

Guia Arduino para Iniciantes: tudo que você precisa saber para começar



Preencha abaixo para receber o material:

Baixar material

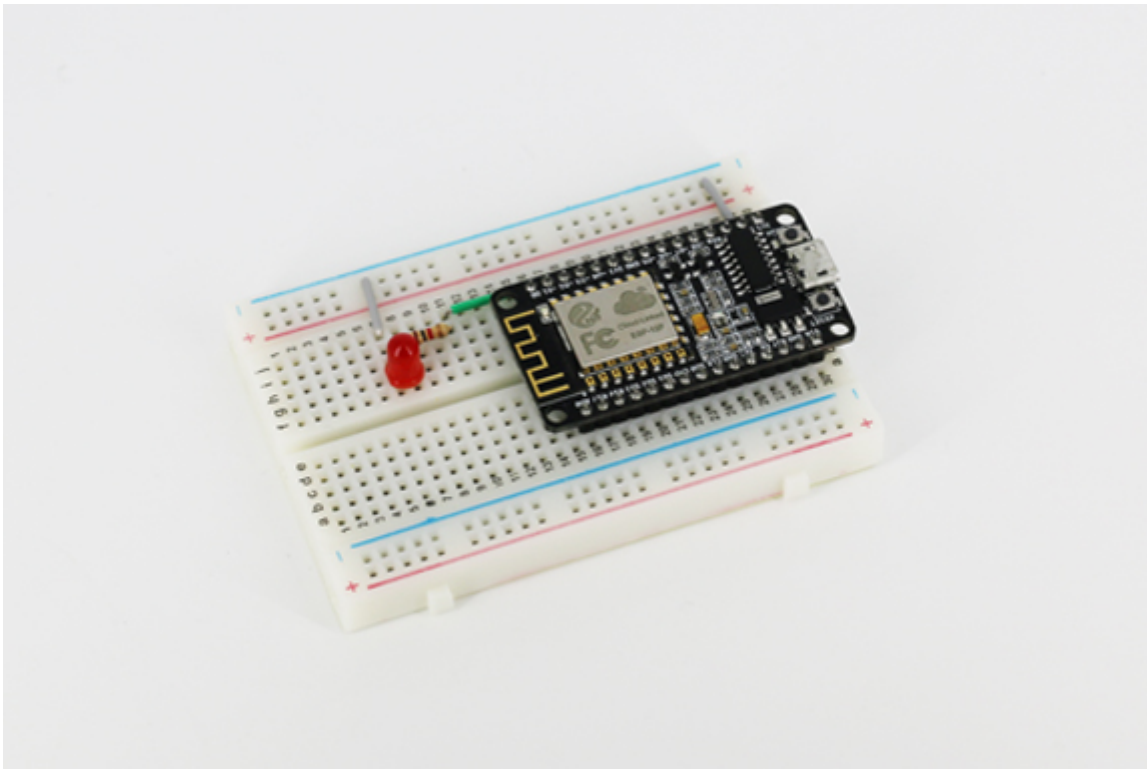
Início > ESP8266 > Programação do ESP8266 via OTA com wifi



Programação do ESP8266 via OTA com wifi

ESP8266, Wireless • 22 de junho de 2017 • André Curvello

O [módulo ESP8266](#) é outro divisor de águas que veio para ficar. Barato, prático, útil, esse microcontrolador com WiFi integrado pode ser programado pela sua SDK nativa, [usando LUA](#) e até mesmo com [Arduino IDE](#). E agora, uma funcionalidade extremamente útil será apresentada: a gravação via WiFi, também chamada de **OTA** (Over-The-Air) ou **FOTA** (Firmware Over-The-Air).



Gravação Over-The-Air (OTA)

Com o advento de dispositivos com comunicação sem-fio incorporada, seja Bluetooth, WiFi, GSM e até mesmo outros tipos, os fabricantes começaram a disponibilizar formas de programar o dispositivo por meio da comunicação sem-fio.

E qual o motivo de tanto “alarde” em prol dessa funcionalidade? É porque isso torna extremamente prático a atualização remota de equipamentos em campo. Pensa comigo: Você acabou de programar um sensor e colocou ele no forro da sua casa, monitorando o nível da caixa d’água, por exemplo. E de repente você lembrou que um parâmetro está errado e precisa ser corrigido... Sem atualização remota, você teria que subir no forro, desmontar o equipamento, ligar no seu computador, programar e remontar tudo.

Com atualização remota, se o equipamento está conectado (via WiFi, por exemplo), você é capaz de acessá-lo e reprogramá-lo! Dessa forma, não será necessário seu deslocamento nem o desmonte do equipamento para isso.

E olha que eu citei um exemplo bem simples, mas isso é usado por gente séria no mercado. Quer saber quem usa? A Tesla, por exemplo! Aquela grande fabricante de carros elétricos dos EUA. A atualização dos sistemas de seus veículos é feita remotamente por meio de GSM com conexões 3G/4G (depende do plano e do modelo do veículo).

Tendo por base um humilde NodeMCU com ESP8266, você pode remotamente atualizar o firmware do seu robô móvel do TCC, pode remotamente atualizar o firmware do sistema de domótica, controle de irrigação, monitoramento de caixas d’água, tangues, etc que está implementando, desde que o módulo esteja em uma mesma rede de acesso que seu computador.

Na FILIPEFLOP há uma outra plaquinha que é programada via WiFi por padrão, desde que a rede WiFi tenha conexão com a Internet – é a [Particle Photon](#).

Neste exemplo, vou tratar do NodeMCU ESP8266 com Arduino IDE, tendo por base um cenário de rede WiFi com acesso local, ou seja, você e o NodeMCU estão numa mesma rede WiFi.

Setup Necessário

É de extrema importância que você tenha o Arduino instalado. De preferência, instale a última versão, que no momento de escrita desse artigo é a versão 1.8.2, e está disponível no site [Arduino.cc](#).

Depois, faça a configuração do Arduino IDE para suportar o desenvolvimento com ESP8266 – Você pode fazer essa configuração tendo por base esse tutorial escrito no [Blog FILIPEFLOP](#).

Para fazer uso da funcionalidade de gravação via WiFi, é necessário ter o Python 2.7.x instalado na sua máquina. Você pode baixar o instalador no site <https://www.python.org/downloads> ou <https://www.python.org/downloads/release/python-2710> para já ir na sessão de downloads da sequência 2.7.10. Tendo por base máquinas Windows, baixe o arquivo **Windows x86-64 MSI Installer** para máquinas 64-bits, ou **Windows x86 MSI Installer** para máquinas 32-bits.

Instalação do Python 2.7.x

Após baixar o instalador do Python 2.7.x adequado, abra o executável baixado para efetivamente instalar o Python e toda sua infraestrutura no Windows. O processo de instalação é simples, é o famoso “próximo->próximo->próximo”, mas é preciso atentar para dois pontos importantes do processo de instalação:

1 – Na primeira janela de opções, marque a opção de instalar o programa para todos os usuários

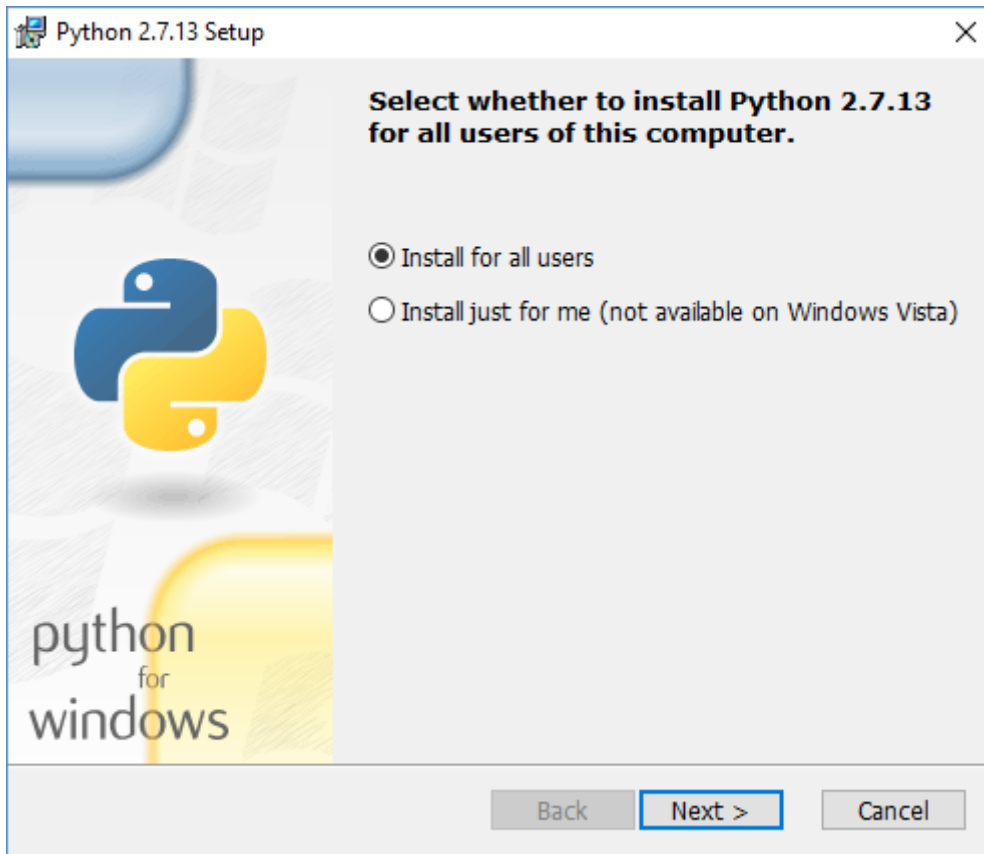


Figura 1 – Instalação do Python para todos os usuários.

2 – Logo depois, marque a opção para adicionar o **python.exe** ao PATH de sistema do Windows

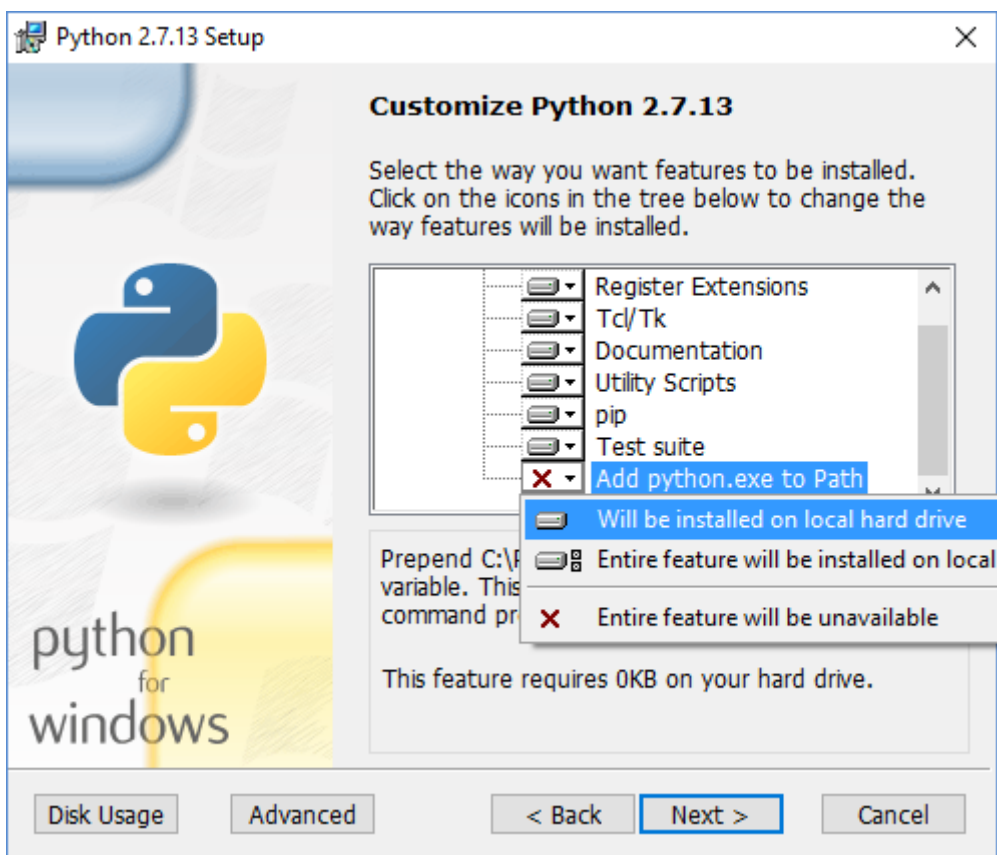


Figura 2 – Adicionar Python ao PATH do Windows.

Feito isso, tudo OK para prosseguirmos!

Não adianta pular essa parte. Sem o Python, o plugin de ESP8266 para o Arduino não conseguirá fazer as chamadas para gravação remota, ok?

Gravação via OTA – A Base

O âmago da coisa é SEMPRE (e eu repito: SEMPRE) gravar os sketches – nome bonito para “programa com Arduino” – com a estrutura do exemplo BasicOTA fornecido com o plugin de ESP8266 para o Arduino IDE.

As versões mais recentes do plugin ESP8266 passaram a vir com suporte para gravação remota via WiFi, sendo esse suporte (muito bem) exemplificado no sketch BasicOTA.

Se você já instalou o Arduino IDE, já configurou o suporte para ESP8266 e já instalou o Python na sua máquina, abra o programa Arduino IDE e selecione a placa NodeMCU 1.0 na opção “Boards”, para habilitar a listagem de exemplos para ESP8266.

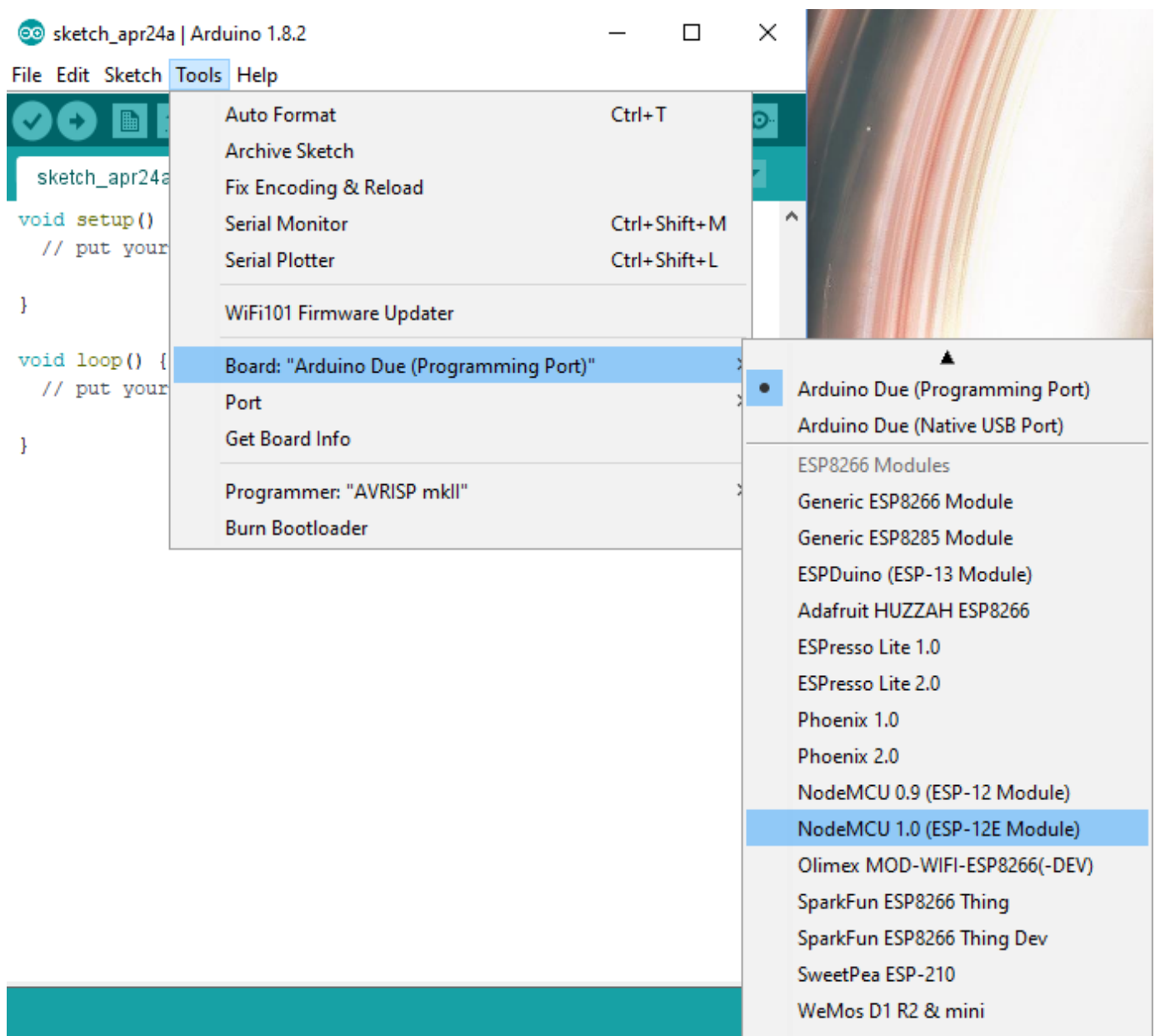
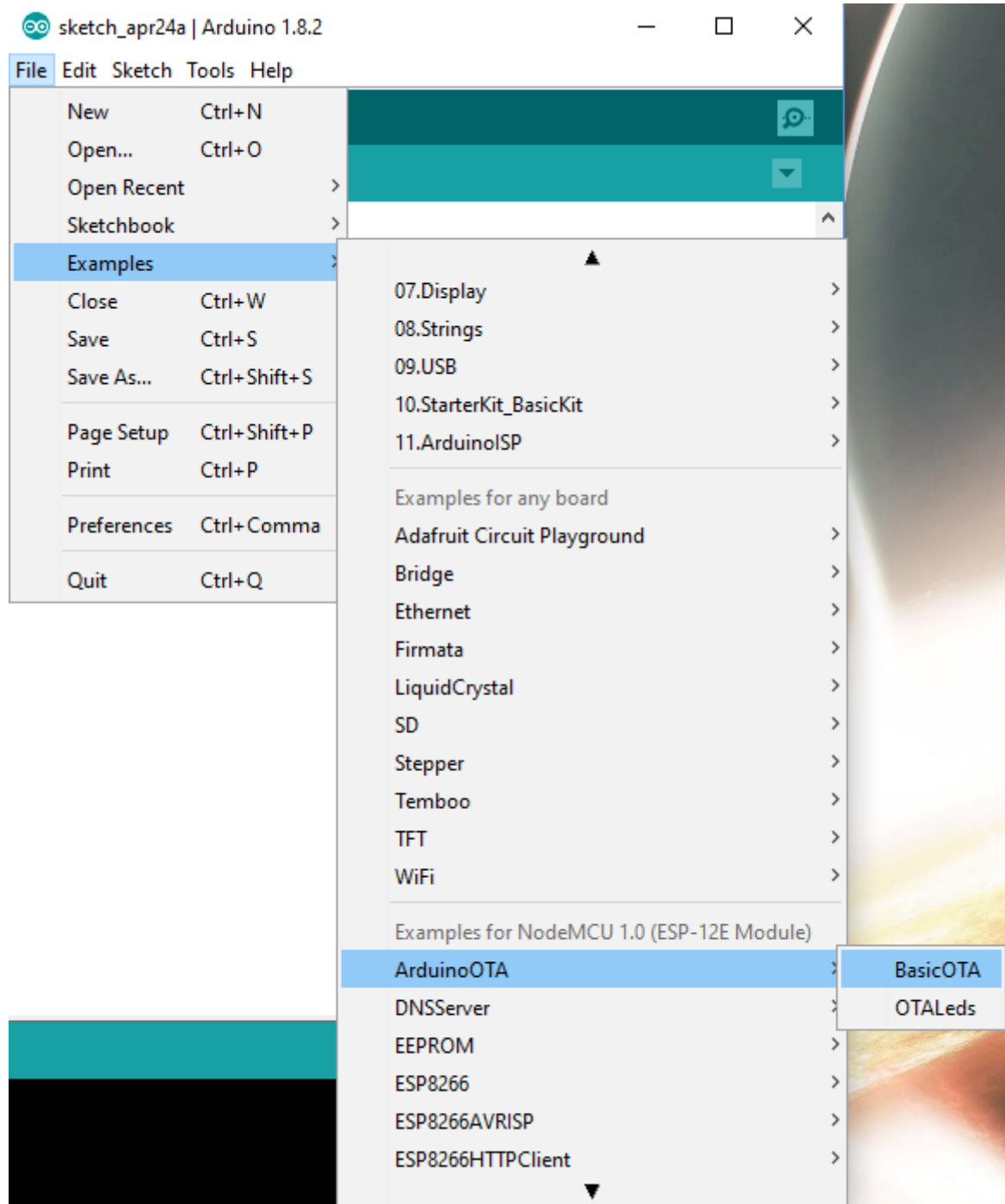


Figura 3 – Arduino IDE – Seleção da placa NodeMCU 1.0 – ESP-12E.

Configure adequadamente a opção Port de acordo com a porta serial COM atribuída ao seu módulo NodeMCU no Windows.

Assim, veja então na Figura 4 como abrir esse exemplo.

*Figura 4 – Arduino IDE – Seleção do Exemplo BasicOTA.*

Eu tomei a liberdade de usar esse Sketch como base, e fiz algumas modificações com comentários em português. Veja o código desse exemplo abaixo, com meus comentários:

```

1  #include <ESP8266WiFi.h>
2  #include <ESP8266mDNS.h>
3  #include <WiFiUdp.h>
4  #include <ArduinoOTA.h>
5
6  const char* ssid = "NOME DA SUA REDE WIFI";
7  const char* password = "SENHA DA SUA REDE WIFI";
8
9  void setup() {
10     Serial.begin(115200);
11     Serial.println("Iniciando...");
12     WiFi.mode(WIFI_STA);
13     WiFi.begin(ssid, password);
14     while (WiFi.waitForConnectResult() != WL_CONNECTED) {
15         Serial.println("Conexao falhou! Reiniciando...");
16         delay(5000);
17         ESP.restart();
18     }
19
20     // Porta padrao do ESP8266 para OTA eh 8266 - Voce pode mudar ser quiser, mas deixe indicado
21     // ArduinoOTA.setPort(8266);
22
23     // O Hostname padrao eh esp8266-[ChipID], mas voce pode mudar com essa funcao
24     // ArduinoOTA.setHostname("nome_do_meu_esp8266");
25
26     // Nenhuma senha eh pedida, mas voce pode dar mais seguranca pedindo uma senha pra gravar
27     // ArduinoOTA.setPassword((const char *)"123");
28
29     ArduinoOTA.onStart([]() {
30         Serial.println("Inicio...");
31     });
32     ArduinoOTA.onEnd([]() {
33         Serial.println("nFim!");
34     });
35     ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
36         Serial.printf("Progresso: %u%%r", (progress / (total / 100)));
37     });
38     ArduinoOTA.onError([](ota_error_t error) {
39         Serial.printf("Erro [%u]: ", error);
40         if (error == OTA_AUTH_ERROR) Serial.println("Autenticacao Falhou");
41         else if (error == OTA_BEGIN_ERROR) Serial.println("Falha no Inicio");
42         else if (error == OTA_CONNECT_ERROR) Serial.println("Falha na Conexao");
43         else if (error == OTA_RECEIVE_ERROR) Serial.println("Falha na Recepcao");
44         else if (error == OTA_END_ERROR) Serial.println("Falha no Fim");
45     });
46     ArduinoOTA.begin();
47     Serial.println("Pronto");
48     Serial.print("Endereco IP: ");
49     Serial.println(WiFi.localIP());
50 }
51
52 void loop() {
53     ArduinoOTA.handle();
54 }

```

A biblioteca **ESP8266WiFi.h** cuida das chamadas para acesso à rede WiFi, propriamente. Já uma boa parte das chamadas diz respeito à **ArduinoOTA.h**, que cuida das chamadas para tratar o recebimento de código e autogravação via WiFi.

No início do código há 2 variáveis que armazenam o nome e senha da sua rede WiFi, **ssid** e **password**, respectivamente.

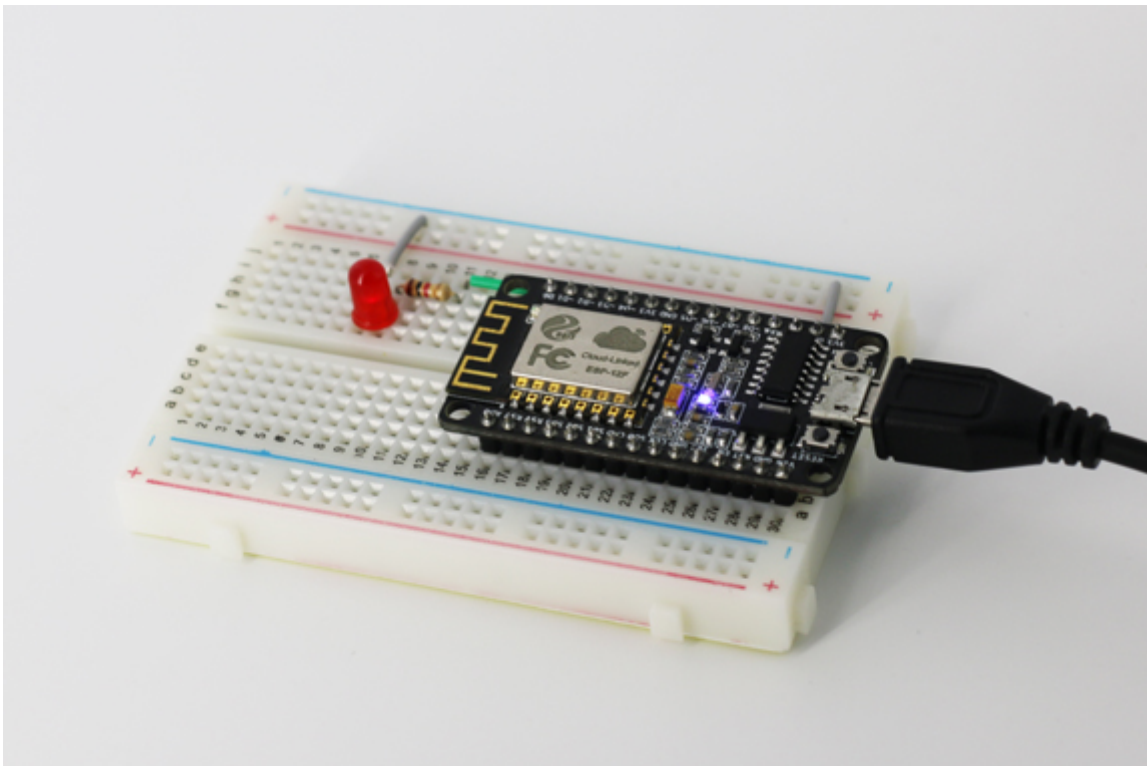
Na rotina void setup() é configurada a Serial (como forma de você acompanhar pela Serial se tudo está correndo bem ou não), o módulo é colocado no modo Station (ele se conecta à uma rede WiFi), e é começado todo um processo para configuração da gravação via OTA.

Como disse, tomei a liberdade de comentar algumas partes do código original em inglês para português.

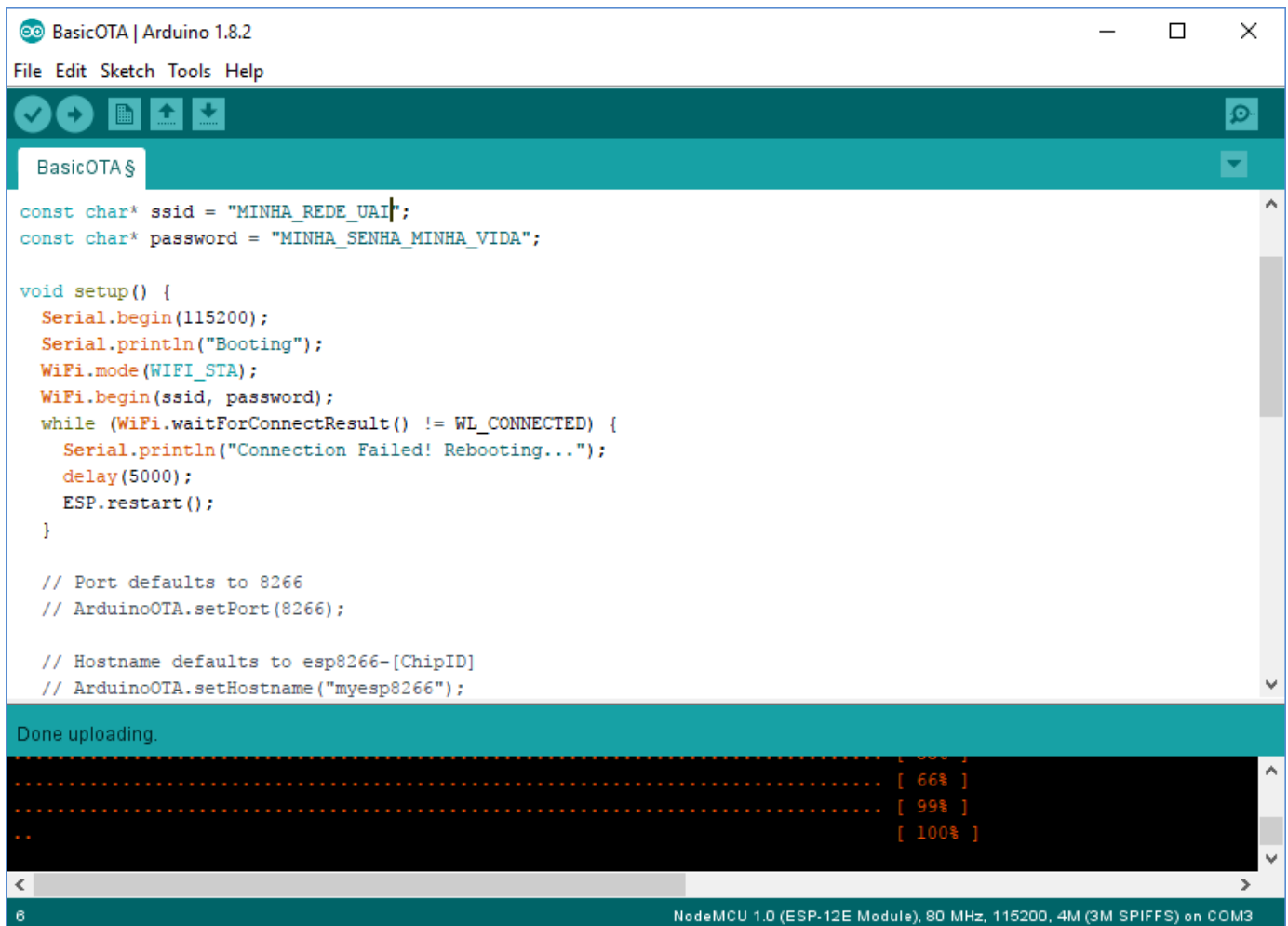
Essa estrutura-base é o que vai permitir seu ESP8266 ser conectado à rede WiFi e ficar disponível, quando requisitado, para receber e auto-gravar o programa recebido via WiFi com o Arduino IDE. Sem essa estrutura, o Arduino IDE não irá localizar o módulo ESP8266, e a programação sem-fio não funcionará.

Gravando o primeiro firmware para OTA

Como seu NodeMCU ESP8266 ainda não está preparado para gravação via WiFi, é preciso prepará-lo, certo? Dessa forma, conecte o NodeMCU via USB ao seu computador. Configure seu Arduino IDE para placa NodeMCU V1.0 e selecione a porta serial atribuída ao NodeMCU pelo Windows. No meu caso, como mostrado no canto inferior direito da Figura 5, minha NodeMCU ficou com a porta COM3 no Windows.



No código-exemplo de BasicOTA, coloque nos campos ssid e password as informações para acesso à sua rede WiFi, e coloque o Arduino IDE para compilar e gravar o código no NodeMCU, via USB (afinal de contas, ele ainda não tem a estrutura para gravação via WiFi!). Para efeitos de segurança e privacidade, como manda a boa educação, não deixei esses dados à mostra 😊



The screenshot shows the Arduino IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar contains icons for opening files, saving, and uploading. The main editor window displays the 'BasicOTA' sketch with the following code:

```
const char* ssid = "MINHA_REDE_WIFI";
const char* password = "MINHA_SENHA_MINHA_VIDA";

void setup() {
  Serial.begin(115200);
  Serial.println("Booting");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
    ESP.restart();
  }

  // Port defaults to 8266
  // ArduinoOTA.setPort(8266);

  // Hostname defaults to esp8266-[ChipID]
  // ArduinoOTA.setHostname("myesp8266");
}
```

Below the code editor, a status bar indicates 'Done uploading.' and a progress bar shows the upload progress. The serial monitor at the bottom displays the output of the sketch, showing the connection progress to the WiFi network:

```
..... [ 66% ]
..... [ 99% ]
.. [ 100% ]
```

The status bar at the bottom of the IDE shows 'NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on COM3'.

Figura 5 – Processo de gravação do BasicOTA no Arduino IDE.

Terminado o processo de gravação, tenha uma janela do Terminal Serial do Arduino aberta – é só clicar no ícone da Lupa no Arduino IDE, no canto superior direito – para ver o resultado (positivo ou não...) do processo de conexão do módulo à rede WiFi. Caso não apareça nada, aperte o botão **Reset** do NodeMCU para reiniciar o módulo e forçar a aparição das mensagens. Veja no meu caso como apareceu:

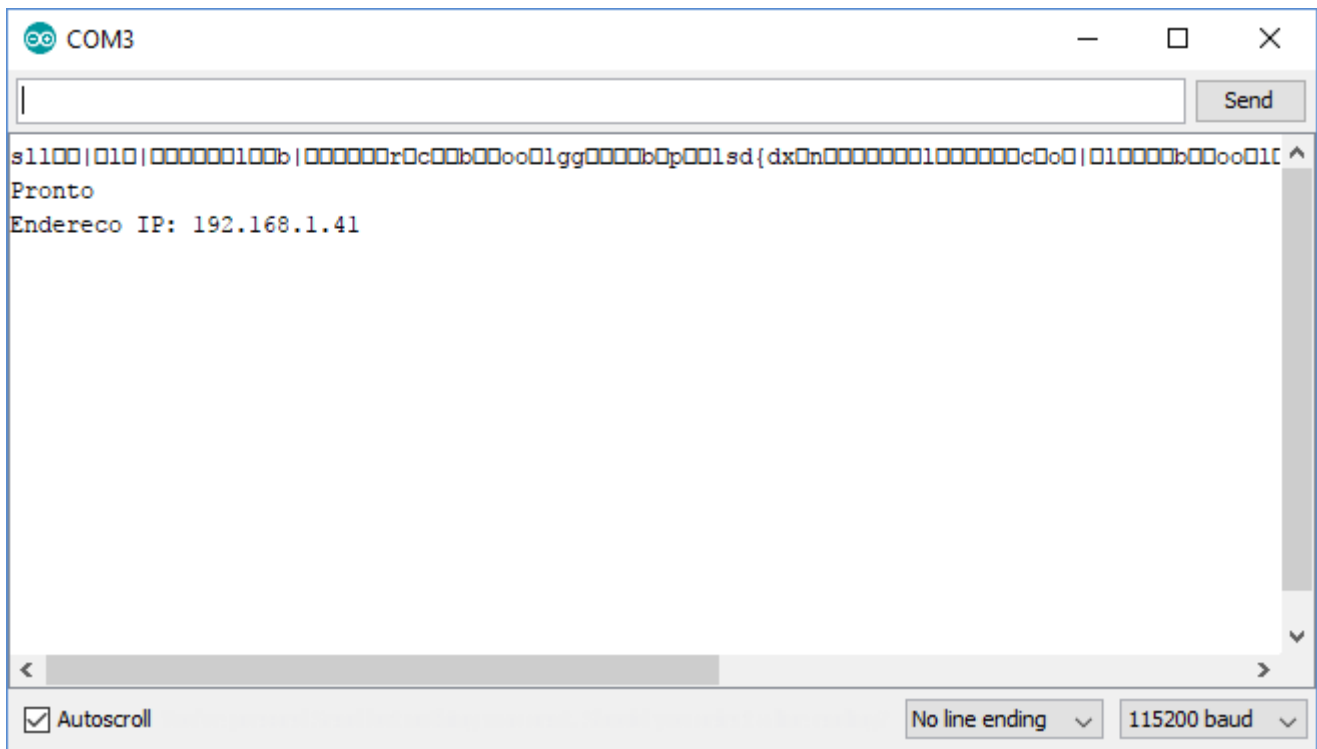


Figura 6 – Endereço IP do módulo na Rede.

Se correu tudo bem, o Arduino IDE agora é capaz de ver o endereço IP do NodeMCU ESP8266 pela rede. **Não sei se isso é geral**, mas **comigo** é preciso **fechar** o **Arduino IDE** e **abrir o programa novamente**, para que o endereço IP da placa apareça na listagem de Portas. Bom... Feito isso, veja como agora, na opção “Tools->Port” (Ou Ferramentas->Porta), e agora veja que além das portas Seriais, aparece também o módulo e seu endereço IP, na parte Network ports!

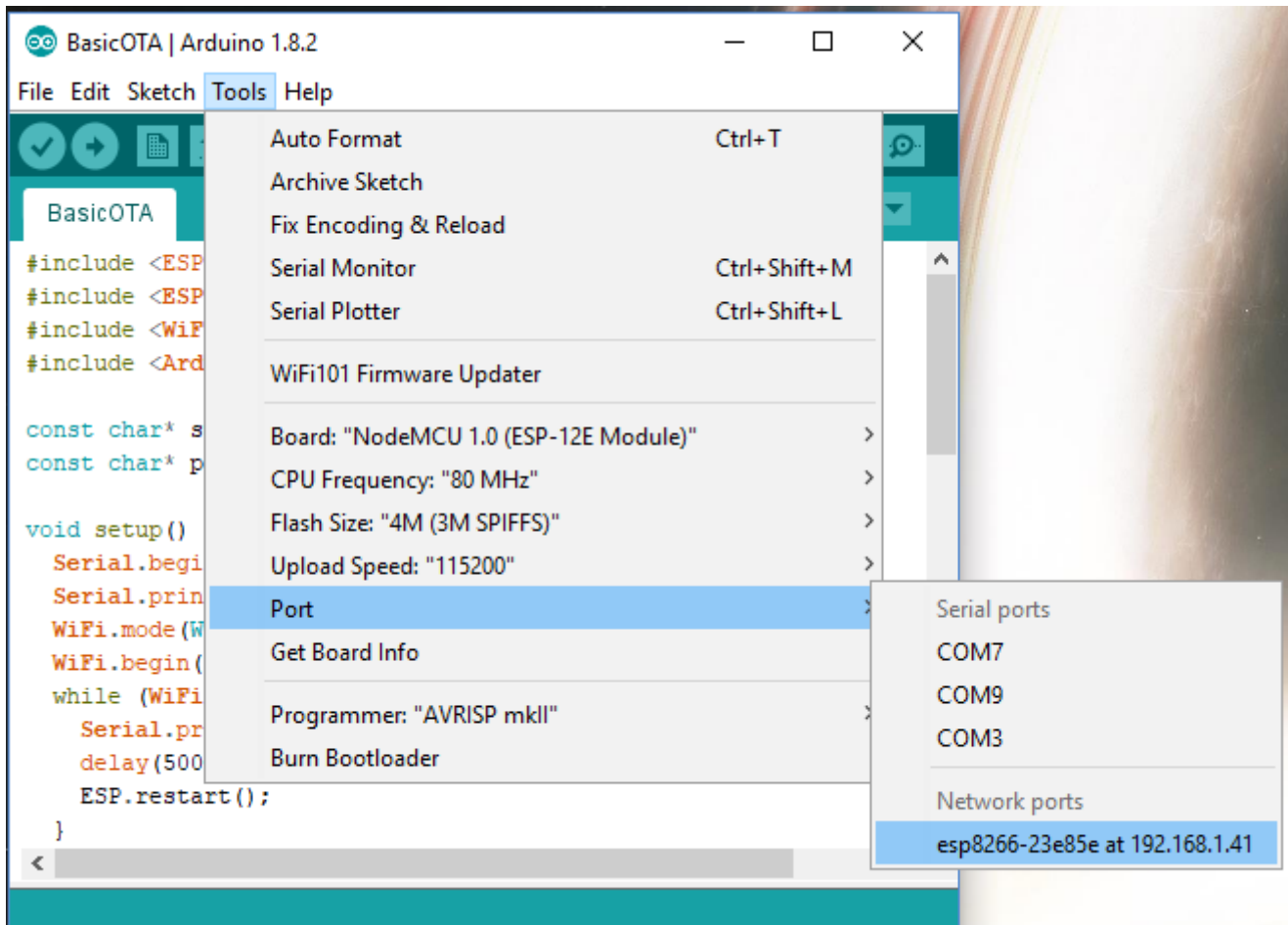


Figura 7 – NodeMCU agora consta como elemento de rede para gravação!

Porém... Esse código por si só não faz nada. Só deixa o módulo esperando algum comando de gravação remota.

Gravação via OTA – Exemplo

Sem tirar a estrutura-base do exemplo que começamos, principalmente no que tange às configurações que estão em **setup()**, mantenha a chamada **ArduinoOTA.handle()** no início da rotina **loop()**.

Essa chamada **ArduinoOTA.handle()** é quem lida com as requisições para gravação de firmware remotamente. Ou seja, seu novo programa além de funcionar com a sua lógica específica (controle de relé, monitoramento de sensor, exibição de dados em display, etc!), ficará “atento” à requisições de gravações remotas.

Além da estrutura lógica do OTA, também é preciso manter as configurações de Rede WiFi (ssid e password) para que o módulo seja conectado na rede WiFi, ok?

Vejam abaixo como fica um programa com o esqueleto do **BasicOTA**, mas agora com uma lógica para controle de um LED conectado na GPIO 2, por exemplo:

```
1 #include <ESP8266WiFi.h>
2 #include <ESP8266mDNS.h>
3 #include <WiFiUdp.h>
4 #include <ArduinoOTA.h>
5
6 const char* ssid = "Minha_rede_WiFi";
7 const char* password = "Senha_da_minha_rede_wifi";
8
9 void setup() {
10     //Colocamos o sinal D4 (GPIO02) do NodeMCU como saída
11     pinMode(D4, OUTPUT);
12     Serial.begin(115200);
13     Serial.println("Iniciando...");
14     WiFi.mode(WIFI_STA);
15     WiFi.begin(ssid, password);
16     while (WiFi.waitForConnectResult() != WL_CONNECTED) {
17         Serial.println("Conexao falhou! Reiniciando...");
18         delay(5000);
19         ESP.restart();
20     }
21
22     ArduinoOTA.onStart([]() {
23         Serial.println("Inicio...");
24     });
25     ArduinoOTA.onEnd([]() {
26         Serial.println("Fim!");
27     });
28     ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
29         Serial.printf("Progresso: %u%%r", (progress / (total / 100)));
30     });
31     ArduinoOTA.onError([](ota_error_t error) {
32         Serial.printf("Erro [%u]: ", error);
33         if (error == OTA_AUTH_ERROR) Serial.println("Autenticacao Falhou");
34         else if (error == OTA_BEGIN_ERROR) Serial.println("Falha no Inicio");
35         else if (error == OTA_CONNECT_ERROR) Serial.println("Falha na Conexao");
36         else if (error == OTA_RECEIVE_ERROR) Serial.println("Falha na Recepcão");
37         else if (error == OTA_END_ERROR) Serial.println("Falha no Fim");
38     });
39     ArduinoOTA.begin();
40     Serial.println("Pronto");
41     Serial.print("Endereco IP: ");
42     Serial.println(WiFi.localIP());
43 }
44
45 void loop() {
46     // Mantenha esse trecho no inicio do laço "loop" - verifica requisicoes OTA
47     ArduinoOTA.handle();
48     digitalWrite(D4, HIGH); // Aciona sinal 2
49     delay(2000); // Espera por 2 segundos
50     digitalWrite(D4, LOW); // Apaga sinal 2
51     delay(2000); // Espera por 2 segundos
52 }
```

Apenas para dar uma noção da montagem, conectei um LED vermelho no sinal D4 (GPIO02) do NodeMCU, conforme mostrado no seguinte esquemático de Fritzing:

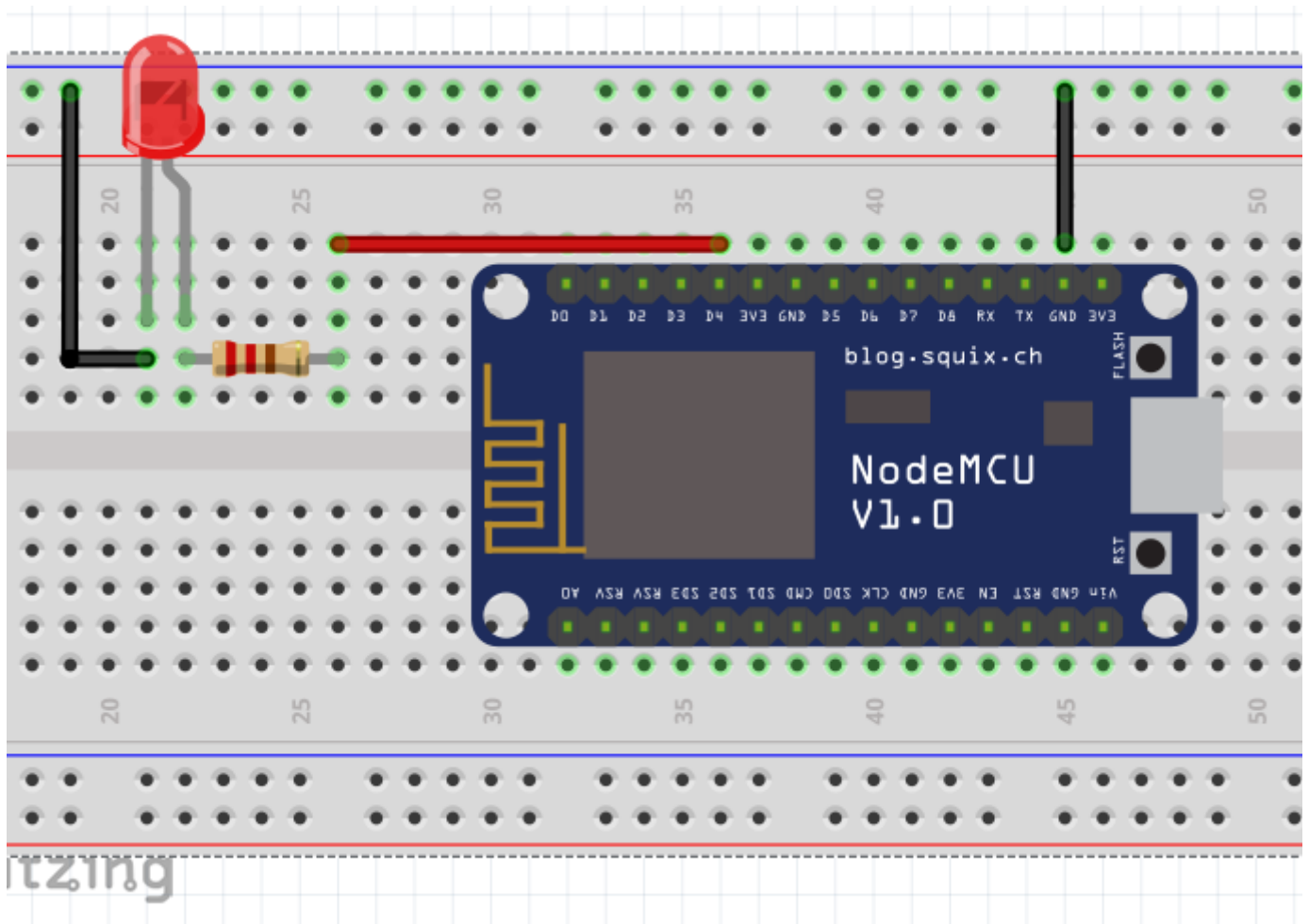


Figura 8 – Esquemático de Ligação com NodeMCU.

Para testar esse programa em execução no Arduino IDE, coloque o código com a estrutura mostrada acima (lembrando de substituir a parte de ssid/senha com os dados para acesso à sua rede WiFi), e **selecione agora o módulo pelo seu endereço IP** em Ferramentas->Portas (Tools->Ports) do Arduino IDE. Mande o Arduino compilar e gravar o código.

Lembra que tivemos que instalar o Python para realizar a gravação remota? Observe que após o Arduino IDE **compilar** o código e começar o processo de gravação, irá aparecer uma janela do Windows pedindo autorização para o Python acessar recursos da Rede. Autorize marcando as duas caixas de opções, para que tudo corra bem.

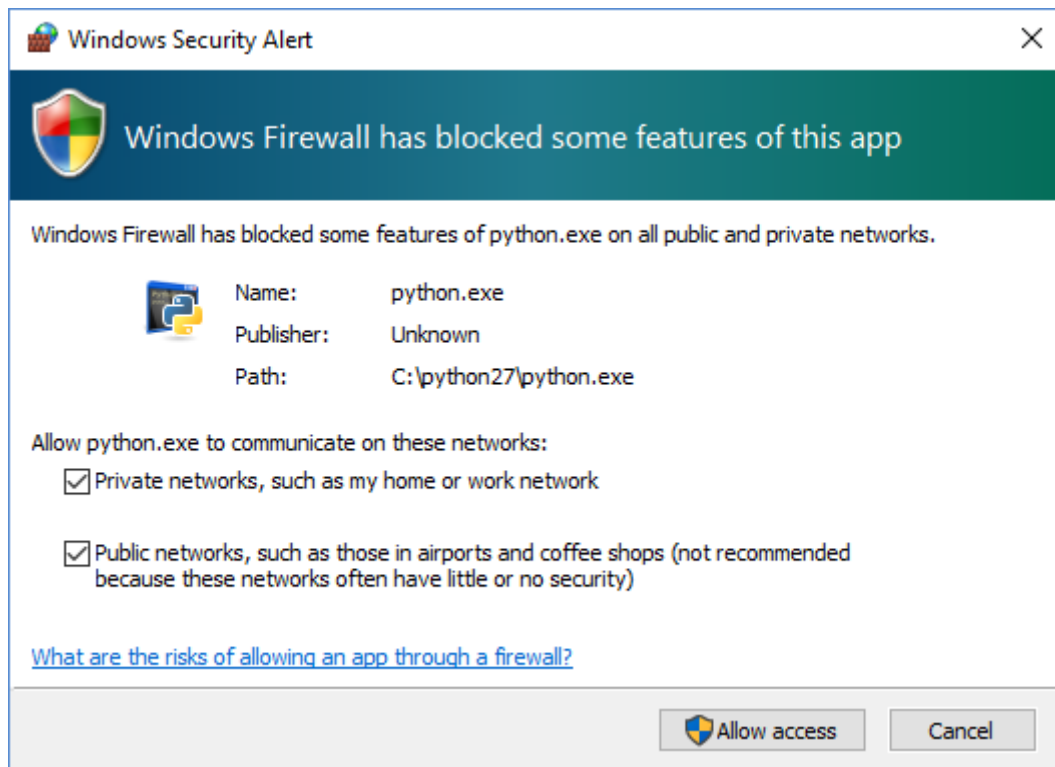
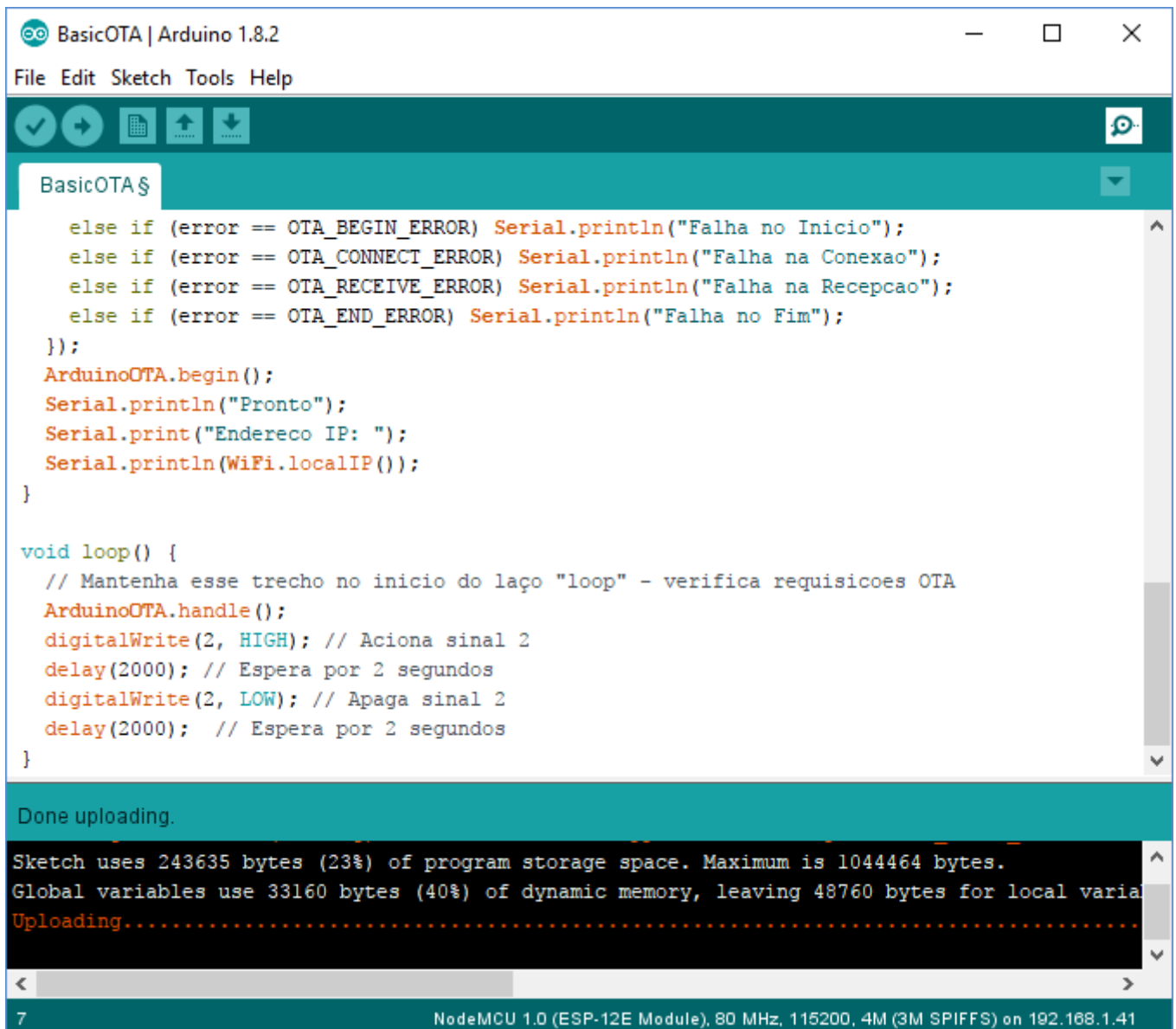


Figura 9 – Autorização do Python para acesso à rede em Windows.

Por fim, acompanhe o resultado. Veja que agora a gravação ocorrerá via WiFi – Observe a minha seleção no canto inferior direito do Arduino IDE, veja que o “Port” é o endereço IP do módulo NodeMCU!



```
BasicOTA | Arduino 1.8.2
File Edit Sketch Tools Help

BasicOTA$

else if (error == OTA_BEGIN_ERROR) Serial.println("Falha no Inicio");
else if (error == OTA_CONNECT_ERROR) Serial.println("Falha na Conexao");
else if (error == OTA_RECEIVE_ERROR) Serial.println("Falha na Recepcao");
else if (error == OTA_END_ERROR) Serial.println("Falha no Fim");
});
ArduinoOTA.begin();
Serial.println("Pronto");
Serial.print("Endereco IP: ");
Serial.println(WiFi.localIP());
}

void loop() {
  // Mantenha esse trecho no inicio do laço "loop" - verifica requisicoes OTA
  ArduinoOTA.handle();
  digitalWrite(2, HIGH); // Aciona sinal 2
  delay(2000); // Espera por 2 segundos
  digitalWrite(2, LOW); // Apaga sinal 2
  delay(2000); // Espera por 2 segundos
}

Done uploading.

Sketch uses 243635 bytes (23%) of program storage space. Maximum is 1044464 bytes.
Global variables use 33160 bytes (40%) of dynamic memory, leaving 48760 bytes for local variables.
Uploading.....

7 NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on 192.168.1.41
```

Figura 10 – NodeMCU programado com Arduino IDE via WiFi!

Pelo fato de você ter gravado seu programa com a estrutura do OTA, será possível, assim, novamente gravar o ESP8266 via WiFi usando o Arduino, e assim sucessivamente, mantendo a estrutura!

Gostou? Deixe seu comentário logo abaixo. Em caso de dúvidas, caso queira trocar uma ideia, ou até mesmo dividir seu projeto, acesse nosso [Fórum!](#)

Posts Relacionados

Como gravar dados
no cartão RFID

Linkit Smart 7688
Duo - Uma placa 2
em 1

Linkit Smart 7688
com MQTT e
MediaTek Cloud

Bluetooth Low
Energy com ESP32 e
DUT11



André Curvello

Graduado em Engenharia de Computação com ênfase em Sistemas Embarcados pela USP, campus São Carlos, possui MBA em gestão de TI pela UNIFRAN e Mestrado em Ciências pela EESC-USP. Atua como Analista de Tecnologia para Sistemas Embarcados na Padtec S/A, sendo também professor de pós-graduação e instrutor de cursos na área de sistemas embarcados. Como hobby, gosta de programar tudo que pode ser programado, escovando bits sempre ao lado de um bom café. Gosta de compartilhar seu conhecimento por meio de palestras, e publicando artigos como colaborador dos sites FILIPEFLOP e Embarcados.

← Intel Galileo Gen2 com nova imagem em microSD

Projeto com Arduino cria ilusão hipnotizante →

Deixe uma resposta

Conectar com:



O seu endereço de e-mail não será publicado. Campos obrigatórios são marcados com *

Comentário

Nome ***E-mail *****Site****Publicar comentário**

17 Comentários



SHOW, vou testar ainda hoje. PARABÉNS André.

Adriano Matos 22 de junho de 2017

[Responder](#)

Muito obrigado!

André Curvello 23 de junho de 2017

[Responder](#)



Bom dia excelente trabalho. Como faco para ligar por exemplo um led atravez de uma rede externa?

Isael 22 de junho de 2017

[Responder](#)



Olha, através de uma rede externa aí é bom você ver alguma abordagem como uso de MQTT!

André Curvello 23 de junho de 2017

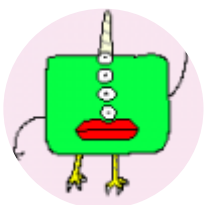
[Responder](#)



Bom dia excelente trabalho. Como faco para ligar por exemplo um led atravez de uma rede externa?
Acesso remoto

Isael 22 de junho de 2017

[Responder](#)



Olá André,

Você fez os testes usando windows 10?

Obrigado.

Abs,

Carlos

Carlos 22 de junho de 2017

[Responder](#)



É possível fazer isso com o arduino wifi D1R2 ??

Tiago Borges 9 de julho de 2017

[Responder](#)



Ótimo artigo! porem estou tentando carregar meu esp 8266 12e com ftdi 232RL mas na hora de compilar e carregar da falha com a mensagem:

"error: espcom_upload_mem failed"

Alguém pode me socorrer por favor!!

gleison 19 de julho de 2017

[Responder](#)



Ótimo artigo, parabéns André. Só queria confirmar a seguinte dúvida que apareceu aí no meio:

"(...) Não sei se isso é geral, mas comigo é preciso fechar o Arduino IDE e abrir o programa novamente"

Tive que fazer a mesma coisa aqui, e estou utilizando o Arduino IDE no macOS Sierra.

Essa frase sua me fez poupar vários muitos de dor de cabeça... 😊

Maurício de O. Pena 3 de setembro de 2017

[Responder](#)



Eu consigo realizar esta gravação utilizando apenas o ESP8266 sem nodeMCU?
Teria alguma grande alteração no tutorial?

Johnny 25 de setembro de 2017

[Responder](#)



ótimo trabalho..parabéns... eu consigo ter as informações de varias plantas em uma só conta? um só twitter?

Rubens de Souza 27 de setembro de 2017

[Responder](#)



É possível colocar ip fixo? É possível fazer leitura? ex. `char rcv = client.read();`
Obrigado!

Daniel 5 de dezembro de 2017

[Responder](#)



Boa noite,

Ótimo artigo André! porém o Arduino IDE não encontrou meu endereço IP na lista de portas. Mesmo reiniciando o programa 2x.

O programa compilou e enviou normal, assim como mostrou que esta conectado em um IP específico.

Você tem mais alguma ideia do que possa ser? utilizo o Windows 10.

Fabio Fernandes 12 de dezembro de 2017

[Responder](#)



Muito obrigado, deu certo. Grande abraço.

Fabio Cardoso 5 de março de 2018

[Responder](#)



Olá, parabéns pelo artigo, montei tudo e funcionou perfeitamente com a placa de desenvolvimento, mas quando uso apenas o ESP-01 ou ESP-07 sem placa de desenvolvimento, não consigo efetuar o update via OTA neles, consigo pela serial perfeitamente, mas via wifi da erro. Poderia me ajudar, obrigado!

xandcapcom 14 de março de 2018

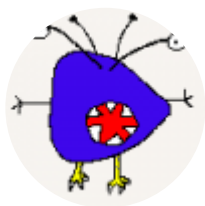
[Responder](#)



boa noite tenho esp8266 e 12 fiz CONFORME ESCRITO E APARECEU ISSO "Connection Failed! Rebooting..."

marcos antonio oliveira silva 17 de abril de 2018

[Responder](#)



Ola, boa noite.

Fiz todo o passo a passo, porem o IP nao aparece. Apenas a frase : Conexao falhou! Reiniciando...

Poderia me informar o motivo?

Drielly Cavalcanti 3 de maio de 2018

[Responder](#)

Categorias

- > Arduino (92)
- > Display (7)
- > Embarcados (46)
- > ESP8266 (17)
- > Módulos (15)
- > Motores e Servos (18)
- > News (44)
- > Outros (11)
- > Raspberry Pi (61)

> Sensores (23)

> Shield (2)

> Wearable (4)

> Wireless (51)



Assine a Newsletter

...e junte-se à **maior comunidade maker do Brasil**

Digite seu E-mail

Enviar

Dúvidas Frequentes

Como comprar

Frete Grátis

Retirada em mãos

Pagamento e envio

Trocas e devoluções

Institucional

Quem Somos

Termos de Uso

Trabalhe Conosco



Alguma dúvida? Entre em contato!

(48) 3307-3495

Atendimento Online

Seg-Sex das 08h30-11h30 às 13h-17h



FilipeFlop Componentes Eletrônicos | CNPJ: 12.672.380/0001-90

Rod. José Carlos Daux 4850 Galpão 19, Florianópolis/SC, 88032-005

