INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
ALAGOAS

# Internet das coisas

Curso: Técnico em Informática para internet
Projeto Oficinas 4.0
Professor: Leonardo Silva

01/04/2022

# O que é Internet das coisas?

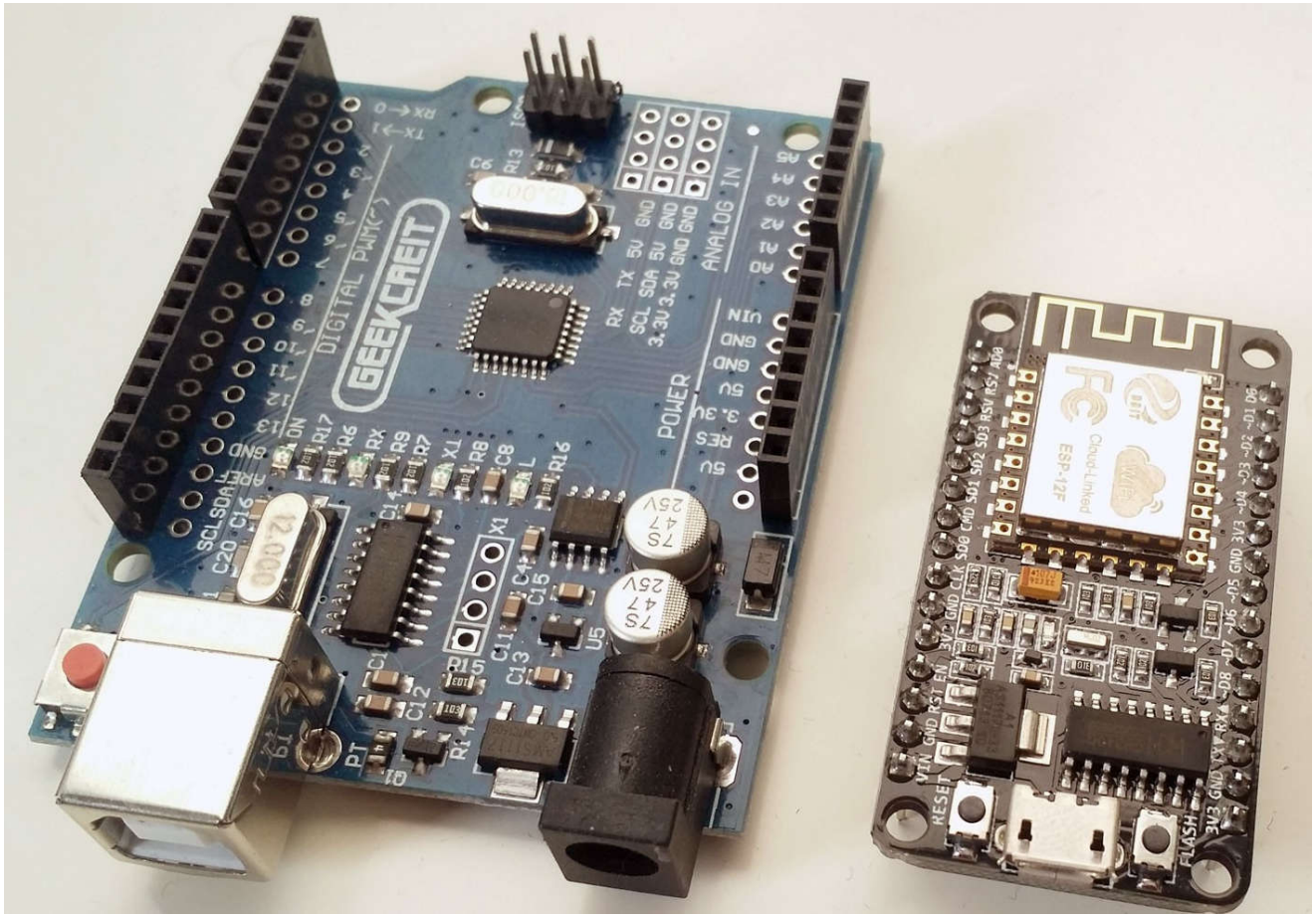
# O que é Internet das coisas?

Vídeo: http://g1.globo.com/como-sera/noticia/2018/04/tecnologia-por-tras-das-casas-inteligentes.html

# Como podemos conectar coisas à internet?

# Módulo NodeMCU

- Com esse módulo é possível desenvolver aplicações baseadas em Arduino que se conectam à internet usando Wi-Fi.



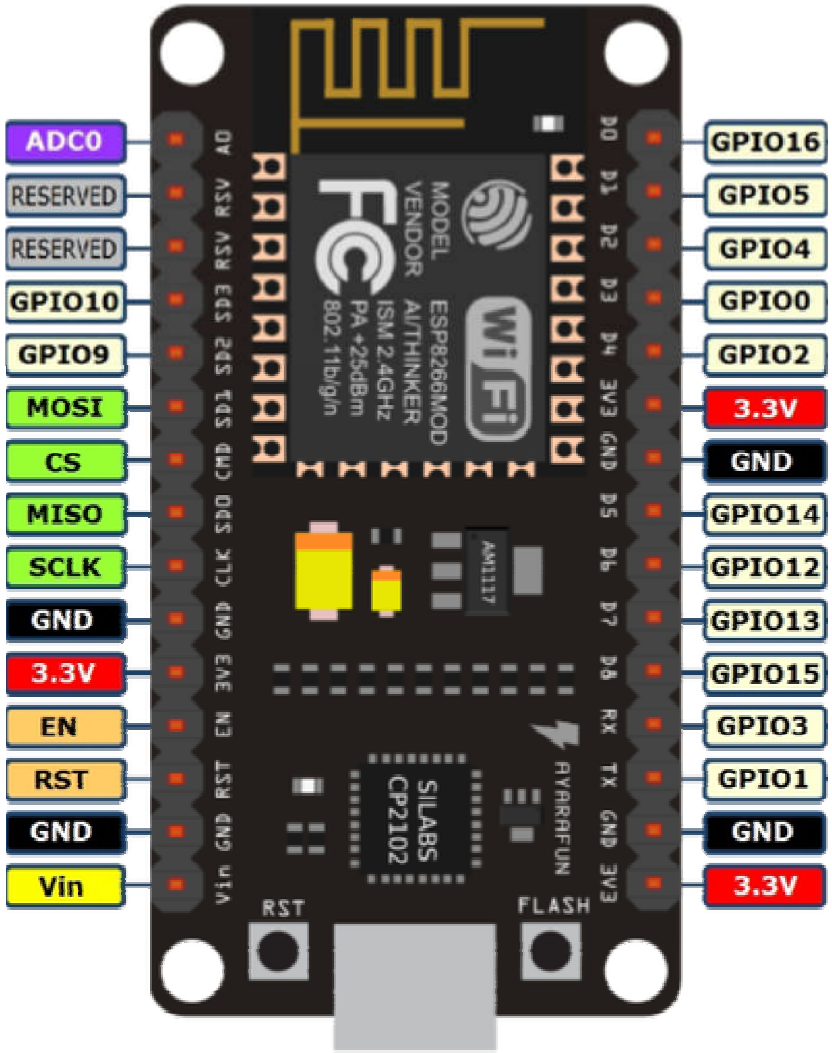

Arduino UNO

Módulo WiFi ESP8266
NodeMcu Amica ESP-12E

# Módulo NodeMCU

- Comparação entre os pinos:

| Parâmetro | Arduino UNO | NodeMCU |
|---|---|---|
| Suporte WiFi | Não | Sim |
| Tensão dos pinos | 5 V | 3,3 V |
| # pinos digitais | 14 | 11 |
| # pinos analógicos | 6 | 1 |
| Cabo USB | Tipo B | micro |




Cabo micro USB

# Módulo NodeMCU

- O que poderemos fazer:
  - Controlar o NodeMCU pela internet por meio de aplicativo.
  - Utilizá-lo como um servidor para disponibilizar páginas WEB.
  - Enviar dados de sensores para um banco de dados na internet.

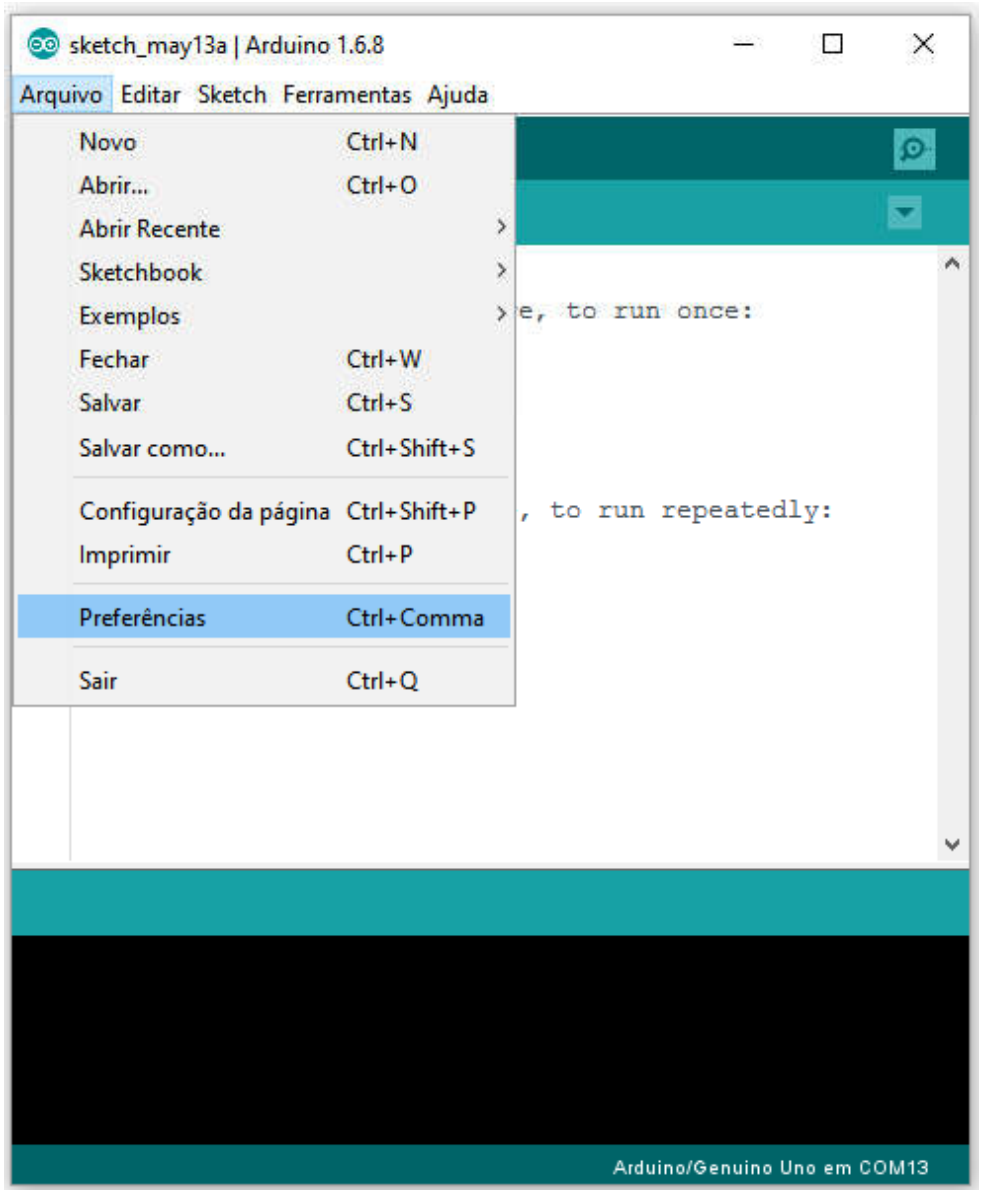
# Módulo NodeMCU

- De maneira geral, a maior parte dos programas disponíveis para o Arduino pode também ser compilados e carregados na plataforma NodeMCU.

- É preciso que a IDE do Arduino possua as bibliotecas para o microcontrolador ESP8266.

# Configuração da IDE do Arduino para o NodeMCU

1-Entrar na IDE do Arduino e clicar em **Arquivo -> Preferências**:



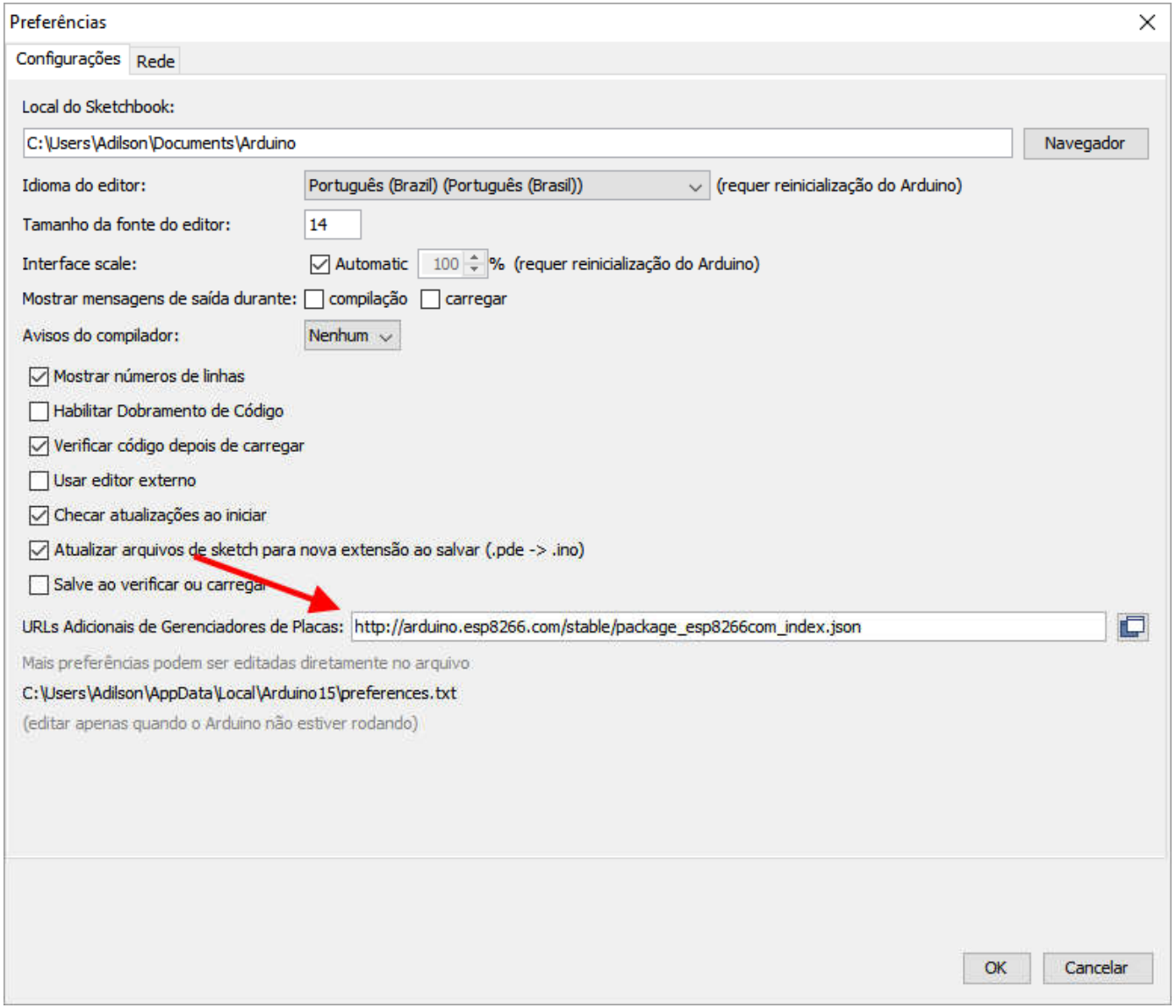
2- Na tela seguinte, digite o link abaixo no campo

**URLs adicionais de Gerenciadores de Placas:**

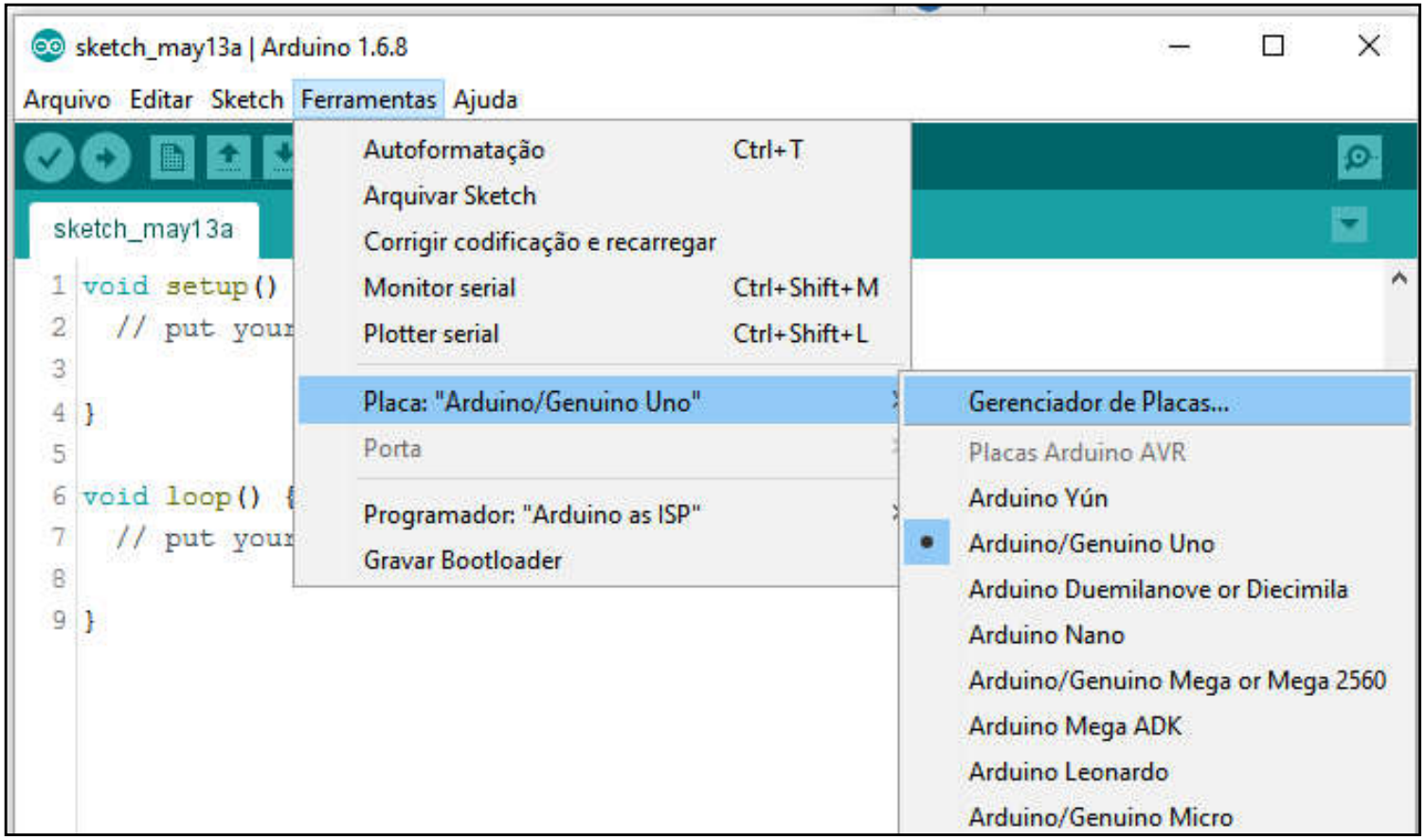
http://arduino.esp8266.com/stable/package_esp8266com_index.json

A sua tela ficará assim:

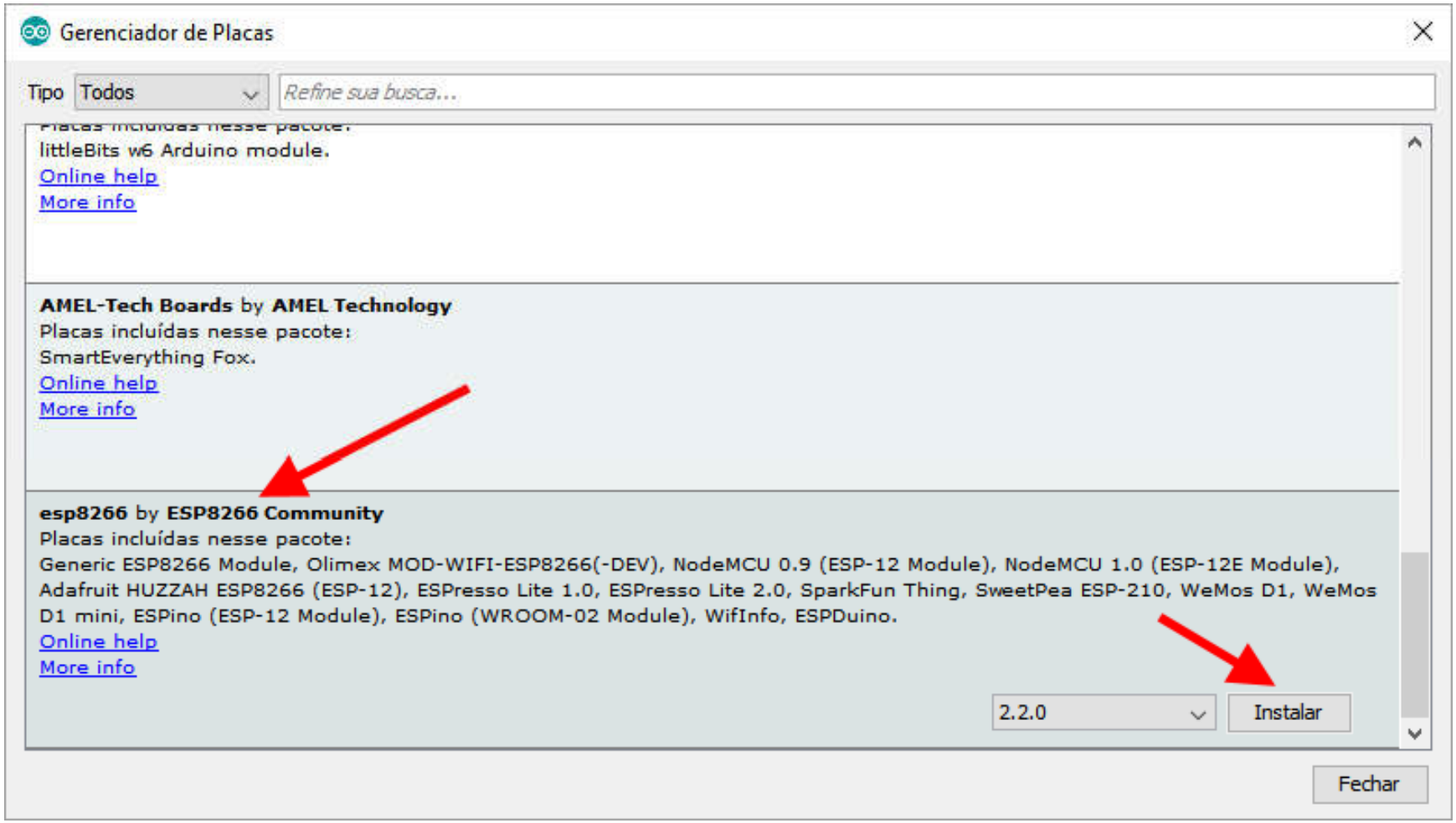
Clique em **OK** para retornar à tela principal



3- Agora clique em **Ferramentas -> Placa -> Gerenciador de Placas:**

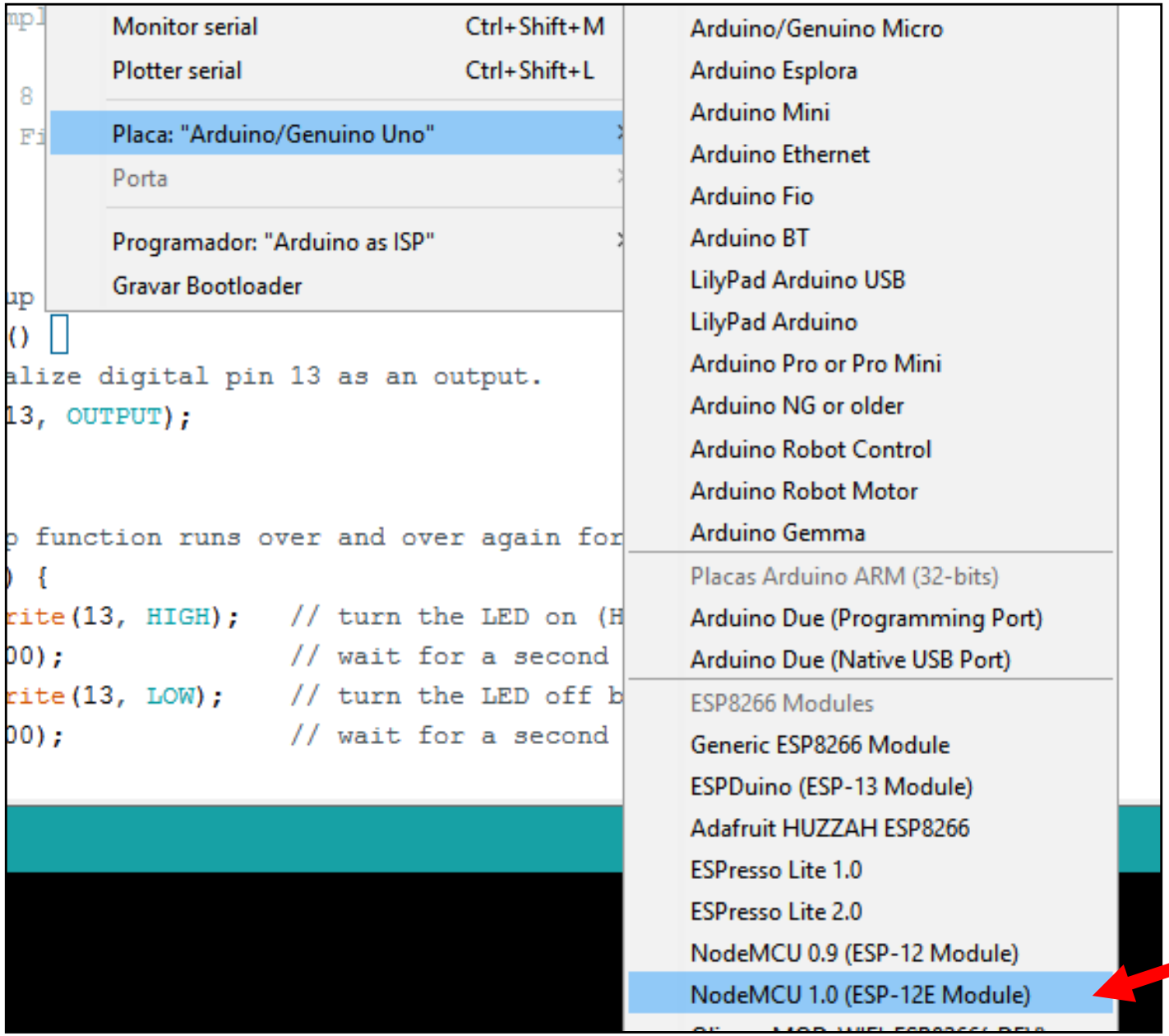


4- Utilize a barra de rolagem para encontrar o **esp8266 by ESP8266 Community** e clique em **INSTALAR**



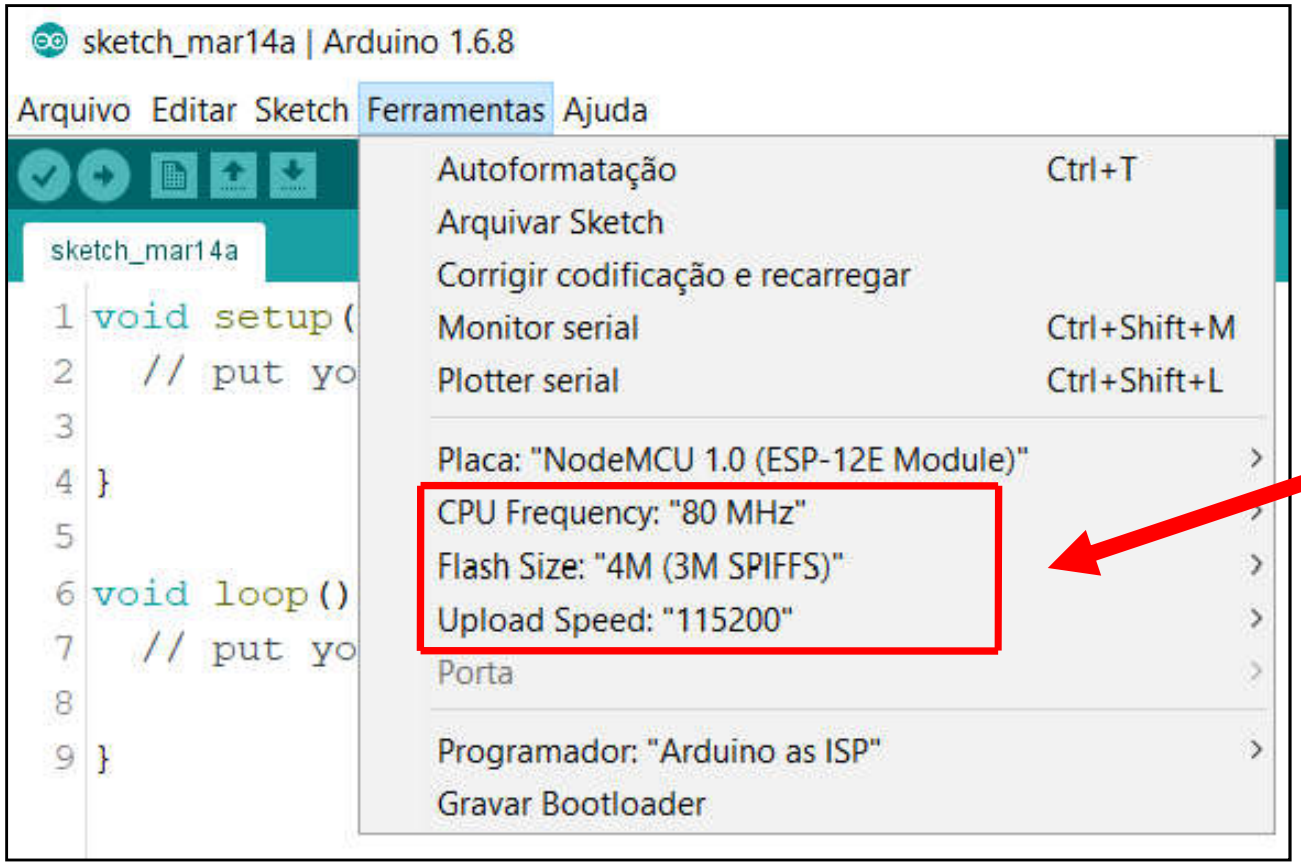

Após a instalação pode fechar a janela.

5- Após alguns minutos as placas da linha ESP8266 já estarão disponíveis na lista de placas da IDE do Arduino.




Selecione essa opção

6- Visualização após seleção da placa:

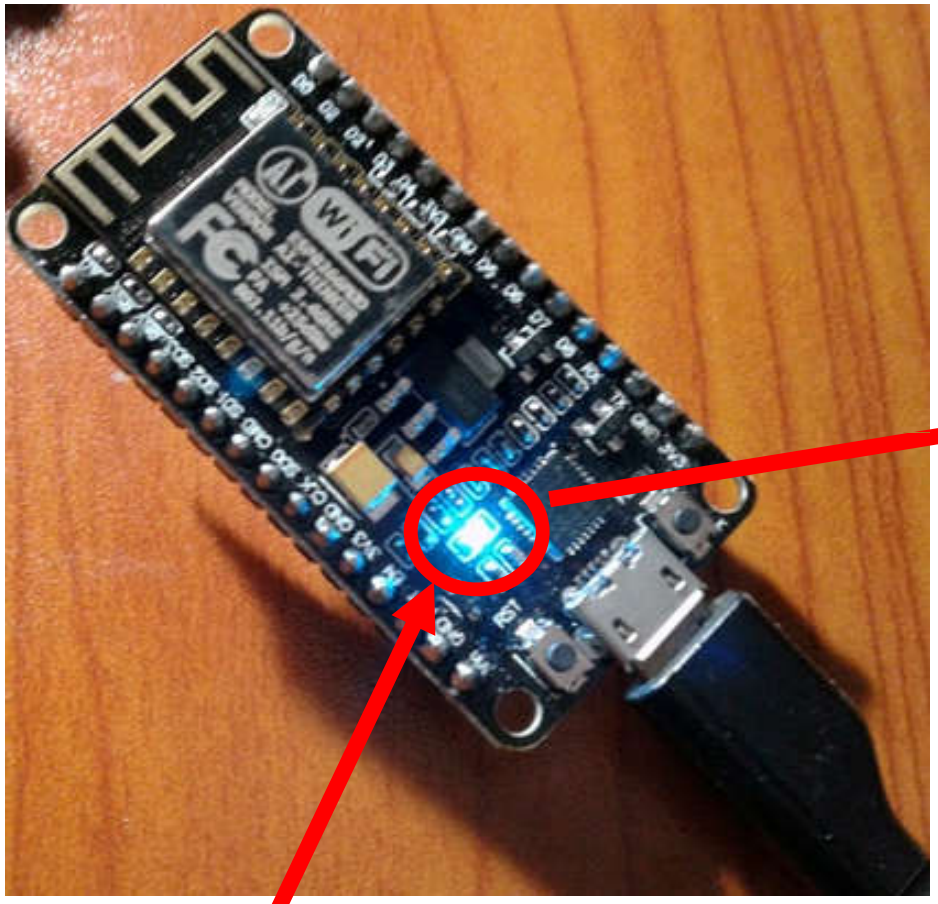



Não altere esses valores

7- Selecione a **Porta** e transfira o código normalmente para o NodeMCU, do mesmo jeito que você faz com as outras placas Arduino.

# 1º projeto: Hello world

- Pisca-pisca simples.



LED do NodeMCU
Pino D0 (GPIO 16)

blink_com_1_led_para_nodemcu | Arduino 1.6.8

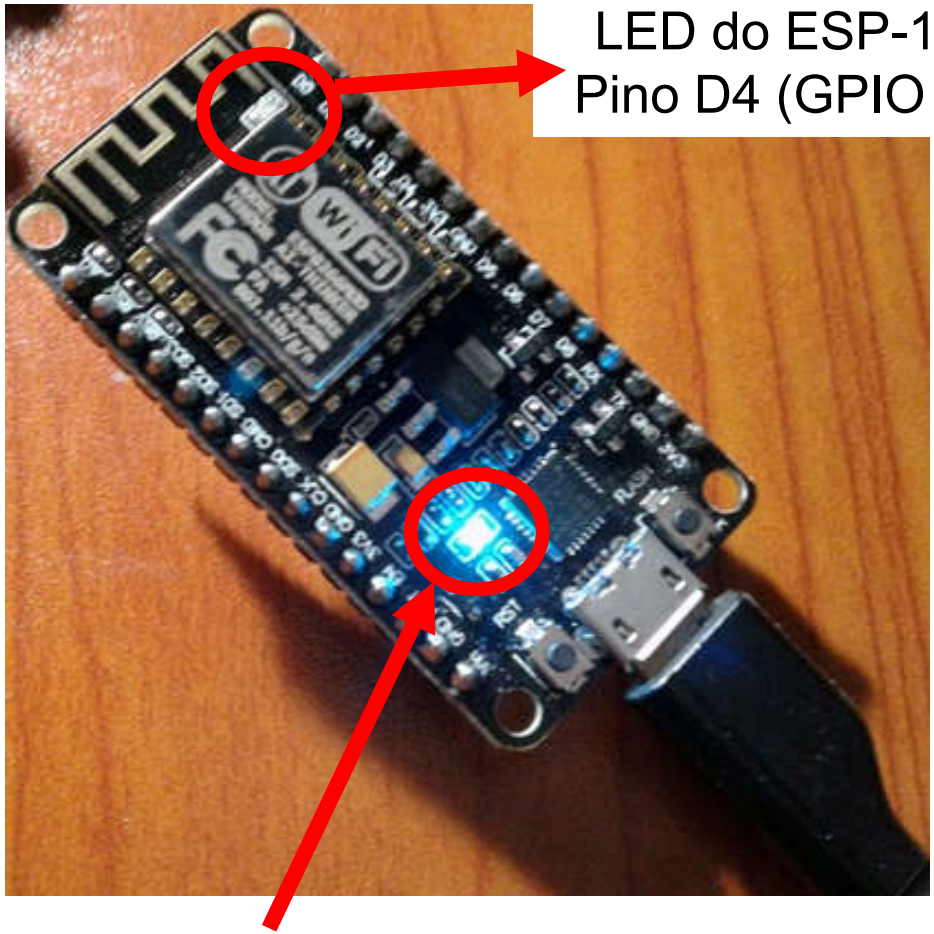
Arquivo Editar Sketch Ferramentas Ajuda

blink_com_1_led_para_nodemcu

```
1 void setup() {
2   pinMode(D0, OUTPUT); // GPIO16
3 }
4 void loop() {
5   digitalWrite(D0, HIGH);
6   delay(400);
7   digitalWrite(D0, LOW);
8   delay(400);
9 }
```

# 1º projeto: Hello world

- Pisca-pisca duplo.




blink_2_leds_para_nodemcu_v2 | Arduino 1.6.8

Arquivo Editar Sketch Ferramentas Ajuda

blink_2_leds_para_nodemcu_v2

```
1  void setup() {
2    pinMode(D0, OUTPUT); // GPIO16
3    pinMode(D4, OUTPUT); // GPIO2
4  }
5  void loop() {
6    digitalWrite(D0, LOW);
7    digitalWrite(D4, HIGH);
8    delay(100);
9    digitalWrite(D0, HIGH);
10   digitalWrite(D4, LOW);
11   delay(900);
12 }
```

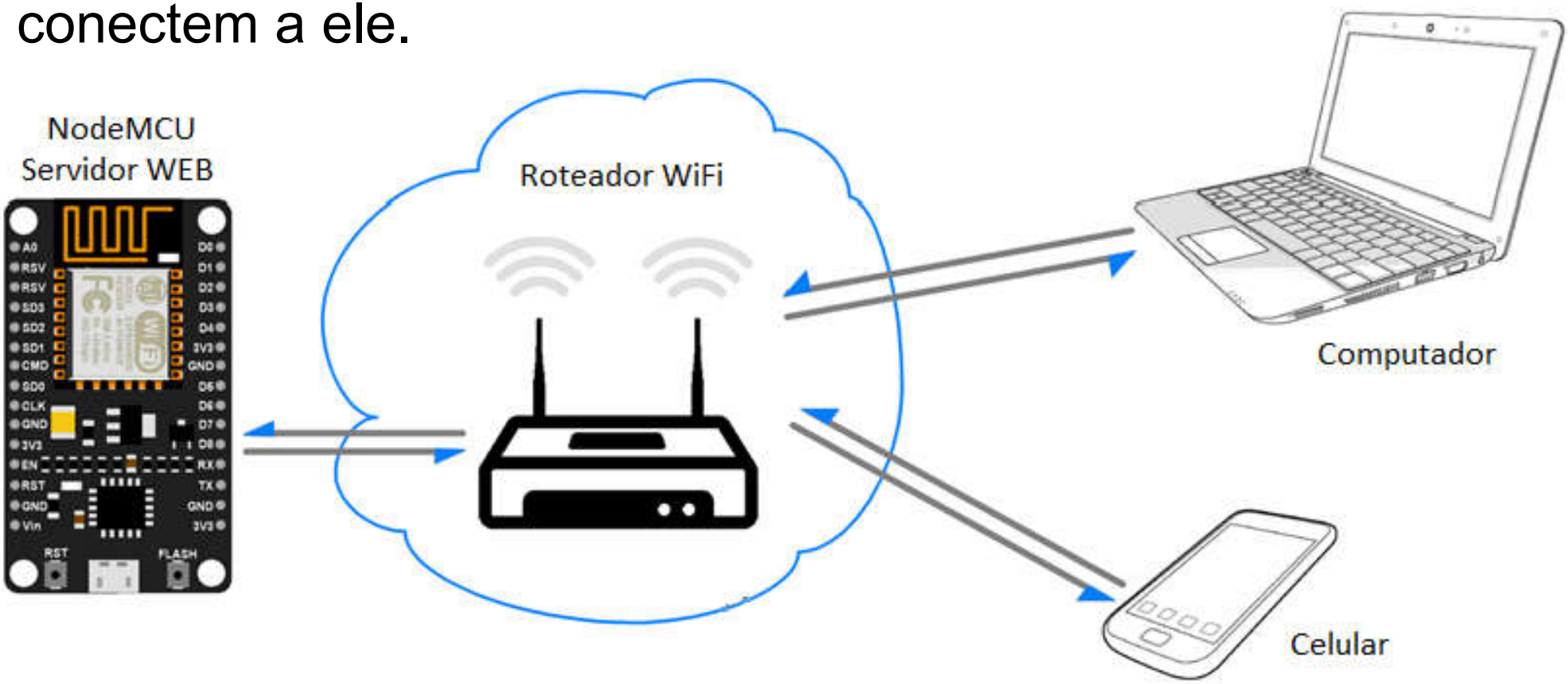
Observe que haverá uma inversão de HIGH ↔ LOW.

# Configurações de rede

- Inicialmente deve-se saber o modo de operação do NodeMCU quanto a conexão WIFI através da função **WiFi.mode()**:
  - **WIFI_STA** (modo station)
  - **WIFI_AP** (modo ponto de acesso)
  - WIFI_AP_STA
  - WIFI_OFF

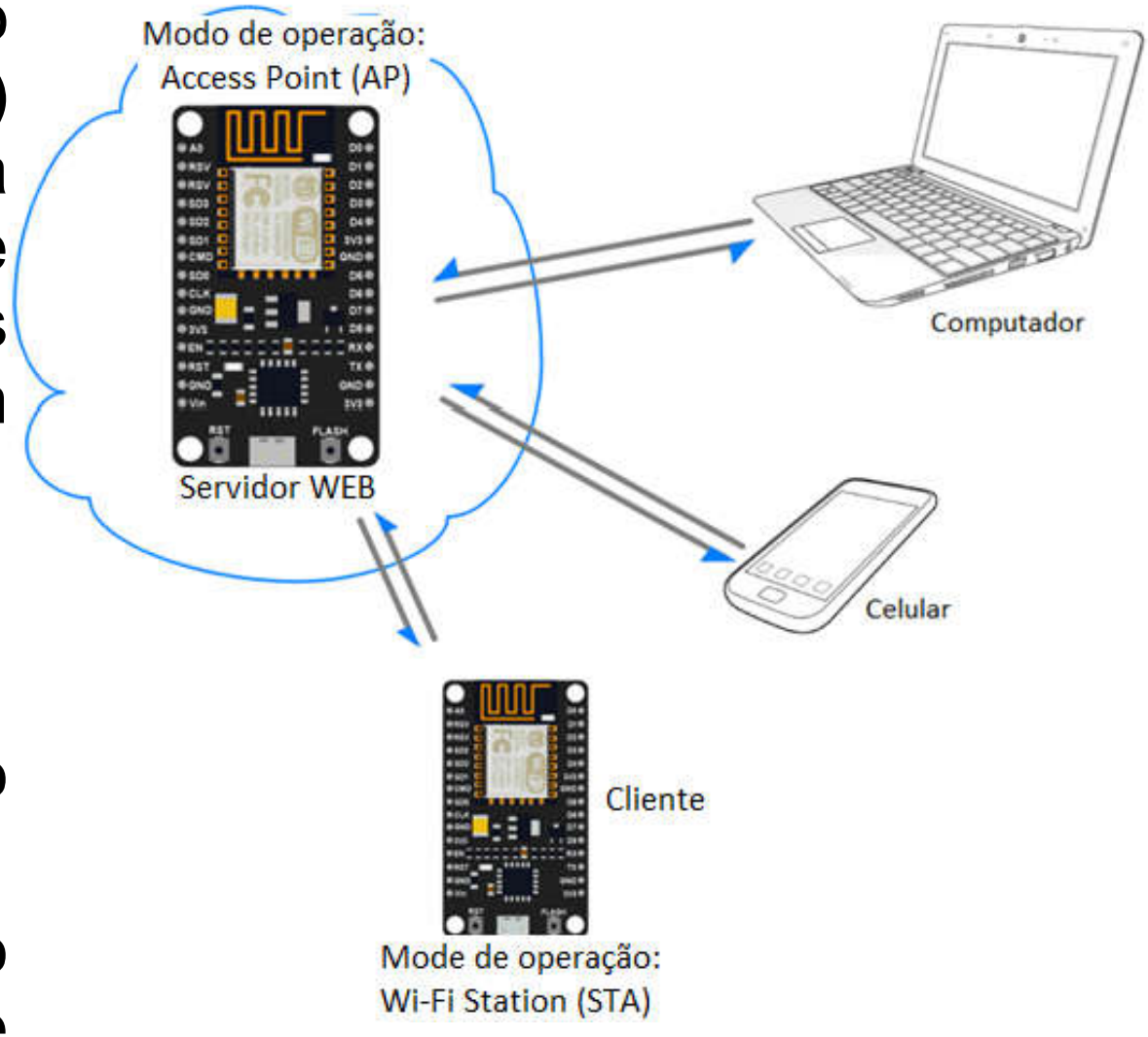
# Configurações de rede

- Na imagem abaixo o NodeMCU atua no modo **Station**, pode fornecer páginas WEB atuando, então como um servidor WEB, mas necessita de um roteador para que os clientes se conectem a ele.



# Configurações de rede

- Agora o NodeMCU atua como um Ponto de Acesso (**Access Point - AP**) criando uma rede WiFi a partir dele mesmo e permitindo que outros dispositivos se conectem a essa rede.




- Aplicações:
  - Áreas remotas que não tenha WiFi.
  - Aplicação em que não se quer depender de roteadores.

# 2º projeto: modo station

- Usaremos as seguintes funções para conectar o NodeMCU a rede WiFi:
  - **WiFi.mode(WIFI_STA)**
  - **WiFi.begin(ssid, senha)**: inicia a conexão WiFi, passando o SSID e a senha, se necessário.
  - **WiFi.status()**: retorna a situação da rede, sendo que *WL_CONNECTED* indica que a conexão foi feita com sucesso.
  - **WiFi.local(IP)**: informa o endereço IP atribuído pela rede.

# 2º projeto: modo station

- Dispositivos números MAC definidos no roteador:

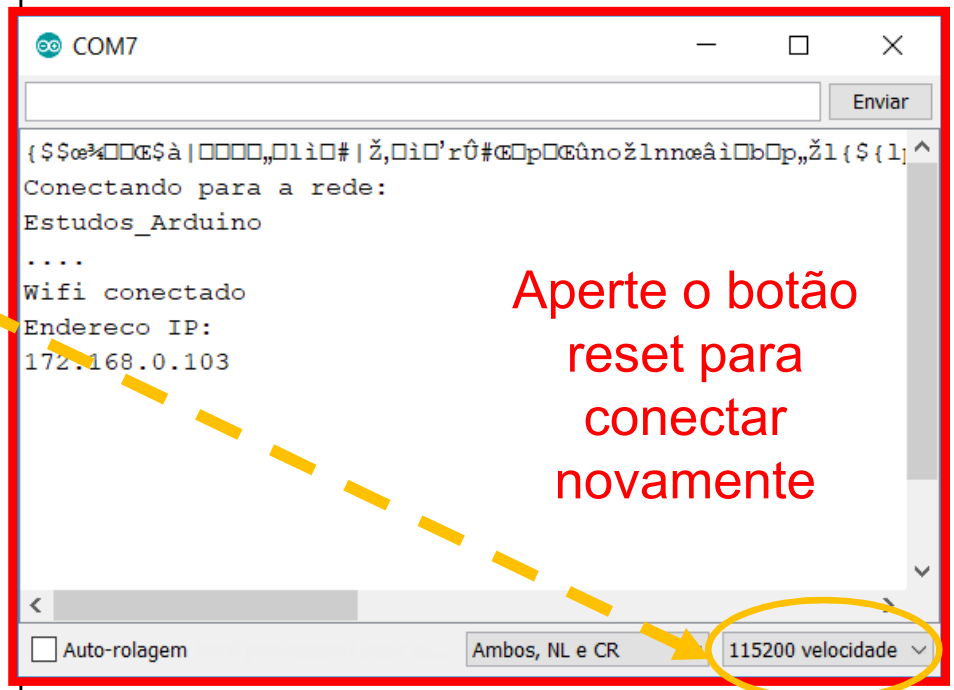
| MAC | IP |
|---|---|
| 60:01:94:51:EB:8A | 172.168.0.130 |
| 60:01:94:51:DD:A4 | 172.168.0.131 |
| 60:01:94:45:CD:11 | 172.168.0.132 |
| 2C:3A:E8:37:DA:A2 | 172.168.0.133 |
| 60:01:94:51:DE:51 | 172.168.0.134 |
| 60:01:94:51:E8:A4 | 172.168.0.135 |

- Código. Após gravá-lo abra o serial monitor.

Bibliotecas

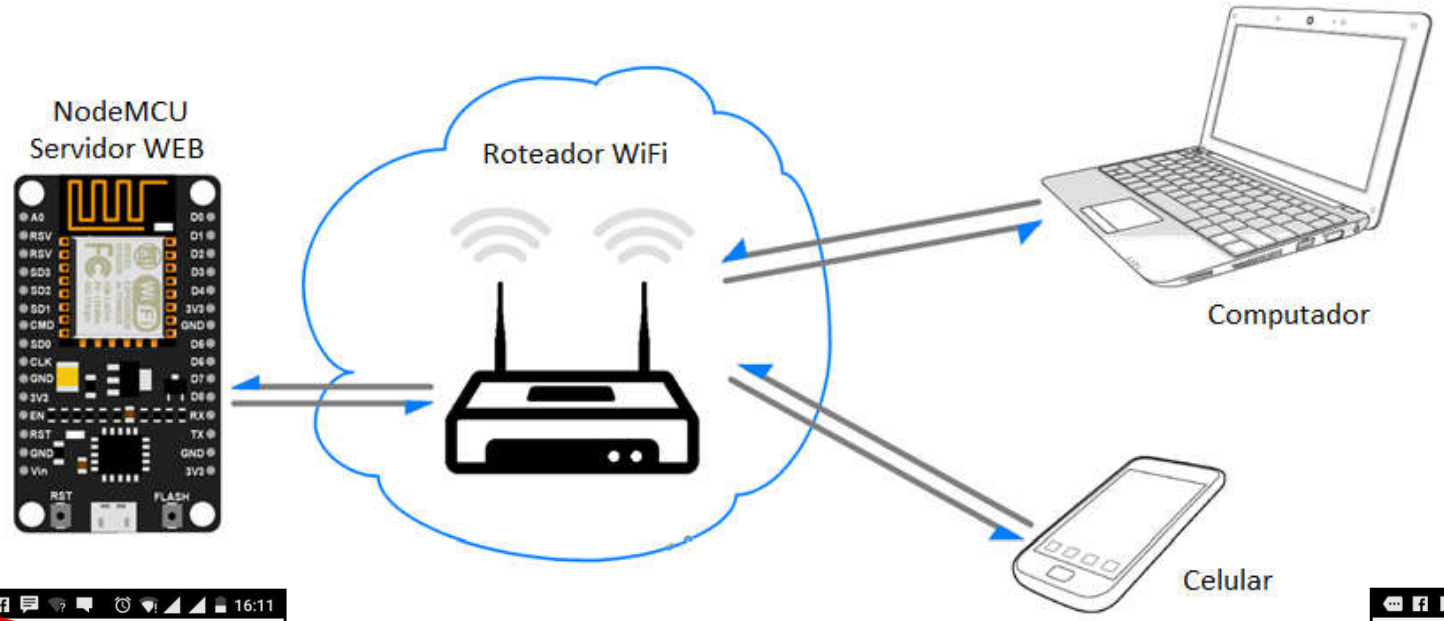
```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

const char* ssid = "Estudos_Arduino";
const char* senha = "arduinoifal";

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

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, senha);

  Serial.println("\n");
  Serial.print("Conectando a rede: ");
  Serial.println(ssid);

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  Serial.println("");
  Serial.println("WiFi conectado!");
  Serial.print("Endereço IP: ");
  Serial.println(WiFi.localIP());
}

void loop() {
}
```

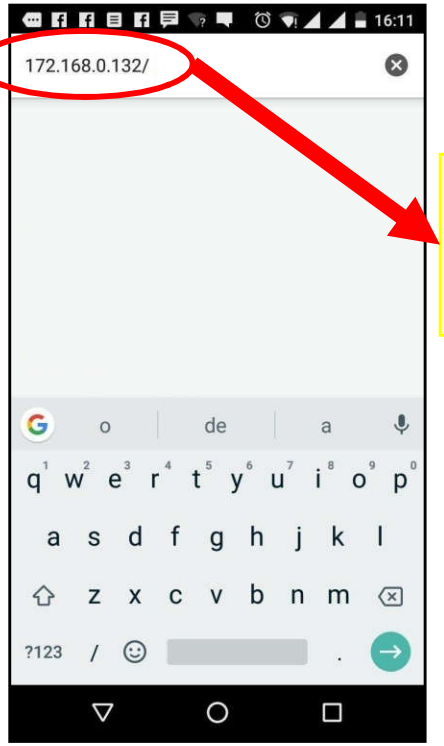
Serial Monitor:




Aperte o botão reset para conectar novamente

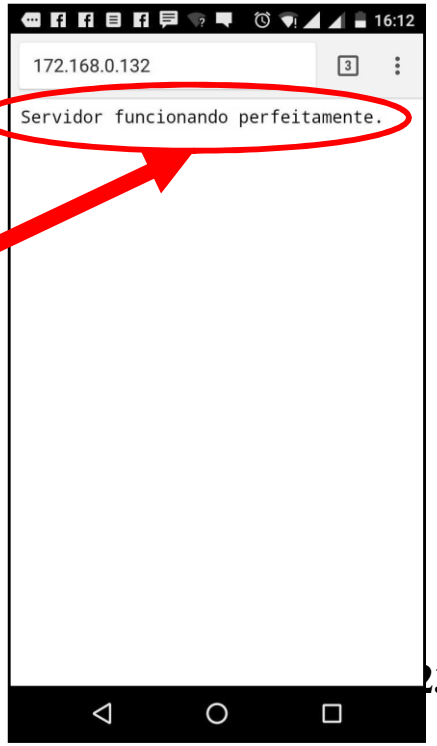
# 3º projeto: respondendo uma requisição simples




(1)
Cliente faz uma requisição: “/”.

(2)
O servidor responde a requisição.



# 3º projeto: respondendo uma requisição simples

- Basicamente duas funções serão bastante úteis para tratar as requisições de clientes:
  - server.on => essa função **recebe** as requisições dos clientes.
  - server.send => essa função **envia** uma resposta para o cliente.
- Exemplo 1:

A função server.on será executada quando a URI "/" for requisitada.

```
server.on("/", []() {
  server.send(200, "text/plain", "Servidor funcionando perfeitamente.");
});
```

O servidor responde com o código de status HTTP 200 (Ok).

"text/plain" significa que será enviado um texto simples.

O texto entre aspas será enviado para o navegador.

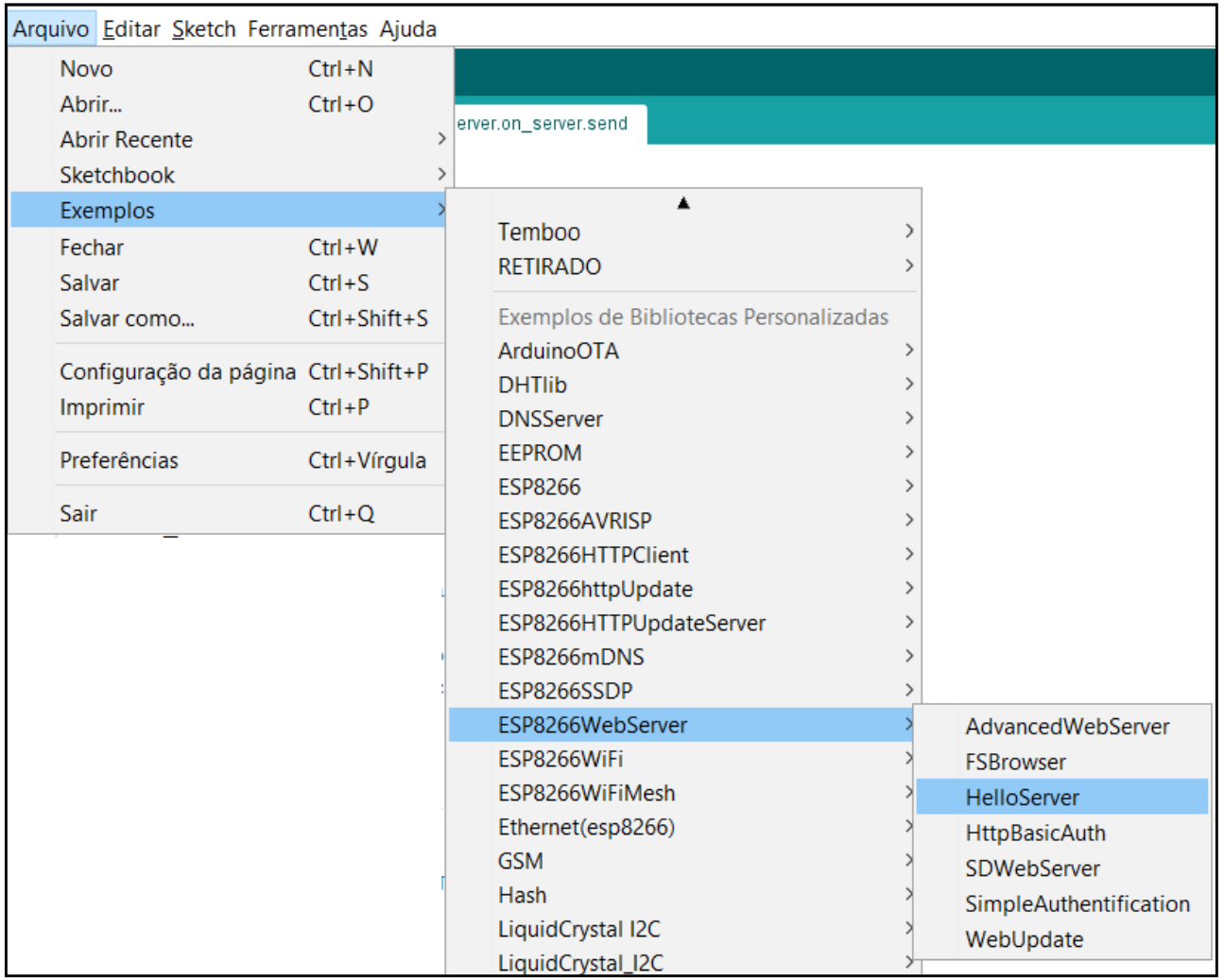
Porta 80

Configuração modo station

Explicação no slide anterior

```
1  #include <ESP8266WiFi.h>
2  #include <WiFiClient.h>
3  #include <ESP8266WebServer.h>
4
5  const char* ssid = "Estudos_Arduino";
6  const char* senha = "arduinoifal";
7
8  ESP8266WebServer server(80); // objeto server atenderá na
9  //porta 80 (porta padrão) quando solicitado no navegador
10
11 void setup() {
12   Serial.begin(115200);
13
14   WiFi.mode(WIFI_STA);
15   WiFi.begin(ssid, senha);
16
17   Serial.println("\n");
18   Serial.print("Conectando a rede: ");
19   Serial.println(ssid);
20
21   while (WiFi.status() != WL_CONNECTED) {
22     Serial.print(".");
23     delay(500);
24   }
25
26   Serial.println("");
27   Serial.println("WiFi conectado!");
28   Serial.print("Endereço IP: ");
29   Serial.println(WiFi.localIP());
30
31   server.on("/", []() {
32     server.send(200, "text/plain", "Servidor funcionando perfeitamente.");
33   });
34
35   server.begin();
36 }
37
38 void loop() {
39   server.handleClient(); //função que permite ouvir as solicitações externas.
40 }
```

- Os códigos apresentados são baseados no exemplo **HelloServer.** Usa-se as funções da biblioteca **ESP8266WebServer.h** para implementar um servidor web.



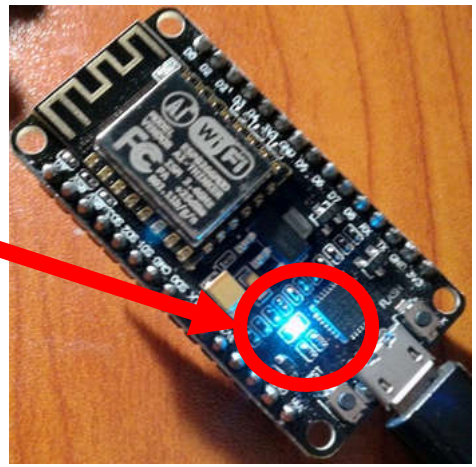
# 4º projeto: pisca-pisca ao receber requisição

A função server.on será executada quando a URI "/pisca" for requisitada.

A função "piscaled" será chamada.

```
server.on("/pisca", piscaled);
```

```
44 void piscaled() {
45   server.send(200, "text/plain", "LED D0 piscando");
46   for (int i = 0; i < 50; i++) {
47     digitalWrite(D0, 0);
48     delay(40);
49     digitalWrite(D0, 1);
50     delay(40);
51   }
52 }
```



LED D0

Pino LED D0 como saída

Configuração modo station

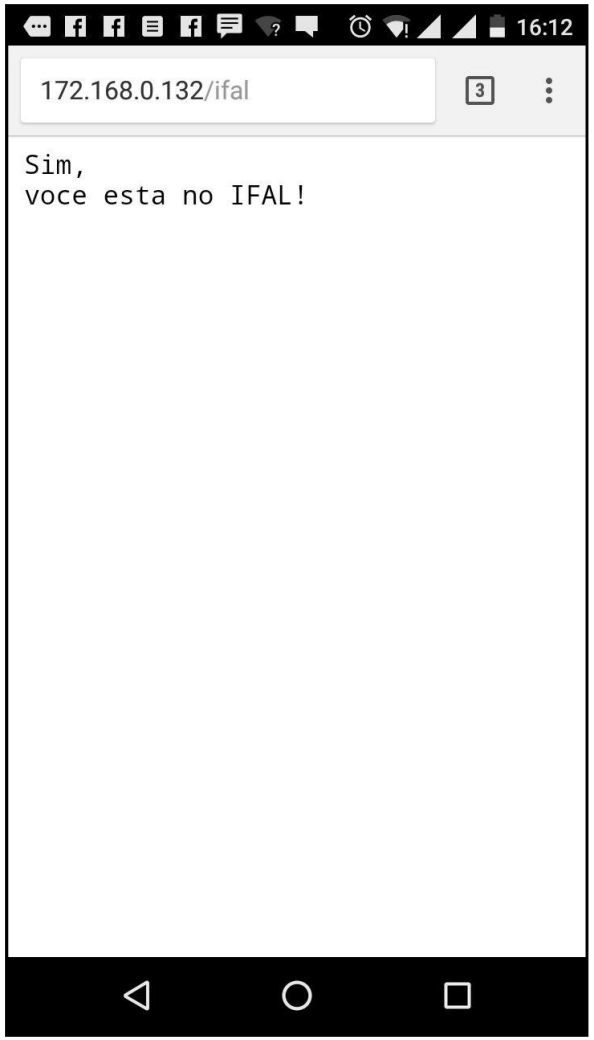
Requisição /pisca

Tratando a requisição /pisca

```
1 #include <ESP8266WiFi.h>
2 #include <WiFiClient.h>
3 #include <ESP8266WebServer.h>
4
5 const char* ssid = "Estudos_Arduino";
6 const char* senha = "arduinoifal";
7
8 ESP8266WebServer server(80);
9
10 void setup() {
11   pinMode(D0, OUTPUT);
12   Serial.begin(115200);
13
14   WiFi.mode(WIFI_STA);
15   WiFi.begin(ssid, senha);
16
17   Serial.println("\n");
18   Serial.print("Conectando a rede ");
19   Serial.println(ssid);
20
21   while (WiFi.status() != WL_CONNECTED) {
22     Serial.print(".");
23     delay(500);
24   }
25
26   Serial.println("");
27   Serial.println("WiFi conectado!");
28   Serial.print("Endereço IP: ");
29   Serial.println(WiFi.localIP());
30
31   server.on("/", []() {
32     server.send(200, "text/plain", "Servidor funcionando perfeitamente.");
33   });
34
35   server.on("/pisca", piscaled);
36
37   server.begin(); // ativa o servidor
38 }
39
40 void loop() {
41   server.handleClient(); //função que permite ouvir as solicitações externas.
42 }
43
44 void piscaled() {
45   server.send(200, "text/plain", "LED D0 piscando");
46   for (int i = 0; i < 50; i++) {
47     digitalWrite(D0, 0);
48     delay(40);
49     digitalWrite(D0, 1);
50     delay(40);
51   }
52 }
```

# Exercício

- Elabore uma requisição “/ifal” que quando solicitada mostre a tela abaixo e faça os dois leds do NodeMCU piscarem invertidamente.



# Referências bibliográficas


- OLIVEIRA, S. Internet das Coisas com ESP8266, Arduino e Raspberry Pi. São Paulo: Novatec, 2017.

- MCROBERTS, M. Arduino Básico. São Paulo: Novatec, 2015.
- JAVED, A. Criando projetos com Arduino para a Internet das coisas. São Paulo: Novatec, 2016.