**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HCM**



**CHUYÊN ĐỀ IOT**

**BÀI BÁO CÁO 10**

**NHÓM 2**

Giảng viên : **CAO VĂN KIÊN**

Sinh viên :

Phạm Gia Bách 20026331

Trần Công Hòa 20017691

Doãn Đình Khánh 20054731

**TP.HCM – 2023**

**Mức độ 3 (10 điểm).**

# Yêu cầu đề bài:

* Có topic để gửi **từng** dữ liệu nhiệt độ, độ ẩm, 2 giá trị trạng thái 2 LED lên server; Topic để gửi **toàn bộ** thông tin lên server theo định dạng json và form-urlencoded.
* Có topic đọc **từng** dữ liệu nhiệt độ, độ ẩm, 2 giá trị trạng thái 2 LED từ server theo định dạng json **và** form-urlencoded. Có hỗ trợ topic đọc toàn bộ dữ liệu từ server theo cả 2 định dạng.
* Dữ liệu gửi lên server phải có thông tin \_id, thời gian, tên thiết bị gửi lên.

**\*Lưu ý:** Topic gửi từng dữ liệu lên cũng phải kích hoạt chế độ Subscribe của topic đọc toàn bộ dữ liệu.

**Nâng cao (+3 Đ):**

Xây dựng giao diện web quản lý các thiết bị trong MQTT broker bao gồm trạng thái hiện tại (Connected/ Disconnected), dữ liệu cuối cùng gửi lên, đồ thị của từng dữ liệu.

**Gợi ý:** Google, ChatGPT, Google Bard,...

# Sơ đồ cấm dây

A circuit board with wires and wires

Description automatically generated

# A white background with black lines Description automatically generatedA white rectangular sign with many lines connected to it Description automatically generatedCấu hình EMQX:

Chương trình trong **Funtion:**

FOREACH  first(jq('def rem\_first:

                 if length > 2 then del(.[0]) else . end;

              def rem\_last:

                 if length > 1 then del(.[-1]) else . end;

              .id as $id |

              .time as $time |

              .name as $name |

              .data1 as $data1 |

              .data2 as $data2 |

              .data3 as $data3 |

              .data4 as $data4 |

                {$name,$id,$time,$data4,$data3, $data2, $data1}',

             payload)) as dataf,

          map\_get('data1',first(jq('def rem\_first:

                 if length > 2 then del(.[0]) else . end;

              def rem\_last:

                 if length > 1 then del(.[-1]) else . end;

              .data1 as $data1 |

                {$data1}',

             payload))) as data1l,

          map\_get('data2',first(jq('def rem\_first:

                 if length > 2 then del(.[0]) else . end;

              def rem\_last:

                 if length > 1 then del(.[-1]) else . end;

              .data2 as $data2 |

                {$data2}',

             payload))) as data2l,

          map\_get('data3',first(jq('def rem\_first:

                 if length > 2 then del(.[0]) else . end;

              def rem\_last:

                 if length > 1 then del(.[-1]) else . end;

              .data3 as $data3 |

                {$data3}',

             payload))) as data3l,

          jq('def rem\_first:

                 if length > 2 then del(.[0]) else . end;

              def rem\_last:

                 if length > 1 then del(.[-1]) else . end;

              .data4 as $data4 |

                $data4',

             payload) as data4l

FROM

  "Loo/publish/json", "Loo/publish/json/data1", "Loo/publish/json/data2", "Loo/publish/json/data3","Loo/publish/json/data4"

# Mã nguồn chương trình:

**Code API:**

import uvicorn

from typing import Optional

import paho.mqtt.client as mqtt

from urllib import parse

from fastapi import FastAPI

from pydantic import BaseModel

import pymongo

import json

from fastapi import Form

app = FastAPI()

myclient = pymongo.MongoClient("mongodb+srv://Loo:24@loo.isuovt2.mongodb.net/?retryWrites=true&w=majority")

mydb = myclient["DataBase"]#tao database

mycol = mydb["LooData"]#tao colection moi

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

    print("Connected with Result Code {}".format(*rc*))

def on\_disconnect(*client*, *userdata*, *rc*):

    print("Disconnected from Broker")

client = mqtt.Client("Loo\_03")

client.on\_connect = on\_connect

client.on\_disconnect = on\_disconnect

client.username\_pw\_set(*username*="Loo3", *password*="1")

client.connect("192.168.1.3", 1883, 60)

class Item(BaseModel):

    id: Optional[int]

    time: Optional[str]

    name: Optional[str]

    data1: Optional[float]

    data2: Optional[float]

    data3: Optional[float]

    data4: Optional[float]

def MQTTJson(*data*:dict):

*dict* = {

        "id": *data*["id"],

        "time": *data*["time"],

        "name": *data*["name"],

        "data1": *data*["data1"],

        "data2": *data*["data2"],

        "data3": *data*["data3"],

        "data4": *data*["data4"]

    }

    json\_data = json.dumps(*dict*)

    print(json\_data)

    client.publish("Loo/subscribed/json/data1", *data*["data1"])

    client.publish("Loo/subscribed/json/data2", *data*["data2"])

    client.publish("Loo/subscribed/json/data3", *data*["data3"])

    client.publish("Loo/subscribed/json/data4", *data*["data4"])

    client.publish("Loo/subscribed/json", json\_data)

def MQTTUrl(*data*:dict):

*dict* = {

        "id": *data*["id"],

        "time": *data*["time"],

        "name": *data*["name"],

        "data1": *data*["data1"],

        "data2": *data*["data2"],

        "data3": *data*["data3"],

        "data4": *data*["data4"]

    }

    url\_data = parse.urlencode(*dict*)

    print(url\_data)

    client.publish("Loo/subscribed/url", url\_data)

    client.publish("Loo/subscribed/url/data1", *data*["data1"])

    client.publish("Loo/subscribed/url/data2", *data*["data2"])

    client.publish("Loo/subscribed/url/data3", *data*["data3"])

    client.publish("Loo/subscribed/url/data4", *data*["data4"])

@app.post("/update\_post")

async def update\_data\_post(*item*: Item):

    mydict = {

        "id": *item*.id,

        "time": *item*.time,

        "name": *item*.name,

        "data1": *item*.data1,

        "data2": *item*.data2,

        "data3": *item*.data3,

        "data4": *item*.data4,

    }

    print("update\_post: {}".format(mydict))

    mycol.insert\_one(mydict)

    x = mycol.find().sort("\_id", -1).limit(1)

    print("get: {}".format(x))

    data\_return  = {

        "id": x[0]["id"],

        "time": x[0]["time"],

        "name": x[0]["name"],

        "data1": x[0]["data1"],

        "data2": x[0]["data2"],

        "data3": x[0]["data3"],

        "data4": x[0]["data4"]}

    MQTTUrl(data\_return)

    return {"ok"}

@app.post("/update/data1")

async def update\_data(*id*: Optional[int] = Form(...), *time*: Optional[str] = Form(...), *name*: Optional[str] = Form(...),*data1*: Optional[float]= Form(...)):

*dict* = {

        "id": *id*,

        "time": *time*,

        "name": *name*,

        "data1": *data1*,

        "data2": None,

        "data3": None,

        "data4": None

    }

    mycol.insert\_one(*dict*)

    x = mycol.find().sort("\_id", -1).limit(1)

    print("get: {}".format(x))

    data\_return  = {

        "id": x[0]["id"],

        "time": x[0]["time"],

        "name": x[0]["name"],

        "data1": x[0]["data1"],

        "data2": x[0]["data2"],

        "data3": x[0]["data3"],

        "data4": x[0]["data4"],

    }

    MQTTJson(data\_return)

    MQTTUrl(data\_return)

    return {"ok"}

@app.post("/update/data2")

async def update\_data(*id*: Optional[int] = Form(...), *time*: Optional[str] = Form(...), *name*: Optional[str] = Form(...),*data2*: Optional[float]= Form(...)):

*dict* = {

        "id": *id*,

        "time": *time*,

        "name": *name*,

        "data1": None,

        "data2": *data2*,

        "data3": None,

        "data4": None

    }

    mycol.insert\_one(*dict*)

    x = mycol.find().sort("\_id", -1).limit(1)

    print("get: {}".format(x))

    data\_return  = {

        "id": x[0]["id"],

        "time": x[0]["time"],

        "name": x[0]["name"],

        "data1": x[0]["data1"],

        "data2": x[0]["data2"],

        "data3": x[0]["data3"],

        "data4": x[0]["data4"],

    }

    MQTTJson(data\_return)

    MQTTUrl(data\_return)

    return {"ok"}

@app.post("/update/data3")

async def update\_data(*id*: Optional[int] = Form(...), *time*: Optional[str] = Form(...), *name*: Optional[str] = Form(...),*data3*: Optional[float] = Form(...)):

*dict* = {

        "id": *id*,

        "time": *time*,

        "name": *name*,

        "data1": None,

        "data2": None,

        "data3": *data3*,

        "data4": None

    }

    mycol.insert\_one(*dict*)

    x = mycol.find().sort("\_id", -1).limit(1)

    print("get: {}".format(x))

    data\_return  = {

        "id": x[0]["id"],

        "time": x[0]["time"],

        "name": x[0]["name"],

        "data1": x[0]["data1"],

        "data2": x[0]["data2"],

        "data3": x[0]["data3"],

        "data4": x[0]["data4"],

    }

    MQTTJson(data\_return)

    MQTTUrl(data\_return)

    return {"ok"}

@app.post("/update/data4")

async def update\_data(*id*: Optional[int] = Form(...), *time*: Optional[str] = Form(...), *name*: Optional[str] = Form(...),*data4*: float= Form(...)):

*dict* = {

        "id": *id*,

        "time": *time*,

        "name": *name*,

        "data1": None,

        "data2": None,

        "data3": None,

        "data4": *data4*

    }

    mycol.insert\_one(*dict*)

    x = mycol.find().sort("\_id", -1).limit(1)

    print("get: {}".format(x))

    data\_return  = {

        "id": x[0]["id"],

        "time": x[0]["time"],

        "name": x[0]["name"],

        "data1": x[0]["data1"],

        "data2": x[0]["data2"],

        "data3": x[0]["data3"],

        "data4": x[0]["data4"],

    }

    MQTTJson(data\_return)

    MQTTUrl(data\_return)

    return {"ok"}

@app.post("/update")

async def update\_data(*id*: int = Form(...), *time*: str = Form(...), *name*: str = Form(...),*data1*: float= Form(...),

*data2*: float= Form(...), *data3*: float= Form(...), *data4*: float= Form(...)):

*dict* = {

        "id": *id*,

        "time": *time*,

        "name": *name*,

        "data1": *data1*,

        "data2": *data2*,

        "data3": *data3*,

        "data4": *data4*

    }

    mycol.insert\_one(*dict*)

    x = mycol.find().sort("\_id", -1).limit(1)

    print("get: {}".format(x))

    data\_return  = {

        "id": x[0]["id"],

        "time": x[0]["time"],

        "name": x[0]["name"],

        "data1": x[0]["data1"],

        "data2": x[0]["data2"],

        "data3": x[0]["data3"],

        "data4": x[0]["data4"],

    }

    MQTTJson(data\_return)

    MQTTUrl(data\_return)

    return {"ok"}

if \_\_name\_\_ == "\_\_main\_\_":

    uvicorn.run(app, *host*="0.0.0.0", *port*=8000)

**Code gửi với định dạng json:**

import paho.mqtt.client as mqtt

from time import sleep

from gpiozero import LED

import Adafruit\_DHT

import json

import datetime

import platform

sensor = Adafruit\_DHT.DHT11

gpio = 5

led1 = LED(14)

led2 = LED(15)

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

    print("Connected with Result Code {}".format(*rc*))

def on\_disconnect(*client*, *userdata*, *rc*):

    print("Disconnected from Broker")

client = mqtt.Client("Loo\_01")

client.on\_connect = on\_connect

client.on\_disconnect = on\_disconnect

client.username\_pw\_set(*username*="Loo1", *password*="1")

client.connect("192.168.1.3", 1883, 60)

def mqttJson(*id*,*data1*, *data2*, *data3*, *data4*):

    current\_time = datetime.datetime.now()

    formatted\_time = current\_time.strftime('%Y-%m-%d %H:%M:%S')

    device\_name = platform.node()

    data = {

        "id": *id*,

        "time": formatted\_time,

        "name": device\_name,

        "data1": *data1*,

        "data2": *data2*,

        "data3": *data3*,

        "data4": *data4*

    }

    json\_data = json.dumps(data)

    print(json\_data)

    client.publish("Loo/publish/json", json\_data)

i = 0

status = 0

while True:

    humi, temp = Adafruit\_DHT.read\_retry(sensor, gpio)

    if status == 0:

        led1.on()

        led2.on()

        status+=1

    else:

        led1.off()

        led2.off()

        status-=1

    mqttJson(i, humi,temp, status, status)

    i+=1

    sleep(10)

**Code gửi với định dạng Url:**

import paho.mqtt.client as mqtt

from urllib import parse

from time import sleep

import Adafruit\_DHT

from gpiozero import LED

import datetime

import platform

sensor = Adafruit\_DHT.DHT11

gpio = 5

led1 = LED(14)

led2 = LED(15)

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

    print("Connected with Result Code {}".format(*rc*))

def on\_disconnect(*client*, *userdata*, *rc*):

    print("Disconnected from Broker")

client = mqtt.Client("Loo\_01")

client.on\_connect = on\_connect

client.on\_disconnect = on\_disconnect

client.username\_pw\_set(*username*="Loo1", *password*="1")

client.connect("192.168.1.3", 1883, 60)

def mqttUrl(*id*,*data1*, *data2*, *data3*, *data4*):

    current\_time = datetime.datetime.now()

    formatted\_time = current\_time.strftime('%Y-%m-%d %H:%M:%S')

    device\_name = platform.node()

    data = {

        "id": *id*,

        "time": formatted\_time,

        "name": device\_name,

        "data1": *data1*,

        "data2": *data2*,

        "data3": *data3*,

        "data4": *data4*

    }

    json\_data = parse.urlencode(data)

    print(json\_data)

    client.publish("Loo/publish/url", json\_data)

i= 0

status = 0

while True:

    humi, temp = Adafruit\_DHT.read\_retry(sensor, gpio)

    if status == 0:

        led1.on()

        led2.on()

        status+=1

    else:

        led1.off()

        led2.off()

        status-=1

    mqttUrl(i, humi,temp, status, status)

    i+=1

    sleep(10)

# Video minh chứng:

**Link:** [**https://youtu.be/PfKxygFSIFk**](https://youtu.be/PfKxygFSIFk)