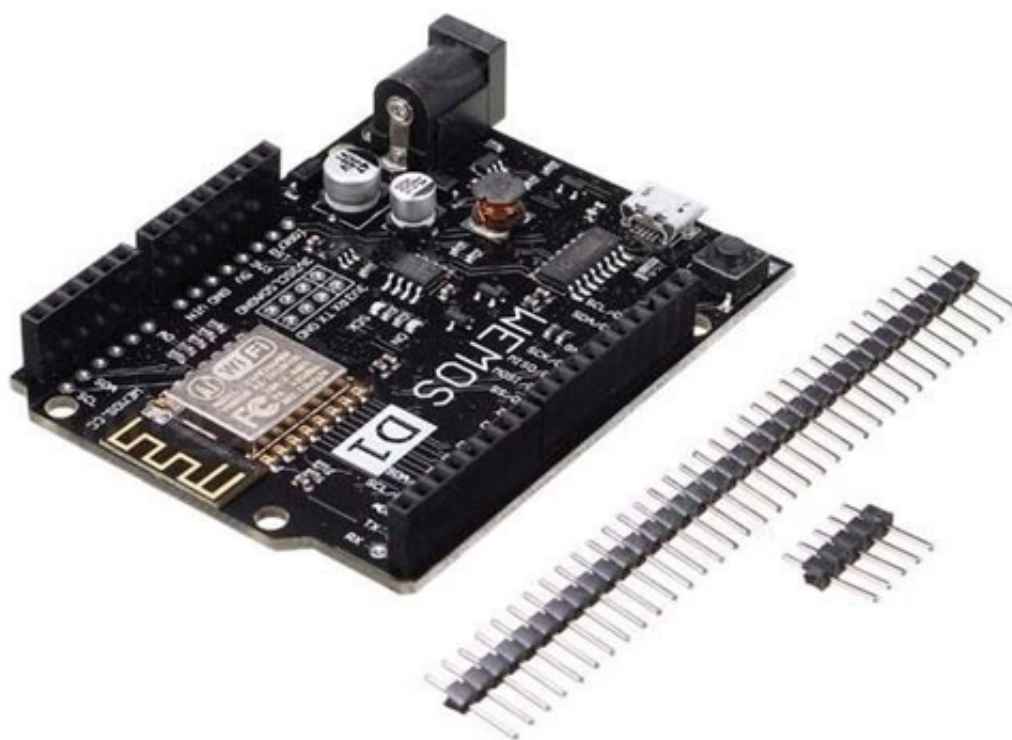


WEMOS D1 e MQTT – Acionando Cargas pela Rede IoT

Direto do nosso Baú, confira como acionar cargas pela rede wifi via MQTT. Tutorial completo.

Placa WEMOS D1

Wemos D1 é uma placa de desenvolvimento que utiliza o módulo ESP 12S, módulo este que é controlado pelo chip ESP – 8266 que oferece conectividade Wifi, sendo uma ótima opção para projetos IoT (Internet das Coisas).



Placa WEMOS

D1

Para saber mais sobre as características da placa Wemos D1, [clique aqui](#).
O tutorial abaixo também pode ser aplicado com a [Wemos D1 Mini](#).

Para realizarmos a programação é necessário Configurar a IDE do Arduino para a placa WEMOS D1,
confira o [post completo](#).

O Protocolo MQTT

O MQTT (Message Queue Telemetry Transport) consiste em um protocolo de troca de

mensagens leves, com largura de banda limitada e pouco consumo de internet. Este protocolo é adequado para o uso de sistemas embarcados e consequentemente no mundo IoT.

O padrão de troca de mensagens MQTT é baseado em publishers, subscribers e brokers. Os publishers (que disponibilizam informações) enviam informações para o broker (servidor MQTT na nuvem) e os subscribers recebem as informações do broker, ou seja, quando uma mensagem é publicada, estas são direcionadas aos subscribers.

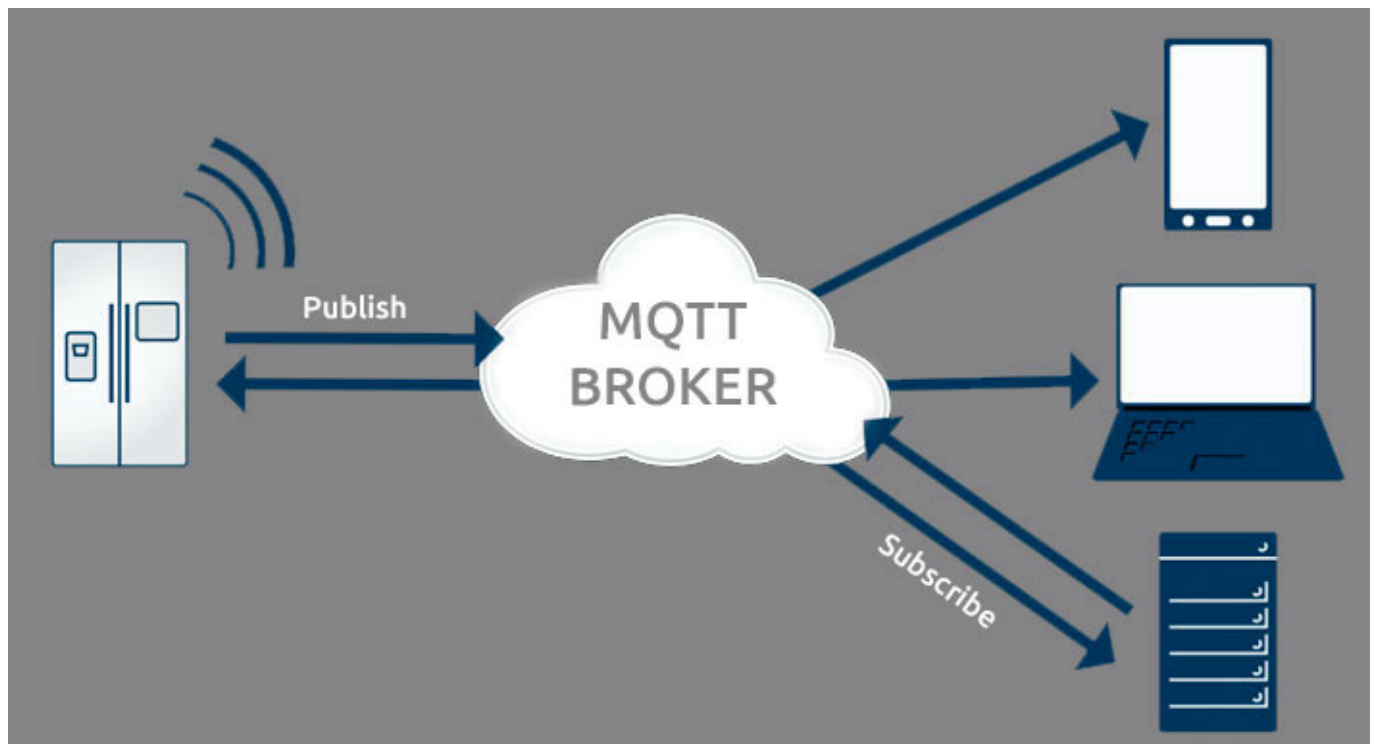


Figura 01: Ilustração Publishers, subscribers e brokers

Como já foi citado é necessário um broker para gerenciar a troca de mensagens e para este tutorial utilizamos o CloudMQTT.

Configurando o CloudMQTT

Passo 01: Entre no site do CloudMQTT (<https://www.cloudmqtt.com>), registre-se e faça o login.

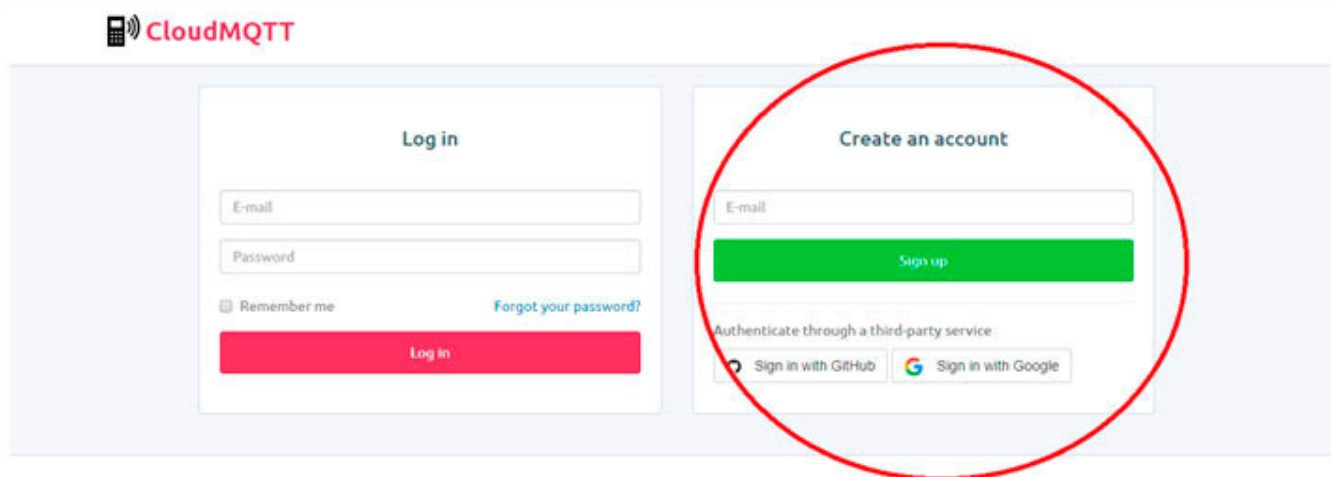


Figura 02: Registro CloudMQTT

Passo 02: Crie uma nova instância:

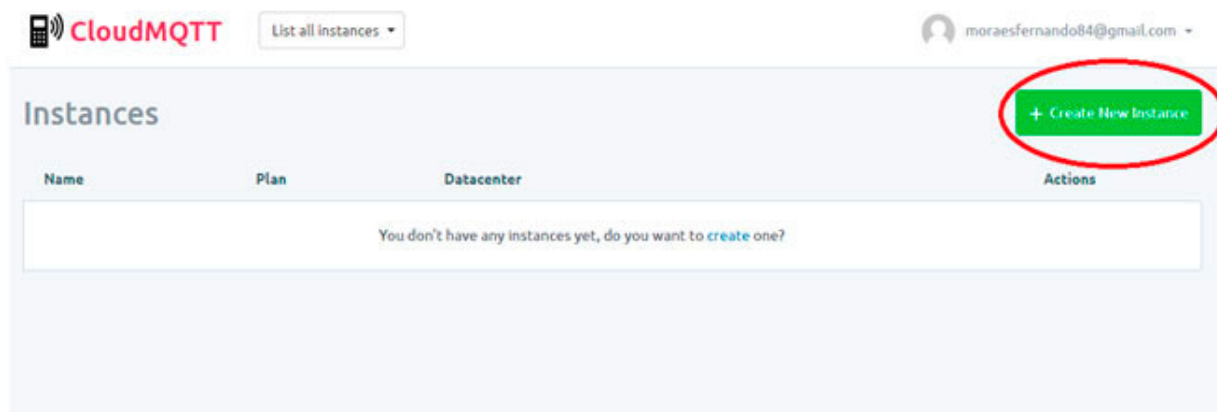


Figura 03: Criando uma nova instância

Passo 03: Preencha o nome da instância e clique em '*Create New Instance*':


Create new instance

No credit card Please [add a credit card](#) if you want to subscribe to a paid plan

Name

Plan


Data center

 powered by
amazon
web services

Tags

Admins can [manage](#) tag access control.

Plan





Cute Cat

See the [plan page](#) to learn about the different plans.

Figura 04: Nome da Nova Instância

Passo 04: Abra a nova instância criada:

 **CloudMQTT**

 [moraesfernando84@gmail.com](#)

Instances

Name	Plan	Datacenter	Actions
Teste	Cat	Amazon Web Services US-East-1 (Northern Virginia)	<input type="button" value="Edit"/>

Figura 05: Abrindo Nova Instância

Passo 05: A tela com as informações para a conexão será aberta. Essas informações serão importantes para o código que será criado.

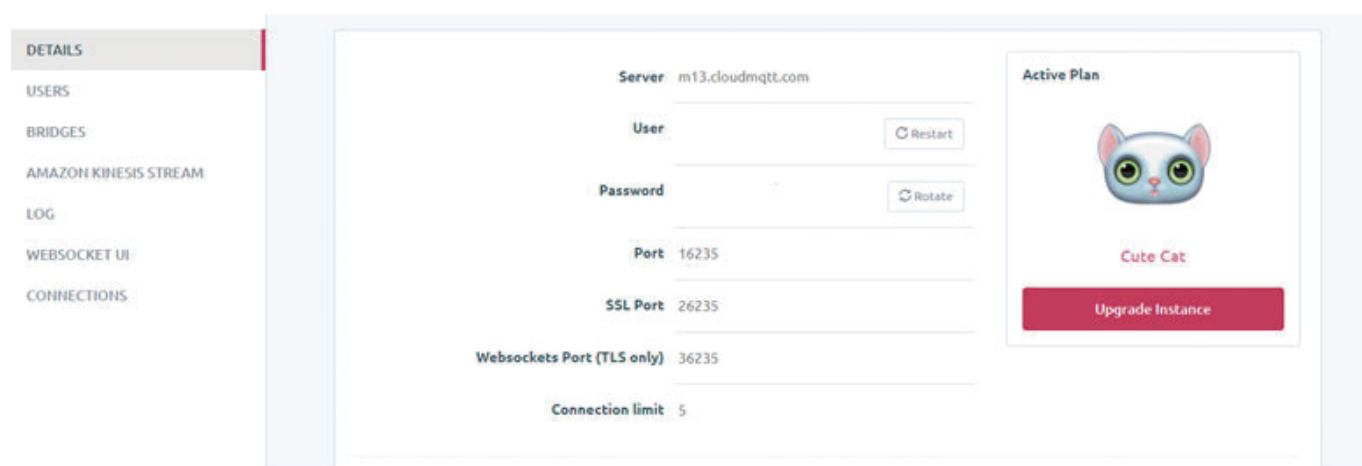


Figura 06: Informações para criar código

Circuito

Para este tutorial, será acionado via wifi um motor DC, conforme o circuito abaixo:

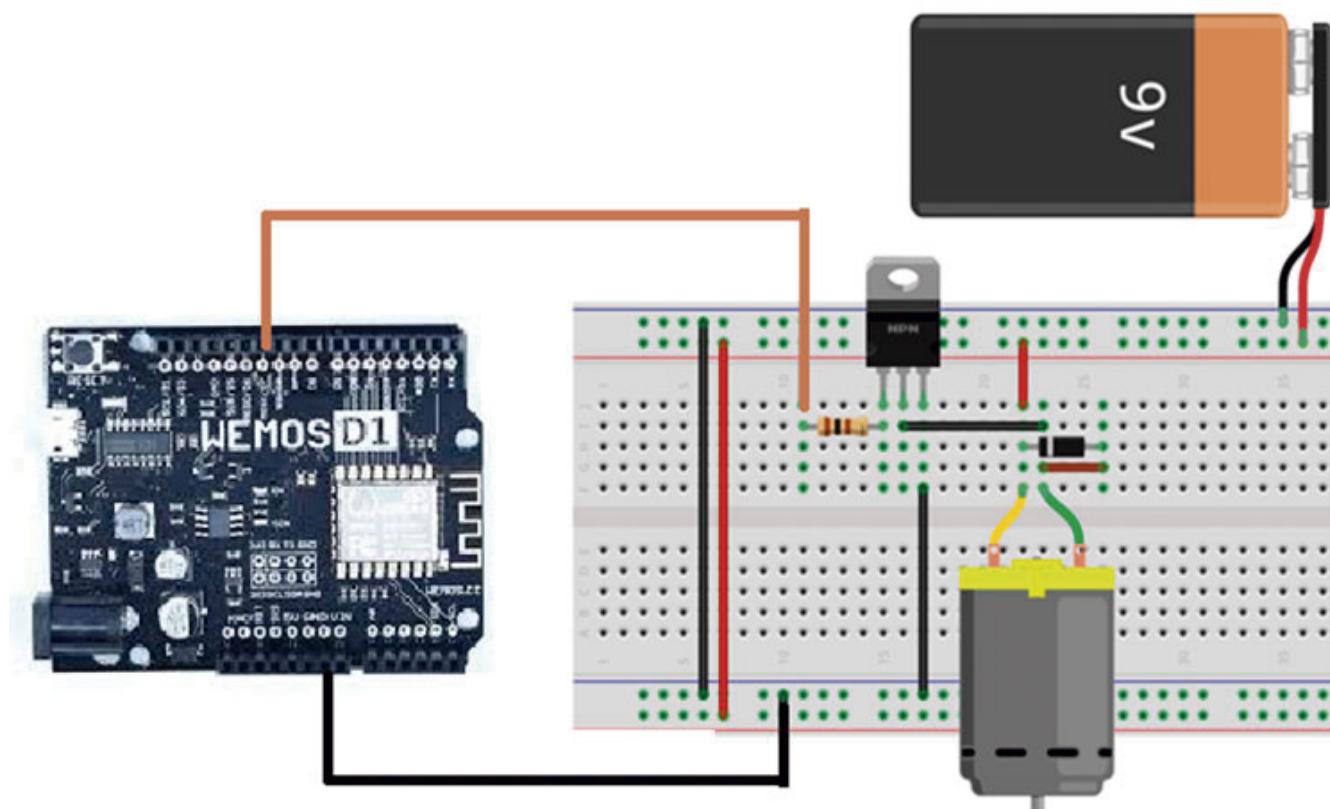


Figura 07: Circuito completo

Lista de Material:

- [1- Placa Wemos D1 R2 Wifi ESP8266](#)
- [1- Micro Motor DC 5V/ 4000rpm – Wotiom](#)

[1- Transistor TIP122](#)[1 - Diodo 1N4007](#)[1 - Resistor 100R](#)[Jumpers \(macho – macho\)](#)

Programa

```
//Projeto: Desenvolvimento IOT - Acionando Cargas com WEMOS D1 e MQTT
//Autor: Baú da Eletrônica      #include #include #define DEBU
G #define P1 13 //pino de saída para acionamento da Lampada L1
//informações da rede WIFI const char* ssid = "xxxxxxxxx";
//SSID da rede WIFI const char* password = "xxxxxxx"; //senha da
rede wifi //informações do broker MQTT - Verifique as informações
geradas pelo CloudMQTT const char* mqttServer = "xxxxxxxxx"; //ser
ver const char* mqttUser = "xxxxxxxxx"; //user const char* mqttPa
ssword = "xxxxxxxxx"; //password const int mqttPort = xxxxxx;
//port const char* mqttTopicSub ="xxxxx"; //tópico que sera ass
inado WiFiClient espClient;// Cria o objeto espClient PubSubClient
client(espClient);// Instancia o Cliente MQTT passando o objeto espCl
ient void setup() { Serial.begin(115200); pinMode(P1, OUTP
UT); WiFi.begin(ssid, password); //Função: verifica o estado d
as conexões WiFi e ao broker MQTT. // Em caso de desconexão (qualque
r uma das duas), a conexão // é refeita. while (WiFi.status() !=
WL_CONNECTED) { delay(500); #ifdef DEBUG Serial.println
("Conectando ao WiFi.."); #endif } client.setServer(mqtt
Server, mqttPort);//informa qual broker e porta deve ser conectado
client.setCallback(callback); //atribui função de callback (função ch
amada quando qualquer informação de um dos tópicos subscritos chega)
while (!client.connected()) { #ifdef DEBUG Serial.pri
ntln("Conectando ao Broker MQTT..."); #endif if (client.c
onnect("ESP8266Client", mqttUser, mqttPassword )) { #ifdef DEBU
G Serial.println("Conectado"); #endif } else
{ #ifdef DEBUG Serial.print("falha estado ");
Serial.print(client.state()); #endif delay(2000);
} } //subscreve no tópico client.subscribe(mqttTopic
Sub); } void callback(char* topic, byte* payload, unsigned int
length) { // Função: função de callback // esta função é chamad
a toda vez que uma informação de // um dos tópicos subscritos chega
) //armazena msg recebida em uma string payload[length] = '\
0'; String strMSG = String((char*)payload); #ifdef DEBUG S
erial.print("Tópico: "); Serial.println(topic); Serial.print("Me
nsagem:"); Serial.print(strMSG); Serial.println(); Serial.pri
ntln("xxxxxxxxxxxxxxxxxxxx"); #endif if (strMSG == "1"){ //
/se msg "1" digitalWrite(P1, HIGH); //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"){ //s
e msg "0" digitalWrite(P1, LOW); //coloca saída em HIGH para d
```

```
esligar a Lampada - > o módulo RELE usado tem acionamento invertido. S
e necessário ajuste para o seu modulo      }      }      //função pra recon
ectar ao servidor 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
    //subscrive no tópico      client.subscribe(mqt
tTopicSub, 1); //nível de qualidade: QoS 1      } else {      #ifdef
DEBUG      Serial.println("Falha durante a conexão.Code: ");
    Serial.println( String(client.state()).c_str());      Serial.printl
n("Tentando novamente em 10 s");      #endif      //Aguard
a 10 segundos      delay(10000);      }      }      }      void l
oop() {      if (!client.connected()) {      reconnect();      }      client.
loop();
```

Explicando o Programa

```
#include      #include      #define DEBUG      #define P1 13
```

Inicialmente foram incluídas as bibliotecas *ESP8266WiFi.h* e *PubSubClient.h*, esta última está sendo utilizada para a publicação e leitura de dados do broker. Também foram definidas as variáveis `DEBUG` e o pino 13 como `P1`.

```
const char* ssid = "xxxxxxxxx";      const char* password = "xxxx
xxx";      const char* mqttServer = "xxxxxxxxx";      const char* mqttUser =
"xxxxxxxxx";      const char* mqttPassword = "xxxxxxxxx";      const int
mqttPort = xxxxx;      const char* mqttTopicSub ="xxxxx";
```

Em seguidas foram fornecidas as informações do ssid da rede wifi e senha da rede (password). Logo em seguida devem ser informados os dados do broker gerados pelo CloudMQTT.

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

Acima são criados objetos globais que serão utilizados mais adiante.

```
void setup() {      Serial.begin(115200);      pinMode(P1, OUTPUT);
    WiFi.begin(ssid, password);
```

A velocidade de comunicação serial é configurada para 115200, o pino 13 (P1) configurado como saída e a conexão wifi é iniciada *WiFi.begin(ssid, password)*.

```
while (WiFi.status() != WL_CONNECTED) {      delay(500);      #ifdef D
DEBUG      Serial.println("Conectando ao WiFi..");      #endif      }
      client.setServer(mqttServer, mqttPort); //informa qual broker e port
a deve ser conectado      client.setCallback(callback);
```

Podemos notar que são feitas tentativas de conexão:

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

Caso seja feita a conexão, é informada a qual broker e porta deve ser conectada:

```
client.setServer(mqttServer, mqttPort);
```

A função callback é ativada. Toda vez que uma mensagem é subscrita essa função é chamada:

```
client.setCallback(callback);
```

```
while (!client.connected()) {      #ifdef DEBUG      Serial.println("C
onectando 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);      }
}      client.subscribe(mqttTopicSub)}
```

É feita a tentativa de se conectar ao broker :

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

Caso a conexão seja satisfeita:

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


O tópico é subscrito (recebido):

```
client.subscribe(mqttTopicSub);
```

Caso a conexão não seja satisfeita será escrita uma mensagem de falha no monitor serial:

```
else {  
  #ifdef DEBUG  
    Serial.print("falha estado ");  
    Serial.print(client.state());  
  #endif  
  delay(2000);  
  
void callback(char* topic, byte* payload, unsigned int length) {  
    payload[length] = '\0';    String strMSG = String((char*)payload);  
    #ifdef DEBUG    Serial.print("Tópico: ");    Serial.println(top  
ic);    Serial.print("Mensagem:");    Serial.print(strMSG);    Serial.  
println();    Serial.println("xxxxxxxxxxxxxxxxxxxx");    #endif    if (s  
trMSG == "1"){        digitalWrite(P1, HIGH);    }else if (  
strMSG == "0"){        digitalWrite(P1, LOW);    }    }
```

Temos a função callback, como já citado, é chamada quando uma informação é subscrita (recebida) pelo cliente. Esta função tem a finalidade de tratar as informações da mensagem. A função tem a seguinte estrutura:

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

topic – tópico (identificação) da mensagem;

payload – mensaem enviada;

length – tamanho da mensagem enviada.

A mensagem recebida é armazenada e escrita no monitor serial:

```
payload[length] = '\0';  
String strMSG = String((char*)payload);  
#ifdef DEBUG  
Serial.print("Tópico: ");  
Serial.println(topic);  
Serial.print("Mensagem:");  
Serial.print(strMSG);  
Serial.println();  
Serial.println("xxxxxxxxxxxxxxxxxxxx"); #endif
```

Se a mensagem armazenada em strMSG for igual a 1 o motor será acionado e se strMSG for igual a 0 o motor desligará:

```
if (strMSG == "1"){  
  digitalWrite(P1, HIGH);
```

```
}else if (strMSG == "0"){  
    digitalWrite(P1, LOW);  
  
}  
  
void reconnect() {  
    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(con  
ectado) {            #ifdef DEBUG        Serial.println("Conectado!");  
        #endif        client.subscribe(mqttTopicSub, 1);        }  
        else {            #ifdef DEBUG        Serial.println("Falha durante a c  
onexão.Code: ");        Serial.println( String(client.state()).c_str()  
);        Serial.println("Tentando novamente em 10 s");        #endif  
            //Aguarda 10 segundos        delay(10000);  
        }    } }
```

O trecho acima se refere à tentativa de reconexão `void reconnect()` conforme já foi conectado. Vale ressaltar que a função `client.subscribe(mqttTopicSub, 1)` se refere ao nível de qualidade QoS=1. Isso significa que acaba gerando várias mensagens iguais garantindo que uma delas será entregue.

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

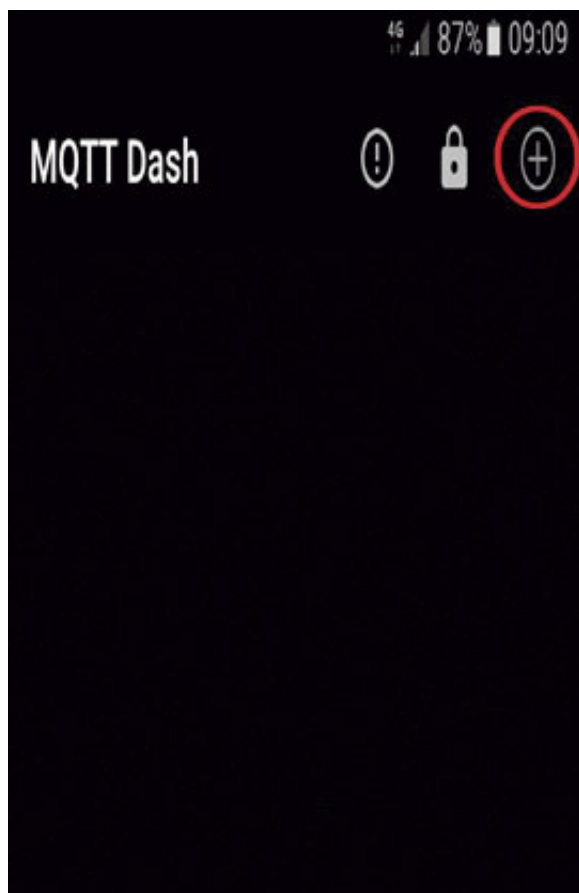
Por fim temos o programa principal onde se houver problema de conexão com a rede, a rotina de reconexão será chamada.

Obs: sobre `#ifdef DEBUG`: toda vez que `ifdef DEBUG` for “encontrado” as funções dentro desta serão executadas até “encontrar” `#endif`. Lembrando que `DEBUG` é uma variável que foi definida no início do código.

Configurando o MQTT Dash

O **MQTT Dash** é um aplicativo gratuito para uso do MQTT no smartphone que pode ser baixado no Google Play.

Passo 01: Após a instalação, clique no sinal + conforme é mostrado abaixo:



Passo 02: Preencha as informações fornecidas pelo Cloud MQTT (figura 06) conforme é mostrado abaixo e clique no disquete do lado superior direito:

MQTT Dash

up).
Note: this option is useful if you have just one connection configured.

☐ If you have more than one connection, you can create home screen shortcut for every connection.
To create shortcut long press on any connection in connections list.

☒ Keep screen on when connected to this broker
Allow metrics management. If disabled, you can't add, edit, delete or rearrange metrics. This serves as protection from unintentional metrics changing.

Name
Baú

Address
m13.cloudmqtt.com

Port
16235

MQTT Dash

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.

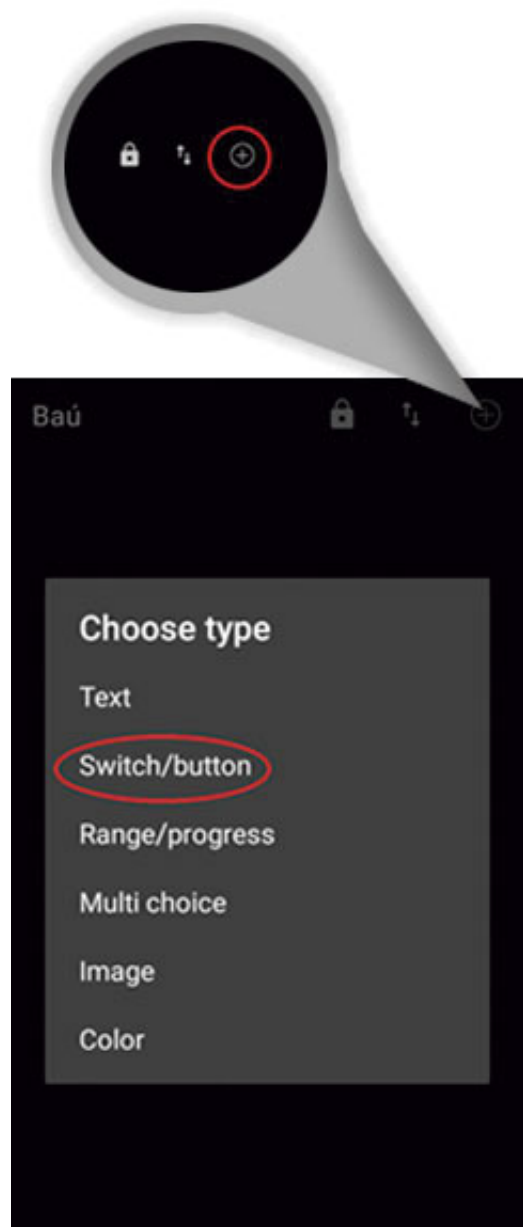
☐ This broker uses self-signed SSL/TLS certificate. I trust this certificate at my own risk.

User name
ahvjcpqg

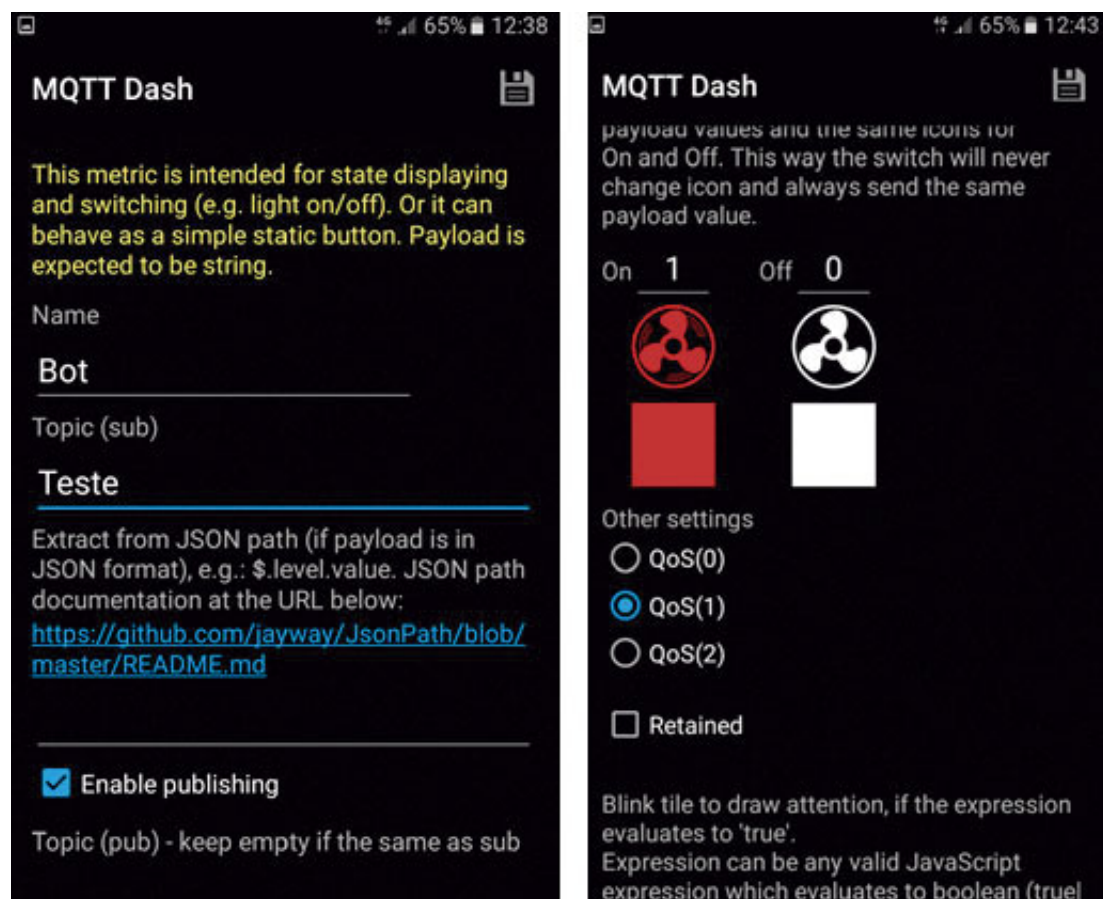
User password
.....

Client ID (must be unique)

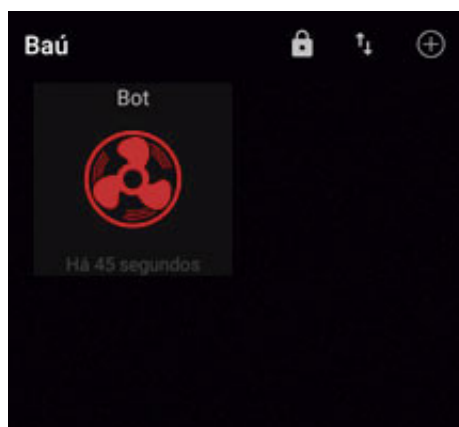
Passo 03: Em seguida clique em Baú, caso as informações estejam corretas, não aparecerá nenhuma mensagem, caso contrário, aparecerá uma mensagem de erro de conexão. Quando estiver tudo correto, clique no sinal + e em seguida selecione Switch/button:



Passo 04: A tela a seguir será aberta, preencha com o nome do botão e em seguida o nome do tópico criado no CloudMQTT. Logo abaixo da mesma tela escolha o ícone que será utilizado para motor ligado e desligado e selecione o nível de qualidade Qos(1):



Passo 05: Clique no disquete. Será aberta a tela com o botão criado:



Mensagens exibidas no monitor serial:

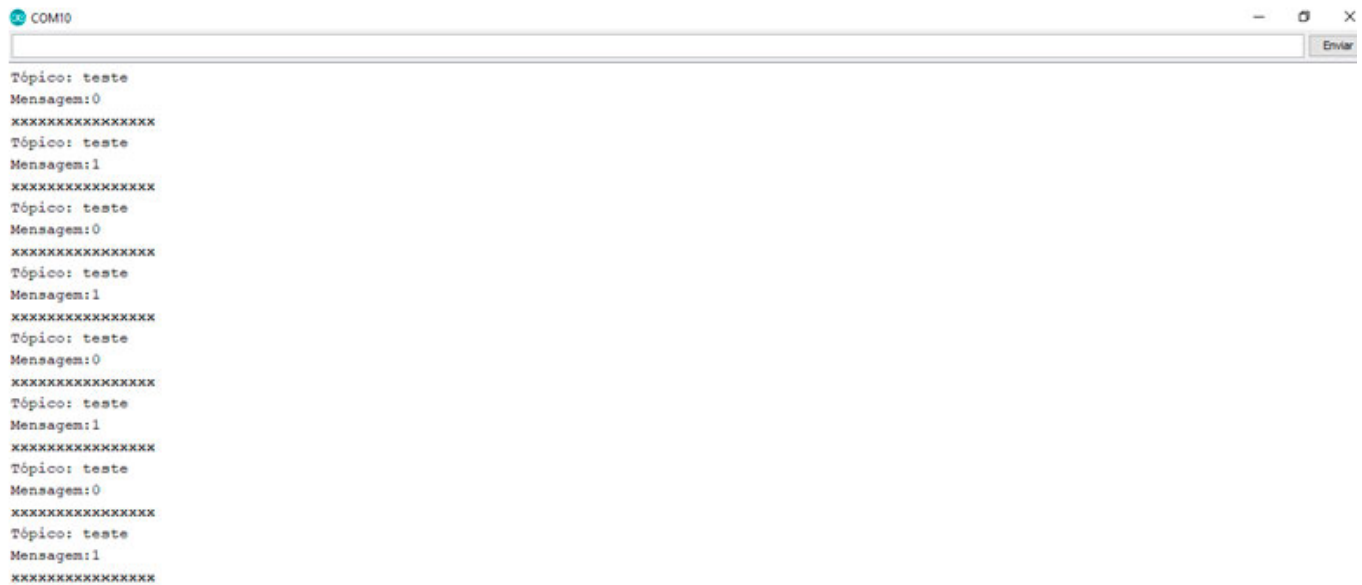


Figura 13: Monitor Serial

Esperamos que tenham gostado deste tutorial.

Post anterior: [Como configurar a IDE do Arduino para utilizar a placa WEMOS D1](#), confira!

Para tirar dúvidas e sugestões, deixe um comentário abaixo. Não esqueça de conferir nossa loja.