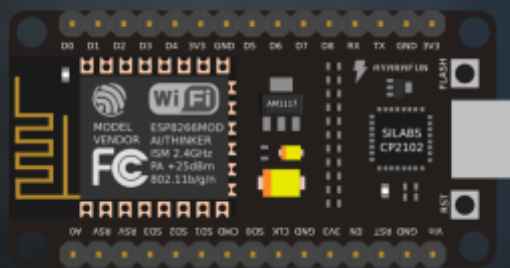
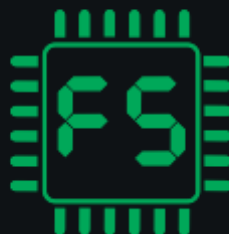


# Controle Dispositivos Remotamente com o ESP8266



Fábio Souza



# Controle dispositivos remotamente com o ESP8266

Fábio Souza

2018

## Aviso legal

Este eBook foi escrito com fins didáticos, e com todos os esforço para que ele ficasse o mais claro e didático possível.

O objetivo deste eBook é educar. O autor não garante que as informações contidas neste eBook estão totalmente completas e não deve ser responsável por quaisquer erros ou omissões.

O autor não será responsabilizado para com qualquer pessoa ou entidade com relação a qualquer perda ou dano causado ou alegado a ser causado direta ou indiretamente por este eBook.

Este eBook contém exemplos de código que você pode usar em seus próprios projetos.

Você pode acessar todos os códigos em: <http://bit.ly/2zH7z7u>

Se achar algum erro, ou tiver sugestões de tópicos ou melhorias, me envie um e-mail: [fs.embarcados@gmail.com](mailto:fs.embarcados@gmail.com)



Este obra está licenciado com uma Licença [Creative Commons Atribuição-NãoComercial-CompartilhaIgual 4.0 Internacional](https://creativecommons.org/licenses/by-nc-sa/4.0/).

# Sumário

<b>Sumário</b>	<b>4</b>
<b>Sobre o Autor</b>	<b>5</b>
<b>Introdução</b>	<b>6</b>
<b>Internet das coisas</b>	<b>7</b>
<b>ESP8266</b>	<b>9</b>
<b>ESP8266 com Arduino IDE</b>	<b>14</b>
Arduino IDE	14
Configuração da IDE Arduino para programar o ESP8266	18
Blink - Hello World	24
Conexão WIFI	26
<b>ESP8266 como Web server</b>	<b>29</b>
Web Server Hello	29
Acionando o LEDs com webserver	32
<b>MQTT (Message Queue Telemetry Transport)</b>	<b>35</b>
Por que MQTT?	35
Broker	38
Criando uma instância no CloudMQTT	39
MQTT DASH	41
<b>Projeto com MQTT Dash e CloudMQTT para acionamento de lâmpada com ESP8266</b>	<b>42</b>
Materiais necessários	42
Circuito	43
Código para placa ESP8266 no Arduino	44
Teste de escrita no tópico com CloudMQTT	48
Configurando a aplicação no MQTT Dash	50
Desafio	56
<b>Referências</b>	<b>57</b>

## Sobre o Autor



### Fábio Souza

Engenheiro com experiência no desenvolvimento de projetos eletrônicos embarcados. Hoje é diretor de operações do portal Embarcados, onde trabalha para levar conteúdos de eletrônica, sistemas embarcados e IoT para o Brasil. Também atua no ensino eletrônica e programação pelo Brasil. É entusiasta do movimento maker, da cultura DIY e do compartilhamento de conhecimento, publica diversos artigos sobre eletrônica e projetos open hardware, como o projeto Franzininho Participou da residência hacker 2018 no Red bull Basement. Quando não está ministrando palestras, cursos ou workshops, dedica seu tempo “escovando bits” ou projetando placas eletrônicas.

Me adicione nas redes sociais: <https://about.me/fabio.souza>

## Introdução

O controle de dispositivos remotamente sempre foi um assunto muito procurado pelo makers e entusiastas. Existem diversos projetos com controle com acionamento remoto via comunicação sem fio. Com a facilidade de acesso a internet através de dispositivos mobile, aliado a disponibilidade de placas com conexão WIFI, hoje ficou fácil controlar dispositivos remotamente, em qualquer lugar do mundo.

Esse material visa apresentar os primeiros passos para se trabalhar com placas baseadas em ESP8266 para acionamento de dispositivos remotamente, seja através de um navegador web ou através de um aplicativo mobile publicando em um broker MQTT.

Também serão apresentados os conceitos básicos de IoT, MQTT e aplicações práticas.

Ao final você estará apto para criar aplicações e controlar dispositivos em sua casa.

Bora fazer!

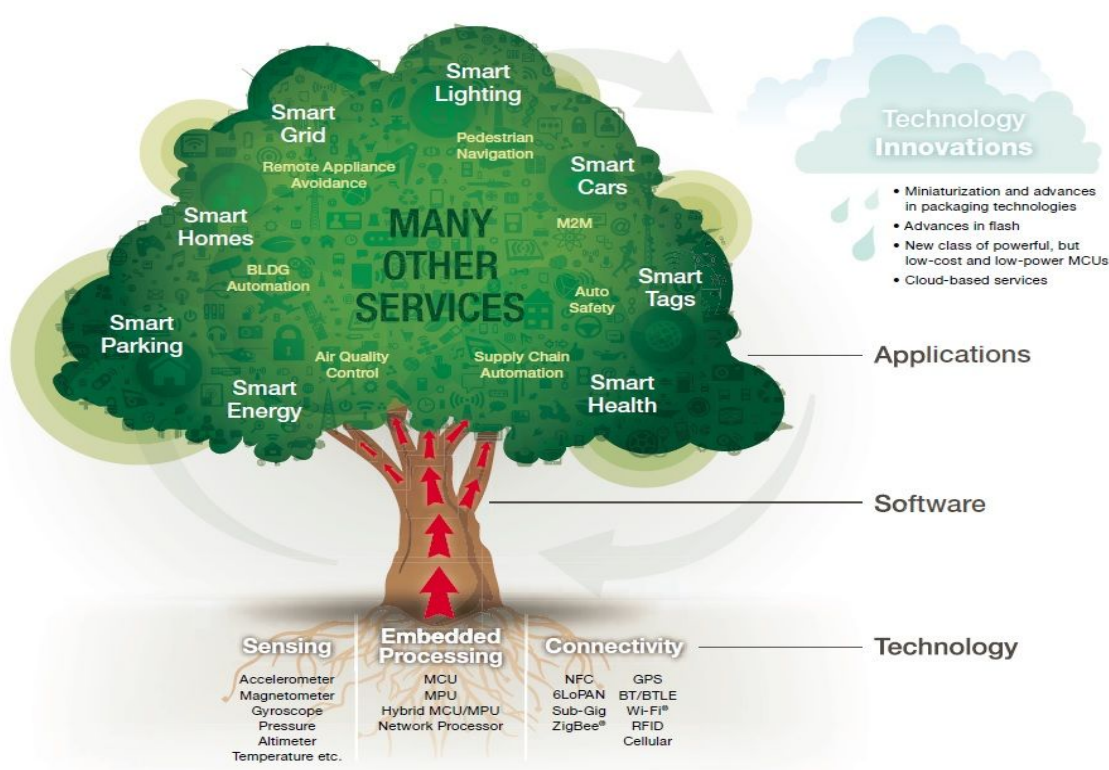
## Internet das coisas

A Internet das coisas, ou em inglês Internet of Things, é um termo que ficou em alta nos últimos anos. Mas esse termo não é tão novo assim, Kevin Ashton trouxe esse termo em 1999 durante uma apresentação sobre uma solução de RFID que ele estava trabalhando. Além disso, Mark Weiser, autor do *The computer of the 21 st Century*, trouxe o conceito de ubiquidade no início dos anos 90.

A internet das coisas é uma rede de objetos físicos que possuem tecnologia embarcada para comunicar, captar sinais e interagir consigo mesmos ou com o ambiente externo.

Os sensores são chamados de “coisas” e podem ser qualquer objeto, possuindo seu próprio IP e com capacidade de enviar e receber dados por uma rede.

O gráfico a seguir dá uma noção da Internet das coisas:



Fonte:

[https://www.researchgate.net/figure/The-IoT-Different-Services-Technologies-Meanings-for-Everyone-77\\_fig6\\_278798179](https://www.researchgate.net/figure/The-IoT-Different-Services-Technologies-Meanings-for-Everyone-77_fig6_278798179)

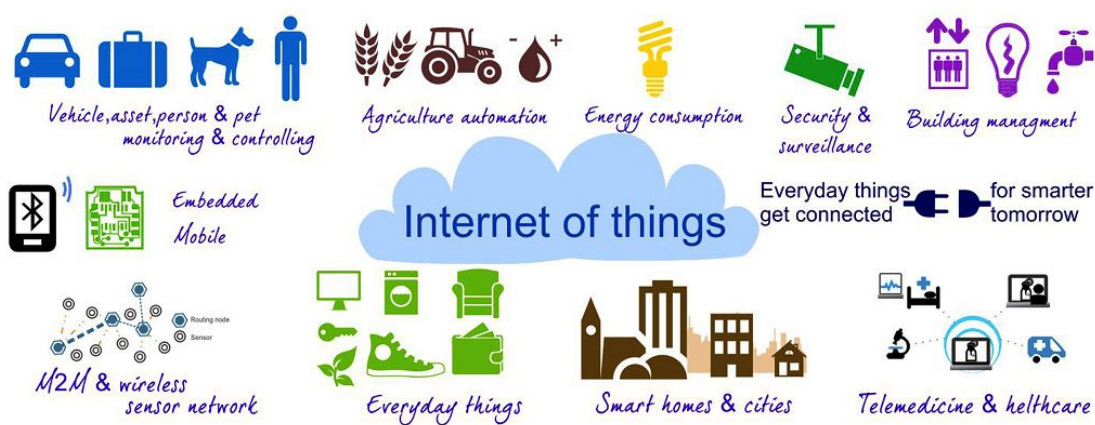


Nas raízes temos os sistemas Embarcados, ou seja, a “coisa” que através de sensores captura as grandezas do ambiente e envia através de algum tipo de comunicação (geralmente sem fio).

No caule temos o software que faz o tratamento dos dados e envia para nuvem.

Na nuvem temos as aplicações. O mesmo dado pode ser aplicado em diferentes serviços.

Existem uma infinidade de aplicações para internet das coisas, em diversas áreas:



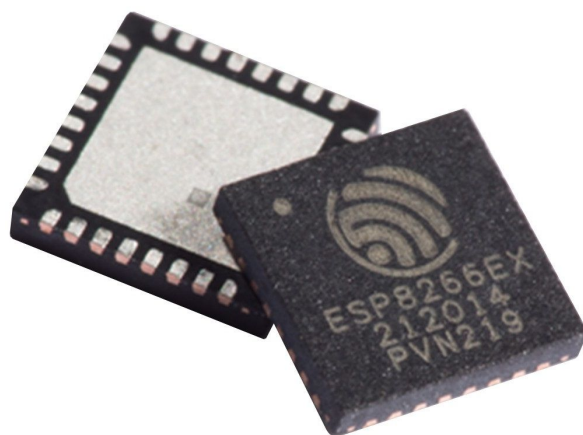
Empresas consagradas e muitas novas empresas estão criando soluções para IoT atualmente.

Veja o [The 2018 Internet of Things Landscape](#)



## ESP8266

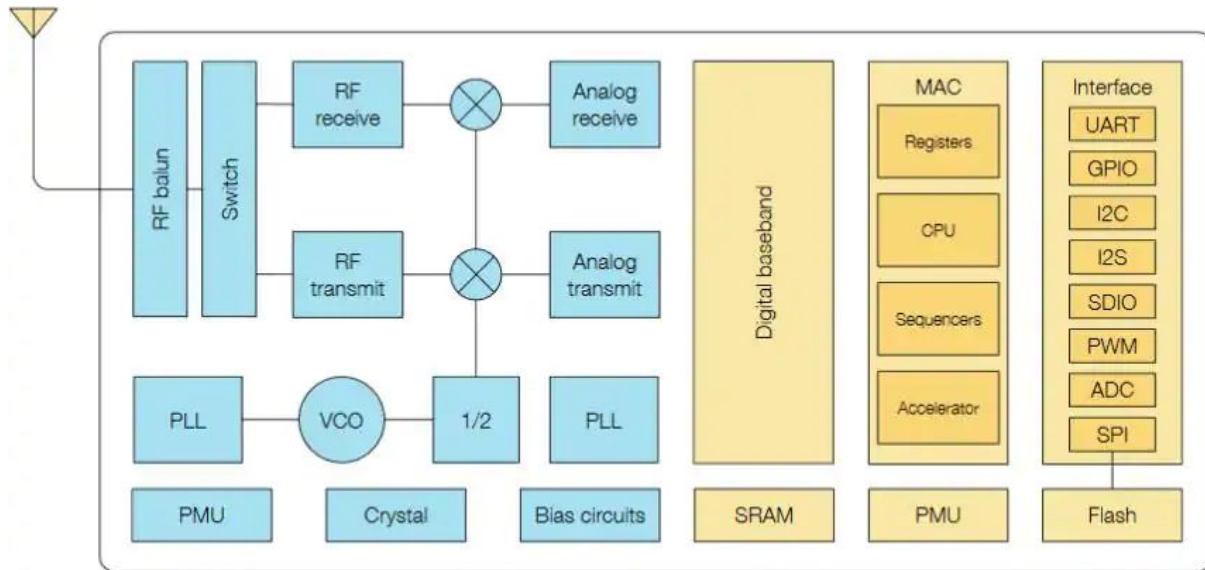
O ESP8266 é SoC da chinesa Espressif. Construído em torno do processador Tensilica Xtensa LX3, inclui pilha TCP/IP integrada que possibilita acesso a rede WIFI. Criado originalmente para ser um adaptador WIFI, rapidamente se tornou independente devido os seus recursos, desenvolvimento de firmware pela comunidade (Apesar da falta de documentação inicial) e principalmente pelo baixíssimo custo.



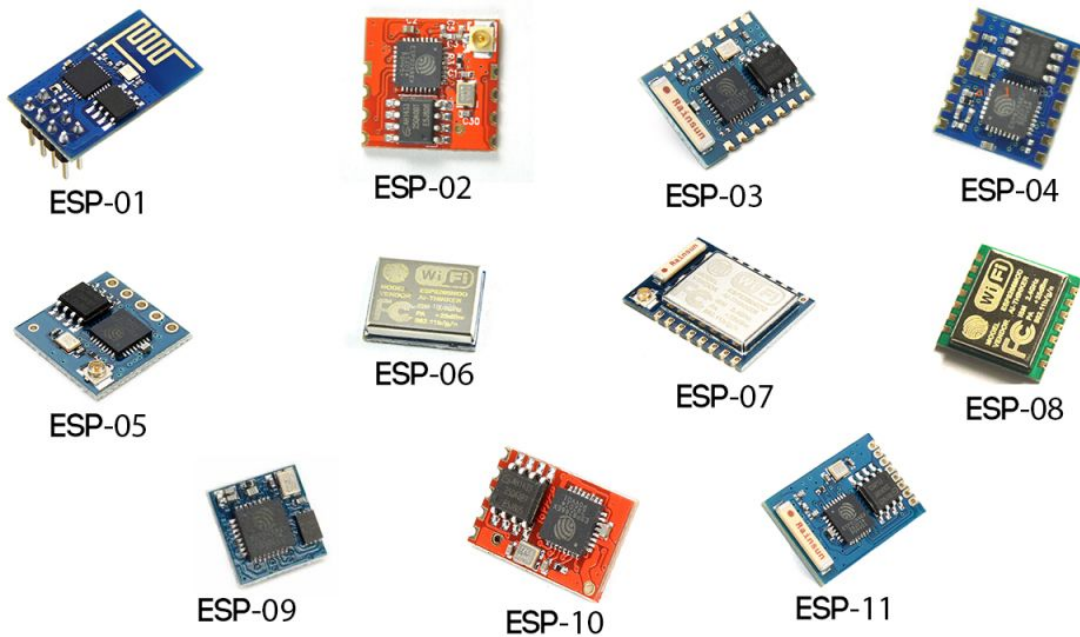
### Características do SoC ESP8266:

- Integrated low power Tensilica L106 32-bit MCU
- 802.11b/g/n
- Wi-Fi 2.4GHz, WPA/WPA2 support
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- Interfaces:SDIO 2.0, SPI, UART, I2C, I2S, PWM, GPIO
- FCC, CE, TELEC, Wi-Fi Alliance, and SRRC certified
- 32-pin QFN (5x5mm)

Diagrama de blocos:



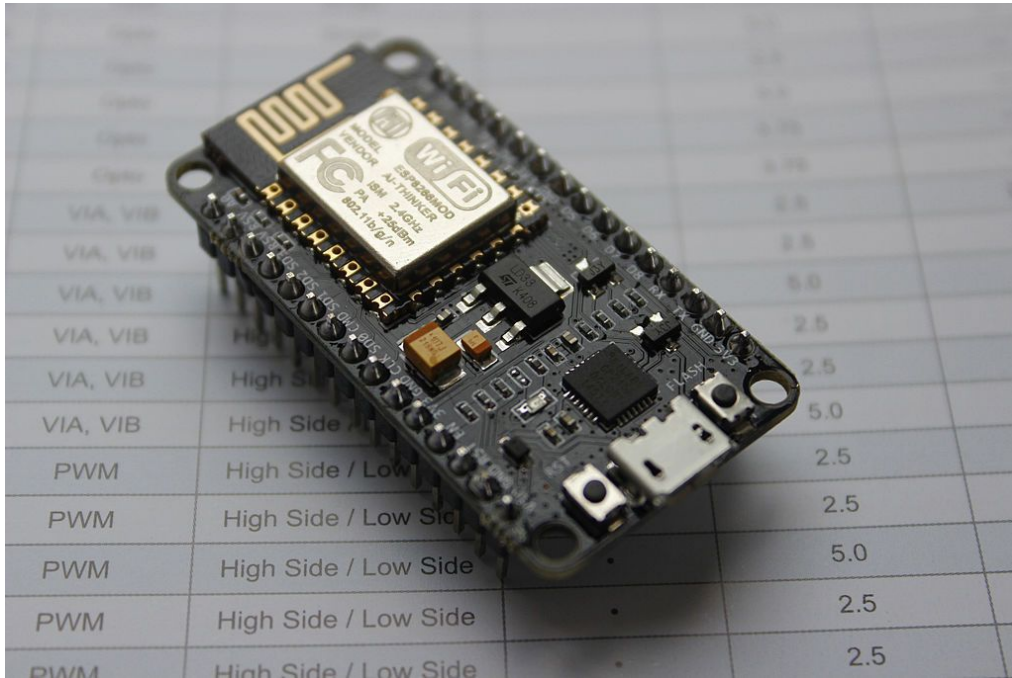
Hoje, o chip é usado em uma infinidade de placas e se tornou a porta de entrada para aplicações IoT:



Fonte:

<https://pyliaorachel.github.io/tutorial/hardware/arduino/2017/04/13/esp8266-with-arduino-trials-and-errors.html>

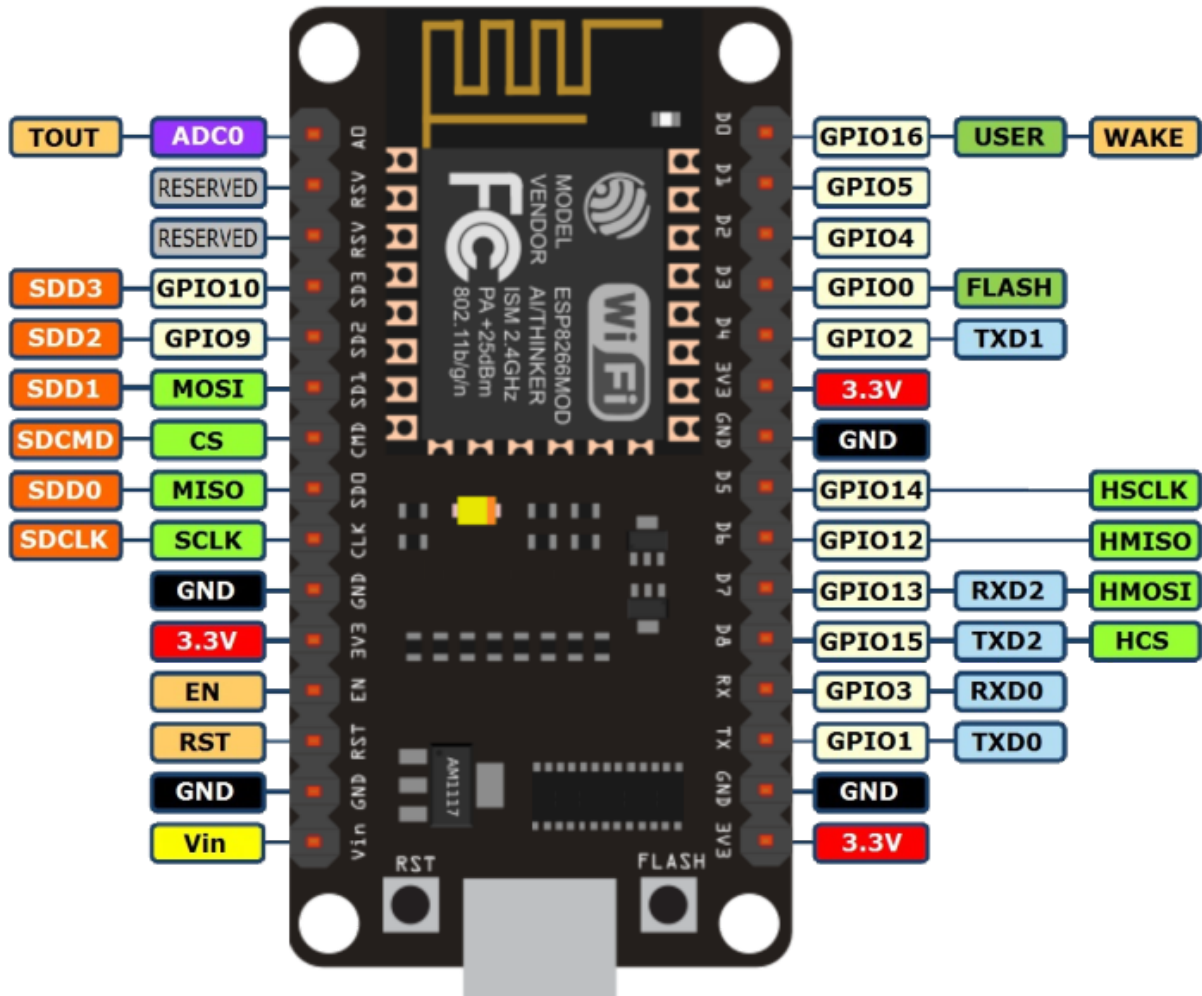
O Módulo ESP-01 se tornou popular devido ao seu baixo custo, porém a dificuldade de usá-lo como standalone, tornou-se uma barreira de entrada para os iniciantes. Felizmente, outras placas foram criadas, como a famosa nodeMCU (que usaremos neste material)



Fonte: <https://en.wikipedia.org/wiki/NodeMCU>

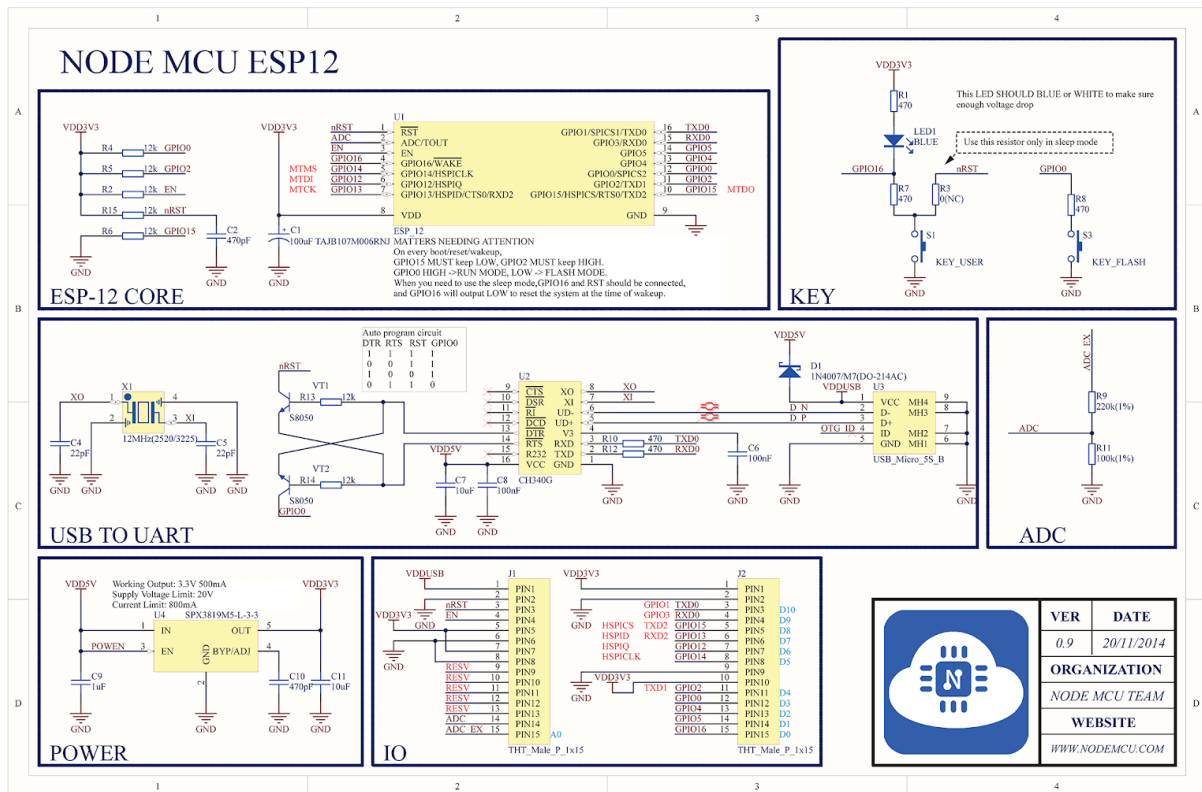
A nodeMCU criada com base no módulo ESP 12E, facilita o processo de programação do ESP8266 por já possuir onboard, o conversor USB serial, regulador de tensão e pino de I/O para conexão em protoboard. Hoje você programa uma placa com ESP8266 diretamente na IDE do Arduino, como se fosse um.

A figura abaixo exibe o pinout da placa:



# Controle Dispositivos Remotamente com o ESP8266

Abaixo é exibido o esquemático da placa:



Existem versões da placa que usam diferentes chips para comunicação serial. A seguir você acessa os drivers para os dois chips mais comuns:

- [CH340](#)
- [CP2102](#)

O projeto é open hardware e você pode acessar os arquivos no github:

<https://github.com/nodemcu/nodemcu-devkit-v1.0>



## ESP8266 com Arduino IDE

Nesta seção, você fará o download, instalará e preparará a Arduino IDE para programar placas com o ESP8266. Usaremos a IDE e facilidades do Arduino para programar o ESP8266, juntos com as bibliotecas criadas. Ao final desse tópico você terá a ambiente pronto para avançar nos projetos.

### Arduino IDE

O ambiente de desenvolvimento Arduino é bem simples de usar. Baseado no processing, facilita a programação das placas Arduino. Por ser uma plataforma de código aberto, possibilita a integração de novas placas, como o caso do ESP8266 que usaremos a seguir. Esse é o repositório de integração do ESP8266 no Arduino: <https://github.com/esp8266/Arduino>

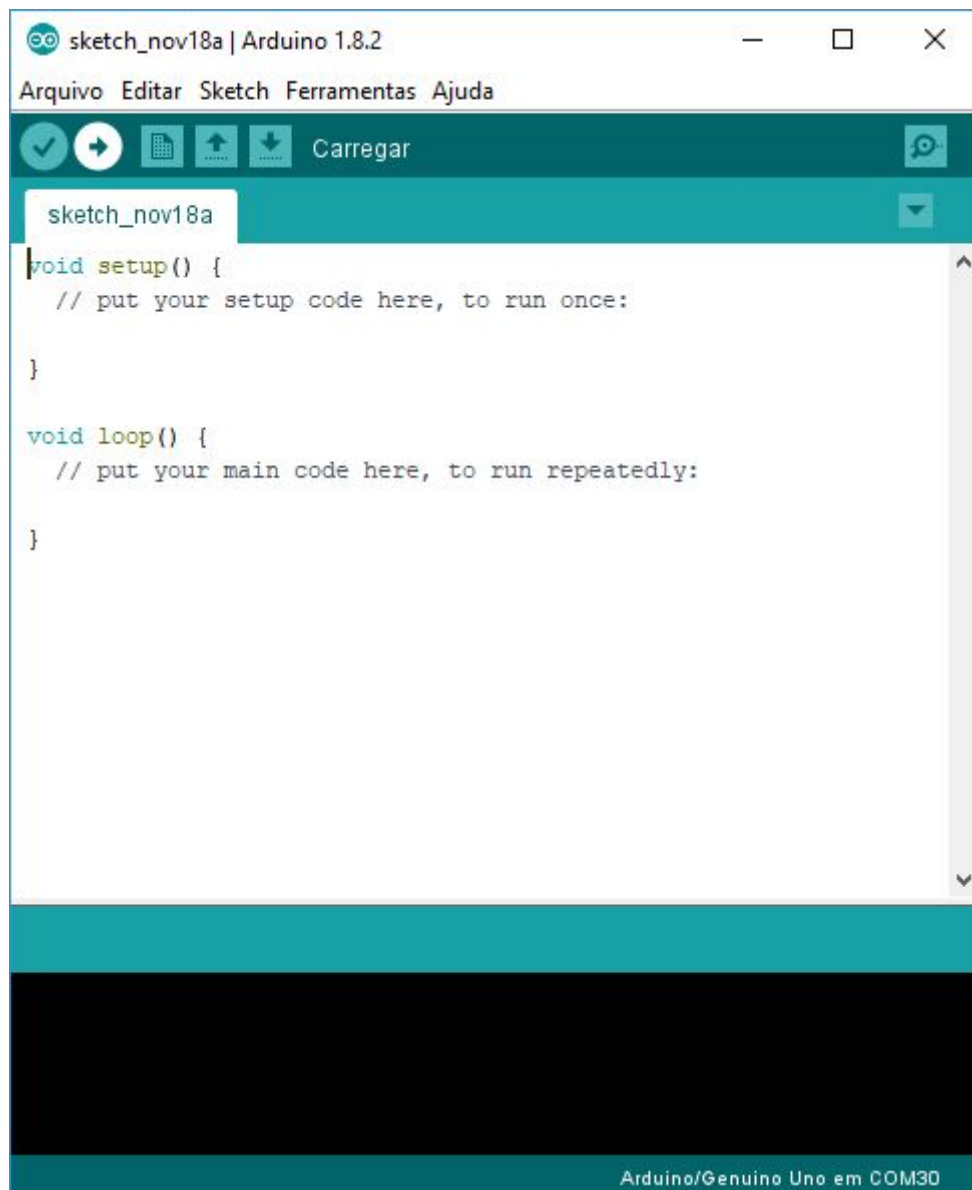
A plataforma Arduino é multiplataforma e necessita do JAVA instalado no computador.

A IDE pode ser baixada gratuitamente no [site do Arduino](http://arduino.cc), onde pode ser escolhida a melhor opção de download conforme o sistema operacional que você utiliza.



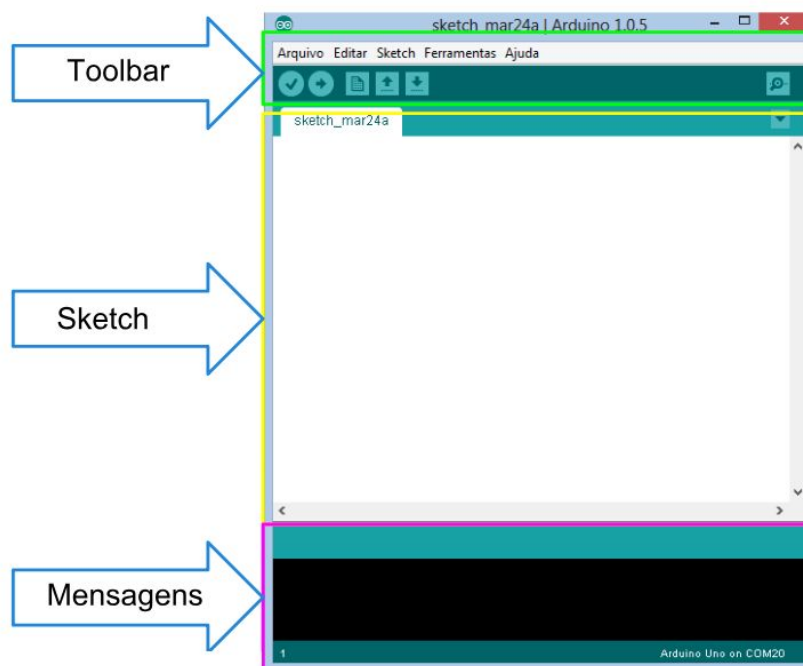
Instale a versão mais adequada para você!

Após a instalação da IDE no seu computador, você pode executar. Quando se abre a IDE do Arduino, será exibido algo semelhante à figura abaixo:





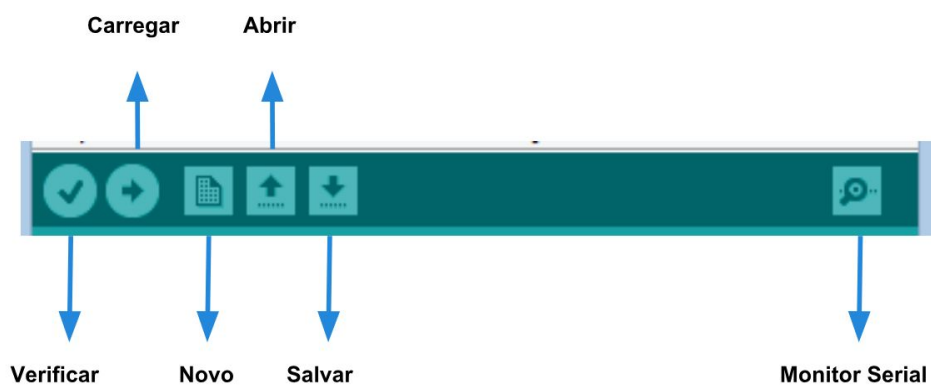
A IDE é dividida em três partes: A Toolbar no topo, o código ou a Sketch Window no centro, e a janela de mensagens na base, conforme é exibido na figura a seguir.



Na Toolbar há uma guia, ou um conjunto de guias, com o nome do sketch. Ao lado direito há um botão que habilita o serial monitor. No topo há uma barra de menus, com os itens File, Edit, Sketch, Tools e Help. Os botões na Toolbar fornecem acesso rápido às funções mais utilizadas dentro desses menus.

Abaixo são identificados os ícones de atalho da IDE:

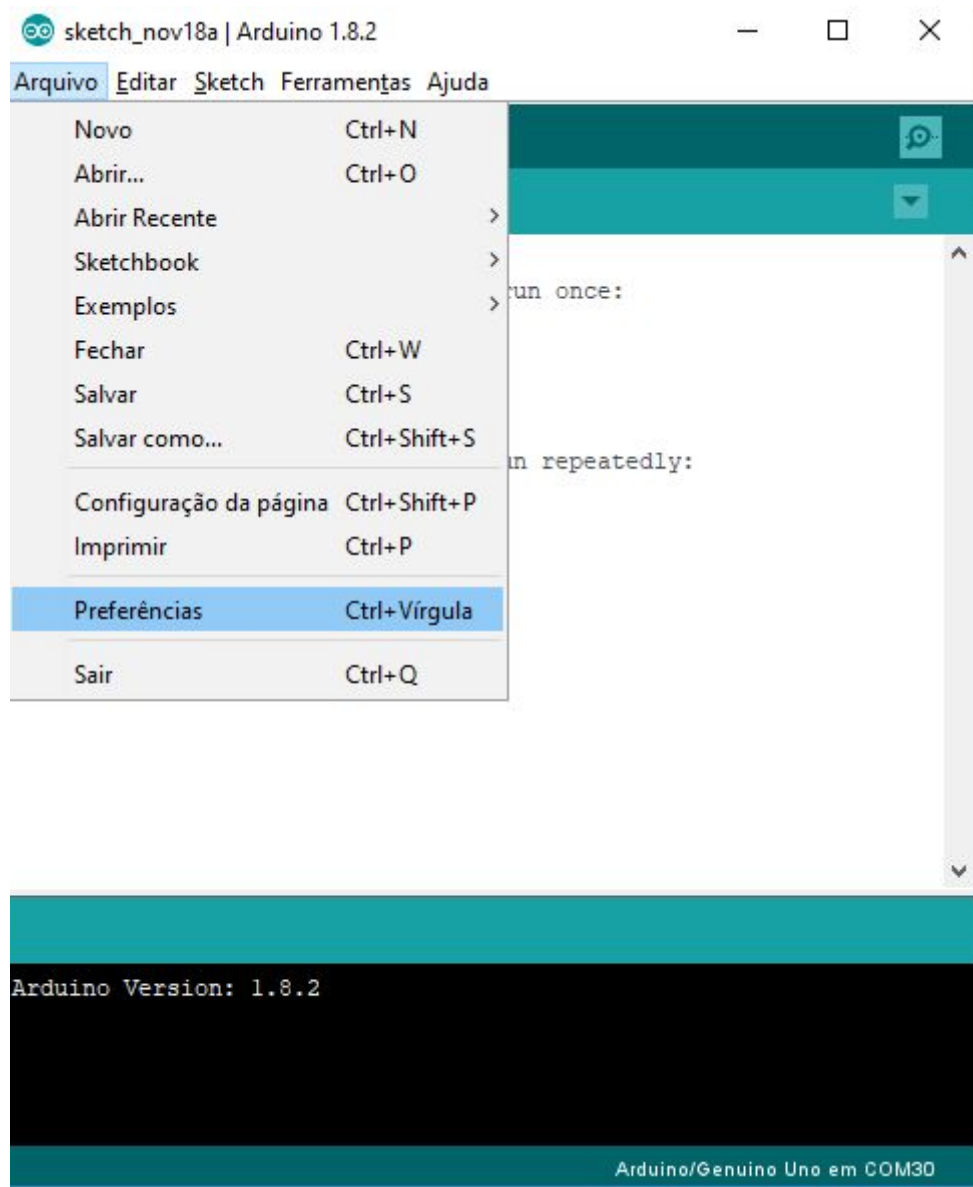
- **Verify**
  - Verifica se existe erro no código digitado.
- **Upload**
  - Compila o código e grava na placa Arduino se corretamente conectada;
- **New**
  - Cria um novo *sketch* em branco.
- **Open**
  - Abre um *sketch*, presente no sketchbook.
- **Save**
  - Salva o *sketch* ativo
- **Seria monitor**
  - Abre o monitor serial.



Os demais comandos presentes na barra de menus podem ser consultados através do menu **Ajuda > Ambiente**.

## Configuração da IDE Arduino para programar o ESP8266

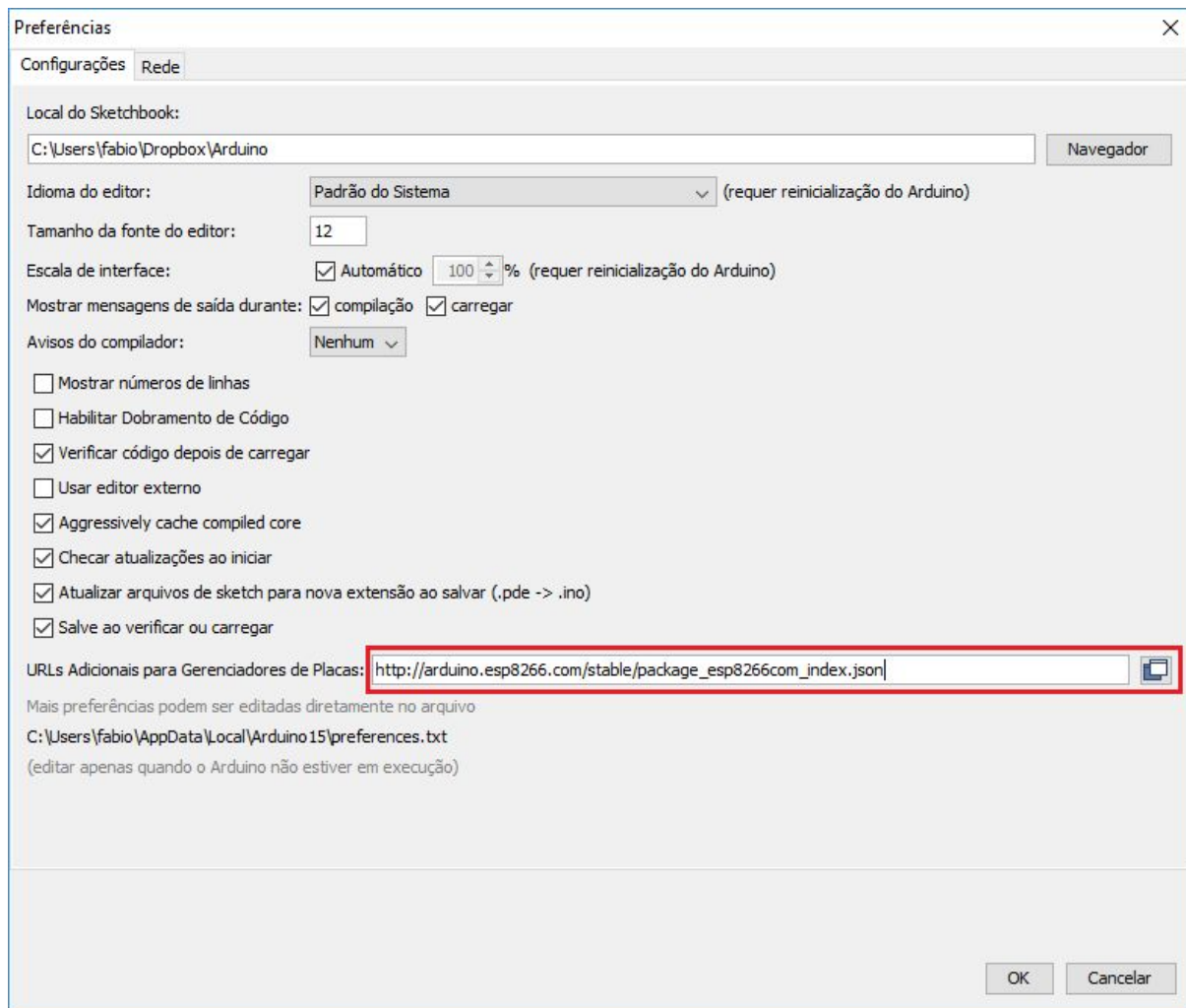
Na IDE acesse **Arquivo > Preferências**



Na janela de configurações de preferências, digite a URL abaixo no campo “**URLs adicionais de Gerenciadores de Placas**”:

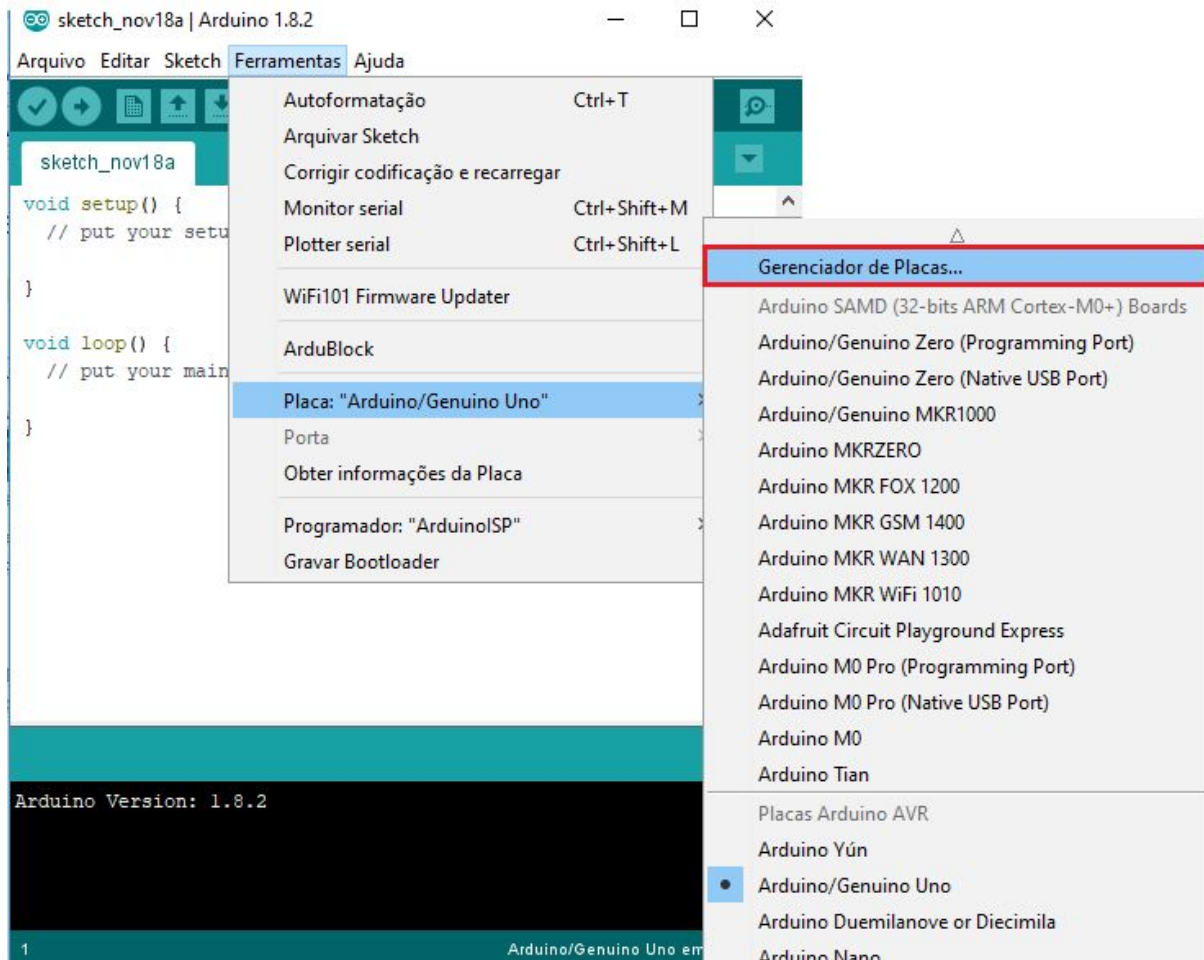
[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

## Controle Dispositivos Remotamente com o ESP8266



Clique em **OK**.

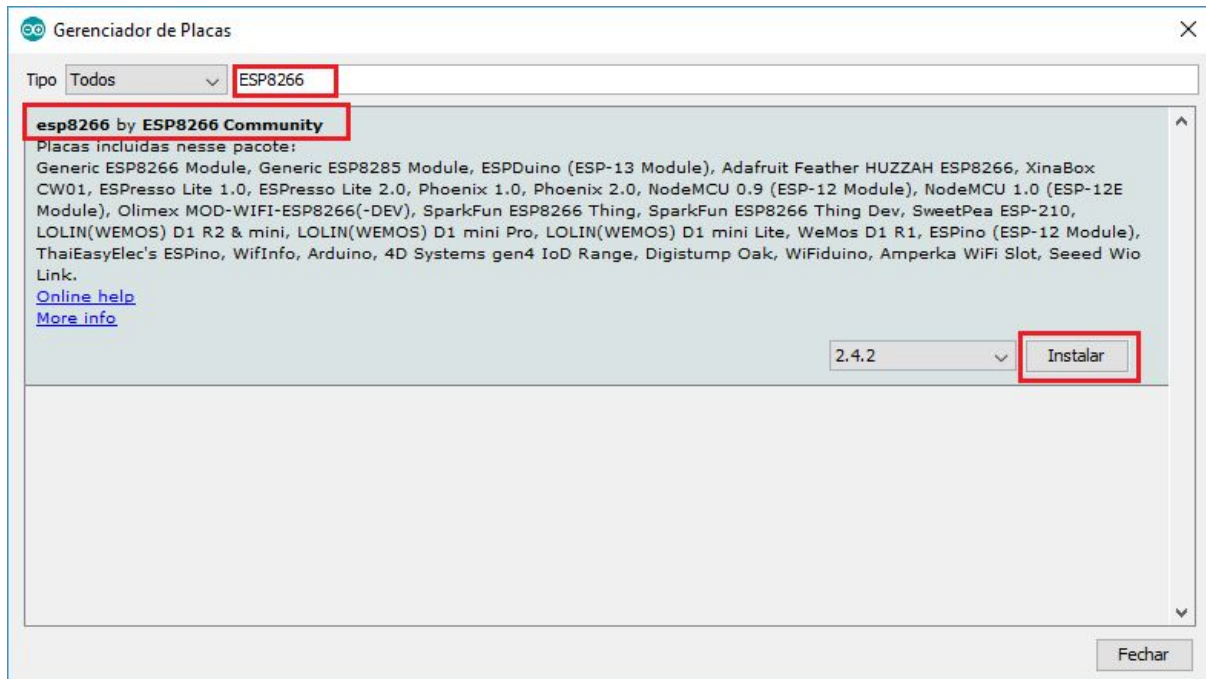
Agora acesse **Ferramentas > placas > Gerenciadores de Placas:**



Aguarde alguns segundos para atualização dos índices de plataformas.

Procure por **ESP8266**.

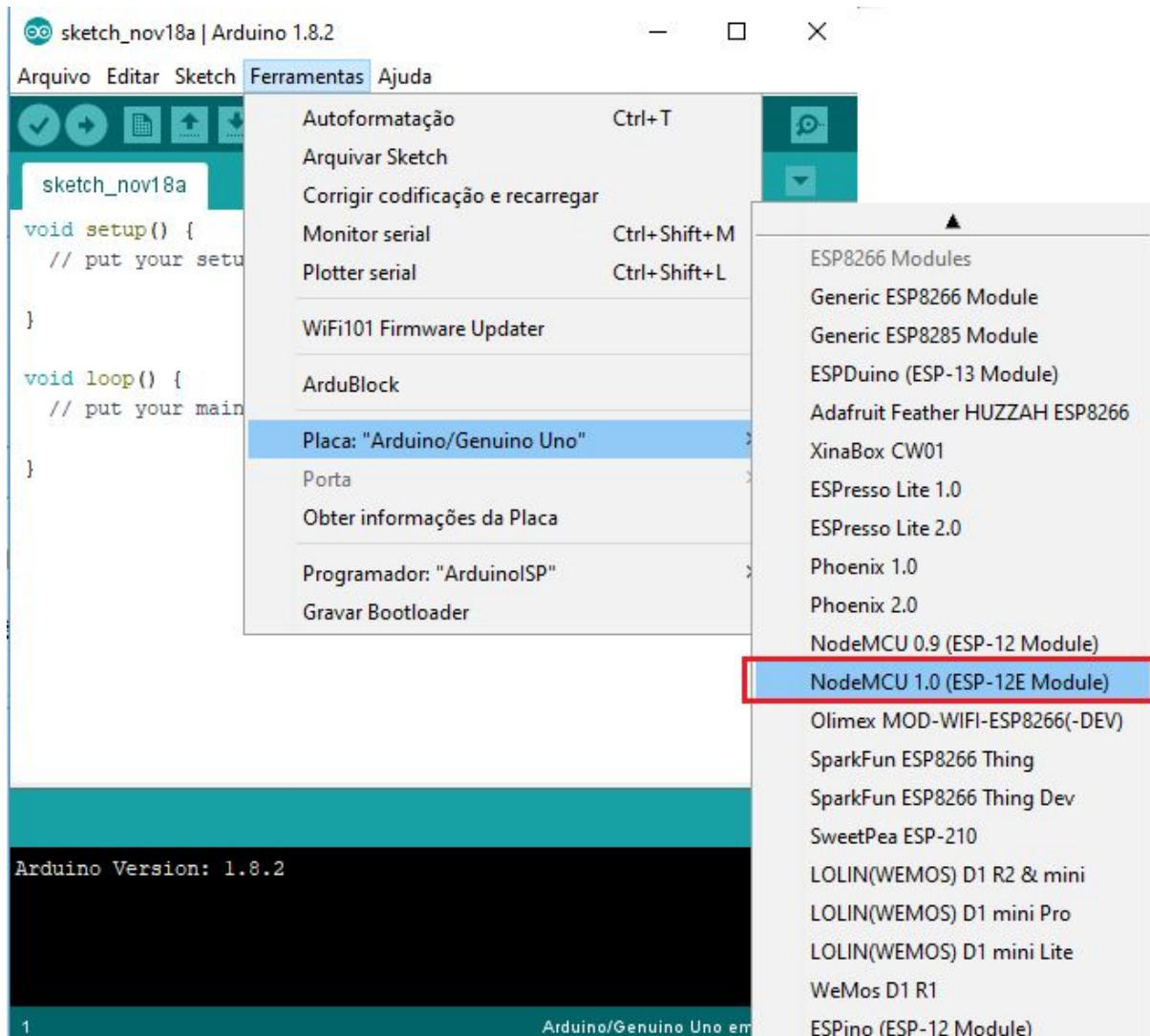
Encontrando a opção “**esp8266 by ESP8266 Community**” clique em instalar:



Aguarde a Instalação e então clique em Fechar

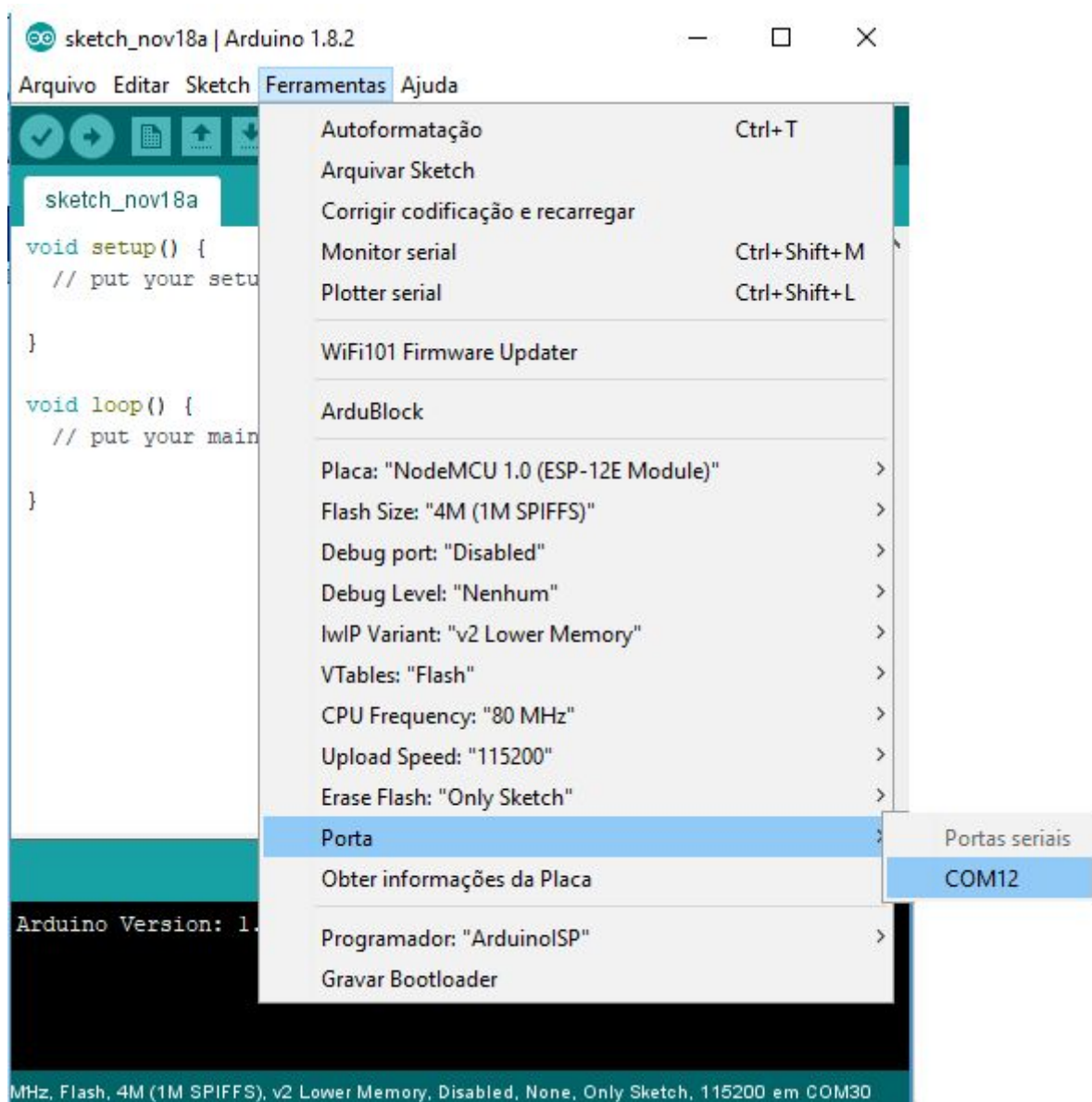
**Obs: É necessário ter acesso a internet**

Agora veja a lista de placas ESP8266 instaladas na IDE. Para nosso tutorial, vamos usar a **NodeMCU 1.0 (ESP 12E Module)**:



Conecte a placa ao computador e selecione a porta COM:





Pronto a IDE está configurada para programar a sua placa NodeMCU.

Agora vamos ao primeiro exemplo.

## Blink - Hello World

Para verificar se a configuração da IDE foi feita corretamente e se a placa está funcionando, vamos fazer um blink para piscar o LED da placa em intervalos de 1 segundo.

Digite o seguinte código:

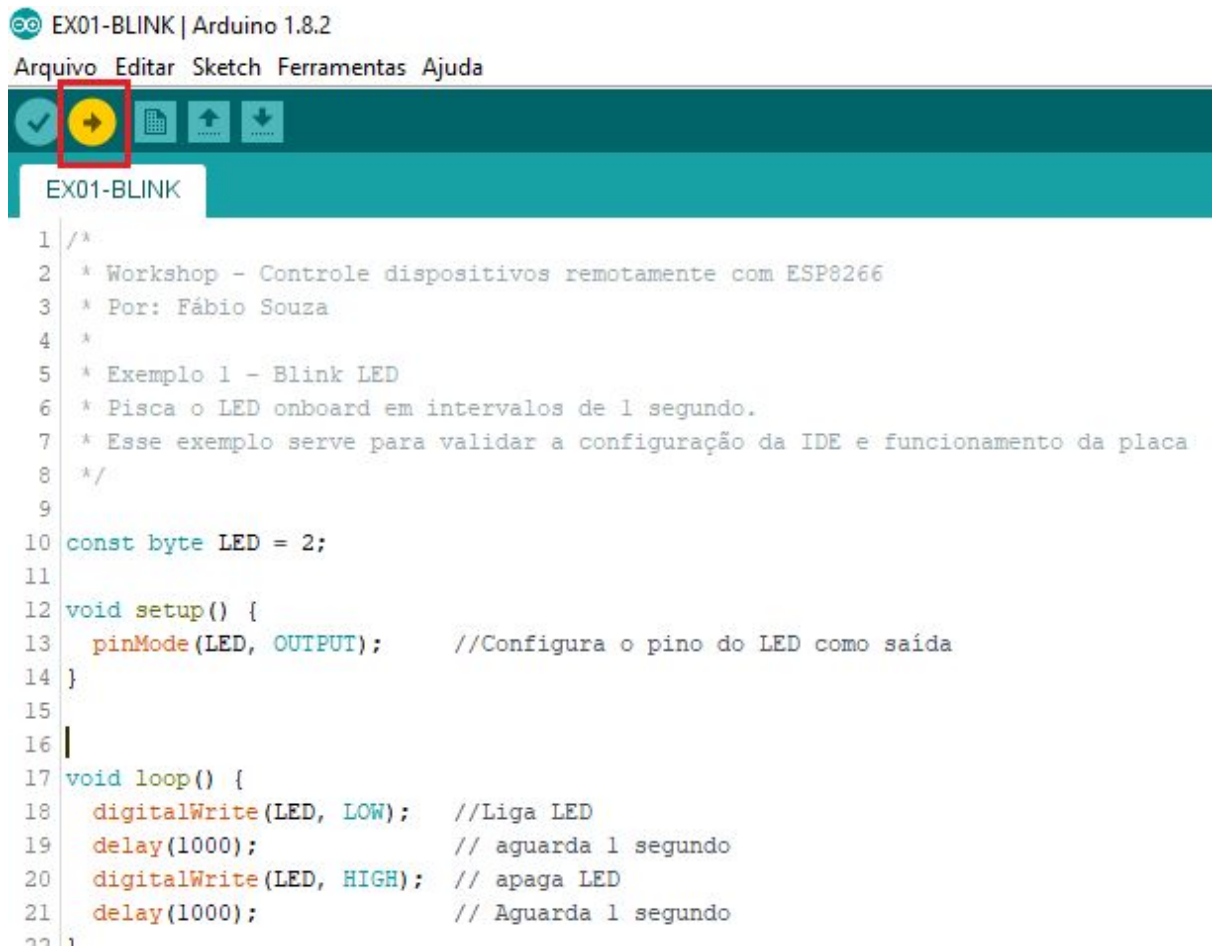
```
/*
 * Workshop - Controle dispositivos remotamente com ESP8266
 * Por: Fábio Souza
 *
 * Exemplo 1 - Blink LED
 * Pisca o LED onboard em intervalos de 1 segundo.
 * Esse exemplo serve para validar a configuração da IDE e funcionamento da placa
 */

const byte LED = 2;

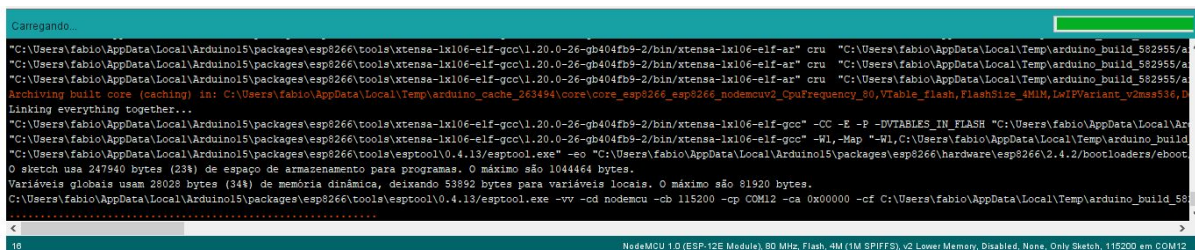
void setup() {
  pinMode(LED, OUTPUT);    //Configura o pino do LED como saída
}

void loop() {
  digitalWrite(LED, LOW);  //Liga LED
  delay(1000);             // aguarda 1 segundo
  digitalWrite(LED, HIGH); // apaga LED
  delay(1000);             // Aguarda 1 segundo
}
```

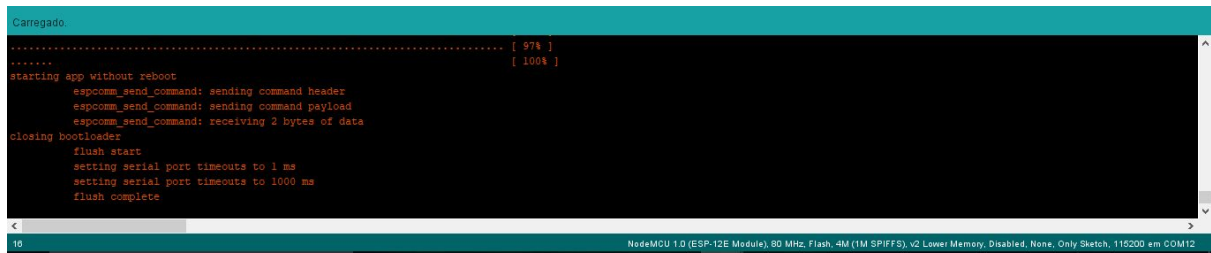
Após digitar o código, clique no botão **carregar**:



Será iniciado o processo de compilação e em seguida o carregamento.



Acompanhe o processo através da barra de carregamento e das mensagens. Após alguns segundos o processo será finalizada e será exibida a mensagem **“Carregado”**:



```
Carregado.
..... [ 97% ]
..... [ 100% ]
starting app without reboot
  espcomm_send_command: sending command header
  espcomm_send_command: sending command payload
  espcomm_send_command: receiving 2 bytes of data
closing bootloader
  flush start
  setting serial port timeouts to 1 ms
  setting serial port timeouts to 1000 ms
  flush complete
```

Verifique se o LED está piscando na placa.

Parabéns! Você fez a configuração da IDE corretamente para programar sua placa. Agora vamos para o próximo nível!

## Conexão WIFI

Para conectar nossa placa ao WIFI vamos usar a biblioteca **ESP8266Wifi.h**.

Digite o código a seguir; Lembre -se de atualizar os campos “ssid” e “senha”, com os valores da rede que deseja conectar:

```
/*
 * Workshop - Controle dispositivos remotamente com ESP8266
 * Por: Fábio Souza
 *
 * Exemplo 2 - WIFI
 * Conecta a placa a uma REDE WIFI
 */

#include <ESP8266Wifi.h>

//configurações da rede
const char* ssid = "ssid";
const char* senha = "senha";

void setup() {
  Serial.begin(115200); //configura comunicação serial
  delay(10);
```

## Controle Dispositivos Remotamente com o ESP8266

```
// mensagem de debug serial
Serial.print("Conectando para a rede ");
Serial.println(ssid);

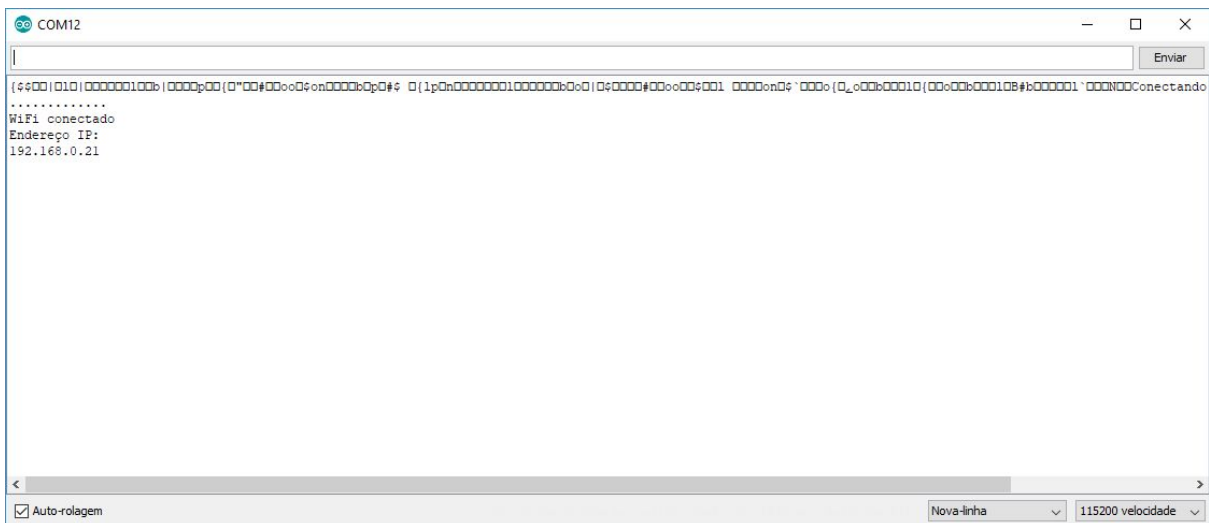
// Iniciando a conexão WiFi
WiFi.begin(ssid, senha);

// aguarda a conexão WIFI
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}

//mensagem de conectado
Serial.println("");
Serial.println("WiFi conectado");
Serial.println("Endereço IP: ");
Serial.println(WiFi.localIP());
}

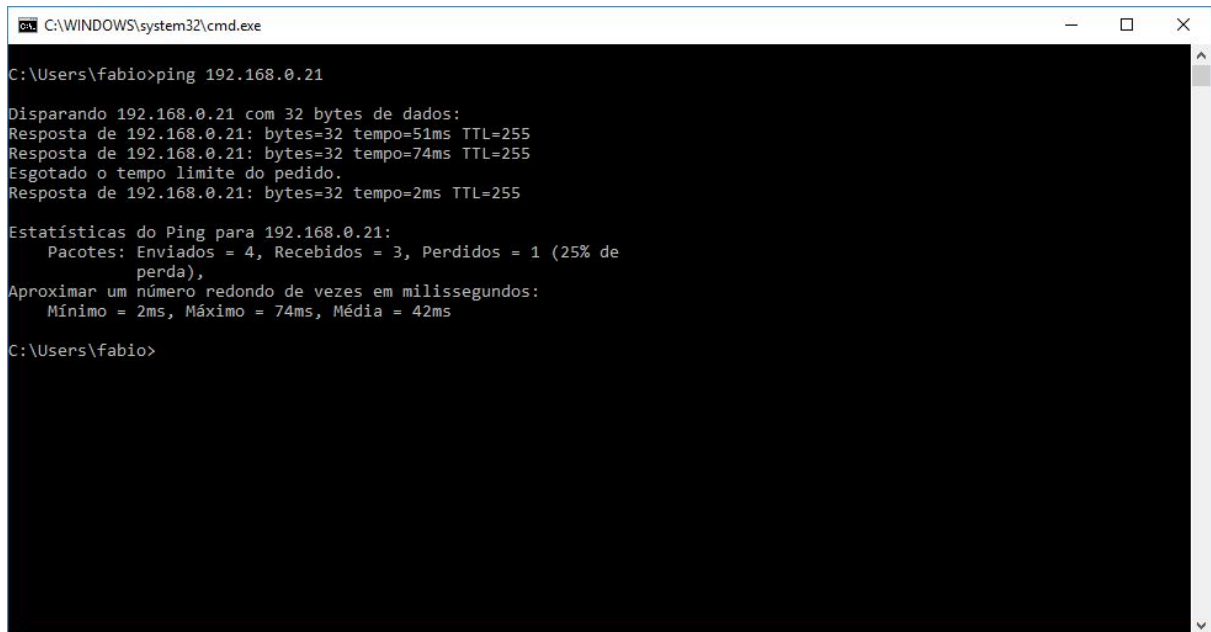
void loop(){
//não faz nada no Loop
}
```

Abra o terminal Serial e veja se sua placa conectou a rede:



Você pode dar um ping no ping no IP que foi conectado:

## Controle Dispositivos Remotamente com o ESP8266



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Fabio>ping 192.168.0.21

Disparando 192.168.0.21 com 32 bytes de dados:
Resposta de 192.168.0.21: bytes=32 tempo=51ms TTL=255
Resposta de 192.168.0.21: bytes=32 tempo=74ms TTL=255
Esgotado o tempo limite do pedido.
Resposta de 192.168.0.21: bytes=32 tempo=2ms TTL=255

Estatísticas do Ping para 192.168.0.21:
    Pacotes: Enviados = 4, Recebidos = 3, Perdidos = 1 (25% de
    perda),
Aproximar um número redondo de vezes em milissegundos:
    Mínimo = 2ms, Máximo = 74ms, Média = 42ms

C:\Users\Fabio>
```

Pronto, já aprendemos como conectar a placa ao WIFI, agora vamos criar uma aplicação para acionar dispositivos. :)

## ESP8266 como Web server

Agora vamos fazer algo mais bacana com o ESP8266.

Um web server é um dispositivo conectado rede que armazena e serve arquivos. Os clientes podem solicitar tal arquivo ou outro dado, e o servidor enviará os dados/arquivos corretos de volta ao cliente. As solicitações são feitas usando HTTP.

O HTTP ou Hypertext Transfer Protocol é um protocolo baseado em texto usado para se comunicar com servidores (web). Existem vários métodos de solicitação HTTP, os mais comuns são o GET e o POST.

### Web Server Hello

Agora vamos configurar nossa placa com ESP8266 como web server e controlar saídas digitais através de um navegador web.

Como primeiro exemplo vamos criar uma página web simples para exibir informações no navegador:

```
/*
 * Workshop - Controle dispositivos remotamente com ESP8266
 * Por: Fábio Souza
 *
 * Exemplo 3 - webserver hello
 * Cria um webserver na placa
 */

#include <ESP8266WiFi.h>

int x = 0;

//configurações da rede
```



```
const char* ssid      = "ssid";
const char* senha = "senha";

WiFiServer server(80); //instancia o server na porta 80

void setup() {
  Serial.begin(115200); //configura comunicação serial
  delay(10);

  // mensagem de debug serial
  Serial.print("Conectando para a rede ");
  Serial.println(ssid);

  // Iniciando a conexão WiFi
  WiFi.begin(ssid, senha);

  // aguarda a conexão WIFI
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  server.begin();
  Serial.println("Servidor inicializado!");

  //mensagem de conectado
  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Endereço IP: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  // Verifica se o cliente está conectado
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Aguarda dados do cliente
  Serial.println("cliente");
  while (!client.available()) {
    delay(1);
  }

  String txt = ""; //string para montar o html

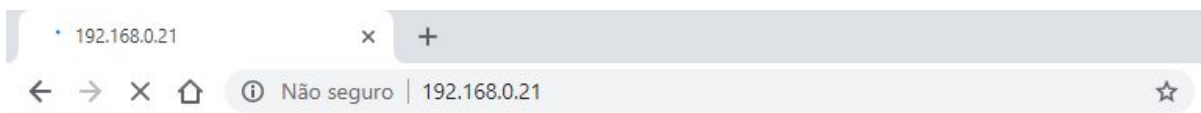
  txt += "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html><meta
http-equiv='refresh' content='3'\r\n";
  txt += "<h1>Ola Makers!</h1>";
}
```

```
txt += "<h2>Workshop ESP8266</h2>";  
txt += "<h3>teste:</h3>";  
txt += "</html>\n";  
client.print(txt);  
client.print(x);  
x++;  
  
delay(3000);  
Serial.println("");  
}
```

Abra o terminal serial e veja qual o IP atribuído a placa:



Então, digite esse IP no navegador do seu computador( ligado a mesma rede que a placa).



## Ola Makers!

### Workshop ESP8266

teste:

## Acionando o LEDs com webserver

Vamos criar agora um botão que irá ligar e desligar o LED da placa:

```
/*
 * Workshop - Controle dispositivos remotamente com ESP8266
 * Por: Fábio Souza
 *
 * Exemplo 4 - webserver IO
 * Aciona I/O através do browser
 */

#include <ESP8266WiFi.h>

#define OUT1 2

boolean statusOUT1 = LOW;

//configurações da rede
const char* ssid      = "ssid";
const char* senha     = "senha";

WiFiServer server(80); //instancia o server na porta 80

void setup() {
  Serial.begin(115200); //configura comunicação serial
  delay(10);

  //configura pino de saída
  pinMode(OUT1, OUTPUT);
  digitalWrite(OUT1, LOW);

  // mensagem de debug serial
  Serial.print("Conectando para a rede ");
  Serial.println(ssid);

  // Iniciando a conexão WiFi
  WiFi.begin(ssid, senha);

  // aguarda a conexão WIFI
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```

}

server.begin();
Serial.println("Servidor inicializado!");

//mensagem de conectado
Serial.println("");
Serial.println("WiFi conectado");
Serial.println("Endereço IP: ");
Serial.println(WiFi.localIP());
}

void loop() {
    // Verifica se o cliente está conectado
    WiFiClient client = server.available();
    if (!client) {
        return;
    }

    // Aguarda dados do cliente
    Serial.println("connect");
    while (!client.available()) {
        delay(1);
    }

    String req = client.readStringUntil('\r');
    Serial.println(req);
    client.flush();

    if (req.indexOf("ioon") != -1)
    {
        digitalWrite(OUT1, HIGH);
        statusOUT1 = HIGH;
    }
    else if (req.indexOf("iooff") != -1)
    {
        digitalWrite(OUT1, LOW);
        statusOUT1 = LOW;
    }

    // Se o Servidor, conseguiu entender a chamada que fizemos acima, Retorna o Valor Lido e
    mostra no Navegador.
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/html");
    client.println("");
    client.println("<!DOCTYPE HTML>");
    client.println("<html>");
    client.println("<h1>Workshop ESP8266</h1>");
}

```

## Controle Dispositivos Remotamente com o ESP8266

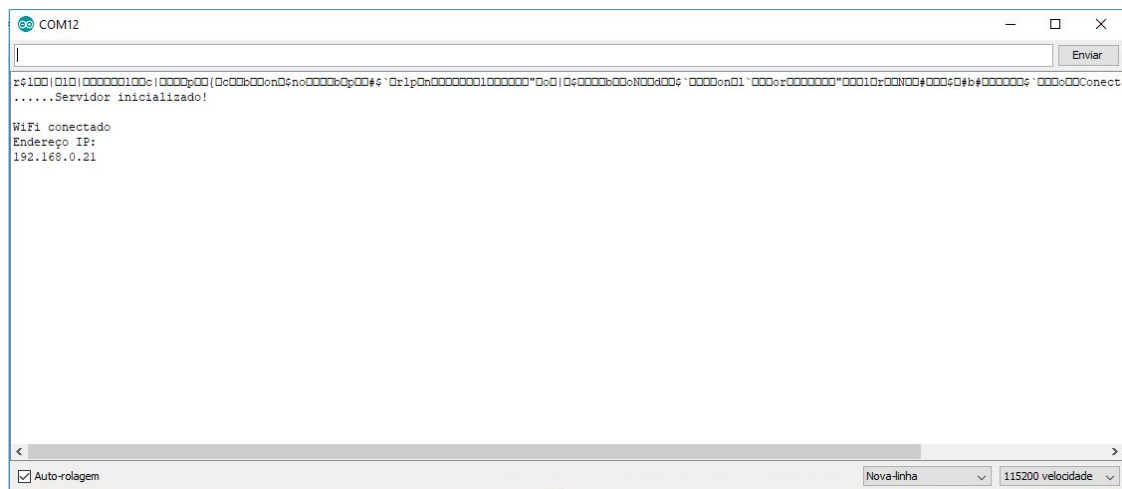
```
if(!statusOUT1)
client.println("<p>OUT1 <a href=\"ioon\"><button>LIGAR</button></a></p>");
else
client.println("<p>OUT1 <a href=\"iooff\"><button>DESLIGAR</button></a></p>");

client.println("</html>");

delay(10);

}
```

Abra o terminal e veja qual IP foi atribuído a sua placa:



Digite esse IP no navegador:



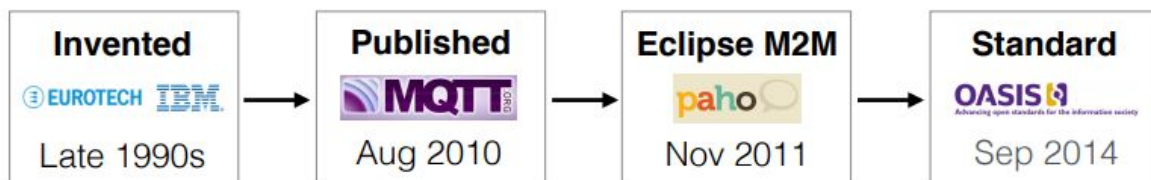
Clique no botão e veja se o estado do LED muda na placa.

## MQTT (Message Queue Telemetry Transport)

Criado pela IBM no final da década de 90, é um protocolo de mensagens leve, utilizado para comunicação Machine to Machine (M2M). É um protocolo de mensagens assíncrono, que facilita a aplicação em diversos cenários em IoT.

Ele usa o um modelo de comunicação de publicação e assinatura, facilitando a expansão de dispositivos no sistema.

Em 2014 ele se tornou oficialmente um padrão aberto OASIS.



Hoje o MQTT é muito utilizado em aplicações IoT e apesar da sua simplicidade, apresenta características como segurança, qualidade de serviço e flexibilidade.

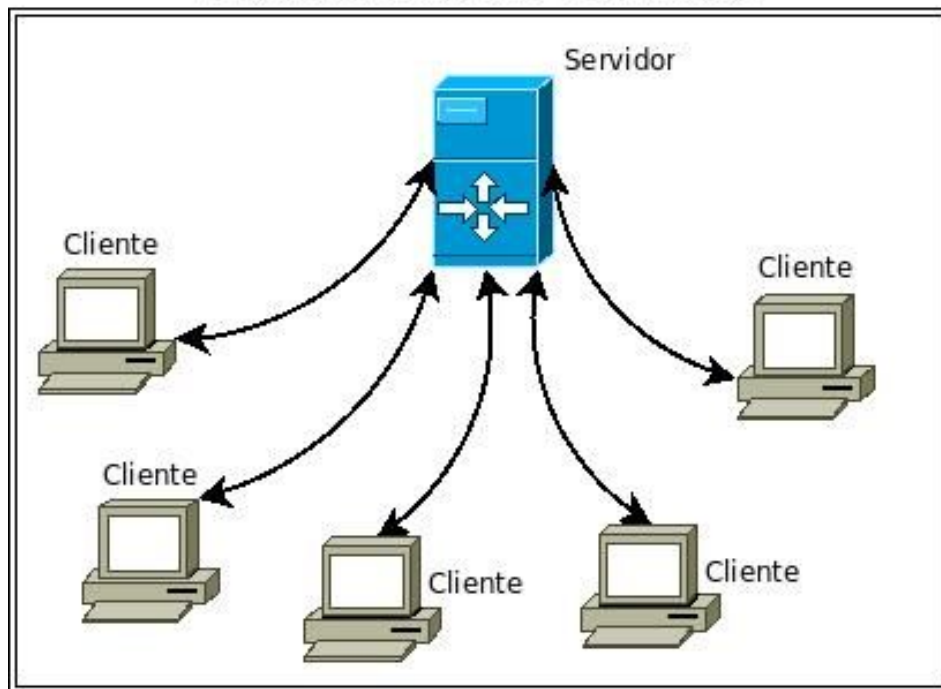
Necessita de pouca banda e processamento, podendo ser usado por hardware mais simplificados, com microcontroladores. É baseado no TCP/IP possui payload menor que o HTTP.

### Por que MQTT?

Apesar das diversas vantagens apresentadas acima, vamos analisar a estrutura de comunicação do MQTT.

É importante lembrarmos do modelo cliente-servidor:

## Modelo Cliente-Servidor

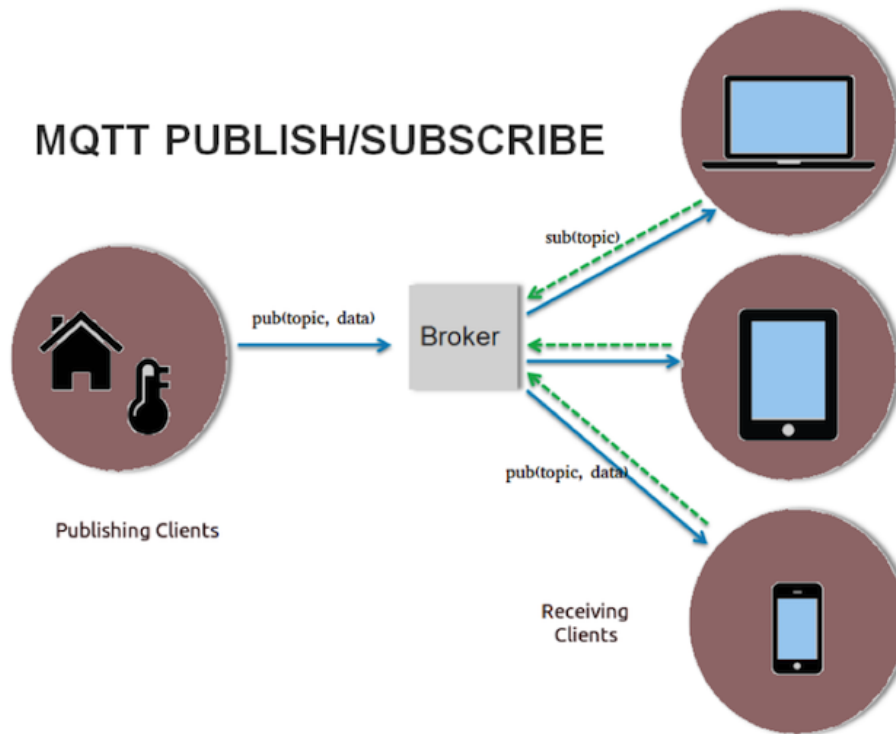


A característica do modelo cliente-servidor, descreve a relação de programas numa aplicação. O componente de servidor fornece uma função ou serviço a um ou mais clientes, que iniciam os pedidos de serviço. É uma comunicação síncrona, o cliente inicia a comunicação e aguarda a resposta do servidor.

Esse modelo é muito utilizado em aplicações WEB, porém não é muito eficiente para aplicações IoT.

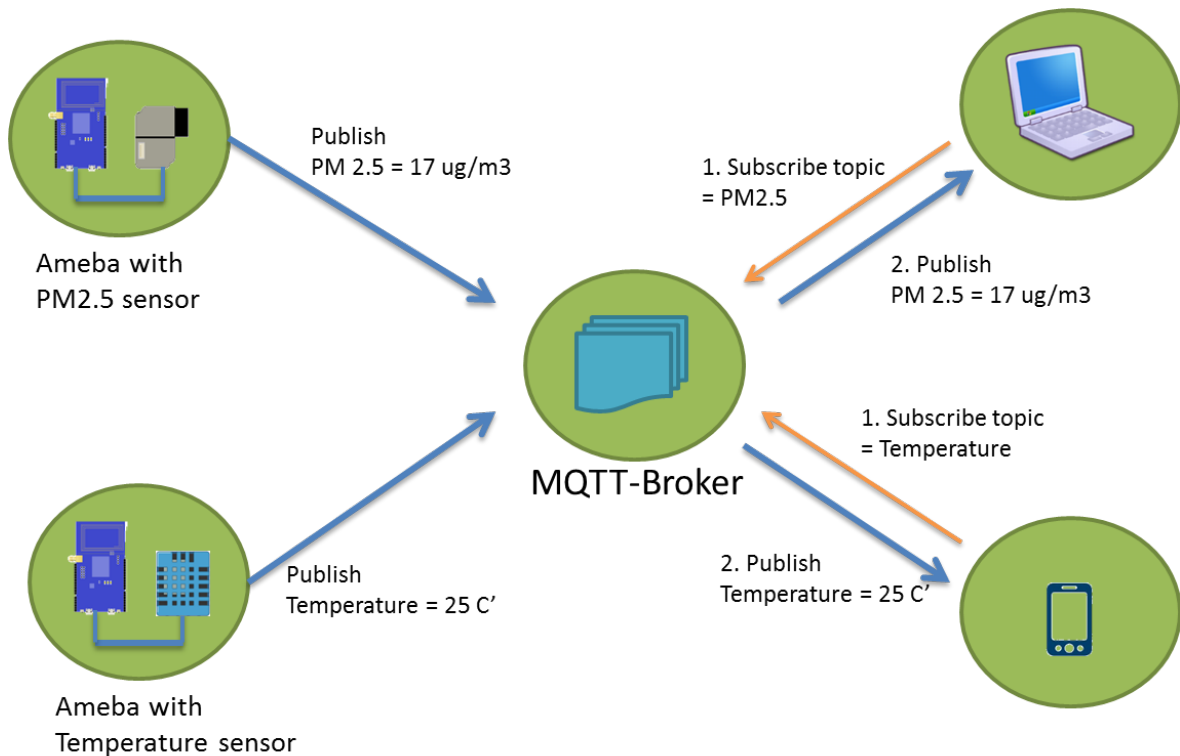
O MQTT tem a arquitetura adequada para aplicações com diversos sensores. Possui dois elementos na rede: Um broker e inúmeros clientes:





O broker é um servidor que recebe as mensagens dos clientes envia para os clientes que assinam os tópicos. O padrão de troca de mensagens utilizado é o publish/subscriber (publicador/subscritor), que facilita a troca de mensagem entre os elementos de forma assíncrona. A comunicação acontece da seguinte forma:

1. O cliente conecta-se ao broker através de uma conexão TCP/IP, assinando um "tópico" de mensagem;
2. O cliente publica as mensagens em um tópico no broker;
3. O broker encaminha a mensagem a todos os clientes que assinam esse tópico.



Exemplo de tópicos:

casa/temperatura

casa/umidade

casa/L1

## Broker

Existem diversas implementações de [brokers MQTT](#), open source ou não. Uma delas, é o [Eclipse Paho](#) que fornece SDKs e bibliotecas do MQTT em várias linguagens de programação.

Você pode instalar o broker em um computador local e interagir através de linha de comando ou conectar dispositivos à rede local. Para fazer download e instalar o módulo mosquitto acesse o [site do mosquitto](#).

Para esse tutorial vamos utilizar o CloudMQTT, um serviço em nuvem que facilitará o controle de dispositivos através da internet.

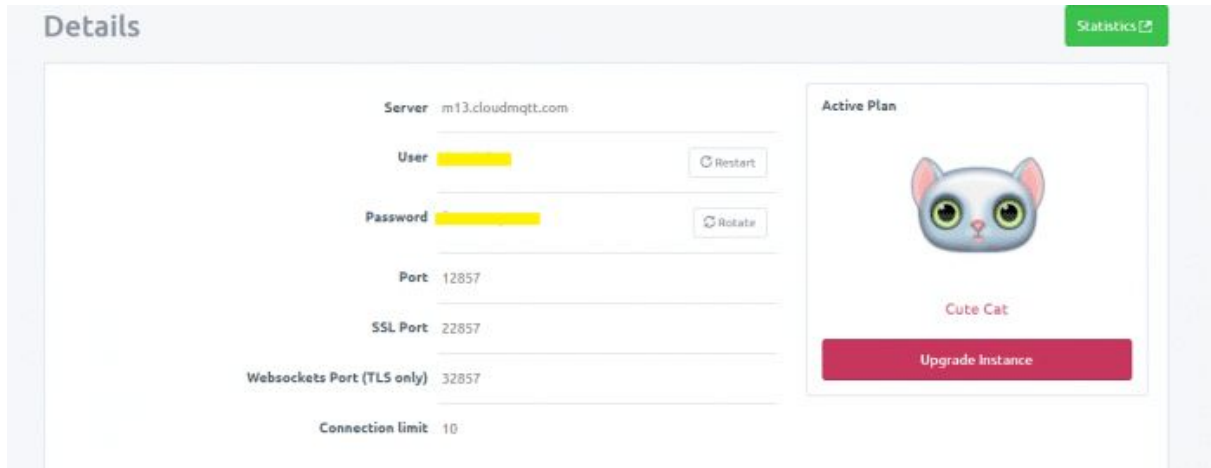
O CloudMQTT possui um plano gratuito que permite 10 conexões com velocidade de 10 Kbit/s.

### Criando uma instância no CloudMQTT

Acesse o [site do CloudMQTT](#) e faça o seu registro. Após o login, clique no botão “+*Create New Instance*”. Será aberta a seguinte página:

Preencha o nome, escolha o plano **Free (Cute Cat)** e clique em *Create a New Instance*.

Abra a instância criada:

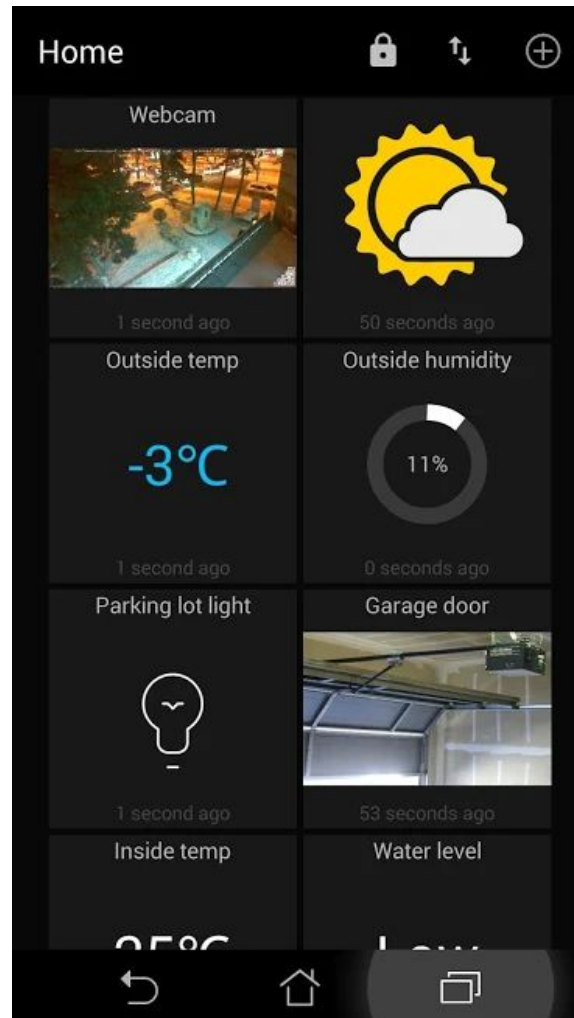


The screenshot shows the 'Details' page of an MQTT Dash instance. The page has a light blue header with the title 'Details' and a green 'Statistics' button. The main content area is divided into two columns. The left column contains configuration fields: 'Server' (m13.cloudmqtt.com), 'User' (masked with yellow), 'Password' (masked with yellow), 'Port' (12857), 'SSL Port' (22857), 'Websockets Port (TLS only)' (32857), and 'Connection limit' (10). There are 'Restart' and 'Rotate' buttons next to the User and Password fields respectively. The right column is titled 'Active Plan' and features a cartoon cat avatar, the text 'Cute Cat', and a red 'Upgrade Instance' button.

Aqui estão as informações necessárias para conexão com o Broker: Server, User, Password e Port. Vamos usar essas informações no código do ESP8266 e no aplicativo MQTT Dash, mais a frente.

## MQTT DASH

O MQTT Dash é um dos melhores aplicativos para interface gráfica no smartphone. Possui uma interface agradável, de fácil customização e configuração, sendo um dos melhores aplicativos que já utilizei para esse fim.



Você pode baixá-lo na [Google Play](https://play.google.com/store/apps/details?id=com.mqtt.dash) e instalar no seu smartphone.

Deixe no jeito, logo vamos configurar nossa aplicação.

## **Projeto com MQTT Dash e CloudMQTT para acionamento de lâmpada com ESP8266**

A seguir apresento como utilizar esse app em conjunto com o ESP8266 e um broker MQTT configurado no CloudMQTT, para comandar uma lâmpada remotamente. Veja como é fácil!

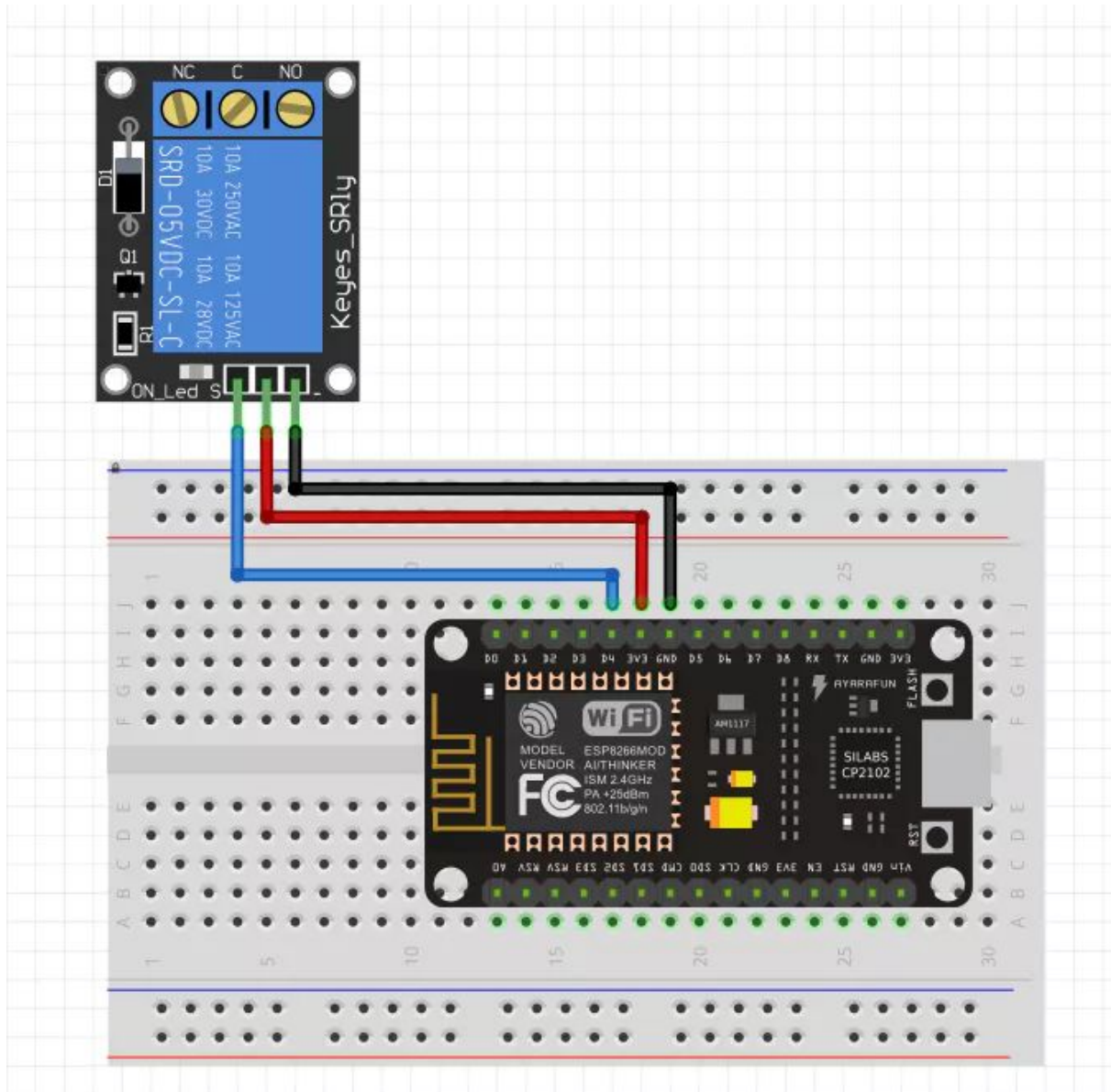
### **Materiais necessários**

1. Placa com ESP8266 (nodeMCU, Wemos D1 ou mini, etc);
2. Módulo RELÊ;
3. Lâmpada;
4. Cabos e Jumpers;

Você também pode testar o exemplo apresentado aqui, sem o uso de relê e lâmpada, apenas usando uma placa com ESP8266 e LEDs em uma protoboard, por exemplo.

**TOME CUIDADO QUANDO FOR TRABALHAR COM ACIONAMENTO DE CARGAS AC.**

## Circuito



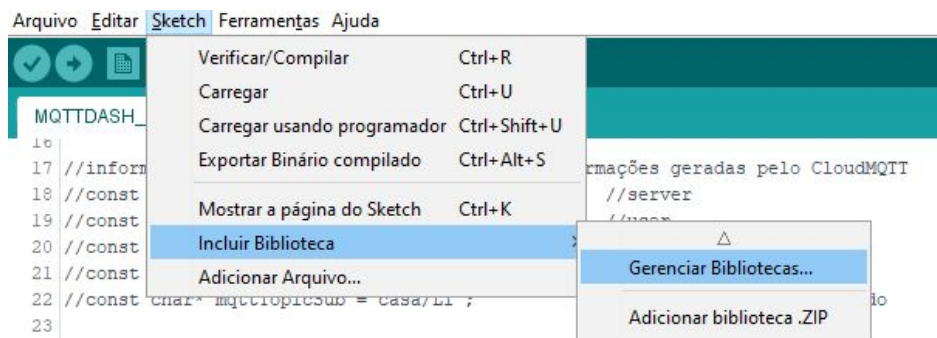
O relé foi ligado ao pino D4 (GPIO2). Você pode ligar em outro pino, só lembre de mapear corretamente no código.

## Código para placa ESP8266 no Arduino

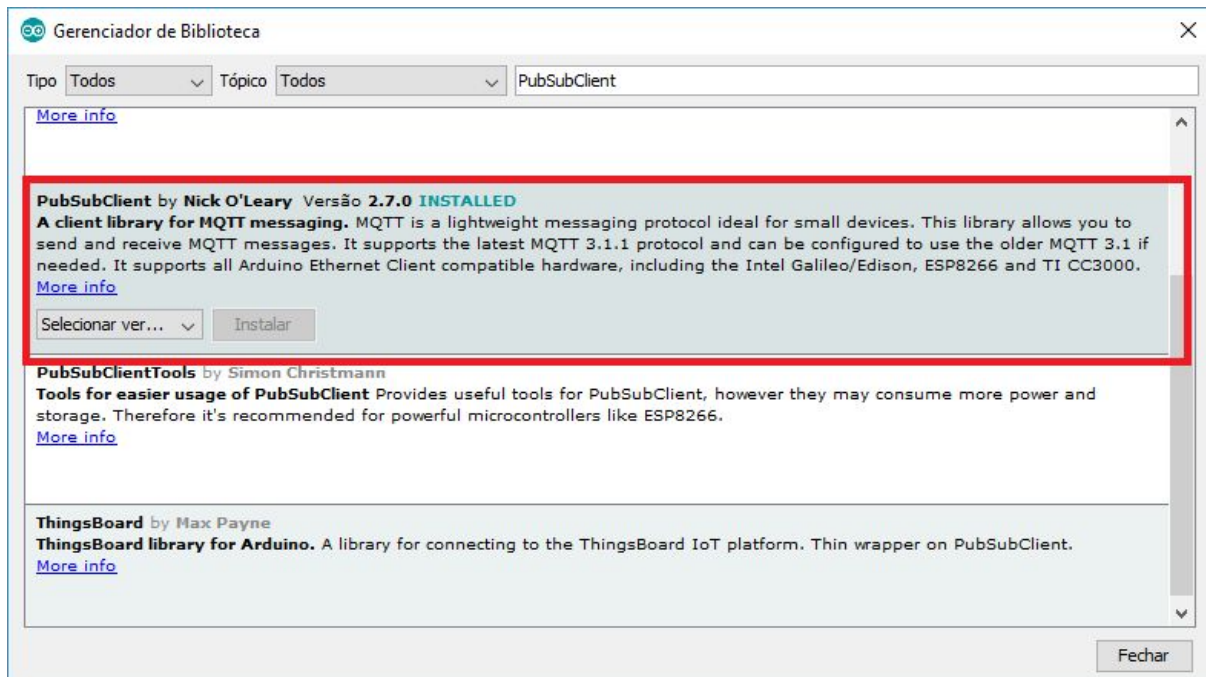
Como subscriber vamos utilizar uma placa nodeMCU que irá acionar uma lâmpada.

Faça as configurações da IDE, conforme apresentado na seção de configuração da IDE.

Você vai precisar instalar a biblioteca PubSubClient.h. Para isso, acesse **Sketch> Incluir Biblioteca> Gerenciar Bibliotecas**:



Procure por PubSubClient e instale a seguinte opção:





No código abaixo, substitua as informações da rede WIFI (ssid, password) e do Broker MQTT (mqttServer, mqttUser, mqttPassword, mqttPort), indicados com x. Compile e carregue para a sua placa com ESP8266.

```
/*
 * Workshop - Controle dispositivos remotamente com ESP8266
 * Por: Fábio Souza
 *
 * Exemplo 5 - MQTT
 * Assina tópico no servidor MQTT
 */

#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#define DEBUG

#define L1 2 //pino de saída para acionamento da Lâmpada L1

//informações da rede WIFI
const char* ssid = "ssid"; //SSID da rede WIFI
const char* password = "senha"; //senha da rede wifi

//informações do broker MQTT - Verifique as informações geradas pelo CloudMQTT
const char* mqttServer = "xxxxxxxxxxxxxxxx"; //server
const char* mqttUser = "xxxxxxx"; //user
const char* mqttPassword = "xxxxxxxxx"; //password
const int mqttPort = xxxxxx; //port

const char* mqttTopicSub = "casa/L1"; //tópico que sera assinado

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {

  Serial.begin(115200);
  pinMode(L1, OUTPUT);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    #ifdef DEBUG
    Serial.println("Conectando ao WiFi..");
    #endif
  }
}
```

```

    #endif
}
#ifdef DEBUG
Serial.println("Conectado na rede WiFi");
#endif

client.setServer(mqttServer, mqttPort);
client.setCallback(callback);

while (!client.connected()) {
    #ifdef DEBUG
    Serial.println("Conectando ao Broker MQTT...");
    #endif

    if (client.connect("ESP8266Client", mqttUser, mqttPassword )) {
        #ifdef DEBUG
        Serial.println("Conectado");
        #endif

    } else {
        #ifdef DEBUG
        Serial.print("falha estado ");
        Serial.print(client.state());
        #endif
        delay(2000);
    }
}

//subscreve no tópico
client.subscribe(mqttTopicSub);
}

void callback(char* topic, byte* payload, unsigned int length) {

    //armazena msg recebida em uma string
    payload[length] = '\0';
    String strMSG = String((char*)payload);

    #ifdef DEBUG
    Serial.print("Mensagem chegou do tópico: ");
    Serial.println(topic);
    Serial.print("Mensagem:");
    Serial.print(strMSG);
    Serial.println();
    Serial.println("-----");
    #endif

    //aciona saída conforme msg recebida

```

```

if (strMSG == "1"){           //se msg "1"
    digitalWrite(L1, LOW); //coloca saída em LOW para ligar a Lampada - > o módulo RELE usado
    tem acionamento invertido. Se necessário ajuste para o seu modulo
}else if (strMSG == "0"){    //se msg "0"
    digitalWrite(L1, HIGH); //coloca saída em HIGH para desligar a Lampada - > o módulo RELE
    usado tem acionamento invertido. Se necessário ajuste para o seu modulo
}

}

//função pra reconectar ao servido MQTT
void reconnect() {
    //Enquanto estiver desconectado
    while (!client.connected()) {
        #ifdef DEBUG
        Serial.print("Tentando conectar ao servidor MQTT");
        #endif

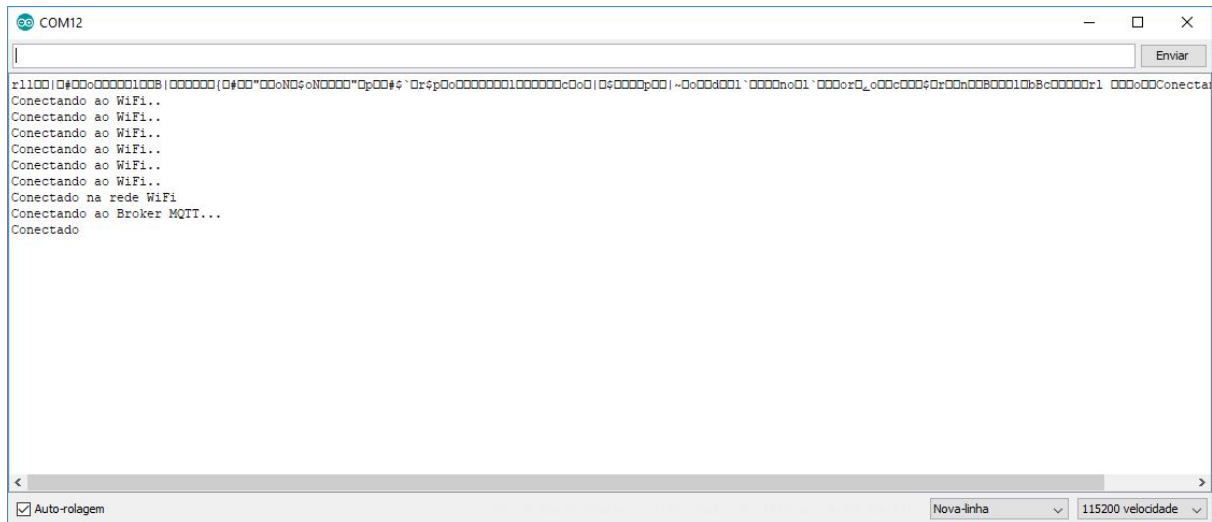
        bool conectado = strlen(mqttUser) > 0 ?
            client.connect("ESP8266Client", mqttUser, mqttPassword) :
            client.connect("ESP8266Client");

        if(conectado) {
            #ifdef DEBUG
            Serial.println("Conectado!");
            #endif
            //subcreve no tópico
            client.subscribe(mqttTopicSub, 1); //nivel de qualidade: QoS 1
        } else {
            #ifdef DEBUG
            Serial.println("Falha durante a conexão.Code: ");
            Serial.println( String(client.state()).c_str());
            Serial.println("Tentando novamente em 10 s");
            #endif
            //Aguarda 10 segundos
            delay(10000);
        }
    }
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}

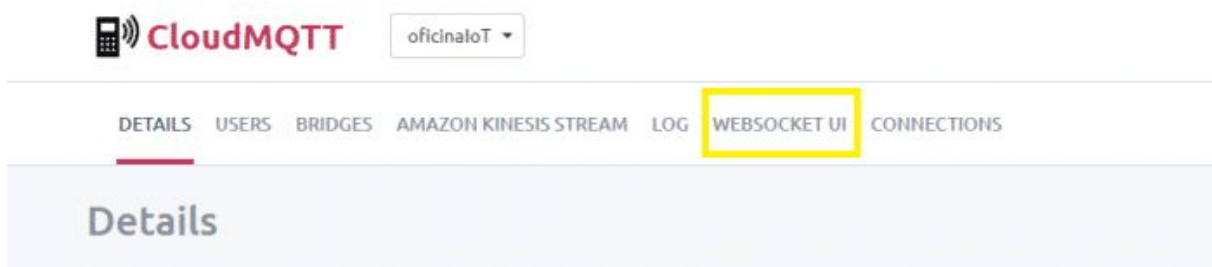
```

Abra o terminal e veja se a placa conectou a rede WIFI e ao broker:



## Teste de escrita no tópico com CloudMQTT

Abra a guia Websocket UI no CloudMQTT:



Envie uma mensagem no tópico casa/L1 pelo Websocket UI:

o - Saída em nível LOW

1 - Saída em nível HIGH

## Controle Dispositivos Remotamente com o ESP8266

### Websocket

#### Send message

Topic

Message

#### Received messages

Topic	Message
esp8266/pincmd	desliga
casa/L1	1
casa/L1	0
casa/L1	1
casa/L1	0
casa/L1	1

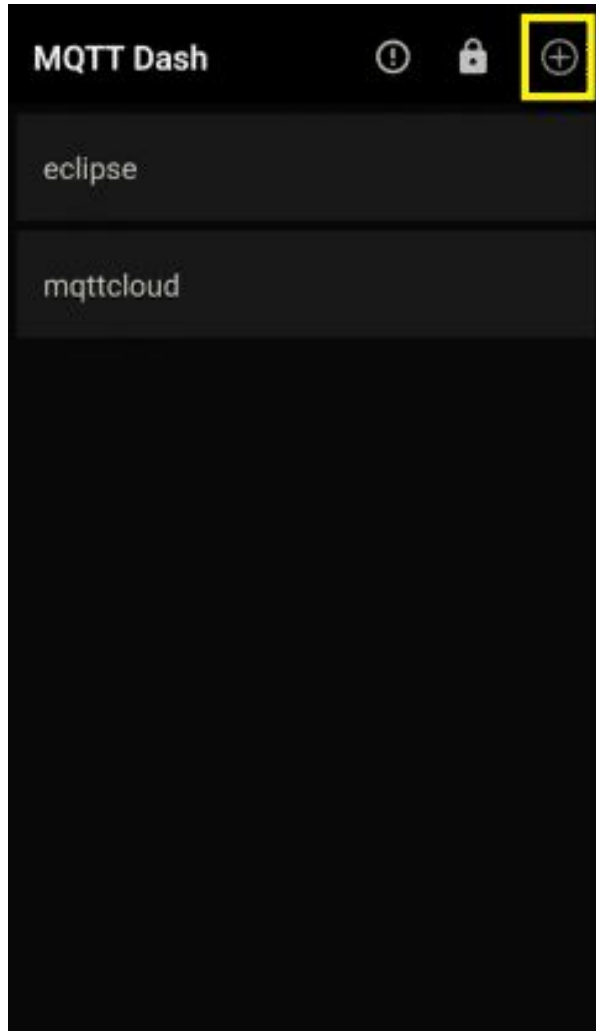
Verifique se o estado do saída mudo no aplicativo mudou.

Você também pode ver se a placa recebeu a mensagem através do terminal Serial:

```
COM12
f1100 | 0#000000000000 | 000000 | 0#00"000N0#0N0000"0p00#0'0r0p000000000000000000 | 0#0000p00 | ~0000001'0000m001'0000r0_c000c000#0r00m00800010b0c00000r1 000000Conecta
Conectando ao WiFi...
Conectando ao WiFi...
Conectando ao WiFi...
Conectando ao WiFi...
Conectando ao WiFi...
Conectando ao WiFi...
Conectado na rede WiFi
Conectando ao Broker MQTT...
Conectado
Mensagem chegou do tópico: casa/L1
Mensagem:1
-----
Mensagem chegou do tópico: casa/L1
Mensagem:0
-----
< [Auto-rolagem] Nova-linha 115200 velocidade
```

## Configurando a aplicação no MQTT Dash

Após instalado, clique no sinal “+” na sua tela inicial:



Será aberta a configuração de uma nova conexão. Insira as seguinte informações:

- Name
- Address
- Port
- User Name
- User Password



The image shows a mobile application interface titled "MQTT Dash". It has a dark theme. At the top right is a save icon (a floppy disk). Below the title, there are several input fields: "Name" with the value "casa", "Address" with the value "m13.cloudmqtt.com", and "Port" with the value "12857". Below these fields is a section for SSL/TLS configuration. It contains a paragraph of text explaining the options, followed by two checkboxes. The first checkbox is unchecked and is for enabling connection encryption. The second checkbox is also unchecked and is for trusting a self-signed certificate. Below the SSL/TLS section are fields for "User name" (value: "dreqdghr") and "User password" (masked with dots). At the bottom is a field for "Client ID (must be unique)".

**MQTT Dash**

Name  
casa

Address  
m13.cloudmqtt.com

Port  
12857

Enable connection encryption (SSL/TLS).  
Note: if server certificate is self-signed, you need to install it to your device or enable option below, otherwise connection will fail. If server certificate issued by a known Certificate Authority (CA), it will work out of box, without installing to you device. Also don't forget, that MQTT servers have different ports for plain and SSL/TLS connections.

☐ This broker uses self-signed SSL/TLS certificate. I trust this certificate at my own risk.

User name  
dreqdghr

User password  
.....

Client ID (must be unique)

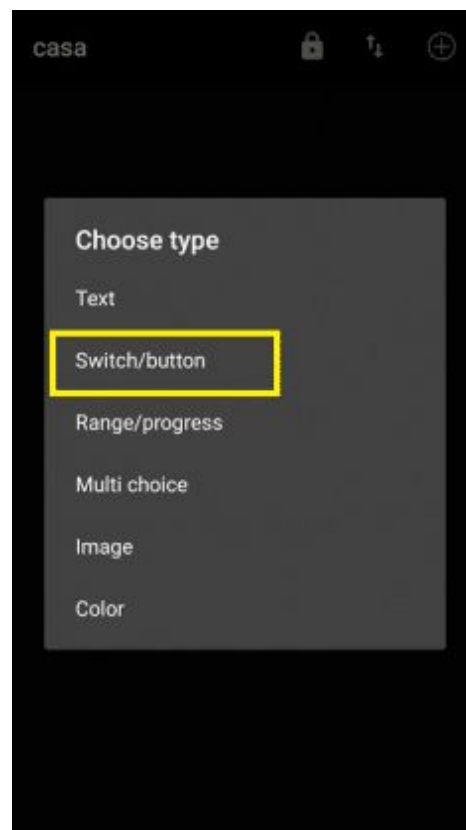
E, por fim, salve tocando no disquete na parte superior direita.

Abra a conexão criada. Caso as configurações estejam corretas não será exibida nenhuma mensagem. Caso contrário será exibida a mensagem de falha de conexão. Se isso acontecer, refaça as configurações.

Com a conexão configurada corretamente, clique no sinal de “+” dentro do dashboard criado:



Insira um Switch/button:





O botão será usado para acionamento da lâmpada. Você pode dar o nome L1 para ele, ou outro que achar melhor. Para o tópico, configure para “casa/L1”:



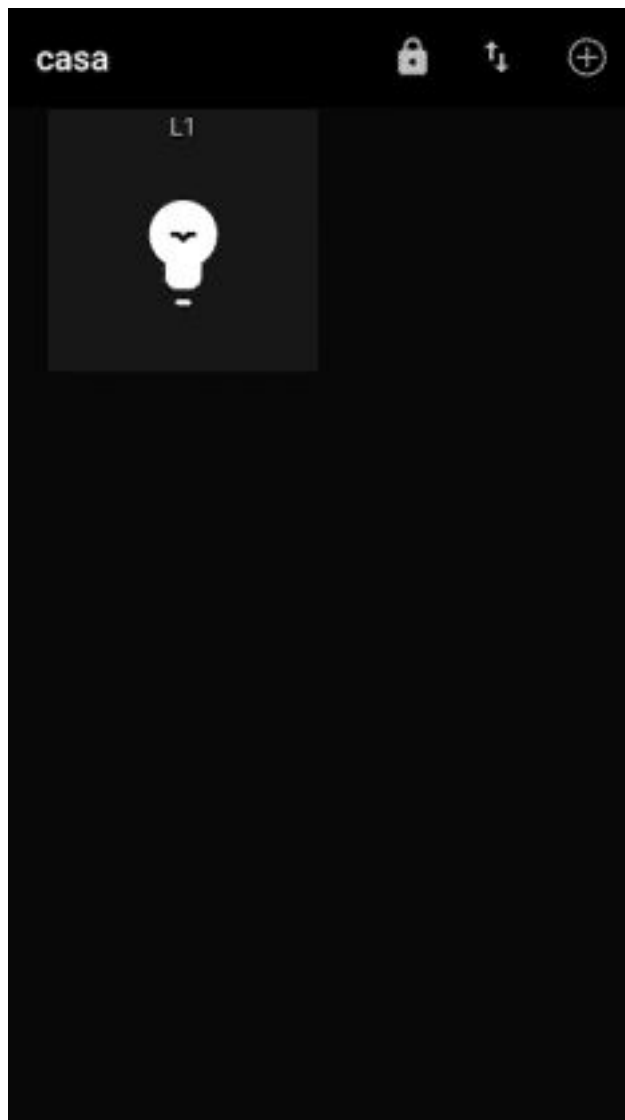
The image shows a mobile application interface titled "MQTT Dash". It contains a text box with instructions: "This metric is intended for state displaying and switching (e.g. light on/off). Or it can behave as a simple static button. Payload is expected to be string." Below this, there are two input fields. The first is labeled "Name" and contains the text "L1". The second is labeled "Topic (sub)" and contains the text "casa/L1". Below these fields is a link: "Extract from JSON path (if payload is in JSON format), e.g.: \$.level.value. JSON path documentation at the URL below: [https://github.com/jayway/JsonPath/blob/master/README.md](\"https://github.com/jayway/JsonPath/blob/master/README.md\")". There are two checkboxes, both of which are checked. The first checkbox is labeled "Enable publishing". The second checkbox is labeled "Update metric on publish immediately (do not wait for incoming message to update visual state)". At the bottom, there is a paragraph: "Payload and icons. If you need not a switch, but a simple button, just set the same payload values and the same icons for On and Off. This way the switch will never change icon and always send the same".

Para a parte visual desse botão, configure para exibir ícones de uma lâmpada acesa e outra apagada. Para On, enviaremos o valor “1” e para Off, o valor “0”:



Por fim, selecione o nível de qualidade para QoS(1).

Pronto, a aplicativo está conectado e configurado:




Abra a guia Websocket UI no CloudMQTT e veja se as mensagens estão chegando ao pressionar o botão:

### Websocket

#### Send message

Topic

Message



Send

#### Clear session

Clear any data for a client id

#### Received messages

▼

Topic	Message
casa/L1	0
casa/L1	1
casa/L1	1
casa/L1	0
casa/L1	0
casa/L1	1
casa/L1	0
casa/L1	1

Se placa estiver conectada saída também mudará o seu estado. Faça o teste!

Agora envie o comando via Websocket UI no CloudMQTT e veja o que acontece no aplicativo.

MQTT é demais!

## Desafio

Adicione mais saídas e outros tópicos ao seu projeto

## Referências

<https://www.embarcados.com.br/mqtt-dash/>

<https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html>

<https://www.embarcados.com.br/mqtt-protocolos-para-iot/>