

FICHA 1

- 1) O Eclipse Paho é plataforma grátis que usa MQTT (Message Queuing Telemetry Transport) para diversas plataformas e linguagens de programação.

```
[1] !pip install paho-mqtt

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting paho-mqtt
  Downloading paho-mqtt-1.6.1.tar.gz (99 kB)
    |----- 99.4/99.4 kB 3.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: paho-mqtt
  Building wheel for paho-mqtt (setup.py) ... done
  Created wheel for paho-mqtt: filename=paho_mqtt-1.6.1-py3-none-any.whl size=62135 sha256=8747de8edc2ca348e660856583daa6b29b2ebc0dd2be89d80c2e6f770abc
  Stored in directory: /root/.cache/pip/wheels/0f/90/29/db29bb8ddc98ec5f2363b959130c9ddbcf5cfd4a08b6184dd
Successfully built paho-mqtt
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.6.1
```

```
import paho.mqtt.client as mqtt

# Criar uma instância do cliente MQTT
client = mqtt.Client()

# Conectar ao broker MQTT
client.connect("test.mosquitto.org", 1883)

# Publicar uma mensagem em um tópico
client.publish("topico/de/interesse", "Mensagem de exemplo")

# Encerrar a conexão com o broker MQTT
client.disconnect()

# Criar uma instância do cliente MQTT
client = mqtt.Client()

# Conectar ao broker MQTT
client.connect("test.mosquitto.org", 1883)

# Publicar uma mensagem em um tópico
client.publish("topico/de/interesse", "Mensagem de exemplo")

# Encerrar a conexão com o broker MQTT
client.disconnect()
```

- 7) Os conceitos de publicador, assinante, broker e tópico estão diretamente relacionados ao protocolo MQTT, que é um protocolo de comunicação leve e eficiente para dispositivos IoT.

Assinante (Subscriber): é um cliente MQTT que se inscreve em um ou mais tópicos MQTT específicos para receber mensagens publicadas nesses tópicos.

Broker: é um intermediário de mensagens que recebe e encaminha as mensagens publicadas pelos publicadores para os assinantes inscritos nos respectivos tópicos.

broker também pode gerenciar a autenticação, autorização e segurança das conexões MQTT.

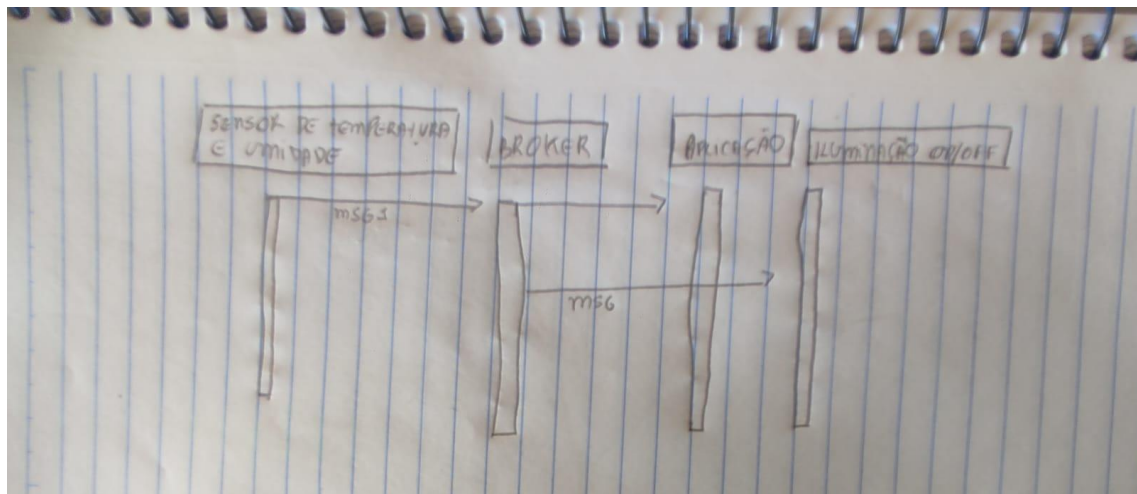
Tópico (Topic): é uma sequência hierárquica de caracteres que representa um canal de comunicação no MQTT.

8) A troca de mensagens entre os dois scripts não funcionará corretamente porque eles estão se comunicando em tópicos diferentes. O primeiro script está se inscrevendo no tópico "topic-test_A" e recebendo mensagens desse tópico, enquanto o segundo script está publicando mensagens no tópico "topic_test_B". E eles estão conectados em servidores diferentes

.

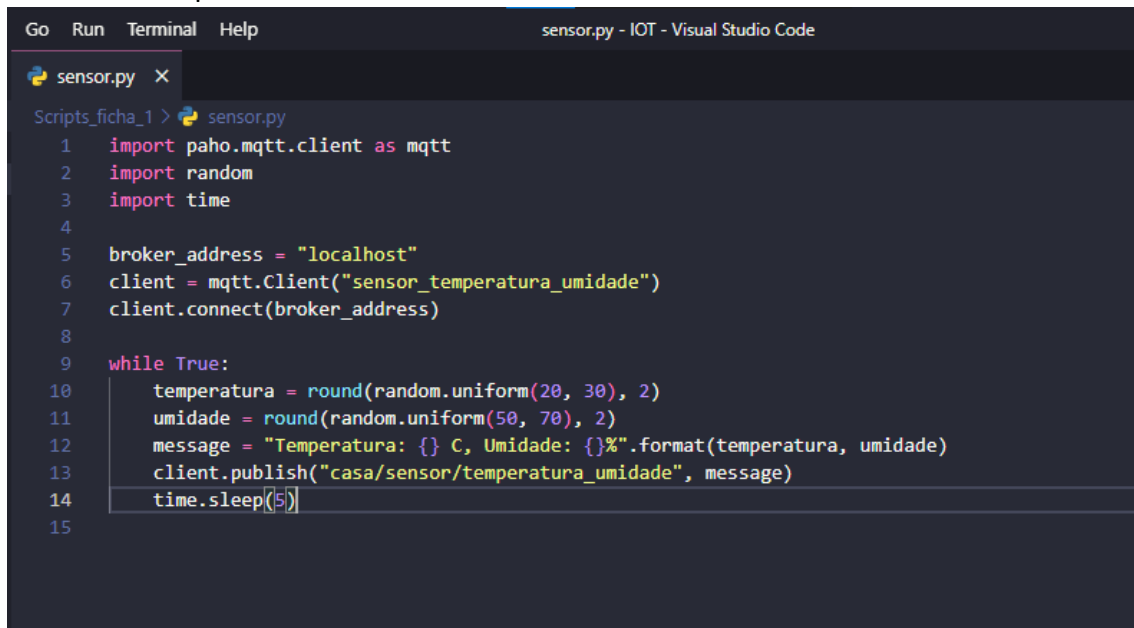
10-)

1-



2 –

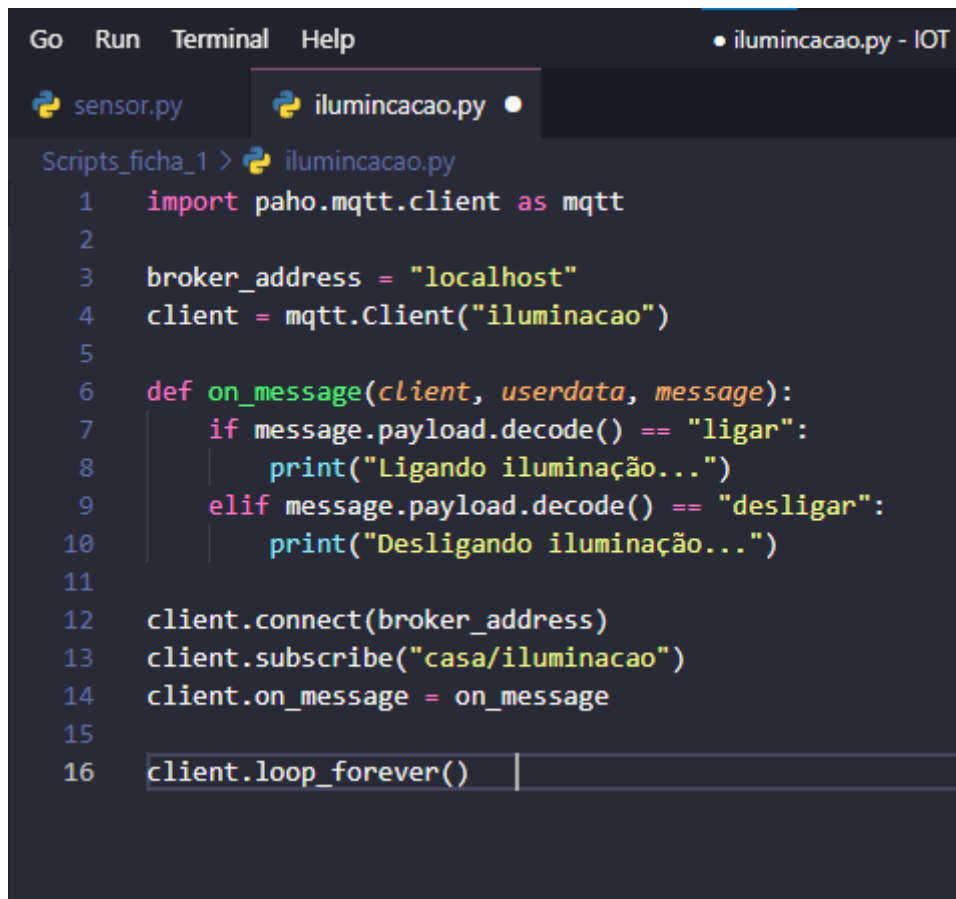
Sensor de temperatura e Umidade:



```
Go Run Terminal Help sensor.py - IOT - Visual Studio Code

sensor.py X
Scripts_ficha_1 > sensor.py
1 import paho.mqtt.client as mqtt
2 import random
3 import time
4
5 broker_address = "localhost"
6 client = mqtt.Client("sensor_temperatura_umidade")
7 client.connect(broker_address)
8
9 while True:
10     temperatura = round(random.uniform(20, 30), 2)
11     umidade = round(random.uniform(50, 70), 2)
12     message = "Temperatura: {} C, Umidade: {}%".format(temperatura, umidade)
13     client.publish("casa/sensor/temperatura_umidade", message)
14     time.sleep(5)
15
```

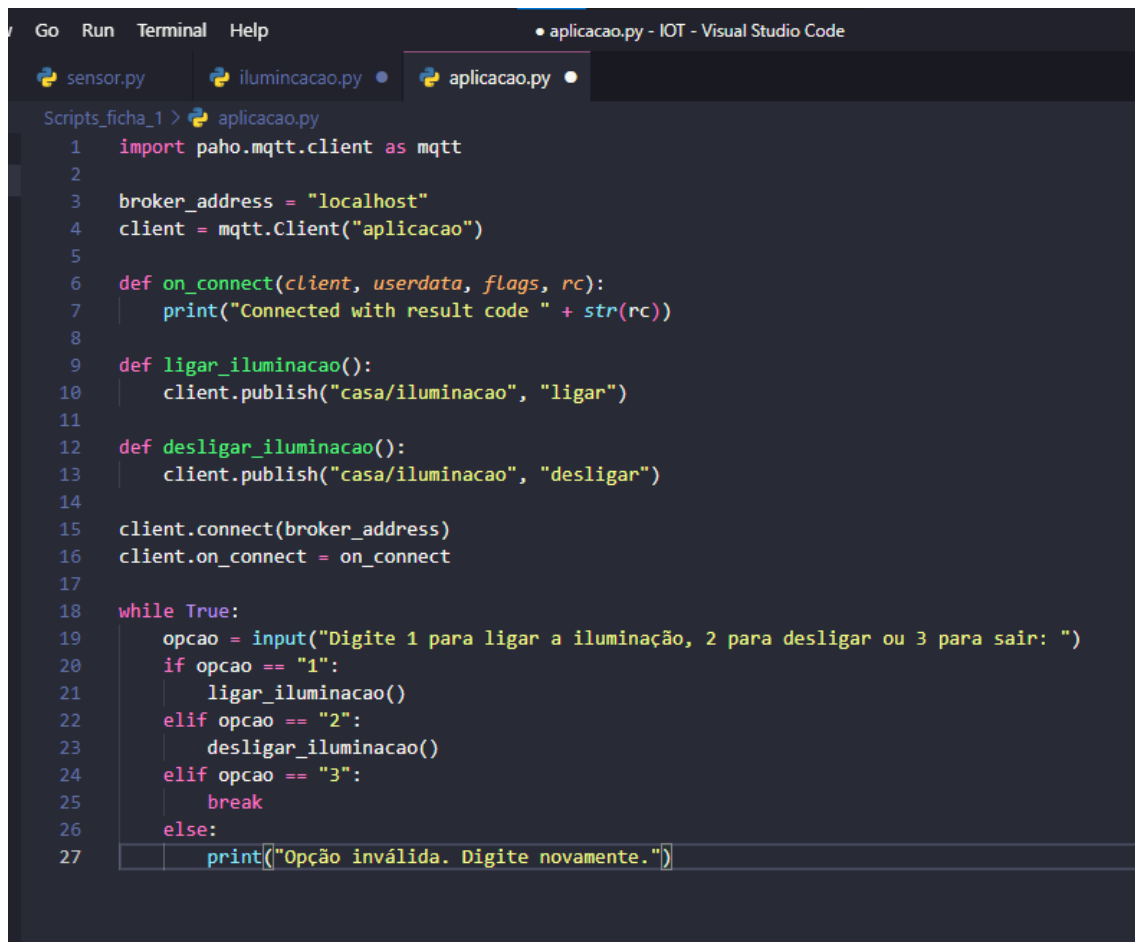
Iluminação:



The image shows a code editor window with a dark theme. The top menu bar includes 'Go', 'Run', 'Terminal', and 'Help'. The title bar on the right says 'ilumincacao.py - IOT'. Below the menu bar, there are two tabs: 'sensor.py' and 'ilumincacao.py', with the latter being the active tab. The editor content shows a Python script for an MQTT client. The script imports 'paho.mqtt.client' as 'mqtt', sets 'broker_address' to 'localhost', and creates a 'client' object. It defines an 'on_message' function that checks for 'ligar' and 'desligar' commands in the message payload and prints corresponding status messages. The client connects to the broker, subscribes to the 'casa/iluminacao' topic, and enters a loop to receive messages.

```
Go Run Terminal Help • ilumincacao.py - IOT
sensor.py ilumincacao.py
Scripts_ficha_1 > ilumincacao.py
1 import paho.mqtt.client as mqtt
2
3 broker_address = "localhost"
4 client = mqtt.Client("iluminacao")
5
6 def on_message(client, userdata, message):
7     if message.payload.decode() == "ligar":
8         print("Ligando iluminação...")
9     elif message.payload.decode() == "desligar":
10        print("Desligando iluminação...")
11
12 client.connect(broker_address)
13 client.subscribe("casa/iluminacao")
14 client.on_message = on_message
15
16 client.loop_forever()
```

Aplicação:

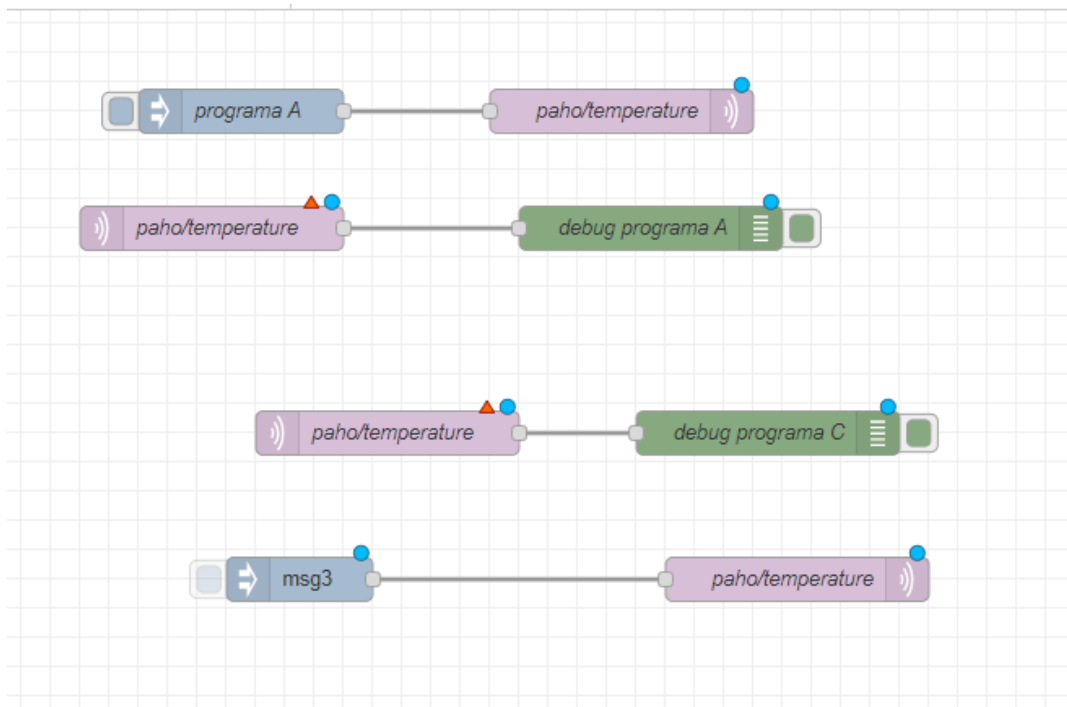


```
Go Run Terminal Help • aplicacao.py - IOT - Visual Studio Code

sensor.py iluminacao.py aplicacao.py

Scripts_ficha_1 > aplicacao.py
1 import paho.mqtt.client as mqtt
2
3 broker_address = "localhost"
4 client = mqtt.Client("aplicacao")
5
6 def on_connect(client, userdata, flags, rc):
7     print("Connected with result code " + str(rc))
8
9 def ligar_iluminacao():
10     client.publish("casa/iluminacao", "ligar")
11
12 def desligar_iluminacao():
13     client.publish("casa/iluminacao", "desligar")
14
15 client.connect(broker_address)
16 client.on_connect = on_connect
17
18 while True:
19     opcao = input("Digite 1 para ligar a iluminação, 2 para desligar ou 3 para sair: ")
20     if opcao == "1":
21         ligar_iluminacao()
22     elif opcao == "2":
23         desligar_iluminacao()
24     elif opcao == "3":
25         break
26     else:
27         print("Opção inválida. Digite novamente.")
```

12) Realizar o exercício 9 utilizando o Node-RED. Considere que os “Programas A” e “C” anteriormente implementados em Python, agora devem ser implementados no NodeRED (Dica: siga os procedimentos em <https://cookbook.nodered.org/#mqtt>). O “Programa B” continua a ser em Python



Programa B em python

```
1 import paho.mqtt.client as mqtt
2
3 client = mqtt.Client()
4
5 client.connect("broker.mqtt-dashboard.com", 1883, 60)
6 client.publish("paho/temperature", "Programa B")

<paho.mqtt.client.MQTTMessageInfo at 0x7fb02c0715e0>
```

```
1 import paho.mqtt.client as mqtt
2
3 # The callback for when the client receives a CONNACK response from the server.
4 def on_connect(client, userdata, flags, rc):
5     print("Connected with result code "+str(rc))
6
7     # Subscribing in on_connect() means that if we lose the connection and
8     # reconnect then subscriptions will be renewed.
9     client.subscribe("paho/temperature")
10
11 # The callback for when a PUBLISH message is received from the server.
12 def on_message(client, userdata, msg):
13     print(msg.topic+" "+str(msg.payload))
14
15 client = mqtt.Client()
16 client.on_connect = on_connect
17 client.on_message = on_message
18
19 client.connect("broker.mqtt-dashboard.com", 1883, 60)
20
21 # Blocking call that processes network traffic, dispatches callbacks and
22 # handles reconnecting.
23 # Other loop*() functions are available that give a threaded interface and a
24 # manual interface.
25 client.loop_forever()
```

Connected with result code 0
paho/temperature b'Programa A'
paho/temperature b'Programa B'

13) realizar o exercício 10 utilizando o Node-RED.

