INSTITUTO FEDERAL
São Paulo
Câmpus Hortolândia

ARDUINO DAY 2023

OFICINA DE INTERNET DAS COISAS COM NODEMCU E PROTOCOLO MQTT

Prof. Isaías Mendes de Oliveira
Prof. Rodolfo Francisco de Oliveira

IFSP – Campus Hortolândia
2023

INTERNET DAS COISAS

- A Internet das Coisas estuda formas de conectar as “coisas” à Internet.
  - As “coisas” podem ser muito variadas. Alguns exemplos incluem: lâmpadas, carros, casas, plantas, etc.
- Basicamente, as “coisas” podem conversar com outros sistemas computacionais (como a Internet) através de sensores e atuadores.
  - Sensores: coletam dados do meio ambiente (sensores de luz, presença, temperatura, etc)
  - Atuadores: atuam de alguma forma no meio ambiente (lampadas, motores, etc).
- Os sensores e atuadores, por sua vez, se comunicam com a Internet através de uma rede de comunicação com fio ou sem fio, como por exemplo Ethernet, Wi-Fi, LoRa, Bluetooth, etc.
- No entanto, os dados trafegados entre os sensores e atuadores e a Internet não podem ser encaminhados de qualquer maneira: eles precisam ser preparados e organizados. Quem realiza essa tarefa é um **protocolo de comunicação**, como o HTTP ou MQTT.

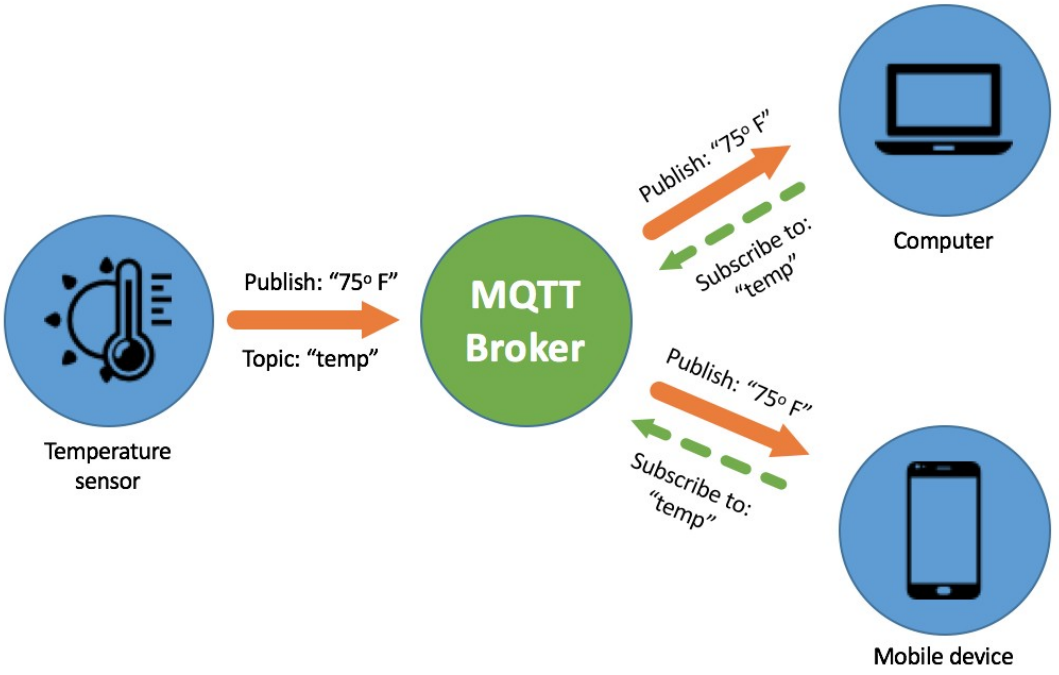
MQTT

- Significado: *Message Queue Telemetry Transport* (Transporte Telemétrico de Filas de Mensagens).
- Atualmente indicado para aplicações envolvendo Internet das Coisas. No entanto, foi desenvolvido pela IBM no início da década de 1990. Se tornou um protocolo aberto em 2014.
- É um protocolo assíncrono, ou seja, tanto emissor quanto receptor (“cliente e servidor”) não necessitam estar diretamente conectados entre si.
- **Internet das Coisas:** MQTT é indicado para aplicações relacionadas a Internet das Coisas pois se trata de um protocolo leve (baixo overhead), indicado para instalação em dispositivos com baixo poder computacional (como o Arduino e NodeMCU).

MQTT

- Para a transferência dos dados, o MQTT utiliza um paradigma denominado **Publisher / Subscriber** (Publicador / Assinante).
- Neste modelo, existe um único servidor (denominado *Broker*), e inúmeros clientes, que se dividem entre *publishers* e *subscribers.*
  - 1º Um cliente publicador se conecta ao servidor Broker, encaminhando dados (mensagens) para um determinado tópico.
  - 2º Um cliente assinante conectado ao broker pode assinar um ou mais tópicos;
  - 3º Quando uma mensagem chega a determinado tópico, o broker a encaminha a todos os assinantes conectados;

MQTT



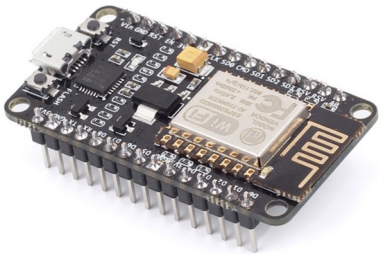
Exemplo de comunicação entre 3 dispositivos utilizando MQTT. Um determinado equipamento **publisher** denominado "**Temperature sensor"** publica o valor de uma temperatura (75ºF), em um tópico denominado "temp", no servidor Broker. Este envia o dado em questão para os **subscribers** "**Computer" e "Mobile device",** que estão assinando o tópico "temp".

BROKER MQTT

- Existem vários servidores brokers MQTT públicos e gratuitos disponíveis para serem utilizados na Internet.
- Também é possível instalar e configurar seu próprio servidor Broker. A solução mais conhecida (gratuita e de código aberto) é o **Mosquitto** (disponível para GNU/Linux e Windows).
- Exemplos de servidores brokers públicos e gratuitos:
  - https://test.mosquitto.org/
  - https://www.hivemq.com/downloads/
  - http://www.dioty.co/

Tais soluções devem ser utilizadas com cuidado. Nunca publique dados sensíveis em servidores brokers públicos. Além disso, os mesmos podem ficar indisponíveis sem prévio aviso, já que se trata de soluções gratuitas.

NODEMCU

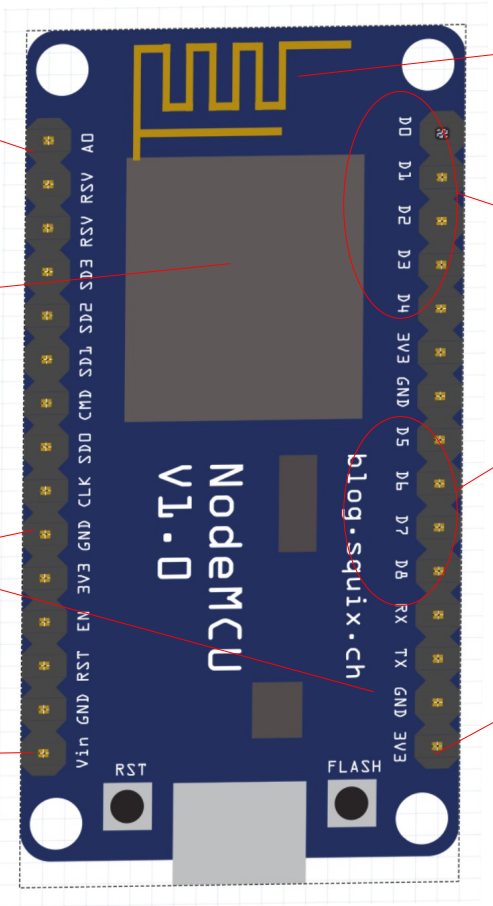


Especificações:

- CPU: ESP8266
- Armazenamento: 4 MB
- Memoria RAM: 128 Kbytes
- Wi-Fi 802.11 b / g / n
- Tensão de operação: 3,3v

- NodeMCU é um *hardware* de código aberto voltado para projetos relacionados a Internet das Coisas, que possui como principal diferencial um módulo Wi-Fi (módulo ESP8266) embutido, permitindo implementar comunicação sem fio com facilidade a um baixo custo.
- É possível programar o NodeMCU através da Arduino IDE, instalando as devidas bibliotecas.

NODEMCU



Pino analógico

Módulo ESP8266

Pinos Terra

Tensão de 5v

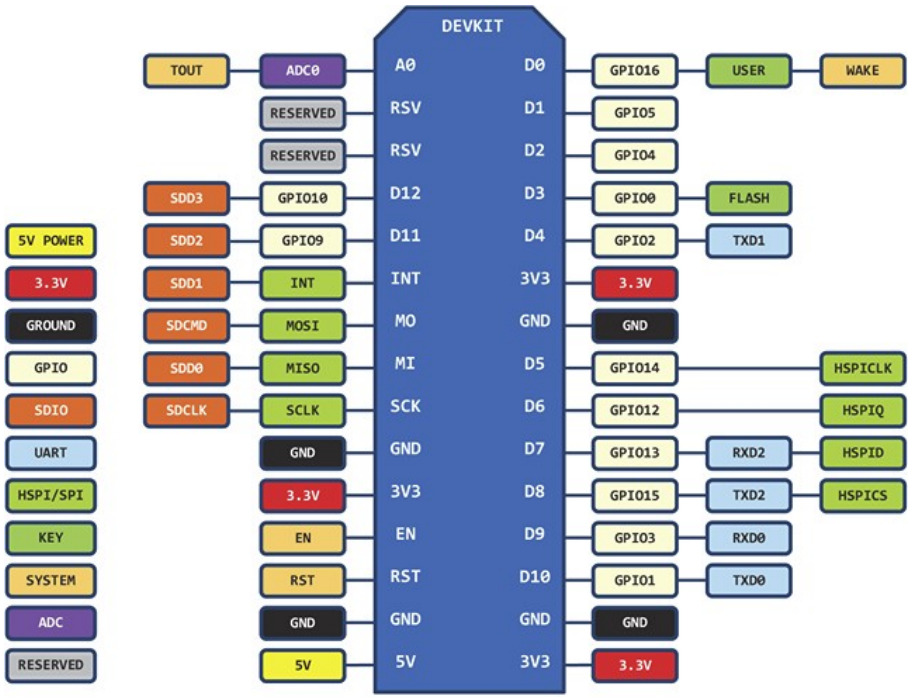
Antena

Pinos digitais

Tensão de 3,3v

NODEMCU

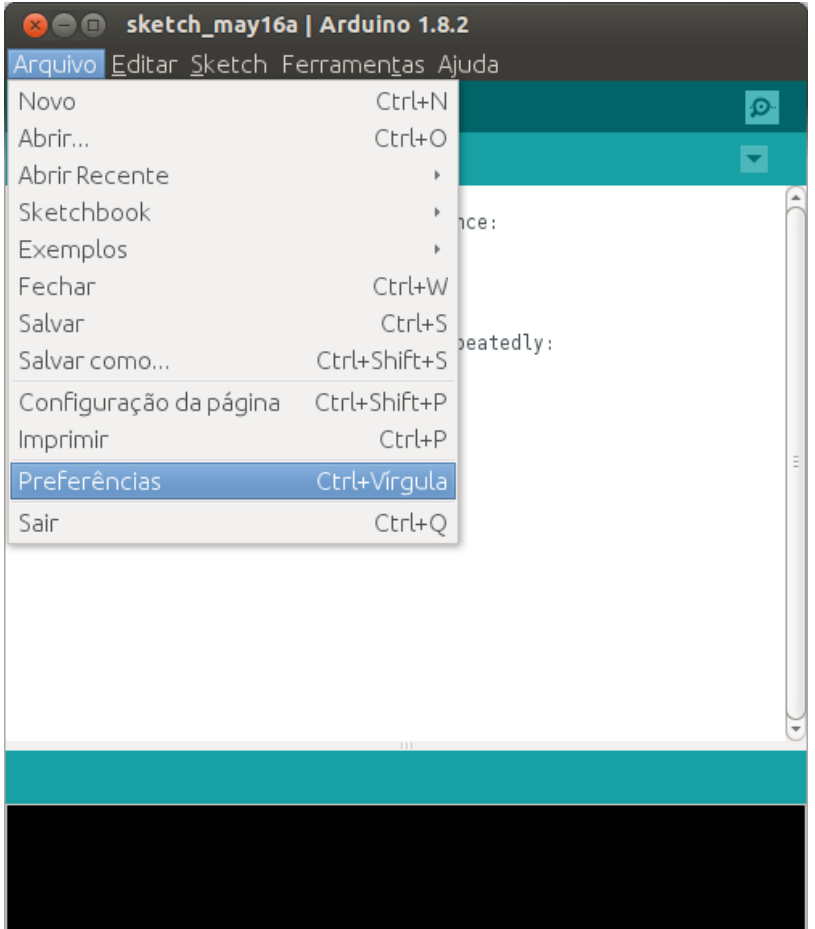
- No Arduino IDE, a referência aos pinos do NodeMCU é feita conforme ilustra a figura abaixo:



D0(GPIO16) can only be used as gpio read/write, no interrupt supported, no pwm/i2c/ow supported.

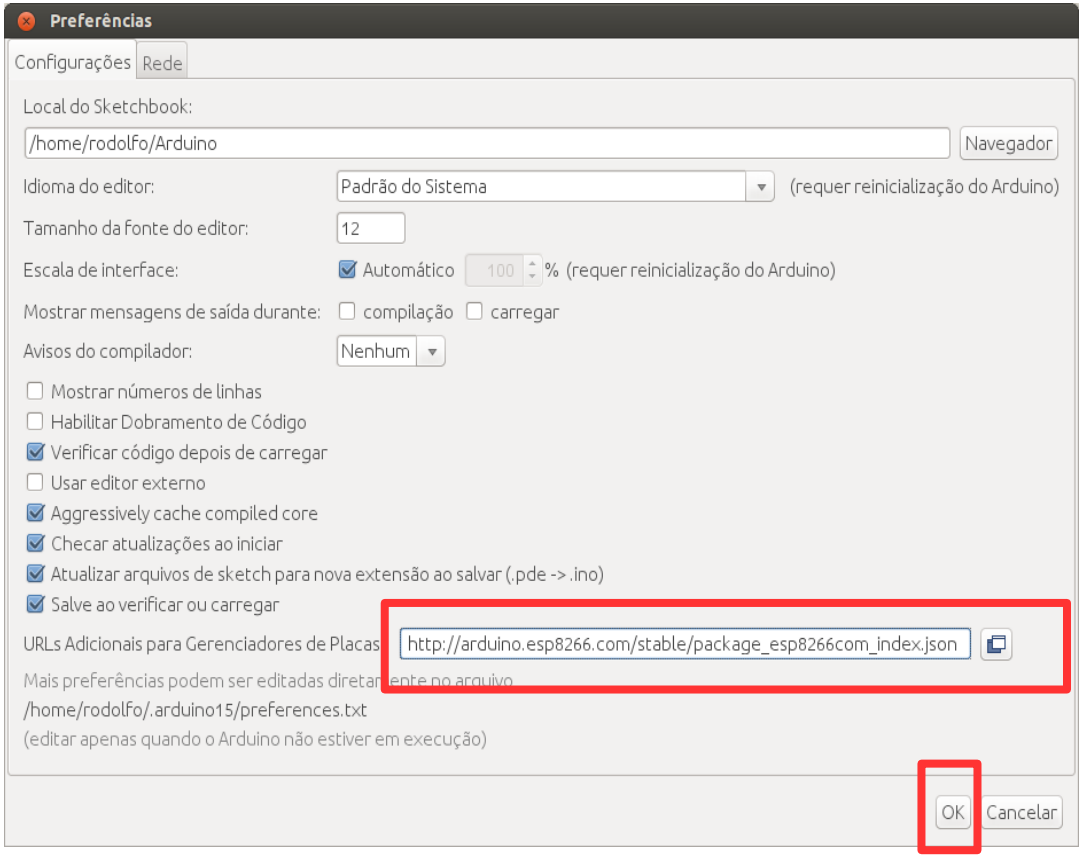
NODEMCU – INSTALAÇÃO DAS BIBLIOTECAS

- Para programar o NodeMCU no Arduino IDE, é necessário instalar as bibliotecas do mesmo. Para tal, siga os seguintes procedimentos:
- 1º Clique no menu **Arquivos** e escolha a opção **Preferências.**



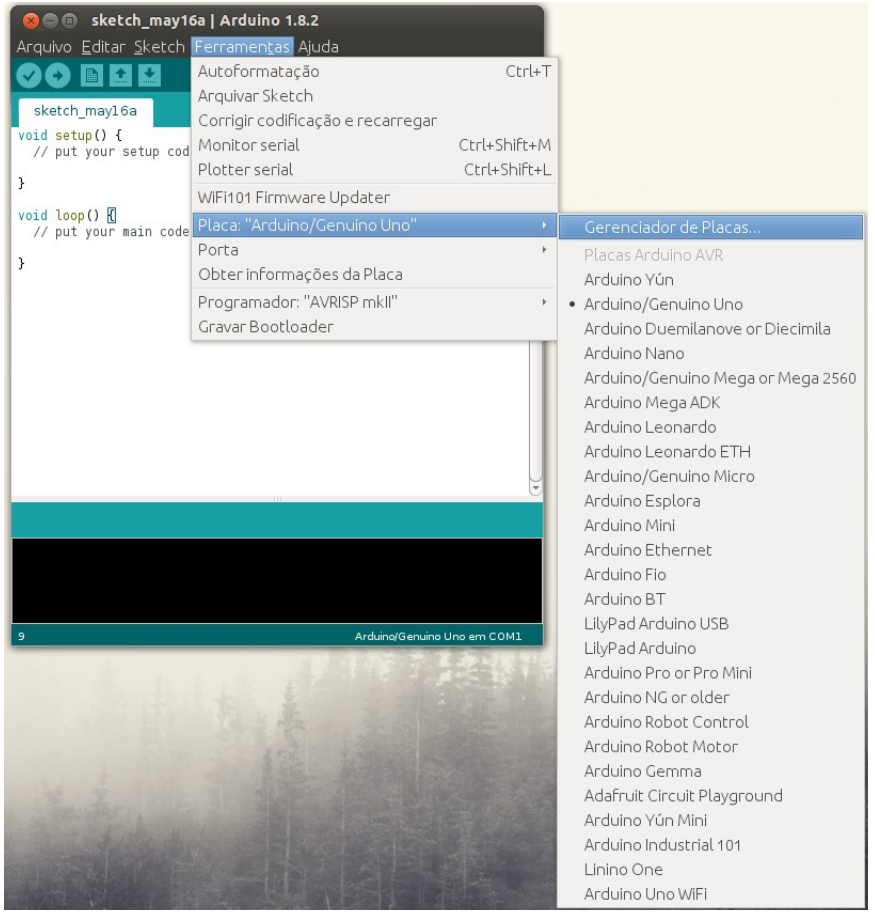
NODEMCU – INSTALAÇÃO DAS BIBLIOTECAS

- 2º No campo “URL para Gerenciadores de Placas”, digite “ http://arduino.esp8266.com/stable/package_esp8266com_index.json ”. Ao final, clique em “OK”



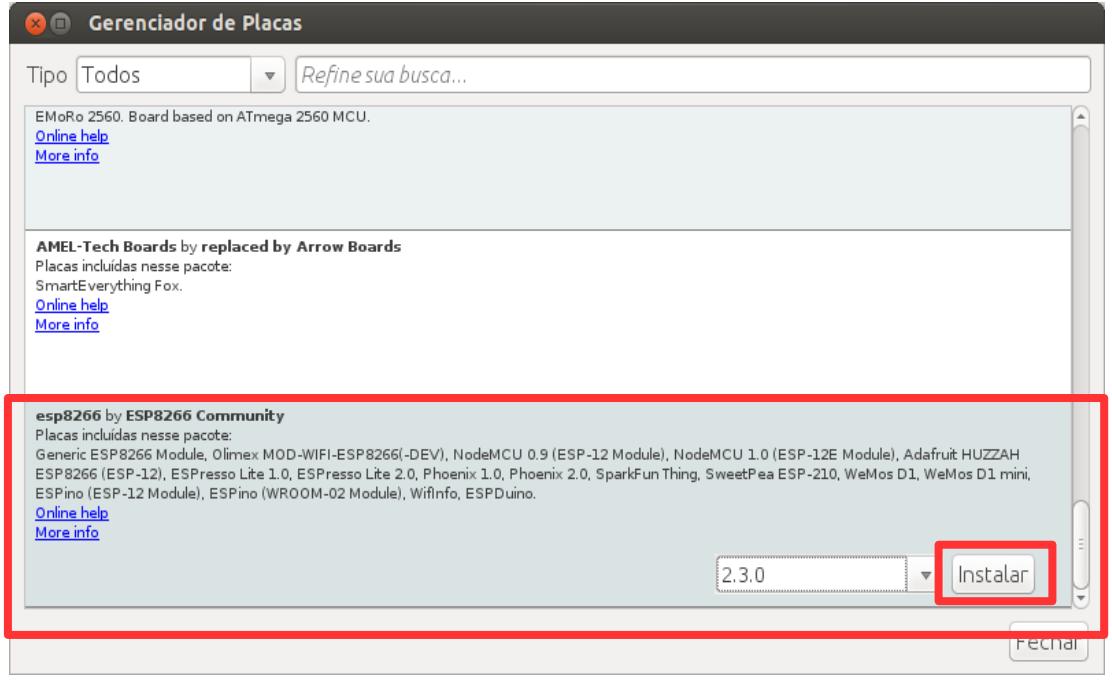
NODEMCU – INSTALAÇÃO DAS BIBLIOTECAS

- 3º No menu **Ferramentas,** clique em “**Placa: Arduino Uno”** e escolha **Gerenciador de Placas**.



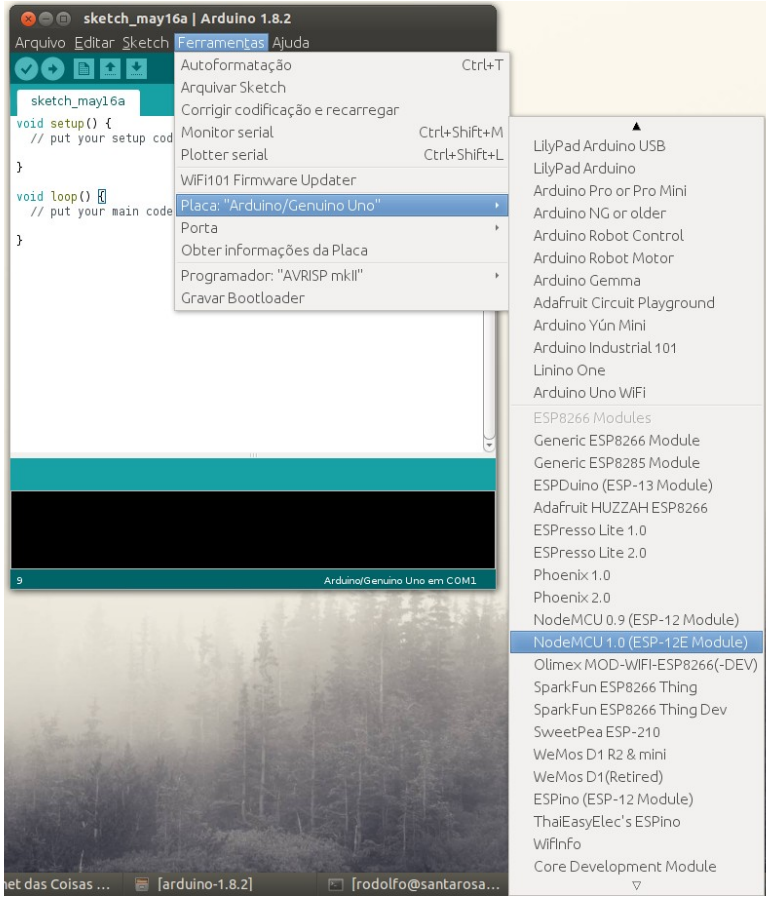
NODEMCU – INSTALAÇÃO DAS BIBLIOTECAS

- 4º Na janela que aparece, digite “esp8266 by ESP8266 Community” e clique em **Instalar**.



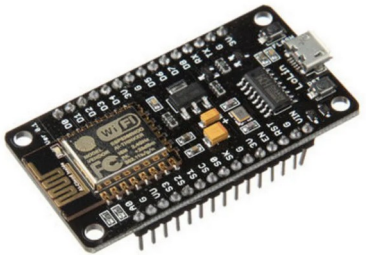
NODEMCU – INSTALAÇÃO DAS BIBLIOTECAS

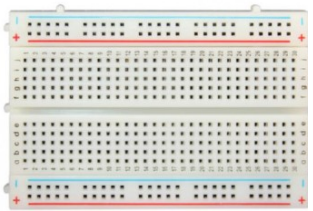
- 5º Após a instalação, será possível selecionar a placa NodeMCU através do menu **Ferramentas**, **Placas**.



LABORATÓRIO 01

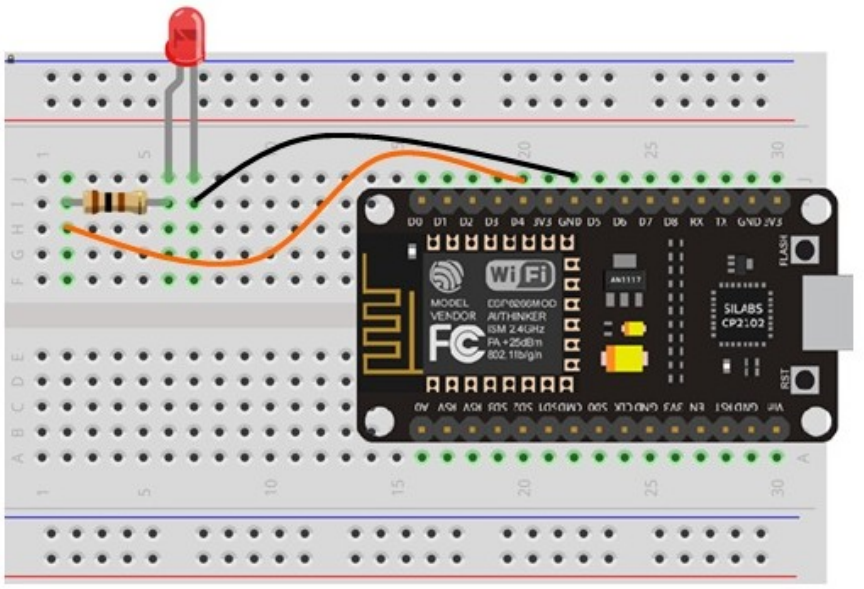
- **Laboratório 02 –** Led piscando no NodeMCU
- **Objetivo:** Conectar um Led externo ao NodeMCU e fazê-lo piscar.
- **Pré requisito:** Ter realizado o Laboratório 01.
  - **Equipamentos necessários:**
    - 01 NodeMCU
    - 01 Cabo USB – microUSB
    - 01 *Protoboard*
    - 01 LED (qualquer cor)
    - 01 Resistor 220 ohms
    - 02 Jumpers macho x fêmea
    - 01 Jumper macho x macho






LABORATÓRIO 01

- 1º Crie o seguinte circuito utilizando os componentes apresentados. Repare que o *Led* está conectado no pino “D4” do Arduino:



LABORATÓRIO 01

- 2º Conecte o cabo USB ao NodeMCU e ao computador.
- 3º Aba o Arduino IDE. Certifique-se de que a porta USB e o tipo de placa (menu Ferramentas) esteja selecionado de maneira correta.

LABORATÓRIO 01

- 4º Faça *upload* do seguinte código (*sketch).* Ao final, o *led* deverá começar a piscar:

```
void setup() {
 //Pino 2 (o pino D4 no NodeMCU) está sendo configurado como um pino de saída
 pinMode(2,OUTPUT);

}

void loop() {

  //Envia um sinal "High" ao pino D4, acendendo o LED
  digitalWrite(2,HIGH);
  delay(1000);
  //Envia um sinal "Low" ao pino D4, apagando o LED
  digitalWrite(2,LOW);
  delay(1000);

}
```

LABORATÓRIO 01

- 4º Faça *upload* do seguinte código (*sketch).* Ao final, o *led* deverá começar a piscar **(Explicação)**:

```
void setup() {
 //Pino 2 (o pino D4 no NodeMCU) está sendo configurado como um pino de saída
 pinMode(2,OUTPUT);

}

void loop() {

  //Envia um sinal "High" ao pino D4, acendendo o LED
  digitalWrite(2,HIGH);
  delay(1000);
  //Envia um sinal "Low" ao pino D4, apagando o LED
  digitalWrite(2,LOW);
  delay(1000);

}
```

Define o pino 2 (D4 do NodeMCU) como pino de saída)

LABORATÓRIO 01

- 4º Faça *upload* do seguinte código (*sketch).* Ao final, o *led* deverá começar a piscar **(Explicação):**

```
void setup() {
 //Pino 2 (o pino D4 no NodeMCU) está sendo configurado como um pino de saída
 pinMode(2,OUTPUT);

}

void loop() {

  //Envia um sinal "High" ao pino D4, acendendo o LED
  digitalWrite(2,HIGH);
  delay(1000);
  //Envia um sinal "Low" ao pino D4, apagando o LED
  digitalWrite(2,LOW);
  delay(1000);

}
```

Escreve HIGH (“ligado”) no pino 2 (D4), acendendo o mesmo.

LABORATÓRIO 01

- 4º Faça *upload* do seguinte código (*sketch).* Ao final, o *led* deverá começar a piscar **(Explicação)**:

```
void setup() {
 //Pino 2 (o pino D4 no NodeMCU) está sendo configurado como um pino de saída
 pinMode(2,OUTPUT);

}

void loop() {

  //Envia um sinal "High" ao pino D4, acendendo o LED
  digitalWrite(2,HIGH);
  delay(1000);
  //Envia um sinal "Low" ao pino D4, apagando o LED
  digitalWrite(2,LOW);
  delay(1000);

}
```

Aguarda 1 segundo

LABORATÓRIO 01

- 4º Faça *upload* do seguinte código (*sketch).* Ao final, o *led* deverá começar a piscar **(Explicação)**:

```
void setup() {
 //Pino 2 (o pino D4 no NodeMCU) está sendo configurado como um pino de saída
 pinMode(2,OUTPUT);

}

void loop() {

  //Envia um sinal "High" ao pino D4, acendendo o LED
  digitalWrite(2,HIGH);
  delay(1000);
  //Envia um sinal "Low" ao pino D4, apagando o LED
  digitalWrite(2,LOW);
  delay(1000);

}
```

Escreve LOW (“deligado”) no pino 2 (D4), desligando o mesmo.

LABORATÓRIO 01

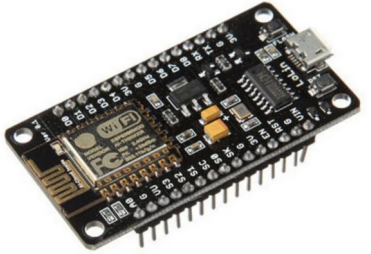
- 4º Faça *upload* do seguinte código (*sketch).* Ao final, o *led* deverá começar a piscar **(Explicação)**:

```
void setup() {
 //Pino 2 (o pino D4 no NodeMCU) está sendo configurado como um pino de saída
 pinMode(2,OUTPUT);

}

void loop() {

  //Envia um sinal "High" ao pino D4, acendendo o LED
  digitalWrite(2,HIGH);
  delay(1000);
  //Envia um sinal "Low" ao pino D4, apagando o LED
  digitalWrite(2,LOW);
  delay(1000);

}
```

Aguarda 1 segundo

LABORATÓRIO 02

- **Laboratório 02 –** Conectando em uma rede Wi-Fi
- **Objetivo:** Conectar o NodeMCU em uma rede Wi-Fi.
- **Pré requisito:** nenhum.
  - **Equipamentos necessários:**
    - 01 NodeMCU
    - 01 Cabo USB – microUSB



LABORATÓRIO 02

- 1º Conecte o cabo USB ao NodeMCU e ao computador.
- 2º Aba o Arduino IDE. Certifique-se de que a porta USB e o tipo de placa (menu Ferramentas) esteja selecionado de maneira correta.

LABORATÓRIO 02

- 3º Faça *upload* do seguinte código (*sketch)*:

```
#include <ESP8266WiFi.h>

//SSID e senha da rede Wi-Fi
const char* SSID = "SSID";
const char* PASS = "password";

void setup() {

    //Inicia a serial
    Serial.begin(9600);

    //Conecta a rede WiFi utilizando o modo Access Point
    WiFi.mode(WIFI_AP_STA);
    WiFi.begin(SSID,PASS);

     //Aguarda conexão WiFi
    while (WiFi.status() != WL_CONNECTED)
      {
        delay(300);
        Serial.print(".");
      }

    //Informa qual é o endereço IP obtido no DHCP
    Serial.print("Endereço IP obtido: ");
    Serial.println(WiFi.localIP());


}

void loop() {
  // put your main code here, to run repeatedly:

}
```

LABORATÓRIO 02

- 4º Acesse o Monitor Serial. Caso o NodeMCU consiga conexão, ele irá imprimir o endereço IP obtido na conexão.
- 5º Em qualquer computador da rede, realize um teste de comunicação com o NodeMCU através do comando **ping**, conforme exemplo abaixo:

```
rodolfo@epoch:~$ ping 192.168.0.14
PING 192.168.0.14 (192.168.0.14) 56(84) bytes of data.
64 bytes de 192.168.0.14: icmp_seq=1 ttl=64 tempo=0.053 ms
64 bytes de 192.168.0.14: icmp_seq=2 ttl=64 tempo=0.049 ms
64 bytes de 192.168.0.14: icmp_seq=3 ttl=64 tempo=0.049 ms
64 bytes de 192.168.0.14: icmp_seq=4 ttl=64 tempo=0.045 ms
```

LABORATÓRIO 02

- Explicação do código:

```
#include <ESP8266WiFi.h>

//SSID e senha da rede Wi-Fi
const char* SSID = "SSID";
const char* PASS = "password";

void setup() {

    //Inicia a serial
    Serial.begin(9600);

    //Conecta a rede WiFi utilizando o modo Access Point
    WiFi.mode(WIFI_AP_STA);
    WiFi.begin(SSID,PASS);

     //Aguarda conexão WiFi
    while (WiFi.status() != WL_CONNECTED)
      {
        delay(300);
        Serial.print(".");
      }

    //Informa qual é o endereço IP obtido no DHCP
    Serial.print("Endereço IP obtido: ");
    Serial.println(WiFi.localIP());


}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Incluí a biblioteca “ESP8266WiFi.h”, responsável pelos métodos de conexão em redes Wi-Fi.

Declara duas constantes do tipo “char” (SSID e PASS), que irão armazenar, respectivamente, o nome da rede Wi-Fi (SSID) e a senha de acesso.

LABORATÓRIO 02

- Explicação do código:

```
#include <ESP8266WiFi.h>

//SSID e senha da rede Wi-Fi
const char* SSID = "SSID";
const char* PASS = "password";

void setup() {

    //Inicia a serial
    Serial.begin(9600);

    //Conecta a rede WiFi utilizando o modo Access Point
    WiFi.mode(WIFI_AP_STA);
    WiFi.begin(SSID,PASS);

     //Aguarda conexão WiFi
    while (WiFi.status() != WL_CONNECTED)
      {
        delay(300);
        Serial.print(".");
      }

    //Informa qual é o endereço IP obtido no DHCP
    Serial.print("Endereço IP obtido: ");
    Serial.println(WiFi.localIP());


}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Inicia a conexão com a serial, a fim de escrever o estado da conexão Wi-Fi na mesma.

Define o modo de conexão Wi-Fi como “WIFI_AP_STA”. Neste modo, o NodeMCU irá procurar um Access Point (ou Roteador) para se conectar.

LABORATÓRIO 02

- Explicação do código:

```
#include <ESP8266WiFi.h>

//SSID e senha da rede Wi-Fi
const char* SSID = "SSID";
const char* PASS = "password";

void setup() {

    //Inicia a serial
    Serial.begin(9600);

    //Conecta a rede WiFi utilizando o modo Access Point
    WiFi.mode(WIFI_AP_STA);
    WiFi.begin(SSID,PASS);

     //Aguarda conexão WiFi
    while (WiFi.status() != WL_CONNECTED)
      {
        delay(300);
        Serial.print(".");
      }

    //Informa qual é o endereço IP obtido no DHCP
    Serial.print("Endereço IP obtido: ");
    Serial.println(WiFi.localIP());


}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Inicia a conexão Wi-Fi, com o SSID e senha informados.

Enquanto o estados da conexão for diferente de “WL_CONNECTED” (ou seja, enquanto a conexão não estiver estabelecida), será impresso um ponto (“.”) a cada 300 milissegundos na serial.

LABORATÓRIO 02

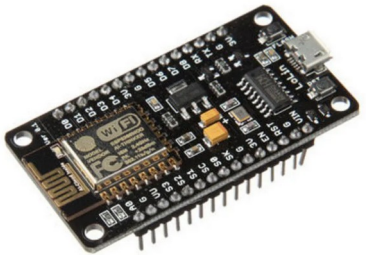
- Explicação do código:

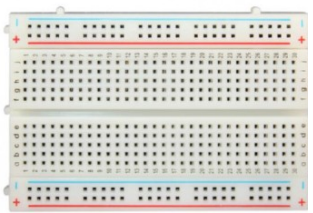
```
#include <ESP8266WiFi.h>

//SSID e senha da rede Wi-Fi
const char* SSID = "SSID";
const char* PASS = "password";

void setup() {

    //Inicia a serial
    Serial.begin(9600);

    //Conecta a rede WiFi utilizando o modo Access Point
    WiFi.mode(WIFI_AP_STA);
    WiFi.begin(SSID,PASS);

     //Aguarda conexão WiFi
    while (WiFi.status() != WL_CONNECTED)
      {
        delay(300);
        Serial.print(".");
      }

    //Informa qual é o endereço IP obtido no DHCP
    Serial.print("Endereço IP obtido: ");
    Serial.println(WiFi.localIP());


}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Quando a conexão for efetivada, o endereço IP obtido será impresso, através do método “WiFi.localIP()”

LABORATÓRIO 03

- **Laboratório 03 –** Utilizando MQTT para transporte de dados (*Subscriber*)
- **Objetivo:** Acender e apagar um led conectado ao NodeMCU através de mensagens enviadas por um aplicativo no celular. A comunicação será mediada pelo protocolo MQTT, onde o NodeMCU atuará como *subscriber* de um determinado tópico.
- **Pré requisito:** Ter realizado o Laboratório 01 e 02.
  - **Equipamentos necessários:**
    - 01 NodeMCU
    - 01 Cabo USB – microUSB
    - 01 *Protoboard*
    - 01 LED (qualquer cor)
    - 01 Resistor 220 ohms
    - 02 Jumpers macho x fêmea
    - 01 Jumper macho x macho
    - 01 Smartphone com sistema operacional Android

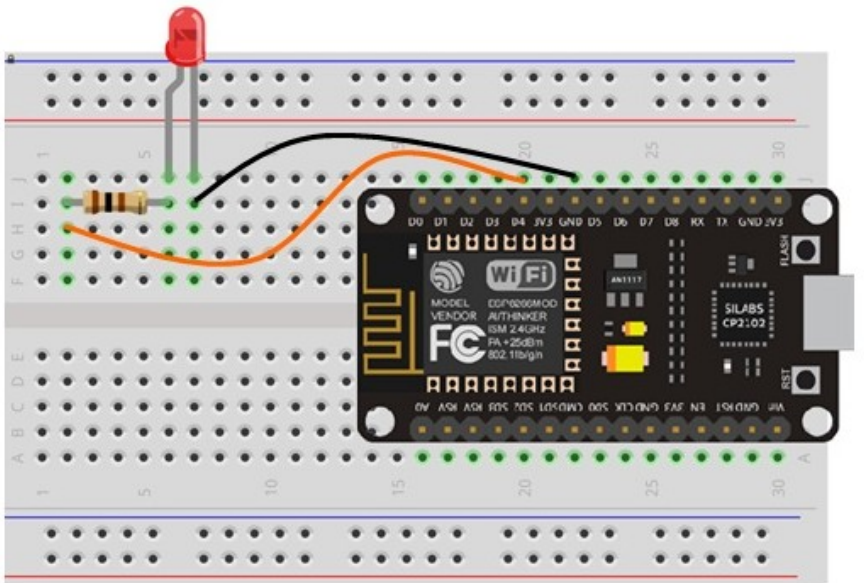






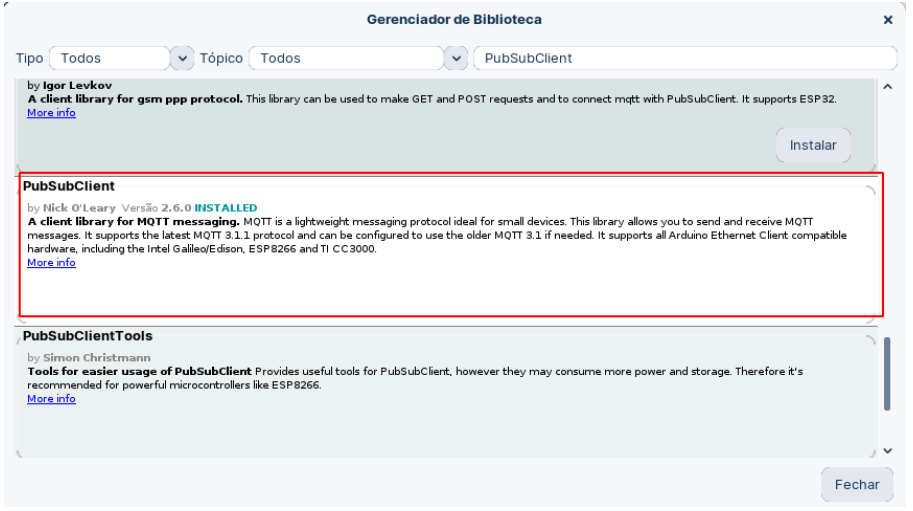
LABORATÓRIO 03

- 1º Crie o seguinte circuito utilizando os componentes apresentados (mesmo circuito do Laboratório 08):



LABORATÓRIO 03

- 2º Conecte o cabo USB ao NodeMCU e ao computador.
- 3º Aba o Arduino IDE. Certifique-se de que a porta USB e o tipo de placa (menu Ferramentas) esteja selecionado de maneira correta.
- 4º Através do Gerenciador de Bibliotecas (Menu Ferramentas – Gerenciar Bibliotecas), instale a biblioteca **PubSubClient**.



LABORATÓRIO 03

- 5º Faça *upload* do seguinte código. Ele irá fazer com que o NodeMCU atue como subscriber, permitindo receber comandos para acender e apagar o led (Parte 1 de 5):

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

const char* SSID = "SSID";
const char* PASS = "senha";

const char* enderecoBroker = "test.mosquitto.org";
const char* topico = "10495285/ledCasa";

WiFiClient espClient;
PubSubClient client(espClient);
```

LABORATÓRIO 03

- 5º Faça *upload* do seguinte código. Ele irá fazer com que o NodeMCU atue como subscriber, permitindo receber comandos para acender e apagar o led (Parte 2 de 5):

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

  Serial.print("Mensagem recebida [");
  Serial.print(topic);
  Serial.print("]:");

  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
   }

  Serial.println();

  //Se a mensagem recebida for "1", o LED será ligado
  if ((char)payload[0] == '1') {
    digitalWrite(2, HIGH);

  //Caso contrário, se a mensagem recebida for "0", o led será desligado.
  }
  else if ((char)payload[0] == '0') {
    digitalWrite(2, LOW);  // Turn the LED off by making the voltage HIGH
   }

}
```

LABORATÓRIO 03

- 5º Faça *upload* do seguinte código. Ele irá fazer com que o NodeMCU atue como subscriber, permitindo receber comandos para acender e apagar o led (Parte 3 de 5):

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Tentando se conectar ao Broker MQTT - ");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c_str())) {

      Serial.println("Conectado!");

      client.subscribe(topico);
    }

    else {
      Serial.print("Falha, rc=");
      Serial.print(client.state());
      Serial.println(" tentando novamente em 5 segundos");
      delay(5000);
    }
  }
}
```

LABORATÓRIO 03

- 5º Faça *upload* do seguinte código. Ele irá fazer com que o NodeMCU atue como subscriber, permitindo receber comandos para acender e apagar o led (Parte 4 de 5):

```
void setup() {
  pinMode(2, OUTPUT);

  Serial.begin(9600);

  WiFi.mode(WIFI_AP_STA);
  WiFi.begin(SSID,PASS);
  while (WiFi.status() != WL_CONNECTED)
   {
    delay(300);
    Serial.print(".");
   }
  Serial.print("Endereço IP obtido: ");
  Serial.println(WiFi.localIP());

  //Define qual é o servidor broker, bem como sua porta
  client.setServer(enderecoBroker, 1883);

  //Define qual é a função a ser executada quando uma mensagem é executada
  client.setCallback(callback);
}
```

LABORATÓRIO 03

- 5º Faça *upload* do seguinte código. Ele irá fazer com que o NodeMCU atue como subscriber, permitindo receber comandos para acender e apagar o led (Parte 5 de 5):

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

LABORATÓRIO 03

- 5º Faça *upload* do seguinte código. Ele irá fazer com que o NodeMCU atue como subscriber, permitindo receber comandos para acender e apagar o led (Parte 1 de 5 - **Explicação**):

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

const char* SSID = "SSID";
const char* PASS = "senha";

const char* enderecoBroker = "test.mosquitto.org";
const char* topico = "10495285/ledCasa";

WiFiClient espClient;
PubSubClient client(espClient);
```

Inclui as bibliotecas necessárias para conectar na rede WiFi e trabalhar com o protocolo MQTT.

Define o SSID e senha da rede Wi-FI (conforme visto no Laboratório 09)

Define qual será o endereço do servidor Broker e o nome do tópico que o NodeMCU irá assinar. No exemplo, estamos utilizando o broker “test.mosquitto.org” e o nome de tópico “10495285/ledCasa”. Teoricamente, o nome do tópico pode ser qualquer um, desde que seja único no servidor Broker.

Cria um objeto denominado **client** da classe PubSubClient. Este objeto será responsável por conectar no servidor Broker. Tal objeto depende de um objeto da classe WiFiClient, aqui denominado **espClient**.

LABORATÓRIO 03

- 5º Faça *upload* do seguinte código. Ele irá fazer com que o NodeMCU atue como subscriber, permitindo receber comandos para acender e apagar o led (Parte 2 de 5 - **Explicação**):

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

  Serial.print("Mensagem recebida [");
  Serial.print(topic);
  Serial.print("]:");

  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
   }

  Serial.println();

  //Se a mensagem recebida for "1", o LED será ligado
  if ((char)payload[0] == '1') {
    digitalWrite(2, HIGH);

  //Caso contrário, se a mensagem recebida for "0", o led será desligado.
  }
  else if ((char)payload[0] == '0') {
    digitalWrite(2, LOW);  // Turn the LED off by making the voltage HIGH
   }

}
```

Define uma função denominada **callback**. Esta função é importante, pois será executada toda vez que uma mensagem chegar ao NodeMCU.

Este trecho de código simplesmente imprime na serial do computador a mensagem recebida. Não é um trecho necessário para o funcionamento da aplicação, mas importante para o teste da mesma, sobretudo para checar se a mensagem está de fato chegando ao NodeMCU.

Verifica se a mensagem (denominada de **payload**) contém o valor “1” em seu primeiro caractere ([0]). Caso seja, acende o Led conectado ao pino D4 (2 no Arduino). Caso contrário, se for igual a “0”, apaga o mesmo Led.

LABORATÓRIO 03

- 5º Faça *upload* do seguinte código. Ele irá fazer com que o NodeMCU atue como subscriber, permitindo receber comandos para acender e apagar o led (Parte 3 de 5 - **Explicação**):

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Tentando se conectar ao Broker MQTT - ");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c_str())) {

      Serial.println("Conectado!");

      client.subscribe(topico);
    }

    else {
      Serial.print("Falha, rc=");
      Serial.print(client.state());
      Serial.println(" tentando novamente em 5 segundos");
      delay(5000);
    }
  }
}
```

Define uma função denominada **reconnect**. Esta função será executada caso a conexão com o *broker* seja perdida. Seu objetivo é restabelecer a conexão com o Broker.

Enquanto a conexão não for estabelecida (! client.connected(), a mesma tentará ser realizada (client.connect(clientID.c_str()). O cliente é identificado na conexão através da variável **clientID**. Ao se estabelecer a conexão, o cliente assina (cliente.subscribe) identificado na variável tópico**.**

Caso a conexão não seja estabelecida, o código da falha é impresso na Serial. Aguarda-se 5 segundos para uma nova tentativa de conexão.

LABORATÓRIO 03

- 5º Faça *upload* do seguinte código. Ele irá fazer com que o NodeMCU atue como subscriber, permitindo receber comandos para acender e apagar o led (Parte 4 de 5 - **Explicação**):

```
void setup() {
  pinMode(2, OUTPUT);

  Serial.begin(9600);

  WiFi.mode(WIFI_AP_STA);
  WiFi.begin(SSID,PASS);
  while (WiFi.status() != WL_CONNECTED)
   {
    delay(300);
    Serial.print(".");
   }
  Serial.print("Endereço IP obtido: ");
  Serial.println(WiFi.localIP());

  //Define qual é o servidor broker, bem como sua porta
  client.setServer(enderecoBroker, 1883);

  //Define qual é a função a ser executada quando uma mensagem é executada
  client.setCallback(callback);
}
```

Define o pino 2 como saída, inicia a serial e a conexão com a rede WiFi, conforme laboratórios anteriores.

Define qual será o servidor broker a ser utilizado, e também sua porta (1883). A porta é um identificador único para servidores em um determinado equipamento.

Define qual será a função a ser executada quando uma mensagem chega. No nosso exemplo, a função se denominada **callback**).

LABORATÓRIO 03

- 5º Faça *upload* do seguinte código. Ele irá fazer com que o NodeMCU atue como subscriber, permitindo receber comandos para acender e apagar o led (Parte 5 de 5 - **Explicação**):

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

Caso o cliente não esteja conectado, a função **reconnect()** é acionada. É possível afirmar com certeza que esta função será executada ao menos uma vez (na 1º execução do código, para se estabelecer a conexão inicial com o servidor. A partir deste momento, a mesma só será executada quando houver perda de conexão.

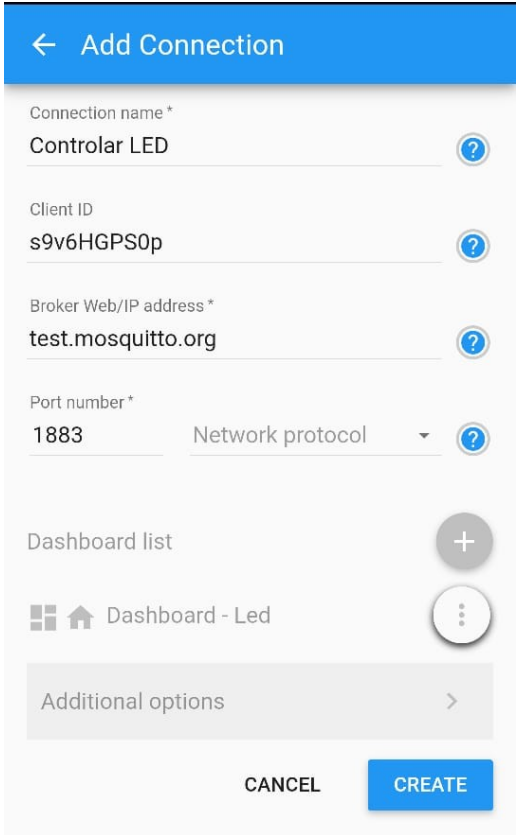
Caso a conexão esteja ativa, será executado o método **loop()**, onde o NodeMCU irá "escutar" as mensagens recebidas. Ao recebê-las, a função **callback** é executada.

LABORATÓRIO 03

- 6º Agora iremos criar um **publisher** (publicador) para enviar os comandos para o NodeMCU (acender e desligar o LED). Para tal, iremos utilizar nosso *smartphone* (sistema operacional Android).
- 7º Instale o aplicativo **IoT MQTT Panel** a partir da Google Play. Ao final, abra o aplicativo.
- 8º Na tela inicial do mesmo, escolha a opção **Setup a Connection** (“Configurar a conexão”), para que possamos conectar ao Broker.

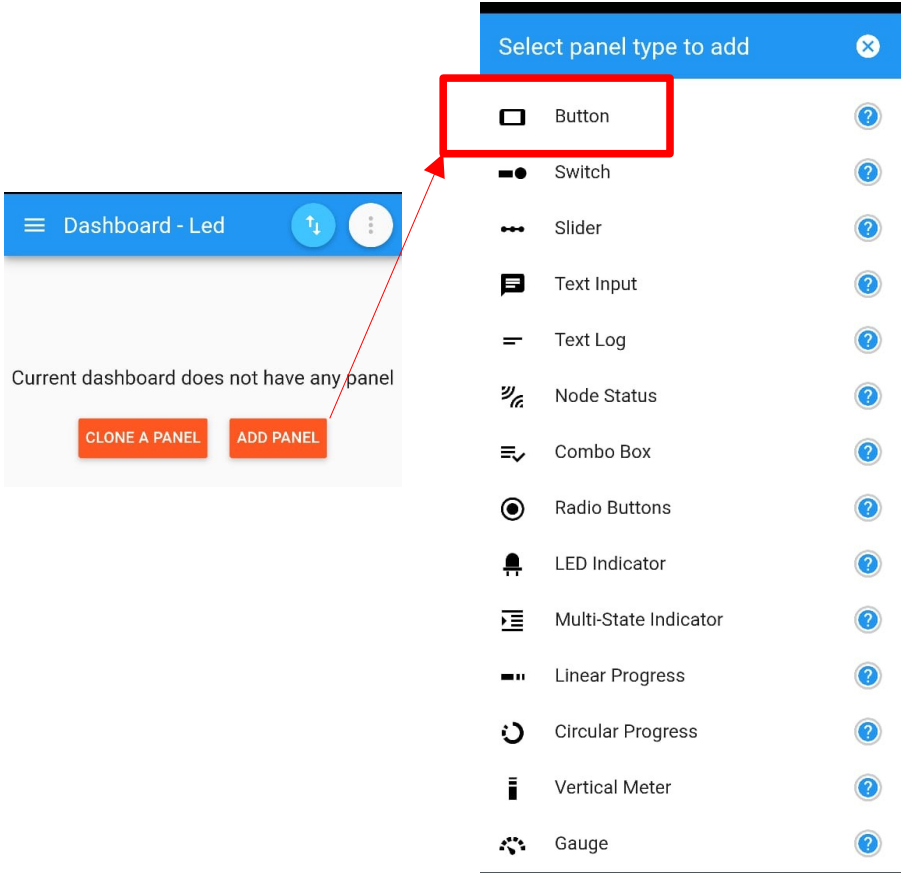
LABORATÓRIO 03

- 9º Em “**Connection name**”, defina um nome de identificação qualquer (Exemplo: Controlar LED)
- 10º Em “**Broker Web/IP address”,** defina o endereço do servidor broker (test.mosquitto.org)
- 11º Em “**Port number**”, digite o número da porta que o servidor está escutando (1883)
- 12º Em **Dashboard list”,** clique no ícone “+” (botão cinza). Em “Dashboard name”, digite “Dashboard – LED”.
- 13º Clique no botão **Create**.



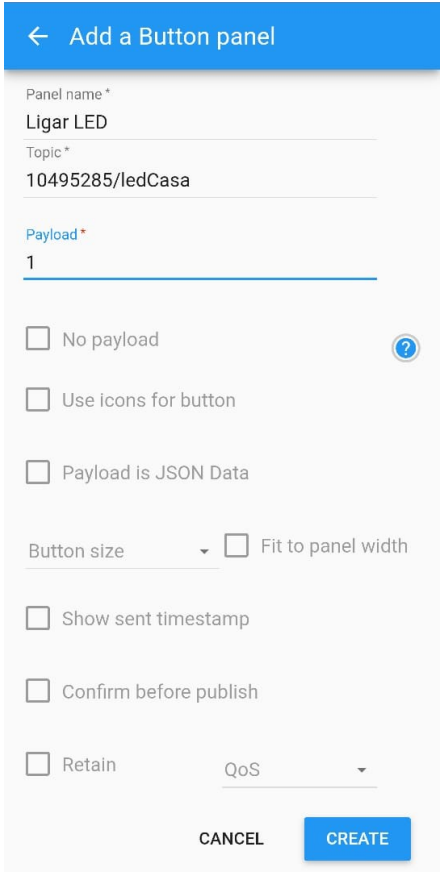
LABORATÓRIO 03

- 14º Na tela inicial do programa, clique sobre “Controlar LED”. Agora iremos adicionar os “painéis”, ou seja, funcionalidades de interação, para criar uma *dashboard* de administração.
- 15º Na tela que aparece, clique em **Add Panel.**
- 16º Iremos criar um botão para ligar o Led. Para tal, escolha **Button**.



LABORATÓRIO 03

- 17º Em **Panel Name**, escreva “Ligar LED”.
- 18º Em **Topic**, escreva o nome do tópico que iremos escrever (10495285/ledCasa)
- 19º Em **Payload**, digite a mensagem que será encaminhada ao servidor *Broker* (1)
- 20º Clique em **Create**. O botão será criado na *dashboard*.



LABORATÓRIO 03

- 21º Repita os procedimentos 15º a 20º, mas desta vez, crie um botão para desligar o Led. Lembre-se de que o nome do painel deverá ser “Desligar LED”, e o payload deverá ser 0, conforme figura ao lado.