Họ tên: Nguyễn Quang Minh

MSSV: 201404024

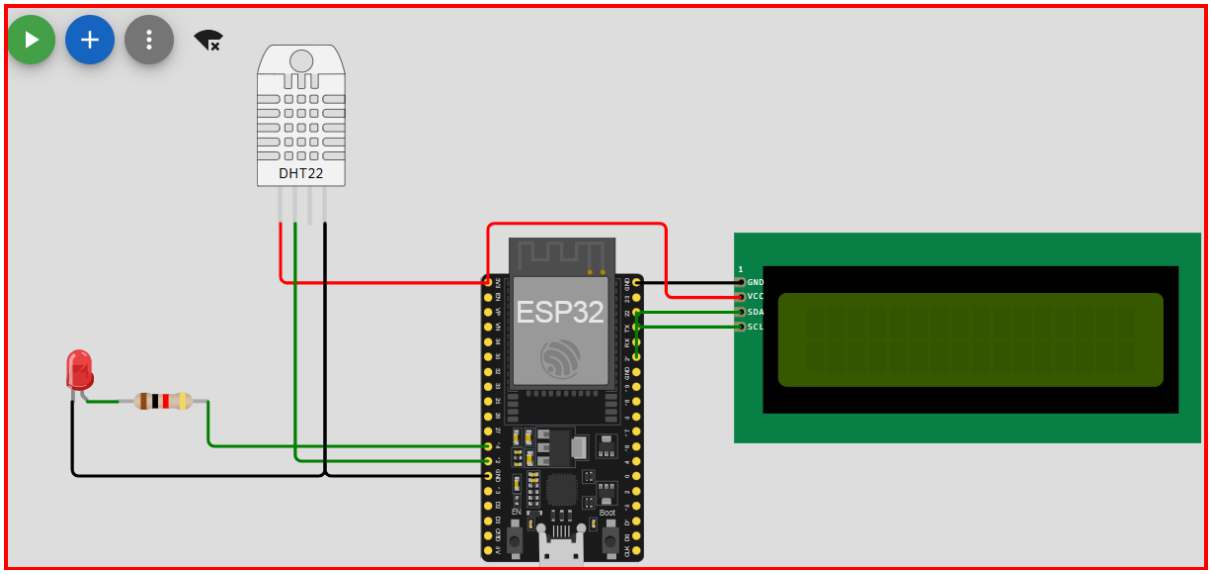
Lớp: Điện tử và tin học công nghiệp 1

Các project

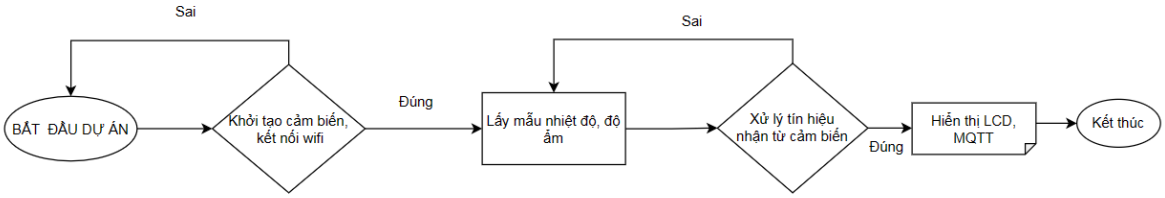
Project 1: Sử dụng giao thức MQTT gửi dữ liệu nhiệt độ, độ ẩm và hiển thị lcd sử dụng vi điều khiển esp32

Bài làm

- **Sơ đồ đấu nối:**



- **Thuật toán:**



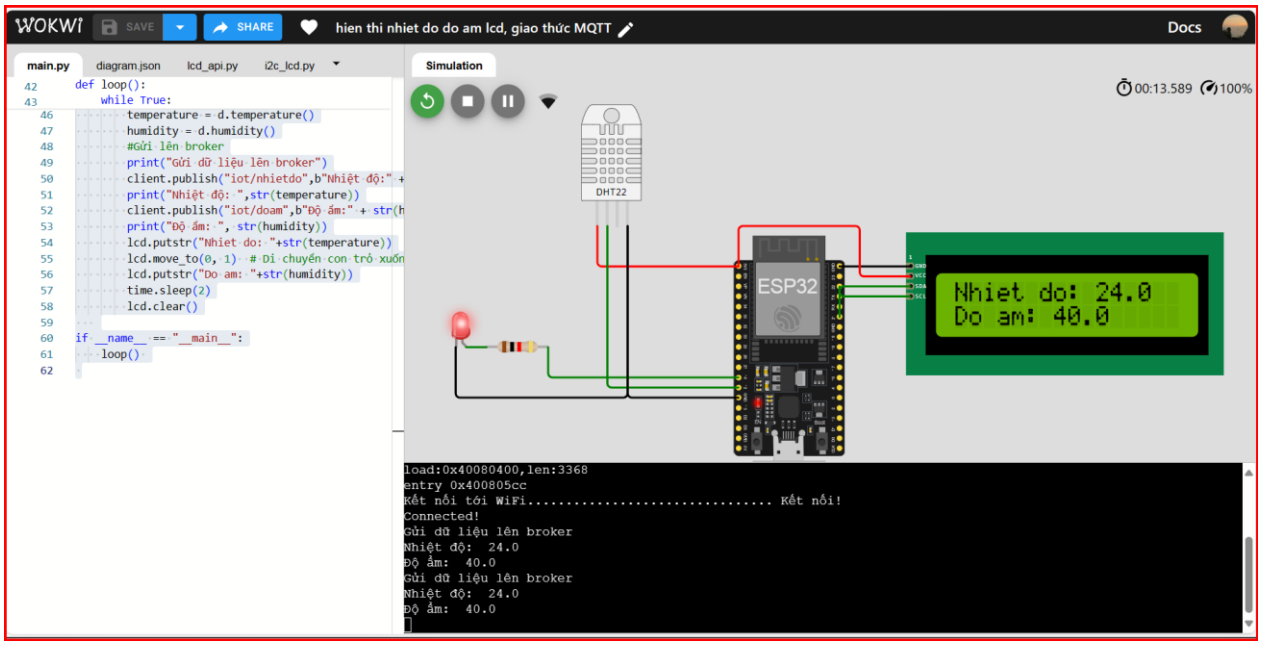
- **Code:**

```
import network

import time
import dht
from umqtt.simple import MQTTClient
import machine
from machine import Pin, SoftI2C
from utime import sleep_us,sleep
from lcd_api import LcdApi
```

```
from i2c_lcd import I2cLcd
I2C_ADDR = 0x27
totalRows = 2
totalCol = 14

led = Pin(14,Pin.OUT)
d = dht.DHT22(Pin(12))

i2c = SoftI2C(scl = Pin(22),sda = Pin(21), freq = 10000)
lcd = I2cLcd(i2c, I2C_ADDR, totalRows, totalCol)


print("Kết nối tới WiFi", end="")
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect('Wokwi-GUEST', '')
while not sta_if.isconnected():
  print(".", end="")
  time.sleep(0.1)
print(" Kết nối!")


# MQTT Server Parameters
client_id="Nguyen Quang Minh"
server = "broker.hivemq.com"
client = MQTTClient(client_id, server)
client.connect()
print("Connected!")


def loop():
    while True:
        d.measure()
        led.on()
        temperature = d.temperature()
        humidity = d.humidity()
        #Gửi lên broker
        print("Gửi dữ liệu lên broker")
        client.publish("iot/nhietdo",b"Nhiệt độ:" + str(temperature))
        print("Nhiệt độ: ",str(temperature))
        client.publish("iot/doam",b"Độ ẩm:" + str(humidity))
        print("Độ ẩm: ", str(humidity))
        lcd.putstr("Nhiet do: "+str(temperature))
```

```
        lcd.move_to(0, 1)  # Di chuyển con trỏ xuống dòng thứ 2
        lcd.putstr("Do am: "+str(humidity))
        time.sleep(2)
        lcd.clear()

if __name__ == "__main__":
    loop()
```

- **Kết quả**

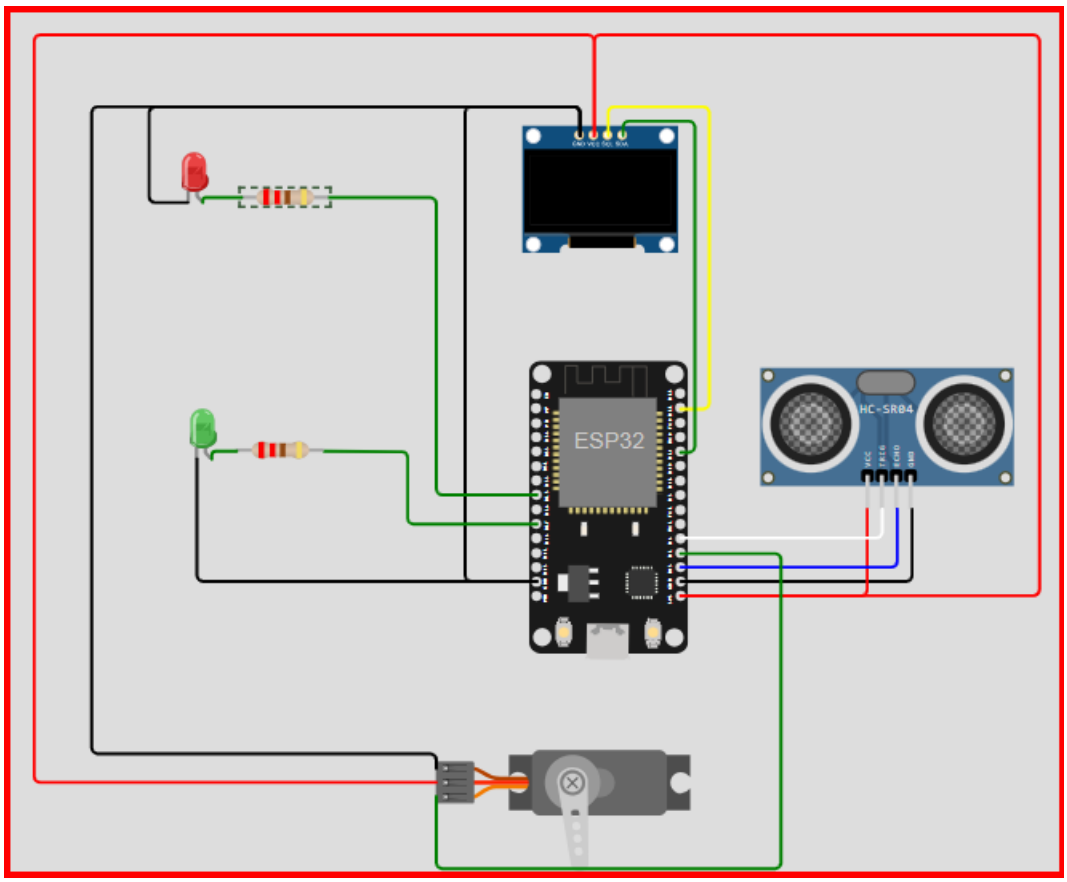


- **Link mô phỏng: https://wokwi.com/projects/397511081437933569**

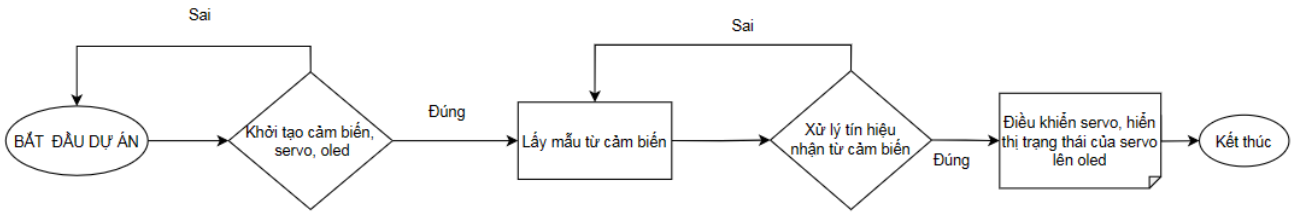
Project 2: Điều khiển đóng mở cổng sử dụng servo và cảm biến siêu âm HC-SR04 và hiển thị trạng thái đóng mở cổng lên màn hình Oled

Bài làm

- Sơ đồ đấu nối:



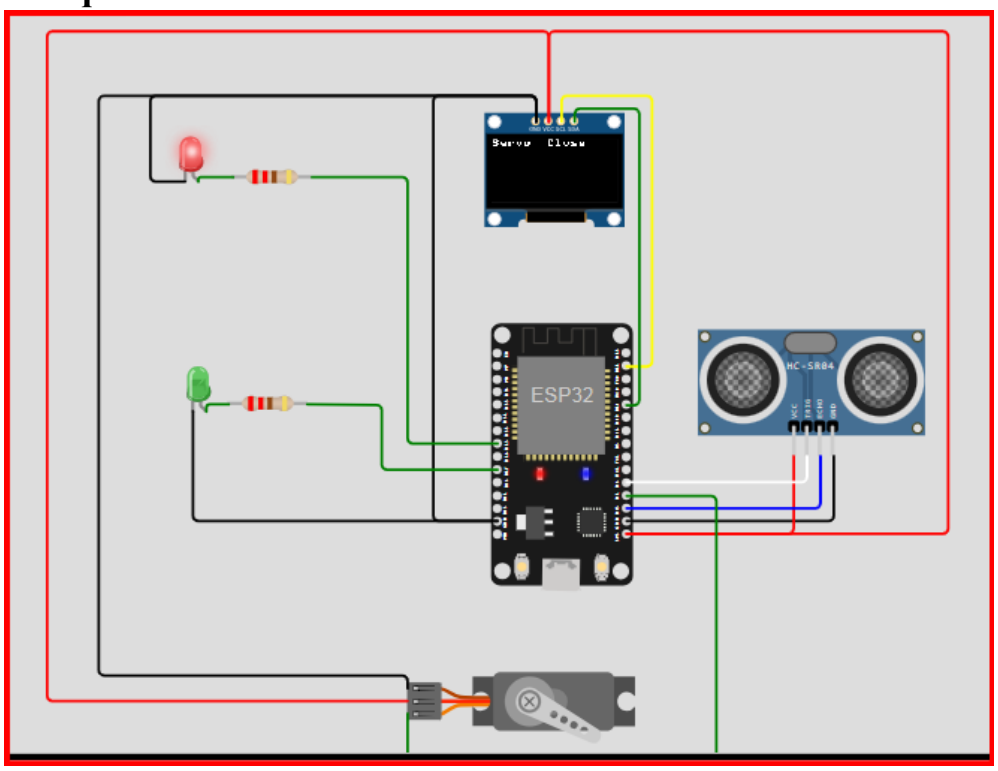
- **Thuật toán**



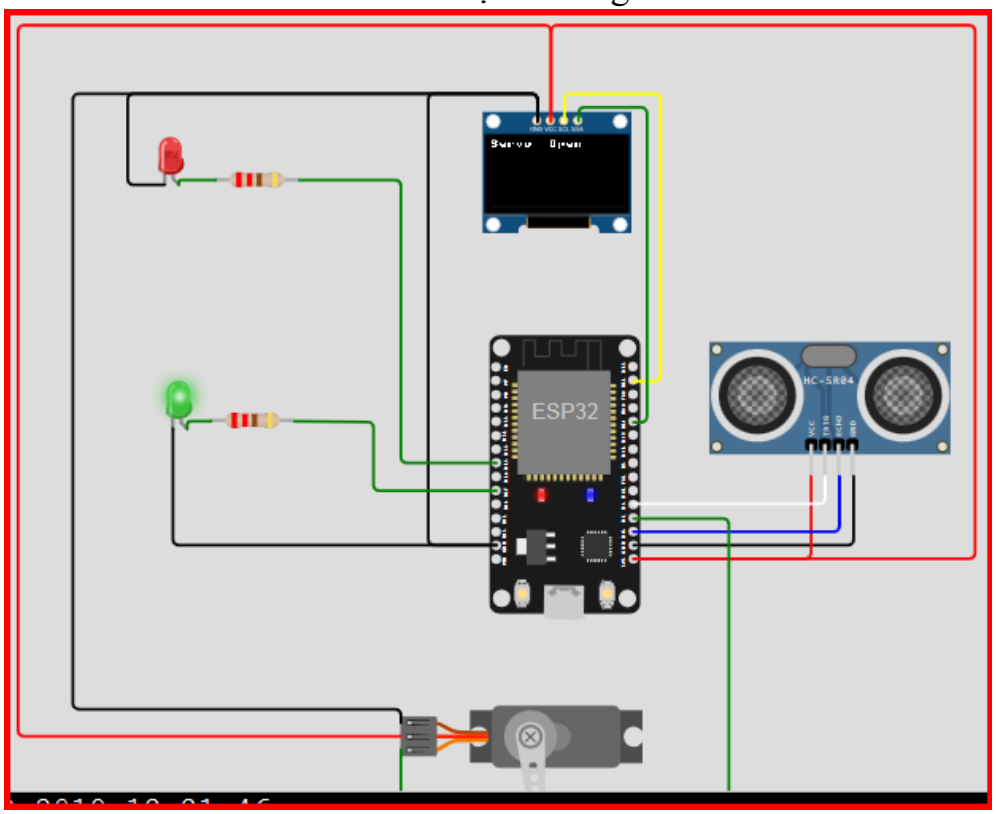
- **Code**

```
import time
from machine import Pin, PWM, I2C
from SSD1306 import SSD1306_I2C
from HCSR04 import HCSR04

i2c = I2C(0, scl=Pin(22), sda=Pin(21))
oled = SSD1306_I2C(128, 64, i2c)

red_led = PWM(Pin(25), freq=1000, duty=0)
green_led = PWM(Pin(27), freq=1000, duty=0)

pwm_servo = PWM(Pin(2), freq=50, duty=0)  # Assuming the servo is

sensor = HCSR04(trigger_pin=4, echo_pin=15)

def control_servo_and_leds(distance):
    if distance < 100:  # mở cửa
        pwm_servo.duty(120)
        green_led.duty(500)
        red_led.duty(0)
        oled.fill(0)
        oled.text("Servo: Open", 0, 0)
        oled.show()
    else: # đóng cửa
        pwm_servo.duty(90)
        red_led.duty(500)
        green_led.duty(0)
        oled.fill(0)
        oled.text("Servo: Close", 0, 0)
        oled.show()

while True:
    try:
        distance = sensor.distance_cm()
        control_servo_and_leds(distance)
        time.sleep(0.1)
    except OSError:
        print("Sensor error")
```

- **Kết quả**



Khi servo mở với điều kiện khoảng cách nhỏ hơn 100cm



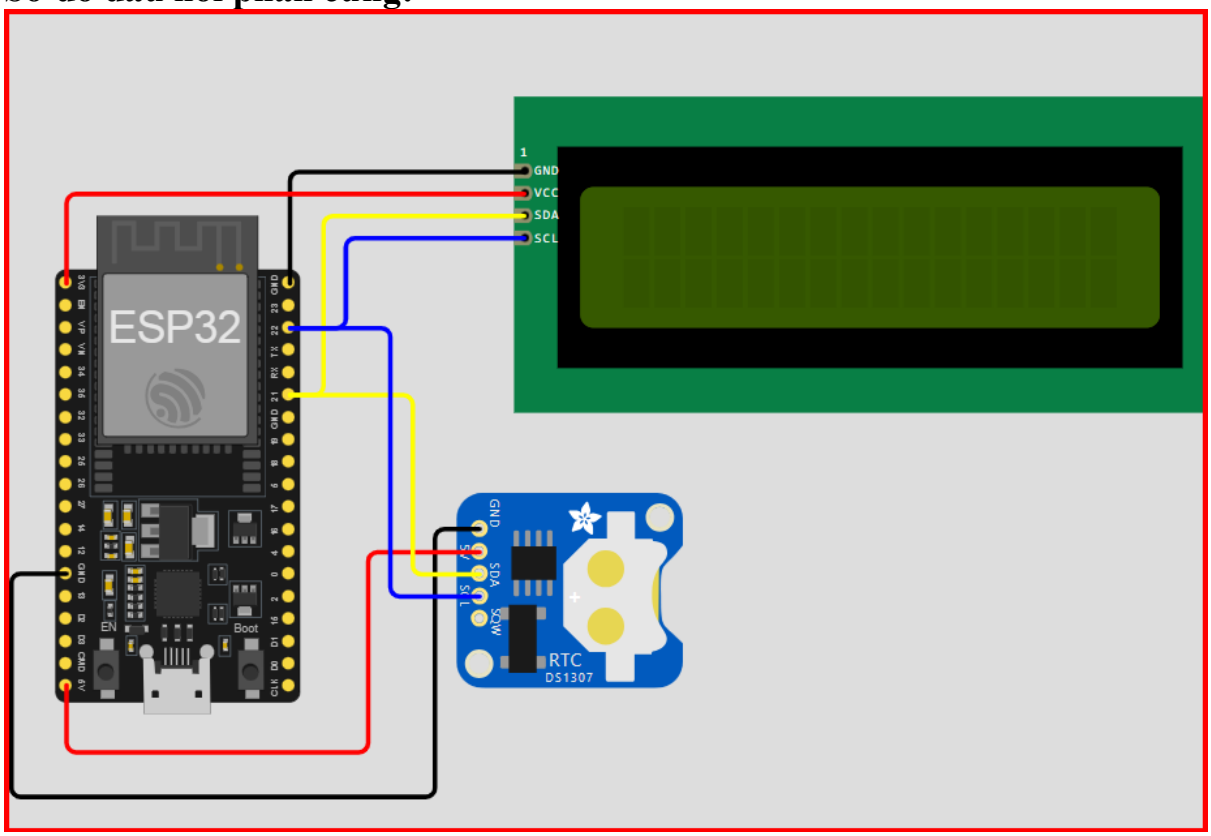
Khi servo đóng với điều kiện khoảng cách lớn hơn 100cm

- **Link mô phỏng:** https://wokwi.com/projects/391237865000997889

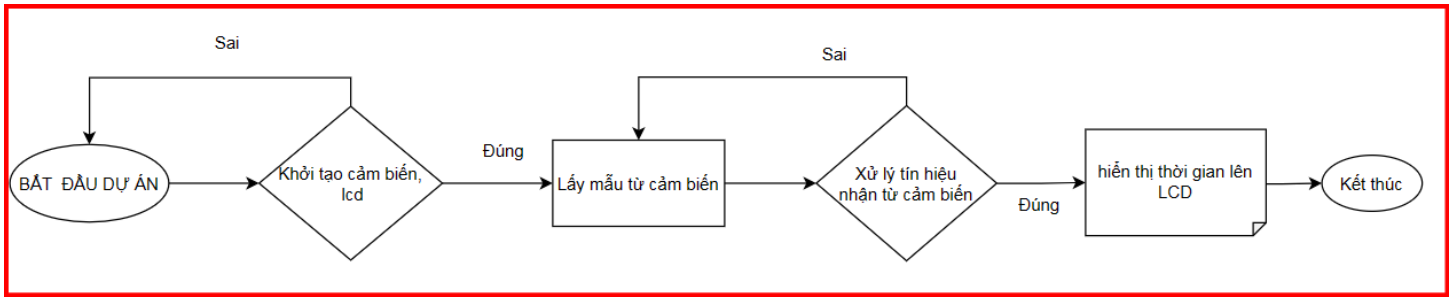
Project 3: Hiển thị đồng hồ thời gian thực sử dụng module RTC-ds1307 hiển thị LCD

Bài làm

- **Sơ đồ đấu nối phần cứng:**



- **Thuật toán**

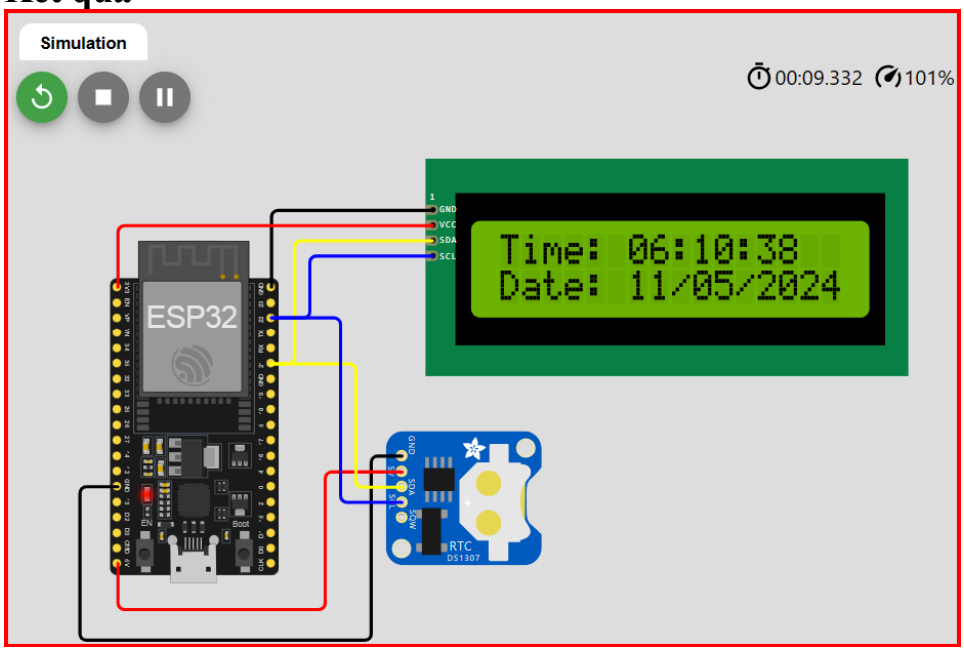


- **Code**

```
from machine import Pin, SoftI2C, RTC
import ds1307
import time
from lcd_api import LcdApi
from i2c_lcd import I2cLcd
```

```
i2c0 = SoftI2C(scl=Pin(22), sda=Pin(21), freq=100000)
i2c_lcd = SoftI2C(scl=Pin(22), sda=Pin(21), freq=10000)

I2C_ADDR = 0x27
totalRows = 2
totalCol = 16
lcd = I2cLcd(i2c_lcd, I2C_ADDR, totalRows, totalCol)

ds1307rtc = ds1307.DS1307(i2c0, 0x68)

ds1307rtc.disable_oscillator = True
print("disable_oscillator = ", ds1307rtc.disable_oscillator)
ds1307rtc.disable_oscillator = False
print("disable_oscillator = ", ds1307rtc.disable_oscillator, "\n")

while True:
    dt = ds1307rtc.datetime
    lcd.clear()

    lcd.move_to(0, 0)
    lcd.putstr("Time: {:02d}:{:02d}:{:02d}".format(dt[6], dt[4], dt[5]))

    lcd.move_to(0, 1)
    lcd.putstr("Date: {:02d}/{:02d}/{:04d}".format(dt[2], dt[1], dt[0]))

    time.sleep(1)
```

- **Kết quả**

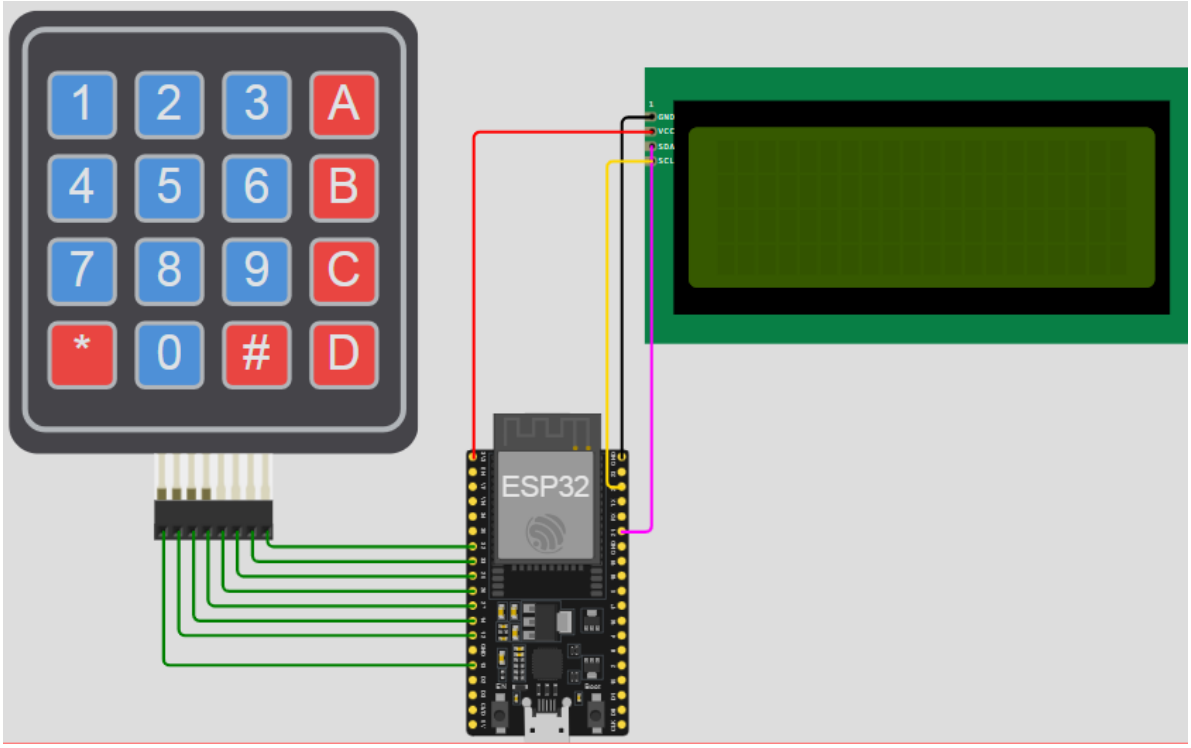


- **Link mô phỏng: https://wokwi.com/projects/397551128074201089**

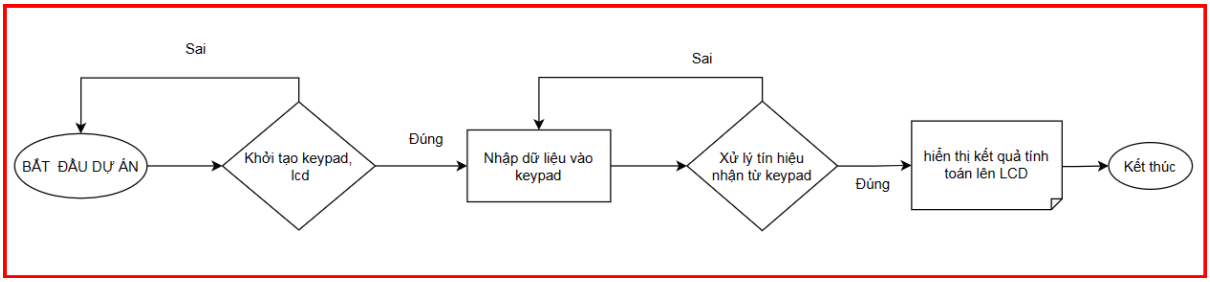
Project 4: làm máy tính cầm tay từ Keypad, LCD

Bài làm

- Sơ đồ đấu nối phần cứng



- **Thuật toán**



- **Code**

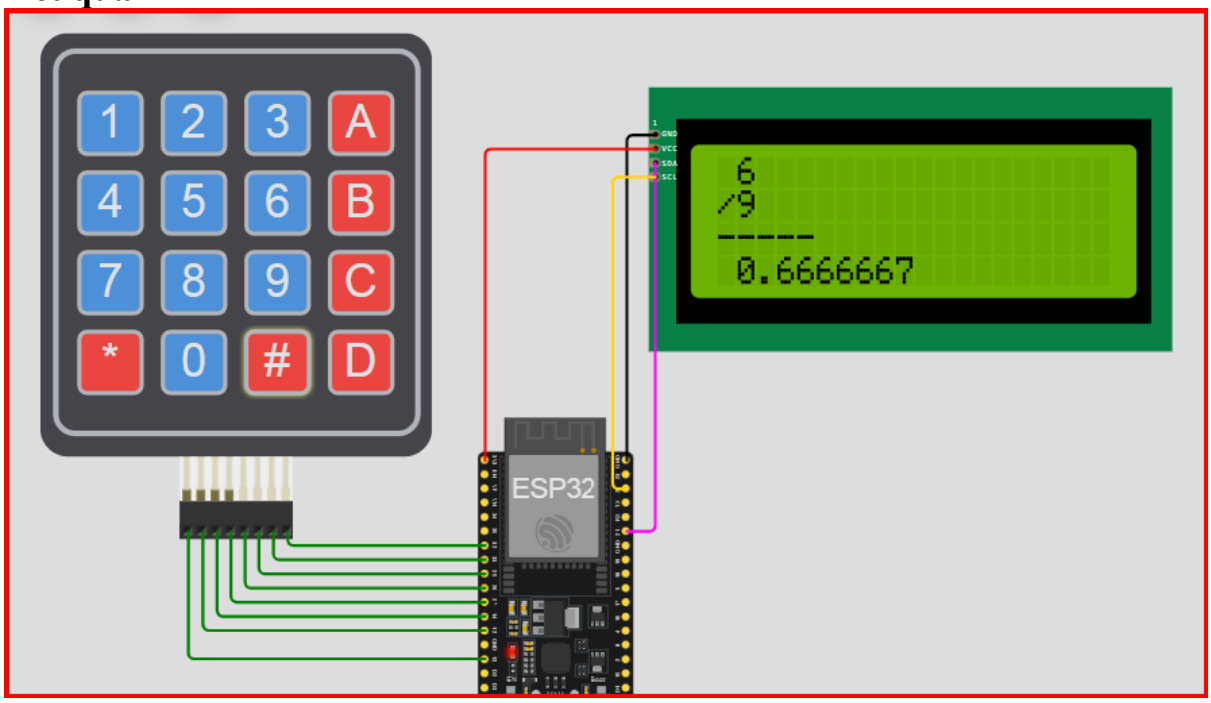
```
from time import sleep_ms
from machine import Pin, SoftI2C
from i2c_lcd import I2cLcd

AddressOfLcd = 0x27
i2c = SoftI2C(scl=Pin(22), sda=Pin(21), freq=400000) # connect scl to GPIO 22, sda to GPIO 21
lcd = I2cLcd(i2c, AddressOfLcd, 4, 20)
```

```
keypad = [
    ['1', '2', '3', '+' ],
    ['4', '5', '6', '-' ],
    ['7', '8', '9', '*' ],
    ['C', '0', '=', '/' ]
]

row_pins = [Pin(13, Pin.OUT), Pin(12, Pin.OUT), Pin(14, Pin.OUT), Pin(27,
Pin.OUT)]
col_pins = [Pin(26, Pin.IN, Pin.PULL_UP), Pin(25, Pin.IN, Pin.PULL_UP),
Pin(33, Pin.IN, Pin.PULL_UP), Pin(32, Pin.IN, Pin.PULL_UP)]


for row_pin in row_pins:
    row_pin.value(1)


for col_pin in col_pins:
    col_pin.value(0)

user_input = ""
result = None
math_sign = ""
sign_applied = False

def pad_string(input_string, desired_length = 15):
    current_length = len(input_string)
    if current_length >= desired_length:
        return input_string

    spaces_needed = desired_length - current_length

    padded_string = input_string + " " * spaces_needed

    return padded_string

def lcd_print(row, value, start_col = 1, space_padding = True):
    print("move to : " + str(row))
    lcd.move_to(start_col,row)
    if space_padding:
        lcd.putstr(pad_string(str(value)))
    else:
        lcd.putstr(str(value))
```

```
def get_key():
    keys_detected = []
    for i, row_pin in enumerate(row_pins):
        row_pin.value(0)

        for j, col_pin in enumerate(col_pins):
            if col_pin.value() == 0:
                keys_detected.append(keypad[i][j])
                return keypad[i][j]

        row_pin.value(1)

    return None

def evaluate_expression():
    global user_input
    global result
    global math_sign

    try:
        calc_result = float(result)
    except ValueError:
        calc_result = 0.0
        print("Invalid Result float format: " + result)

    try:
        calc_user_input = float(user_input)
    except ValueError:
        calc_user_input = 0.0
        print("Invalid User Input format: " + user_input)

    try:
        return str(eval(str(calc_result) + " " + str(math_sign) + " " +
str(user_input)))
    except:
        return "Error"

def keyboard_scan():
    global user_input
    global result
    global math_sign
    global sign_applied
```

```
    key = get_key()
    if key is not None:
        if key == 'C':
            user_input = ''
            result = None
            lcd.clear()
            math_sign = ""
            sign_applied = False
            lcd_print(2, "-----", 0, False)
        elif key == '=':
            try:
                if user_input:
                    result = evaluate_expression()
                    #result = evaluate_expression(result if result is not None else "" +
math_sign + user_input)
                    lcd_print(3, " " if result is None else str(result))
                    user_input = ""     # Clear input after result is calculated
            except Exception as e:
                print("Error Occured: ", e)

        elif key == '+' or key == '-' or key == '*' or key == '/':
            math_sign = key
            lcd_print(1, key, 0, False)
            result = user_input if sign_applied == False else result
            user_input = ""
            sign_applied = True

        else:
            user_input += key
            lcd_input_row = 0 if sign_applied == False else 1
            lcd_print(lcd_input_row, str(user_input))

    sleep_ms(100)


lcd_print(2, "-----", 0, False)

while True:
    keyboard_scan()
```

- **Kết quả**

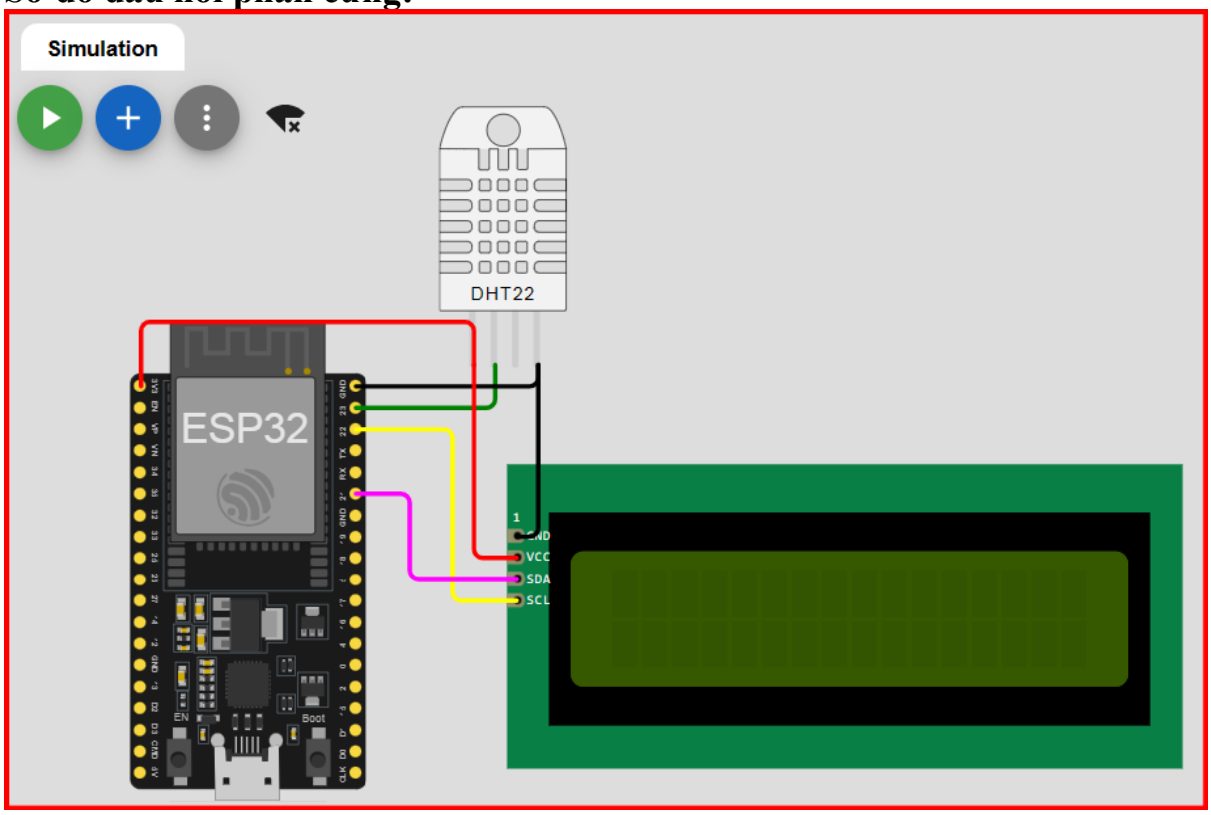


- **Link mô phỏng: https://wokwi.com/projects/397559905813595137**

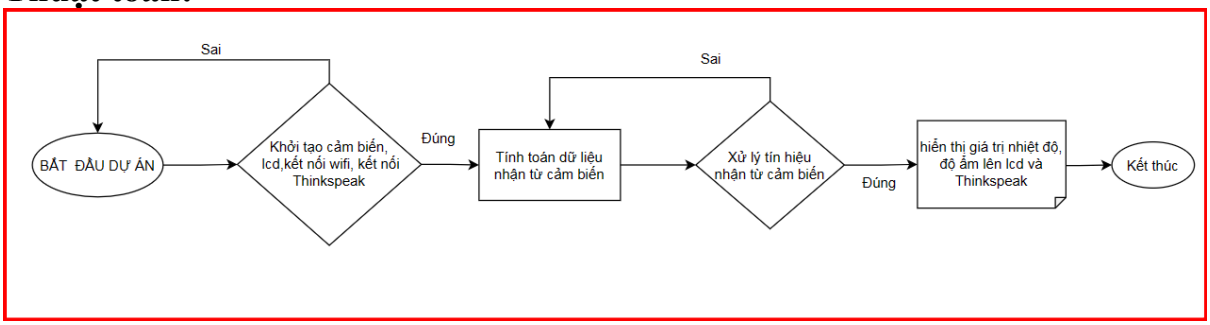
Project 5: Gửi dữ liệu từ cảm biến DHT11 lên Thinkspeak và hiển thị giá trị nhiệt độ, độ ẩm lên LCD.

Bài làm

- **Sơ đồ đấu nối phần cứng:**



- **Thuật toán:**

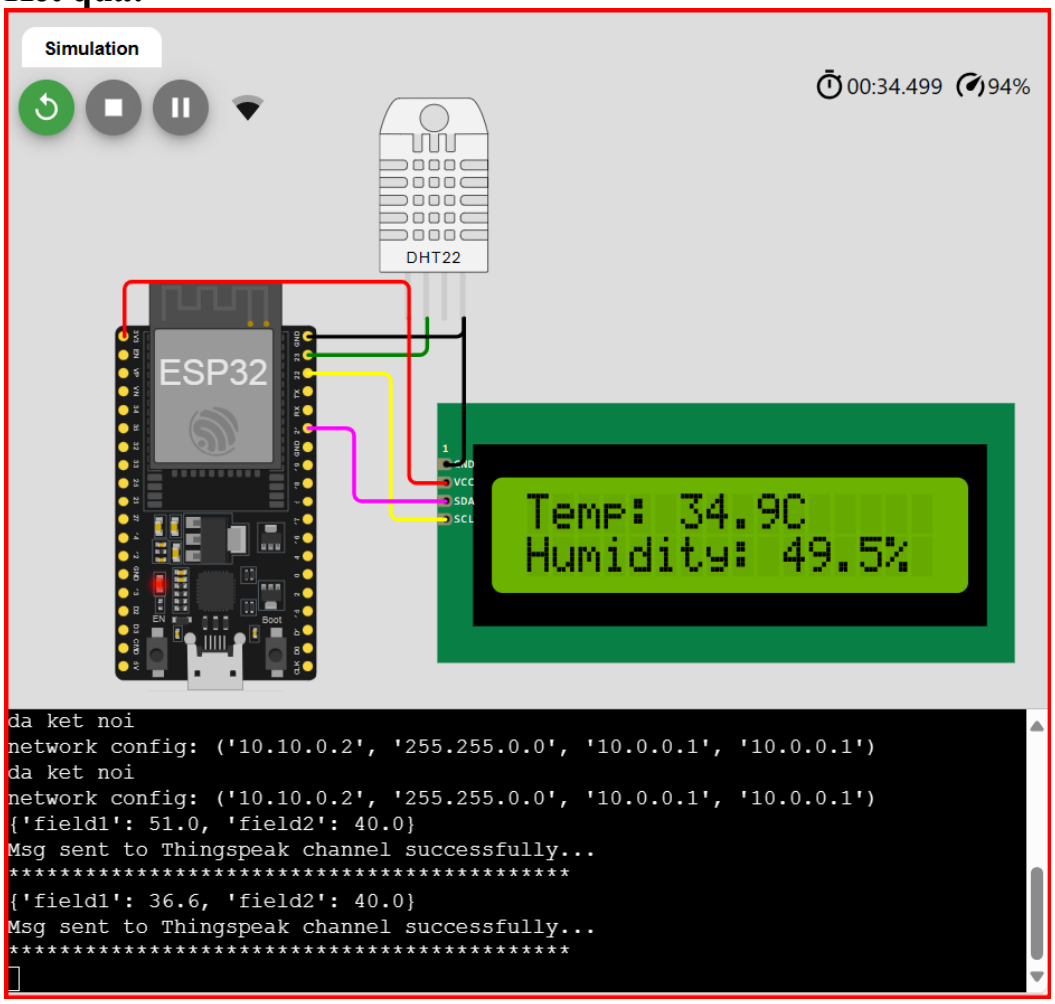


- **Code**

```
from machine import Pin, Timer, SoftI2C
import network
import urequests
import time
import sys
from dht import DHT22
from i2c_lcd import I2cLcd

dht22 = DHT22(Pin(23))


I2C_ADDR = 0x27
totalRows = 2
totalCol = 16
i2c = SoftI2C(scl=Pin(22), sda=Pin(21), freq=10000)
lcd = I2cLcd(i2c, I2C_ADDR, totalRows, totalCol)

# Connect with WiFi
import network
WIFI_SSID = "Wokwi-GUEST"
WIFI_PASSWORD = ""

def connect_wifi():
    wifi = network.WLAN(network.STA_IF)
    wifi.active(True)
    wifi.disconnect()
    wifi.connect(WIFI_SSID,WIFI_PASSWORD)
    if not wifi.isconnected():
        print('dang ket noi...')
        timeout = 0
        while (not wifi.isconnected() and timeout < 10):
            print(10 - timeout)
```

```
            timeout = timeout + 1
            time.sleep(1)
    if(wifi.isconnected()):
        print('da ket noi')
    else:
        print('khong ket noi')
        sys.exit()
    print('network config:', wifi.ifconfig())
connect_wifi()
# Main function
def main():
    connect_wifi()
    # Thingspeak HTTP API Protocol (Connection)
    HTTP_HEADERS = {'Content-Type': 'application/json'}
    THINGSPEAK_WRITE_API_KEY = 'P71Z0G5D2PHSAJLX' //thay đổi
theo api của từng người sử dụng
    while True:
        time.sleep(5)
        # đọc giá trị cảm biến DHT11
        dht22.measure()
        temp = dht22.temperature()
        hum = dht22.humidity()

        # Hiển thị LCD
        lcd.clear()
        lcd.putstr("Temp: {}C\n".format(temp))
        lcd.putstr("Humidity: {}%".format(hum))
        dht_readings = {'field1': temp, 'field2': hum}

        # gửi dữ liệu lên Thingspeak
        request = urequests.post('http://api.thingspeak.com/update?api_key=' +
THINGSPEAK_WRITE_API_KEY,
                                 json=dht_readings, headers=HTTP_HEADERS)
        request.close()

        print(dht_readings)
        print("Msg sent to Thingspeak channel successfully...")
        print("**********************************************")
if __name__ == "__main__":
    main()
```

- **Kết quả:**



- **Link mô phỏng: https://wokwi.com/projects/397602690620212225**

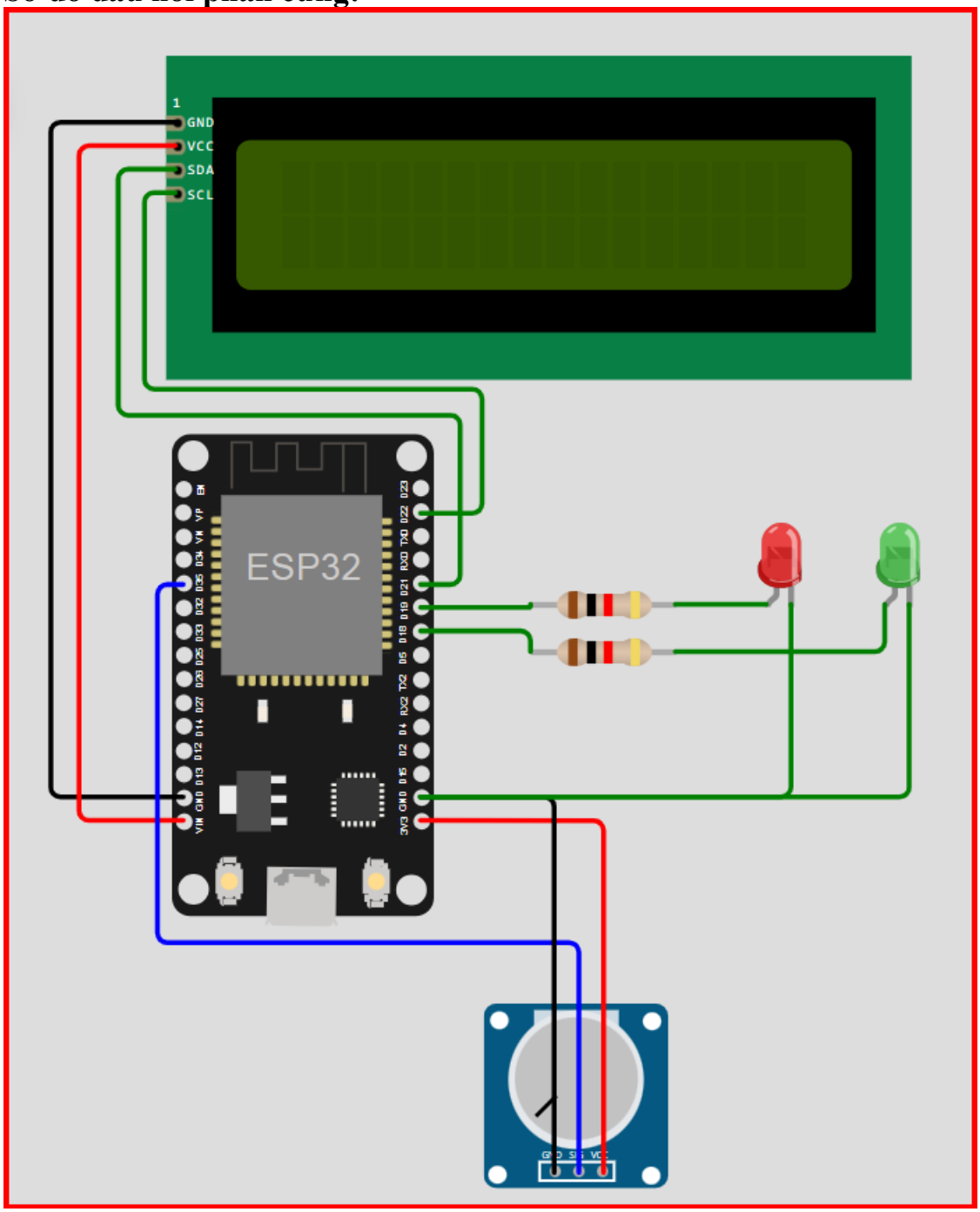
Project 6:

Câu 2:

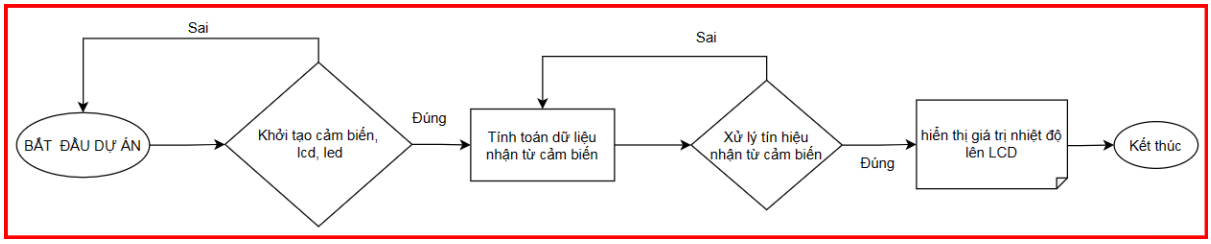
Vẽ sơ đồ ghép nối ESP32 với LCD 1602 và 1 biến trở. Viết chương trình bằng Micropython đọc giá trị đo từ ADC khi vặn thay đổi giá trị của biến trở và hiển thị trên LCD. Kết nối 2 led đơn hiển trị trên và dưới ngưỡng nhiệt độ đặt.

Bài làm

- **Sơ đồ đấu nối phần cứng:**



- **Thuật toán**



- **Code:**

```
from machine import Pin, ADC, I2C
from time import sleep_ms
from lcd_i2c_esp32 import I2cLcd

# Define pin numbers for the LEDs
led1_pin = 19
led2_pin = 18

# Define the I2C address of the LCD
DEFAULT_I2C_ADDR = 0x27

# Initialize I2C communication
i2c = I2C(0, scl=Pin(22), sda=Pin(21), freq=400000)

# Initialize the LCD
lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)

# Initialize ADC for temperature reading
adc = ADC(Pin(34))  # Assuming the temperature sensor is connected to pin 34

# Function to map temperature to LED states
def map_temperature_to_led_states(temperature):
    # Map temperature in the range of 0-100 degrees Celsius to LED states
    if temperature <= 50:
        return (1, 0)  # LED 1 on, LED 2 off
    else:
        return (0, 1)  # LED 1 off, LED 2 on

while True:
    # Read the analog value from the temperature sensor
    temperature_adc = adc.read()
```

```
    # Convert the ADC value to temperature in Celsius (assuming linear relationship)
    temperature_celsius = temperature_adc * (100 / 4095)  # Assuming ADC is 12-bit

    # Map temperature to LED states
    led1_state, led2_state = map_temperature_to_led_states(temperature_celsius)

    # Control LEDs based on states
    led1 = Pin(led1_pin, Pin.OUT)
    led2 = Pin(led2_pin, Pin.OUT)
    led1.value(led1_state)
    led2.value(led2_state)

    # Display temperature on LCD
    lcd.clear()
    lcd.move_to(0, 0)
    lcd.putstr("Temp: {:.1f} C".format(temperature_celsius))

    # Wait for a short duration before repeating the process
    sleep_ms(1000)
```

Project 7:

Câu 1:

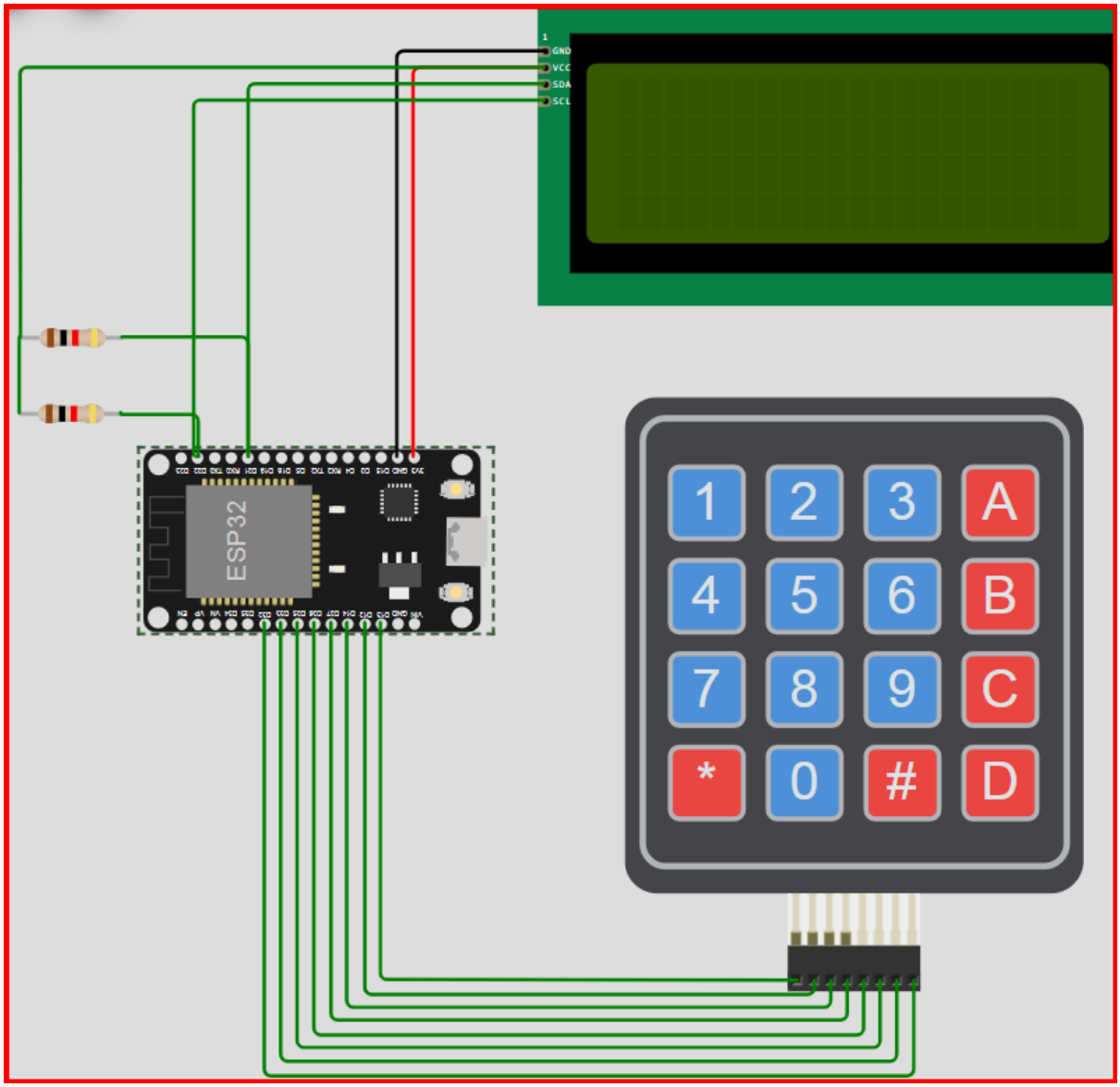
Vẽ sơ đồ ghép nối ESP32 với ma trận phím 4x4 với LCD DM 1602 giao tiếp I2C.

Viết chương trình bằng Micropython đọc và hiển thị trên LCD

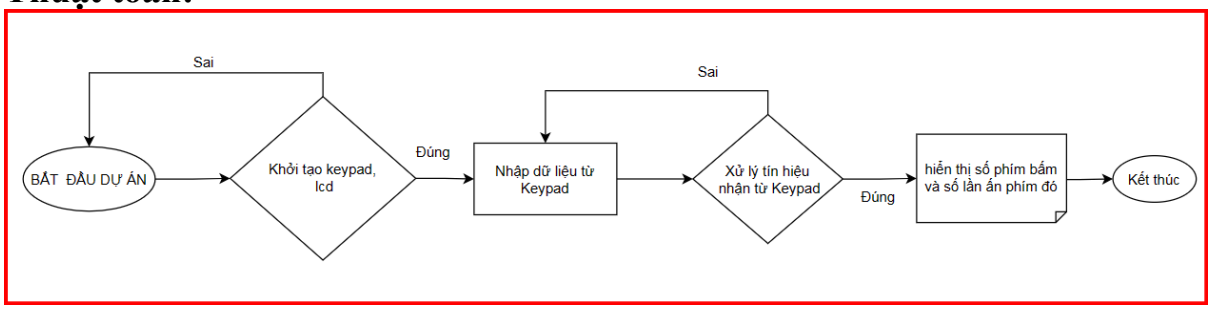
1 - Số thứ tự của phím được bấm

2 – Số lần bấm phím đó.

Bài làm



- **Thuật toán:**



- **Code:**

```
from time import sleep_ms
from machine import Pin, SoftI2C
from i2c_lcd import I2cLcd

######## Definitions ########
```

```
# Define LCD params
AddressOfLcd = 0x27
i2c = SoftI2C(scl=Pin(22), sda=Pin(21), freq=400000) # connect scl to GPIO
22, sda to GPIO 21
lcd = I2cLcd(i2c, AddressOfLcd, 4, 20)

# Define keypad layout
keypad = [
    ['1', '2', '3', 'A' ],
    ['4', '5', '6', 'B' ],
    ['7', '8', '9', 'C' ],
    ['*', '0', '#', 'D' ]
]

# Define the row and column pins
row_pins = [Pin(13, Pin.OUT), Pin(12, Pin.OUT), Pin(14, Pin.OUT),
Pin(27, Pin.OUT)]
col_pins = [Pin(26, Pin.IN, Pin.PULL_UP), Pin(25, Pin.IN, Pin.PULL_UP),
Pin(33, Pin.IN, Pin.PULL_UP), Pin(32, Pin.IN, Pin.PULL_UP)]

# Initialize the row pins to HIGH
for row_pin in row_pins:
    row_pin.value(1)

# Dictionary to keep track of key presses
key_counts = {key: 0 for row in keypad for key in row}

def read_keypad():
    for i in range(4):
        row_pins[i].value(0)
        for j in range(4):
            if col_pins[j].value() == 0:
                sleep_ms(20)  # Debounce delay
                # Check again to avoid false positives
                if col_pins[j].value() == 0:
                    while col_pins[j].value() == 0:
                        pass  # Wait for key release
                    return keypad[i][j]
        row_pins[i].value(1)
    return None

######## Main ########
```

```
while True:
    key = read_keypad()
    if key is not None:
        key_counts[key] += 1
        lcd.clear()
        lcd.move_to(0, 0)
        lcd.putstr("Key pressed: ")
        lcd.move_to(0, 1)
        lcd.putstr(key)
        lcd.move_to(0, 2)
        lcd.putstr("Count: {}".format(key_counts[key]))
```