This Python script subscribes to an MQTT topic, receives messages, processes them, and stores the data in a **MongoDB database**. Here’s a breakdown of how it works:

## ****1. Importing Required Modules****

python

CopyEdit

import os

import pymongo

import asyncio

import random

from paho.mqtt import client as mqtt\_client

import time

from datetime import datetime

* os: Provides functions to interact with the operating system.
* pymongo: Used to interact with **MongoDB**.
* asyncio: Supports asynchronous programming (though it's not used in this script).
* random: Generates a random **client ID** for MQTT.
* paho.mqtt.client: MQTT client for communication with an MQTT broker.
* time: Provides functions for handling time-related tasks.
* datetime: Helps timestamp each received message.

## ****2. Defining MQTT and Database Configuration****

python

CopyEdit

broker = 'IP address' # MQTT Broker address

dbclient = pymongo.MongoClient("mongodb://") # Connect to MongoDB

db = dbclient["DBClient"] # Select Database

col = db["Collection name"] # Select Collection (Table)

port = 1883 # MQTT Default Port

topic = 'Topic' # MQTT Topic to Subscribe to

mqttuser = 'username' # MQTT Username

password = 'Password' # MQTT Password

client\_id = f'python-mqtt-{random.randint(0,100)}' # Generate a unique client ID

* **MQTT Settings:**
  + The script connects to an **MQTT broker** (IP address required).
  + Uses a **randomized client ID** for MQTT communication.
  + Connects to a **specified topic** to receive messages.
* **MongoDB Settings:**
  + Connects to a **MongoDB database**.
  + Uses a specified **database (**DBClient**)** and **collection (**Collection name**)** to store data.

## ****3. MQTT Connection Function****

python

CopyEdit

def connect\_mqtt():

def on\_connect(client, userdata, flags, rc):

if rc == 0:

print("Ready to receive data "+"\n")

else:

print("connection failed, return code %d\n", rc)

client = mqtt\_client.Client(client\_id) # Create MQTT Client

client.username\_pw\_set(mqttuser, password) # Set Username and Password

client.on\_connect = on\_connect # Assign Callback

client.connect(broker, port) # Connect to MQTT Broker

return client

* connect\_mqtt() function:
  + Creates an **MQTT client**.
  + Assigns **username and password** for authentication.
  + Defines an **on\_connect callback** to check if the connection was successful.
  + Returns the connected **MQTT client**.

## ****4. Subscribing to an MQTT Topic****

python

CopyEdit

def subscribe(client: mqtt\_client):

def on\_message(client, userdata , msg):

data1 = f"{msg.payload.decode('utf8')}"

print(data1)

data2 = str(data1)

res = data2.find("|")

if res == -1:

index = data1.find("-")

device\_id = data2[0:index]

else:

ind = data1.find("-")

index = data1.find("|")

device\_id = data2[0:ind]

consumption = data2[index+1:]

con= str(consumption)

pub = float(con) #if the db is logging data in String comment this line

print("ID = ",device\_id)

print("consumption\_value = ",pub) # If storing as string, use "con" instead of "pub"

mydict = { "device\_id": device\_id, "consumption":pub, "date": datetime.now()} # If storing as string, use "con"

x = col.insert\_one(mydict)

print(datetime.now())

print("Record Successfully Inserted")

client.subscribe(topic)

client.on\_message = on\_message

* subscribe(client) function:
  + Subscribes to the **specified MQTT topic**.
  + When a message is received, the **on\_message()** callback is triggered.
* **Processing the Received MQTT Message:**
  + **Message format is expected to be**: DeviceID-ConsumptionValue **or** DeviceID-ConsumptionValue|ExtraData
  + Extracts:
    - device\_id: The part before the **first** -
    - consumption: The value after the separator (| or -)
  + Converts consumption into a float for numerical storage in MongoDB.
  + Creates a **dictionary (**mydict**)** with:
    - device\_id
    - consumption
    - date (current timestamp)
  + **Inserts data into MongoDB**.

## ****5. Running the MQTT Client****

python

CopyEdit

def run():

client = connect\_mqtt() # Connect to MQTT Broker

subscribe(client) # Subscribe to Topic

client.loop\_forever() # Keep Listening for Messages

* run() **function**:
  + Establishes an MQTT connection.
  + Subscribes to the topic.
  + Runs loop\_forever() to keep listening indefinitely.

## ****6. Script Execution****

python

CopyEdit

if \_\_name\_\_=='\_\_main\_\_':

run()

* This ensures that run() is executed **only when the script is run directly**.