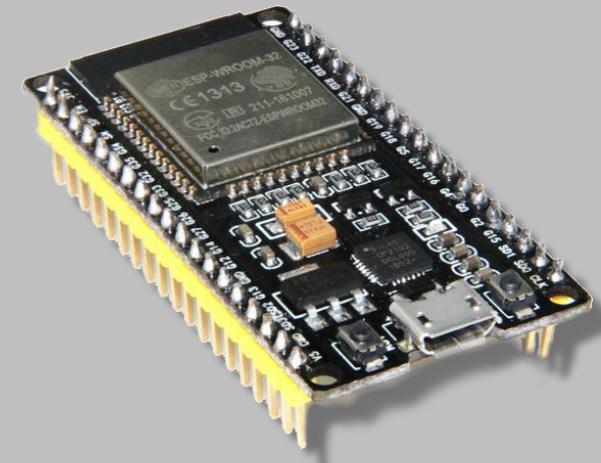
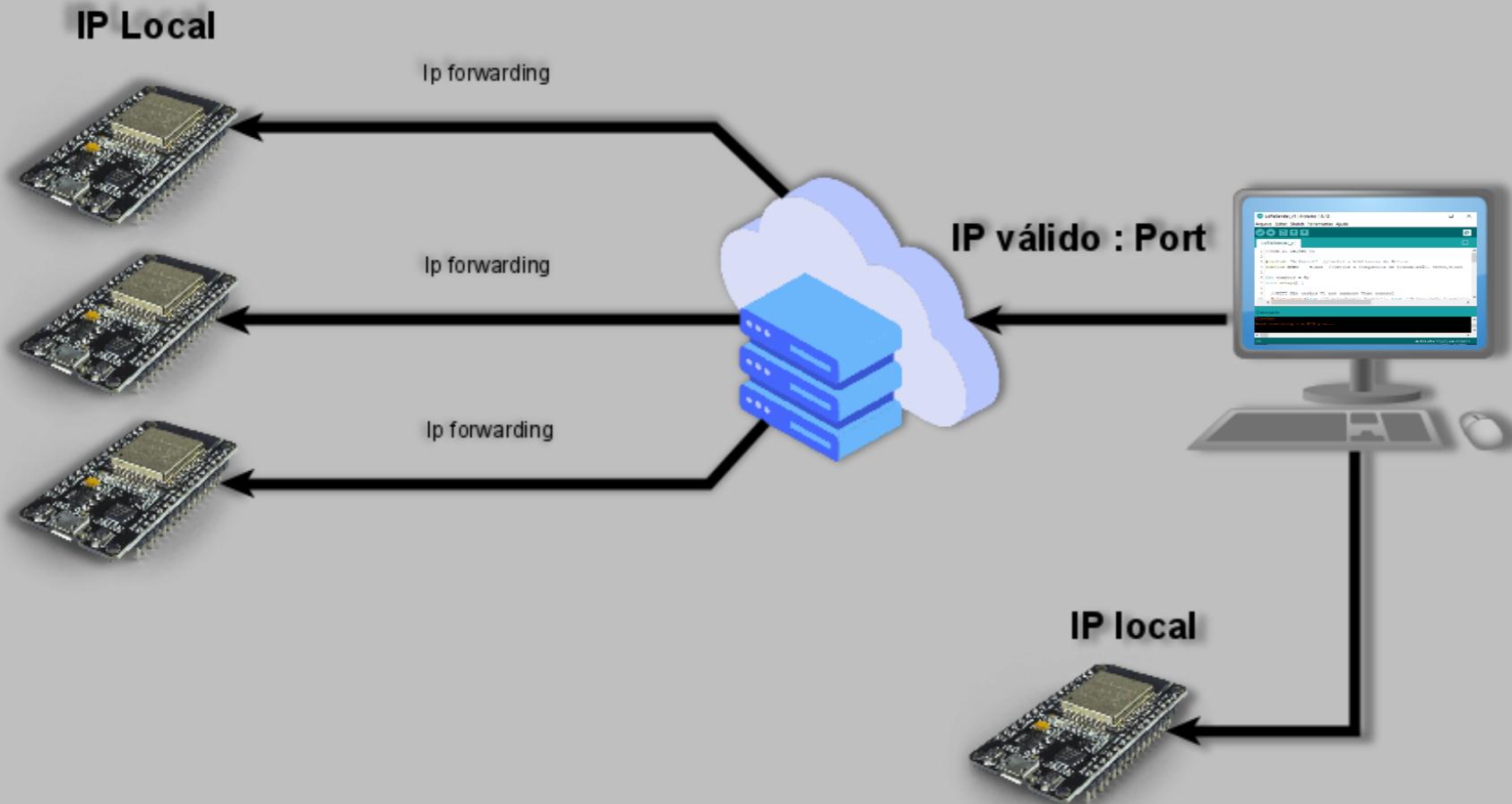


# Gravando o ESP32 pela Internet “OTA”

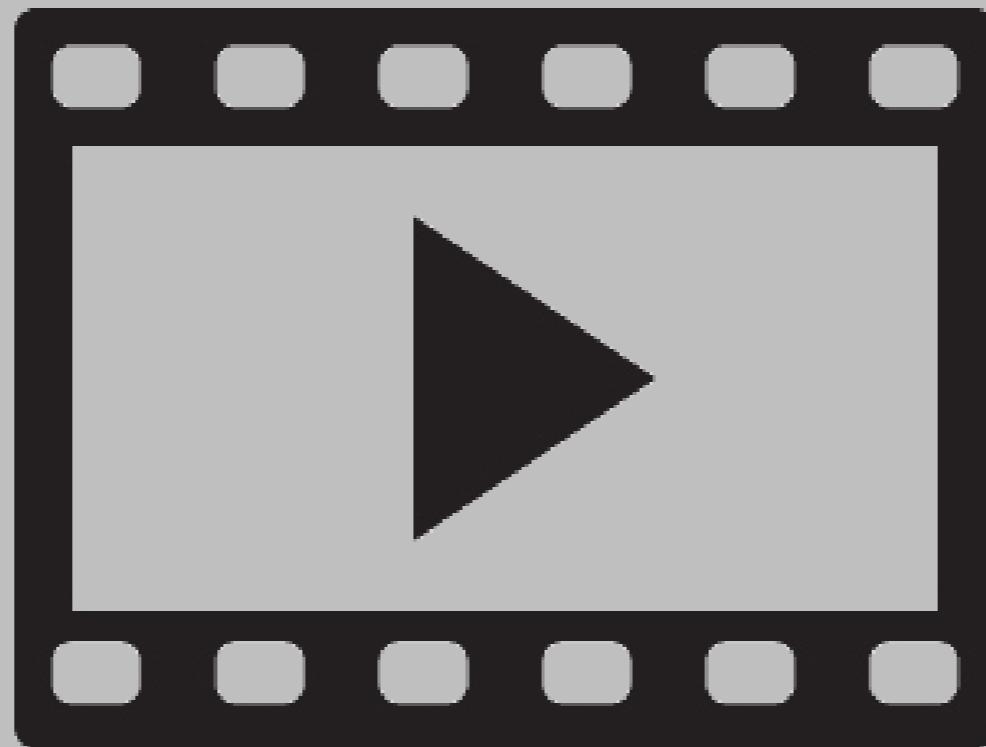


Por Fernando Koyanagi

# **Intenção dessa aula**

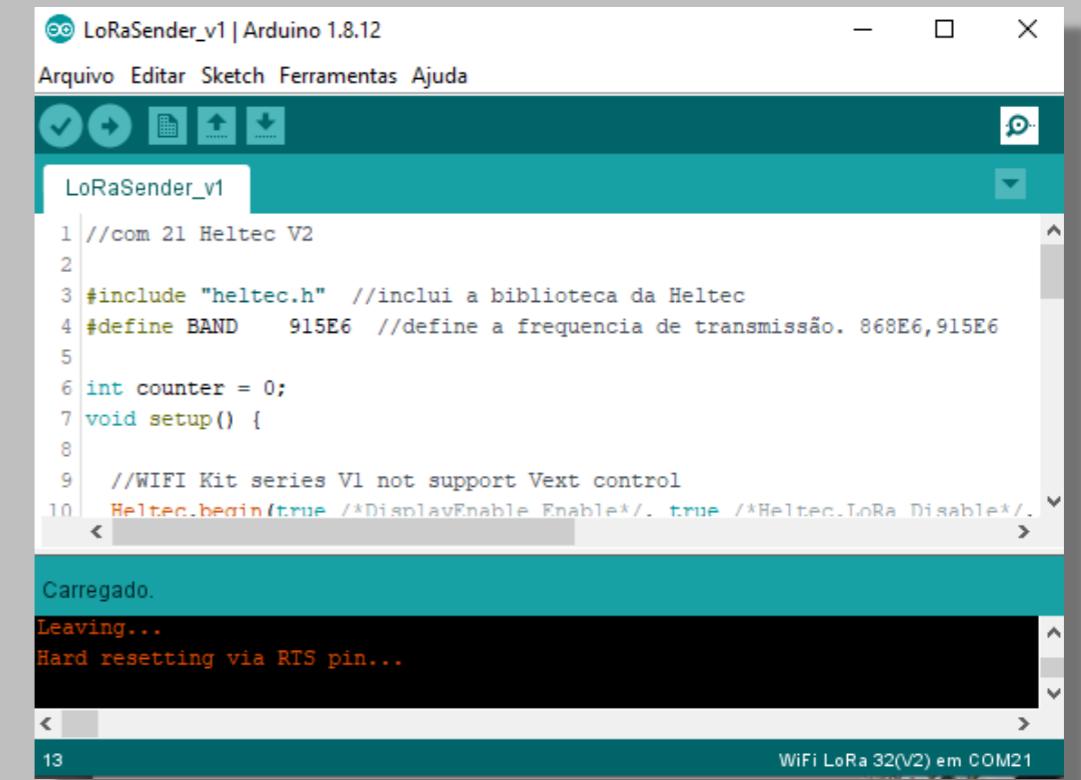
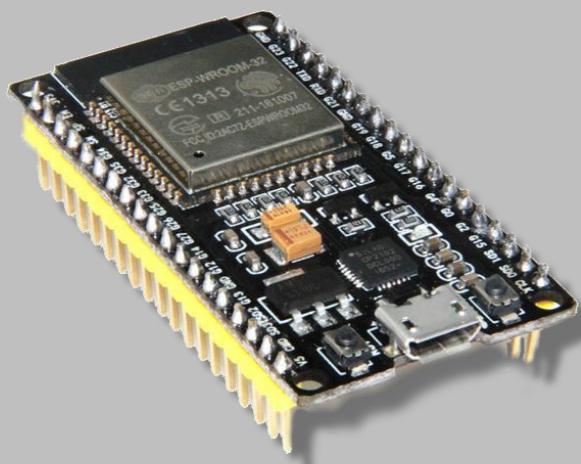
- 1. Oferecer uma opção de OTA com um nível de segurança um pouco superior aos exemplos.**

# Demonstração



# Recursos usados

- 1x ESP Wroom 32.
- 1x WiFi
- Arduino IDE



The image shows the Arduino IDE interface with a sketch named "LoRaSender\_v1". The code is as follows:

```
//com 21 Heltec V2
#include "heltec.h" //inclui a biblioteca da Heltec
#define BAND 915E6 //define a frequencia de transmissao. 868E6,915E6
int counter = 0;
void setup() {
    //WIFI Kit series V1 not support Vext control
    Heltec.begin(true /*DisplayEnable Enable*/, true /*Heltec.Lora Disable*/);
}
Carregado.
Leaving...
Hard resetting via RTS pin...
```

The status bar at the bottom right indicates "WiFi LoRa 32(V2) em COM21".



Tutoriais  
Tecnologia  
Tendências

FÓRUM SOBRE ARDUINO ESP8266 ESP32 LORAWAN STM32 MOTOR DISPLAY MATERIAL PARA DOWNLOAD

Receba o meu conteúdo  
GRATUITAMENTE

Insira aqui seu melhor email...

QUERO RECEBER GRÁTIS



Se eu soubesse disso antes!

by Fernando K - 12 novembro

Voltamos a falar de instrumentação e, hoje, vamos falar de um módulo de obtenção de parâmetros elétricos , que é o RIDEN 100V/10A . ...

[Leia mais](#)



IOT mais barata do mundo com ESP8266

by Fernando K - 05 novembro

Automação utilizando o ESP8266 . Hoje quero te apresentar um exemplo de baixo custo utilizando este modelo de microcontrolador co...

[Leia mais](#)



Osciloscópio 100MHz portátil: Incrivelmente barato!

by Fernando K - 29 outubro

Conhece o osciloscópio ADS5012H – Danii ? Pois, eu ganhei um da Banggood e gostei demais. Primeiro: ele é barato . Segundo: a taxa de ...

[Leia mais](#)



Spoiler do curso de IOT

by Fernando K - 22 outubro

Uma parte do curso de IOT que estou desenvolvendo. É o que mostro hoje para vocês: uma aula sobre Multitask , ou seja, sobre como ser pr...

[Leia mais](#)

Supergrupo de colaboração entre meus seguidores.

Conteúdo exclusivo, que não tem no YouTube!

INSTAGRAM @FERNANDOK\_OFICIAL

FACEBOOK



INSCREVA-SE NO YOUTUBE



Fernando K Tecnologia  
 YouTube 999+

**RECENTE** **POPULAR** **COMENTÁRIOS**

Se eu soubesse disso antes!  
Nov 12, 2019

## SEJA MEMBRO



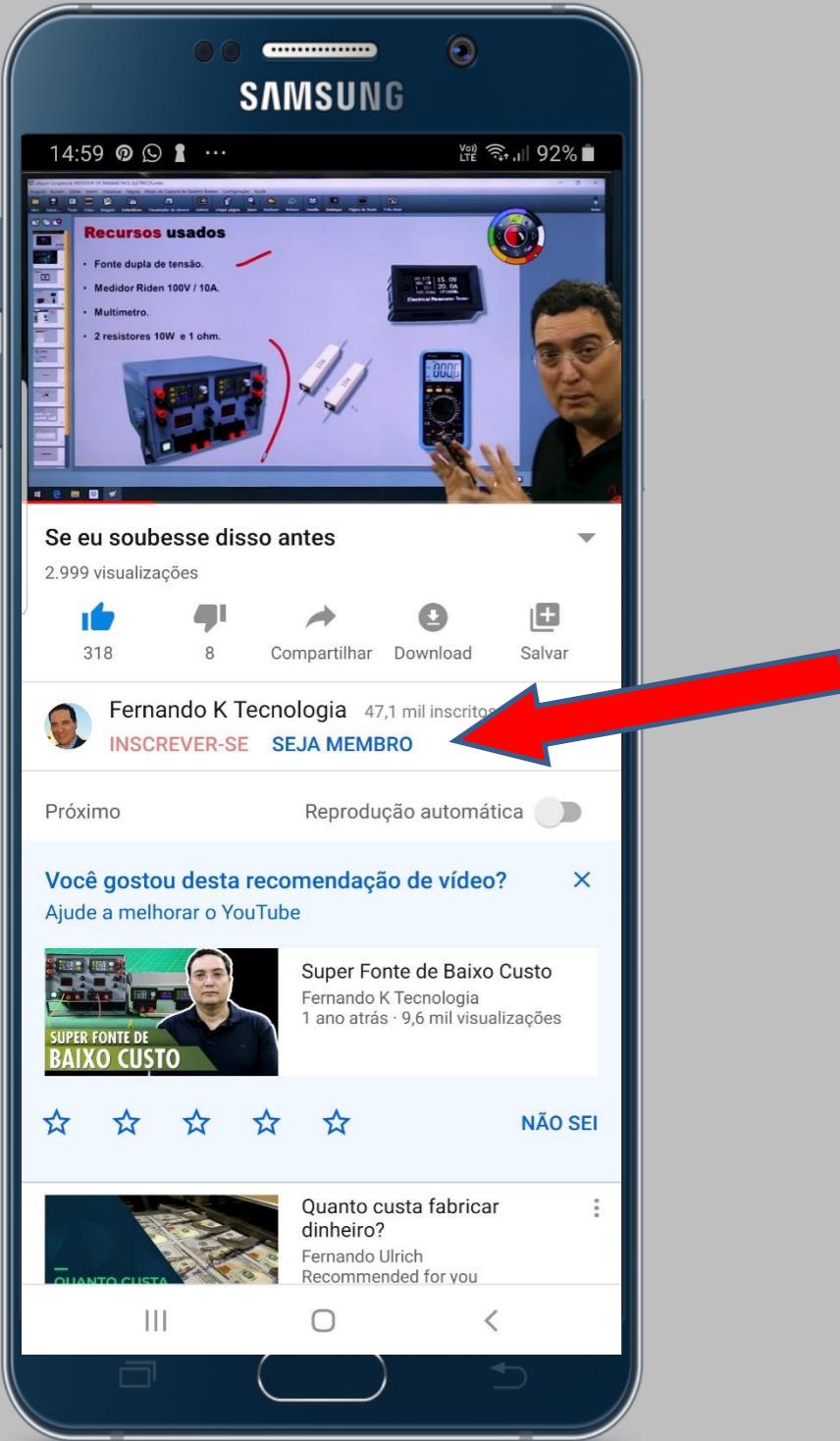
# Links onde comprei os componentes

<https://bit.ly/2VM1ZdQ>

Em [www.fernandok.com](http://www.fernandok.com)



SEJA MEMBRO



# SEJA MEMBRO

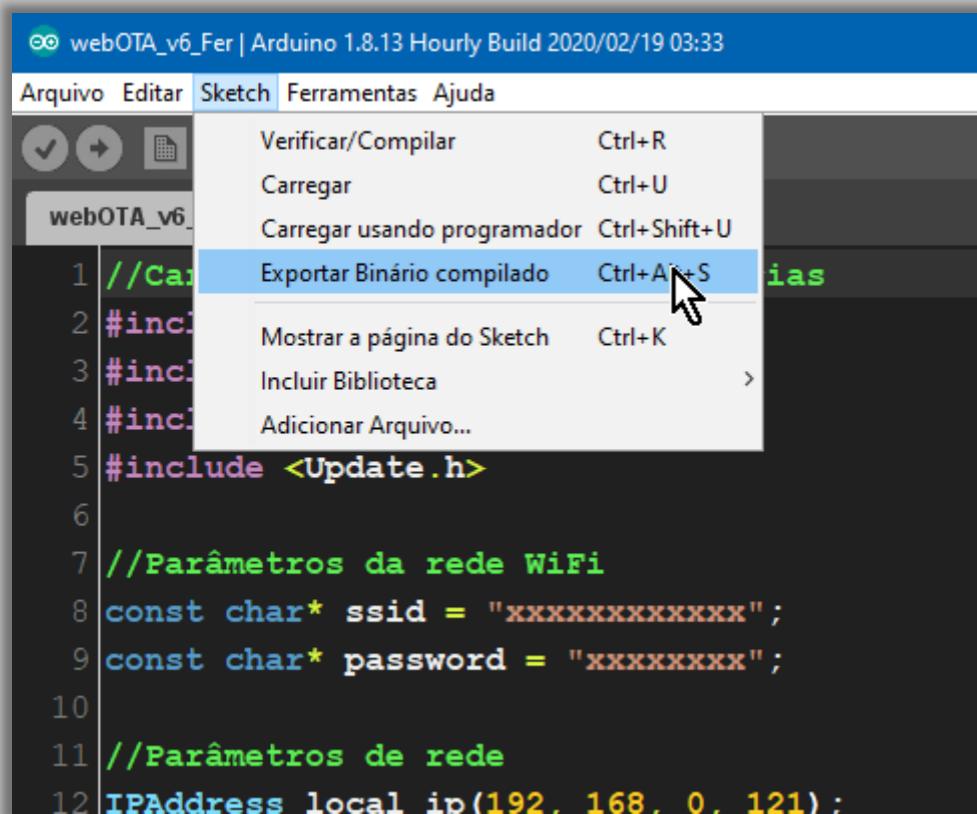
# **Sequência de verificação**

# **Sequência de verificação**

Neste exemplo de OTA, aplicaremos uma sequência de verificação antes de iniciar a atualização de fato. Ela será baseada na verificação de **algum código de autorização**, enviado através da página **html inicial**, e verificado pelo próprio ESP.

**Se a verificação falhar, o ESP não servirá a página para seleção de arquivo e impedirá a continuidade do processo.**

# Gerando o .bin da ser gravado



webOTA\_v6\_Fer | Arduino 1.8.13 Hourly Build 2020/02/19 03:33

Arquivo Editar Sketch Ferramentas Ajuda

Verificar/Compilar Ctrl+R  
Carregar Ctrl+U  
Carregar usando programador Ctrl+Shift+U  
Exportar Binário compilado Ctrl+Alt+S  
Mostrar a página do Sketch Ctrl+K  
Incluir Biblioteca >  
Adicionar Arquivo...  
**//Cabeçalhos**  
#include "Update.h"  
  
//Parâmetros da rede WiFi  
const char\* ssid = "xxxxxxxxxxxx";  
const char\* password = "xxxxxxxx";  
  
//Parâmetros de rede  
IPAddress local\_ip(192, 168, 0, 121);

Nome	Data de modificação	Tipo	Tamanho
webOTA_v6_Fer.ino	05/05/2020 07:10	Arquivo INO	7 KB
webOTA_v6_Fer.ino.esp32.bin	05/05/2020 07:10	Arquivo BIN	713 KB

# **Conexão externa**

# Conexão externa

**Redirecionamento de Portas - Servidores Virtuais**

ID	Porta de Serviço	Porta Interna	Endereço IP	Protocolo	Estado	Modificar
1	3000	3000	192.168.0.105	Tudo	Habilitado	<a href="#">Modificar</a> <a href="#">Apagar</a>

[Adicionar](#) [Habilitar todos](#) [Desabilitar](#) [Apagar todos](#)

[Voltar](#) [Avançar](#)

ESP32 webOTA x +

← → C 186.xxx.yyy.zzz:3000

## ESP32 webOTA

Digite a chave de verificação.

Clique em ok para continuar . . .

Versão: 2.0

Chip ID: 16422

Autorização:

Ok

É possível também efetuar uma conexão através da internet. Para isso, mantenha o ESP com um endereço fixo e redirecione seu endereço IP e sua porta usando os recursos disponíveis no seu roteador.

Para se conectar, basta utilizar o endereço fornecido pelo seu provedor, para sua conexão WAN, e apontar para a porta roteada.

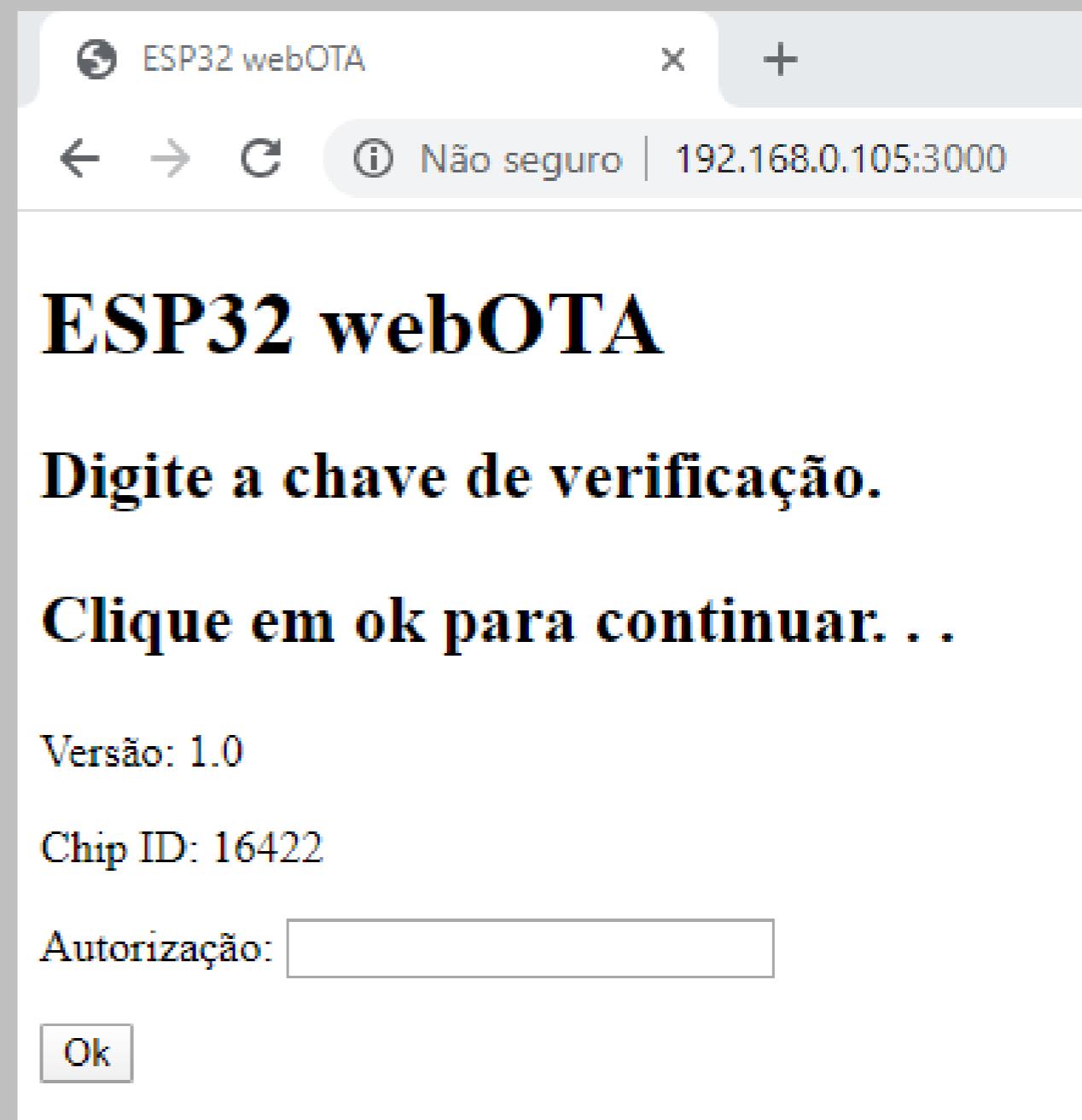
Nem todos os roteadores possuem essa função e as particularidades da rede em questão devem ser levadas em consideração.

# Sequência de verificação

Ao acessar o endereço IP do ESP a ser atualizado, ele servirá a primeira página.

No código de exemplo, introduzimos nesta página algumas informações sobre o chip e a versão do código gravado. Aqui elas servem simplesmente para exemplificar que isso é possível e pode ser bastante útil.

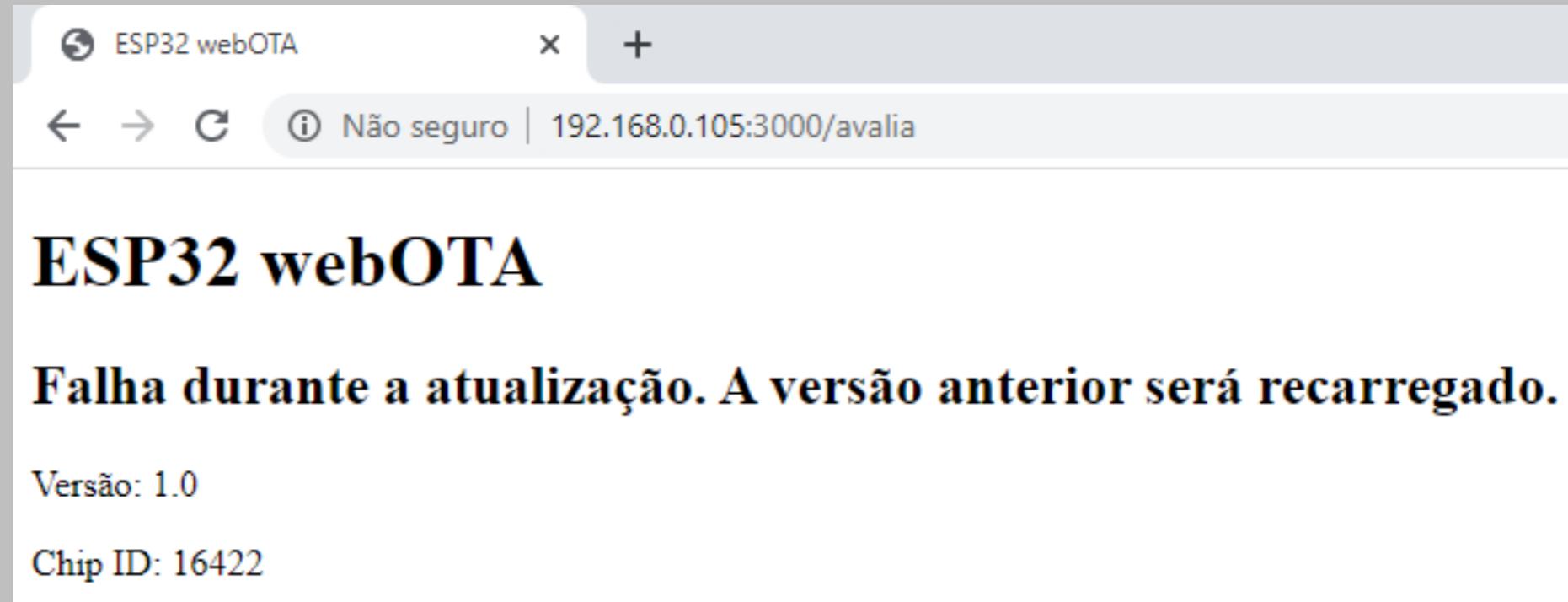
Em seguida, observamos a solicitação de uma chave de autorização.



# Sequência de verificação

Se uma chave incorreta for introduzida, o ESP servirá uma página indicando o um ERRO no processo de atualização.

Isso ocorre porque a chave de verificação enviada não correspondeu com a esperada pelo código do ESP.

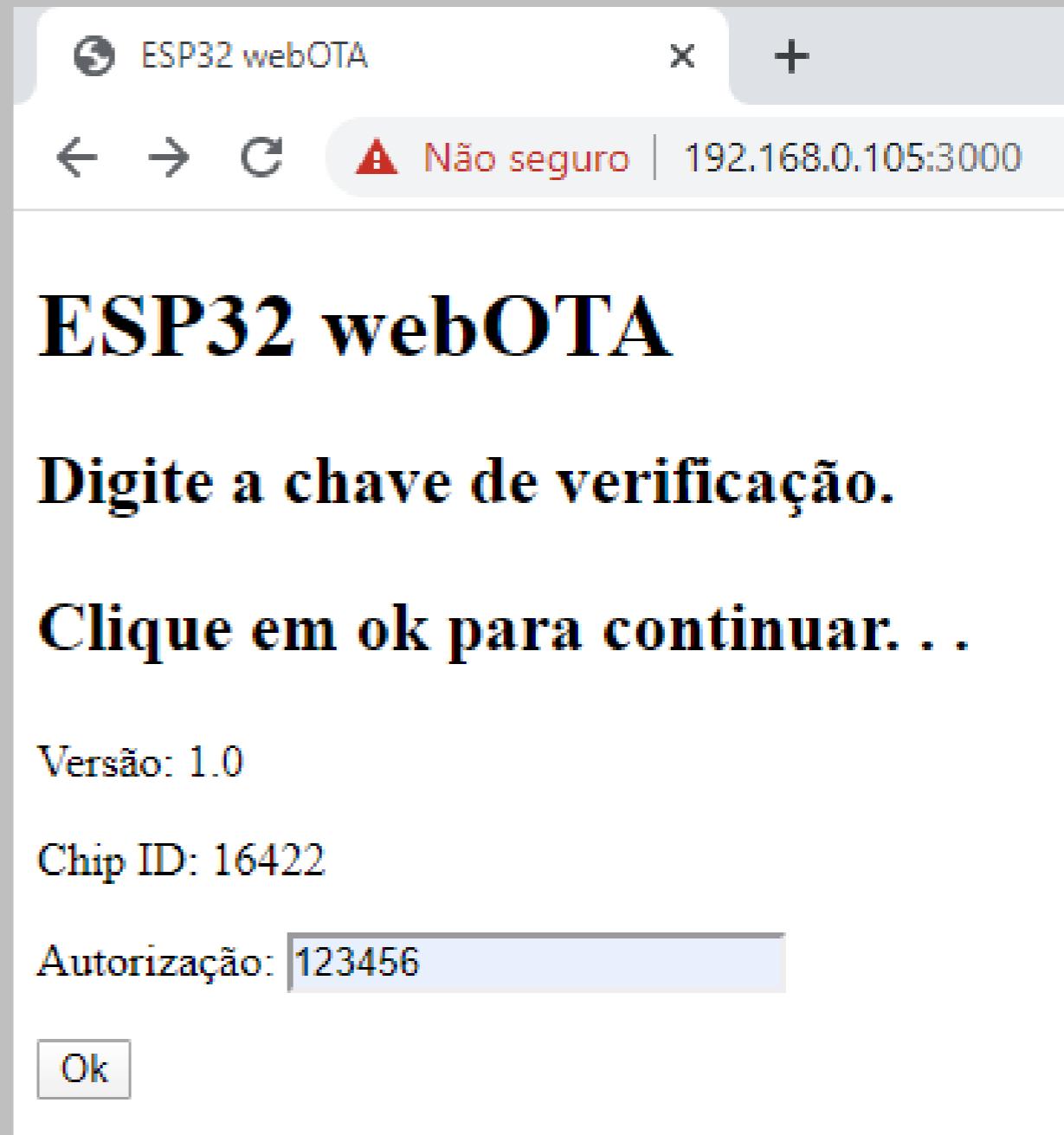


# Sequência de verificação

A chave de verificação dependerá do algoritmo de verificação implementado pelo desenvolvedor. Neste exemplo colocamos uma verificação bastante simples. Basta que a chave seja "123456" para ser aceita.

Uma forma interessante seria usar, por exemplo, uma verificação que evolvesse o id do chip. Isso traria variedade a proteção.

Outro detalhe é que neste exemplo, usamos uma caixa de texto comum, para facilitar a visualização. Mas poderíamos ter utilizado uma caixa de senha.

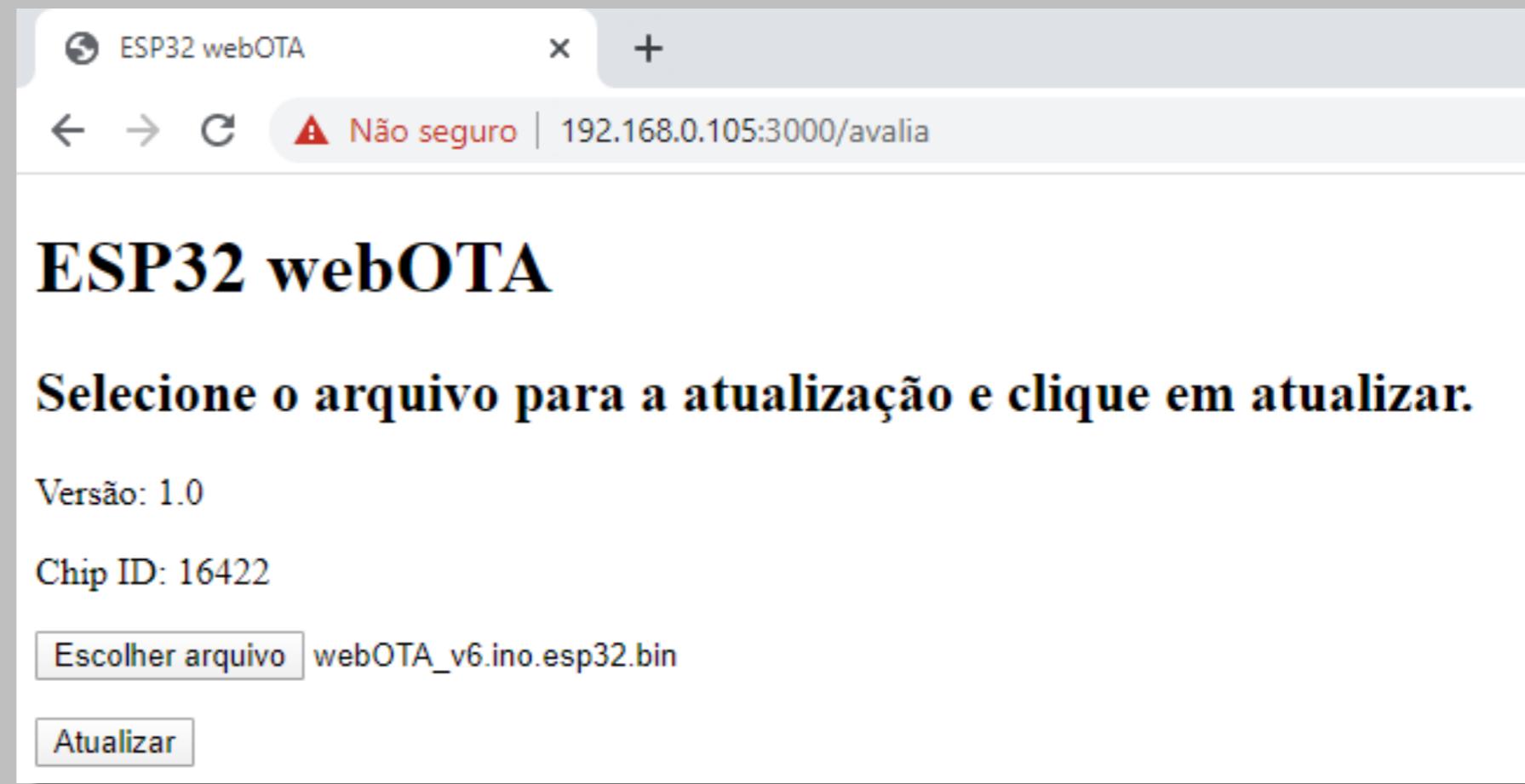


# Sequência de verificação

Se a verificação da chave foi positiva, o ESP servirá a página de seleção do arquivo para upload.

Se um arquivo inválido for enviado, a página de ERRO será servida. Caso contrário, uma página de conclusão deverá ser enviada.

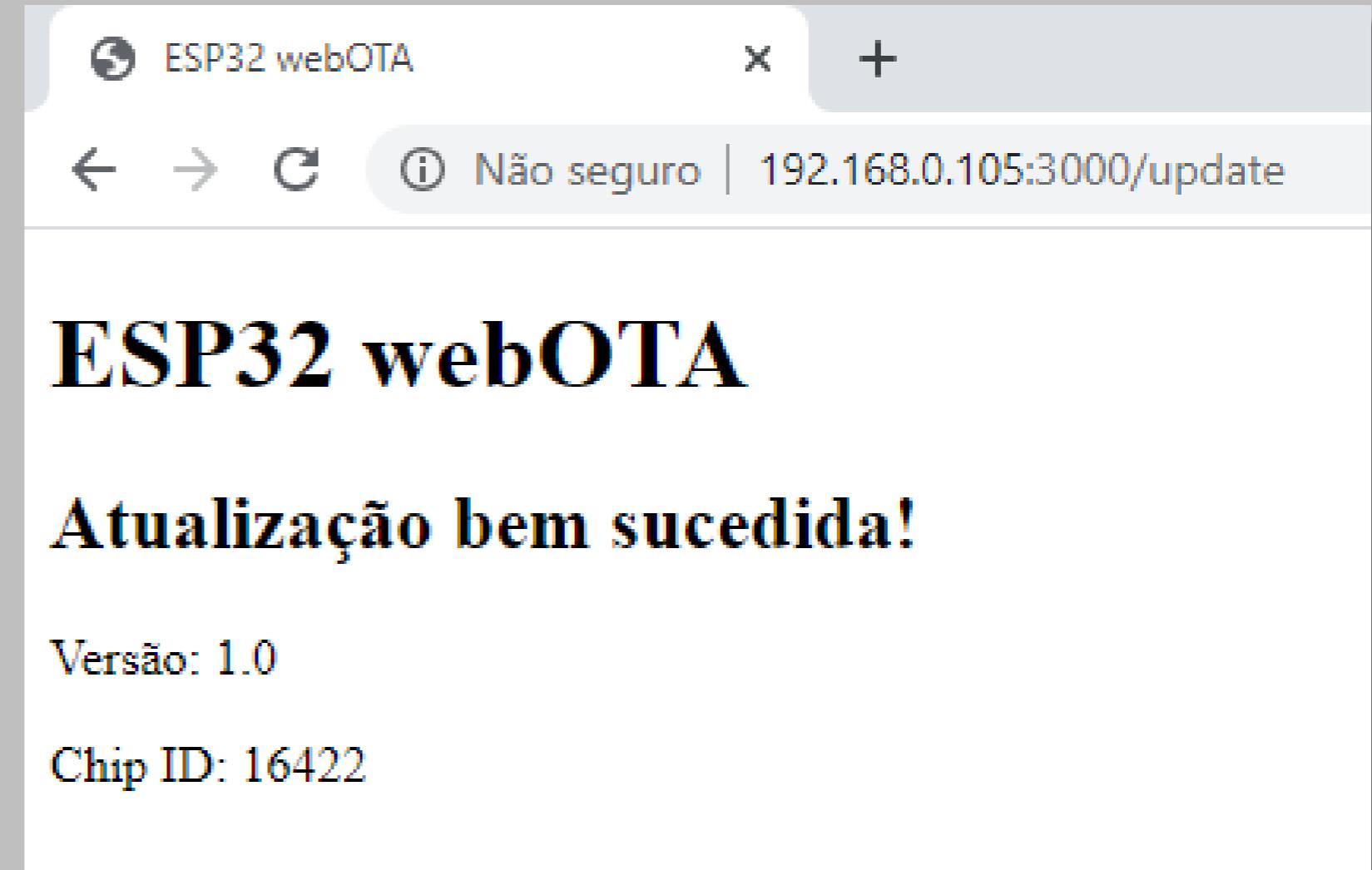
Em alguns casos, devido ao reinicio do ESP (reset), a conexão com o navegador pode ser perdida e a página de resposta pode não ser recebida. Na dúvida, reenvie o arquivo.



# Sequência de verificação

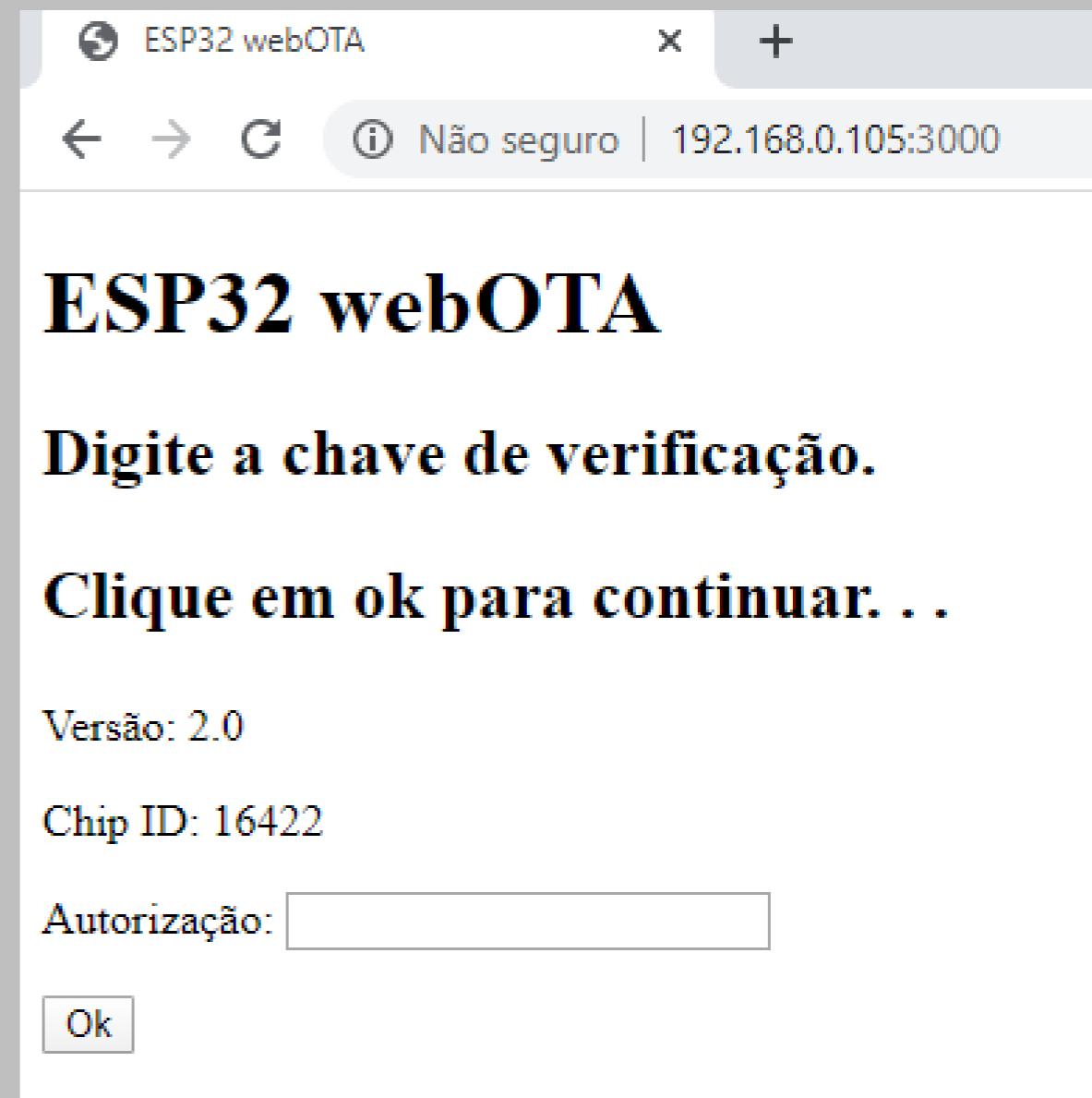
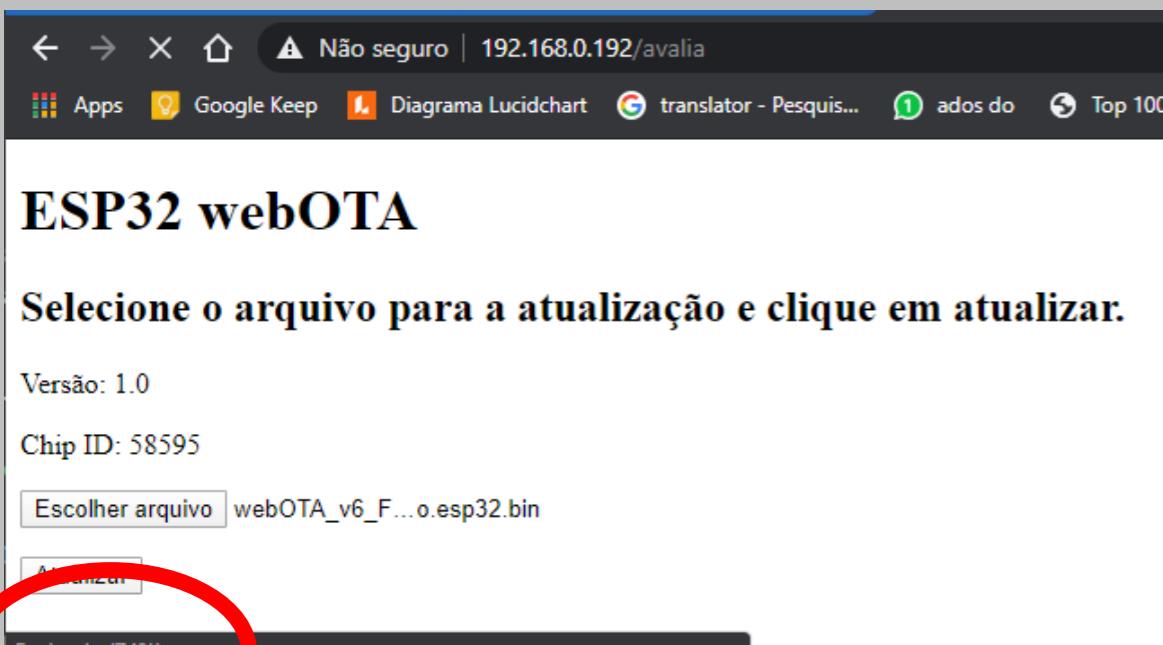
Esta é a página recebida depois da conclusão bem sucedida da atualização.

Note que a versão indica por ela ainda é a mesma. Isso porque ela é enviada antes do reinicio do ESP.



# Sequência de verificação

Mas se carregarmos novamente a página inicial, poderemos observar a alteração da versão que introduzimos nas informações do código.



# Sequência de verificação

Atenção: observe o código fonte a seguir para perceber que essa atualização do número da versão foi introduzida para que fosse possível verificar que um novo código foi gravado no ESP, mas não existe nenhum gerenciamento automático de versão.

```
const String VERSION = "<p> Versão: 1.0 </p>"; //Exemplo de um controle de versão
```



```
const String VERSION = "<p> Versão: 2.0 </p>"; //Exemplo de um controle de versão
```

# Sequência de verificação

Podemos acompanhar todo processo também pela serial, para debug.

Ao lado vemos as mensagens iniciais do ESP. Além das informações padrão, enviamos também a versão e o id do chip.

```
COM11
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
flash read err, 1000
ets_main.c 371
ets Jun  8 2016 00:22:57

rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:9720
ho 0 tail 12 room 4
load:0x40080400,len:6352
entry 0x400806b8
<p> Versão: 1.0 </p><p> Chip ID: 16422</p>
SERVIDOR EM: 192.168.0.105:5000
```

# Sequência de verificação

Em seguida, podemos acompanhar pela serial:

- O recebimento do número de verificação que enviamos através da página.
- O início da atualização, indicando o nome do arquivo selecionado.
- Uma mensagem de atualização bem sucedida.
- Uma mensagem indicando que o ESP será reiniciado.
- E um novo conjunto de informações iniciais indicando a nova versão do firmware.

```
<p> Versão: 1.0 </p><p> Chip ID: 16422</p>
Servidor em: 192.168.0.105:3000
Em server.on /avalia: args= 123456
Atualizando: webOTA_v6.ino.esp32.bin
Atualização bem sucedida! 729312
Reiniciando...
ets Jun 8 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:9720
ho 0 tail 12 room 4
load:0x40080400,len:6352
entry 0x400806b8
<p> Versão: 2.0 </p><p> Chip ID: 16422</p>
Servidor em: 192.168.0.105:3000
```

# **Código-fonte**

# Declarações

```
//Carrega as bibliotecas necessárias
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <Update.h>

//Parâmetros da rede WiFi
const char* ssid = "SEU_SSID";
const char* password = "SUA_SENHA";

//Parâmetros de rede
IPAddress local_ip(192, 168, 1, 105);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
const uint32_t PORTA = 3000; //A porta que será utilizada (padrão 80)

//Algumas informações que podem ser interessantes
const uint32_t chipID = (uint32_t)(ESP.getEfuseMac() >> 32); //um ID exclusivo do Chip...
const String CHIP_ID = "<p> Chip ID: " + String(chipID) + "</p>"; // montado para ser usado no HTML
const String VERSION = "<p> Versão: 1.0 </p>"; //Exemplo de um controle de versão
```

# Declarações

```
//Informações interessantes agrupadas
const String INFOS = VERSION + CHIP_ID;

//Sinalizador de autorização do OTA
boolean OTA_AUTORIZADO = false;

//inicia o servidor na porta selecionada
//aqui testamos na porta 3000, ao invés da 80 padrão
WebServer server(PORTA);

//Páginas HTML utilizadas no procedimento OTA
String verifica = "<!DOCTYPE html><html><head><title>ESP32 webOTA</title><meta charset='UTF-8'></head><body><h1>ESP32 webOTA</h1><h2>Digite a chave de verificação.<p>Clique em ok para continuar. . .</p></h2>" + INFOS + "<form method='POST' action='/avalia' enctype='multipart/form-data'> <p><label>Autorização:</label><input type='text' name='autorizacao'></p><input type='submit' value='Ok'></form></body></html>";
String serverIndex = "<!DOCTYPE html><html><head><title>ESP32 webOTA</title><meta charset='UTF-8'></head><body><h1>ESP32 webOTA</h1><h2>Selecione o arquivo para a atualização e clique em atualizar.</h2>" + INFOS + "<form method='POST' action='/update' enctype='multipart/form-data'><p><input type='file' name='update'></p><p><input type='submit' value='Atualizar'></p></form></body></html>";
String Resultado_Ok = "<!DOCTYPE html><html><head><title>ESP32 webOTA</title><meta charset='UTF-8'></head><body><h1>ESP32 webOTA</h1><h2>Atualização bem sucedida!</h2>" + INFOS + "</body></html>";
String Resultado_Falha = "<!DOCTYPE html><html><head><title>ESP32 webOTA</title><meta charset='UTF-8'></head><body><h1>ESP32 webOTA</h1><h2>Falha durante a atualização. A versão anterior será recarregado.</h2>" + INFOS + "</body></html>";
```

# Declarações – A páginas HTML (verifica)

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>ESP32 webOTA</title>
5      <meta charset='UTF-8'>
6  </head>
7  <body>
8      <h1>ESP32 webOTA</h1>
9      <h2>Digite a chave de verificação.
10         <p>Clique em ok para continuar. . .</p>
11     </h2>""
12 + INFOS +
13 <form method='POST' action='/avalia' enctype='multipart/form-data'>
14     <p>
15         <label>Autorização: </label>
16         <input type='text' name='autorizacao'>
17     </p>
18     <input type='submit' value='Ok'>
19 </form>
20 </body>
21 </html>";
```

# Declarações – A páginas HTML (serverIndex)

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>ESP32 webOTA</title>
5      <meta charset='UTF-8'>
6  </head>
7  <body>
8      <h1>ESP32 webOTA</h1>
9      <h2>Selecione o arquivo para a atualização e clique em atualizar.</h2>
10     + INFOS +
11     <form method='POST' action='/update' enctype='multipart/form-data'>
12         <p>
13             <input type='file' name='update'>
14         </p>
15         <p>
16             <input type='submit' value='Atualizar'>
17         </p>
18     </form>
19 </body>
20 </html>";|
```

# Declarações – A páginas HTML (Resultado\_Ok)

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>ESP32 webOTA</title>
5      <meta charset='UTF-8'>
6  </head>
7  <body>
8      <h1>ESP32 webOTA</h1>
9      <h2>Atualização bem sucedida!</h2>
10     + INFOS +
11  </body>
12 </html>
```

# Declarações – A páginas HTML (Resultado\_Falha)

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>ESP32 webOTA</title>
5      <meta charset='UTF-8'>
6  </head>
7  <body>
8      <h1>ESP32 webOTA</h1>
9      <h2>Falha durante a atualização. A versão anterior será recarregado.</h2>
10     + INFOS +
11 </body>
12 </html>
```

# Setup()

```
void setup(void)
{
    Serial.begin(115200); //Serial para debug

    WiFi.mode(WIFI_AP_STA); //Configura o ESP32 como ponto de acesso e estação

    WiFi.begin(ssid, password); // inicia a conexão com o WiFi

    if (WiFi.waitForConnectResult() == WL_CONNECTED) //aguarda a conexão
    {
        //atende uma solicitação para a raiz
        // e devolve a página 'verifica'
        server.on("/", HTTP_GET, []() //atende uma solicitação para a raiz
        {
            server.sendHeader("Connection", "close");
            server.send(200, "text/html", verifica);
        });
    }
}
```

# Setup()

```
//atende uma solicitação para a página avalia
server.on("/avalia", HTTP_POST, [] ()
{
    Serial.println("Em server.on /avalia: args= " + String(server.arg("autorizacao"))); //somente para debug

    if (server.arg("autorizacao") != "123456") // confere se o dado de autorização atende a avaliação
    {
        //se não atende, serve a página indicando uma falha
        server.sendHeader("Connection", "close");
        server.send(200, "text/html", Resultado_Falha);
        //ESP.restart();
    }
    else
    {
        //se atende, solicita a página de índice do servidor
        // e sinaliza que o OTA está autorizado
        OTA_AUTORIZADO = true;
        server.sendHeader("Connection", "close");
        server.send(200, "text/html", serverIndex);
    }
});
```

# Setup()

```
//serve a página de indice do servidor
//para seleção do arquivo
server.on("/serverIndex", HTTP_GET, []()
{
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", serverIndex);
});

//tenta iniciar a atualização . . .
server.on("/update", HTTP_POST, []()
{
    //verifica se a autorização é false.
    //Se for falsa, serve a página de erro e cancela o processo.
    if (OTA_AUTORIZADO == false)
    {
        server.sendHeader("Connection", "close");
        server.send(200, "text/html", Resultado_Falha);
        return;
    }
    //Serve uma página final que depende do resultado da atualização
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", (Update.hasError()) ? Resultado_Falha : Resultado_Ok);
    delay(1000); //aguarda para envio do resultado
    ESP.restart();
}, []()
```

# Setup()

```
{  
    //Mas estiver autorizado, inicia a atualização  
    HTTPUpload& upload = server.upload();  
    if (upload.status == UPLOAD_FILE_START)  
    {  
        Serial.setDebugOutput(true);  
        Serial.printf("Atualizando: %s\n", upload.filename.c_str());  
        if (!Update.begin())  
        {  
            //se a atualização não iniciar, envia para serial mensagem de erro.  
            Update.printError(Serial);  
        }  
    }  
    else if (upload.status == UPLOAD_FILE_WRITE)  
    {  
        if (Update.write(upload.buf, upload.currentSize) != upload.currentSize)  
        {  
            //se não conseguiu escrever o arquivo, envia erro para serial  
            Update.printError(Serial);  
        }  
    }  
}
```

# Setup()

```
else if (upload.status == UPLOAD_FILE_END)
{
    if (Update.end(true))
    {
        //se finalizou a atualização, envia mensagem para a serial informando
        Serial.printf("Atualização bem sucedida! \u263a\nReiniciando...\n", upload.totalSize);
    }
    else
    {
        //se não finalizou a atualização, envia o erro para a serial.
        Update.printError(Serial);
    }
    Serial.setDebugOutput(false);
}
else
{
    //se não conseguiu identificar a falha no processo, envia uma mensagem para a serial
    Serial.printf("Atualização falhou inesperadamente! (possivelmente a conexão foi perdida.): status=%d\n",
upload.status);
}
```

# Setup()

```
server.begin(); //inicia o servidor

Serial.println(INFOS); //envia as informações armazenadas em INFOs, para debug

//Envia para a serial o IP atual do ESP
Serial.print("Servidor em: ");
Serial.println( WiFi.localIP().toString() + ":" + PORTA);
}

else
{
    //avisa se não onseguir conectar no WiFi
    Serial.println("Falha ao conectar ao WiFi.");
}
}
```

# Loop()

```
void loop(void)
{
    //manipula clientes conectados
    server.handleClient();

    delay(1); //somente um instante
}
```

**Em www.fernandok.com**

**Download arquivo PDF e INO**

