

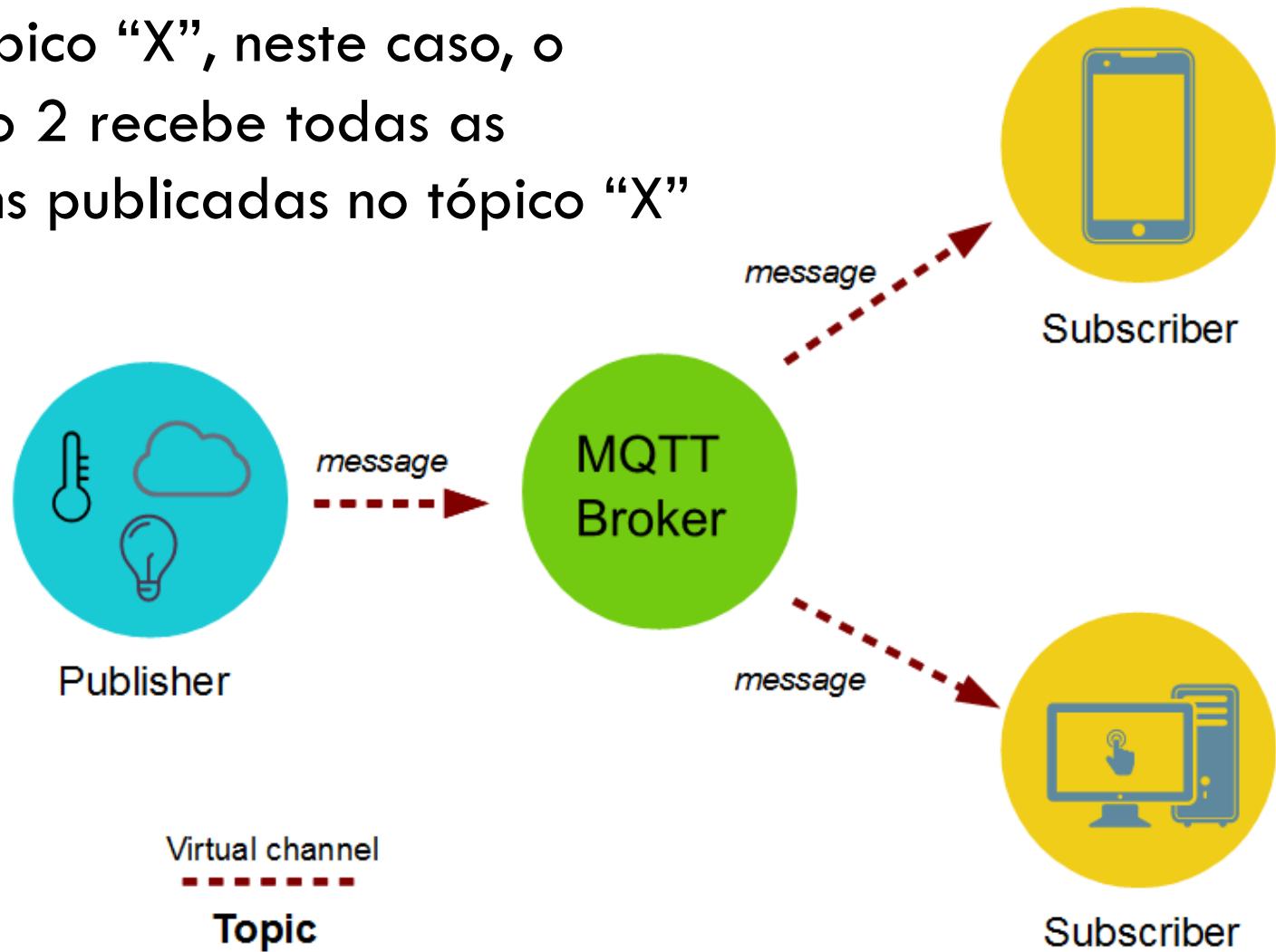
DevTitans

MQTT – Definições e uso do broker Mosquitto

O que é MQTT?

- Message Queuing Telemetry Transport (MQTT) - Transporte de Telemetria por Enfileiramento de Mensagens
- MQTT é um protocolo de mensagens **simples**, projetado para dispositivos **limitados** e com **baixa** largura de banda
- Por isso é a solução perfeita para trocar dados entre vários **dispositivos IoT**
- A comunicação MQTT funciona como um sistema de **publicação** (*publish*) e **assinatura** (*subscribe*)
- Os dispositivos **publicam** mensagens em **tópico** específico
- Todos os dispositivos **inscritos** (que **assinam**) nesse tópico recebem a mensagem

Se o dispositivo 1 publica em um **tópico “X”** e o dispositivo 2 assina o mesmo tópico “X”, neste caso, o dispositivo 2 recebe todas as mensagens publicadas no tópico “X”



MQTT - Tópicos

- O tópico é o link entre **publicador** e o **assinante**
- Os tópicos podem ser representados com strings separadas por uma barra, onde cada barra indica um nível de tópico. Por exemplo: **home/office/lamp**
- Observação: os tópicos diferenciam maiúsculas de minúsculas, o que torna esses dois tópicos diferentes:
- **home/office/lamp** != **home/office/Lamp**

MQTT – Broker

- Outro conceito importante é o **broker** (corretor)
- O broker MQTT é **responsável** por receber todas as mensagens, filtrar as mensagens, decidir quem está interessado nelas e depois publicar a mensagem para todos os clientes inscritos.
- Existem **vários brokers disponíveis** (pagos ou de graça)
- É comum utilizarmos o **Mosquitto Broker**, que pode ser instalado em PCs e, inclusive, no Raspberry Pi
- Não é muito **conveniente instalar no PC** porque você precisa manter seu computador **funcionando o tempo todo** para manter a conexão MQTT entre seus dispositivos.

MQTT Broker Mosquitto

Mosquitto

- Vamos utilizar o Mosquitto para acessar um Broker na nuvem
 - Broker Open Source desenvolvido pela Eclipse
 - Compatível com Windows/Linux/Mac
 - Disponível para outras plataformas, incluindo Raspberry PI



<http://mosquitto.org/>

Instalação



[Home](#) [Blog](#) [Download](#) [Documentation](#) ▾

Download

Source

- [mosquitto-1.5.tar.gz](#) (319kB) ([GPG signature](#))
- [mosquitto-1.4.15.tar.gz](#) (via Eclipse)
- [Git source code repository](#) (github.com)

Older downloads are available at <http://mosquitto.org/files/>

Binary Installation

The binary packages listed below are supported by the Mosquitto project. In many cases Mosquitto is also available directly from official Linux/BSD distributions.

Windows

- [mosquitto-1.5.1-install-windows-x64.exe](#) (~360 kB) (64-bit build, Windows Vista and up, built with Visual Studio Community 2017)
- [mosquitto-1.5.1-install-windows-x32.exe](#) (~360 kB) (32-bit build, Windows Vista and up, built with Visual Studio Community 2017)

You will also need to install Win64 OpenSSL v1.1.0 Light or Win32OpenSSL v1.1.0 Light from slproweb.com

See also `readme-windows.txt` after installing.

Mac

Mosquitto can be installed from the homebrew project. See [brew.sh](#) and then use `brew install mosquitto`

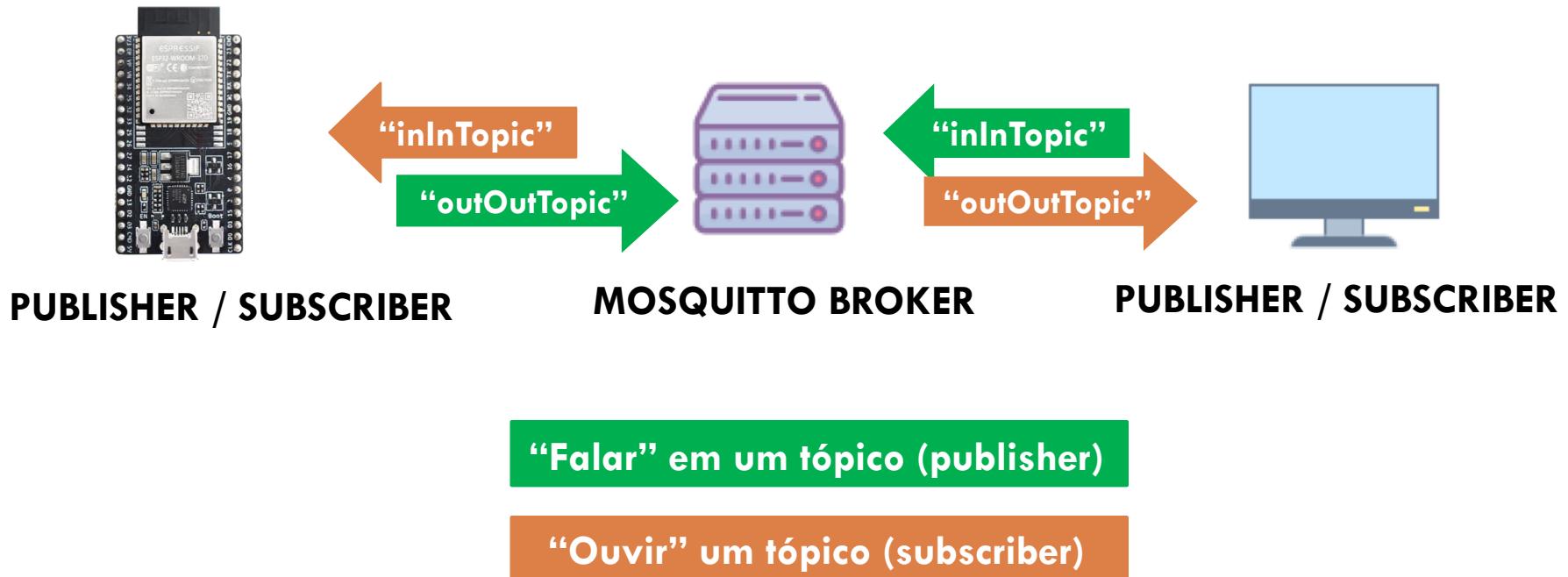
Instalação de biblioteca

- Verifique como instalar o broker em seu sistema operacional
- Instalar lib no Arduino: “PubSubClient”



Utilização

Visão geral



Vamos programar um ESP32 para publicar (“**outOutTopic**”) e assinar (“**inInTopic**”) através do broker “**test.mosquitto.org**”

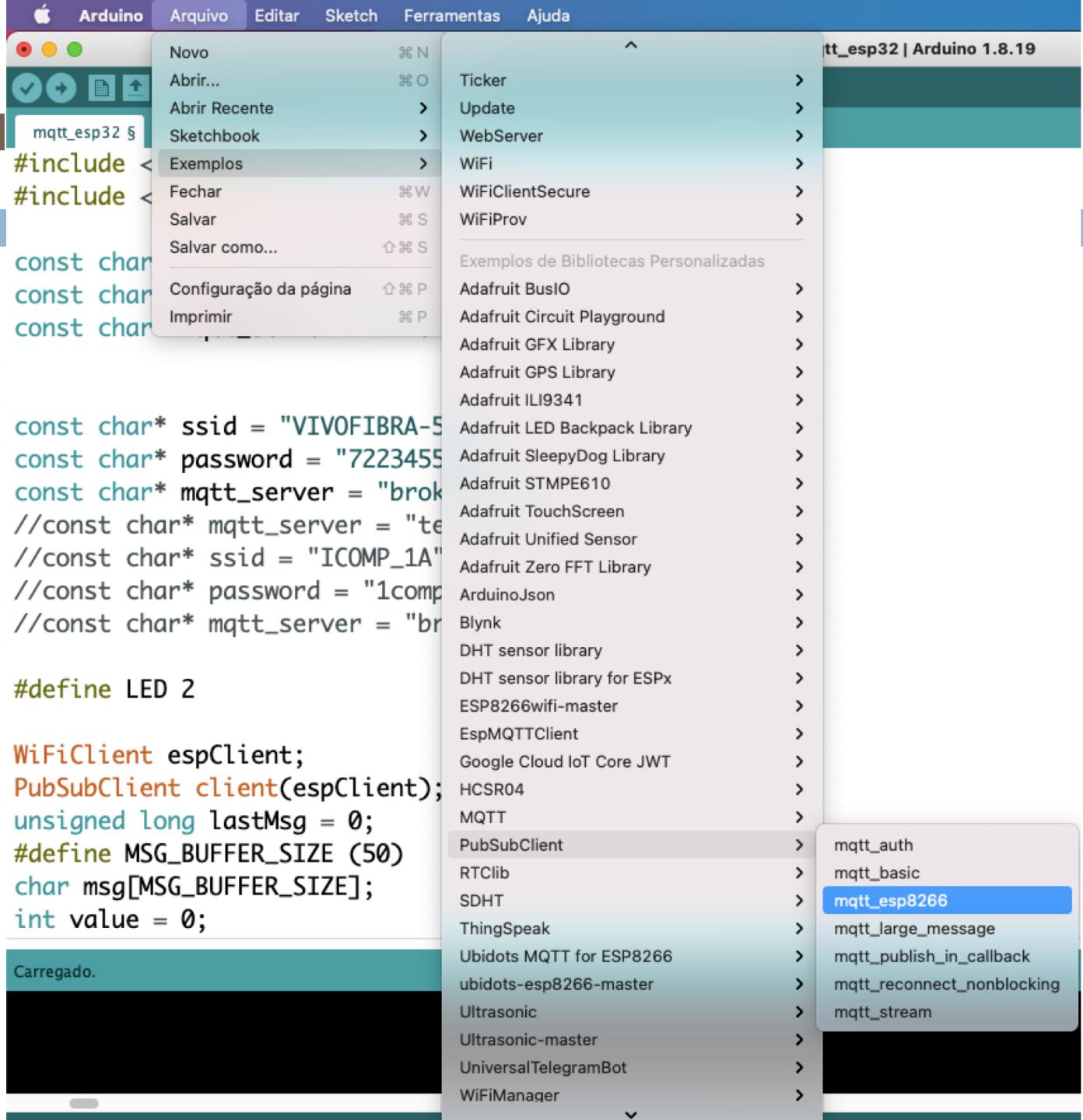
Codifica

Carregar
sketch de
exemplo:
Arquivos>

Exemplos>

PubSubClient>

mqtt_esp8266



Novo ⌘ N
Abrir... ⌘ O
Abrir Recente >
Sketchbook >
Exemplos >
Fechar ⌘ W
Salvar ⌘ S
Salvar como... ⌘ ⌥ S
Configuração da página ⌘ ⌥ P
Imprimir ⌘ P

Exemplos de Bibliotecas Personalizadas

- Adafruit BusIO >
- Adafruit Circuit Playground >
- Adafruit GFX Library >
- Adafruit GPS Library >
- Adafruit ILI9341 >
- Adafruit LED Backpack Library >
- Adafruit SleepyDog Library >
- Adafruit STMPE610 >
- Adafruit TouchScreen >
- Adafruit Unified Sensor >
- Adafruit Zero FFT Library >
- ArduinoJson >
- Blynk >
- DHT sensor library >
- DHT sensor library for ESPx >
- ESP8266wifi-master >
- EspMQTTClient >
- Google Cloud IoT Core JWT >
- HCSR04 >
- MQTT >
- PubSubClient >**
- RTClib >
- SDHT >
- ThingSpeak >
- Ubidots MQTT for ESP8266 >
- ubidots-esp8266-master >
- Ultrasonic >
- Ultrasonic-master >
- UniversalTelegramBot >
- WiFiManager >

mqtt_auth
mqtt_basic
mqtt_esp8266
mqtt_large_message
mqtt_publish_in_callback
mqtt_reconnect_nonblocking
mqtt_stream

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

const char* ssid = "VIVOFIBRA-5";
const char* password = "7223455";
const char* mqtt_server = "brok";
//const char* mqtt_server = "te";
//const char* ssid = "ICOMP_1A";
//const char* password = "1comp";
//const char* mqtt_server = "br"

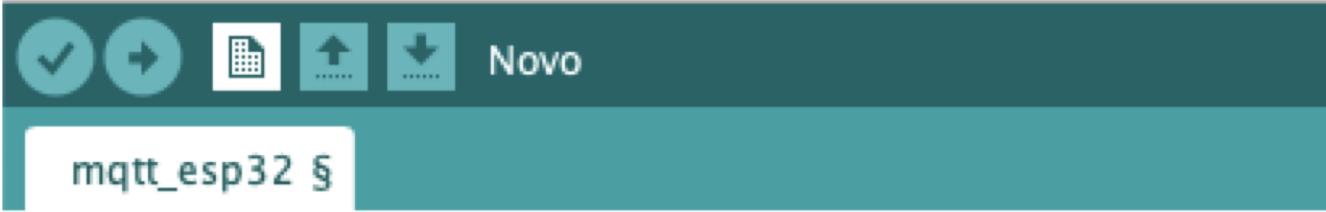
#define LED 2

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;

Carregado.
```

Codificação ESP32

Substituir valores: **ssid**, **password** e **mqtt_server**



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

const char* ssid = "SSID";
const char* password = "PWD";
const char* mqtt_server = "BROKER";
```

Só incluir o SSID da rede, senha e informação do Broker que já dá para executar uma boa aplicação

Brokers

Há diversos tipos de brokers grátis disponíveis:

- test.mosquitto.org
- broker.emqx.io
- broker.hivemq.com
- broker.mqtt-dashboard.com
- . . .

Codificação ESP32 - Subscribe

```
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

// Switch on the LED if an 1 was received as first character
if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW); // Turn the LED off
} else {
    digitalWrite(BUILTIN_LED, HIGH); // Turn the LED on
}
}
```

A função **callback()** é a que trata das mensagens que chegam em **tópicos** que o ESP32 **assina**. Se o primeiro caractere da msg for “1” o ESP apaga o LED

Codificação ESP32 - Publisher

```
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    unsigned long now = millis();
    if (now - lastMsg > 3000) {
        lastMsg = now;
        ++value;
        snprintf (msg, MSG_BUFFER_SIZE, "hello world #%ld", value);
        Serial.print("Publish message: ");
        Serial.println(msg);
        client.publish("outOutTopic", msg);
    }
}
```

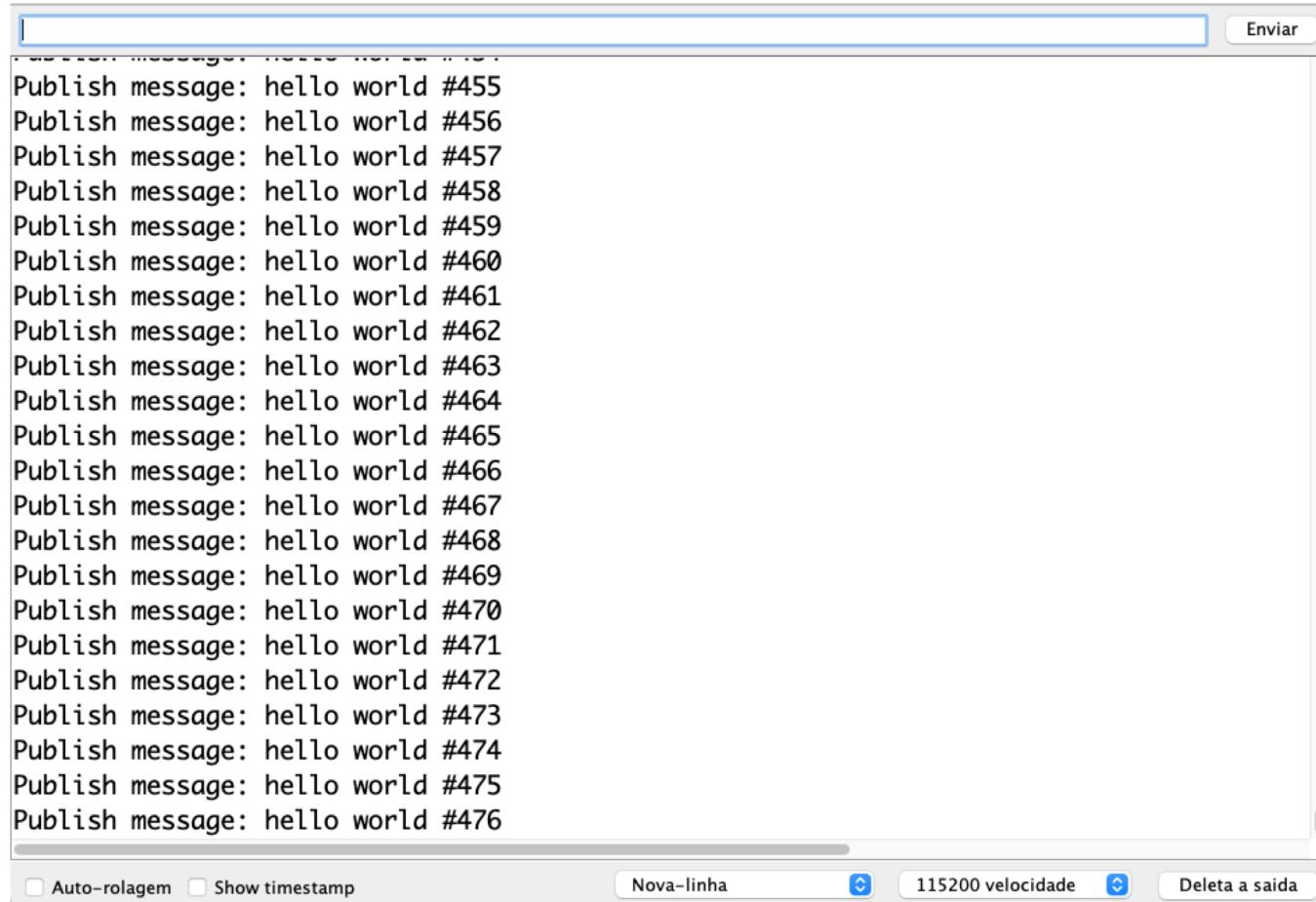
A cada 3 segundos **publica** as mensagens no **tópico “outOutTopic”**

Codificação ESP32 - Pub/Sub

```
void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Create a random client ID
        String clientId = "ESP8266Client-";
        clientId += String(random(0xffff), HEX);
        // Attempt to connect
        if (client.connect(clientId.c_str())) {
            Serial.println("connected");
            // Once connected, publish an announcement...
            client.publish("outOutTopic", "hello world");
            // and resubscribe
            client.subscribe("inInTopic");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}
```

Indicação dos tópicos onde publicar (**outOutTopic**) e assinar (**inInTopic**)

ESP32 publicando

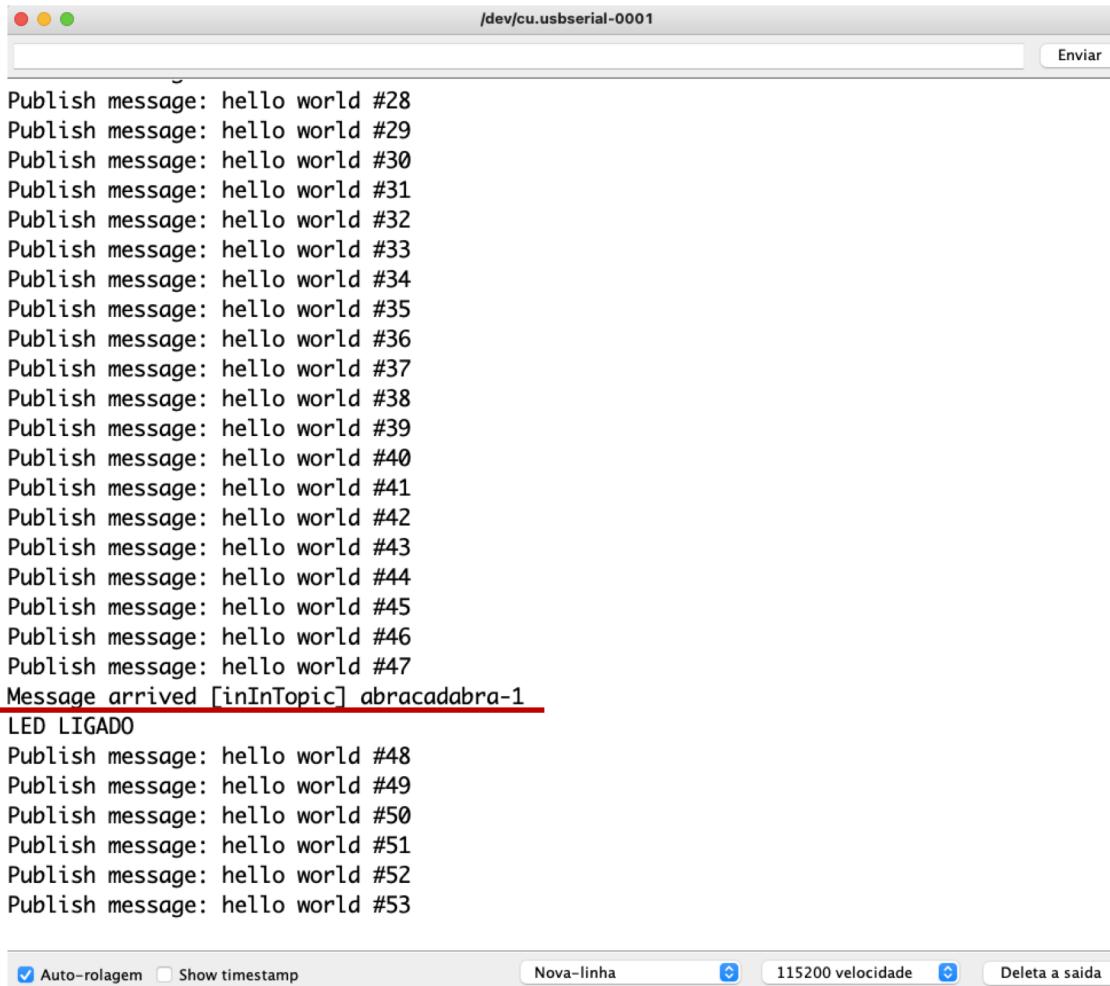


```
Publish message: hello world #455
Publish message: hello world #456
Publish message: hello world #457
Publish message: hello world #458
Publish message: hello world #459
Publish message: hello world #460
Publish message: hello world #461
Publish message: hello world #462
Publish message: hello world #463
Publish message: hello world #464
Publish message: hello world #465
Publish message: hello world #466
Publish message: hello world #467
Publish message: hello world #468
Publish message: hello world #469
Publish message: hello world #470
Publish message: hello world #471
Publish message: hello world #472
Publish message: hello world #473
Publish message: hello world #474
Publish message: hello world #475
Publish message: hello world #476
```

Auto-rolagem Show timestamp Nova-linha 115200 velocidade Deleta a saída

ESP32 publica no tópico “`outOutTopic`”

ESP32 também assina



A screenshot of a terminal window titled '/dev/cu.usbserial-0001'. The window shows a series of 'Publish message' logs followed by a message arrival log. The logs are as follows:

```
Publish message: hello world #28
Publish message: hello world #29
Publish message: hello world #30
Publish message: hello world #31
Publish message: hello world #32
Publish message: hello world #33
Publish message: hello world #34
Publish message: hello world #35
Publish message: hello world #36
Publish message: hello world #37
Publish message: hello world #38
Publish message: hello world #39
Publish message: hello world #40
Publish message: hello world #41
Publish message: hello world #42
Publish message: hello world #43
Publish message: hello world #44
Publish message: hello world #45
Publish message: hello world #46
Publish message: hello world #47
Message arrived [inInTopic] abracadabra-1
LED LIGADO
Publish message: hello world #48
Publish message: hello world #49
Publish message: hello world #50
Publish message: hello world #51
Publish message: hello world #52
Publish message: hello world #53
```

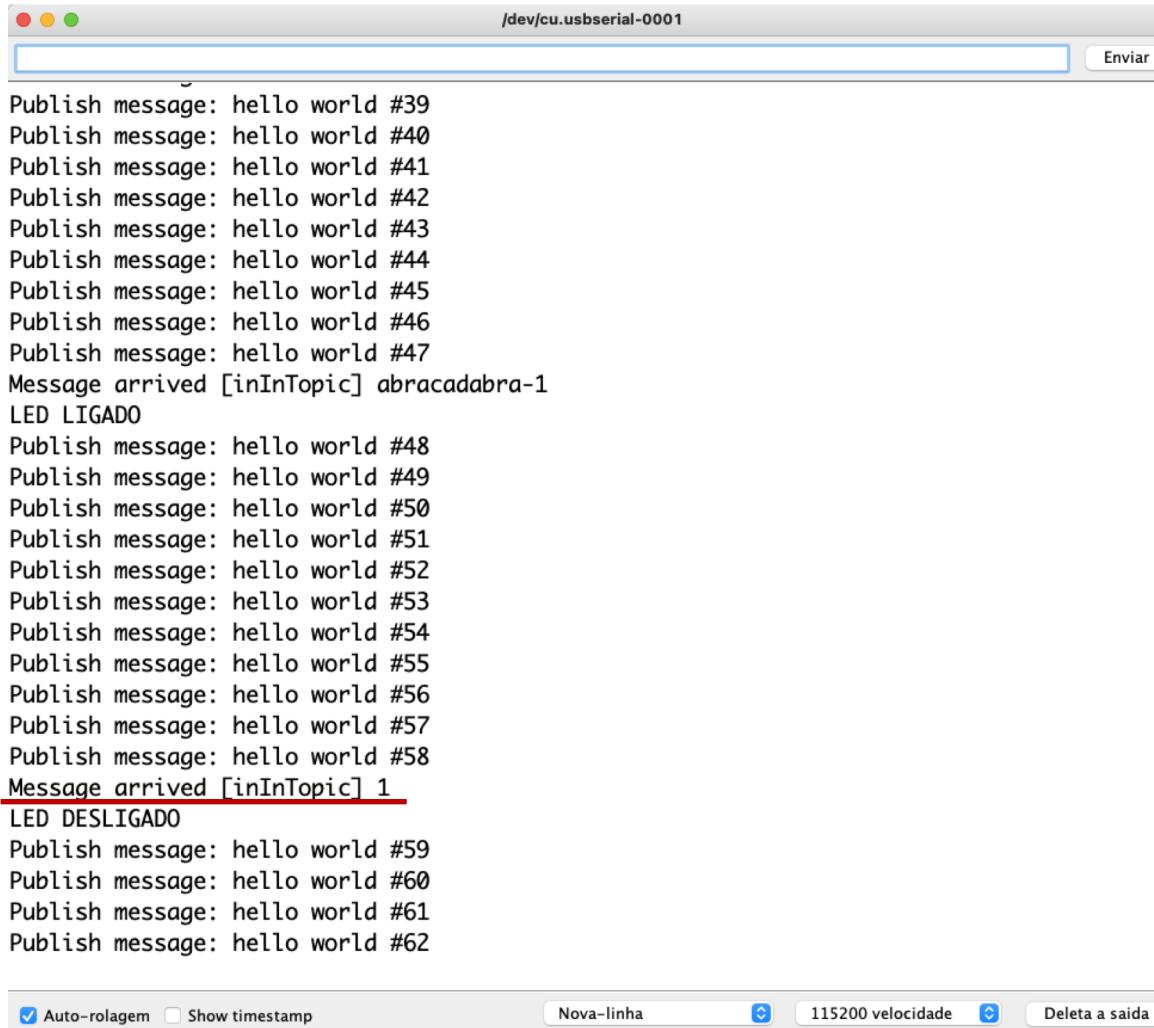
The message 'Message arrived [inInTopic] abracadabra-1' is underlined in red, indicating it triggered an event. The word 'LED LIGADO' is also underlined in red, indicating the LED state.

At the bottom of the terminal window, there are several status indicators: a checked checkbox for 'Auto-rolagem', an unchecked checkbox for 'Show timestamp', a 'Nova-linha' button, a dropdown menu set to '115200 velocidade', and a 'Delete a saida' button.

**ESP32 assina o
tópico “inInTopic”.**

**A mensagem
recebida não começa
com “1”, portanto o
LED é LIGADO**

ESP32 também assina



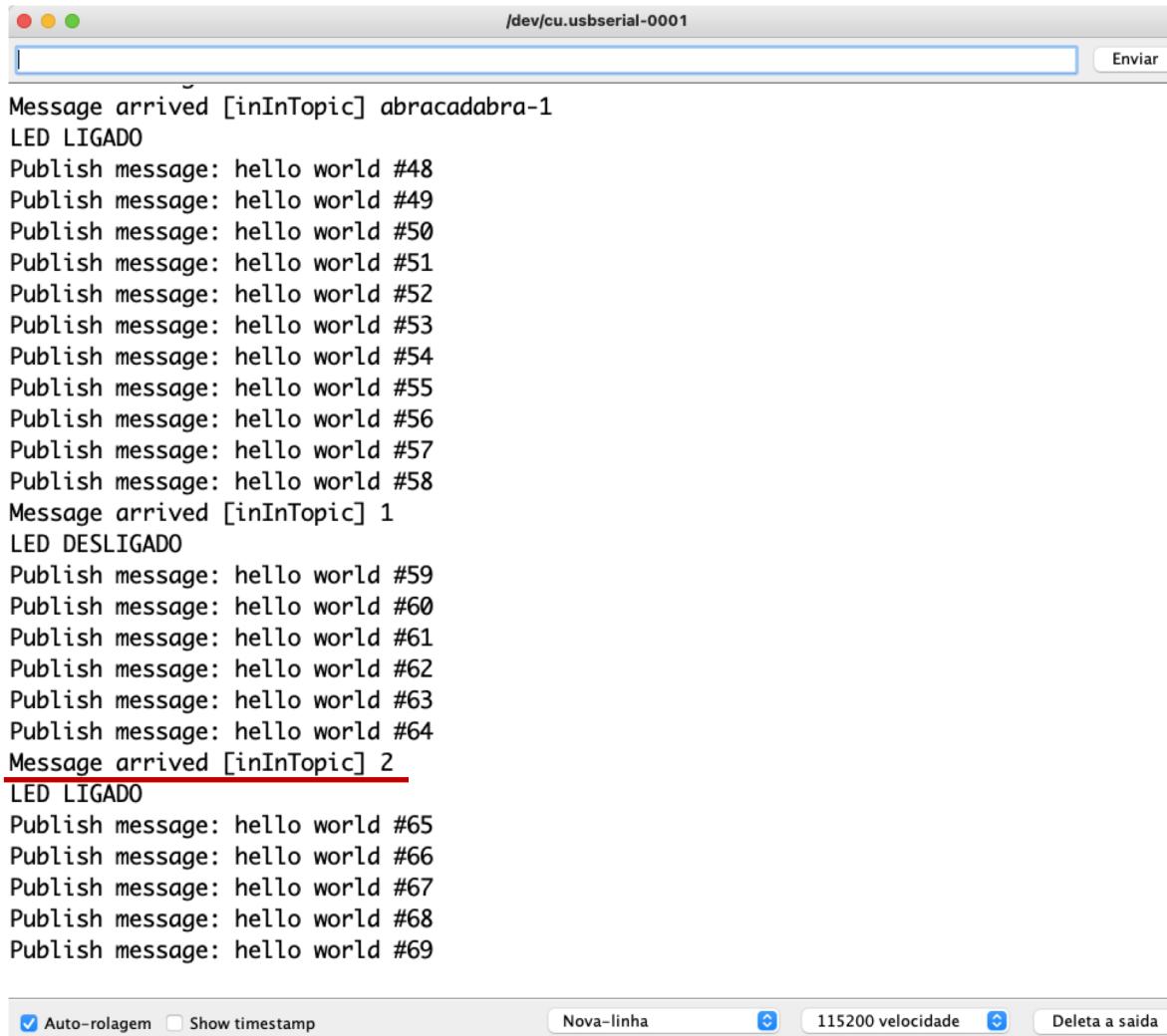
A screenshot of a terminal window titled '/dev/cu.usbserial-0001'. The window has a blue header bar with the title and a red, yellow, and green close button. Below the title is a text input field with a blue border and a 'Enviar' (Send) button. The main area of the terminal shows the following text:

```
Publish message: hello world #39
Publish message: hello world #40
Publish message: hello world #41
Publish message: hello world #42
Publish message: hello world #43
Publish message: hello world #44
Publish message: hello world #45
Publish message: hello world #46
Publish message: hello world #47
Message arrived [inInTopic] abracadabra-1
LED LIGADO
Publish message: hello world #48
Publish message: hello world #49
Publish message: hello world #50
Publish message: hello world #51
Publish message: hello world #52
Publish message: hello world #53
Publish message: hello world #54
Publish message: hello world #55
Publish message: hello world #56
Publish message: hello world #57
Publish message: hello world #58
Message arrived [inInTopic] 1
LED DESLIGADO
Publish message: hello world #59
Publish message: hello world #60
Publish message: hello world #61
Publish message: hello world #62
```

At the bottom of the terminal window, there are several status indicators: a checked checkbox for 'Auto-rolagem' (Auto-scroll), an unchecked checkbox for 'Show timestamp', a 'Nova-linha' (New line) button, a '115200 velocidade' (115200 speed) button, and a 'Delete a saída' (Delete output) button.

**A mensagem
recebida começa com
“1”, portanto o LED é
DESLIGADO**

ESP32 também assina



The screenshot shows a terminal window titled '/dev/cu.usbserial-0001'. The window contains the following text:

```
Message arrived [inInTopic] abracadabra-1
LED LIGADO
Publish message: hello world #48
Publish message: hello world #49
Publish message: hello world #50
Publish message: hello world #51
Publish message: hello world #52
Publish message: hello world #53
Publish message: hello world #54
Publish message: hello world #55
Publish message: hello world #56
Publish message: hello world #57
Publish message: hello world #58
Message arrived [inInTopic] 1
LED DESLIGADO
Publish message: hello world #59
Publish message: hello world #60
Publish message: hello world #61
Publish message: hello world #62
Publish message: hello world #63
Publish message: hello world #64
Message arrived [inInTopic] 2
LED LIGADO
Publish message: hello world #65
Publish message: hello world #66
Publish message: hello world #67
Publish message: hello world #68
Publish message: hello world #69
```

At the bottom of the terminal window, there are several status indicators: 'Auto-rolagem' (checked), 'Show timestamp' (unchecked), 'Nova-linha' (New Line), '115200 velocidade' (Baud Rate), and 'Delete a saída' (Delete a output).

A mensagem
recebida **não** começa
com “1”, portanto o
LED é **LIGADO**

Usando o Mosquitto para publicar dados

```
% mosquitto_pub -h test.mosquitto.org -t "inInTopic" -m  
"abracadabra-1"  
  
% mosquitto_pub -h test.mosquitto.org -t "inInTopic" -m  
"abracadabra-2"  
  
% mosquitto_pub -h test.mosquitto.org -t "inInTopic" -m  
"abracadabra-3"
```

Usando um terminal também é possível **publicar** em um tópico
Neste caso foi no tópico “**inInTopic**” (use o comando **mosquito_pub**)

Usando o Mosquitto para receber dados

```
% mosquitto_sub -h test.mosquitto.org -t "outOutTopic"  
hello world #1  
hello world #2  
hello world #3  
hello world #4  
. . .
```

Usando um terminal também é possível **assinar** um tópico
Neste caso foi o tópico “**outOutTopic**” (use o comando **mosquito_sub**)