

# Prática 03:

## Programando ESP32 para Comunicação MQTT

Disciplina: **Introdução à Internet das Coisas - IMD0902**

Prof. Heitor Florencio

heitorm@imd.ufrn.br

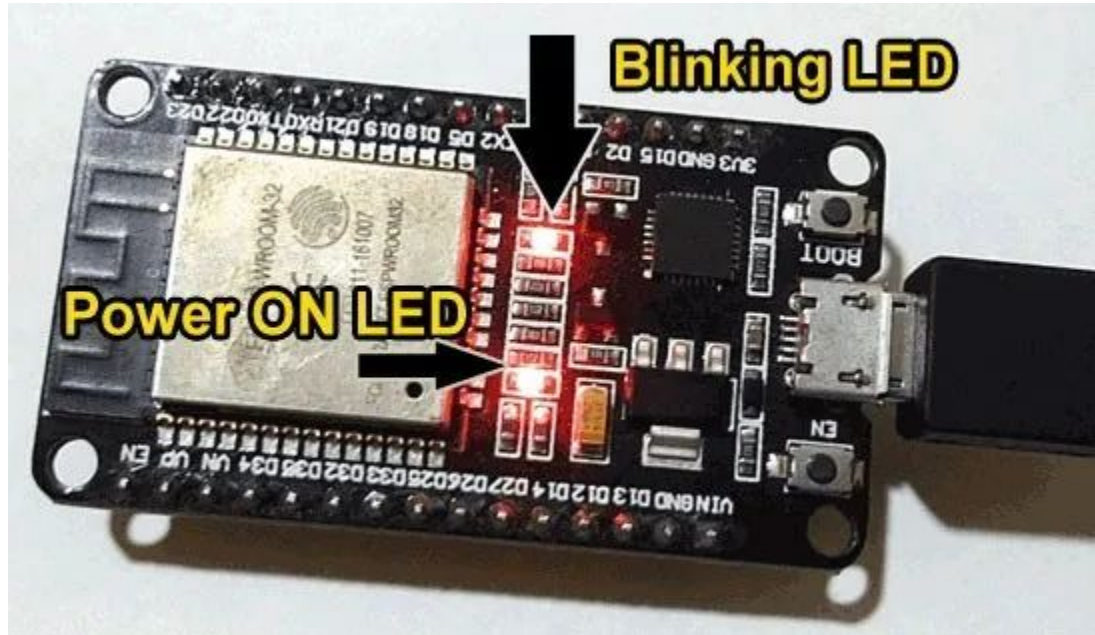
Aula:  
**Prática 03: Programando  
ESP32 para Comunicação  
MQTT**

## Tópicos

- Experimento 01: Conexão com a rede WiFi
- Conhecer o broker HiveMQ
- Experimento 02: Publicar no broker público MQTT Hivemq
- Experimento 03: Assinar tópico no broker público MQTT Hivemq

# Experimento da Prática 01

- Objetivo: Acionar um LED a partir do pino GPIO2 do ESP32.



# Experimento 01: Conexão com a rede WiFi

- Objetivo: Conectar o ESP32 à rede WiFi
- Requisitos funcionais:
  - Configurar o ESP32 para acessar a rede aberta “UFRN”;
  - Usar as bibliotecas:
    - WiFi.h : <https://www.arduino.cc/reference/en/libraries/wifi/>

```
#include <WiFi.h>
```

- Comandos:

```
WiFi.mode()
```

```
WiFi.begin()
```

# Experimento 01: Conexão com a rede WiFi

## WiFi - WiFi.begin()

### Description

Initializes the WiFi library's network settings and provides the current status.

### Syntax

```
WiFi.begin();  
WiFi.begin(ssid);  
WiFi.begin(ssid, pass);
```

```
WiFi.mode(WIFI_STA)
```

station mode: the ESP32 connects to an access point

```
WiFi.mode(WIFI_AP)
```

access point mode: stations can connect to the ESP32

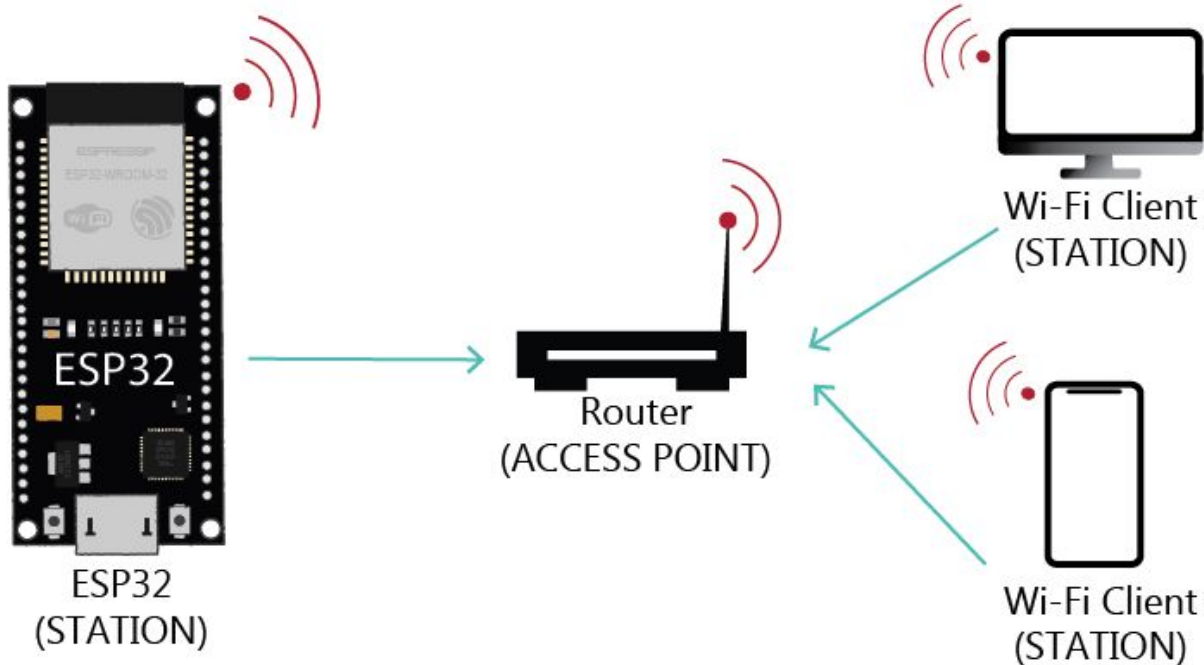
```
WiFi.mode(WIFI_STA_AP)
```

access point and a station connected to another access point

# Experimento 01: Conexão com a rede WiFi

```
WiFi.mode(WIFI_STA)
```

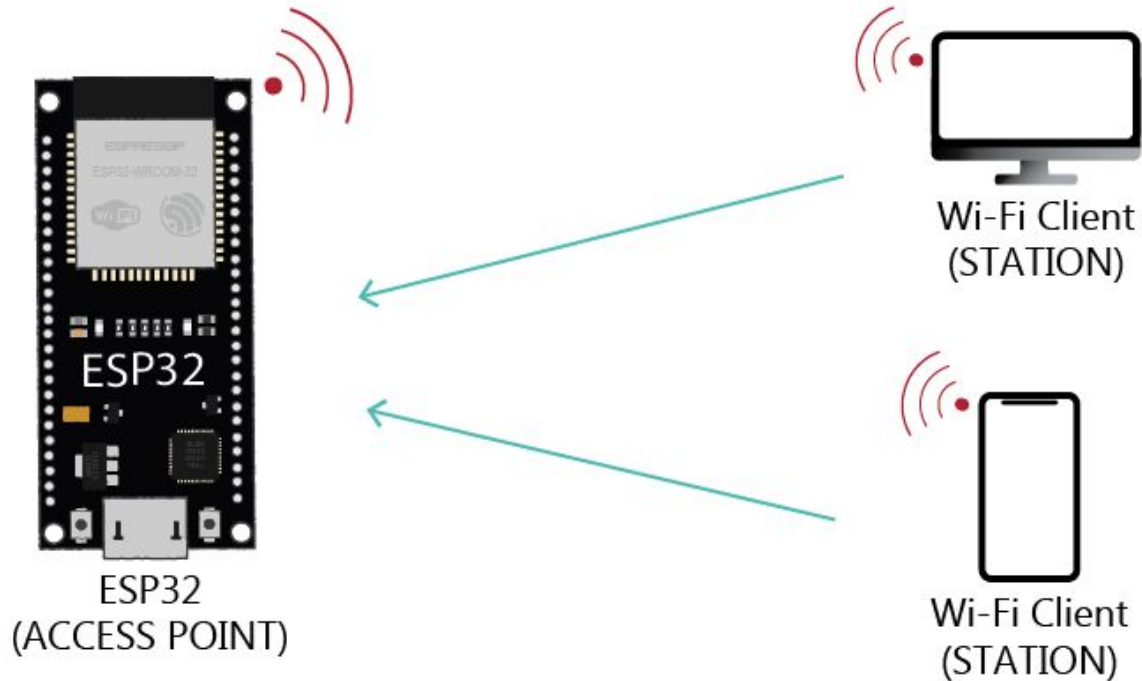
station mode: the ESP32 connects to an access point



# Experimento 01: Conexão com a rede WiFi

```
WiFi.mode(WIFI_AP)
```

access point mode: stations can connect to the ESP32



# Experimento 01: Conexão com a rede WiFi

```
#include <WiFi.h>

const char* wifi_ssid = "UFRN";
const char* wifi_password = "";
int wifi_timeout = 200000;

void conectaWiFi(){
    WiFi.mode(WIFI_STA); //"station mode": permite o ESP32 ser um cliente da rede WiFi
    WiFi.begin(wifi_ssid, wifi_password);
    Serial.print("Conectando à rede WiFi .. ");

    unsigned long startTime = millis();

    while(WiFi.status() != WL_CONNECTED && (millis() - startTime < wifi_timeout)){
        Serial.print(".");
        delay(100);
    }
    Serial.println();

    if(WiFi.status() != WL_CONNECTED){
        Serial.println("Falhou!");
    } else{
        Serial.print("Conectado com o IP: ");
        Serial.println(WiFi.localIP());
    }
}
```



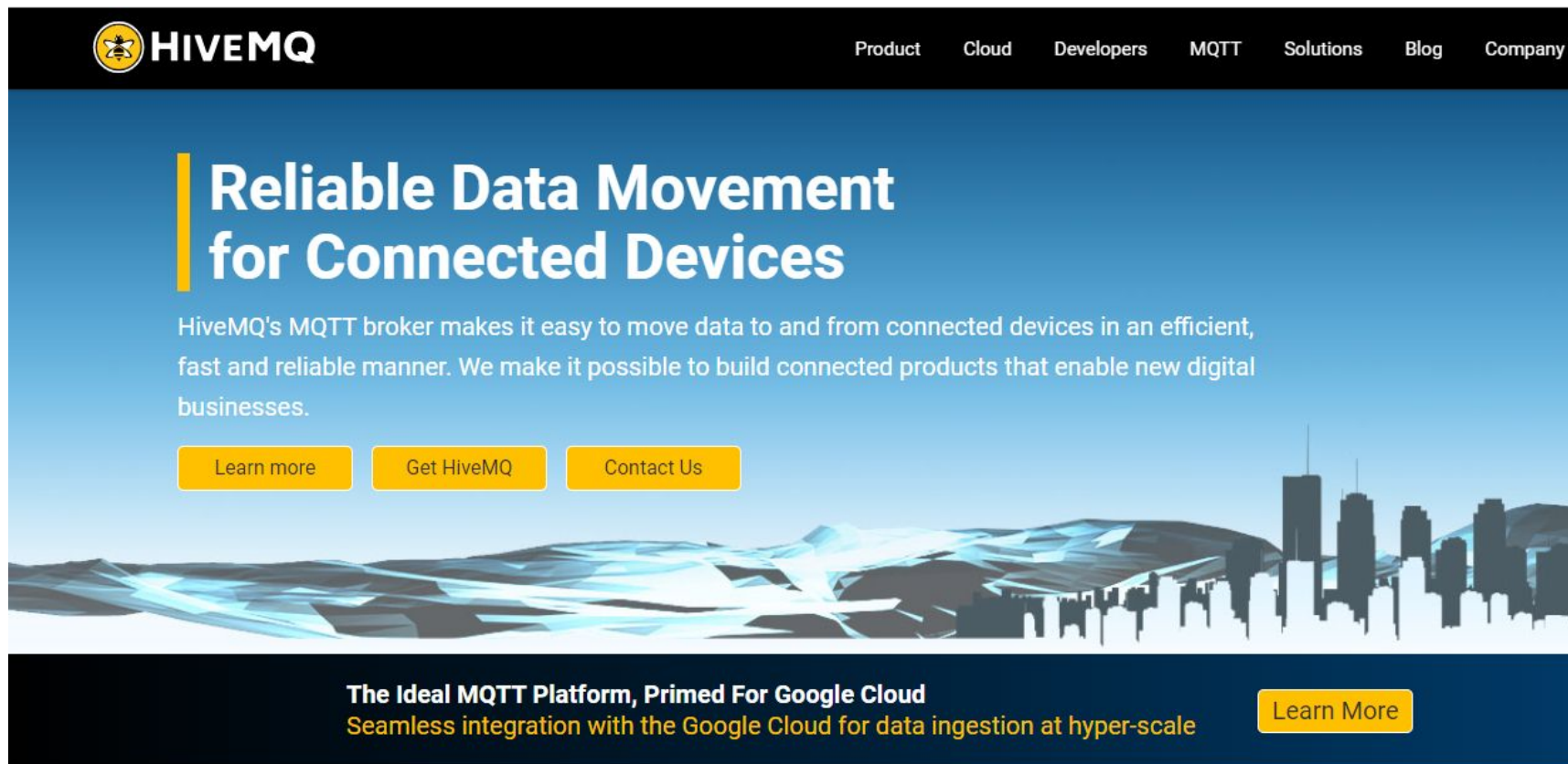
## Experimento 02: Publicar no broker público MQTT Hivemq

- Objetivo: Conectar-se ao broker público do Hivemq com o protocolo MQTT para publicar no tópico “/imd0902/pratica03/distancia”.



**O que é o  
HiveMQ?**

# Experimento 02: Publicar no broker público MQTT Hivemq

The image is a screenshot of the HiveMQ website's main banner. It features a dark blue header with the HiveMQ logo and navigation links. The main content area has a light blue background with a stylized city skyline and mountains at the bottom. The headline is 'Reliable Data Movement for Connected Devices'. Below it is a paragraph describing the MQTT broker. There are three yellow buttons: 'Learn more', 'Get HiveMQ', and 'Contact Us'. At the bottom, there is a dark blue footer with more text and a 'Learn More' button.

**HIVEMQ**

Product Cloud Developers MQTT Solutions Blog Company

## Reliable Data Movement for Connected Devices

HiveMQ's MQTT broker makes it easy to move data to and from connected devices in an efficient, fast and reliable manner. We make it possible to build connected products that enable new digital businesses.

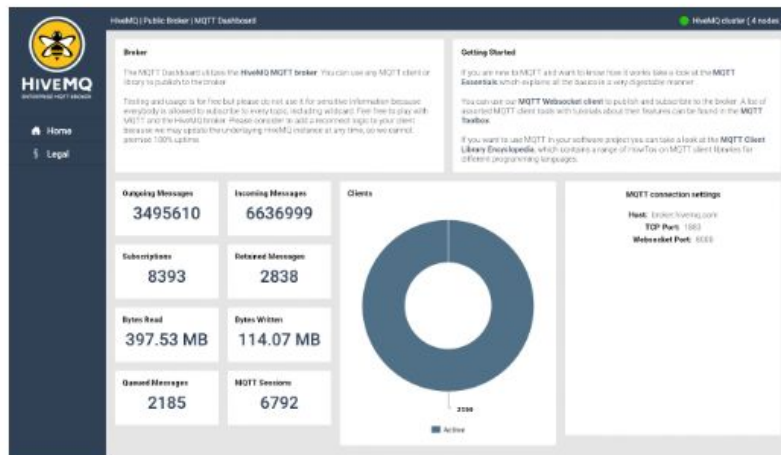
[Learn more](#) [Get HiveMQ](#) [Contact Us](#)

**The Ideal MQTT Platform, Primed For Google Cloud**  
Seamless integration with the Google Cloud for data ingestion at hyper-scale

[Learn More](#)

Link: <https://www.hivemq.com/public-mqtt-broker/>

## Public MQTT Broker



Our **Public HiveMQ MQTT broker** is open for anyone to use. Feel free to write an MQTT client that connects with this broker. We have a **dashboard** so you can see the amount of traffic on this broker. We also keep a list of **MQTT client libraries** that can be used to connect to HiveMQ.

You can access the MQTT broker securely at:

## MQTT Browser Client

The screenshot shows the HiveMQ MQTT Browser Client interface. It features a sidebar with the HiveMQ logo and navigation links for Home and Legal. The main content area includes a 'Connection' section with fields for Host, Port, ClientID, Username, Password, Keep Alive, and Clean Session. Below the connection fields are sections for 'Publish' and 'Subscriptions', each with a 'Messages' button.

Connection settings:  
Host: broker.mqttdashboard.com  
Port: 8080  
ClientID: clientID-ajxwLcFCA  
Username:   
Password:   
Keep Alive: 60  
Clean Session: ☒  
Last-Will Topic:   
Last-Will QoS: 0  
Last-Will Retain: ☐  
Last-Will Message:   
Publish Messages  
Subscriptions Messages

The HiveMQ MQTT Browser Client is an MQTT WebSocket client interface. Use any modern browser on any device as a full-fledged MQTT client and take full advantage of the MQTT protocol.

Try MQTT Browser Client

# Experimento 02: Publicar no broker público MQTT Hivemq

MQTT Connection Settings

Host: **broker.hivemq.com**

TCP Port: **1883**

Websocket Port: **8000**

TLS TCP Port: **8883**

TLS Websocket Port: **8884**

**WARNING!** ⚠

Testing and usage is for free but please do not use it for sensitive information because everybody is allowed to subscribe to every topic, including wildcard.

Feel free to play with MQTT and the HiveMQ broker.

Please consider to add a reconnect logic to your client because we may update the underlying HiveMQ instance at any time, so we cannot promise 100% uptime.



HIVEMQ

Dashboard

Legal



HIVEMQ  
CLOUD

HiveMQ Cloud is now free for up to  
100 MQTT clients.

Learn more



# HIVEMQ

Websockets Client Showcase


**HIVEMQ**  
CLOUD

**Need a fully managed MQTT broker?**

Get your own Cloud broker and connect up to 100 devices for free.

[Get your free account](#)

## Connection

Host

Port

ClientID

[Connect](#)

Username

Password

Keep Alive

SSL

☒

Clean Session

☒

Last-Will Topic

Last-Will QoS

Last-Will Retain

☐

Last-Will Message

Publish



Subscriptions



Messages





Websockets Client Showcase



Need a fully managed MQTT broker?

Get your own Cloud broker and connect up to 100 devices for free.

Get your free account

## Connection

connected

## Publish

Topic

testtopic/1

QoS

0

Retain

☐

Publish

Message

## Subscriptions

Add New Topic Subscription

## Messages

## **Experimento HiveMQ:**

Uso do Node-red para publicar e ler no broker do HiveMQ.

## Experimento 02: Publicar no broker público MQTT Hivemq

- Objetivo: Conectar-se ao broker público do Hivemq com o protocolo MQTT para publicar no tópico “/imd0902/pratica03/variavel”
- Requisitos funcionais:
  - Configurar o ESP32 para acessar a rede aberta “UFRN”;
  - Abrir uma conexão com o broker na porta padrão 1883
    - Use o WiFiClient()
    - Endereço do broker: "broker.hivemq.com"
    - Porta MQTT do broker: 1883
  - Comunicar via MQTT com o broker
    - Use a biblioteca **PubSubClient.h**
  - Publicar no tópico



# Experimento 02: Publicar no broker público MQTT Hivemq

## Passos:

- Incluir bibliotecas;
- Definir objetos do WiFiClient e PubSubClient;
- Definir variáveis de credenciais da rede WiFi e do servidor broker MQTT;

```
sketch_pratica03_esp32-mqtt-hivemq1883-publish $
```

```
#include <WiFi.h>
```

```
#include <WiFiClient.h>
```

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

```
WiFiClient ESPWiFiClient;
```

```
PubSubClient mqttClient(ESPWiFiClient);
```

```
const char* wifi_ssid = "UFRN";
```

```
const char* wifi_password = "";
```

```
int wifi_timeout = 100000;
```

```
const char* mqtt_broker = "broker.hivemq.com";
```

```
const int mqtt_port = 1883;
```

```
int mqtt_timeout = 10000;
```

# Experimento 02: Publicar no broker público MQTT Hivemq

## Passos: setup()

- Configurar Serial para debug do código;
- Configurar pino de saída do LED;
- Executar função de configurar WiFi;
- Configurar o servidor do broker.

```
void setup() {  
  Serial.begin(115200);  
  pinMode(led, OUTPUT);  
  
  connectWifi();  
  if(WiFi.status() == WL_CONNECTED) {  
    Serial.println("Conectando ao broker MQTT ..");  
    mqttClient.setServer(mqtt_broker, mqtt_port);  
  }  
}
```

# Experimento 02: Publicar no broker público MQTT Hivemq

```
void connectWiFi() {
    WiFi.mode(WIFI_STA); // "station mode": permite o ESP32 ser um cliente da rede WiFi
    WiFi.begin(wifi_ssid, wifi_password);
    Serial.print("Conectando à rede WiFi .. ");

    unsigned long startTime = millis();
    while(WiFi.status() != WL_CONNECTED && (millis() - startTime < wifi_timeout)){
        Serial.print(".");
        delay(100);
    }
    Serial.println();

    if(WiFi.status() != WL_CONNECTED) {
        Serial.println("Falhou!");
    } else {
        Serial.print("Conectado com o IP: ");
        Serial.println(WiFi.localIP());
    }
}
```

## Experimento 02: Publicar no broker público MQTT Hivemq

```
void loop() {  
  if (!mqtt_client.connected()){  
    connectMQTT();  
  }  
  
  if (mqtt_client.connected()){  
    mqtt_client.loop();  
  
    variavel = random(1,100);  
  
    mqtt_client.publish("/imd0902/pratica03/variavel", String(variavel).c_str(), true);  
    delay(1000);  
  }  
}
```

# Experimento 02: Publicar no broker público MQTT Hivemq

```
void connectMQTT() {  
    Serial.print("Reconectando ao MQTT Broker..");  
  
    unsigned long startTime = millis();  
    while (!mqttClient.connected() && (millis() - startTime < mqtt_timeout)) {  
        Serial.print(".");  
        String clientId = "ESP32ClientHeitor-";  
        clientId += String(random(0xffff), HEX);  
  
        if (mqttClient.connect(clientId.c_str())) {  
            Serial.println();  
            Serial.print("Conectado ao broker MQTT!");  
        }  
        delay(100);  
    }  
    Serial.println();  
}
```

## Experimento 02: Publicar no broker público MQTT Hivemq

**Implemente  
o código!**

# Dúvidas?

**Prof Heitor Florencio**  
**IMD/UFRN**  
**Sala 103 - nPITI/IMD**  
**[heitorm@imd.ufrn.br](mailto:heitorm@imd.ufrn.br)**

---