**BỘ CÔNG THƯƠNG**

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



**CHUYÊN ĐỀ IOT**

**BÀI BÁO CÁO THỰC HÀNH BUỔI 10**

**NHÓM 6**

Giảng viên : **PHẠM QUANG TRÍ**

Sinh viên :

Hoàng Đỗ Duy Hải- 21130351

Đoàn Mạnh Tú - 21007261

Dương Minh Toàn - 21069251

Yêu cầu 1:

**Mục lục:**

[I.Yêu cầu 2: Sơ đồ nguyên lý. 3](#_Toc179408580)

[II.Yêu cầu 3: Lưu đồ giải thuật. 5](#_Toc179408581)

[III.Yêu cầu 4: Mã nguồn chương trình. 7](#_Toc179408583)

[IV.Yêu cầu 5: Giải thích mã nguồn. 15](#_Toc179408584)

[V.Yêu cầu 6: Minh chứng kết quả thực nghiệm. 22](#_Toc179408585)

[VI.Yêu cầu 7: Video.: 25](#_Toc179408586)

**Bảng phân nhiệm vụ từng thành viên****:**

|  |  |
| --- | --- |
| **Họ và tên** | **Nhiệm vụ** |
| Hoàng Đỗ Duy Hải | Vẽ sơ đồ nguyên lý, làm code |
| Đoàn Mạnh Tú | Làm word, lưu đồ giải thuật |
| Dương Minh Toàn | Làm video |

**Yêu cầu bài: 10 điểm**

Hệ thống IOT bao gồm 4 thành phần: mô-đun Rasberry, mô đun “IOT Gateway”, mô-đun”IOT Node” và giao diện Web Node-RED.

- Mô-đun “IOT Gateway”: Rasberry Pi có vai trò “Master”, sử dụng giao tiếp LoRa các mô-đun “IOT Node” và giao tiếp TCP/UDP với mô-đun Rasberry “Slave”.

+ Đọc trạng thái bật/tắt của 2 công tắc trên mô-đun “IOT Node” và lưu trữ thông tin này lên Server.

+ Đọc dữ liệu đã lưu trữ trên Server (dữ liệu trạng thại của các công tắc) để điều khiển trạng thái sáng/tắt của đèn chiếu sáng và đèn cảnh báo trên mô-đun “IOT Node”.

+ Đọc dữ liệu nhiệt độ, độ ẩm từ mô-đun Rasberry “Slave” và lưu trữ thông tin này lên Server

+ Đọc dữ liệu đã lưu trữ trên Server để điều khiển trạng thái sáng/tắt của các LED đơn trên mô-đun Rasberry “Slave”.

+ Tùy chọn sử dụng giao thức HTTP hoặc MQTT để trao đổi dữ liệu với Server.

- Mô-đun “IOT Node”: Kết nối phần cứng với 3 công tắc và 2 đèn (đèn chiếu sáng và đèn cảnh báo). Sử dụng giao tiếp LoRa với mô-đun “IOT Gateway”.

- Mô-đun Rasberry “Slave”: Rasberry Pi có vai trò “Slave”, giao tiếp TCP/UDP với mô-đun “IOT Gateway”. Kết nối phần cứng với cảm biến DHT11 và 3 LED đơn. Đọc dữ liệu nhiệt độ, độ ẩm từ cảm biến DHT11 vầ guuiwr thông tin này cho mô-đun “IOT Gateway”.

- Giao diện Web Node-RED: Điều khiển trạng thái sáng/tắt của 3 LED đơn trên mô-đun Rasberry “Slave” bằng các nút nhấn ở giao diện. Giao dienj phải hiển thị đầy đủ thông tin trạng thái của tất cả các thiết bị trong hệ thống. Hệ thống IOT có chức năng hoạt động tự động hoặc bằng tay.

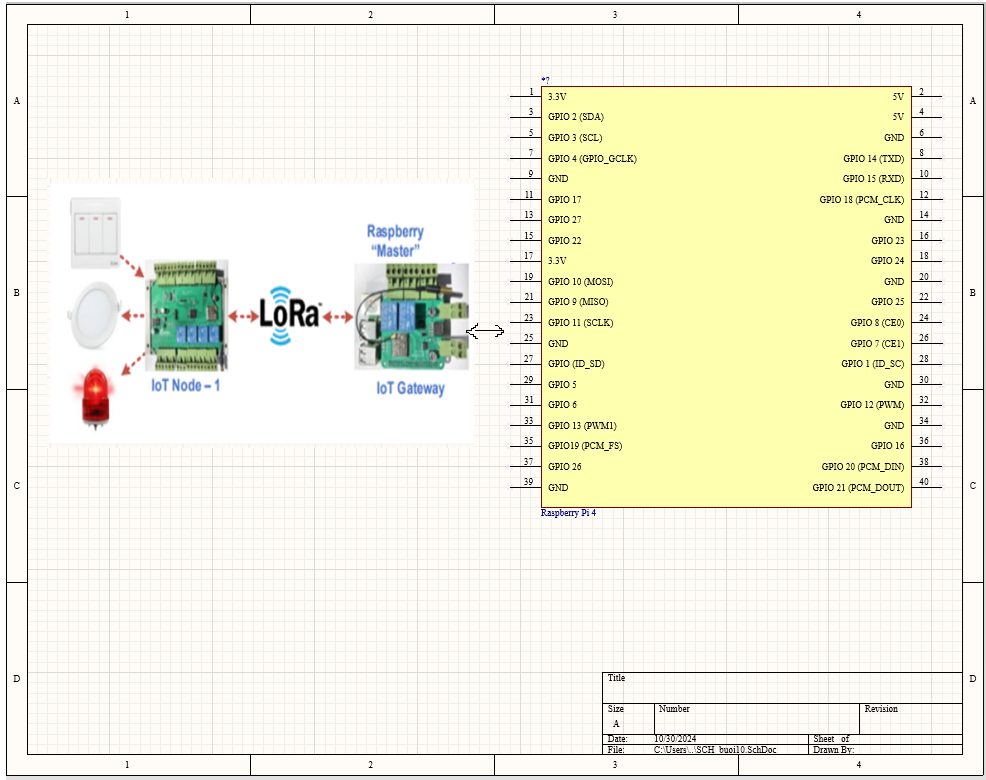
+ Chế độ tự động: Cho phép bật/tắt đèn(đèn chiếu sáng và đèn cảnh báo) theo thời gian, thời điểm bật/tắt đèn phải được cài đặt từ giao diện.

+Chế độ bằng tay: Người dung điều khiển bật/tắt đèn(đèn chiếu sáng và đèn cảnh báo) bằng các công tắc trên mô-đun “IOT Node”.

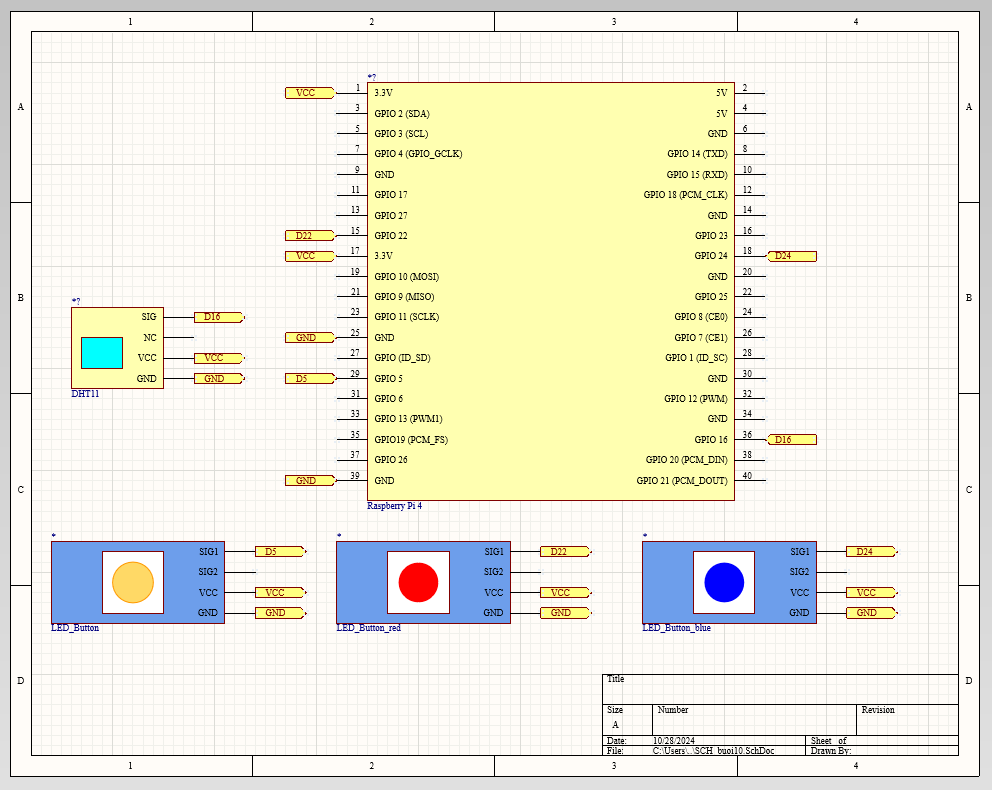
+Tùy chọn sử dụng giao thức HTTP hoặc MQTT để trao đổi dữ liệu với Server.

# **I.Yêu cầu 2: Sơ đồ nguyên lý**

Master:

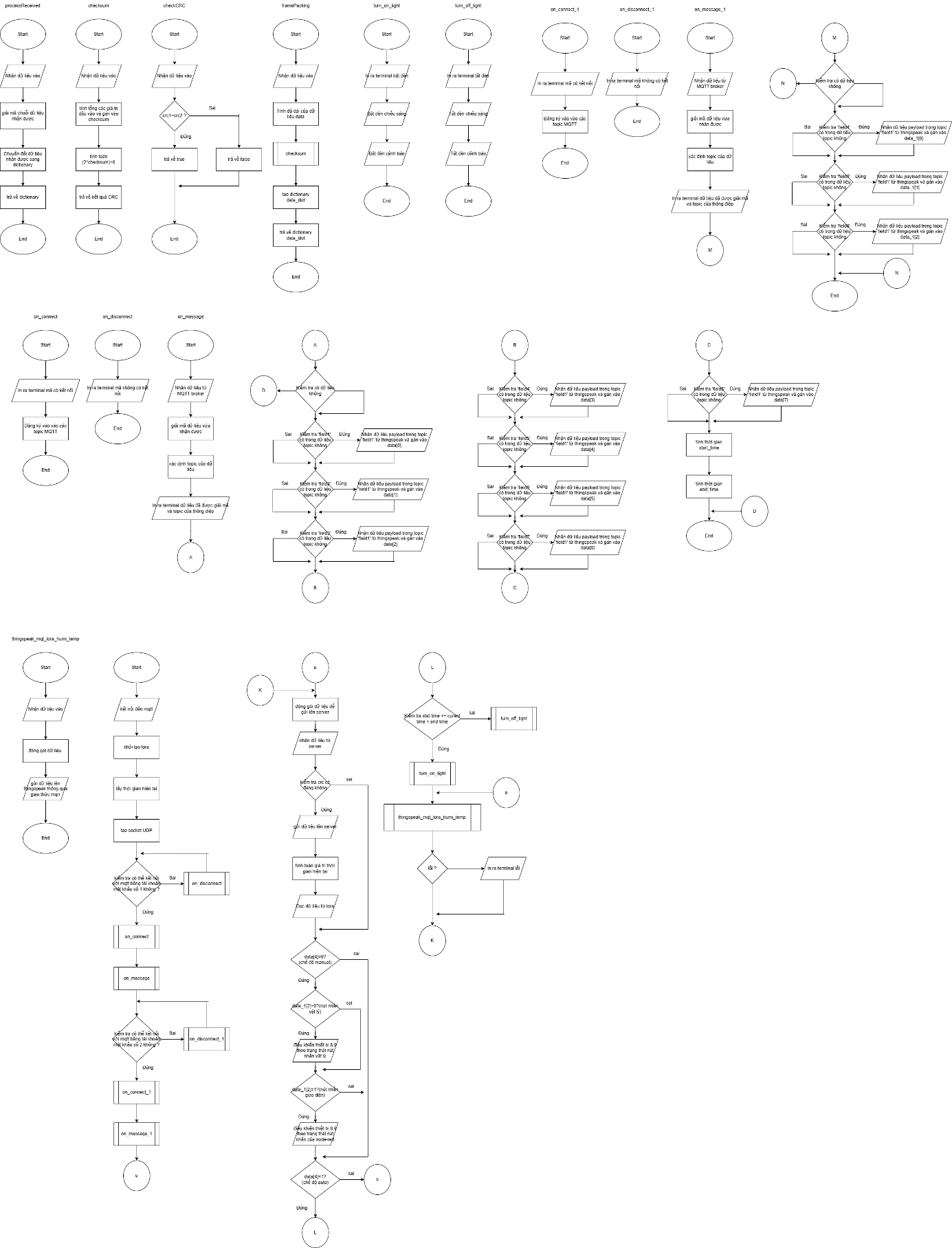


Slave:

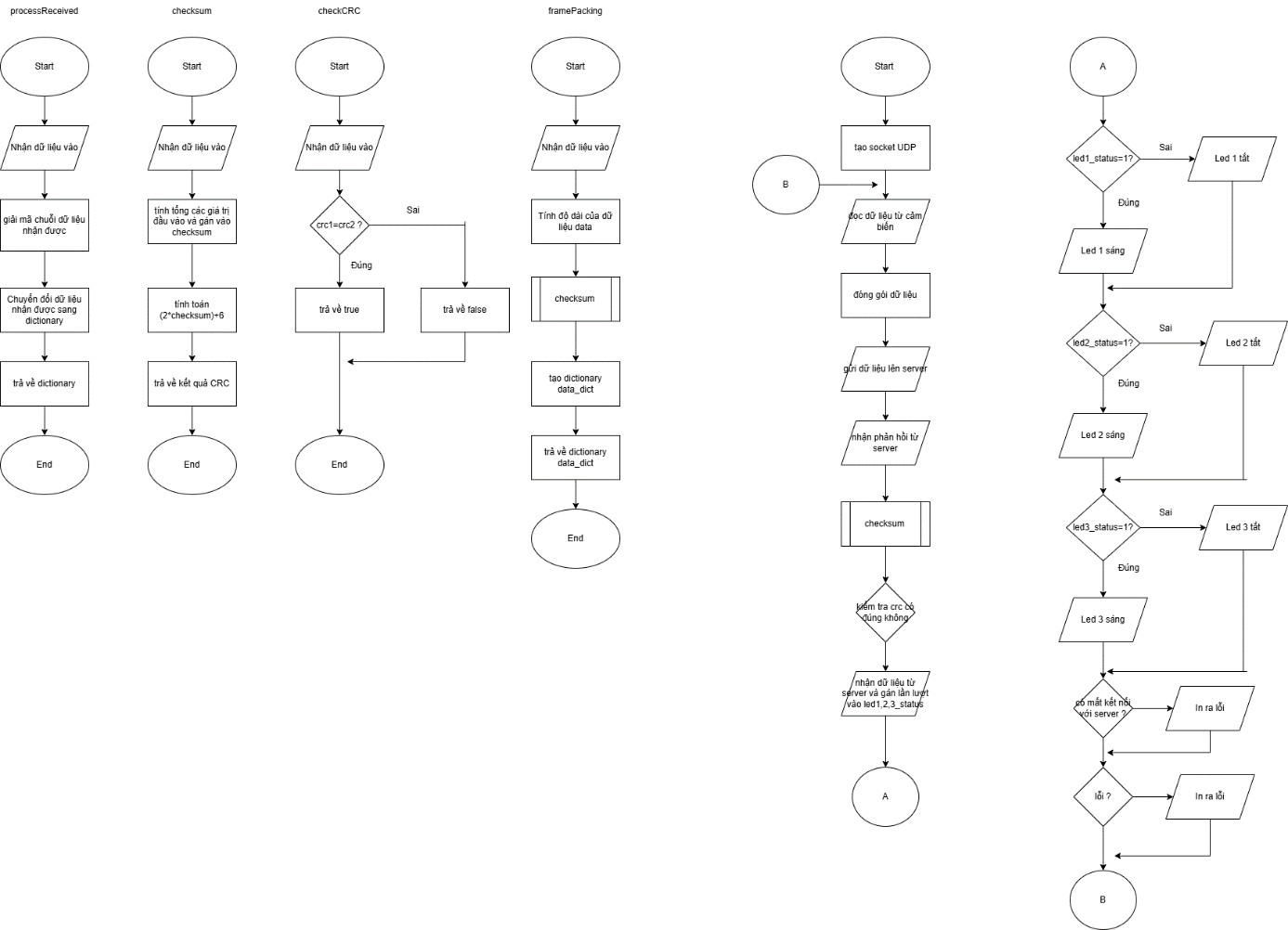


# **II.Yêu cầu 3: Lưu đồ giải thuật.**

Master:



Slave:



# **III.Yêu cầu 4: Mã nguồn chương trình.**

Master:

import sys

import socket

import ast

from gpiozero import LED

from datetime import datetime

import paho.mqtt.client as mqtt

import time

from Lora\_Driver\_USB.IoT\_Driver\_USB import LORA\_USB

from time import sleep

#Khởi tạo Lora

lora = LORA\_USB(COM='/dev/ttyUSB0', address=8)

#UDP

localIP = "0.0.0.0"

localPort = 11000

bufferSize = 1024

# Tạo socket UDP một lần duy nhất

UDP\_Server = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_DGRAM)

UDP\_Server.bind((localIP, localPort))

print("UDP server up and listening")

def processReceived(dataReceived):

    input\_string = dataReceived.decode("utf-8")

    data\_dict = ast.literal\_eval(input\_string)

    return data\_dict

def checksum(start, id, cmd, length, datasets, stop):

    checksum = 0

    checksum += (start + id + cmd + length + sum(data for data in datasets) + stop)

    return (2 \* checksum) + 6

def framePacking(start, id, cmd, data, stop):

    data\_dict = {

        "Start": start,

        "ID": id,

        "CMD": cmd,

        "Length": len(data),

        "Data": data,

        "CRC": checksum(start, id, cmd, len(data), data, stop),

        "Stop": stop

    }

    return data\_dict

def checkCRC(crc1, crc2):

    return crc1 == crc2

# Khởi tạo dữ liệu

data = [0, 0, 0, 0, 0,0,0,0]

data\_1 = [0, 0, 0, 0, 0,0,0,0]

start\_time = 0

end\_time = 0

#ham manual

def turn\_on\_light():

    print("Bật đèn")

    lora.write\_data(6, 8, 1)

    lora.write\_data(6, 9, 1)

def turn\_off\_light():

     print("Tắt đèn")

     lora.write\_data(6, 8, 0)

     lora.write\_data(6, 9, 0)

client\_post = mqtt.Client("OygqFTIpBwEOKw4gDgc0DCQ")

client\_post.username\_pw\_set(username="OygqFTIpBwEOKw4gDgc0DCQ",

                                password="IVj2XFZgk7YWWtbt/tS3y1jw")

client\_post.connect("mqtt3.thingspeak.com", 1883, 60)

client\_post.loop\_start()

# Hàm gửi dữ liệu lên ThingSpeak// noed-iot

def thingspeak\_mqt\_lora\_humi\_temp(data1, data2, data3, data4, data5,data6,data7):

        channel\_ID = "2714865"

        payload = "field1=%s&field2=%s&field3=%s&field4=%s&field5=%s&field6=%s&field7=%s&status=MQTTPUBLISH" % (data1, data2, data3, data4, data5,data6,data7)

        client\_post.publish("channels/%s/publish" % channel\_ID, payload)

# Hàm xử lý kết nối

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

        print('Connected to channel button\_MQTT with code {}'.format(rc))

        channel\_ID = '2714863'

        client.subscribe('channels/{}/subscribe/fields/field1'.format(channel\_ID))

        client.subscribe('channels/{}/subscribe/fields/field2'.format(channel\_ID))

        client.subscribe('channels/{}/subscribe/fields/field3'.format(channel\_ID))

        client.subscribe('channels/{}/subscribe/fields/field4'.format(channel\_ID))

        client.subscribe('channels/{}/subscribe/fields/field5'.format(channel\_ID))

        client.subscribe('channels/{}/subscribe/fields/field6'.format(channel\_ID))

        client.subscribe('channels/{}/subscribe/fields/field7'.format(channel\_ID))

        client.subscribe('channels/{}/subscribe/fields/field8'.format(channel\_ID))

# Hàm xử lý ngắt kết nối

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

        print('Disconnected with result code {}'.format(rc))

# Hàm xử lý tin nhắn

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

        global data, start\_time,end\_time, current\_time

        topic = message.topic

        payload = message.payload.decode()

        if payload != 'null':

            if 'field1' in topic :

                data[0] = int(payload)  # Gio\_batdau

            elif 'field2' in topic:

                data[1] = int(payload)  # Phut\_batdau

            elif 'field3' in topic:

                data[2] = int(payload)  # Gio\_ketthuc

            elif 'field4' in topic:

                data[3] = int(payload)  # Phut\_ketthuc

            elif 'field5' in topic:

                data[4] = int(payload)  # Auto/Manual

            elif 'field6' in topic:

                data[5] = int(payload)  # LED1

            elif 'field7' in topic:

                data[6] = int(payload)  # LED2

            elif 'field8' in topic:

                data[7] = int(payload)  # LED3

            start\_time = data[0] \* 60 + data[1]

            end\_time = data[2] \* 60 + data[3]

# Khởi tạo client MQTT

client\_id = 'HjANIQ0PLR4VCg80MSoWGyg'

client = mqtt.Client(client\_id)

client.on\_connect = on\_connect

client.on\_disconnect = on\_disconnect

client.on\_message = on\_message

client.username\_pw\_set(username='HjANIQ0PLR4VCg80MSoWGyg', password='+oXwWzsqPDZaC3jRkdTY5W/l')

client.connect("mqtt3.thingspeak.com", 1883, 68)

client.loop\_start()

# Hàm xử lý kết nối

def on\_connect\_1(client, userdata, flags, rc):

        print('Connected to channel button\_MQTT with code {}'.format(rc))

        channel\_ID = '2714865'

        client.subscribe('channels/{}/subscribe/fields/field4'.format(channel\_ID))

        client.subscribe('channels/{}/subscribe/fields/field5'.format(channel\_ID))

        client.subscribe('channels/{}/subscribe/fields/field8'.format(channel\_ID))

# Hàm xử lý ngắt kết nối

def on\_disconnect\_1(client, userdata, rc):

        print('Disconnected with result code {}'.format(rc))

# Hàm xử lý tin nhắn

def on\_message\_1(client, userdata, message):

        global data, start\_time,end\_time, current\_time

        topic = message.topic

        payload = message.payload.decode()

        if payload != 'null':

            if 'field4' in topic :

                data\_1[0] = int(payload)  # ALRAM

            elif 'field5' in topic:

                data\_1[1] = int(payload)  # LIGHT

            elif 'field8' in topic:

                data\_1[2] = int(payload)  # MODE

        print("mode sw =",data\_1[2])

# Khởi tạo client MQTT

client\_id = 'MjAFNycZNTItLSMaLCYAAgg'

client\_1 = mqtt.Client(client\_id)

client\_1.on\_connect = on\_connect\_1

client\_1.on\_disconnect = on\_disconnect\_1

client\_1.on\_message = on\_message\_1

client\_1.username\_pw\_set(username='MjAFNycZNTItLSMaLCYAAgg', password='cxOf5c2gXOMEiR3dvVMKsMEh')

client\_1.connect("mqtt3.thingspeak.com", 1883, 68)

client\_1.loop\_start()

humi = None

temp = None

# Vòng lặp chính

while True:

    try:

        led\_status = [data[5],data[6],data[7]]

        msgToClient = str(framePacking(0x01, 0x00, 0x01, led\_status, 0x00))

        bytesToSend = str.encode(msgToClient)

        # Nhận dữ liệu từ client

        ClientMsg = UDP\_Server.recvfrom(bufferSize)

        ClientMsg\_mgs = ClientMsg[0]

        ClientMsg\_adr = ClientMsg[1]

        data\_dict = processReceived(ClientMsg\_mgs)

        msg = "Message from Client: {}".format(data\_dict)

        crc2 = checksum(data\_dict['Start'], data\_dict['ID'], data\_dict['CMD'], len(data\_dict['Data']),

                        data\_dict['Data'], data\_dict['Stop'])

        if checkCRC(data\_dict['CRC'], crc2):

            adr = "Client IP + Port: {}".format(ClientMsg\_adr)

            humi = data\_dict['Data'][0]

            temp = data\_dict['Data'][1]

            print("humi  = {}, temp = {} ".format(humi,temp))

            #send chung voi lora

            print(data\_dict)

            print(adr)

            UDP\_Server.sendto(bytesToSend, ClientMsg\_adr)

            print("-----------------------------------------------")

        now = datetime.now()

        current\_time = now.hour \* 60 + now.minute

        val\_sw1 = lora.read\_data(6, 4)

        val\_sw2 = lora.read\_data(6, 5)

        val\_sw3 = lora.read\_data(6, 6)

        if data[4] == 0:  #Manual

            print("MODE Manual - ON")

            if data\_1[2] == 0 : #nut nhan vat ly

                lora.write\_data(6, 8, val\_sw1)

                lora.write\_data(6, 9, val\_sw2)

            if data\_1[2] == 1 : #nut nhan giao dien

                if data\_1[0] == 1:

                    lora.write\_data(6, 8, 1)

                else:

                    lora.write\_data(6, 8 ,0)

                if data\_1[1] == 1:

                    lora.write\_data(6, 9, 1)

                else:

                    lora.write\_data(6, 9 ,0)

        if data[4] == 1:  #Auto

            if start\_time <= current\_time < end\_time :

                turn\_on\_light()

            else:

                turn\_off\_light()

        # In thời gian bắt đầu và kết thúc

        print("Trang thai LED1 = {}, LED2 = {}, LED3 = {}".format(data[5],data[6],data[7]))

        print("Thời gian bắt đầu: {}:{}".format(data[0], data[1]))

        print("Thời gian kết thúc: {}:{}".format(data[2], data[3]))

        print("thoi gian hien tai :", current\_time)

        print("thoi gian hien tai :", now)

        print(f"start : {start\_time} , end : {end\_time}")

        # sleep(5)

        #######################################

        print("device: {}:{}:{}".format(val\_sw1, val\_sw2, val\_sw3))

        val\_light = lora.read\_data(6, 9)

        val\_alarm = lora.read\_data(6, 8)

        thingspeak\_mqt\_lora\_humi\_temp(val\_sw1,val\_sw2,val\_sw3,val\_alarm,val\_light,humi,temp)

    except Exception as e:

            print("Có lỗi: {}".format(e))

            sleep(1)

Slave:

import socket

from time import sleep

from gpiozero import LED

from seeed\_dht import DHT

import ast  # Thêm import ast

# # Khai báo 3 LED trên các chân GPIO

led1 = LED(5)   # LED 1 ở chân GPIO 5

led2 = LED(22)  # LED 2 ở chân GPIO 22

led3 = LED(24)  # LED 3 ở chân GPIO 24

sensor = DHT("11", 16)

ServerAddressPort = ("192.168.1.22", 11000)

bufferSize = 1024

# Tạo socket UDP một lần duy nhất

UDP\_Client = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_DGRAM)

UDP\_Client.settimeout(5)  # Thêm timeout để tránh bị treo

def processReceived(dataReceived):

    input\_string = dataReceived.decode("utf-8")

    data\_dict = ast.literal\_eval(input\_string)

    return data\_dict

def checksum(start, id, cmd, length, datasets, stop):

    checksum = 0

    checksum += (start + id + cmd + length + sum(data for data in datasets) + stop)

    return (2 \* checksum) + 6

def framePacking(start, id, cmd, data, stop):

    data\_dict = {

        "Start": start,

        "ID": id,

        "CMD": cmd,

        "Length": len(data),

        "Data": data,

        "CRC": checksum(start, id, cmd, len(data), data, stop),

        "Stop": stop

    }

    return data\_dict

def checkCRC(crc1, crc2):

    return crc1 == crc2

while True:

    try:

        humi, temp = sensor.read()

        print("Nhiet do = ", temp)

        print("Do Am = ", humi)

        # Đóng gói dữ liệu gửi lên server (nhiệt độ và độ ẩm)

        msgToServer = str(framePacking(0x01, 0x01, 0x01, [humi, temp], 0x00))

        bytesToSend = str.encode(msgToServer)

        # Gửi dữ liệu lên server

        UDP\_Client.sendto(bytesToSend, ServerAddressPort)

        print("UDP Client up and sending")

        # Nhận phản hồi từ server

        ServerMsg = UDP\_Client.recvfrom(bufferSize)

        ServerMsg\_msg = ServerMsg[0]

        ServerMsg\_adr = ServerMsg[1]

        msg = "Message from Server: {}".format(ServerMsg\_msg)

        adr = "Address from Server: {}".format(ServerMsg\_adr)

        # Xử lý dữ liệu nhận được từ server

        data = processReceived(ServerMsg\_msg)

        # Tính toán và kiểm tra checksum (CRC)

        crc2 = checksum(data['Start'], data['ID'], data['CMD'], len(data['Data']), data['Data'], data['Stop'])

        print("crc2 = ", crc2)

        if checkCRC(data['CRC'], crc2):

            print(data)

            # Điều khiển 3 đèn LED dựa trên dữ liệu nhận được

            led1\_status = data['Data'][0]  # Trạng thái của LED 1

            led2\_status = data['Data'][1]  # Trạng thái của LED 2

            led3\_status = data['Data'][2]  # Trạng thái của LED 3

            # Bật/tắt LED 1

            if led1\_status == 1:

                print("LED1 ON")

                led1.on()

            else:

                print("LED1 OFF")

                led1.off()

            # Bật/tắt LED 2

            if led2\_status == 1:

                print("LED2 ON")

                led2.on()

            else:

                print("LED2 OFF")

                led2.off()

            # Bật/tắt LED 3

            if led3\_status == 1:

                print("LED3 ON")

                led3.on()

            else:

                print("LED3 OFF")

                led3.off()

        print(msg)

        print(adr)

    except socket.timeout:

        print("Timeout: Không nhận được phản hồi từ server.")

    except Exception as e:

        print(f"Lỗi: {e}")

    print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

    sleep(1)

# **IV.Yêu cầu 5: Giải thích mã nguồn.**

Master:

import sys

import socket

import ast

from gpiozero import LED

from datetime import datetime

import paho.mqtt.client as mqtt

import time

from Lora\_Driver\_USB.IoT\_Driver\_USB import LORA\_USB

from time import sleep

sys: Sử dụng để quản lý các tham số và chức năng hệ thống.

socket: Được sử dụng để thiết lập giao thức UDP (User Datagram Protocol) nhằm nhận và gửi dữ liệu qua mạng.

ast: Thư viện cung cấp literal\_eval, dùng để chuyển chuỗi thành dictionary (định dạng Python object).

gpiozero: Thư viện dùng để điều khiển GPIO, đặc biệt là LED.

datetime: Dùng để lấy thời gian hiện tại.

paho.mqtt.client: Thư viện dùng để kết nối và giao tiếp qua giao thức MQTT (Message Queuing Telemetry Transport) với các thiết bị IoT.

LORA\_USB: Thư viện cho phép giao tiếp với module LoRa thông qua USB, dùng để truyền dữ liệu không dây.

sleep: Hàm để dừng chương trình tạm thời.

#Khởi tạo Lora

lora = LORA\_USB(COM='/dev/ttyUSB0', address=8)

#UDP

localIP = "0.0.0.0"

localPort = 11000

bufferSize = 1024

# Tạo socket UDP một lần duy nhất

UDP\_Server = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_DGRAM)

UDP\_Server.bind((localIP, localPort))

print("UDP server up and listening")

LORA\_USB: Khởi tạo đối tượng Lora với cổng /dev/ttyUSB0 và địa chỉ là 8, giúp giao tiếp với thiết bị LoRa.

UDP Socket: Thiết lập socket UDP để lắng nghe các gói tin từ bất kỳ địa chỉ IP nào (0.0.0.0) trên cổng 11000 với kích thước bộ đệm là 1024 bytes. Khi có gói tin đến, UDP\_Server sẽ nhận và xử lý chúng.

def processReceived(dataReceived):

    input\_string = dataReceived.decode("utf-8")

    data\_dict = ast.literal\_eval(input\_string)

    return data\_dict

Hàm này chuyển đổi dữ liệu dạng chuỗi nhận từ server thành một dictionary.

def checksum(start, id, cmd, length, datasets, stop):

    checksum = 0

    checksum += (start + id + cmd + length + sum(data for data in datasets) + stop)

    return (2 \* checksum) + 6

Tính toán giá trị checksum để đảm bảo tính toàn vẹn của dữ liệu. Hàm này cộng các giá trị start, id, cmd, length, tổng các phần tử trong datasets, và stop rồi nhân đôi kết quả và cộng thêm 6.

def framePacking(start, id, cmd, data, stop):

    data\_dict = {

        "Start": start,

        "ID": id,

        "CMD": cmd,

        "Length": len(data),

        "Data": data,

        "CRC": checksum(start, id, cmd, len(data), data, stop),

        "Stop": stop

    }

    return data\_dict

Hàm này đóng gói các thông tin cần thiết vào một dictionary để gửi tới server, bao gồm start, id, cmd, chiều dài của data, data, CRC (giá trị checksum), và stop.

def checkCRC(crc1, crc2):

    return crc1 == crc2

Hàm này so sánh hai giá trị checksum để kiểm tra tính toàn vẹn của dữ liệu.

def turn\_on\_light():

    print("Bật đèn")

    lora.write\_data(6, 8, 1)

    lora.write\_data(6, 9, 1)

def turn\_off\_light():

     print("Tắt đèn")

     lora.write\_data(6, 8, 0)

     lora.write\_data(6, 9, 0)

turn\_on\_light() và turn\_off\_light(): Điều khiển các đèn bằng cách gửi lệnh bật/tắt qua module LoRa. write\_data(6, 8, 1) là lệnh bật đèn và write\_data(6, 8, 0) là lệnh tắt đèn.

client\_post = mqtt.Client("OygqFTIpBwEOKw4gDgc0DCQ")

client\_post.username\_pw\_set(username="OygqFTIpBwEOKw4gDgc0DCQ",

                                password="IVj2XFZgk7YWWtbt/tS3y1jw")

client\_post.connect("mqtt3.thingspeak.com", 1883, 60)

client\_post.loop\_start()

Kết nối MQTT: Khởi tạo kết nối MQTT đến máy chủ mqtt3.thingspeak.com trên cổng 1883 với thông tin xác thực.

def thingspeak\_mqt\_lora\_humi\_temp(data1, data2, data3, data4, data5,data6,data7):

        channel\_ID = "2714865"

        payload = "field1=%s&field2=%s&field3=%s&field4=%s&field5=%s&field6=%s&field7=%s&status=MQTTPUBLISH" % (data1, data2, data3, data4, data5,data6,data7)

        client\_post.publish("channels/%s/publish" % channel\_ID, payload)

Gửi dữ liệu: Tạo chuỗi dữ liệu payload và gửi qua MQTT tới kênh 2714865 của ThingSpeak.

# Khởi tạo client MQTT

client\_id = 'HjANIQ0PLR4VCg80MSoWGyg'

client = mqtt.Client(client\_id)

client.on\_connect = on\_connect

client.on\_disconnect = on\_disconnect

client.on\_message = on\_message

client.username\_pw\_set(username='HjANIQ0PLR4VCg80MSoWGyg', password='+oXwWzsqPDZaC3jRkdTY5W/l')

client.connect("mqtt3.thingspeak.com", 1883, 68)

client.loop\_start()

Client ID và Thông tin Đăng nhập: Thiết lập thông tin đăng nhập và client ID, kết nối tới máy chủ MQTT.

Các hàm sự kiện: on\_connect, on\_disconnect, và on\_message sẽ được gọi khi client kết nối, ngắt kết nối và nhận tin nhắn.

while True:

    try:

        led\_status = [data[5],data[6],data[7]]

        msgToClient = str(framePacking(0x01, 0x00, 0x01, led\_status, 0x00))

        bytesToSend = str.encode(msgToClient)

        # Nhận dữ liệu từ client

        ClientMsg = UDP\_Server.recvfrom(bufferSize)

        ClientMsg\_mgs = ClientMsg[0]

        ClientMsg\_adr = ClientMsg[1]

        data\_dict = processReceived(ClientMsg\_mgs)

        msg = "Message from Client: {}".format(data\_dict)

        crc2 = checksum(data\_dict['Start'], data\_dict['ID'], data\_dict['CMD'], len(data\_dict['Data']),

                        data\_dict['Data'], data\_dict['Stop'])

        if checkCRC(data\_dict['CRC'], crc2):

            adr = "Client IP + Port: {}".format(ClientMsg\_adr)

            humi = data\_dict['Data'][0]

            temp = data\_dict['Data'][1]

            print("humi  = {}, temp = {} ".format(humi,temp))

            #send chung voi lora

            print(data\_dict)

            print(adr)

            UDP\_Server.sendto(bytesToSend, ClientMsg\_adr)

            print("-----------------------------------------------")

        now = datetime.now()

        current\_time = now.hour \* 60 + now.minute

        val\_sw1 = lora.read\_data(6, 4)

        val\_sw2 = lora.read\_data(6, 5)

        val\_sw3 = lora.read\_data(6, 6)

        if data[4] == 0:  #Manual

            print("MODE Manual - ON")

            if data\_1[2] == 0 : #nut nhan vat ly

                lora.write\_data(6, 8, val\_sw1)

                lora.write\_data(6, 9, val\_sw2)

            if data\_1[2] == 1 : #nut nhan giao dien

                if data\_1[0] == 1:

                    lora.write\_data(6, 8, 1)

                else:

                    lora.write\_data(6, 8 ,0)

                if data\_1[1] == 1:

                    lora.write\_data(6, 9, 1)

                else:

                    lora.write\_data(6, 9 ,0)

        if data[4] == 1:  #Auto

            if start\_time <= current\_time < end\_time :

                turn\_on\_light()

            else:

                turn\_off\_light()

        # In thời gian bắt đầu và kết thúc

        print("Trang thai LED1 = {}, LED2 = {}, LED3 = {}".format(data[5],data[6],data[7]))

        print("Thời gian bắt đầu: {}:{}".format(data[0], data[1]))

        print("Thời gian kết thúc: {}:{}".format(data[2], data[3]))

        print("thoi gian hien tai :", current\_time)

        print("thoi gian hien tai :", now)

        print(f"start : {start\_time} , end : {end\_time}")

        # sleep(5)

        #######################################

        print("device: {}:{}:{}".format(val\_sw1, val\_sw2, val\_sw3))

        val\_light = lora.read\_data(6, 9)

        val\_alarm = lora.read\_data(6, 8)

        thingspeak\_mqt\_lora\_humi\_temp(val\_sw1,val\_sw2,val\_sw3,val\_alarm,val\_light,humi,temp)

    except Exception as e:

            print("Có lỗi: {}".format(e))

            sleep(1)

Nhận và Gửi Dữ liệu: Nhận dữ liệu từ client qua UDP, xử lý, kiểm tra CRC để đảm bảo tính toàn vẹn và gửi lại cho client.

Điều khiển Đèn: Dựa vào chế độ Auto hoặc Manual để bật hoặc tắt đèn.

Gửi Dữ liệu lên ThingSpeak: Đọc giá trị từ LoRa và gửi lên ThingSpeak

Slave:

import socket

from time import sleep

from gpiozero import LED

from seeed\_dht import DHT

import ast  # Thêm import ast

socket: Dùng để tạo và quản lý kết nối UDP tới server.

time.sleep: Dùng để tạm dừng chương trình giữa các lần gửi dữ liệu.

gpiozero.LED: Dùng để điều khiển các đèn LED trên thiết bị thông qua các chân GPIO.

seeed\_dht.DHT: Dùng để đọc dữ liệu nhiệt độ và độ ẩm từ cảm biến DHT11.

ast: Chuyển đổi chuỗi nhận được từ server sang dạng dictionary.

led1 = LED(5)   # LED 1 ở chân GPIO 5

led2 = LED(22)  # LED 2 ở chân GPIO 22

led3 = LED(24)  # LED 3 ở chân GPIO 24

sensor = DHT("11", 16)

Khởi tạo ba đèn LED tương ứng với các chân GPIO 5, 22, và 24.

sensor là đối tượng cảm biến DHT11 được kết nối ở chân GPIO 16.

ServerAddressPort = ("192.168.1.22", 11000)

bufferSize = 1024

ServerAddressPort là địa chỉ IP và cổng của server mà thiết bị sẽ gửi dữ liệu đến.

bufferSize là kích thước của bộ đệm nhận dữ liệu từ server.

UDP\_Client = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_DGRAM)

UDP\_Client.settimeout(5)  # Thêm timeout để tránh bị treo

Tạo một socket UDP để giao tiếp với server. Socket này có thời gian chờ là 5 giây, nếu không nhận được phản hồi trong thời gian này thì sẽ báo lỗi.

def processReceived(dataReceived):

    input\_string = dataReceived.decode("utf-8")

    data\_dict = ast.literal\_eval(input\_string)

    return data\_dict

Hàm này chuyển đổi dữ liệu dạng chuỗi nhận từ server thành một dictionary.

def checksum(start, id, cmd, length, datasets, stop):

    checksum = 0

    checksum += (start + id + cmd + length + sum(data for data in datasets) + stop)

    return (2 \* checksum) + 6

Tính toán giá trị checksum để đảm bảo tính toàn vẹn của dữ liệu. Hàm này cộng các giá trị start, id, cmd, length, tổng các phần tử trong datasets, và stop rồi nhân đôi kết quả và cộng thêm 6.

def framePacking(start, id, cmd, data, stop):

    data\_dict = {

        "Start": start,

        "ID": id,

        "CMD": cmd,

        "Length": len(data),

        "Data": data,

        "CRC": checksum(start, id, cmd, len(data), data, stop),

        "Stop": stop

    }

    return data\_dict

Hàm này đóng gói các thông tin cần thiết vào một dictionary để gửi tới server, bao gồm start, id, cmd, chiều dài của data, data, CRC (giá trị checksum), và stop.

def checkCRC(crc1, crc2):

    return crc1 == crc2

Hàm này so sánh hai giá trị checksum để kiểm tra tính toàn vẹn của dữ liệu.

while True:

    try:

        humi, temp = sensor.read()

        print("Nhiet do = ", temp)

        print("Do Am = ", humi)

Đọc dữ liệu nhiệt độ và độ ẩm từ cảm biến, sau đó in ra màn hình.

        # Đóng gói dữ liệu gửi lên server (nhiệt độ và độ ẩm)

        msgToServer = str(framePacking(0x01, 0x01, 0x01, [humi, temp], 0x00))

        bytesToSend = str.encode(msgToServer)

        # Gửi dữ liệu lên server

        UDP\_Client.sendto(bytesToSend, ServerAddressPort)

        print("UDP Client up and sending")

Đóng gói dữ liệu đo được vào một dictionary và chuyển đổi thành chuỗi byte để gửi tới server.

        # Nhận phản hồi từ server

        ServerMsg = UDP\_Client.recvfrom(bufferSize)

        ServerMsg\_msg = ServerMsg[0]

        ServerMsg\_adr = ServerMsg[1]

Nhận phản hồi từ server trong bộ đệm bufferSize.

        # Tính toán và kiểm tra checksum (CRC)

        crc2 = checksum(data['Start'], data['ID'], data['CMD'], len(data['Data']), data['Data'], data['Stop'])

        print("crc2 = ", crc2)

        if checkCRC(data['CRC'], crc2):

            print(data)

Chuyển đổi dữ liệu nhận được thành dictionary, tính lại checksum và kiểm tra xem giá trị có khớp với checksum nhận từ server hay không. Nếu khớp, dữ liệu sẽ được xử lý tiếp.

            # Điều khiển 3 đèn LED dựa trên dữ liệu nhận được

            led1\_status = data['Data'][0]  # Trạng thái của LED 1

            led2\_status = data['Data'][1]  # Trạng thái của LED 2

            led3\_status = data['Data'][2]  # Trạng thái của LED 3

            # Bật/tắt LED 1

            if led1\_status == 1:

                print("LED1 ON")

                led1.on()

            else:

                print("LED1 OFF")

                led1.off()

            # Bật/tắt LED 2

            if led2\_status == 1:

                print("LED2 ON")

                led2.on()

            else:

                print("LED2 OFF")

                led2.off()

            # Bật/tắt LED 3

            if led3\_status == 1:

                print("LED3 ON")

                led3.on()

            else:

                print("LED3 OFF")

                led3.off()

Dựa vào các giá trị trạng thái trong phản hồi, bật hoặc tắt các đèn LED tương ứng.

    except socket.timeout:

        print("Timeout: Không nhận được phản hồi từ server.")

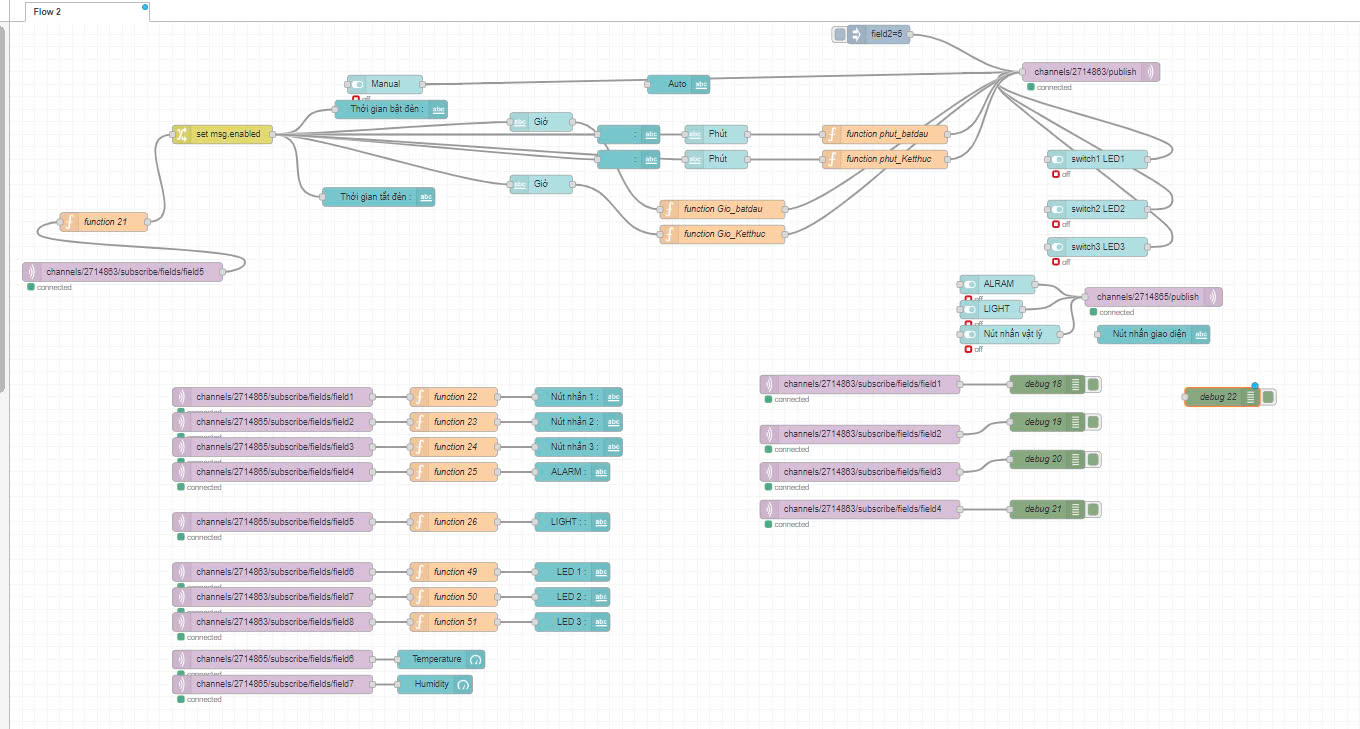
    except Exception as e:

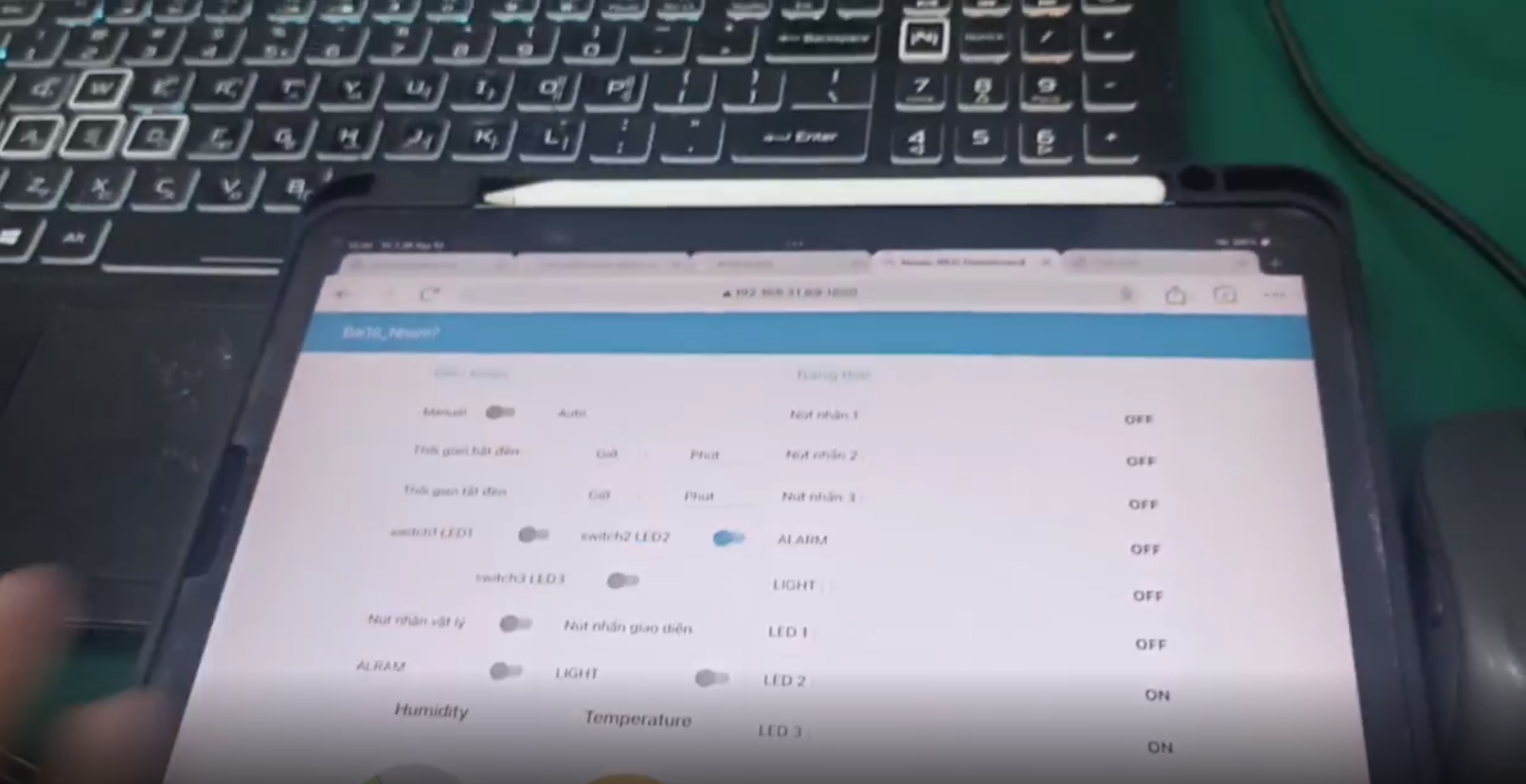
        print(f"Lỗi: {e}")

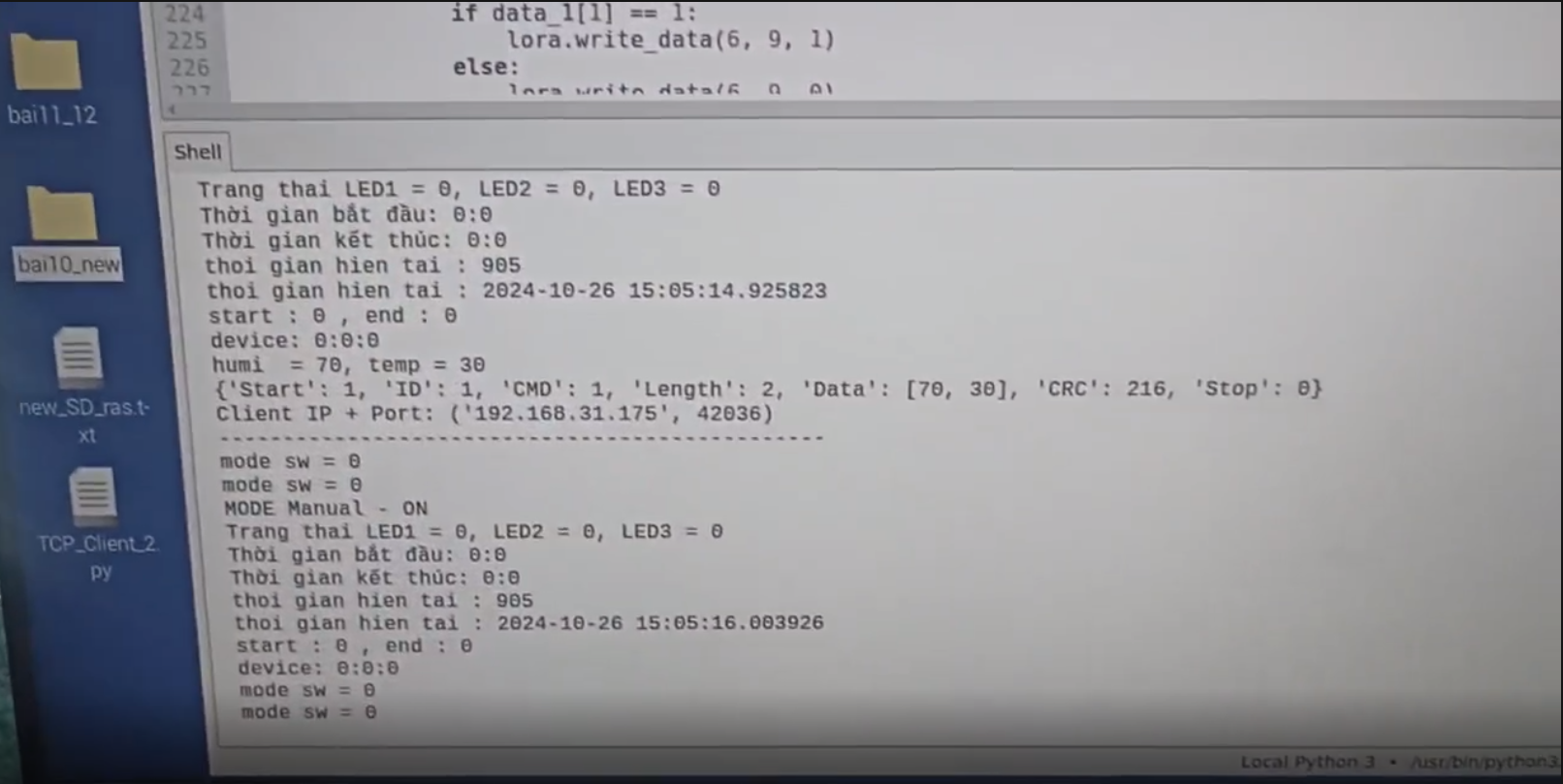
socket.timeout: Nếu không nhận được phản hồi từ server trong thời gian chờ, thông báo sẽ được in ra.

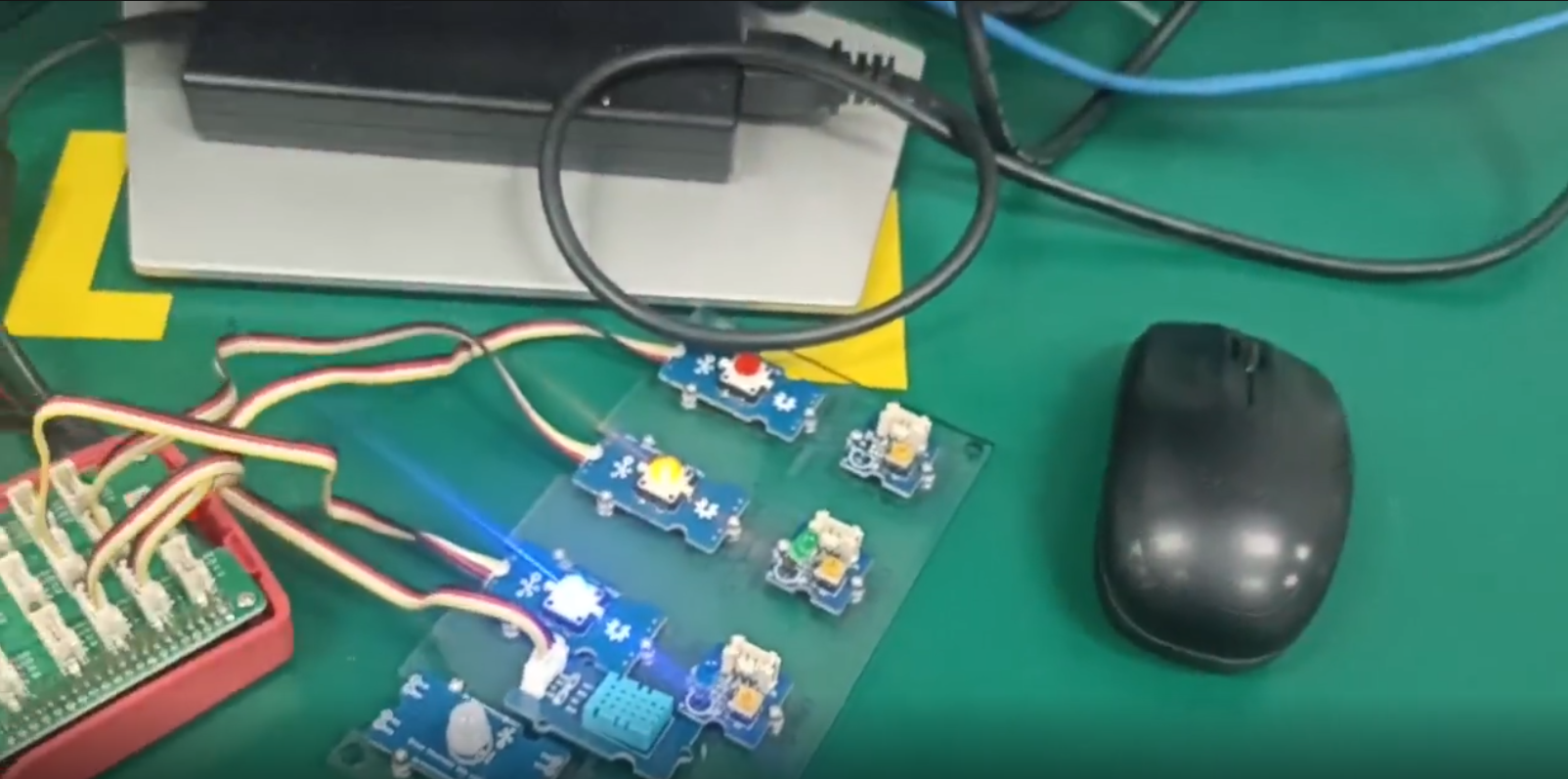
Exception: Xử lý các lỗi khác và in thông báo lỗi ra màn hình.

# **V.Yêu cầu 6: Minh chứng kết quả thực nghiệm.**











# **VI.Yêu cầu 7: Video.:**

https://www.youtube.com/watch?v=ft4uFg13o3E