[utf8]inputec

# Atividade Prática 2

LUIZ FELIPE FERNANDES (387827), MIGUEL SOUSA (382315), THIAGO FERRONATO (386026).

29 de novembro de 2022

## 1 Resumo

O presente relatório tem como objetivo compreender a utilização e comunicação dos protocolos MQTT, assim como exemplificar suas estruturas através de um projeto simples de conexão entre servidor e cliente. Criado na Década de 90, pela IBM e Eurotech, o Message Queuing Telemetry Transport é responsável por oferecer um baixo consumo de dados e outros recursos de software que envolvem a comunicação em rede. Muito utilizado em conectividade IoT e ligações "Machine to Machine", o protocolo mencionado consiste, na sua estrutura básica, em um modelo de Publish/Subscribe. Nesse contexto, um publicador será responsável por enviar uma mensagem a um tópico de destino (Payload), para que um assinante desse tópico possa recebe-la e acessa-la. Como não há um uma ligação direta entre "publicador" e "inscrito", todo o processo de gestão é realizado por um servidor de gerenciamento chamado Broker. Para o projeto desenvolvido fora configurado uma máquina virtual sobre a plataforma da Microsoft Azure, onde essa possuí o papel de entregar um endereço de IP público responsável pela construção do broker server em rede. Na sequência instalado o software livre Eclipse Mosquitto para configuração do message broker e seu respectivos subscribes. Em ajuste do cliente Mqtt, afim de publicar mensagens a partir de outros dispositivos foi necessário o uso do software "MQTT-Explorer" incumbido de permitir uma visão geral e estruturada do tópico postado. Dessa forma, com as estruturas cliente-servidor MQTT finalizada, torna-se objetivo, a inclusão de um dispositivo físico (comunicante ao serviço montado) encarregado de coletar os dados do ambiente e armazena-los em um banco de dados, local ao broker. Nesse sentido, a introdução do hardware foi realizada através do microcontrolador ESP8266, capaz de enviar os pacotes gerados pelo click de um botão-físico conectado aos terminais input da placa. A informação apresentada ao servidor clound (broker) é armazenada em um banco de dados juntamente com suas respectivas datas e horas. Ao final do processo é gerado um relatório gráfico HTML, apresentando o histórico de uso da botoeira acoplada à ESP32.

## 2 Modelagem da Aplicação

### 2.1 Hardware

- 1 x ESP 8266
- 1 x Notbook DELL Inspiron 14, processador i5 7200U, 8GB memória RAM, SSD SATA 128GB
- 1 x Protoboard
- 1 x Pushbutton(chave liga-desliga)
- 2 x jumpers macho-macho

- 1 x cabo Micro-USB para USB

### 2.2 Software

- Plataforma Microsoft Azure (Criação maquina virtual)
- Eclipse Mosquitto v3.1.1
- Arduino IDE 2.0.2
- MQTT-Explorer 0.4.0-beta1
- SQL Server (versão a informar)

## 3 Desenvolvimento da Aplicação

Para montagem e criação do projeto descrito no resumo anterior, com objetivo inicial de gravar as mensagens gerenciadas pelo protocolo MQTT ao banco de dados em questão, foi necessário (junto a plataforma da Microsoft Azure) a preparação de uma máquina virtual disposta em nuvem para a preparação de um IP-público. Esse, necessário para comunicação de "publicadores" e "inscritos" externos a rede local do broker Mosquitto. A mesma, por se tratar de uma assinatura gratuita, possuía configurações básicas de hardware, incluindo 1 GB de RAM disponível atrelada a 128GB de memoria secundária. O Mais importante trata-se do IP-público gerado; "191.233.31.141" ao qual será exibido mais a frente nos códigos desenvolvidos. Com a VM pronta e a versão 3.1.1 do Eclipse mosquito instalada, o próximo passo fora tornar a EXP8266 um comunicante e publicador de mensagens dentro do server broker. A partir disso, o código a baixo descreve o processo de programação utilizando-se da linguagem JAVA e o ambiente Arduíno IDE, afim de realizar a conexão da microcontroladora com a rede WiFI e também enviar a mensagem (Sinal de um pushbutton) ao servidor MQTT:

```
1  //Programa: comunica  o MQTT com ESP32
2  /* Headers */
3  #include <WiFi.h> /* Header para uso das funcionalidades de wi-fi do ESP32
       */
4  #include <PubSubClient.h>  /*  Header para uso da biblioteca PubSubClient */
5
6  /* Defines do MQTT */
7  /* IMPORTANTE: recomendamos fortemente alterar os nomes
8                 desses t picos. Caso contr rio, h  grandes
9                 chances de voc  enviar e receber mensagens de um ESP32
10                de outra pessoa.
11 */
12 /* T pico MQTT para recep  o de informa  es do broker MQTT para ESP32
       */
13 #define TOPICO_SUBSCRIBE "INCB_ESP32_recebe_informacao_Luiz"
14 /* T pico MQTT para envio de informa  es do ESP32 para broker MQTT */
15 #define TOPICO_PUBLISH   "TESTE"
16 /* id mqtt (para identifica  o de sess o) */
17 /* IMPORTANTE: este deve ser  nico  no broker (ou seja,
18                se um client MQTT tentar entrar com o mesmo
19                id de outro j  conectado ao broker, o broker
20                ir  fechar a conex o de um deles).
21 */
22 #define ID_MQTT  "INCB_Cliente_MQTT"
```

```
23 /*  Vari veis e constantes globais */
24 /* SSID / nome da rede WI-FI que deseja se conectar */
25 const char* SSID = "Luiz";
26 /*  Senha da rede WI-FI que deseja se conectar */
27 const char* PASSWORD = "pimpa123";
28
29 /* URL do broker MQTT que deseja utilizar */
30 const char* BROKER_MQTT = "191.233.31.141";
31 /* Porta do Broker MQTT */
32 int BROKER_PORT = 1883;
33
34 /* Vari veis e objetos globais */
35 WiFiClient espClient;
36 PubSubClient MQTT(espClient);
37
38 const int buttonPin = 12;
39 //Prototypes
40 void init_serial(void);
41 void init_wifi(void);
42 void init_mqtt(void);
43 void reconnect_wifi(void);
44 void mqtt_callback(char* topic, byte* payload, unsigned int length);
45 void verifica_conexoes_wifi_mqtt(void);
46 /*
47     Implementa  es das fun  es
48 */
49 void setup()
50 {
51   init_serial();
52   init_wifi();
53   init_mqtt();
54   pinMode(buttonPin, INPUT_PULLUP);
55 }
56 /* Fun  o: inicializa comunica  o serial com baudrate 115200 (para fins
       de monitorar no terminal serial
57             o que est  acontecendo.
58   Par metros: nenhum
59   Retorno: nenhum
60 */
61 void init_serial()
62 {
63   Serial.begin(115200);
64 }
65 /* Fun  o: inicializa e conecta-se na rede WI-FI desejada
66    Par metros: nenhum
67    Retorno: nenhum
68 */
69 void init_wifi(void)
70 {
71   delay(10);
72   Serial.println("------Conexao WI-FI------");
73   Serial.print("Conectando-se na rede: ");
74   Serial.println(SSID);
75   Serial.println("Aguarde");
76   reconnect_wifi();
77 }
78 /* Fun  o: inicializa par metros de conex o MQTT(endere o do
79           broker, porta e seta fun  o de callback)
```

```
80      Par metros: nenhum
81      Retorno: nenhum
82  */
83  void init_mqtt(void)
84  {
85    /* informa a qual broker e porta deve ser conectado */
86    MQTT.setServer(BROKER_MQTT, BROKER_PORT);
87    /* atribui fun   o de callback (fun   o chamada quando qualquer
          informa   o do
88        t pico subescrito chega) */
89    MQTT.setCallback(mqtt_callback);
90  }
91  /* Fun   o: fun   o de callback
92              esta fun   o     chamada toda vez que uma informa   o de
93              um dos t picos subescritos chega)
94      Par metros: nenhum
95      Retorno: nenhum
96   * */
97  void mqtt_callback(char* topic, byte* payload, unsigned int length)
98  {
99    String msg;
100
101   //obtem a string do payload recebido
102   for (int i = 0; i < length; i++)
103   {
104     char c = (char)payload[i];
105     msg += c;
106   }
107   Serial.print("[MQTT] Mensagem recebida: ");
108   Serial.println(msg);
109 }
110 /* Fun   o: reconecta-se ao broker MQTT (caso ainda n o esteja conectado
      ou em caso de a conex o cair)
111             em caso de sucesso na conex o ou reconex o, o subscribe dos
                t picos    refeito.
112     Par metros: nenhum
113     Retorno: nenhum
114 */
115 void reconnect_mqtt(void)
116 {
117   while (!MQTT.connected())
118   {
119     Serial.print("* Tentando se conectar ao Broker MQTT: ");
120     Serial.println(BROKER_MQTT);
121     if (MQTT.connect(ID_MQTT))
122     {
123       Serial.println("Conectado com sucesso ao broker MQTT!");
124       MQTT.subscribe(TOPICO_SUBSCRIBE);
125     }
126     else
127     {
128       Serial.println("Falha ao reconectar no broker.");
129       Serial.println("Havera nova tentatica de conexao em 2s");
130       delay(2000);
131     }
132   }
133 }
134 /* Fun   o: reconecta-se ao WiFi
```

```
135     Par metros: nenhum
136     Retorno: nenhum
137 */
138 void reconnect_wifi()
139 {
140   /* se j  est  conectado a rede WI-FI, nada   feito.
141      Caso contr rio, s o efetuadas tentativas de conex o */
142   if (WiFi.status() == WL_CONNECTED)
143     return;
144 
145   WiFi.begin(SSID, PASSWORD);
146 
147   while (WiFi.status() != WL_CONNECTED)
148   {
149     delay(100);
150     Serial.print(".");
151   }
152   Serial.println();
153   Serial.print("Conectado com sucesso na rede ");
154   Serial.print(SSID);
155   Serial.println("IP obtido: ");
156   Serial.println(WiFi.localIP());
157 }
158 /* Fun  o: verifica o estado das conex es WiFI e ao broker MQTT.
159            Em caso de desconex o (qualquer uma das duas), a conex o
160               refeita.
161     Par metros: nenhum
162     Retorno: nenhum
163 */
164 void verifica_conexoes_wifi_mqtt(void)
165 {
166   /* se n o h  conex o com o WiFI, a conex o   refeita */
167   reconnect_wifi();
168   /* se n o h  conex o com o Broker, a conex o   refeita */
169   if (!MQTT.connected())
170     reconnect_mqtt();
171 }
172 /* programa principal */
173 void loop()
174 {
175   /* garante funcionamento das conex es WiFi e ao broker MQTT */
176   verifica_conexoes_wifi_mqtt();
177 
178   if (digitalRead(buttonPin) == LOW) {
179     /* Envia frase ao broker MQTT */
180     MQTT.publish(TOPICO_PUBLISH, "JACSON MATTE");
181   }
182   /* keep-alive da comunica  o com broker MQTT */
183   MQTT.loop();
184   /* Agurda 1 segundo para pr ximo envio */
185   delay(1000);
186 }
```

Para o contexto de hardware, com objetivo de integrar uma resposta física a uma mensagem, foi utilizado um microcontrolador ESP8266 juntamente a um pushbutton de 4 terminais. Realizando ligações simples, utilizando jumpers e uma protoboard, foram ligados os contatos da chave táctil as portas 12 e Ground da placa de circuito ESP, assim como o esquema abaixo:

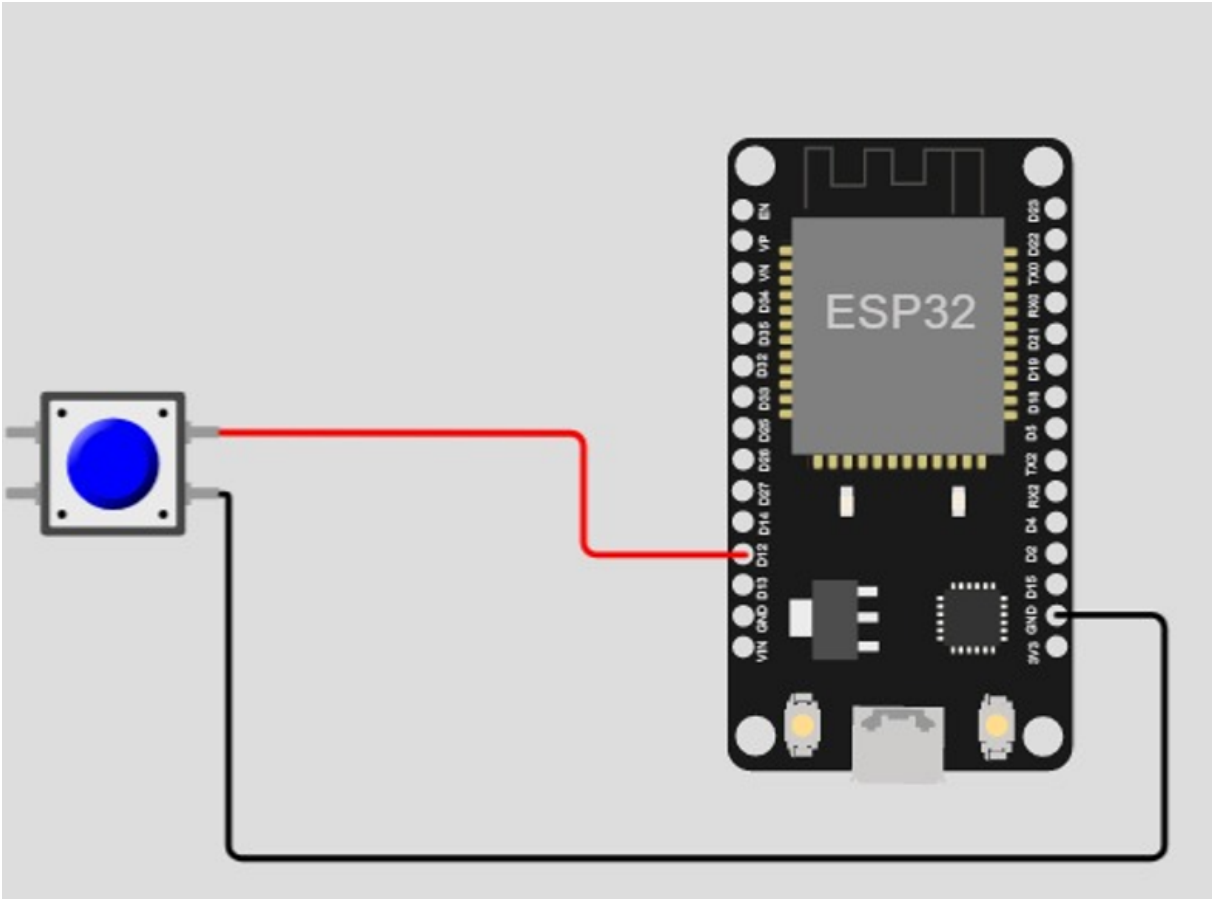


**Figura 1:**Esp 32.

Após a configuração do dispositivo publisher utilizando o microcontrolador, fora desenvolvido um código Python simples para comunicação das mensagens recebidas pelo broker "191.233.247.255" e o banco de dados SQL, como mostra o código abaixo:

```
1  import datetime
2  import random
3
4  from Data import Teste
5
6
7  import random
8  import time
9
10 from paho.mqtt import client as mqtt_client
11
12
13 broker = '191.233.247.255'
14 port = 1883
15 topic = "teste"
16 client_id = f'python-mqtt-{random.randint(0, 1000)}'
17 # python3.6
18 def connect_mqtt() -> mqtt_client:
19     def on_connect(client, userdata, flags, rc):
20         if rc == 0:
21             print("Connected to MQTT Broker!")
22         else:
23             print("Failed to connect, return code %d\n", rc)
24
```

```
    client = mqtt_client.Client(client_id)
    client.on_connect = on_connect
    client.connect(broker, port)
    return client


def subscribe(client: mqtt_client):
    def on_message(client, userdata, msg):
        print(f"Received `{msg.payload.decode()}` from `{msg.topic}` topic")
        t1 = Teste(
            dado=(msg.payload.decode())
        )
        t1.data = datetime.datetime.now()
        t1.save()

    client.subscribe(topic)
    client.on_message = on_message


def run():
    client = connect_mqtt()
    subscribe(client)
    client.loop_forever()


if __name__ == '__main__':
    run()
```

Para a finalização do projeto, com as informações coletadas pelo banco de dados, fora desenvolvido também análises gráficas utilizando a linguagem Python (através da biblioteca matplotlib). O "gerador de gráfico" basicamente busca as pesquisas da tabela "teste"do SQL e passa pra listas. Essas, serão usadas para compor eixo X e Y na geração do diagrama, onde assim o eixo X será o tempo (momento q foi cadastrado) e o eixo Y o valor que foi enviado.

```
import datetime
import random
from time import strptime, mktime
from datetime import datetime

import matplotlib #3.6.2
from peewee import (
    SqliteDatabase, Model, IntegerField, DateTimeField
)
import matplotlib.pyplot as plt
db = SqliteDatabase('Banco.db')


class BaseModel(Model):
    class Meta:
        database = db


class Teste(BaseModel):
    id = IntegerField()
    dado = IntegerField()
    data = DateTimeField()

class Total(BaseModel):
```

```
25      id = IntegerField()
26      dado = IntegerField()
27      data = DateTimeField()
28
29  for h in range(1,12):
30      Teste.insert(dado=(random.randint(1,10)),data=(datetime(datetime.now().
            year, datetime.now().month, datetime.now().day,h))).execute()
31  a= Teste.select(Teste.data).execute()
32  lista = [item for item in Teste.select().dicts()]
33
34  lista.sort(key = lambda x:x['data'])
35
36  dados=[]
37  datas = []
38  for x in lista:
39      dados.append(x['dado'])
40      datas.append(x['data'])
41      Total.insert(dado=x['dado'], data=x['data']).execute()
42
43  nome = str(datetime.now().day)+'_'+str(datetime.now().month)+'_'+str(
        datetime.now().year)+'.png'
44  dates = matplotlib.dates.date2num(datas)
45  plt.plot(datas,dados)
46  plt.title(nome)
47  plt.gcf().autofmt_xdate()
48
49  plt.savefig(nome)
50  plt.show()
51  print(nome)
52  Teste.delete().execute()
```

# 4 Resultados e Considerações

Com os testes feitos a partir do uso do MQQT e o Mosquitto, conseguimos obter resultados de como realizar uma conexão eficaz e rápida entre um publicador e um inscrito. através de uma conexão WIFI conseguimos enviar um sinal de um publisher para o subscriber, com um botão em um circuito com uma ESP32. Com essa aplicação conseguimos obter gráficos onde os mesmos mostram os valores que foram armazenados no banco de dados, esses valores são mostrados em um gráfico de linhas gerado através de um código em python com a biblioteca MatPlotLib.

# 5 Referências

@articlecitacao-exemplo, title = Protocolo MQTT: O Que é, Como Funciona e Vantagens, volume = 1, language = pt, number = 1, journal = Automacão Industrial, author = Santos, Guilherme, year = 2022, pages = 25,


UNOESC