  #include <ESP8266WiFi.h>
  #include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

AsyncWebServer server(80);

// REPLACE WITH YOUR NETWORK CREDENTIALS
const char* ssid    = "ROBOT";
const char* password = "eaci2020"; //tem de ter no minimo 8 char

const char* PARAM_INPUT_1 = "input1";
const char* PARAM_INPUT_2 = "input2";
const char* PARAM_INPUT_3 = "input3";
const char* PARAM_INPUT_4 = "input4";
const char* PARAM_INPUT_5 = "input5";
const char* PARAM_INPUT_6 = "input6";
const char* PARAM_INPUT_7 = "input7";
const char* PARAM_INPUT_8 = "input8";
const char* PARAM_INPUT_9 = "input9";
const char* PARAM_INPUT_10 = "input10";

String palavra;
char letra;

IPAddress local_IP(192,168,4,22);
IPAddress gateway(192,168,4,9);
IPAddress subnet(255,255,255,0);

/* LED */
#define LED_PIN1 15
#define LED_PIN2 16
#define LED_PIN3 17
#define LED_PIN4 18
#define LED_PIN5 19

// HTML web page to handle 3 input fields (input1, input2, input3)
const char index_html[] PROGMEM = R"rawliteral(
```

```
<!DOCTYPE HTML><html>
<head>
<meta charset="UTF-8"> <!--portuguese chars-->
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Comando de Robôs por ESP32</title>
</head>

<body>

<h1>Palavra a enviar ao Robô</h1>
<form action="/get">
    Escreva a palavra (apenas com 10 caracteres): <input type="text" name="input1">
    <input type="submit" value="Enviar">
</form><br>

<div style="background-color:lightblue"><center>
<h1>Jogo do Galo</h1>
<table>
    <!--Linha 1-->
    <tr><td>
        <form action="/get">
            <!--input4:<input type="submit" value="Submit">-->
            <!--input4: <input type="text" name="input4">-->
            <input type="submit" name="input2" value="Posição 1" />
        </form></td>
        <td>
            </&emsp;> <!--4 spaces-->
        </td>
        <td>
            <form action="/get">
                <input type="submit" name="input3" value="Posição 2" />
            </td>
            </form>
        <td>
            </&emsp;> <!--4 spaces-->
        </td>
        <td>
            <form action="/get">
                <input type="submit" name="input4" value="Posição 3" />
            </td>
            </form>
        <!--Linha 2-->
    </tr>
</table>
<tr><td>
    <form action="/get">
        <input type="submit" name="input5" value="Posição 4" />
    </form></td>
    <td>
        </&emsp;> <!--4 spaces-->
    </td>
</tr>
```

```

<td>
<form action="/get">
<input type="submit" name="input6" value="Posição 5" />
</td>
<td>
</&emsp> <!--4 spaces-->
</td>
<td>
<form action="/get">
<input type="submit" name="input7" value="Posição 6" />
</td>
</tr></table>
<!--Linha 3-->
</tr>
</table>
<table>
<tr><td>
<form action="/get">
<input type="submit" name="input8" value="Posição 7" />
</form></td>
<td>
</&emsp> <!--4 spaces-->
</td>
<td>
<form action="/get">
<input type="submit" name="input9" value="Posição 8" />
</td>
<td>
</&emsp> <!--4 spaces-->
</td>
<td>
<form action="/get">
<input type="submit" name="input10" value="Posição 9" />
</td>
</tr></table>

</br>
</div>
<p id="demo">Hi.</p>
<script>
document.getElementById("demo").innerHTML="Programado por: Fernando Costeira e Paulo
Galvão";
</script>
<p><a href="https://github.com/labF212/ESP32">Obter código</a></p>

</body></html>)rawliteral";

void notFound(AsyncWebServerRequest *request) {
    request->send(404, "text/plain", "Not found");
}

```

```

}

void setup() {
  Serial.begin(115200);

  pinMode(LED_PIN1, OUTPUT);
  pinMode(LED_PIN2, OUTPUT);
  pinMode(LED_PIN3, OUTPUT);
  pinMode(LED_PIN4, OUTPUT);
  pinMode(LED_PIN5, OUTPUT);
  delay(10);

// connectWiFi();

Serial.print("Setting soft-AP configuration ... ");

Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");

Serial.print("Setting soft-AP ... ");

Serial.println(WiFi.softAP(ssid,password) ? "Ready" : "Failed!");

Serial.print("Soft-AP IP address = ");

Serial.println(WiFi.softAPIP());



//WiFi.mode(WIFI_STA);
//WiFi.begin(ssid, password);
/* if (WiFi.waitForConnectResult() != WL_CONNECTED) {
  Serial.println("WiFi Failed!");
  return;
}
Serial.println();
Serial.print("IP Address: ");
Serial.println(WiFi.localIP());
Serial.println(WiFi.softAPIP());
*/
// Send web page with input fields to client
server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request){
  request->send_P(200, "text/html", index_html);
});




// Jogo da Palavra
// Send a GET request to <ESP_IP>/get?input1=<inputMessage>
server.on("/get", HTTP_GET, [] (AsyncWebServerRequest *request) {
  String inputMessage;
  String inputParam;
  // GET input1 value on <ESP_IP>/get?input1=<inputMessage>
  //Recebe a Palavra

```

```

if (request->hasParam(PARAM_INPUT_1)) {
    inputMessage = request->getParam(PARAM_INPUT_1)->value();
    //inputParam = PARAM_INPUT_1;
    Serial.println("Letra");
    //Serial.println(inputParam);

    inputMessage.toLowerCase();
    Serial.println(inputMessage);

    //palavra+ = inputParam;
    for (int i = 0; i < 10; i++) {
        //Serial.println(palavra[i]);

        char letra = inputMessage[i];
        Serial.println(letra);

        if(letra == 'a'){
            digitalWrite(LED_PIN1, LOW);
            digitalWrite(LED_PIN2, HIGH);
            digitalWrite(LED_PIN3, HIGH);
            digitalWrite(LED_PIN4, HIGH);
            digitalWrite(LED_PIN5, HIGH);
            delay(400);
        }

        if(letra == 'b'){
            digitalWrite(LED_PIN1, HIGH);
            digitalWrite(LED_PIN2, LOW);
            digitalWrite(LED_PIN3, HIGH);
            digitalWrite(LED_PIN4, HIGH);
            digitalWrite(LED_PIN5, HIGH);
            delay(400);
        }

        if(letra == 'c'){
            digitalWrite(LED_PIN1, LOW);
            digitalWrite(LED_PIN2, LOW);
            digitalWrite(LED_PIN3, HIGH);
            digitalWrite(LED_PIN4, HIGH);
            digitalWrite(LED_PIN5, HIGH);
            delay(400);
        }

        if(letra == 'd'){
            digitalWrite(LED_PIN1, HIGH);
            digitalWrite(LED_PIN2, HIGH);
            digitalWrite(LED_PIN3, LOW);
            digitalWrite(LED_PIN4, HIGH);
            digitalWrite(LED_PIN5, HIGH);
            delay(400);
        }
    }
}

```

```
if(letra == 'e'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'f'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'g'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'h'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'i'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'j'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
```

```
delay(400);
}

if(letra == 'k'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'l'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'm'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'n'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'o'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, LOW);
    digitalWrite(LED_PIN5, HIGH);
    delay(400);
}

if(letra == 'p'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, HIGH);
```

```
digitalWrite(LED_PIN4, HIGH);
digitalWrite(LED_PIN5, LOW);
delay(400);
}

if(letra == 'q'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 'r'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 's'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, LOW);
    digitalWrite(LED_PIN3, HIGH);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 't'){
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2,HIGH);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 'u'){
    digitalWrite(LED_PIN1, LOW);
    digitalWrite(LED_PIN2, HIGH);
    digitalWrite(LED_PIN3, LOW);
    digitalWrite(LED_PIN4, HIGH);
    digitalWrite(LED_PIN5, LOW);
    delay(400);
}

if(letra == 'v'){
    digitalWrite(LED_PIN1, HIGH);
```

```

digitalWrite(LED_PIN2, LOW);
digitalWrite(LED_PIN3, LOW);
digitalWrite(LED_PIN4, HIGH);
digitalWrite(LED_PIN5, LOW);
delay(400);
}

if(letra == 'x'){
  digitalWrite(LED_PIN1, LOW);
  digitalWrite(LED_PIN2, LOW);
  digitalWrite(LED_PIN3, LOW);
  digitalWrite(LED_PIN4, HIGH);
  digitalWrite(LED_PIN5, LOW);
  delay(400);
}

if(letra == 'y'){
  digitalWrite(LED_PIN1, HIGH);
  digitalWrite(LED_PIN2, HIGH);
  digitalWrite(LED_PIN3, HIGH);
  digitalWrite(LED_PIN4, LOW);
  digitalWrite(LED_PIN5, LOW);
  delay(400);
}

if(letra == 'z'){
  digitalWrite(LED_PIN1, LOW);
  digitalWrite(LED_PIN2, HIGH);
  digitalWrite(LED_PIN3, HIGH);
  digitalWrite(LED_PIN4, LOW);
  digitalWrite(LED_PIN5, LOW);
  delay(400);
}

}

}

// JOGO DO GALO

// GET input2 value on <ESP_IP>/get?input2=<inputMessage>
// Recebe o numero da posição 1 da jogada do jogo do galo
else if (request->hasParam(PARAM_INPUT_2)) {
  digitalWrite(LED_PIN1, LOW);          // GET turns the LED off
  digitalWrite(LED_PIN2, HIGH);         // GET turns the LED off
  digitalWrite(LED_PIN3, HIGH);         // GET turns the LED on
  digitalWrite(LED_PIN4, HIGH);         // GET turns the LED on
  delay(2000);
}

// GET input3 value on <ESP_IP>/get?input3=<inputMessage>
// Recebe o numero da posição 2 da jogada do jogo do galo
else if (request->hasParam(PARAM_INPUT_3)) {
  digitalWrite(LED_PIN1, HIGH);        // GET turns the LED off

```

```

digitalWrite(LED_PIN2, LOW);           // GET turns the LED off
digitalWrite(LED_PIN3, HIGH);          // GET turns the LED on
digitalWrite(LED_PIN4, HIGH);          // GET turns the LED on
delay(2000);
}

// Recebe o numero da posição 3 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_4)) {
    digitalWrite(LED_PIN1, LOW);        // GET turns the LED off
    digitalWrite(LED_PIN2, LOW);        // GET turns the LED off
    digitalWrite(LED_PIN3, HIGH);       // GET turns the LED on
    digitalWrite(LED_PIN4, HIGH);       // GET turns the LED on
    delay(2000);

}

// Recebe o numero da posição 4 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_5)) {
    digitalWrite(LED_PIN1, HIGH);       // GET /H turns the LED on
    digitalWrite(LED_PIN2, HIGH);       // GET /H turns the LED on
    digitalWrite(LED_PIN3, LOW);        // GET /H turns the LED on
    digitalWrite(LED_PIN4, HIGH);       // GET /H turns the LED on
    delay(2000);

}

// Recebe o numero da posição 5 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_6)) {
    digitalWrite(LED_PIN1, LOW);        // GET /H turns the LED on
    digitalWrite(LED_PIN2, HIGH);       // GET /H turns the LED on
    digitalWrite(LED_PIN3, LOW);        // GET /H turns the LED on
    digitalWrite(LED_PIN4, HIGH);       // GET /H turns the LED on
    delay(2000);

}

// Recebe o numero da posição 6 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_7)) {
    digitalWrite(LED_PIN1, HIGH);       // GET /H turns the LED off
    digitalWrite(LED_PIN2, LOW);        // GET /H turns the LED off
    digitalWrite(LED_PIN3, LOW);        // GET /H turns the LED off
    digitalWrite(LED_PIN4, HIGH);       // GET /H turns the LED off
    delay(2000);

}

// Recebe o numero da posição 7 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_8)) {
    digitalWrite(LED_PIN1, LOW);        // GET /H turns the LED off
    digitalWrite(LED_PIN2, LOW);        // GET /H turns the LED off
    digitalWrite(LED_PIN3, LOW);        // GET /H turns the LED off
    digitalWrite(LED_PIN4, HIGH);       // GET /H turns the LED off
    delay(2000);

}

// Recebe o numero da posição 8 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_9)) {
    digitalWrite(LED_PIN1, HIGH);       // GET /H turns the LED off
    digitalWrite(LED_PIN2, HIGH);       // GET /H turns the LED off
    digitalWrite(LED_PIN3, HIGH);       // GET /H turns the LED off
    digitalWrite(LED_PIN4, LOW);        // GET /H turns the LED off
}

```

```

    delay(2000);
}

// Recebe o numero da posição 9 da jogada do jogo do galo
else if (request ->hasParam(PARAM_INPUT_10)) {
    digitalWrite(LED_PIN1, LOW);           // GET /H turns the LED off
    digitalWrite(LED_PIN2, HIGH);          // GET /H turns the LED off
    digitalWrite(LED_PIN3, HIGH);          // GET /H turns the LED off
    digitalWrite(LED_PIN4, LOW);           // GET /H turns the LED off
    delay(2000);
}

else {
    inputMessage = "No message sent";
    inputParam = "none";
}
//Usar linhas seguintes para verificar envio de mensagens
/*Serial.println(inputMessage);
request->send(200, "text/html", "HTTP GET request sent to your ESP on input field (" +
    + inputParam + ") with value: " + inputMessage +
    "<br><a href=\"^\">Return to Home Page</a>");*/
*/
});

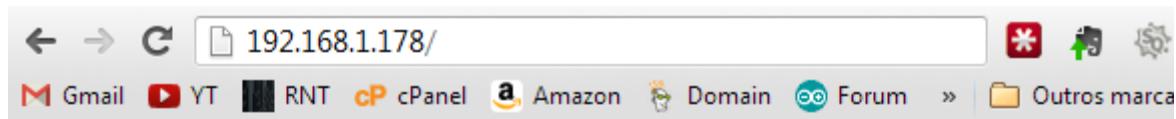
server.onNotFound(notFound);
server.begin();
}

void loop() {
}

```

## Exemplo 11: Controlar LED e servo com webserver

<https://randomnerdtutorials.com/arduino-webserver-with-an-arduino-ethernet-shield/>



# Random Nerd Tutorials Project

## Arduino with Ethernet Shield

Turn On LED

Turn Off LED

Rotate Left

Rotate Right

Created by Rui Santos. Visit <http://randomnerdtutorials.com> for more projects!

```

/*
Created by Rui Santos
Visit: http://randomnerdtutorials.com for more arduino projects

Arduino with Ethernet Shield
*/

#include <SPI.h>
#include <Ethernet.h>
#include <Servo.h>
int led = 4;
Servo microservo;
int pos = 0;
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //physical mac address
byte ip[] = { 192, 168, 1, 178 }; // ip in lan (that's what
you need to use in your browser. ("192.168.1.178")
byte gateway[] = { 192, 168, 1, 1 }; // internet access via
router
byte subnet[] = { 255, 255, 255, 0 }; //subnet mask
EthernetServer server(80); //server port
String readString;

void setup() {
// Open serial communications and wait for port to open:
Serial.begin(9600);
while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
}
pinMode(led, OUTPUT);
microservo.attach(7);
// start the Ethernet connection and the server:
Ethernet.begin(mac, ip, gateway, subnet);
server.begin();
Serial.print("server is at ");
Serial.println(Ethernet.localIP());
}

void loop() {
// Create a client connection
EthernetClient client = server.available();
if (client) {
    while (client.connected()) {
        if (client.available()) {
            char c = client.read();

            //read char by char HTTP request
            if (readString.length() < 100) {
                //store characters to string
                readString += c;
                //Serial.print(c);
            }

            //if HTTP request has ended
            if (c == '\n') {
                Serial.println(readString); //print to serial monitor for debugging

                client.println("HTTP/1.1 200 OK"); //send new page
                client.println("Content-Type: text/html");
                client.println();
                client.println("<HTML>");
                client.println("<HEAD>");
                client.println("<meta name='apple-mobile-web-app-capable' content='yes' />");
            }
        }
    }
}

```

```

        client.println("<meta name='apple-mobile-web-app-status-bar-style' content='black-translucent' />");
        client.println("<link rel='stylesheet' type='text/css' href='http://randomnerdtutorials.com/ethernetcss.css' />");
        client.println("<TITLE>Random Nerd Tutorials Project</TITLE>");
        client.println("</HEAD>");
        client.println("<BODY>");
        client.println("<H1>Random Nerd Tutorials Project</H1>");
        client.println("<hr />");
        client.println("<br />");
        client.println("<H2>Arduino with Ethernet Shield</H2>");
        client.println("<br />");
        client.println("<a href=\"/?button1on\">Turn On LED</a>");
        client.println("<a href=\"/?button1off\">Turn Off LED</a><br />");
        client.println("<br />");
        client.println("<br />");
        client.println("<a href=\"/?button2on\">Rotate Left</a>");
        client.println("<a href=\"/?button2off\">Rotate Right</a><br />");
        client.println("<p>Created by Rui Santos. Visit  

http://randomnerdtutorials.com for more projects!</p>");
        client.println("<br />");
        client.println("</BODY>");
        client.println("</HTML>");

        delay(1);
        //stopping client
        client.stop();
        //controls the Arduino if you press the buttons
        if (readString.indexOf("?button1on") >0){
            digitalWrite(led, HIGH);
        }
        if (readString.indexOf("?button1off") >0){
            digitalWrite(led, LOW);
        }
        if (readString.indexOf("?button2on") >0){
            for(pos = 0; pos < 180; pos += 3) // goes from 0 degrees to 180
degrees
            {
                microservo.write(pos); // in steps of 1 degree
                delay(15); // tell servo to go to
            }
        }
        if (readString.indexOf("?button2off") >0){
            for(pos = 180; pos>=1; pos-=3) // goes from 180 degrees to 0
degrees
            {
                microservo.write(pos); // tell servo to go to
                delay(15); // waits 15ms for the servo
            }
        }
        //clearing string for next read
        readString="";
    }
}
}

```

## 2 Programação através de MicroPython

### Como instalar o programa Thonny

Linux:

```
$ sudo apt install python3-pip  
$ pip3 install esptool  
$ sudo pip3 install thonny  
$ sudo apt install python3-tk
```

Also add permission to user to access the USB port:

```
$ sudo usermod -a -G dialout <user>  
$ sudo chmod a+rwx <port>
```

Run Thonny:

```
$ thonny
```

Para instalar a última versão faça simplesmente:

```
bash <(curl -s https://thonny.org/installer-for-linux)
```

Windows:

Ir a <https://thonny.org/> fazer o download e instalar.

Para quem tem o chocolatery instalado:

```
choco install thonny
```

Depois deve-se descarregar a ferramenta **esptool** no Thonny em:

Tools → Manage Plugins

Também existe a versão portátil.

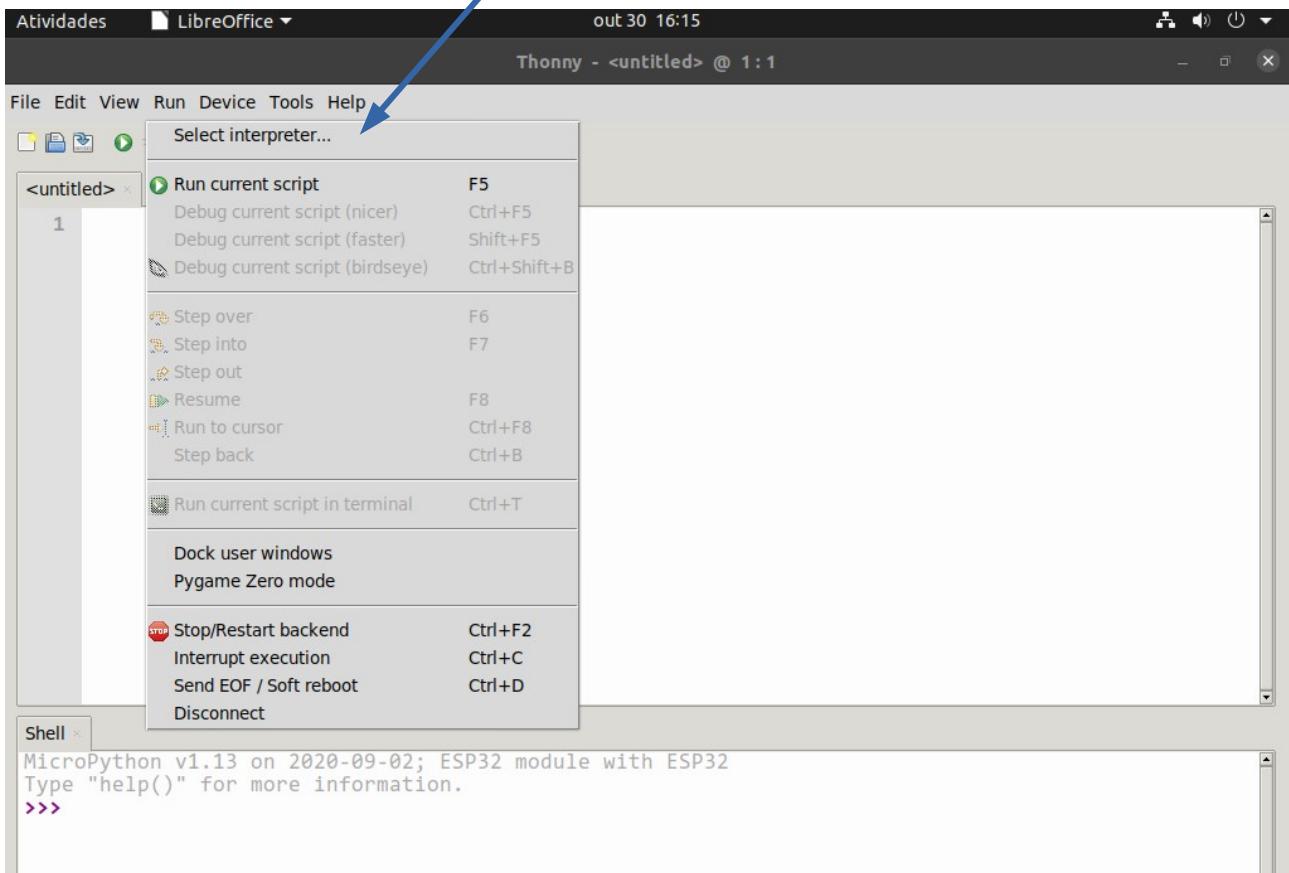
# Como usar o Thonny para programar MicroPython em ESP-32

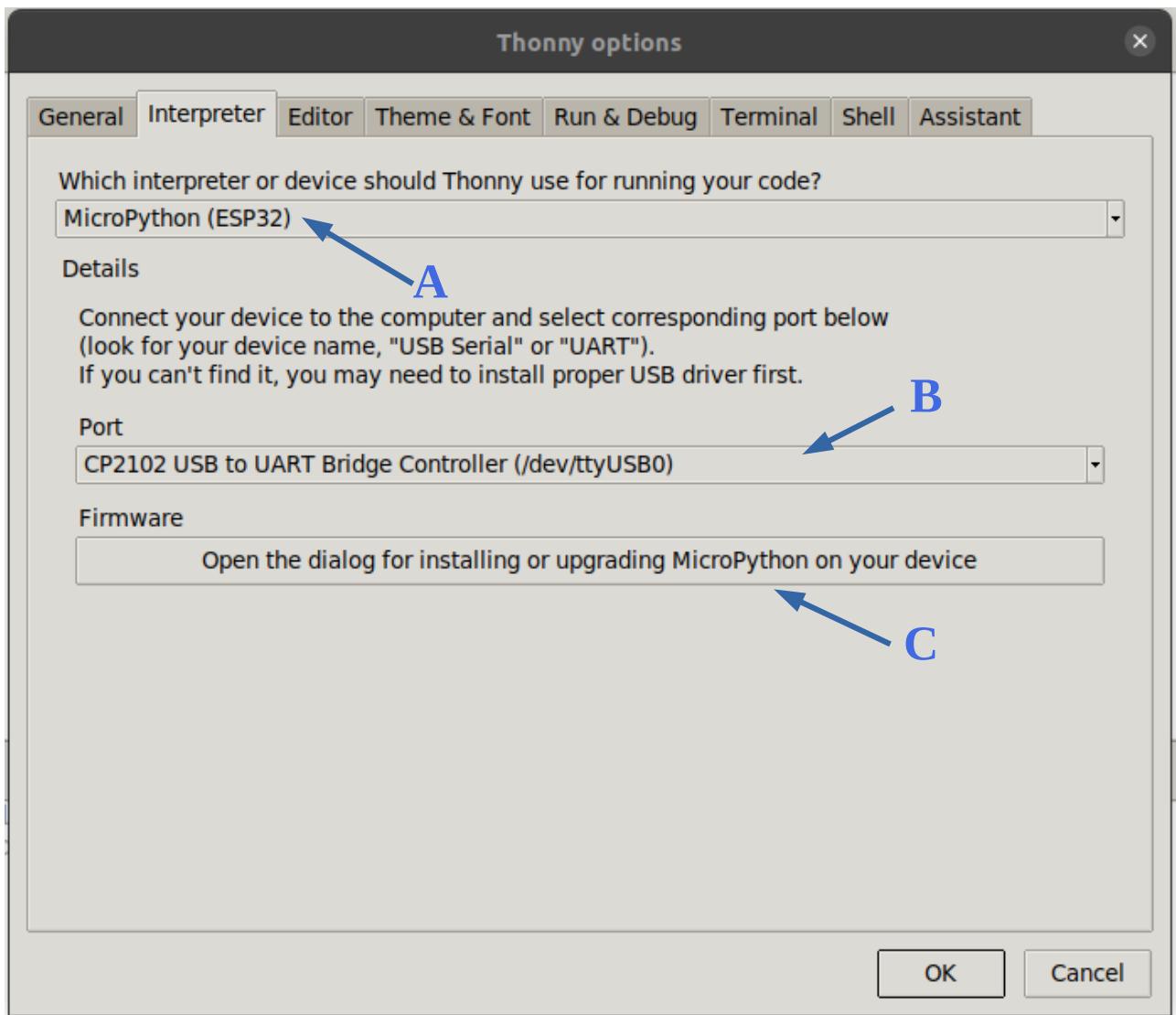
Nota: Também é válido para outros microprocessadores

## 1. Como enviar o firmware para o ESP-32

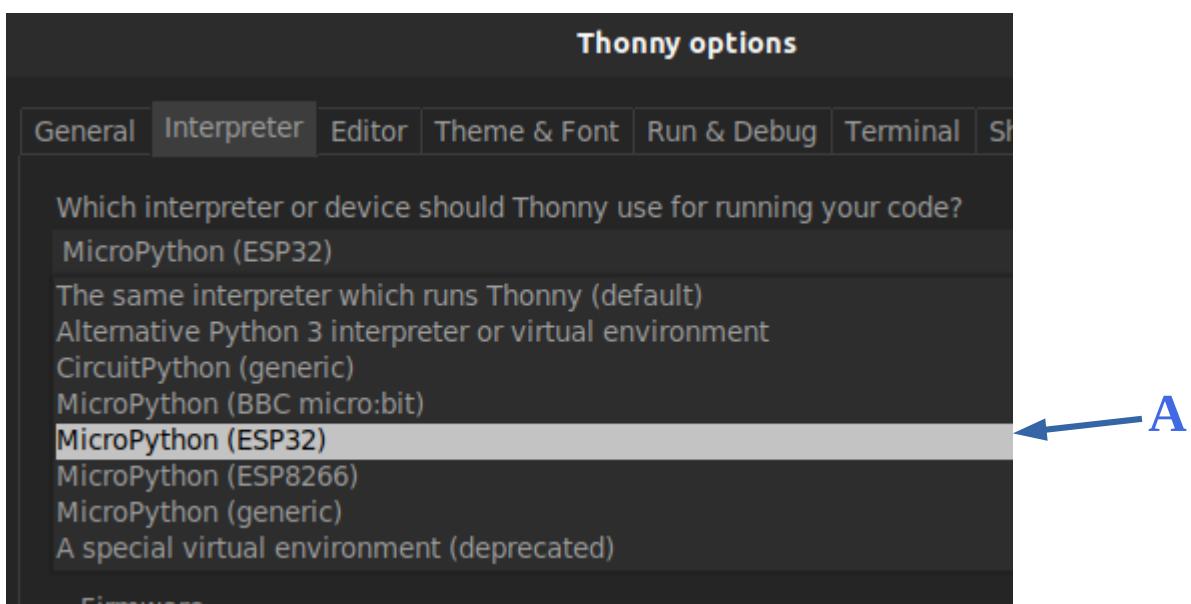
Fazer o download do firmware em: <https://micropython.org/download/esp32/>

Ir ao menu Run → Select interpreter...

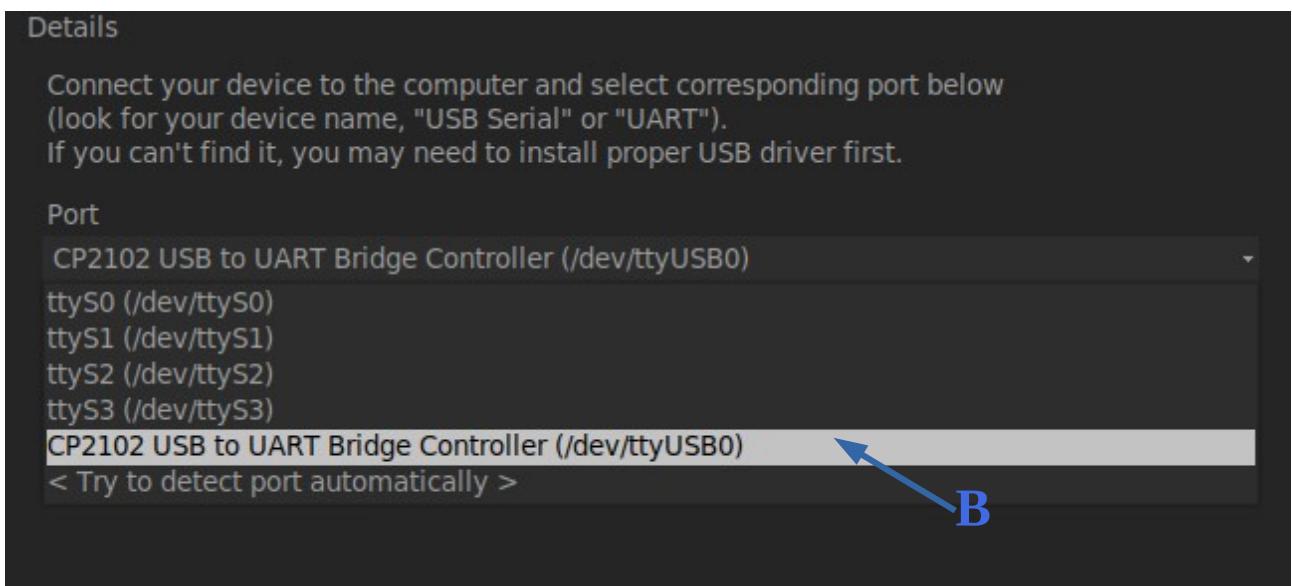




A - Escolher o interpretador:

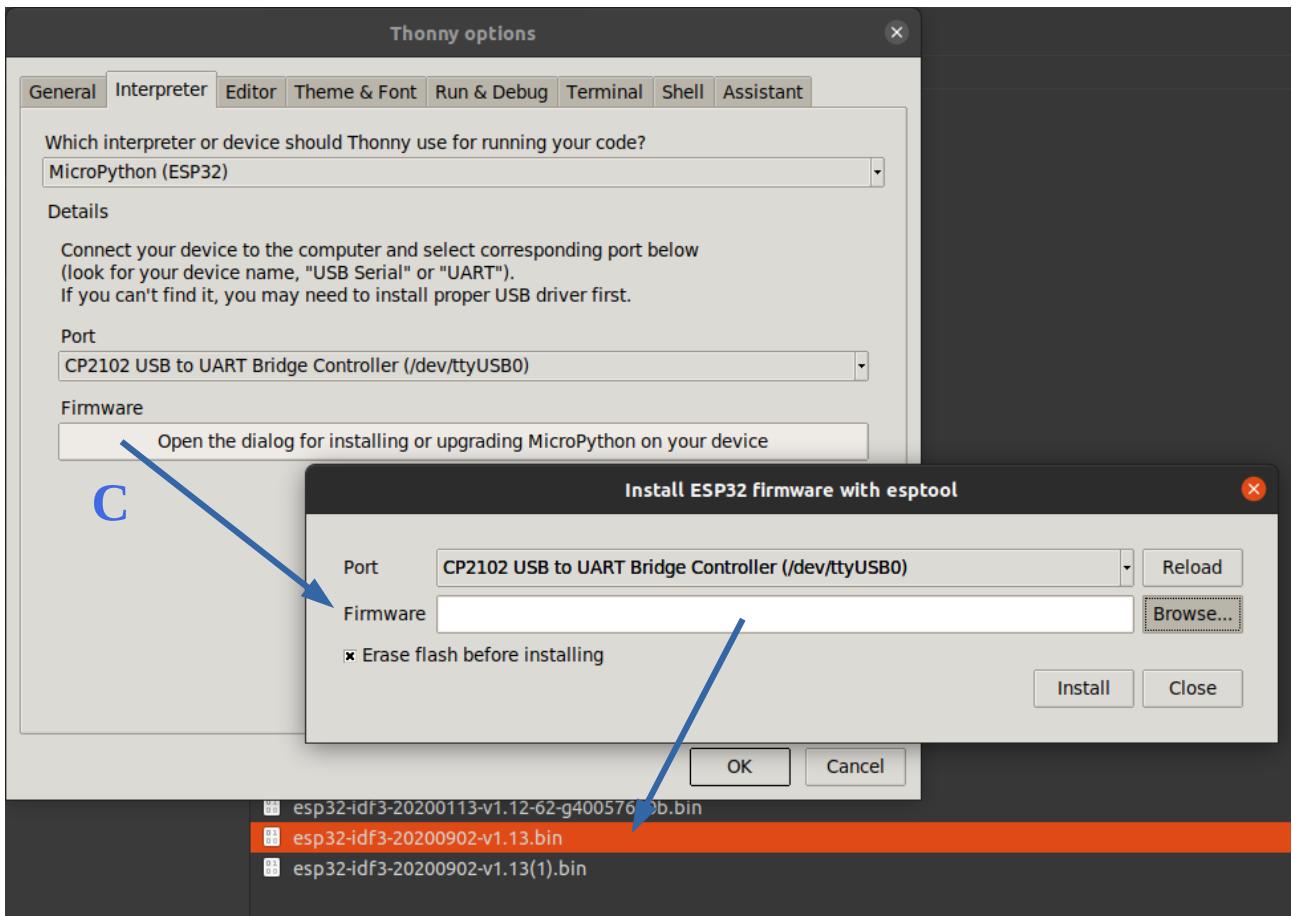


B - Escolher a porta de comunicação:

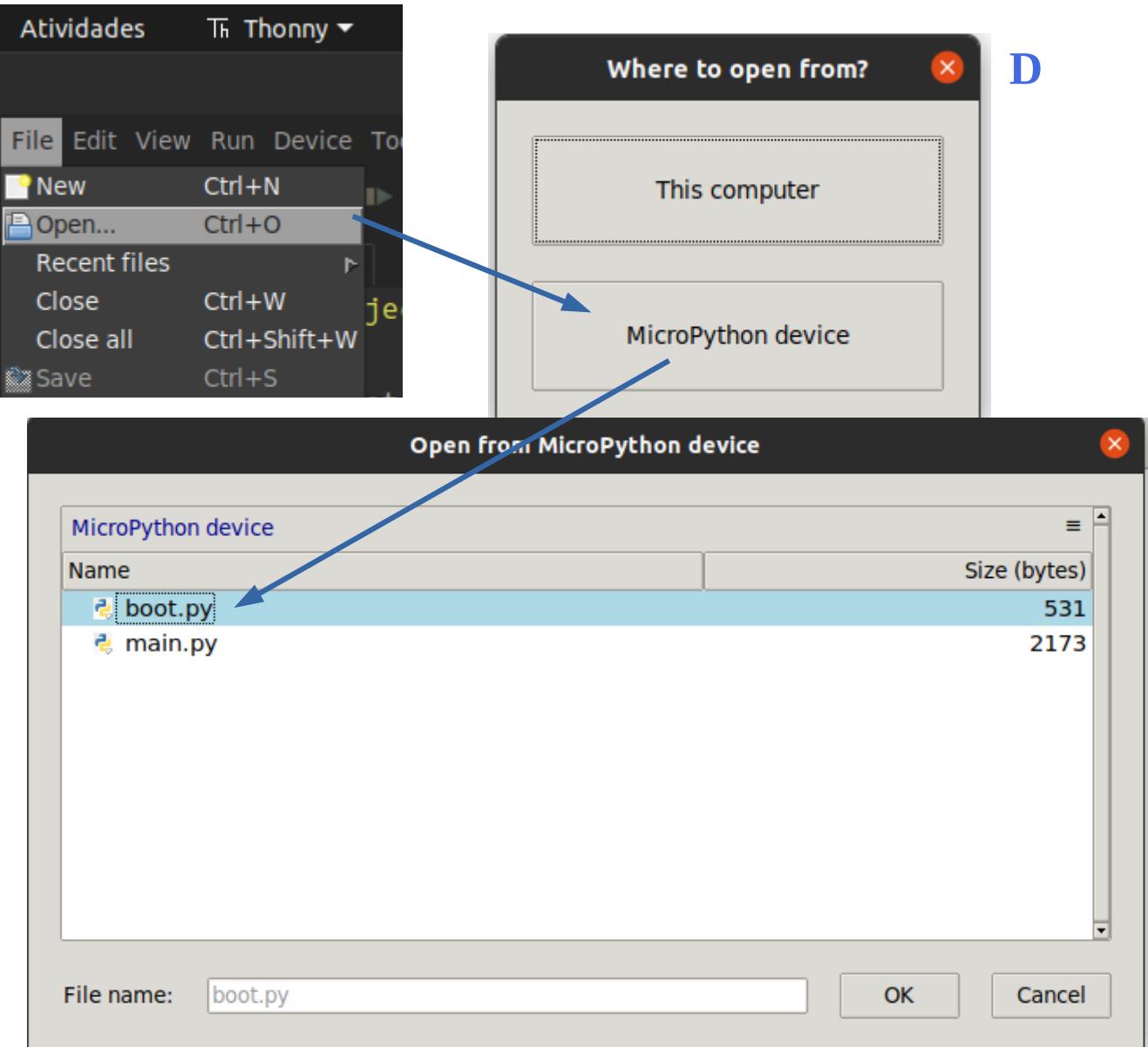


Em Windows será, por exemplo, COM1.

C - Escolher o Firmware de acordo com o interpretador.



## 2. Como ver os ficheiros existentes no ESP-32



O ficheiro dentro de parêntesis rectos [boot.py] encontra-se no ESP e o ficheiro HC\_SR04.py encontra-se no computador.

The screenshot shows the Thonny IDE code editor. A blue arrow points from the top of the editor area to the title bar 'Thonny - MicroPython device :: /boot.py @ 1:12'. The code editor displays two tabs: '[ boot.py ]' and '[ HC\_SR04.py ]'. The HC\_SR04.py tab is active, showing the following Python code:

```
1 # Complete project details at https://RandomNerdTutorials.com
2
3 try:
4     import usocket as socket
5 except:
6     import socket
7
8 from machine import Pin
9 import network
```

## Comunicação:

- Com erro:

Quando a Shell der este problema tente fazer STOP e se continuar a dar o mesmo erro, pressione o botão EN do ESP-32.

```
Shell x

Could not enter REPL. Trying again with 1 second waiting time...
Could not enter REPL. Trying again with 3 second waiting time...
Could not enter REPL. Trying again with 5 second waiting time...
Could not enter REPL. Giving up. Read bytes:
b''

Your options:

- check connection properties;
- make sure the device has suitable firmware;
- make sure the device is not in bootloader mode;
- reset the device and try again;
- try other serial clients (Putty, TeraTerm, screen, ...);
- ask for help in Thonny's forum or issue tracker.

Backend terminated or disconnected. Use 'Stop/Restart' to restart.
```

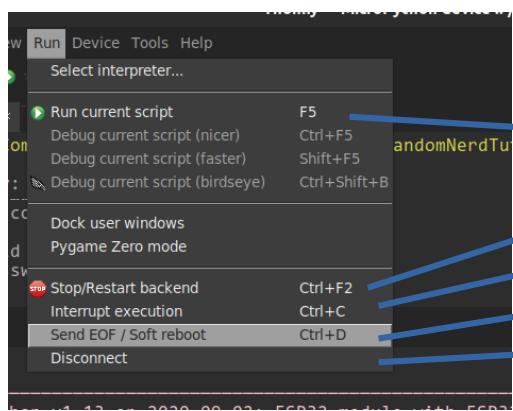
- Comunicação OK:

Nota: Só quando existe comunicação é que aparece a caixa de diálogo D, quando se pretende abrir ficheiros no ESP.

```
Shell x

>>>

MicroPython v1.13 on 2020-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>>
```



Outras opções:

- Correr o programa atual
- Parar/Recomeçar a comunicação
- Interromper programa
- Fazer Reboot por software
- Desligar

O micropython corre por esta ordem:

1. boot.py
2. main.py
3. outros

## Exemplo 1: Como criar funções e importá-la de outro ficheiro

1. Criar um ficheiro com o nome boot.py

```
import esp32
```

```
import blink
```

```
blink.blink_led()
```

2. Dados no ficheiro blink.py

```
from machine import Pin  
import time
```

```
def blink_led():  
    counter=5  
    p= Pin (2,Pin.OUT)  
    while (counter>0):  
        p.on()  
        time.sleep(0.5)  
        p.off()  
        time.sleep(0.15)  
        counter -=1  
    print("Done")
```

## Exemplo 2: Read Hall Effect Sensor and write to file

To read the ESP32 hall effect sensor using MicroPython, you just need to use the following snippet of code:

```
import esp32
esp32.hall_sensor()
```

You need to import the esp32 module. Then, use the hall\_sensor() method.

If you want to print the readings on the shell, you just need to use the print() function:

```
print(esp32.hall_sensor())
```

If you're just getting started with MicroPython, you can read the following tutorial:

- [Getting Started with MicroPython on ESP32](#)

```
from machine import Pin
from time import sleep
import esp32

esp32.hall_sensor()
```

```
while True:
    print(esp32.hall_sensor())
    sleep(0.5)
```

com LED:

```
import esp32

led = Pin(16, Pin.OUT)
esp32.hall_sensor()
led.off()

while True:
    led.on()
    print("ON")
    print(esp32.hall_sensor())
    sleep(0.5)
    led.off()
    print("OFF")
    sleep(0.5)
```

## Escrever num ficheiro

```
from machine import Pin
from time import sleep
import esp32
import os

led = Pin(16, Pin.OUT)

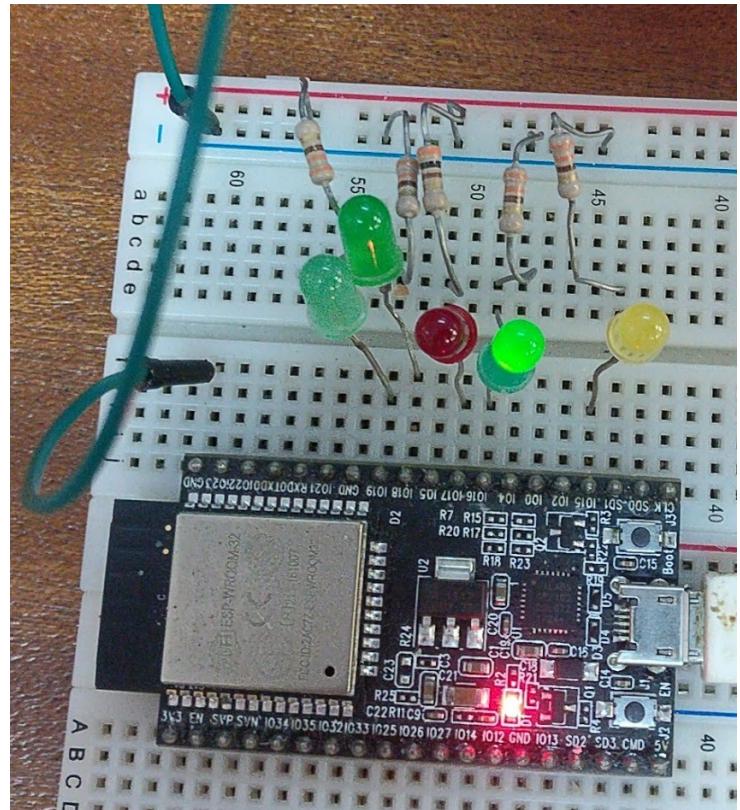
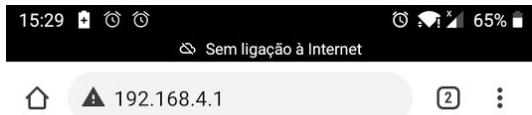
led.off()
print(os.listdir())
a=0

while True:
    led.on()
    a=a+1
    hall_value= str(esp32.hall_sensor())
    temp32= str(int((esp32.raw_temperature()-32)/1.8))
    #f.write(" %s " % a)
    with open("myfile.txt", "a") as f:
        f.write(str(a) + '\t' + temp32 + '\t' + hall_value + '\n')
    #print(str(a) + '\t' + temp32 '\n')
    print(str(a) + '\t' + temp32 + '\t' + hall_value)
    sleep(0.5)
    led.off()
    sleep(0.5)
```

myfile.txt

N	V	T
1	47	42
2	47	38
3	47	40
4	47	44
5	47	45
6	47	47
7	47	47
8	47	45
9	47	45
10	47	49
1	47	39
2	47	49
3	47	47

## Exemplo 3: Comando de um led através de AP (WebServer)



### Boot.py

```
# Complete project details at https://RandomNerdTutorials.com

try:
    import usocket as socket
except:
    import socket

from machine import Pin
import network

import esp32
import esp
import os
esp.osdebug(None)

import gc
gc.collect()

ap = network.WLAN(network.AP_IF) # create access-point interface
ap.active(True)      # activate the interface
```

```

ap.config(essid='ESP-AP') # set the ESSID of the access point
ap.config(max_clients=10) # set how many clients can connect to the network

while ap.isconnected() == False:
    pass

print('Connection successful')
print(ap.ifconfig())

# ESP32 GPIO 16
relay = Pin(16, Pin.OUT)

# ESP8266 GPIO 5
#relay = Pin(5, Pin.OUT)

```

## Main.py

```

# Complete project details at https://RandomNerdTutorials.com

print(os.listdir())
a=0

def web_page():
    if relay.value() == 1:
        relay_state = "on"
    else:
        relay_state = 'checked'
    html = """<html><head><meta charset="UTF-8"><title>Medição de Energia</title><meta name="viewport" content="width=device-width, initial-scale=1"><style>
body{font-family:Arial; text-align: center; margin: 0px auto; padding-top:30px;}
.switch{position:relative;display:inline-block;width:120px;height:68px}.switch
input{display:none}
.slider{position:absolute;top:0;left:0;right:0;bottom:0;background-color:#ccc;border-radius:34px}
.slider:before{position:absolute;content:"";height:52px;width:52px;left:8px;bottom:8px;background-color:#fff;-webkit-transition:.4s;transition:.4s;border-radius:68px}
input:checked+.slider{background-color:#2196F3}
input:checked+.slider:before{-webkit-transform:translateX(52px);-ms-transform:translateX(52px);transform:translateX(52px)}
</style><script>function toggleCheckbox(element) { var xhr = new XMLHttpRequest();
if(element.checked){ xhr.open("GET", "/?relay=on", true); }
else { xhr.open("GET", "/?relay=off", true); } xhr.send(); }</script></head><body>
<h1>Medição de Energia</h1><label class="switch"><input type="checkbox"
onchange="toggleCheckbox(this)" %s><span class="slider">
</span></label></body></html>"""" % (relay_state)
return html

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("", 80))
s.listen(5)
while True:
    try:

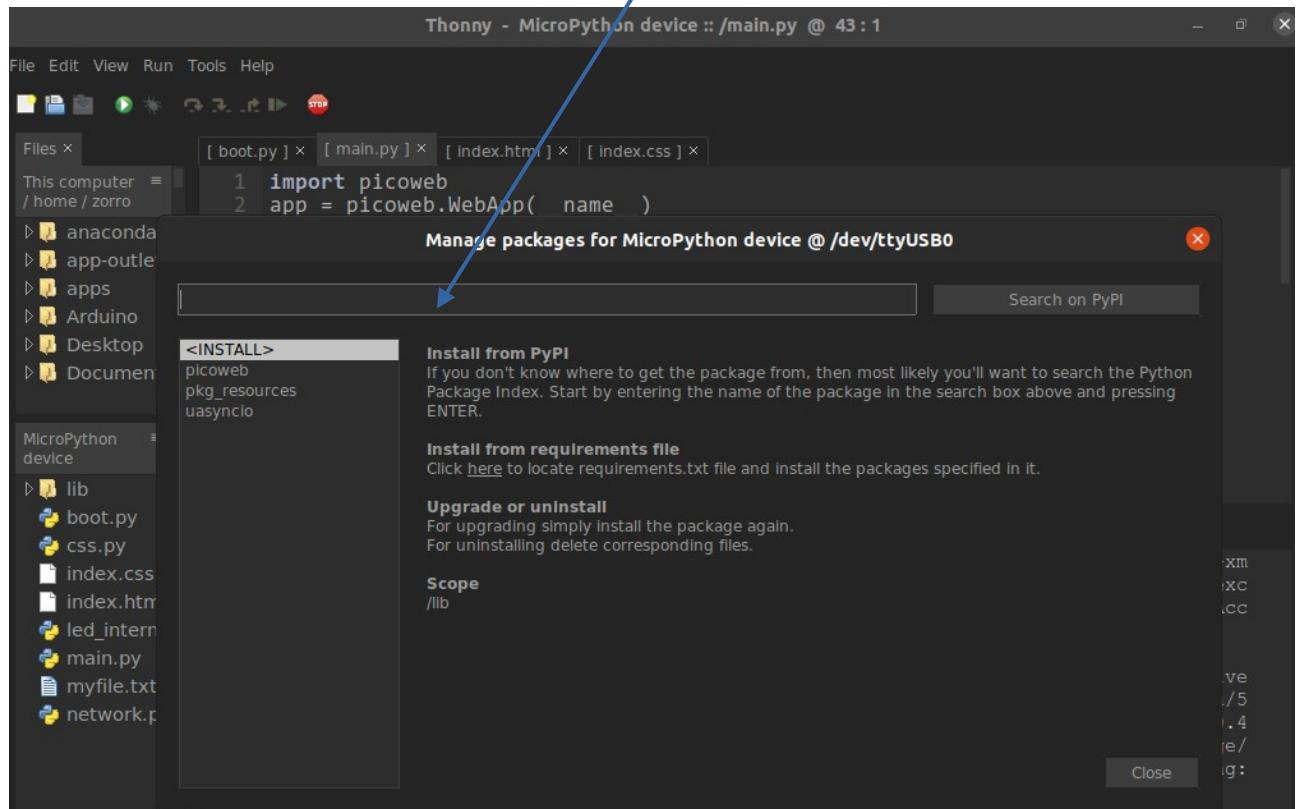
```

```
if gc.mem_free() < 102000:
    gc.collect()
conn, addr = s.accept()
conn.settimeout(3.0)
print('Got a connection from %s' % str(addr))
request = conn.recv(1024)
conn.settimeout(None)
request = str(request)
print('Content = %s' % request)
relay_on = request.find('/?relay=on')
relay_off = request.find('/?relay=off')
if relay_on == 6:
    print('RELAY ON')
    a=a+1
    hall_value= str(esp32.hall_sensor())
    temp32= str(int((esp32.raw_temperature()-32)/1.8))
    with open("myfile.txt", "a") as f:
        f.write(str(a) + '\t' + temp32 + '\t' + hall_value + '\n')
    print(a)
    relay.value(0)
if relay_off == 6:
    print('RELAY OFF')
    relay.value(1)
response = web_page()
conn.send('HTTP/1.1 200 OK\n')
conn.send('Content-Type: text/html\n')
conn.send('Connection: close\n\n')
conn.sendall(response)
conn.close()
except OSError as e:
    conn.close()
    print('Connection closed')
```

# css e html separado

<https://forum.micropython.org/viewtopic.php?t=7018>

Atenção: para isto funcionar adicionar bliblioteca picoweb ao microphyton, fazendo tools → Manage packages e depois escrever no campo:



## Boot

# Complete project details at <https://RandomNerdTutorials.com>  
#<https://forum.micropython.org/viewtopic.php?t=7018>

```
try:  
    import usocket as socket  
except:  
    import socket
```

```
from machine import Pin  
import network
```

```
import esp32  
import esp  
import os
```

```

esp.osdebug(None)

import gc
gc.collect()

ap = network.WLAN(network.AP_IF) # create access-point interface
ap.active(True)     # activate the interface
ap.config(essid='ESP-AP') # set the ESSID of the access point
ap.config(max_clients=10) # set how many clients can connect to the network

while ap.isconnected() == False:
    pass

print('Connection successful')
print(ap.ifconfig())

# ESP32 GPIO 16
relay = Pin(16, Pin.OUT)

# ESP8266 GPIO 5
#relay = Pin(5, Pin.OUT)

```

### main.py

```

import picoweb
app = picoweb.WebApp(__name__)

def web_page():
    if relay.value() == 1:
        gpio_state="ON"
    else:
        gpio_state="OFF"
    #return app.render_template('index.html', ledState = gpio_state)
    #file = open("index.html", "r")
    #page = file.read()
    #html = open("index.html").read().format(ledState=gpio_state)
    #file.close()
    css = open("index.css").read()
    page = css + open("index.html").read().format(relay=gpio_state)
    #page = open("index.html").read().format(ledState=gpio_state)
    return page

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # (address family, TCP Socket)
s.bind(("", 80))
s.listen(5)

while True:
    conn, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = conn.recv(1024)
    request = str(request)
    print('Content = %s' % request)

```

```

relay_on = request.find('?relay=on')
relay_off = request.find('?relay=off')
if relay_on == 6:
    print('LED ON')
    relay.value(1)
if relay_off == 6:
    print('LED OFF')
    relay.value(0)
response = web_page()
conn.send('HTTP/1.1 200 OK\n')
conn.send('Content-Type: text/html\n')
conn.send('Connection: close\n\n')
conn.sendall(response)
conn.close()

```

### **index.html**

```

<!DOCTYPE HTML>
<html>

<head>
    <title>Measure Solar energy</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="shortcut icon" type="image/x-icon" href="data/favicon.ico">

</head>

<body>

<body bgcolor="#3c3c3c">
    <h1>dual pulse spot welder</h1>
    <p>GPIO state: <strong>%s</strong></p>
    <p>
        <a href="/?relay=on">
            <button class="button button2">FIRE</button>
        </a>
        <p>
            #welds: 420
            <form>
                <div>
                    Current: 256
                    <input id="pulse 1" type="number" name="pulse 1" placeholder="pulse 1" min="0"
max="1000" required>
                        <span class="validity"></span>
                        <input type="submit" class="button buttonSubmit">
                </div>
            </form>
            <form>
                <div>
                    Current: 256

```

```

<input id="delay" type="number" name="delay" placeholder="delay" min="0"
max="1000" required>
    <span class="validity"></span>
    <input type="submit" class="button buttonSubmit">
</div>
</form>

<form>
<div>
    Current: 256
    <input id="pulse 2" type="number" name="pulse 2" placeholder="pulse 2" min="0"
max="1000" required>
        <span class="validity"></span>
        <input type="submit" class="button buttonSubmit">
    </div>
</form>

<div>
    <button type="button" class="button button3">M1</button>
    <button type="button" class="button button3">M2</button>
    <button type="button" class="button button3">M3</button>
    <button type="button" class="button button3">M4</button>
    <button type="button" class="button button3">M5</button>
</div>
<p>
    <div>
        <button type="button" class="button button4">save</button>
        <button type="button" class="button button4">load</button>
    <div>
</body>

</html>

```

## index.css

```

<style>
    html {
        font-family: Helvetica;
        display: inline-block;
        margin: 0px auto;
        text-align: center;
        color: #ff5722;
    }

    h1 {
        color: #ff5722;
        padding: 2vh;
    }

    p {
        font-size: 1.5rem;
    }

```

```
}

.button {
    display: inline-block;
    background-color: #263238;
    border: none;
    border-radius: 4px;
    color: white;
    padding: 16px 40px;
    text-decoration: none;
    font-size: 10px;
    margin: 2px;
    cursor: pointer;
}

.button2:hover {
    background-color: #8B0000;
    border-radius: 12px;
}

.button3:hover {
    background-color: #455a64;
    border-radius: 12px;
}

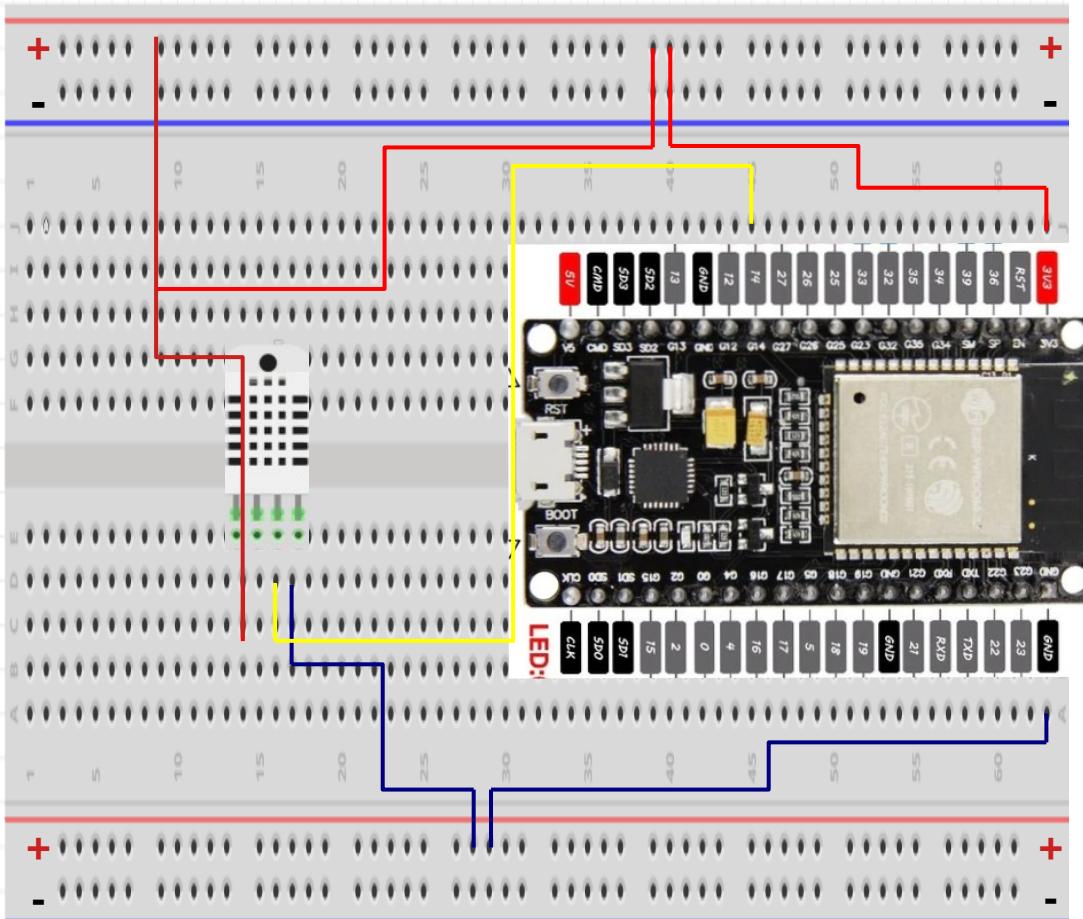
.button4:hover {
    background-color: #49599a;
    border-radius: 12px;
}

input[type=number] {
    width: 20%;
    padding: 12px;
    text-align: center;
    margin: 8px 0;
    box-sizing: border-box;
    border: none;
    background-color: #263238;
    border-radius: 12px;
    color: white;
}

.buttonSubmit:hover {
    background-color: #49599a;
}

</style>
```

## Exemplo 3: DHT11



Exemplo de ligações do ESP-32 Wroom ao DHT11 no pin digital 14  
<https://microcontrollerslab.com/micropython-dht11-dht22-web-server-esp32-esp8266/>



Como obter o Desenho do ESP-32 em FritZing:

<https://drive.google.com/drive/folders/1q7VWBEJq7-LWyp2zIYuVQmyR7q3-Pdx6>

Código fonte:

<https://github.com/labF212/ESP32/blob/master/dht22.py>

```
# Complete project details at https://RandomNerdTutorials.com
```

```
from machine import Pin
```

```
from time import sleep
```

```
import dht
```

```
#sensor = dht.DHT22(Pin(14))
```

```
sensor = dht.DHT11(Pin(14))
```

```
while True:
```

```
    try:
```

```
        sleep(2)
```

```
        sensor.measure()
```

```
        temp = sensor.temperature()
```

```
        hum = sensor.humidity()
```

```
        temp_f = temp * (9/5) + 32.0
```

```
        print('Temperature: %3.1f C' %temp)
```

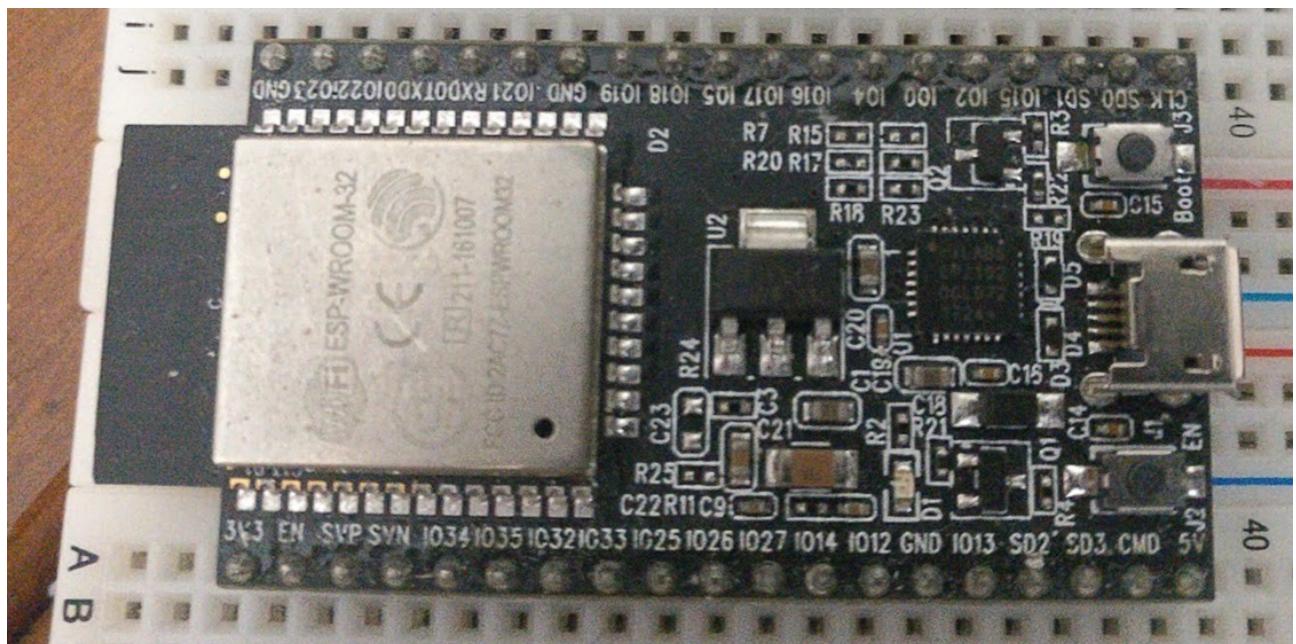
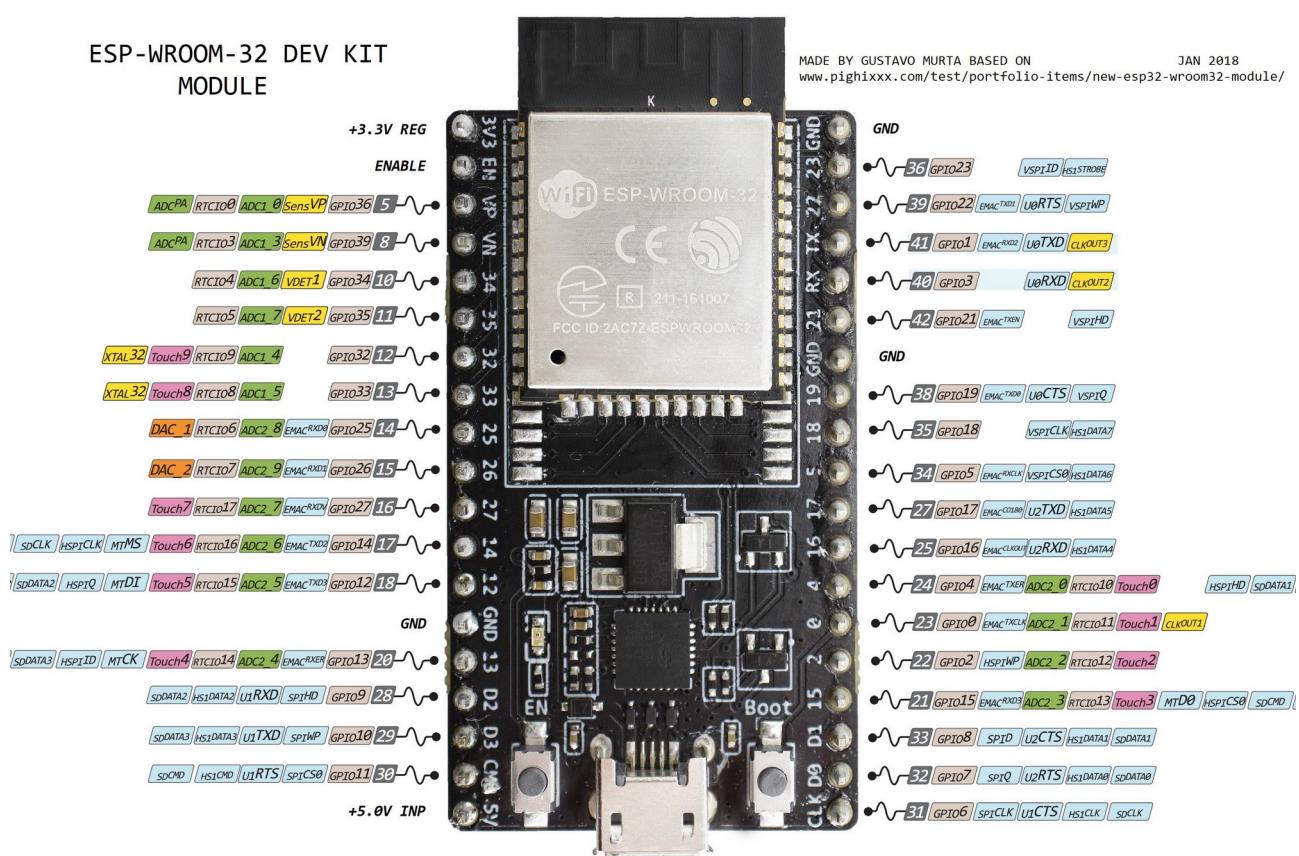
```
        print('Temperature: %3.1f F' %temp_f)
```

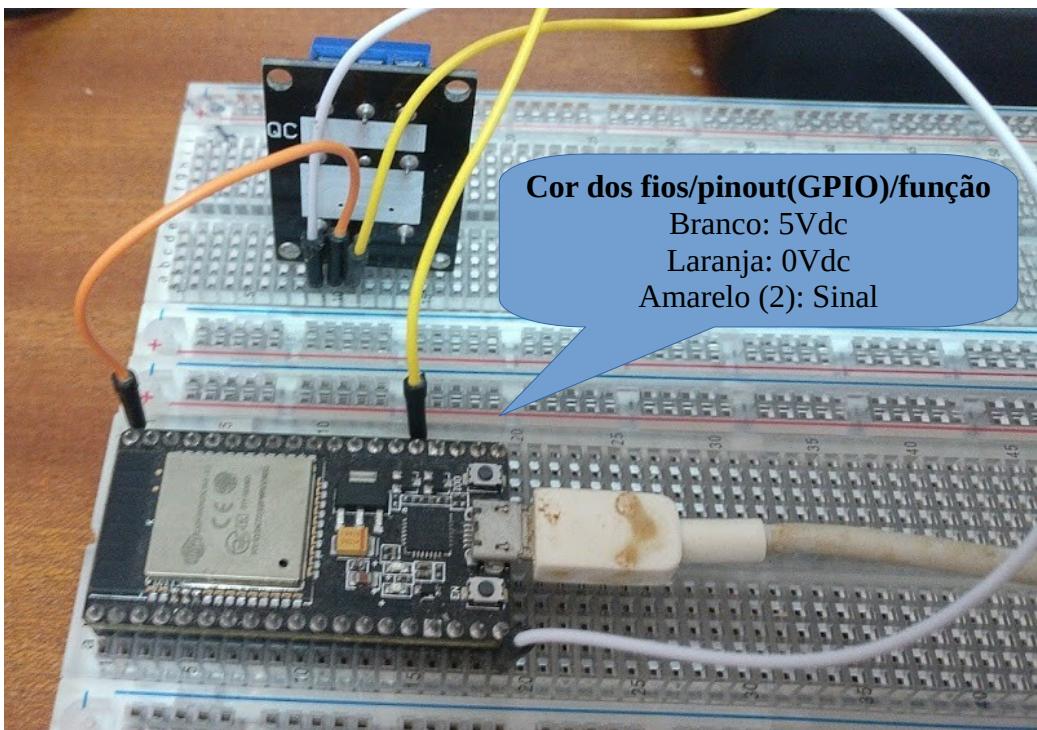
```
        print('Humidity: %3.1f %%' %hum)
```

```
    except OSError as e:
```

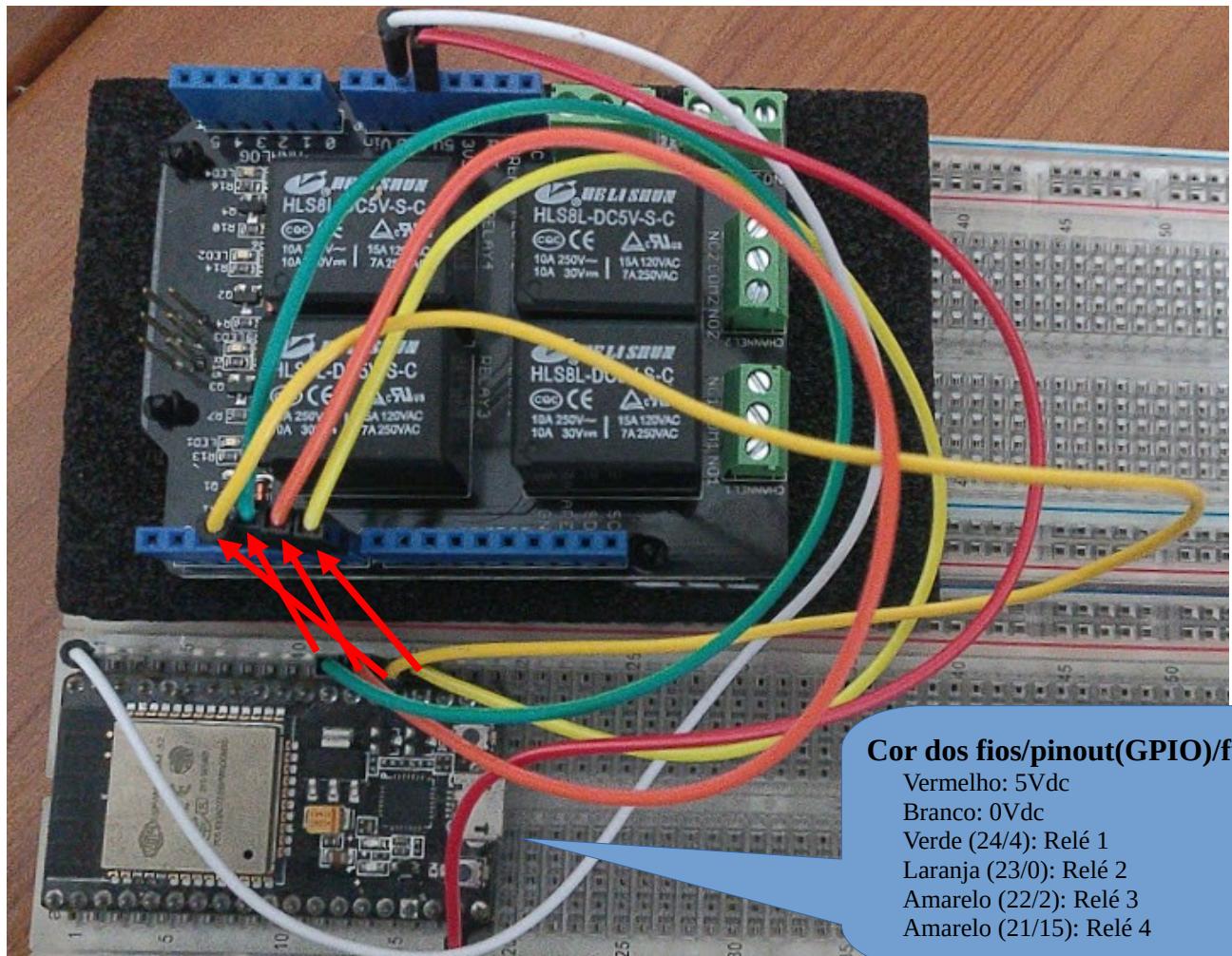
```
        print('Failed to read sensor.')
```

## Exemplo 4: Comando de 4 relés





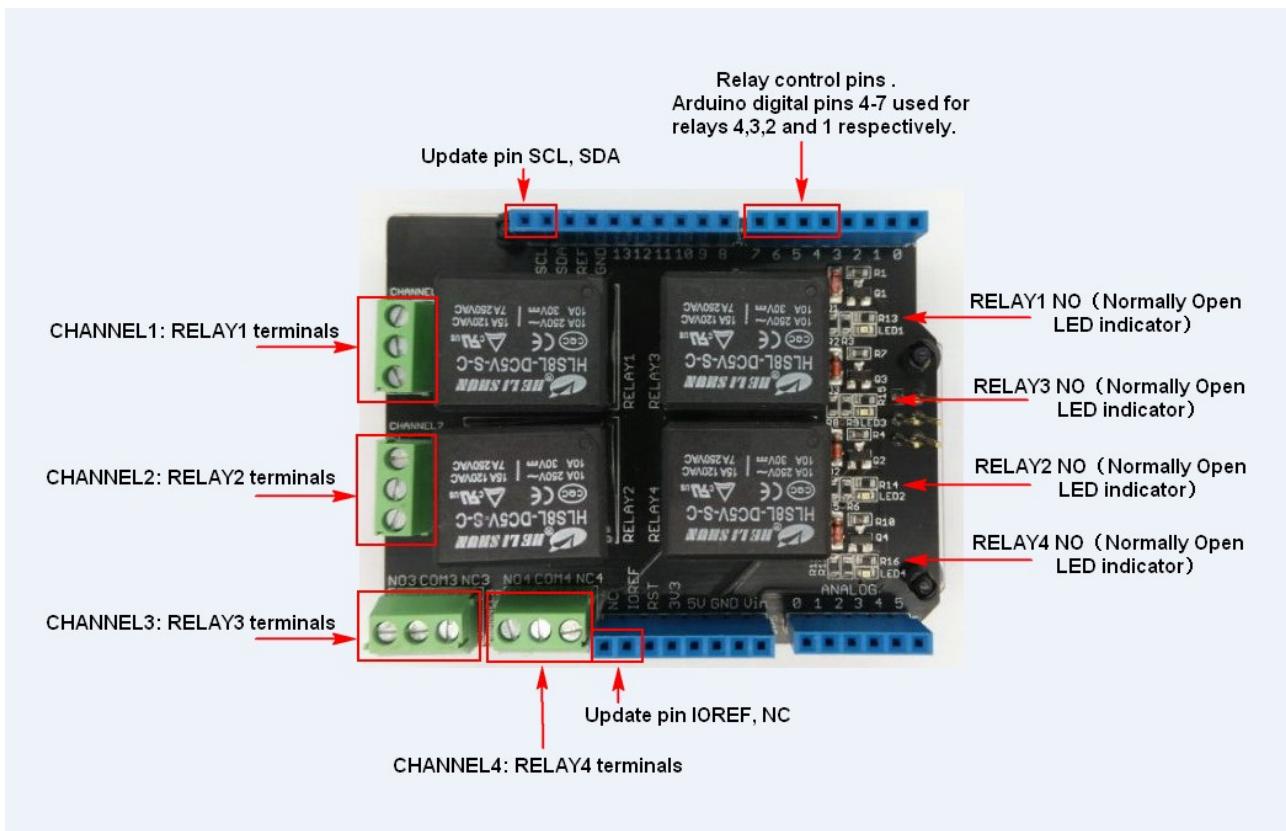
### Exemplo de ligação a 1 relé



### Exemplo de ligação a 4 relés

## pinout e ligações:

[https://wiki.seeedstudio.com/Relay\\_Shield\\_v3/](https://wiki.seeedstudio.com/Relay_Shield_v3/)



```
import time
from machine import Pin
led1=Pin(15,Pin.OUT)
led2=Pin(2,Pin.OUT) #default blue led
led3=Pin(0,Pin.OUT)
led4=Pin(4,Pin.OUT)
```

while True:

```
    led1.value(1)
    time.sleep(2)
    led1.value(0)
    time.sleep(1)
    led2.value(1)
    time.sleep(2)
    led2.value(0)
    time.sleep(1)
    led3.value(1)
    time.sleep(2)
    led3.value(0)
    time.sleep(1)
    led4.value(1)
    time.sleep(2)
    led4.value(0)
```

```
time.sleep(1)
```

## Exemplo 5: Web server on/off

<https://randomnerdtutorials.com/esp32-esp8266-micropython-web-server/>

boot.py

```
# https://raw.githubusercontent.com/RuiSantosdotme/Random-Nerd-Tutorials/
master/Projects/ESP-MicroPython/esp_web_server_boot.py

try:
    import usocket as socket
except:
    import socket

from machine import Pin
import network

import esp
esp.osdebug(None)

import gc
gc.collect()

ssid = 'REPLACE_WITH_YOUR_SSID'
password = 'REPLACE_WITH_YOUR_PASSWORD'

station = network.WLAN(network.STA_IF)

station.active(True)
station.connect(ssid, password)

while station.isconnected() == False:
    pass

print('Connection successful')
print(station.ifconfig())

led = Pin(2, Pin.OUT)
```

main.py

```
# Complete project details at
https://raw.githubusercontent.com/RuiSantosdotme/Random-Nerd-Tutorials/master/
Projects/ESP-MicroPython/esp_web_server_main.py

def web_page():
    if led.value() == 1:
        gpio_state="ON"
    else:
        gpio_state="OFF"

    html = """<html><head> <title>ESP Web Server</title> <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="icon" href="data:,> <style>html{font-family: Helvetica;
display:inline-block; margin: 0px auto; text-align: center;}</style>
    </head> <body> <h1>ESP Web Server</h1> <p>GPIO State: <strong>{}</strong>
        <button>Toggle LED</button>
    </body> </html>""".format(gpio_state)

    return html
```

```

h1{color: #0F3376; padding: 2vh;}p{font-size: 1.5rem;}.button{display: inline-block; background-color: #e7bd3b; border: none; border-radius: 4px; color: white; padding: 16px 40px; text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}
.button2{background-color: #4286f4;}</style></head><body> <h1>ESP Web Server</h1>
<p>GPIO state: <strong>"" + gpio_state + ""</strong></p><p><a href="/?led=on"><button class="button">ON</button></a></p>
<p><a href="/?led=off"><button class="button2">OFF</button></a></p></body></html>"""
return html

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)

while True:
    conn, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = conn.recv(1024)
    request = str(request)
    print('Content = %s' % request)
    led_on = request.find('/?led=on')
    led_off = request.find('/?led=off')
    if led_on == 6:
        print('LED ON')
        led.value(1)
    if led_off == 6:
        print('LED OFF')
        led.value(0)
    response = web_page()
    conn.send('HTTP/1.1 200 OK\n')
    conn.send('Content-Type: text/html\n')
    conn.send('Connection: close\n\n')
    conn.sendall(response)
    conn.close()

```

```
# Complete project details at https://RandomNerdTutorials.com
#https://randomnerdtutorials.com/micropython-esp32-esp8266-dht11-dht22-web-server/

try:
    import usocket as socket
except:
    import socket

import network
from machine import Pin
import dht

import esp
esp.osdebug(None)

import gc
gc.collect()

ssid = 'EACI'
password = 'SMD@EACI'

station = network.WLAN(network.STA_IF)

station.active(True)
station.connect(ssid, password)

while station.isconnected() == False:
    pass

print('Connection successful')
print(station.ifconfig())

#sensor = dht.DHT22(Pin(14))
sensor = dht.DHT11(Pin(14))

def read_sensor():
    global temp, hum
    temp = hum = 0
    try:
        sensor.measure()
        temp = sensor.temperature()
        hum = sensor.humidity()
```

```

if (isinstance(temp, float) and isinstance(hum, float)) or (isinstance(temp, int) and
isinstance(hum, int)):
    msg = (b'{0:3.1f},{1:3.1f}'.format(temp, hum))

    # uncomment for Fahrenheit
    #temp = temp * (9/5) + 32.0

    hum = round(hum, 2)
    return(msg)
else:
    return('Invalid sensor readings.')
except OSError as e:
    return('Failed to read sensor.')

def web_page():
    html = """<!DOCTYPE HTML><html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-
fnmOCqbTlWlj8LyTjo7mOUSTjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
<style>
html {
    font-family: Arial;
    display: inline-block;
    margin: 0px auto;
    text-align: center;
}
h2 { font-size: 3.0rem; }
p { font-size: 3.0rem; }
.units { font-size: 1.2rem; }
.dht-labels{
    font-size: 1.5rem;
    vertical-align:middle;
    padding-bottom: 15px;
}
</style>
</head>
<body>
<h2>ESP DHT Server</h2>
<p>
    <i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
    <span class="dht-labels">Temperature</span>
    <span>"""+str(temp)+"""</span>
    <sup class="units">°C</sup>

```

```
</p>
<p>
<i class="fas fa-tint" style="color:#00add6;"></i>
<span class="dht-labels">Humidity</span>
<span>"""+str(hum)+"+"</span>
<sup class="units">%</sup>
</p>
</body>
</html>"""
return html
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("", 80))
s.listen(5)
```

```
while True:
    conn, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = conn.recv(1024)
    print('Content = %s' % str(request))
    sensor_readings = read_sensor()
    print(sensor_readings)
    response = web_page()
    conn.send('HTTP/1.1 200 OK\n')
    conn.send('Content-Type: text/html\n')
    conn.send('Connection: close\n\n')
    conn.sendall(response)
    conn.close()
```

## 7. Transmitir temperatura e umidade para o Thingspeak

<https://blog.miguelgrinberg.com/post/micropython-and-the-internet-of-things-part-v-temperature-and-humidity>

### main.py

```
import dht
import machine
import network
import sys
import time
import urequests

import config

def connect_wifi():
    ap_if = network.WLAN(network.AP_IF)
    ap_if.active(False)
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print('Connecting to WiFi...')
        sta_if.active(True)
        sta_if.connect(config.WIFI_SSID, config.WIFI_PASSWORD)
        while not sta_if.isconnected():
            time.sleep(1)
    print('Network config:', sta_if.ifconfig())

def show_error():
    led = machine.Pin(config.LED_PIN, machine.Pin.OUT)
    led2 = machine.Pin(config.LED2_PIN, machine.Pin.OUT)
    for i in range(3):
        led.on()
        led2.off()
        time.sleep(0.5)
        led.off()
        led2.on()
        time.sleep(0.5)
    led.on()
    led2.on()

def is_debug():
    debug = machine.Pin(config.DEBUG_PIN, machine.Pin.IN, machine.Pin.PULL_UP)
    if debug.value() == 0:
        print('Debug mode detected.')
        return True
    return False

def get_temperature_and_humidity():
    dht22 = dht.DHT22(machine.Pin(config.DHT22_PIN))
    dht22.measure()
    temperature = dht22.temperature()
    if config.FAHRENHEIT:
        temperature = temperature * 9 / 5 + 32
    return temperature, dht22.humidity()

def log_data(temperature, humidity):
    print('Invoking log webhook')
    url = config.WEBHOOK_URL.format(temperature=temperature,
```

```

        humidity=humidity)
response = urequests.get(url)
if response.status_code < 400:
    print('Webhook invoked')
else:
    print('Webhook failed')
    raise RuntimeError('Webhook failed')

def run():
    try:
        connect_wifi()
        temperature, humidity = get_temperature_and_humidity()
        log_data(temperature, humidity)
    except Exception as exc:
        sys.print_exception(exc)
        show_error()

    if not is_debug():
        machine.deepsleep()

run()

```

## ***config.py***

```

WIFI_SSID = 'your SSID'
WIFI_PASSWORD = 'your Wi-Fi password'
LED_PIN = 2 # D4
DEBUG_PIN = 14 # D5
DHT22_PIN = 4 # D2
FAHRENHEIT = False
WEBHOOK_URL = 'https://api.thingspeak.com/update?
api_key=IQBL4LHQIQ84YH57&field1={temperature}&field2={humidity}'
LOG_INTERVAL = 60

```

## **6. node red mqqt**

<https://randomnerdtutorials.com/micropython-mqtt-publish-bme680-esp32-esp8266/>

### 3 Programação através de Node-Red (rascunho)

O Node-Red é uma forma de programação por blocos em JavaScript. Necessita de ter instalado o node.js anteriormente.

Como instalar no ubuntu 18.03: <https://skjoldtech.wordpress.com/2019/01/29/install-node-red-on-ubuntu-server-18-04-1-lts/>

(ver como proteger o **node-red** com **password**).

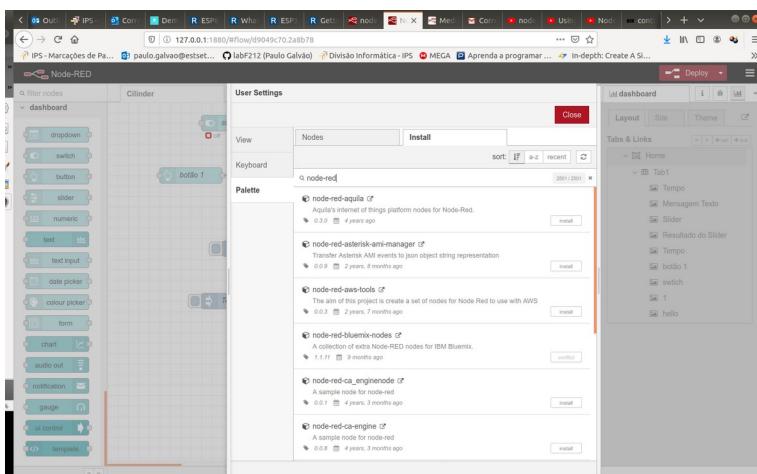
Como instalar no Windows10:

<https://nodered.org/docs/getting-started/local#running>

Dashboard (blocos para páginas web, gráficos):

<https://randomnerdtutorials.com/getting-started-with-node-red-dashboard/>

`npm install node-red-dashboard` (ou via interface)



Instalar ligação blynk node-red:

<https://www.npmjs.com/package/node-red-contrib-blynk-ws>

A screenshot of the Node-RED 'Manage palette' interface. It shows a list of installed nodes under the 'Nodes' tab. Two nodes are listed: 'node-red-contrib-blynk' and 'node-red-contrib-blynk-websockets'. Both nodes have a status of 'installed'. The 'Install' tab is also visible. A search bar at the bottom is set to 'blynk'. A 'Done' button is in the top right corner.



<https://www.curtocircuito.com.br/blog/esp32-node-red-editor-de-fluxo-on-line/>  
usando mqtt...

Home

**Controle de Componentes**

**ON**

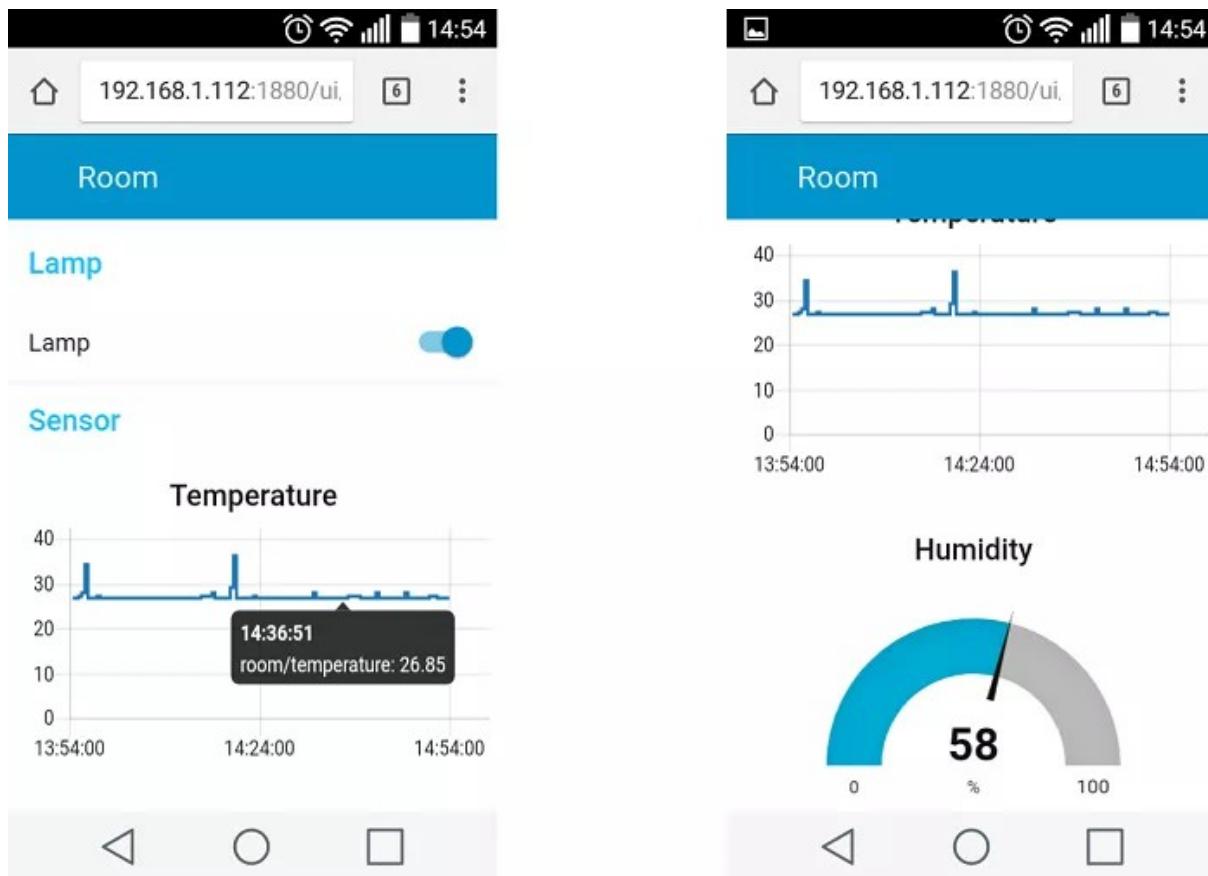
**OFF**

**Dados Meteorológicos**

Temperatura  
27.20

Umidade  
59.50

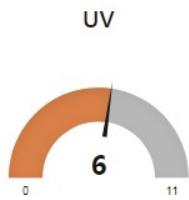
<https://randomnerdtutorials.com/esp8266-and-node-red-with-mqtt/>  
Raspberry PI e mqtt



<https://www.curtocircuito.com.br/blog/estacao-meteorologica-esp32-node-red/>  
IBM iot

## Curto Circuito : Estação Meteorológica

### Dados Meteorológicos



Temperatura (°C)  
28.23

Umidade (%)  
42.72

Pressão Atmosférica...  
926.37

Altitude (m)  
749.85

Intensidade:  
0

Tensão:  
0.9

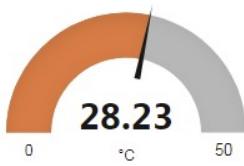
Ultra violeta lvl:  
1123

Contagem ▼

6

### Análise Gráfica: Gauge

#### Temperatura



#### Umidade



▲ LED

### Análise Gráfica: Linhas

#### Pressão



#### Altitude



## 4 Plataformas IOT e processamento de dados (rascunho)

As plataformas de IOT servem para guardar os dados recebidos pelos equipamentos IOT e fazer o seu processamento.

Os dados poderão ficar na nuvem através destas plataformas.

Outra opção é criarmos a nossa nuvem, num servidor próprio (Blynk) ou ainda criarmos nós uma base de dados com os dados recolhidos, para análise posterior por exemplo, numa folha de cálculo ou mesmo fazer uma ligação ao Matlab ou Labview.

### Como instalar Blynk Server

O Blynk Server funciona na rede Eduroam.

Blynk Server:

<https://docs.blynk.cc/>

<https://github.com/blynkkk/blynk-server#blynk-server>

Blynk Server Python (com exemplos):

<https://github.com/blynkkk/lib-python>

Correr o servidor:  
java -jar server-0.41.8.jar -dataFolder /path  
java -jar server-0.41.8.jar -dataFolder  
/home/zorro/Transferências

Url for opening admin page. Must start from "/". For "/admin" url path will look like that "<https://127.0.0.1:9443/admin>".

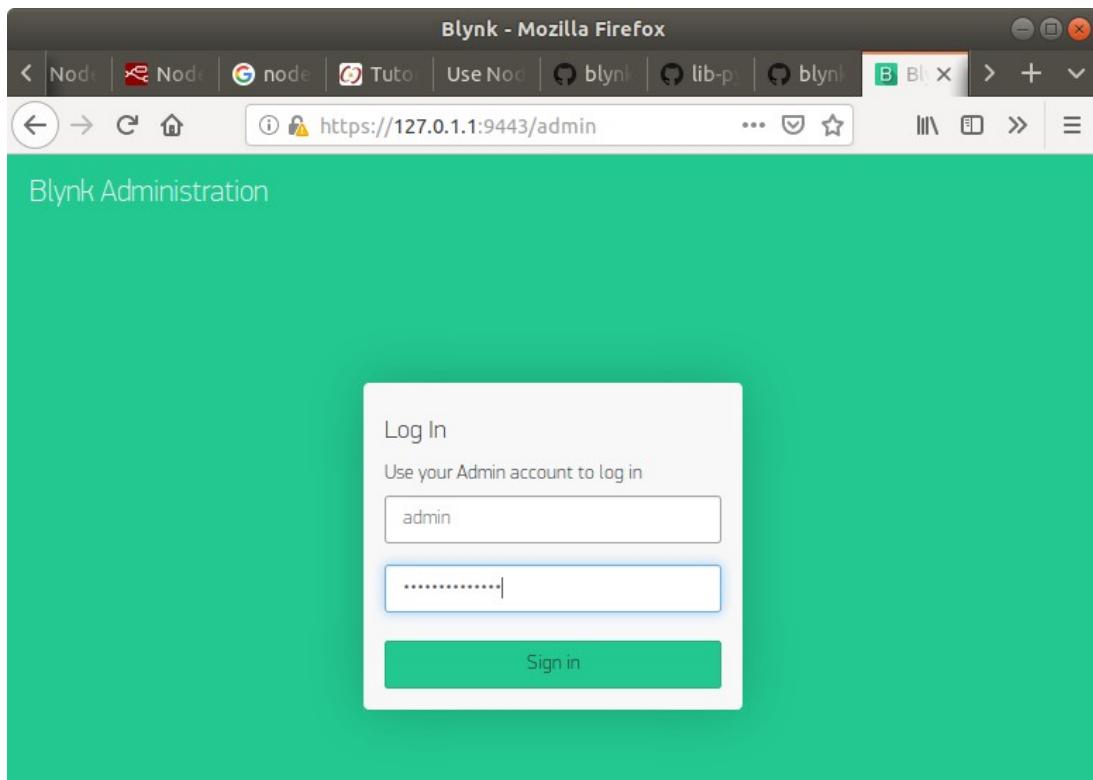
admin.rootPath=/admin

- Comma separated list of administrator IPs. Allow access to admin UI only for those IPs. You may set it for 0.0.0.0/0 to allow access for all. You may use CIDR notation. For instance, 192.168.0.53/24.

allowed.administrator.ips=0.0.0.0/0

- Default admin name and password. Will be created on initial server start

admin.email=admin@blynk.cc  
admin.pass=admin



This screenshot shows the 'Edit user "pauloatcasa@gmail.com"' page. On the left, there's a sidebar with categories: 'Users', 'Stats', 'Hardware Info' (selected), 'Blynk library versions', 'CPU types', 'Hardware boards', 'Connection types', and 'Config'. The main area shows a form with fields: 'Email' (pauloatcasa@gmail.com), 'Name' (pauloatcasa@gmail.com), 'Pass' (\*\*\*\*\*), 'LastModifiedTs' (156395530366), 'Energy' (100000), 'AppName' (Blynk), and 'Region' (local). Below the form is a section for 'LastLoggedIP' and 'Profile Dashboards' with a 'Add new profile dashboards' button. At the bottom is a 'Save Changes' button.

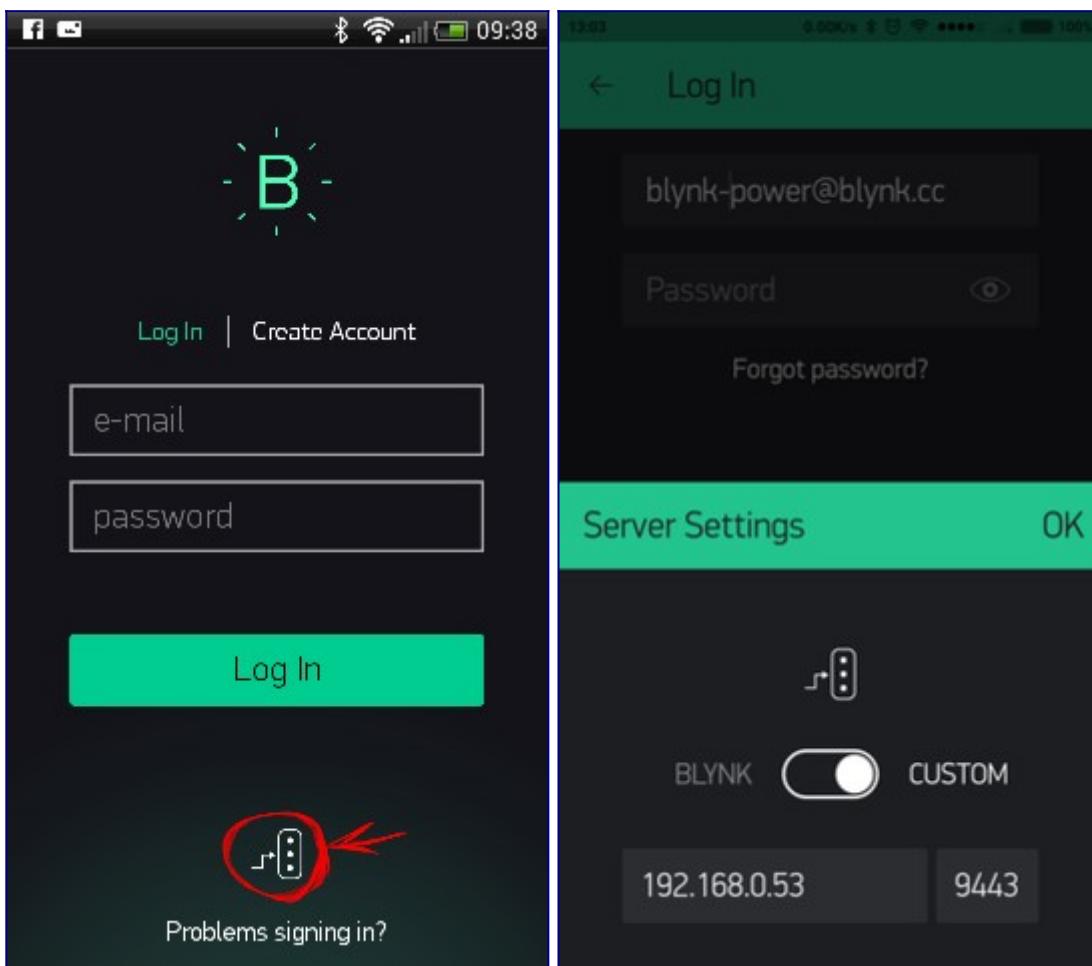
This screenshot shows the 'Edit configuration "mail.properties"' page. The sidebar on the left includes 'Realtime', 'Request per user', 'Messages', 'Board types', 'Login types', 'Widgets', and 'Projects per user'. The main area shows a configuration file with the following content:

```
mail.smtp.username=pauloatcasa@gmail.com
mail.smtp.port=587
mail.smtp.password=
mail.smtp.auth=true
mail.smtp.starttls.enable=true
mail.smtp.timeout=120000
mail.smtp.host=smtp.gmail.com
```

A 'Save Changes' button is located at the bottom of the configuration editor.

## App and sketch changes

- Specify custom server path in your application



- Change your ethernet sketch from:

```
Blynk.begin(auth);  
to  
Blynk.begin(auth, "your_host", 8080);  
or to  
Blynk.begin(auth, IPAddress(xxx,xxx,xxx,xxx), 8080);
```

- Change your WIFI sketch from:

```
Blynk.begin(auth, SSID, pass));  
to  
Blynk.begin(auth, SSID, pass, "your_host", 8080);  
or to  
Blynk.begin(auth, SSID, pass, IPAddress(XXX,XXX,XXX,XXX), 8080);
```

# ESP8266

## Boards Manager

This is the suggested installation method for end users.

### Prerequisites

- Arduino 1.6.8, get it from [Arduino website](#).
- Internet connection

### Instructions

- Start Arduino and open Preferences window.
- Enter  
`http://arduino.esp8266.com/stable/package_esp8266com_index.json`  
into *Additional Board Manager URLs* field. You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and find *esp8266* platform.
- Select the version you need from a drop-down box.
- Click *install* button.
- Don't forget to select your ESP8266 board from Tools > Board menu after installation.

You may optionally use *staging* boards manager package link:

`http://arduino.esp8266.com/staging/package_esp8266com_index.json`.

This may contain some new features, but at the same time, some things might be broken.

## Using git version

This is the suggested installation method for contributors and library developers.

### Prerequisites

- Arduino 1.6.8 (or newer, if you know what you are doing)
- git
- python 2.7
- terminal, console, or command prompt (depending on your OS)
- Internet connection

### Instructions

- Open the console and go to Arduino directory. This can be either your *sketchbook* directory (usually `<Documents>/Arduino`), or the directory of Arduino application itself, the choice is up to you.
- Clone this repository into `hardware/esp8266com/esp8266` directory. Alternatively, clone it elsewhere and create a symlink, if your OS supports them.  
`cd hardware`

```
mkdir esp8266com  
cd esp8266com  
git clone https://github.com/esp8266/Arduino.git esp8266
```

You should end up with the following directory structure:

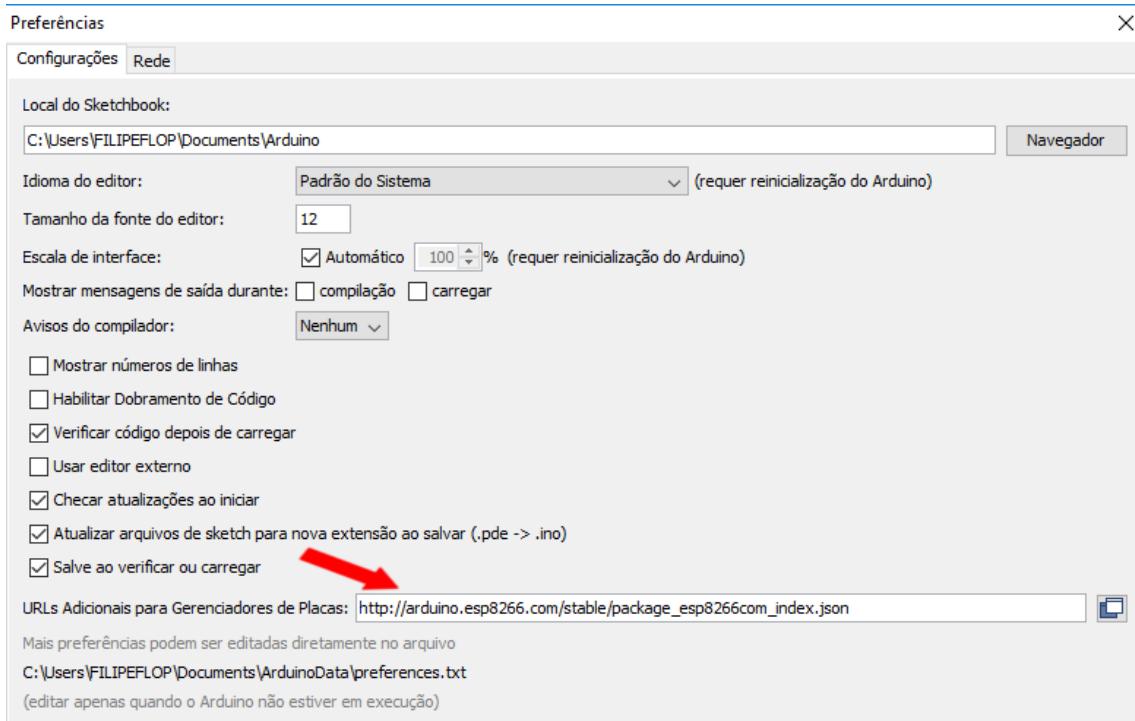
```
Arduino  
|  
--- hardware  
|  
--- esp8266com  
|  
--- esp8266  
|  
--- bootloaders  
--- cores  
--- doc  
--- libraries  
--- package  
--- tests  
--- tools  
--- variants  
--- platform.txt  
--- programmers.txt  
--- README.md  
--- boards.txt  
--- LICENSE
```

- Download binary tools

```
cd esp8266/tools  
python get.py
```

- Restart Arduino

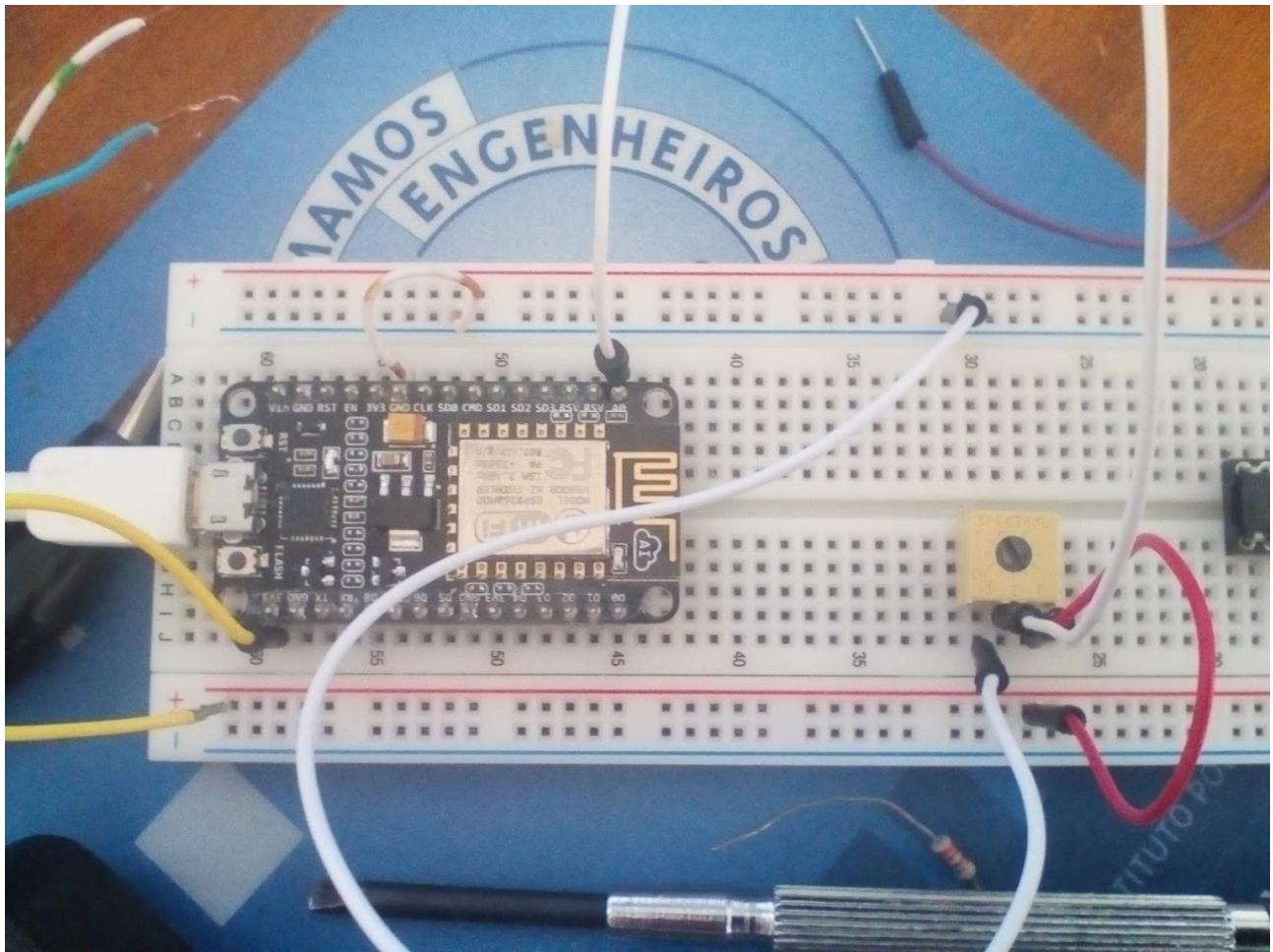
Com a ligação deste deu: <https://www.filipeflop.com/blog/acelerometro-com-esp8266-nodemcu/>



# Exemplos de Programas

## Exemplo 1: Potenciômetro

Atenção o adc provoca um erro de 0,1V



```
float leitura;
```

```
void setup()
{
    Serial.begin(115200);
    Serial.println("Leitura do potenciometro");
    Serial.println("Valor      Volts");
}
```

```
void loop()
{
    Serial.print(analogRead(A0));
    Serial.print("      ");
    leitura = (analogRead(A0)*3.3/1023);
    Serial.println(leitura);
    delay(1500);
}
```



## **Exemplo 2: Geolocalização - <http://ip-api.com/>**

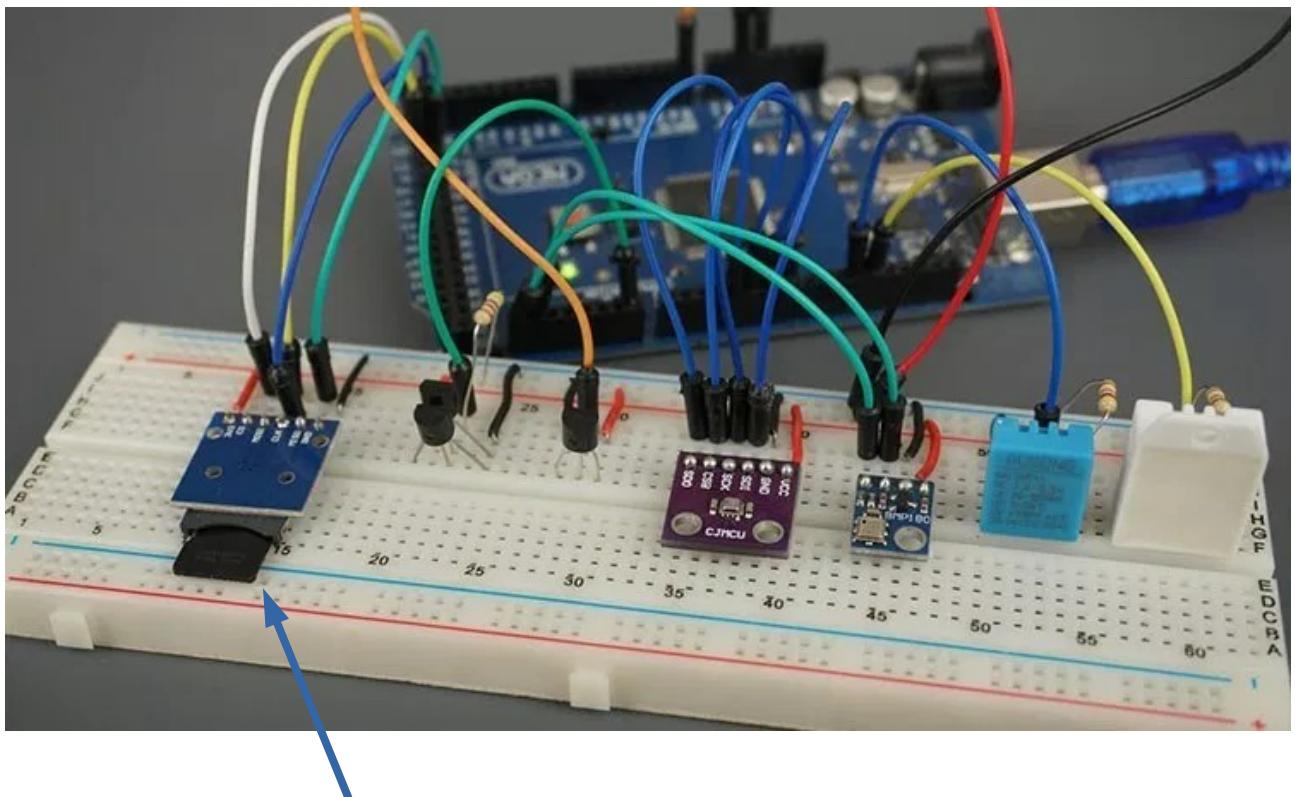
Como usar acelerômetro com ESP8266 NodeMCU - <https://www.filipeflop.com/blog/acelerometro-com-esp8266-nodemcu/>

<https://mjrobot.org/tag/esp8266/> - microphoton e protocolo MQTT

Neste novo tutorial, utilizando-se do protocolo MQTT, enviaremos os dados capturados, à um serviço the IoT, o *ThingSpeak.com* e para um aplicativo móvel, o *Thingsview*.

<https://randomnerdtutorials.com/esp8266-adc-reading-analog-values-with-nodemcu/>

## **Exemplo 3: Gravação de dados no local em cartão microSD**



```

/*
 * Rui Santos
 * Complete Project Details https://randomnerdtutorials.com/dht11-vs-dht22-vs-lm35-vs-
ds18b20-vs-bme280-vs-bmp180/
 */

#include <SD.h> // for the SD card

const int LM35sensorPin = A0;
float LM35sensorValue;
float LM35voltageOut;
float LM35temperature;

const int chipSelectSDCard = 53;
File myFile;

void setup() {
    Serial.begin(9600);

    pinMode(LM35sensorPin, INPUT);

    if(!SD.begin(chipSelectSDCard)) {
        Serial.println("SD card initialization failed!");
        return;
    }
    Serial.println("SD card initialization done.");

    myFile=SD.open("DATA.txt", FILE_WRITE);
    if (myFile) {
        Serial.println("File opened ok");
        // print the headings for our data
        myFile.println("LM35");
    }
    myFile.close();
}

void loop() {

    //LM35 SENSOR
    LM35sensorValue = analogRead(LM35sensorPin);
    LM35voltageOut = (LM35sensorValue * 5000) / 1024;

    // calculate temperature for LM35 (LM35DZ)
    LM35temperature = LM35voltageOut / 10;

    Serial.print("Temperature LM35(°C): ");
    Serial.println(LM35temperature);
}

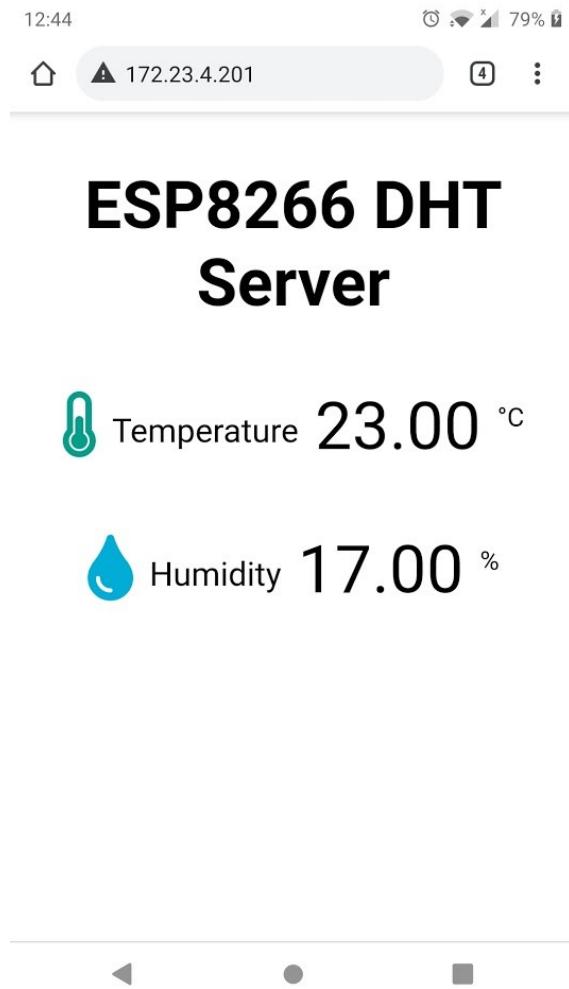
```

```
Serial.println("");

myFile = SD.open("DATA.txt", FILE_WRITE);
if (myFile) {
    Serial.println("File open with success");
    myFile.print(LM35temperature);
    myFile.println(",");
}
myFile.close();

delay(6000);
}
```

## Exemplo 4: DHT11, NTPClient, WebServer (Lab212)



Exemplo A:

```
*****
Rui Santos
Complete project details at https://randomnerdtutorials.com/esp8266-dht11dht22-temperature-and-
humidity-web-server-with-arduino-ide/
*****
```

```
// Import required libraries
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <Hash.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>

// Replace with your network credentials
const char* ssid = "EACI";
```

```

const char* password = "SMD@EACI";

#define DHTPIN 5 // Digital pin connected to the DHT sensor

// Uncomment the type of sensor in use:
#define DHTTYPE DHT11 // DHT 11
//#define DHTTYPE DHT22 // DHT 22 (AM2302)
//#define DHTTYPE DHT21 // DHT 21 (AM2301)

DHT dht(DHTPIN, DHTTYPE);

// current temperature & humidity, updated in loop()
float t = 0.0;
float h = 0.0;

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0; // will store last time DHT was updated

// Updates DHT readings every 10 seconds
const long interval = 10000;

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-
fnmOCqbTlWIj8LyTjo7mOUSTjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
<style>
  html {
    font-family: Arial;
    display: inline-block;
    margin: 0px auto;
    text-align: center;
  }
  h2 { font-size: 3.0rem; }
  p { font-size: 3.0rem; }
  .units { font-size: 1.2rem; }
  .dht-labels{
    font-size: 1.5rem;
    vertical-align:middle;
    padding-bottom: 15px;
  }
</style>
</head>
<body>
  <h2>ESP8266 DHT Server</h2>

```

```

<p>
  <i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
  <span class="dht-labels">Temperature</span>
  <span id="temperature">%TEMPERATURE%</span>
  <sup class="units">°C</sup>
</p>
<p>
  <i class="fas fa-tint" style="color:#00add6;"></i>
  <span class="dht-labels">Humidity</span>
  <span id="humidity">%HUMIDITY%</span>
  <sup class="units">%</sup>
</p>
</body>
<script>
setInterval(function () {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("temperature").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/temperature", true);
  xhttp.send();
}, 10000 );

setInterval(function () {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("humidity").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/humidity", true);
  xhttp.send();
}, 10000 );
</script>
</html>)rawliteral";

// Replaces placeholder with DHT values
String processor(const String& var){
  //Serial.println(var);
  if(var == "TEMPERATURE"){
    return String(t);
  }
  else if(var == "HUMIDITY"){
    return String(h);
  }
  return String();
}

void setup(){
  // Serial port for debugging purposes

```

```

Serial.begin(115200);
dht.begin();

// Connect to Wi-Fi
WiFi.begin(ssid, password);
Serial.println("Connecting to WiFi");
while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println(".");
}

// Print ESP8266 Local IP Address
Serial.println(WiFi.localIP());

// Route for root / web page
server.on("/", HTTP_GET, [](){AsyncWebRequest *request){
    request->send_P(200, "text/html", index_html, processor);
});
server.on("/temperature", HTTP_GET, [](){AsyncWebRequest *request){
    request->send_P(200, "text/plain", String(t).c_str());
});
server.on("/humidity", HTTP_GET, [](){AsyncWebRequest *request){
    request->send_P(200, "text/plain", String(h).c_str());
});

// Start server
server.begin();
}

void loop(){
unsigned long currentMillis = millis();
if (currentMillis - previousMillis >= interval) {
    // save the last time you updated the DHT values
    previousMillis = currentMillis;
    // Read temperature as Celsius (the default)
    float newT = dht.readTemperature();
    // Read temperature as Fahrenheit (isFahrenheit = true)
    //float newT = dht.readTemperature(true);
    // if temperature read failed, don't change t value
    if (isnan(newT)) {
        Serial.println("Failed to read from DHT sensor!");
    }
    else {
        t = newT;
        Serial.println(t);
    }
    // Read Humidity
    float newH = dht.readHumidity();
    // if humidity read failed, don't change h value
    if (isnan(newH)) {
        Serial.println("Failed to read from DHT sensor!");
    }
}

```

```
else {
    h = newH;
    Serial.println(h);
}
}
```

Exemplo B:

09:32 23%



▲ 172.22.2.11

3



### Temperatura e Humidade no LabF212



Temperatura 18.00 °C



Humidade 47.00 %



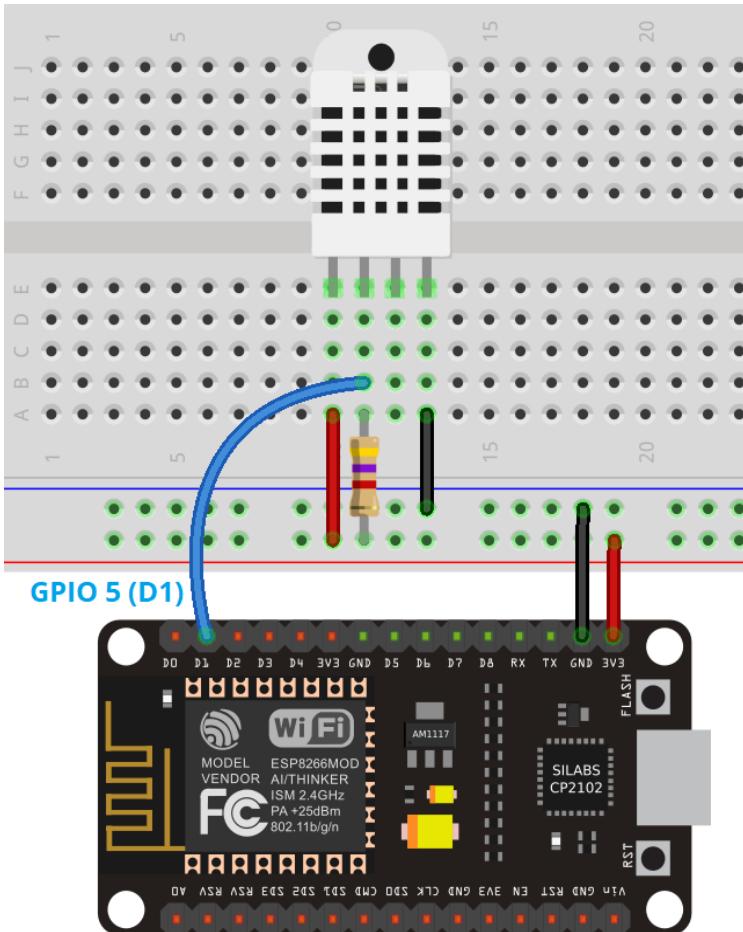
Data 2020-12-21



Hora 10:32:02



## ***Montagem***



**Nota: Não usar resistência !!! Está a ligar a 3,3V**

## Código

```
//Complete project details at https://randomnerdtutorials.com/esp8266-dht11dht22-temperature-and-humidity-web-server-with-arduino-ide/
//Bonecos https://use.fontawesome.com/releases/v5.7.2/css/all.css exemplo: fa-pull-right
```

```
// Import required libraries
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <Hash.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <NTPClient.h>
#include <WiFiUdp.h>

// Define NTP Client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

// Variables to save date and time
String formattedDate;
```

```

String dayStamp;
String timeStamp;

// Replace with your network credentials
const char* ssid = "EACI";
const char* password = "SMD@EACI";

#define DHTPIN 5      // Digital pin connected to the DHT sensor

// Uncomment the type of sensor in use:
#define DHTTYPE    DHT11      // DHT 11
//#define DHTTYPE    DHT22      // DHT 22 (AM2302)
//#define DHTTYPE    DHT21      // DHT 21 (AM2301)

DHT dht(DHTPIN, DHTTYPE);

// current temperature & humidity, updated in loop()
float t = 0.0;
float h = 0.0;

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;      // will store last time DHT was updated

// Updates DHT readings every 10 seconds
const long interval = 10000;

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
  href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
  integrity="sha384-fnmOCqbTlWIlj8LyTjo7m0USTjsKC4p0pQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
  crossorigin="anonymous">
  <style>
    html {
      font-family: Arial;
      display: inline-block;
      margin: 0px auto;
      text-align: center;
    }
    h2 { font-size: 3.0rem; }
    p { font-size: 3.0rem; }
    .units { font-size: 1.2rem; }
    .dht-labels{
      font-size: 1.5rem;
      vertical-align:middle;
      padding-bottom: 15px;
    }
  </style>
</head>
<body>
  <h3>Temperatura e Umidade no LabF212</h3>
  <p>
    <i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
    <span class="dht-labels">Temperatura</span>
    <span id="temperature">%TEMPERATURE%</span>
    <sup class="units">&deg;C</sup>
  </p>
</body>
</html>
)rawliteral";

```

```

<p>
  <i class="fas fa-tint" style="color:#00add6;"></i>
  <span class="dht-labels">Humidade</span>
  <span id="humidity">%HUMIDITY%</span>
  <sup class="units">%</sup>
</p>
<p>
  <i class="fas fa-calendar" style="color:#059e8a;"></i>
  <span class="dht-labels">Data</span>
  <span id="dayStamp">DAYSTAMP</span>
</p>
<p>
  <i class="fas fa-clock" style="color:#00add6;"></i>
  <span class="dht-labels">Hora</span>
  <span id="timeStamp">TIMESTAMP</span>
</p>
</body>
<script>
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("temperature").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/temperature", true);
  xhttp.send();
}, 10000 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("humidity").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/humidity", true);
  xhttp.send();
}, 10000 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("timeStamp").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/timeStamp", true);
  xhttp.send();
}, 10000 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("dayStamp").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/dayStamp", true);
  xhttp.send();
}, 10000 ) ;

</script>
</html>)rawliteral";

```

```

// Replaces placeholder with DHT values
String processor(const String& var){
    //Serial.println(var);
    if(var == "TEMPERATURE"){
        return String(t);
    }
    if(var == "HUMIDITY"){
        return String(h);
    }
    if(var == "TIMESTAMP"){
        return String(timestamp);
    }
    else if(var == "DAYSTAMP"){
        return String(dayStamp);
    }
    return String();
}

void data_hora() {
    while(!timeClient.update()) {
        timeClient.forceUpdate();
    }
    // The formattedDate comes with the following format:
    // 2018-05-28T16:00:13Z
    // We need to extract date and time
    formattedDate = timeClient.getFormattedDate();
    Serial.println(formattedDate);

    // Extract date
    int splitT = formattedDate.indexOf("T");
    dayStamp = formattedDate.substring(0, splitT);
    Serial.print("DATA: ");
    Serial.println(dayStamp);
    // Extract time
    timestamp = formattedDate.substring(splitT+1, formattedDate.length()-1);
    Serial.print("HORA: ");
    Serial.println(timestamp);
    //delay(2000);
}

void setup(){
    // Serial port for debugging purposes
    Serial.begin(115200);
    dht.begin();

    // Connect to Wi-Fi
    WiFi.begin(ssid, password);
    Serial.println("Connecting to WiFi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println(".");
    }

    // Print ESP8266 Local IP Address
    Serial.println(WiFi.localIP());

    // Route for root / web page
    server.on("/", HTTP_GET, [] (AsyncWebRequest *request){
        request->send_P(200, "text/html", index_html, processor);
    });
    server.on("/temperature", HTTP_GET, [] (AsyncWebRequest *request){

```

```

    request->send_P(200, "text/plain", String(t).c_str());
});
server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(h).c_str());
});
server.on("/timeStamp", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(timeStamp).c_str());
});
server.on("/dayStamp", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(dayStamp).c_str());
});

// Start server
server.begin();

// Initialize a NTPClient to get time
timeClient.begin();
// Set offset time in seconds to adjust for your timezone, for example:
// GMT +1 = 3600
// GMT +8 = 28800
// GMT -1 = -3600
// GMT 0 = 0
timeClient.setTimeOffset(3600);
}

void loop(){
    unsigned long currentMillis = millis();

    data_hora();
    if (currentMillis - previousMillis >= interval) {
        // save the last time you updated the DHT values
        previousMillis = currentMillis;
        // Read temperature as Celsius (the default)
        float newT = dht.readTemperature();
        // Read temperature as Fahrenheit (isFahrenheit = true)
        //float newT = dht.readTemperature(true);
        // if temperature read failed, don't change t value
        if (isnan(newT)) {
            Serial.println("Failed to read from DHT sensor!");
        }
        else {
            t = newT;
            Serial.println(t);
        }
        // Read Humidity
        float newH = dht.readHumidity();
        // if humidity read failed, don't change h value
        if (isnan(newH)) {
            Serial.println("Failed to read from DHT sensor!");
        }
        else {
            h = newH;
            Serial.println(h);
        }
    }
    delay(1000);
}

```

## **Bonecos**

<https://use.fontawesome.com/releases/v5.7.2/css/all.css>

exemplo: desenho bateria

```
fa-battery-empty:before{content:"\f244"} .fa-battery-full:before{content:"\f240"} .fa-battery-half:before{content:"\f242"} .fa-battery-quarter:before{content:"\f243"}
```

colocar no código: fa-battery-full

# Referências

- <https://circuits4you.com/2018/02/02/installing-esp32-board-in-arduino-ide-on-ubuntu-linux/>
- <http://blogmasterwalkershop.com.br/embarcados/esp32/nodemcu-32s-esp32-programando-com-a-ide-do-arduino/>
- <https://mjrobot.org/iot-made-simple-playing-with-the-esp32-on-arduino-ide/>
- <https://www.fernandok.com>
- <https://randomnerdtutorials.com/projects-esp32/>
- <https://lastminuteengineers.com/creating-esp32-web-server-arduino-ide/>
- <https://circuits4you.com/2018/02/03/esp8266-nodemcu-adc-analog-value-on-dial-gauge/>
- <https://www.fernandok.com/2018/03/esp32-reconhecimento-de-voz-com-celular.html>
- <https://randomnerdtutorials.com/esp32-servo-motor-web-server-arduino-ide/>
- <https://iotdesignpro.com/projects/iot-home-automation-using-esp32-and-blynk>
- <https://www.curtocircuito.com.br/blog/esp32-node-red-editor-de-fluxo-on-line/>
- <http://www.iotsharing.com/2018/05/how-to-build-iot-dashboard-using-node-red.html>
- <https://www.curtocircuito.com.br/blog/analise-de-sensores-node-red-esp32/>
- [ESP32 Publish Sensor Readings to Google Sheets \(ESP8266 Compatible\)](#)