Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică şi Microelectronică
Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator nr. 7.3
La disciplina “Internetul Lucrurilor”
Tema: Comunicare Internet - MQTT

A efectuat: st. gr. SI-211 Adrian Chihai
A verificat: Valentina Astafi

Chișinău – 2024

1 Definirea problemei

Sa se realizeze o aplicatie ce va implementa comunicatiile intre echipamente dupa cum urmeaza:

1. Realizarea unei aplicatii decomunicare Internet prin protocolul MQTT penru interactiunea cu o resursa Cloud
2. Colectarea datelor de la sensori si trimitere catre un broker MQTT
3. urmarirea mesajelor de la un proker MQTT si setarea starii unui actuator la alegere
4. Datele sunt vizualizate si controlate de la un dashboard Internet *(ThingsBoard sau HiveMQ)*

2 Descrierea funcțiilor

setup()

Această funcție este apelată o singură dată când microcontrollerul ESP32 este repornit sau alimentat. Este folosită pentru a inițializa conexiunea WiFi și clientul MQTT:

- Serial.begin(9600); - Inițializează comunicația serială cu o rată de transfer de 9600 bps, pentru a permite afișarea datelor și mesajelor de diagnostic pe consola serială.
- dht.begin(); - Inițializează senzorul DHT22 pentru a începe citirea datelor de temperatură și umiditate.
- WiFi.begin(ssid, password); - Conectează ESP32 la rețeaua WiFi specificată.
- Bucle (while) pentru a aștepta conectarea la WiFi - Verifică dacă conexiunea la WiFi este stabilită și așteaptă până când acest lucru se întâmplă.
- client.setServer(mqttServer, mqttPort); - Setează adresa serverului MQTT și portul la care clientul ar trebui să se conecteze.
- Bucle (while) pentru a aștepta conectarea la MQTT - Încearcă să se conecteze la brokerul MQTT și așteaptă până când conexiunea este reușită.

loop()

Această funcție este apelată repetat și conține logica principală a programului:

- float humidity = dht.readHumidity(); - Citește umiditatea curentă de la senzorul DHT22.
- float temperature = dht.readTemperature(); - Citește temperatura curentă de la senzorul DHT22.
- Verificare if (client.connected()) - Se asigură că clientul este încă conectat la brokerul MQTT înainte de a încerca să publice date.
- client.publish("v1/devices/me/telemetry", ...) - Publică datele de temperatură și umiditate sub forma unui JSON la topicul specificat pe brokerul MQTT.
- client.loop(); - Menține conexiunea activă cu brokerul MQTT și gestionează orice mesaj care ar putea fi primit sau trebuie să fie trimis.
- delay(2000); - Pauzează execuția programului pentru 2000 de milisecunde (2 secunde), pentru a limita frecvența cu care datele sunt citite și trimise.

3 Diagrama programului

Figura 2.1 funcția *setup*. După ce se termină această funcției, it jumps to the *loop* function.

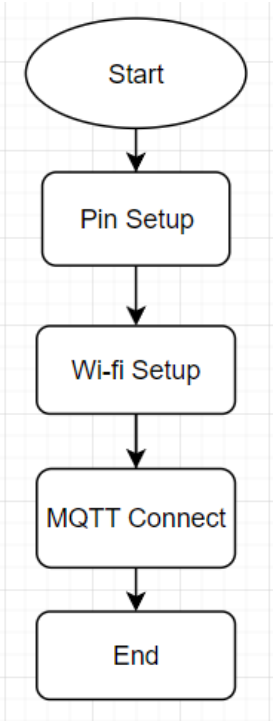

Fig. 2.1. funcția *setup*

Figura 2.2 logica functiei *loop*.

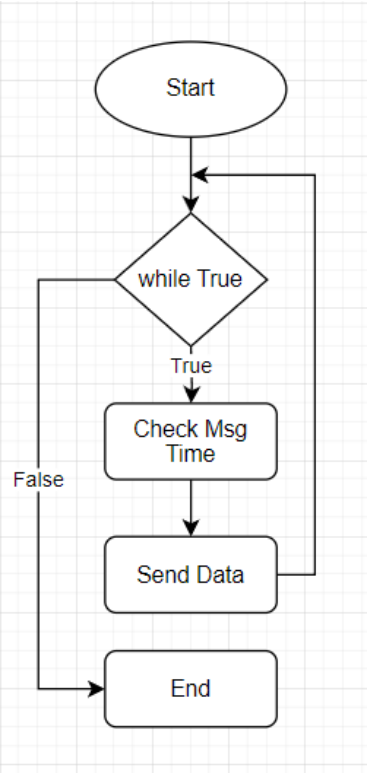

Fig. 2.2. The *loop* function

4 Circuitul realizat

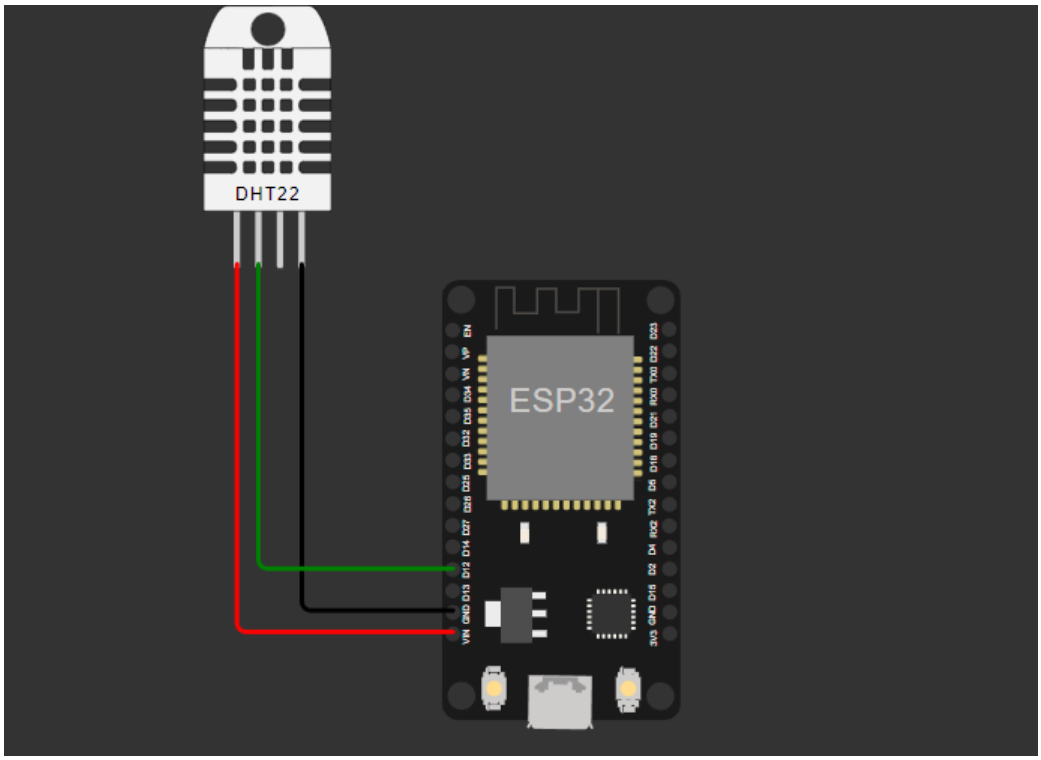

Fig. 2.3. Circuitul asamblat

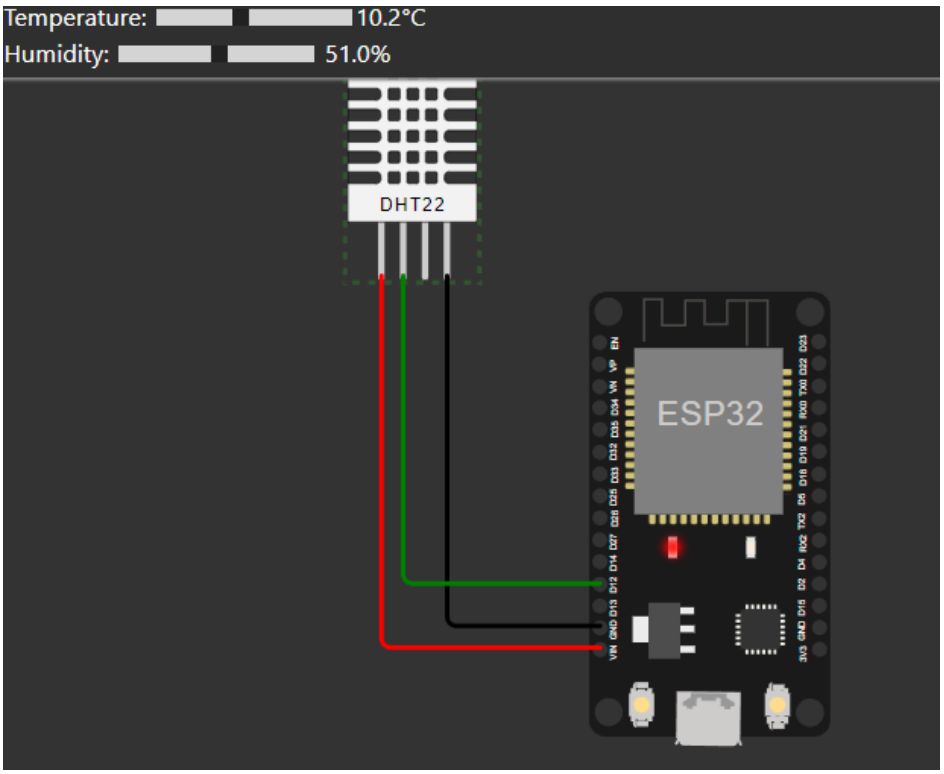

Fig. 2.4. Circuitul rulat

Concluzie

În concluzie, codul prezentat ilustrează eficient utilizarea microcontrolerului ESP32 în cadrul aplicațiilor IoT. Acesta demonstrează cum ESP32 poate fi integrat cu succes pentru a se conecta la rețele Wi-Fi, a comunica cu brokeri MQTT și a gestiona transmiterea datelor de la senzori, precum și recepția de comenzi pentru controlul actuatorilor. Utilizarea protocoalelor Wi-Fi și MQTT îi conferă capacitatea de a facilita monitorizarea și controlul de la distanță, esențiale pentru implementările eficiente în diverse scenarii IoT.

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

#define DHTPIN 12
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

const char* ssid = "yourSSID";
const char* password = "yourPassword";
const char* mqttServer = "mqtt.thingsboard.io";
const int mqttPort = 1883;
const char* mqttUser = "yourDeviceToken";
WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(9600);
  dht.begin();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..");
  }
  client.setServer(mqttServer, mqttPort);
  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect("ESP32Client", mqttUser, NULL)) {
      Serial.println("connected");
    } else {
      Serial.print("failed with state ");
      Serial.print(client.state());
      delay(2000);
    }
  }
}

void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  if (client.connected()) {
    client.publish("v1/devices/me/telemetry", String("{\"temperature\":").c_str() +
String(temperature) + ",\"humidity\":" + String(humidity) + "}").c_str());
  }
  client.loop();
  delay(2000);
}
```

Bibliografie

1. TINKERCAD: *Arduino Simulator*. Online Simulator, © 2024 Autodesk, Inc [21.02.2024], Link: https://www.tinkercad.com/dashboard

2. WOKWI: *Arduino Simulator and Tutorials*. Arduino Examples, ⓒ 2019-2023 CodeMagic LTD [21.02.2024], Link: https://wokwi.com/projects

3. LUCID: *Flow Chart Maker*. Software for Creating Flow Charts and Block Diagrams, ⓒ 2023 [21.02.2024], Link: https://lucid.app

4. ARDUINO DOCS: *Arduino Documentation and Examples*. Documentation, Examples and Libraries, ⓒ2024 Arduino [21.02.2024], Link: https://docs.arduino.cc/built-in-examples/