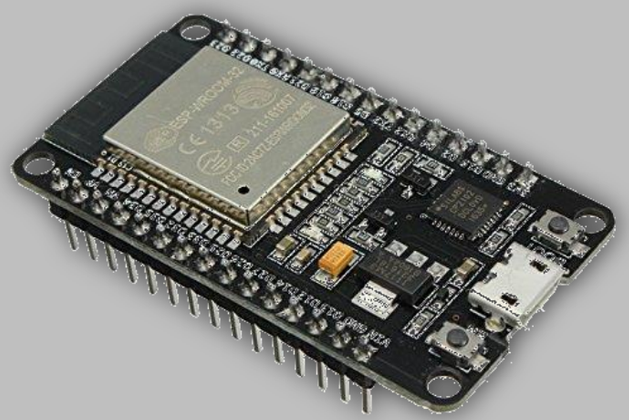
WiFiManager



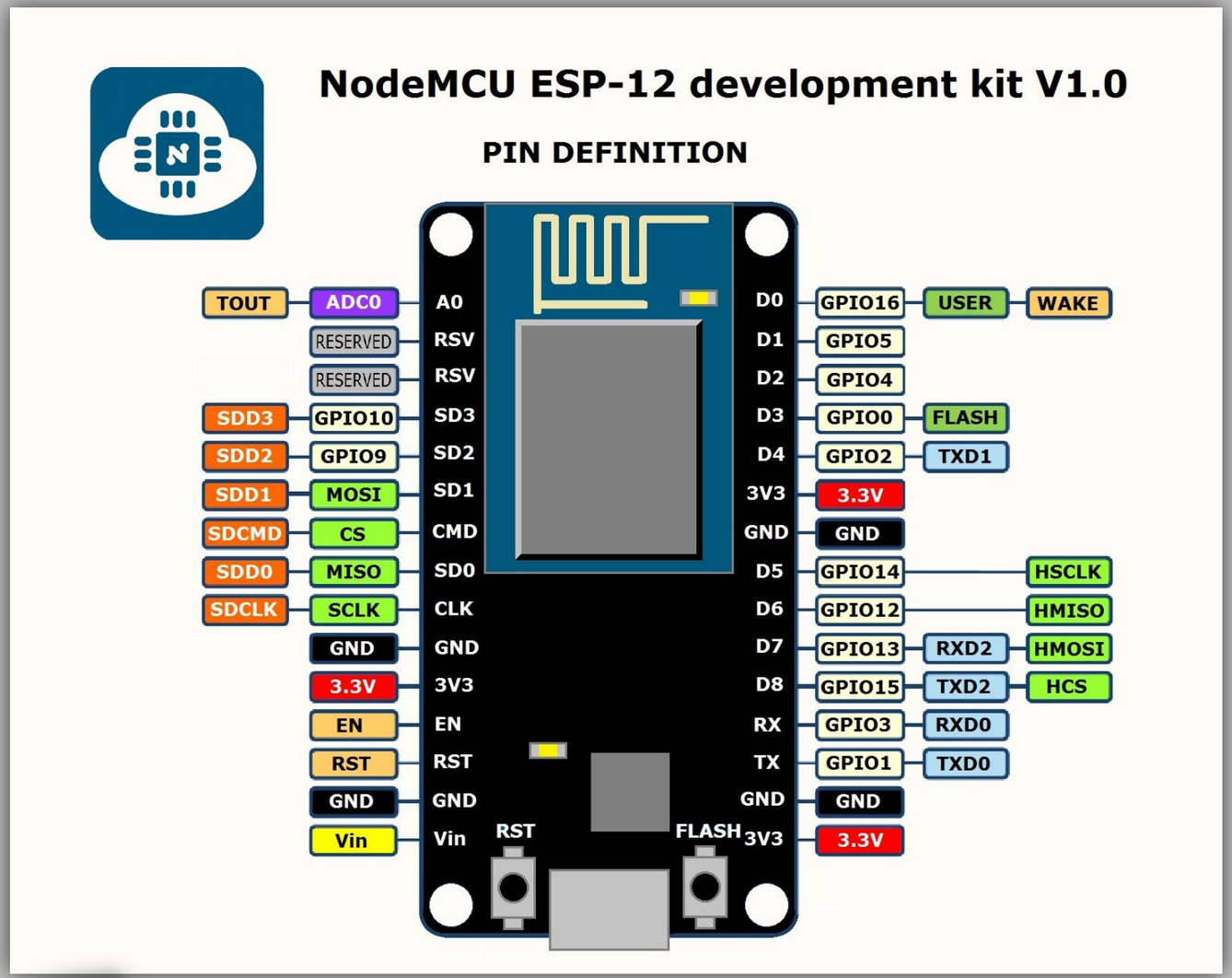
Por Fernando Koyanagi

Intenção da Aula

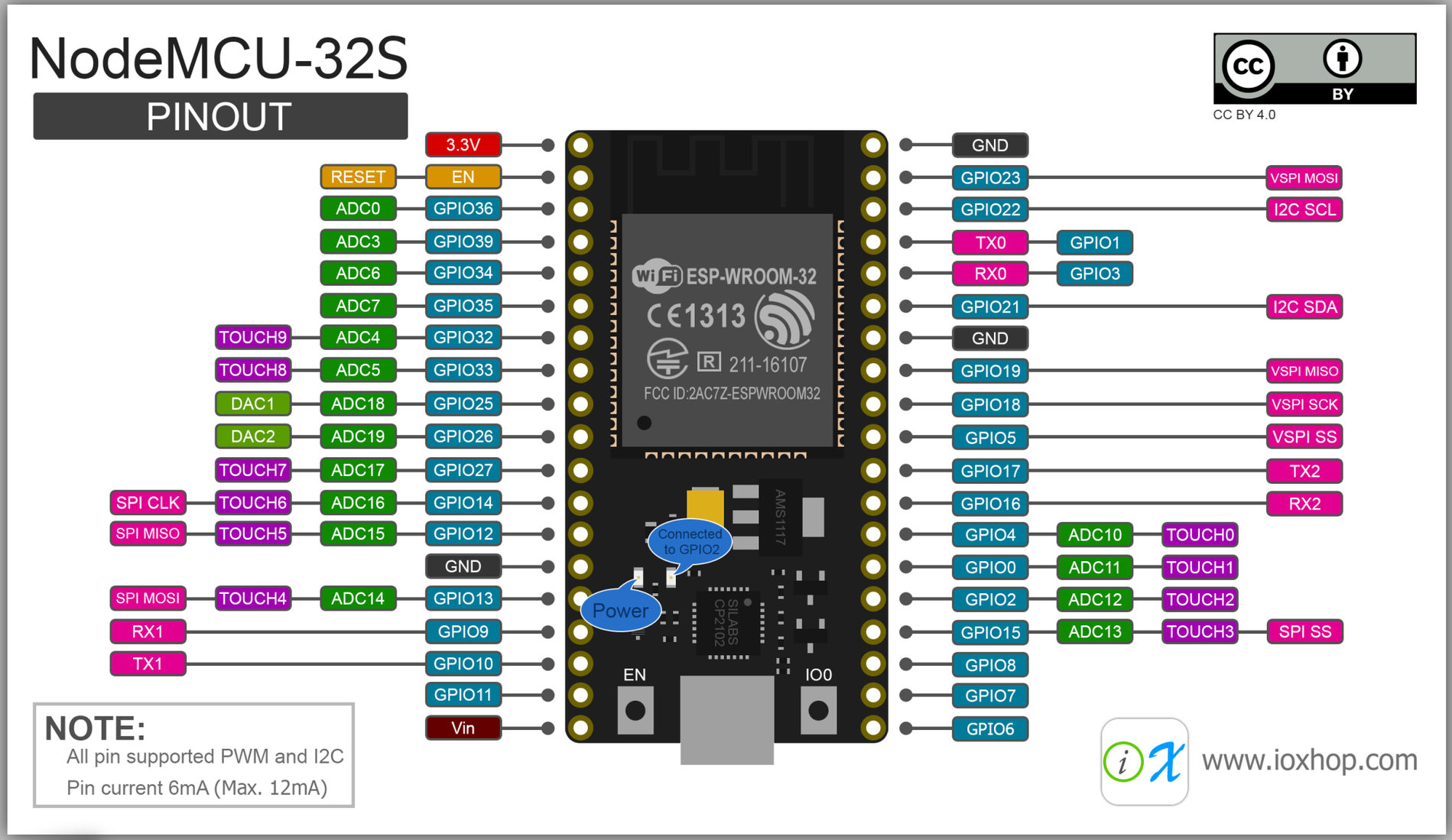
1. Conhecer a biblioteca WiFiManager e suas funções.

2. Fazer uma demonstração de uso

NodeMCU ESP-12E



NodeMCU ESP-WROOM-32



WiFiManager

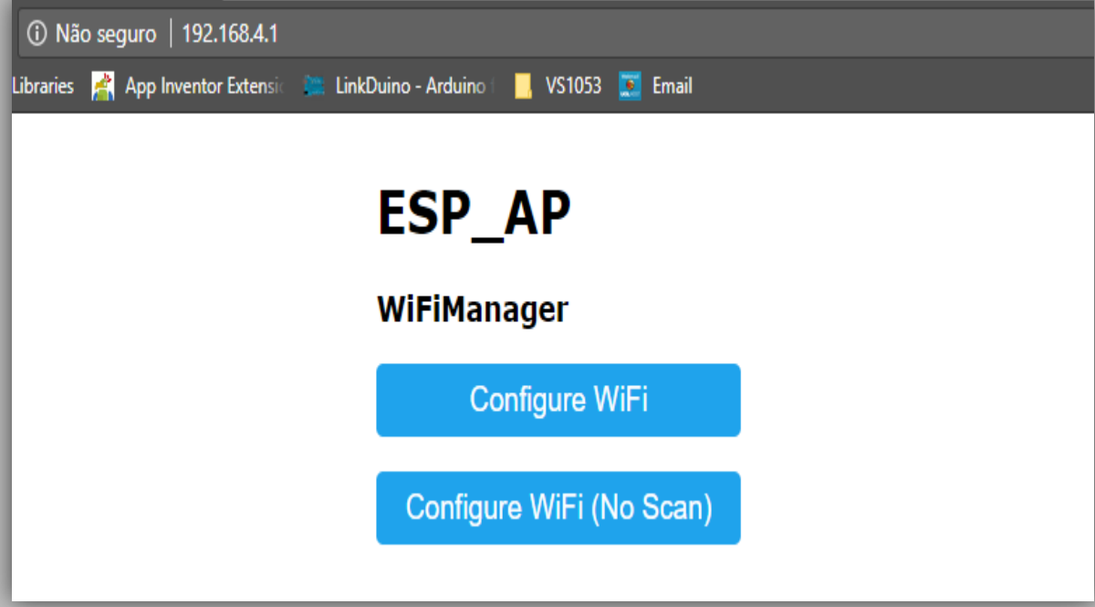
WiFiManager é uma biblioteca que serve como um gerenciador de conexões **WiFi**.
Com ela temos uma maior facilidade para configurar tanto um **Ponto de Acesso** quanto uma **Estação**.
Para o modo **Estação**, configuramos através de um portal no navegador.

Algumas características:

- Depende da conectividade automática
- Inicialização do Portal de Configuração Não automática
- Opera Seletivamente no modo duplo

Como Funciona

O **ESP** iniciará um portal de configuração **WiFi** quando ligado e salvará os dados de configuração em **memória não volátil**. Posteriormente, o portal de configuração só será iniciado novamente se um **botão for pressionado no módulo ESP**.

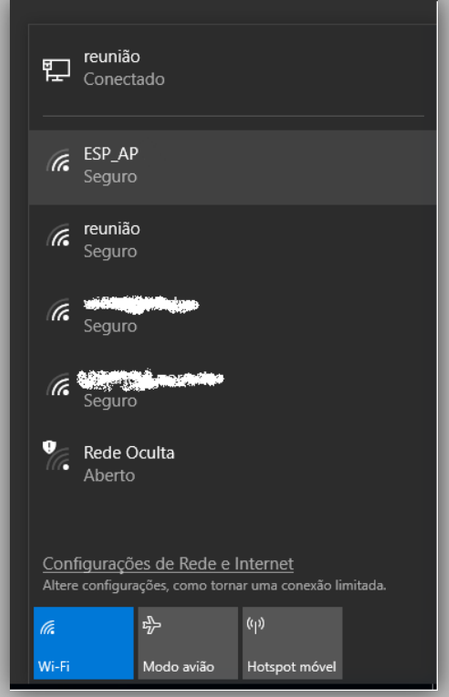


1. Utilizando qualquer dispositivo habilitado para **WiFi** com um navegador, conecte-se ao ponto de acesso recém-criado e digite o endereço **192.168.4.1** .

2. Na tela você terá duas opções para se conectar a uma rede existente:
   - **Configure WiFi**
   - **Configure WiFi (No Scan)**

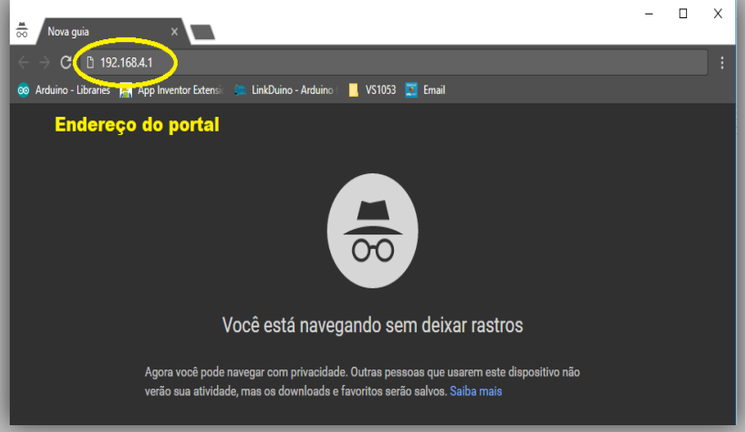
3. Escolha uma das redes e coloque a senha (se precisar), então salve e aguarde o ESP reiniciar.

4. Ao finalizar o boot, o ESP tentará se conectar a rede salva, caso não consiga, irá habilitar um Ponto de Acesso.



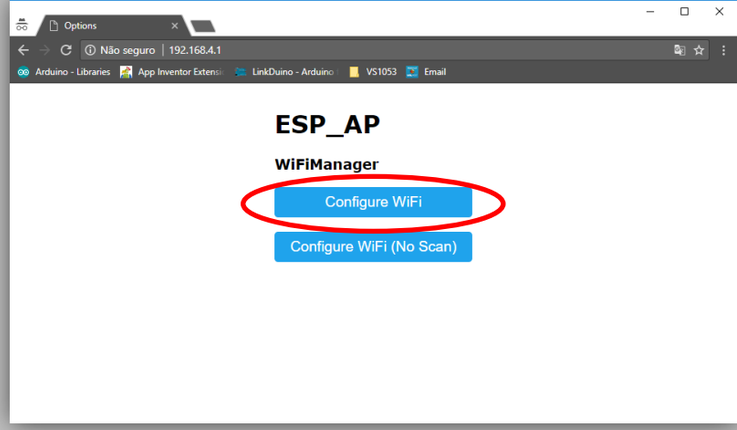
Conectar ao AP.

1



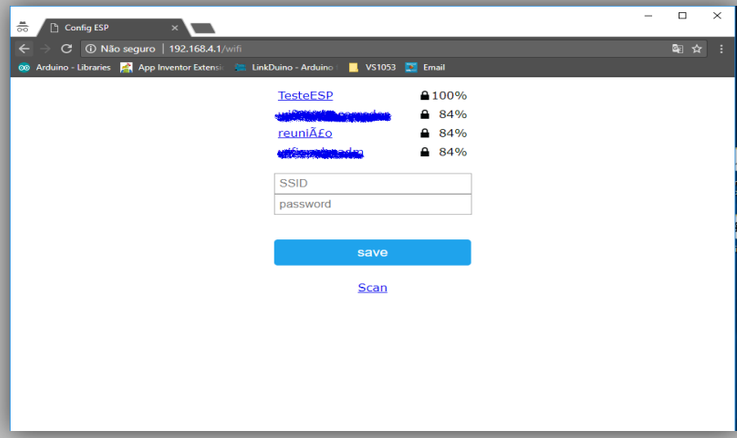
Acessar a url do portal
192.168.4.1

2



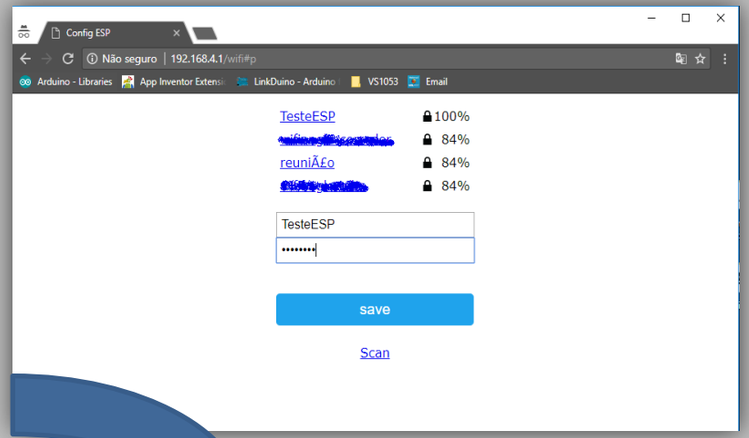
Acessar **Configure Wifi**.

3



Escolher uma rede para nos conectar.

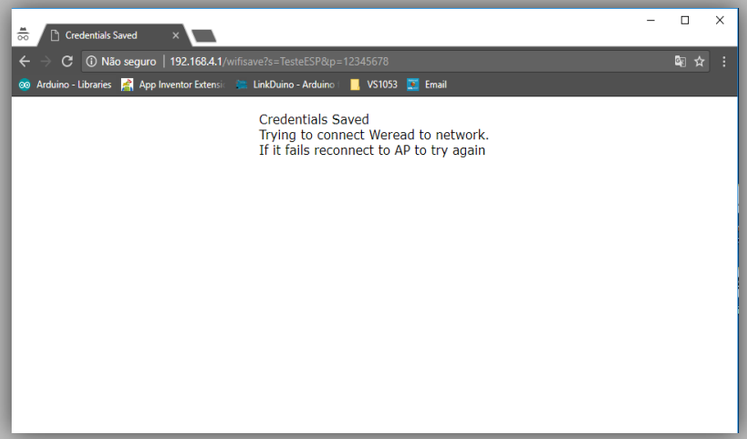
4



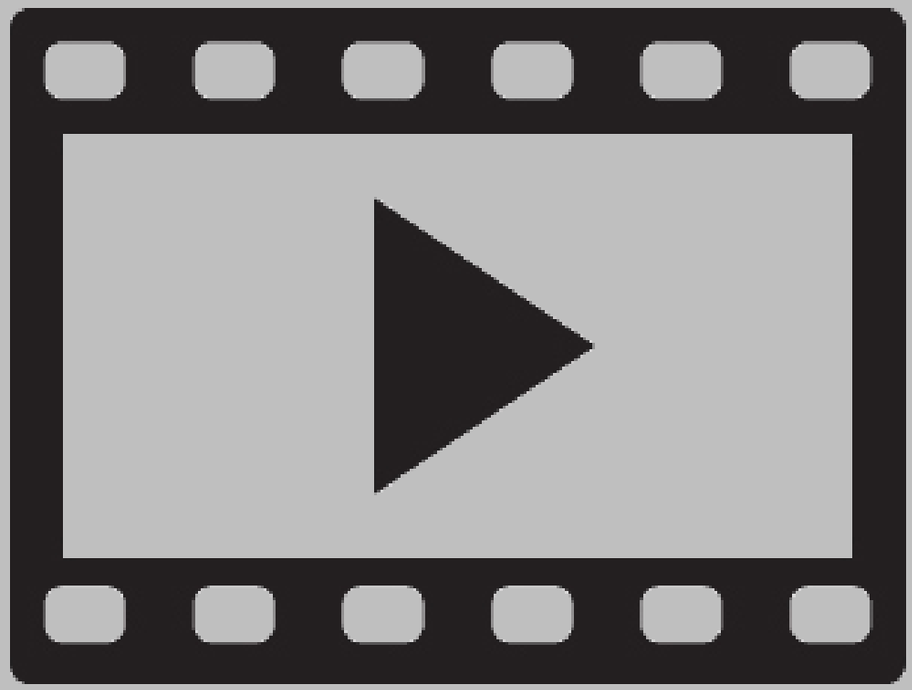
Digitar o **SSID** e
o **password** e **Salvar**.

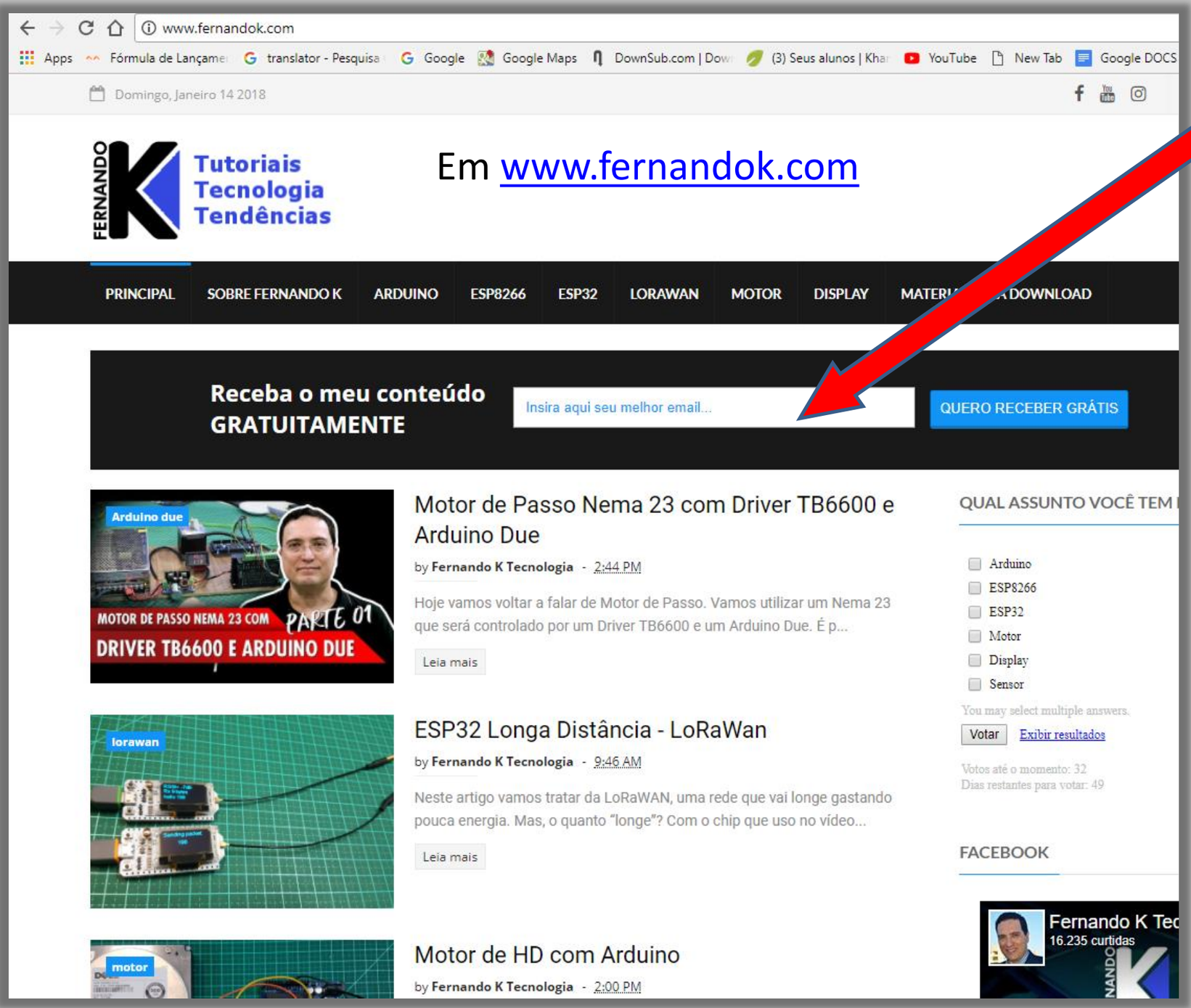
5

Feedback da conexão.
Entrando em modo **Estação**



Demonstração




www.fernandok.com
Apps
Fórmula de Lançame
translator - Pesquisa
Google
Google Maps
DownSub.com | Dow
(3) Seus alunos | Kha
YouTube
New Tab
Google DOCS
Domingo, Janeiro 14 2018
FERNANDO K
Tutoriais
Tecnologia
Tendências
Em www.fernandok.com
PRINCIPAL
SOBRE FERNANDO K
ARDUINO
ESP8266
ESP32
LORAWAN
MOTOR
DISPLAY
DOWNLOAD
Receba o meu conteúdo GRATUITAMENTE
Insira aqui seu melhor email...
QUERO RECEBER GRÁTIS
Arduino due
MOTOR DE PASSO NEMA 23 COM
PARTE 01
DRIVER TB6600 E ARDUINO DUE
Motor de Passo Nema 23 com Driver TB6600 e Arduino Due
by Fernando K Tecnologia - 2:44 PM
Hoje vamos voltar a falar de Motor de Passo. Vamos utilizar um Nema 23 que será controlado por um Driver TB6600 e um Arduino Due. É p...
Leia mais
lorawan
ESP32 Longa Distância - LoRaWan
by Fernando K Tecnologia - 9:46 AM
Neste artigo vamos tratar da LoRaWAN, uma rede que vai longe gastando pouca energia. Mas, o quanto "longe"? Com o chip que uso no vídeo...
Leia mais
motor
Motor de HD com Arduino
by Fernando K Tecnologia - 2:00 PM
QUAL ASSUNTO VOCÊ TEM
Arduino
ESP8266
ESP32
Motor
Display
Sensor
You may select multiple answers.
Votar
Exibir resultados
Votos até o momento: 32
Dias restantes para votar: 49
FACEBOOK
Fernando K Tec
16.235 curtidas

Seu e-mail

Bibliotecas

Adicione biblioteca “WifiManager-ESP32”.

Acesse o link e faça download da biblioteca.

Descompacte o arquivo e cole na pasta de bibliotecas da IDE do arduino.

C:/Program Files (x86)/Arduino/libraries

Bibliotecas

Adicione biblioteca “DNSServer-ESP32”.

Acesse o link e faça download da biblioteca.

Descompacte o arquivo e cole na pasta de bibliotecas da IDE do arduino.

C:/Program Files (x86)/Arduino/libraries

Bibliotecas

Adicione biblioteca “WebServer-ESP32”.

Acesse o link e faça download da biblioteca.

Descompacte o arquivo e cole na pasta de bibliotecas da IDE do arduino.

C:/Program Files (x86)/Arduino/libraries

Observação

A biblioteca **WiFiManager-ESP32**, já traz as configurações que funcionam com o **ESP8266**, por isso utilizaremos ela apenas, e não duas “**WiFiManager**” (uma para cada tipo de chip).

Como veremos mais adiante, **ESP8266WiFi** e **ESP8266WebServer** são bibliotecas que não precisaremos realizar o download, pois, elas já vem quando instalamos o **ESP8266** na **IDE do arduino.**

Funções

A seguir vamos dar uma olhada em algumas funções que o **WiFiManager** nos oferece.

1. autoConnect

A função autoConnect, é responsável por criar um Access Point. Podemos usá-la de três maneiras.

- autoConnect(“nome da rede”, “senha”); → cria uma rede com o nome e a senha definidos.
- autoConnect(“nome da rede”); → cria uma rede aberta com o nome definido
- autoConnect(); → cria uma rede aberta e com nome gerado automaticamente com o nome sendo ‘ESP’ + chipID.

```
boolean autoConnect(char const *apName, char const *apPassword = NULL);
boolean autoConnect(char const *apName);
boolean autoConnect();
```

Funções

2. startConfigPortal

A função startConfigPortal, é responsável por criar um Access Point sem tentar conectar antes a uma rede salva anteriormente.

- startConfigPortal(“nome da rede”, “senha”); → cria uma rede com o nome e a senha definidos.
- startConfigPortal(); → cria uma rede aberta e com nome gerado automaticamente com o nome sendo ‘ESP’ + chipID.

```
boolean startConfigPortal();
boolean startConfigPortal(char const *apName, char const *apPassword = NULL);
```

3. getConfigPortalSSID

Retorna o SSID do portal (Access Point)

4. getSSID

Retorna o SSID da rede a qual está conectado.

5. getPassword

Retorna a senha da rede a qual está conectado.

```
String getConfigPortalSSID();

String getSSID();

String getPassword();
```

Funções

6. setDebugOutput

A função setDebugOutput é responsável por imprimir mensagens de debug no serial monitor, essas mensagens já são definidas na biblioteca, conforme passa pelas funções os dados vão sendo impressos.

Por padrão essa função vem setada como **TRUE**, caso queira desabilitar as mensagens, basta setar a função como **FALSE**.

```
void setDebugOutput(boolean debug);
```

7. setMinimumSignalQuality

A função setMinimumSignalQuality é responsável por filtrar as redes baseadas na qualidade do sinal. Por padrão o WiFiManager não mostrará redes com sinal abaixo de 8%.

```
void setMinimumSignalQuality(int quality = 8);
```

8. setRemoveDuplicateAPs

A função setRemoveDuplicateAPs é responsável por remover as duplicatas de redes.

Por padrão vem setado como **TRUE**.

```
void setRemoveDuplicateAPs(boolean removeDuplicates);
```

Funções

9. setAPStaticIPConfig

A função setAPStaticIPConfig é responsável por setar as configurações de endereço estáticas quando no modo access point.
(IP, GATEWAY, SUBNET)

```
void setAPStaticIPConfig(IPAddress ip, IPAddress gw, IPAddress sn);
```

10. setSTAStaticIPConfig

A função setSTAStaticIPConfig é responsável por setar as configurações de endereço estáticas quando no modo estação.
(IP, GATEWAY, SUBNET)

```
void setSTAStaticIPConfig(IPAddress ip, IPAddress gw, IPAddress sn);
```

Deve-se adicionar o comando antes de **autoConnect** !!!

```
// setAPStaticIPConfig(ip, gateway, subnet);
wifiManager.setAPStaticIPConfig(IPAddress(192,168,16,2), IPAddress(192,168,16,1), IPAddress(255,255,255,0));
//modo AP
wifiManager.autoConnect("ESP_AP"); //cria uma rede sem senha
```

Funções

11. setAPCallback

A função setAPCallback é responsável por informar que o modo AP foi iniciado.
O parâmetro é uma função que deve-se criar para indicá-la como sendo um callback;

```
//protótipo
void setAPCallback( void (*func)(WiFiManager*) );

//exemplo de uso
wifiManager.setAPCallback(configModeCallback);
void configModeCallback (WiFiManager *myWiFiManager){ /*implementar*/ }
```

12. setSaveConfigCallback

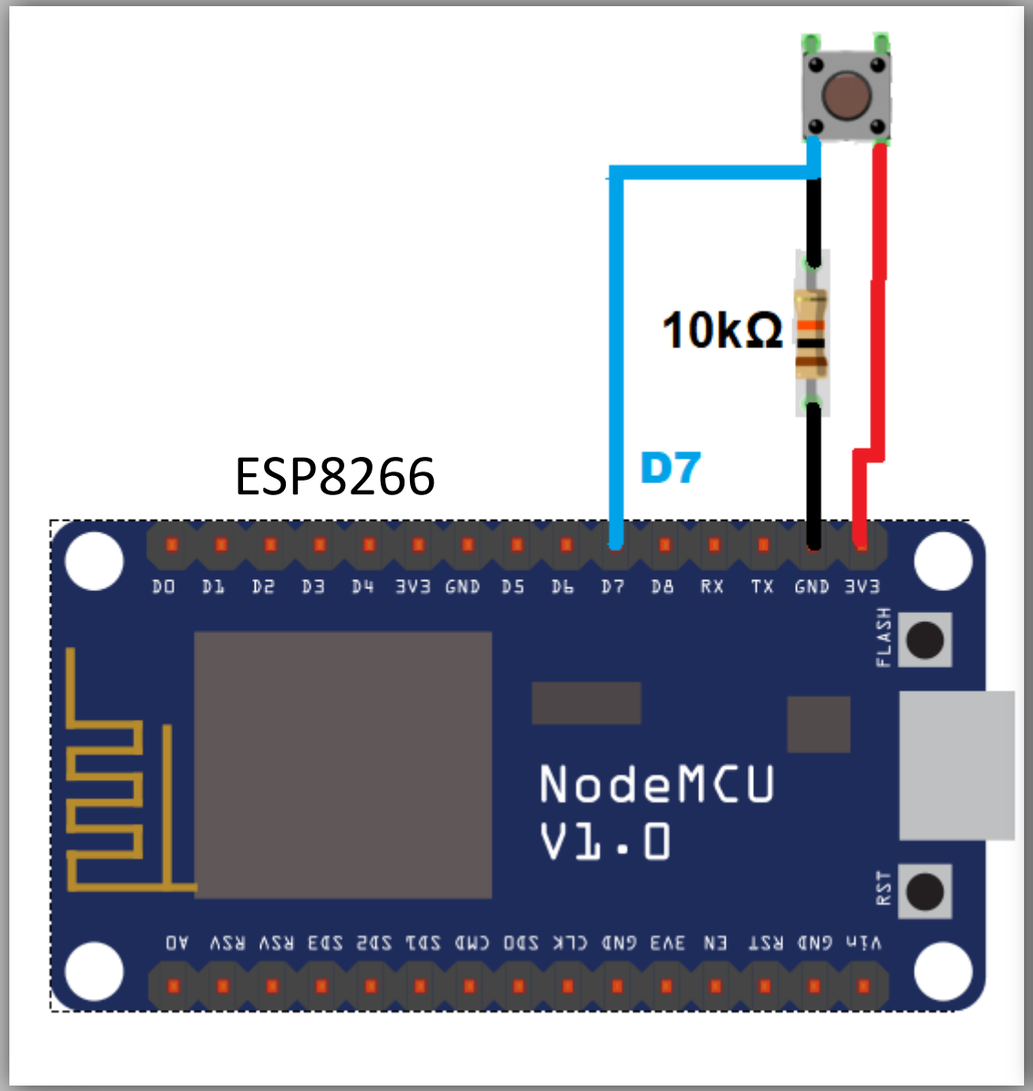
A função setSaveConfigCallback é responsável por informar que uma nova configuração foi salva e a conexão foi realizada com sucesso.
O parâmetro é uma função que deve-se criar e indicá-la como sendo um callback.

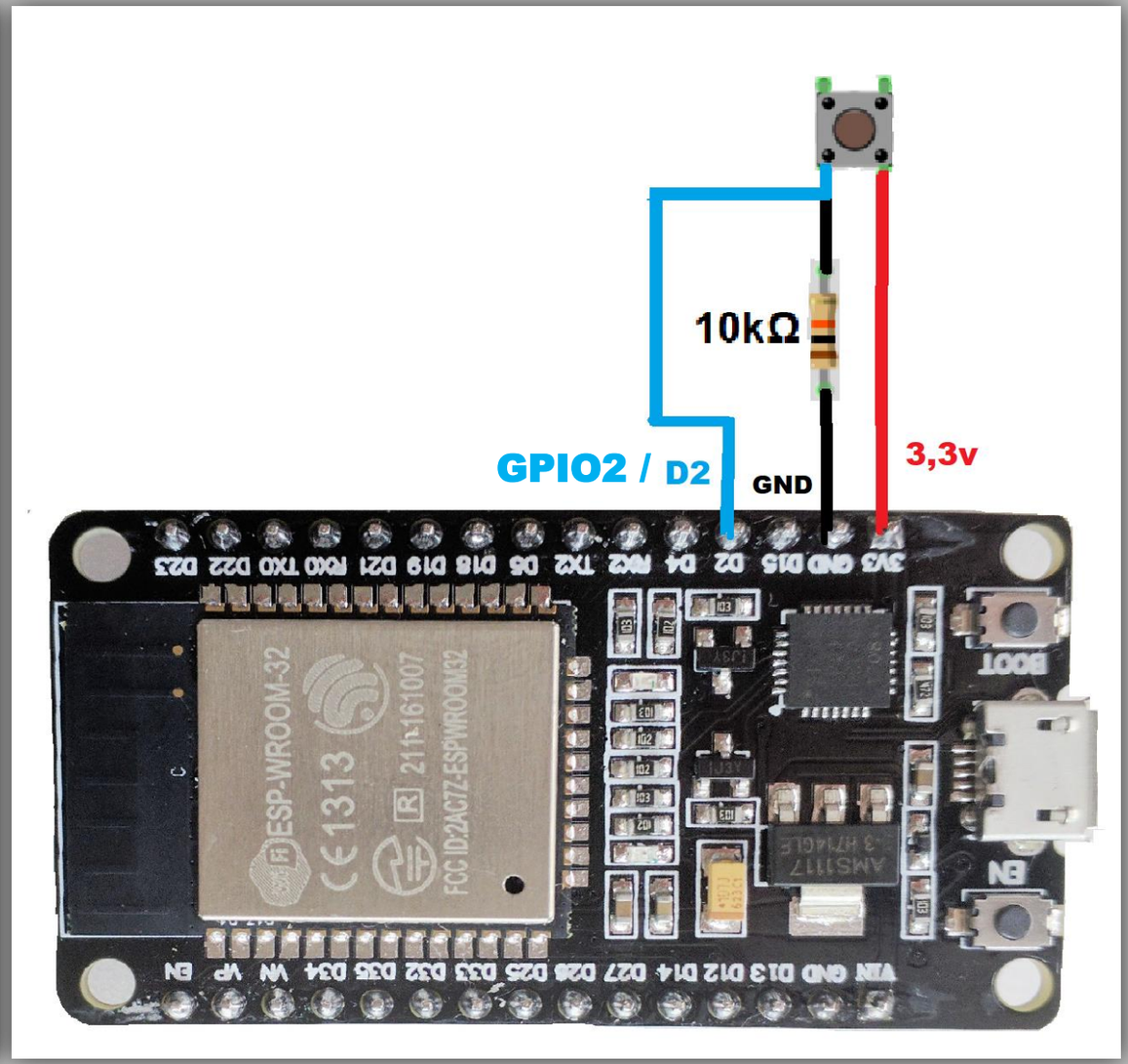
```
//protótipo
void setSaveConfigCallback( void (*func)(void) );

//exemplo de uso
wifiManager.setSaveConfigCallback(saveConfigCallback);
void saveConfigCallback (){ /*implementar*/ }
```

Deve-se adicionar o comando antes de **autoConnect** !!!

Montagem


ESP8266
10kΩ
D7
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
RST
NodeMCU
V1.0


10kΩ
GPIO2 / D2
GND
3,3v
ESP-WROOM-32
BOOT
EN

Exemplo

Veremos a seguir um exemplo de como criar um **Access Point** com o ESP (o código servirá tanto para o **ESP8266** como para o **ESP32**). Após a criação do AP, vamos acessar o portal através do IP **192.168.4.1** (que é o padrão para acessá-lo), então vamos buscar as redes disponíveis , selecionar uma delas e salvar. A partir daí o ESP irá reiniciar e tentará se conectar à ela, trabalhando então como uma **estação** e não mais como um **Access Point**.

Após entrar em modo **estação**, apenas através do botão que irá fazer o ESP voltar ao modo **Access Point**.

Bibliotecas

Primeiramente vamos definir as bibliotecas que utilizaremos.
Repare que temos comandos **#if defined, #else** e **#endif.** Eles são condicionais para incluir bibliotecas necessárias referente ao chip. Essa parte é extremamente importante para poder rodar o mesmo código tanto no ESP8266 quanto no ESP32.

```
#if defined(ESP8266)
#include <ESP8266WiFi.h> //ESP8266 Core WiFi Library
#else
#include <WiFi.h> //ESP32 Core WiFi Library
#endif

#if defined(ESP8266)
#include <ESP8266WebServer.h> //Local WebServer used to serve the configuration portal
#else
#include <WebServer.h> //Local WebServer used to serve the configuration portal
#endif

#include <DNSServer.h> //Local DNS Server used for redirecting all requests to the configuration portal
#include <WiFiManager.h> // WiFi Configuration Magic
```

Setup

No setup estamos configurando nosso **WiFiManager** da maneira mais simples, vamos apenas definir os callbacks e criar a rede.

```
const int PIN_AP = D7; //pino de ligação do botão para esp32 PIN_AP = 2;

void setup() {
     Serial.begin(9600);
     pinMode(PIN_AP, INPUT);
     //declaração do objeto wifiManager
     WiFiManager wifiManager;

     //utilizando esse comando, as configurações são apagadas da memória
     //caso tiver salvo alguma rede para conectar automaticamente, ela é apagada.
     //wifiManager.resetSettings();

     //callback para quando entra em modo de configuração AP
     wifiManager.setAPCallback(configModeCallback);
     //callback para quando se conecta em uma rede, ou seja, quando passa a trabalhar em modo estação
     wifiManager.setSaveConfigCallback(saveConfigCallback);

     //cria uma rede de nome ESP_AP com senha 12345678
    wifiManager.autoConnect("ESP_AP", "12345678");
}
```

Loop

No loop, faremos a leitura do pino do botão para saber se ele foi pressionado e então vamos chamar o método para habilitar novamente o modo AP.

```
void loop()
{
     if ( digitalRead(PIN_AP) == HIGH )
     {
          WiFiManager wifiManager;
          if(!wifiManager.startConfigPortal("ESP_AP", "12345678") )
          {
               Serial.println("Falha ao conectar");
               delay(2000);
               ESP.restart();
          }
     }
}
```

Ao pressionar o botão o ESP sairá do modo **Estação** e abrirá seu **Access Point** e o portal**.** Lembre-se que não utilizamos o comando **resetSettings**(), as configurações ainda permanecem salvas para a próxima vez que o ESP inicializar.

Callbacks

```
//callback que indica que o ESP entrou no modo AP
void configModeCallback (WiFiManager *myWiFiManager) {
     Serial.println(“Entrou no modo de configuração");
     Serial.println(WiFi.softAPIP()); //imprime o IP do AP
     Serial.println(myWiFiManager->getConfigPortalSSID()); //imprime o SSID criado da rede
}
```

```
//callback que indica que salvamos uma nova rede para se conectar (modo estação)
void saveConfigCallback () {
    Serial.println(“Configuração salva");
}
```

As funções de callback, servem para você ter o momento exato de uma operação, no nosso caso a entrada no modo AP e no modo Estação. Podemos então implementar alguma rotina desejada. Como recuperar o SSID da rede conectada por exemplo.

Em www.fernandok.com

Download arquivos PDF e INO do código fonte