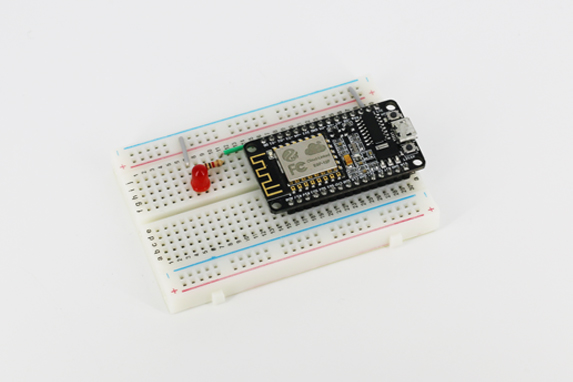


Programação do ESP8266 via OTA com wifi[20](https://www.filipeflop.com/blog/programacao-esp8266-ota-wifi/#comments)

[ESP8266](https://www.filipeflop.com/blog/category/esp8266/), [Wireless](https://www.filipeflop.com/blog/category/wireless/)  [22 de junho de 2017](https://www.filipeflop.com/blog/programacao-esp8266-ota-wifi/) [André Curvello](https://www.filipeflop.com/blog/author/andrecurvello/)

O [módulo ESP8266](https://www.filipeflop.com/categoria/wireless-iot/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](https://www.filipeflop.com/wireless/esp8266-nodemcu-como-programar.html) e até mesmo com [Arduino IDE](https://www.filipeflop.com/wireless/programar-nodemcu-com-ide-arduino.html). 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](https://www.filipeflop.com/blog/projetos-iot-com-a-particle-photon/" \t "_blank).

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](https://www.arduino.cc/en/main/software).

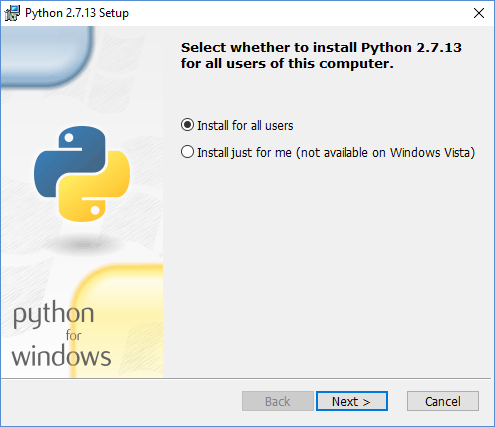
Depois, faça a configuração do Arduino IDE para suportar o desenvolvimento com ESP8266  – Você pode fazer essa configuração tendo por base esse tutorial escrito no [Blog FILIPEFLOP](https://www.filipeflop.com/wireless/programar-nodemcu-com-ide-arduino.html).

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](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.

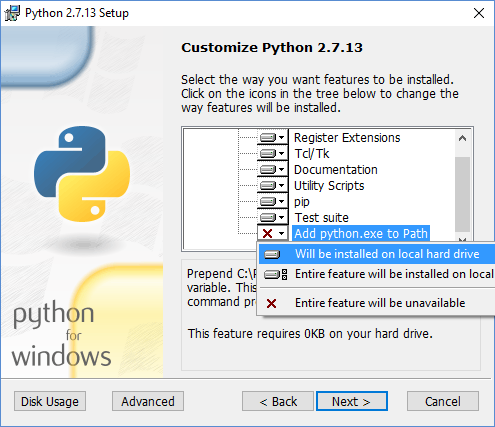
**Instalação do Python 2.7.x**

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!

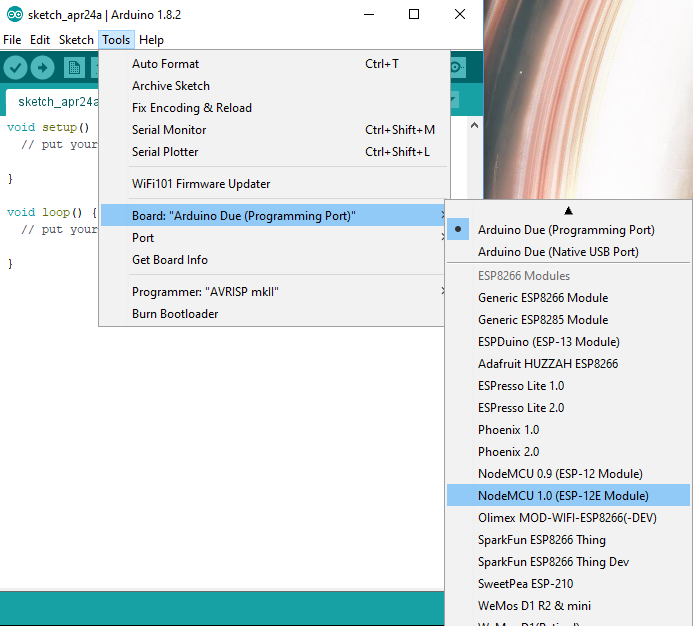
Não adianta pular essa parte. Sem o Python, o plugin de ESP8266 para o Arduino não conseguirá fazer as chamadas para gravação remota, ok?

**Gravação via OTA – A Base**

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

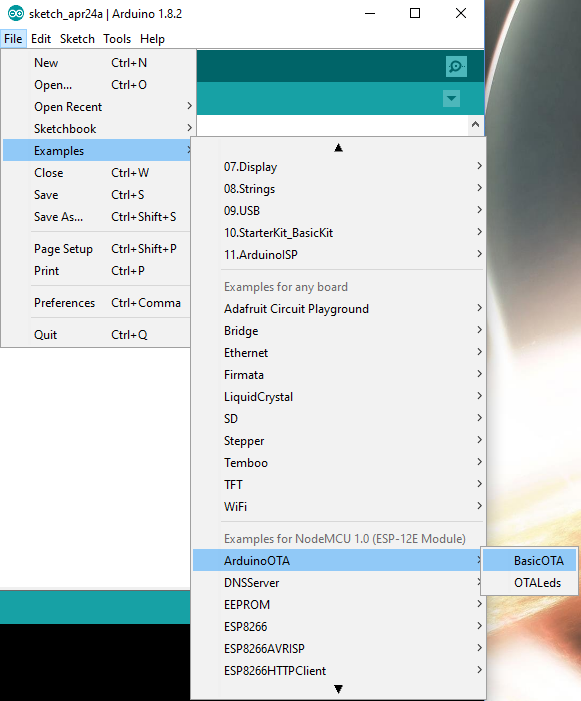
As versões mais recentes do plugin ESP8266 passaram a vir com suporte para gravação remota via WiFi, sendo esse suporte (muito bem) exemplificado no sketch BasicOTA.

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:

[?](https://www.filipeflop.com/blog/programacao-esp8266-ota-wifi/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | #include <ESP8266WiFi.h>  #include <ESP8266mDNS.h>  #include <WiFiUdp.h>  #include <ArduinoOTA.h>    const char\* ssid = "NOME DA SUA REDE WIFI";  const char\* password = "SENHA DA SUA REDE WIFI";    void setup() {    Serial.begin(115200);    Serial.println("Iniciando...");    WiFi.mode(WIFI\_STA);    WiFi.begin(ssid, password);    while (WiFi.waitForConnectResult() != WL\_CONNECTED) {      Serial.println("Conexao falhou! Reiniciando...");      delay(5000);      ESP.restart();    }      // Porta padrao do ESP8266 para OTA eh 8266 - Voce pode mudar ser quiser, mas deixe indicado!    // ArduinoOTA.setPort(8266);      // O Hostname padrao eh esp8266-[ChipID], mas voce pode mudar com essa funcao    // ArduinoOTA.setHostname("nome\_do\_meu\_esp8266");      // Nenhuma senha eh pedida, mas voce pode dar mais seguranca pedindo uma senha pra gravar    // ArduinoOTA.setPassword((const char \*)"123");      ArduinoOTA.onStart([]() {      Serial.println("Inicio...");    });    ArduinoOTA.onEnd([]() {      Serial.println("nFim!");    });    ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {      Serial.printf("Progresso: %u%%r", (progress / (total / 100)));    });    ArduinoOTA.onError([](ota\_error\_t error) {      Serial.printf("Erro [%u]: ", error);      if (error == OTA\_AUTH\_ERROR) Serial.println("Autenticacao Falhou");      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() {    ArduinoOTA.handle();  } |

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.

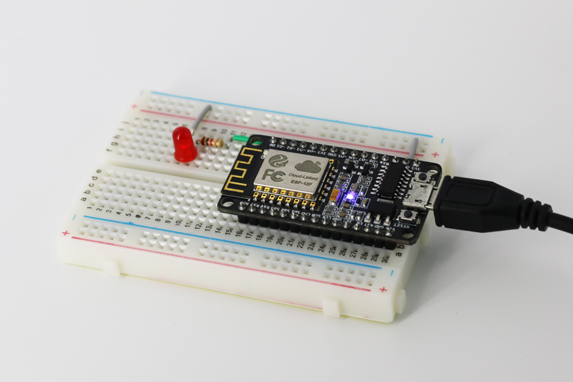
Na rotina void setup() é configurada a Serial (como forma de você acompanhar pela Serial se tudo está correndo bem ou não), o módulo é colocado no modo Station (ele se conecta à uma rede WiFi), e é começado todo um processo para configuração da gravação via OTA.

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

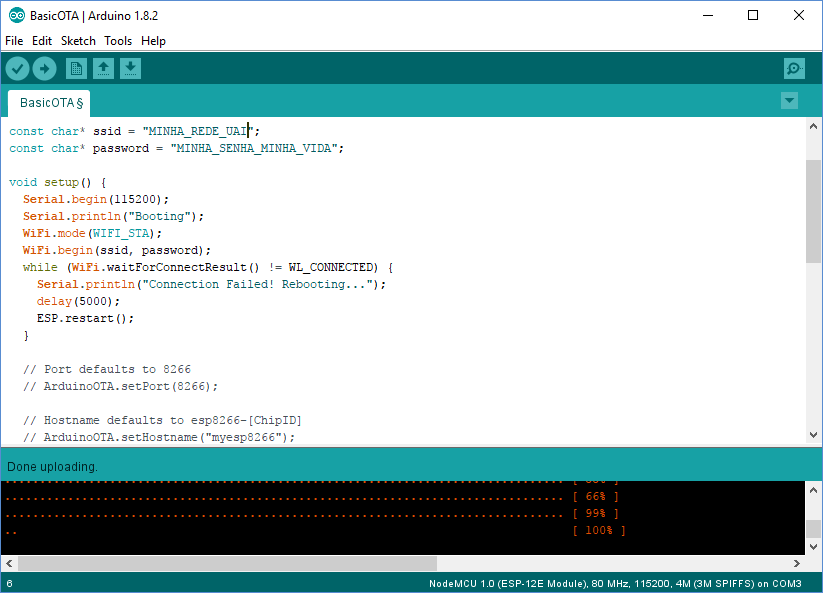
Essa estrutura-base é o que vai permitir seu ESP8266 ser conectado à rede WiFi e ficar disponível, quando requisitado, para receber e auto-gravar o programa recebido via WiFi com o Arduino IDE. Sem essa estrutura, o Arduino IDE não irá localizar o módulo ESP8266, e a programação sem-fio não funcionará.

**Gravando o primeiro firmware para OTA**

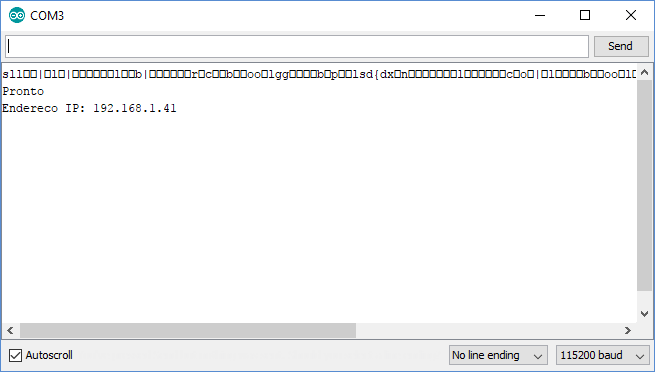
Como seu NodeMCU ESP8266 ainda não está preparado para gravação via WiFi, é preciso prepará-lo, certo? Dessa forma, conecte o NodeMCU via USB ao seu computador. Configure seu Arduino IDE para placa NodeMCU V1.0 e selecione a porta serial atribuída ao NodeMCU pelo Windows. No meu caso, como mostrado no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



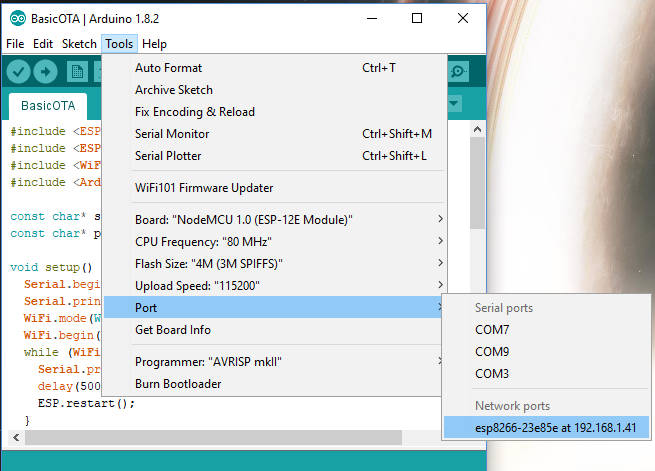
No código-exemplo de BasicOTA, coloque nos campos ssid e password as informações para acesso à sua rede WiFi, e coloque o Arduino IDE para compilar e gravar o código no NodeMCU, via USB (afinal de contas, ele ainda não tem a estrutura para gravação via WiFi!). Para efeitos de segurança e privacidade,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

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:

[](https://uploads.filipeflop.com/2017/06/arduino_console_ip1.png)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!

[](https://uploads.filipeflop.com/2017/06/arduino_nodemcu_ip.png)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.

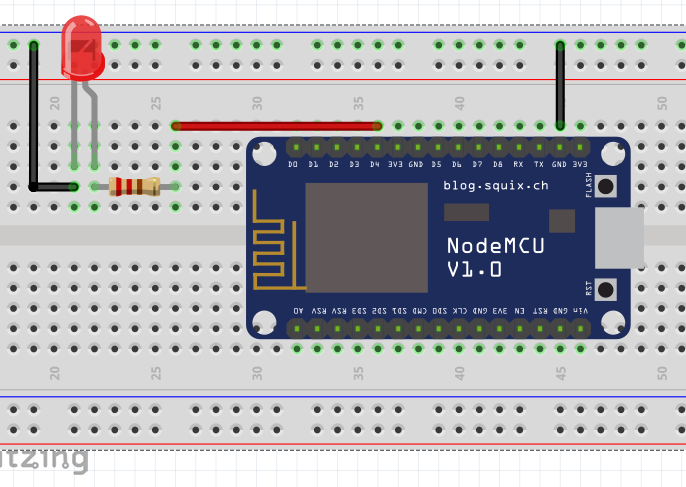
Além da estrutura lógica do OTA, também é preciso manter as configurações de Rede WiFi (ssid e password) para que o módulo seja conectado na rede WiFi, ok?

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:

[?](https://www.filipeflop.com/blog/programacao-esp8266-ota-wifi/)

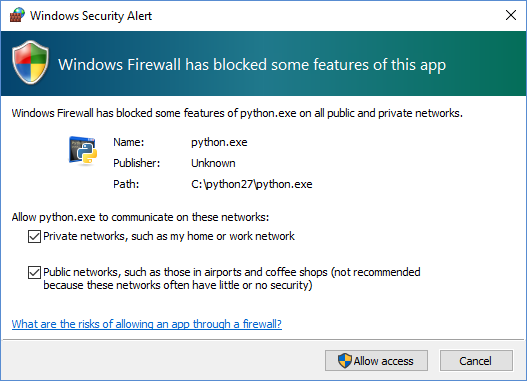
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | #include <ESP8266WiFi.h>  #include <ESP8266mDNS.h>  #include <WiFiUdp.h>  #include <ArduinoOTA.h>    const char\* ssid = "Minha\_rede\_WiFi";  const char\* password = "Senha\_da\_minha\_rede\_wifi";    void setup() {    //Colocamos o sinal D4 (GPIO02) do NodeMCU como saida    pinMode(D4, OUTPUT);    Serial.begin(115200);    Serial.println("Iniciando...");    WiFi.mode(WIFI\_STA);    WiFi.begin(ssid, password);    while (WiFi.waitForConnectResult() != WL\_CONNECTED) {      Serial.println("Conexao falhou! Reiniciando...");      delay(5000);      ESP.restart();    }      ArduinoOTA.onStart([]() {      Serial.println("Inicio...");    });    ArduinoOTA.onEnd([]() {      Serial.println("nFim!");    });    ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {      Serial.printf("Progresso: %u%%r", (progress / (total / 100)));    });    ArduinoOTA.onError([](ota\_error\_t error) {      Serial.printf("Erro [%u]: ", error);      if (error == OTA\_AUTH\_ERROR) Serial.println("Autenticacao Falhou");      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(D4, HIGH); // Aciona sinal 2    delay(2000); // Espera por 2 segundos    digitalWrite(D4, LOW); // Apaga sinal 2    delay(2000);  // Espera por 2 segundos  } |

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

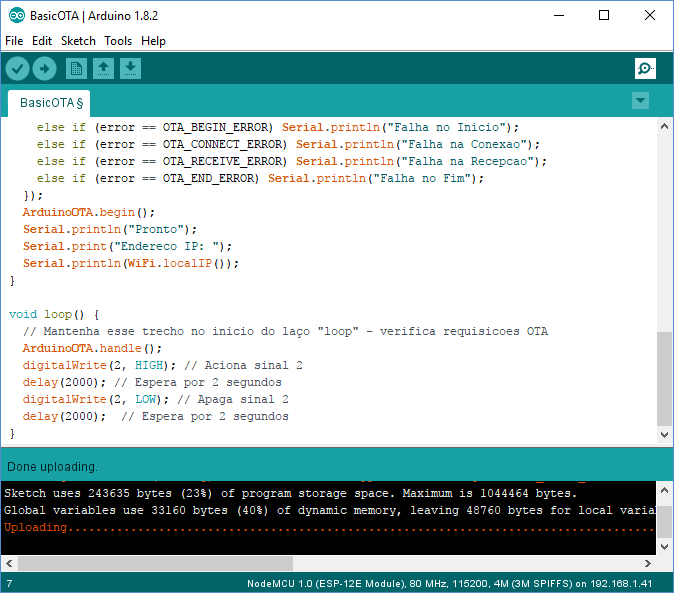
Figura 8 – Esquemático de Ligação com NodeMCU.

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.

[](https://uploads.filipeflop.com/2017/06/python_authorize.png)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!

[](https://uploads.filipeflop.com/2017/06/arduino_fota_ok.png)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!