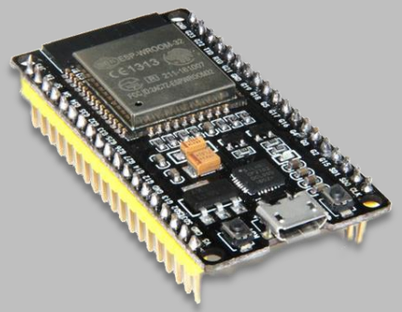
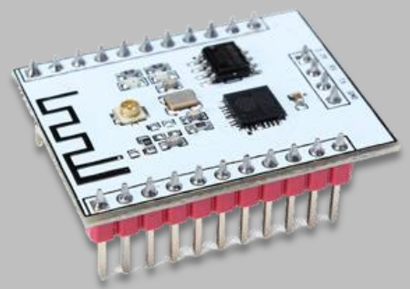
ESP32 e ESP8266: Programação no Ar

Over the Air
OTA

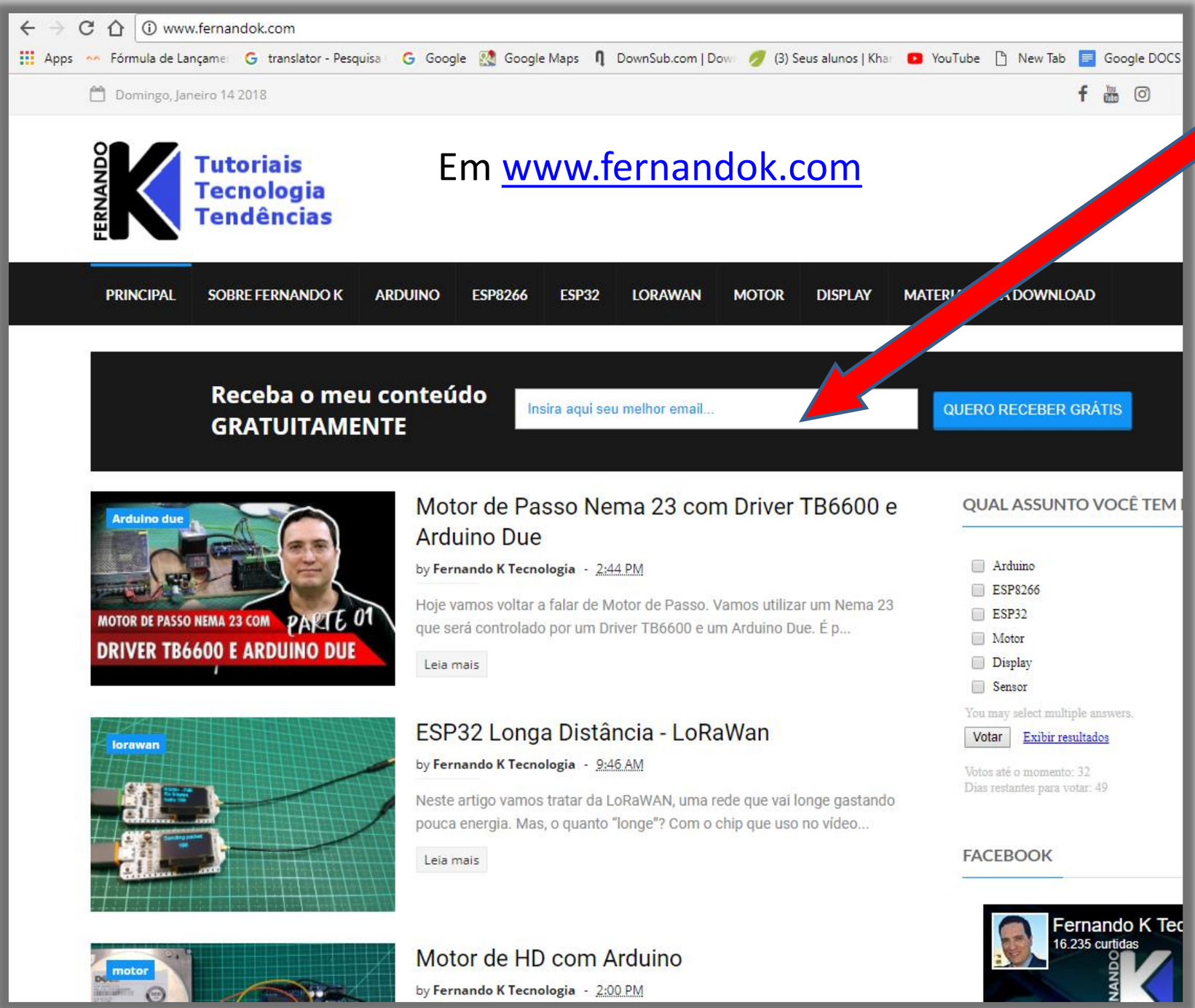




Por Fernando Koyanagi

Intenção dessa aula

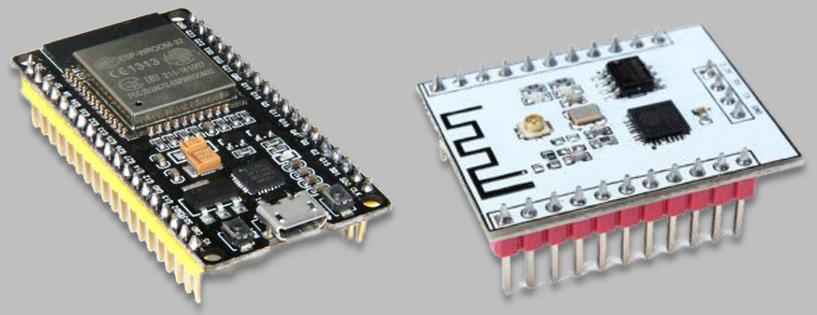
1. Explicar exemplo básico de programação OTA no ESP32 e ESP8266.


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
MATERI
DOWNLOAD
Receba o meu conteúdo GRATUITAMENTE
Insira aqui seu melhor email...
QUERO RECEBER GRÁTIS
Arduino due
MOTOR DE PASSO NEMA 23 COM DRIVER TB6600 E ARDUINO DUE
PARTE 01
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

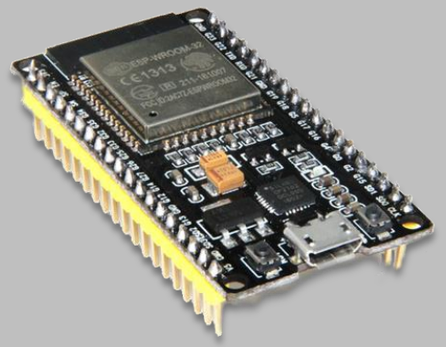
Aviso

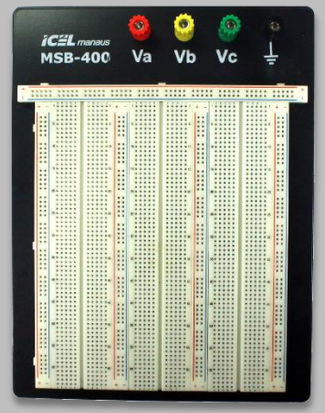
Este vídeo explica um exemplo simples de programação OTA para o ESP32.
Este exemplo pode ser usado também para o ESP8266, basta adaptar a montagem e seguir uma pequena mudança no código, indicada mais a frente.



Recursos usados

- 2 Leds
- 2 Resistores de 220 ohm
- ESP32
- Protoboard





Código OTA

Organização do código e códigos obrigatórios

```
void setup()
{
//(...)
}
```

Obrigatório:

ArduinoOTA.onStart();
ArduinoOTA.onEnd();

ArduinoOTA.onProgress();
ArduinoOTA.onError();
ArduinoOTA.begin();

```
void startOTA()
{
//(...)
}

void endOTA()
{
//(...)
}

void progressOTA(unsigned int progress, unsigned int total)
{
//(...)
}

void errorOTA(ota_error_t error)
{
//(...)
}
```

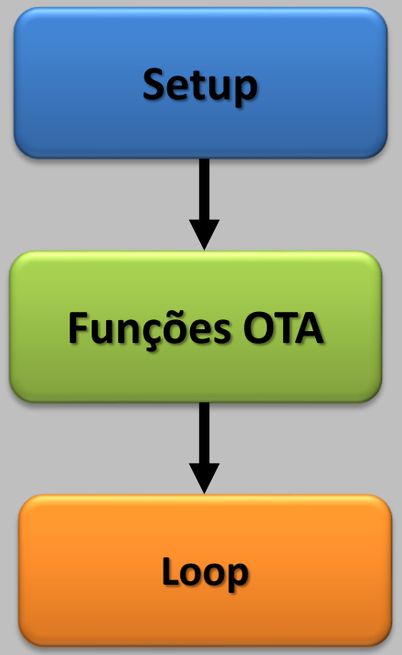
Funções criadas para melhor entendimento do código.

Estas funções são chamadas na configuração do ArduinoOTA (códigos listados acima)dentro de setup.

```
void loop()
{
//(...)
}
```

Obrigatório:

ArduinoOTA.handle();



Setup – ESP32

```
#include <WiFi.h> //lib para configuração do Wifi
#include <ArduinoOTA.h> //lib do ArduinoOTA
#include <ESPmDNS.h> //lib necessária para comunicação network
#include <WiFiUdp.h> //lib necessária para comunicação network

#define ledVerde 23 //este led sinaliza que o ESP foi conectado
//#define ledVermelho 22 //este led é usado como exemplo de atualização

const char* ssid = "robotica"; //nome da rede
const char* password = "12345678"; //senha da rede

void setup()
{
   //define pino como saída
   pinMode(ledVerde,OUTPUT);
   //define pino como saída
   //código da atualização
   //pinMode(ledVermelho,OUTPUT);

   //inicia serial com 115200 bits por segundo
   Serial.begin(115200);
   Serial.println("Booting");

   //define wifi como station (estação)
   WiFi.mode(WIFI_STA);

   //inicializa wifi
   WiFi.begin(ssid, password);
```

Este código será inserido após a primeira gravação do ESP como um teste de atualização do código.

Setup – ESP8266

```
#include <ESP8266WiFi.h> //lib do wifi para o ESP8266
#include <ESP8266WiFiMulti.h> //lib do wifi para o ESP8266
#include <ArduinoOTA.h> //lib do ArduinoOTA

#define ledVerde 23 //este led sinaliza que o ESP foi conectado
//#define ledVermelho 22 //este led é usado como exemplo de atualização

ESP8266WiFiMulti wifiMulti;

const char* ssid = "robotica"; //nome da rede
const char* password = "12345678"; //senha da rede

void setup()
{
   //define pino como saída
   pinMode(ledVerde,OUTPUT);
   //define pino como saída
   //código da atualização
   //pinMode(ledVermelho,OUTPUT);

   //inicia serial com 115200 bits por segundo
   Serial.begin(115200);
   Serial.println("Booting");


   wifiMulti.addAP(ssid, password);

   Serial.println("Connecting ...");
   while (wifiMulti.run() != WL_CONNECTED)
   {
       delay(250);
       Serial.print('.');
   }
```

Mudanças necessárias para o funcionamento do código no ESP8266

Continuação setup – ESP32

```
//enquanto o wifi não for conectado aguarda
while (WiFi.waitForConnectResult() != WL_CONNECTED)
{
//caso falha da conexão, reinicia wifi
Serial.println("Connection Failed! Rebooting...");
delay(5000);
ESP.restart();
}

// A porta fica default como 3232 (caso você esteja usando o ESP8266 o default passa a ser 8266)
// ArduinoOTA.setPort(3232);

// Define o hostname (opcional)
ArduinoOTA.setHostname("myesp32");

// Define a senha (opcional)
ArduinoOTA.setPassword("password123");

// É possível definir uma criptografia hash md5 para a senha usando a função "setPasswordHash"
// Exemplo de MD5 para senha "admin" = 21232f297a57a5a743894a0e4a801fc3
// ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");
```

Setup – Configuração da instância ArduinoOTA

```
//define o que será executado quando o ArduinoOTA iniciar
ArduinoOTA.onStart( startOTA );
//startOTA é uma função criada para simplificar o código

//define o que será executado quando o ArduinoOTA terminar
ArduinoOTA.onEnd( endOTA );
//endOTA é uma função criada para simplificar o código

//define o que será executado quando o ArduinoOTA estiver gravando
ArduinoOTA.onProgress( progressOTA );
//progressOTA é uma função criada para simplificar o código

//define o que será executado quando o ArduinoOTA encontrar um erro
ArduinoOTA.onError( errorOTA );
//errorOTA é uma função criada para simplificar o código

//inicializa ArduinoOTA
ArduinoOTA.begin();

//exibe pronto e o ip utilizado pelo ESP
Serial.println("Ready");
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
digitalWrite(ledVerde,HIGH);
}
```

Funções de exibição dos estágios de upload (start, progress e end)

```
//exibe em qual memória a atualização está sendo gravada
void startOTA()
{
   String type;

   //caso a atualização esteja sendo gravada na memória flash externa, então informa "flash"
   if (ArduinoOTA.getCommand() == U_FLASH)
      type = "flash";
   else //caso a atualização seja feita pela memória interna (file system), então informa "filesystem"
      type = "filesystem"; // U_SPIFFS

   //exibe mensagem junto ao tipo de gravação
   Serial.println("Start updating " + type);
}

//exibe mensagem
void endOTA()
{
   Serial.println("\nEnd");
}

//exibe progresso em porcentagem
void progressOTA(unsigned int progress, unsigned int total)
{
   Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
}
```

Funções de exibição dos estágios de upload (error)

```
//caso aconteça algum erro, exibe especificamente o tipo do erro
void errorOTA(ota_error_t error)
{
      Serial.printf("Error[%u]: ", error);

      if (error == OTA_AUTH_ERROR)
            Serial.println("Auth Failed");
      else
      if (error == OTA_BEGIN_ERROR)
            Serial.println("Begin Failed");
      else
      if (error == OTA_CONNECT_ERROR)
            Serial.println("Connect Failed");
      else
      if (error == OTA_RECEIVE_ERROR)
            Serial.println("Receive Failed");
      else
      if (error == OTA_END_ERROR)
            Serial.println("End Failed");
}
```

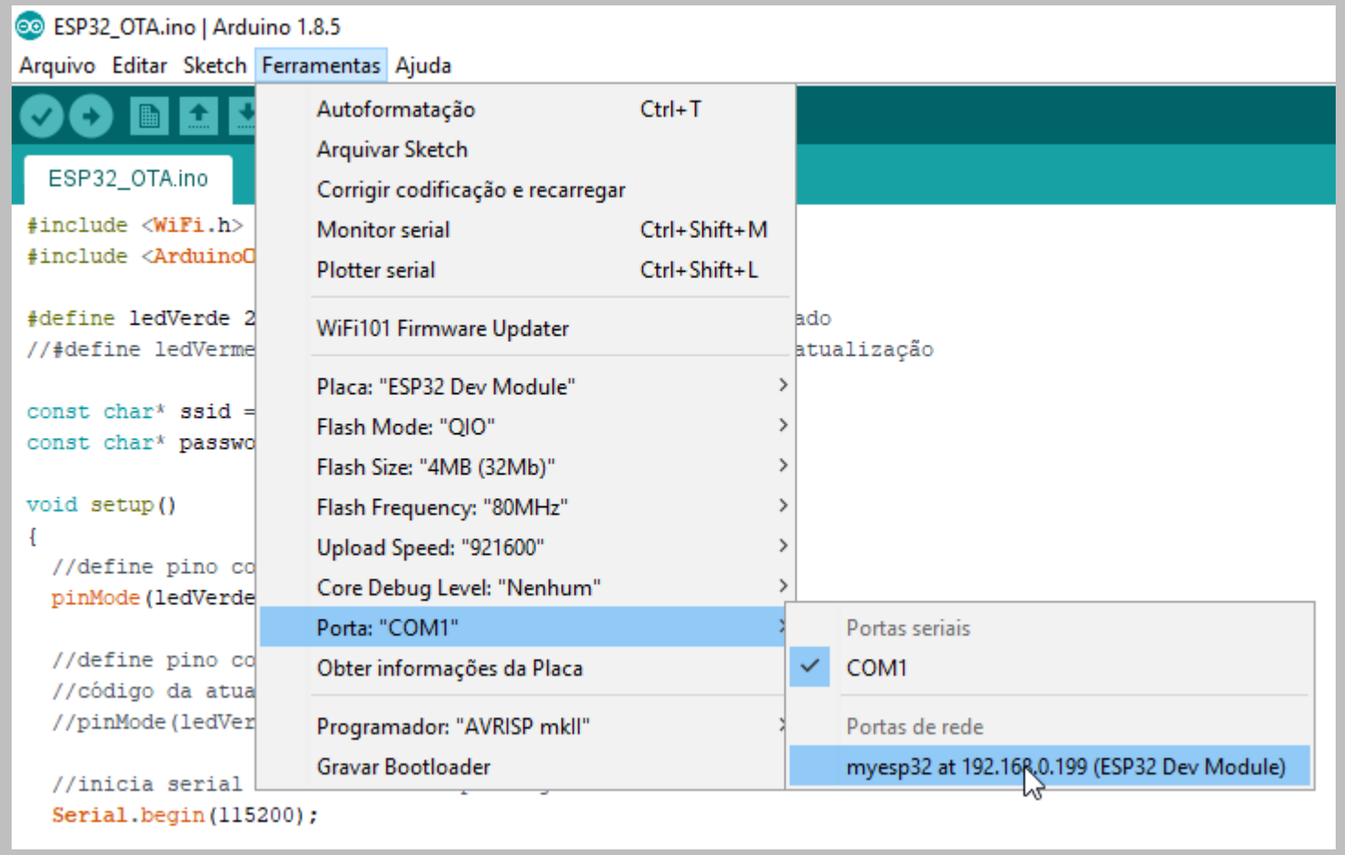
Loop

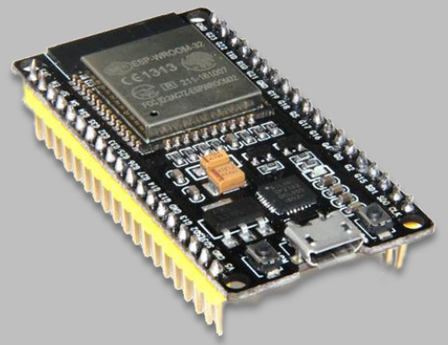
```
void loop()
{
   //Handle é um descritor que referencia variáveis no bloco de memória
   //Ele é usado como um "guia" para que o ESP possa se comunicar com o computador pela rede
   ArduinoOTA.handle();

   //código de atualização
   /*
   digitalWrite(ledVermelho,HIGH);
   delay(1000);
   digitalWrite(ledVermelho,LOW);
   delay(1000);
   */
}
```

Seleção de porta de rede no Arduino IDE

Após a primeira gravação feita pela porta Serial, deve-se selecionar a porta de rede conforme a figura.

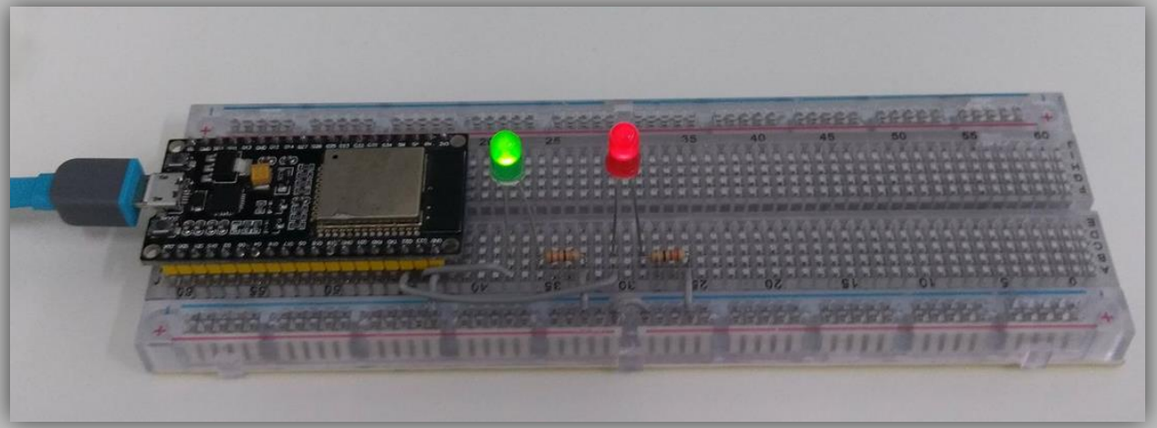
Obs: O computador deve estar conectado na mesma rede que o ESP.

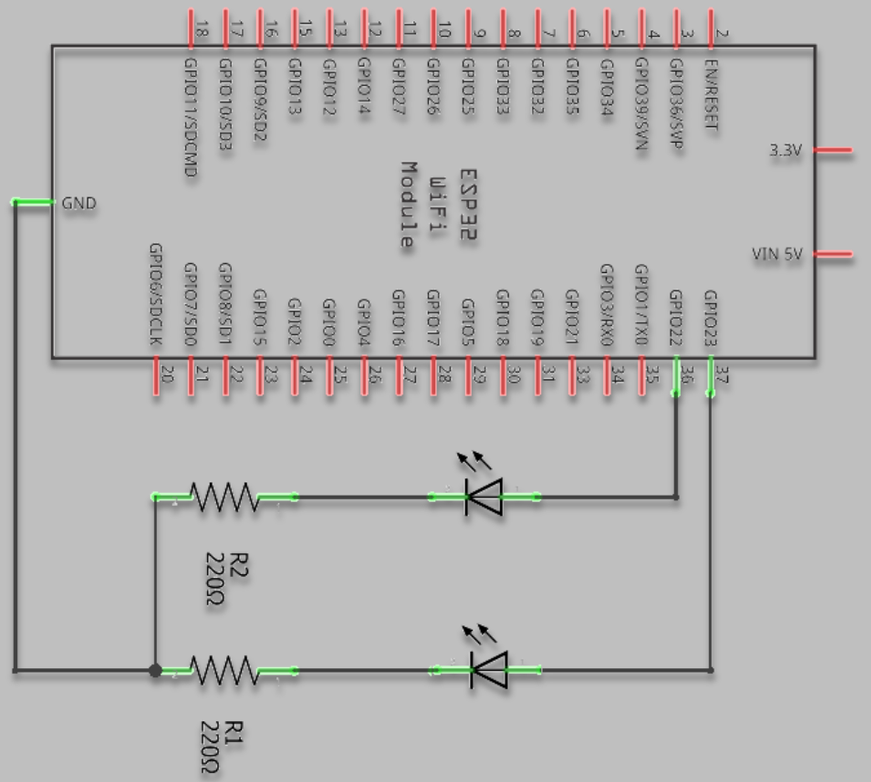




Montagem ESP32

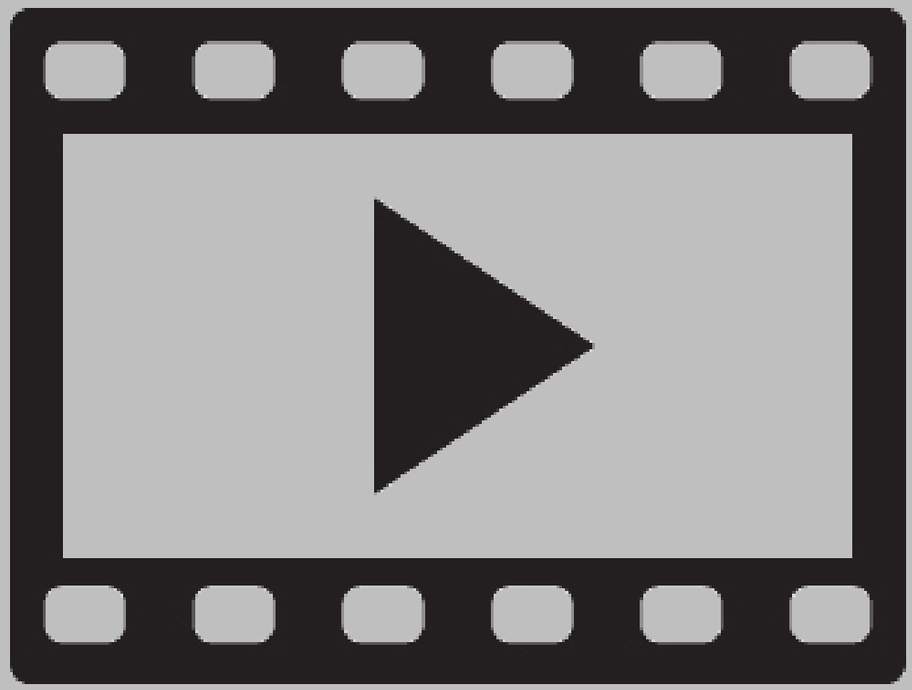
Montagem ESP32





Led verde indica que o ESP está conectado
Led vermelho é usado para visualizar a atualização feita pelo OTA

Demonstração



Em www.fernandok.com

Download arquivos PDF e INO do código fonte