

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

**Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці**

ЗВІТ

з лабораторної роботи №5

**з дисципліни «Розробка застосунків інтернету речей та
сенсорних мереж»**

**Тема: «Розробка та налаштування IoT-пристрою на базі
ESP32 з використанням протоколу MQTT»**

Варіант №17

Виконав:

Студент групи ТР-12

Ковальов Олександр Олексійович

Дата здачі: 05.03.2025

Мета роботи. Ознайомлення з основами роботи з мікроконтролером ESP32 і реалізацією IoT-функціональності через протокол MQTT. Продемонструвати принцип підключення датчиків та інших пристроїв до ESP32. Здійснення програмування мікроконтролера для збору, аналізу та відправки даних через протокол MQTT. Організація віддаленого доступу до систем керування мікроконтролера.

Індивідуальне завдання:

1) Розробка схеми:

- створити схему з'єднання мікроконтролера ESP32, датчика руху, датчика освітленості та LED-лампи. Врахувати необхідні резистори, джерела живлення та інші компоненти.

2) З'єднання з MQTT WebSocket:

- забезпечити з'єднання мікроконтролера з MQTT-брокером через протокол WebSocket;
- налаштувати мікроконтролер для підписки на тему вмикання/вимикання та отримання статусу віддаленого керування.

3) Датчик відстані:

- налаштувати датчик відстані на роботу в режимі, коли освітлення низьке;
- здійснювати зчитування та аналіз даних датчика відстані.

4) Логіка роботи:

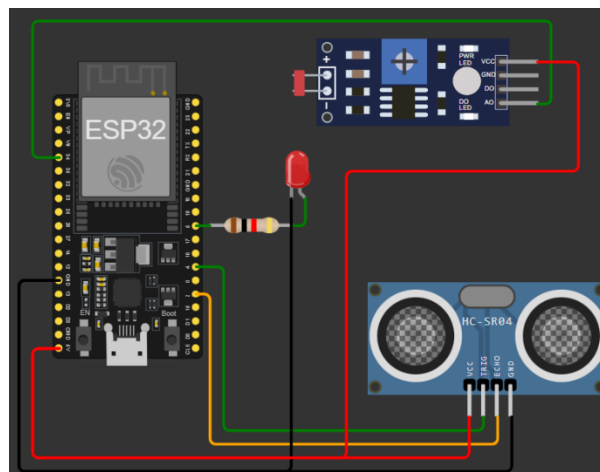
- якщо режим роботи встановлений на «ввімкнено» і датчик відстані виявляє об'єкт на певній відстані, передавати тривогу до MQTT-клієнта;
- якщо об'єкт віддаляється після тривоги, передати повідомлення про нормалізацію ситуації;
- якщо режим роботи встановлений на «вимкнено» та освітлення недостатнє, передати до клієнта помилку і пораду ввімкнути пристрій.

5) Керування режимом:

- реалізувати зміну режиму роботи (ввімкнено/вимкнено) через MQTT-клієнта, використовуючи статус «on» або «off».

Хід роботи.

Була побудована схема з мікроконтролером ESP32, датчиком відстані HC-SR04, фоторезистором, резистором:



Далі, для роботи з брокером, був створений акаунт на hivemq.cloud:

HiveMQ Cloud Profile

Manage your account in one place.

Information

Details about your HiveMQ Cloud account

First Name

Oleksandr

Last Name

Kovalov

Був створений сервер та користувачі:

NAME	PERMISSION	ACTIONS
SuperClient	PUBLISH_SUBSCRIBE	
SlaveClient	PUBLISH_SUBSCRIBE	
<div>Add new credential</div>		

Була створена підписка на топик:

Topic Subscriptions ¹

Subscribe to topics to receive messages from the HiveMQ cluster. You can also set the Quality of Service (QoS) for each topic. The higher the QoS, the more reliable the message delivery is. You can always subscribe to the (#) wildcard to receive all messages.

TOPIC	QOS	ACTIONS
Lab5_Temperature_Topic	QoS: 0	
<input type="text" value="#"/>	<div>Subscribe</div>	

Далі був написаний код призначений для роботи з мікроконтролером ESP32, який підключається до Wi-Fi, взаємодіє з MQTT-брокером через захищене з'єднання TLS і використовує сенсори для контролю навколишнього середовища.

На початку підключаються необхідні бібліотеки: WiFi.h для з'єднання з мережею, WiFiClientSecure.h для безпечного підключення, PubSubClient.h для роботи з протоколом MQTT і ArduinoJson.h для роботи з JSON-повідомленнями.

Далі оголошуються константи для підключення до Wi-Fi і MQTT-брокера. Встановлюються параметри мережі, такі як SSID і пароль, а також дані для аутентифікації на MQTT-сервері, включно з адресою брокера, портом і логіном із паролем.

Оголошуються змінні для роботи з сенсорами: фоторезистором, ультразвуковим датчиком відстані та світлодіодом-попередженням. Також вводяться дві булеві змінні turned і alert, які зберігають стан системи.

Функція callback() викликається при отриманні MQTT-повідомлення. Вона аналізує отримане JSON-повідомлення та змінює стан змінної turned залежно від отриманого значення ("on" або "off").

У функції setup() відбувається ініціалізація серійного зв'язку, встановлення режимів роботи пінів і підключення до Wi-Fi. Після успішного підключення відбувається налаштування захищеного з'єднання MQTT і підписки на вказану тему. Якщо з'єднання з брокером не вдається, ESP32 намагається підключитися повторно кожні 5 секунд.

У головному циклі loop() виконується постійний обмін даними з MQTT-брокером. Читається значення з фоторезистора, і якщо воно перевищує порогове значення (2000) та система увімкнена, активується світлодіод. Далі здійснюється вимірювання відстані за допомогою ультразвукового сенсора. Якщо об'єкт знаходиться ближче за 200 см, надсилається MQTT-повідомлення про присутність когось поруч. Якщо відстань більша за 200 см, система відправляє повідомлення, що все гаразд. Якщо ж система вимкнена, надсилається сповіщення про її деактивацію.

Кожен цикл завершується невеликою затримкою у 5 секунд перед наступним вимірюванням і обробкою MQTT-повідомлень.

Код:

```
#include <WiFi.h>
#include <WiFiClientSecure.h> // Secure TLS client
#include <PubSubClient.h>
#include <ArduinoJson.h>

// Wi-Fi Parameters
const char* WIFI_SSID = "Wokwi-GUEST";
const char* WIFI_PASSWORD = "";

// MQTT Server Parameters (TLS)
const char* MQTT_CLIENT_ID = "ESPLab5";
const char* MQTT_BROKER = "9635d68c61b74e0a9e7adb108b924247.s1.eu.hivemq.cloud";
const char* MQTT_USER = "SlaveClient";
const char* MQTT_PASSWORD = "RootP@ss1";
const char* MQTT_TOPIC = "Lab5_Temperature_Topic";
const int MQTT_PORT = 8883; // Secure MQTT Port

WiFiClientSecure espClient;
PubSubClient client(espClient);

const int photoresistorPin = 34;
const int triggerPin = 4;
const int echoPin = 2;
const int lightWarningBubblePin = 5;
bool turned = true;
bool alert = false;
```

```

// Callback function for incoming MQTT messages
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message received on topic: ");
    Serial.println(topic);

    String message;
    for (int i = 0; i < length; i++) {
        message += (char)payload[i];
    }
    Serial.print("Payload: ");
    Serial.println(message);

    DynamicJsonDocument doc(256);
    DeserializationError error = deserializeJson(doc, message);

    if (error) {
        Serial.print(F("Failed to parse JSON: "));
        Serial.println(error.c_str());
        return;
    }

    String stringtext = doc["Status"];
    if (stringtext == "off") {
        turned = false;
    } else if (stringtext == "on") {
        turned = true;
    }
}

void setup() {
    Serial.begin(115200);
    pinMode(photoresistorPin, INPUT);
    pinMode(triggerPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(lightWarningBubblePin, OUTPUT);

    // Connect to WiFi
    Serial.print("Connecting to WiFi");
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(100);
    }
    Serial.println(" Connected!");

    // Secure connection for TLS
    espClient.setInsecure(); // Skips certificate validation (use only if needed)

    client.setServer(MQTT_BROKER, MQTT_PORT);
    client.setCallback(callback); // Set the callback function for incoming messages
    client.setKeepAlive(60);

    Serial.print("Connecting to MQTT server... ");
    while (!client.connected()) {
        if (client.connect(MQTT_CLIENT_ID, MQTT_USER, MQTT_PASSWORD)) {
            Serial.println("Connected!");
            client.subscribe(MQTT_TOPIC); // Subscribe to MQTT topic
        } else {
            Serial.print("Failed, rc=");
            Serial.print(client.state());
            Serial.println(" Retrying in 5 seconds...");
            delay(5000);
        }
    }
}

void loop() {
    client.loop();
    String message = "";

    if (analogRead(photoresistorPin) > 2000 && turned) {
        digitalWrite(lightWarningBubblePin, HIGH);

        // Trigger ultrasonic sensor
        digitalWrite(triggerPin, LOW);
        delay(2);
        digitalWrite(triggerPin, HIGH);
    }
}

```

```

delay(10);
digitalWrite(triggerPin, LOW);

long value = pulseIn(echoPin, HIGH);
double meters = value / 58.0; // Convert to centimeters

Serial.print("Distance: ");
Serial.print(meters);
Serial.println(" cm");

if (meters > 200) {
  digitalWrite(lightWarningBubblePin, LOW);
  if (alert) {
    alert = false;
    message = "{\"Alert\":\"All Good!\"}";
    client.publish(MQTT_TOPIC, message.c_str(), true);
    Serial.print("Broker Status:");
    Serial.print(client.state());
    Serial.println();
  }
} else {
  alert = true;
  message = "{\"Alert\":\"Someone Close!\"}";
  client.publish(MQTT_TOPIC, message.c_str(), true);
  Serial.print("Broker Status:");
  Serial.print(client.state());
  Serial.println();
} else if (analogRead(photoresistorPin) > 2000) {
  message = "{\"Alert\":\"System disabled!\"}";
  client.publish(MQTT_TOPIC, message.c_str(), true);
  Serial.print("Broker Status:");
  Serial.print(client.state());
  Serial.println();
}

client.loop();
delay(5000);
}

```

Підключення до Wi-Fi та брокеру:

```

Connecting to WiFi..... Connected!
Connecting to MQTT server... Connected!
Message received on topic: Lab5_Temperature_Topic
Payload: {"Alert":"All Good!"}

```

Для тесту було відправлено повідомлення з веб-клієнту:

Send Message ¹

If you cannot see any messages, make sure you are subscribed to the correct topics. You can always subscribe to the (#) wildcard to receive all messages.

Message *

{"Alert":"Hello World! Lab 5 Alex Kovalov"}

Topic *

Lab5_Temperature_Topic

QoS *

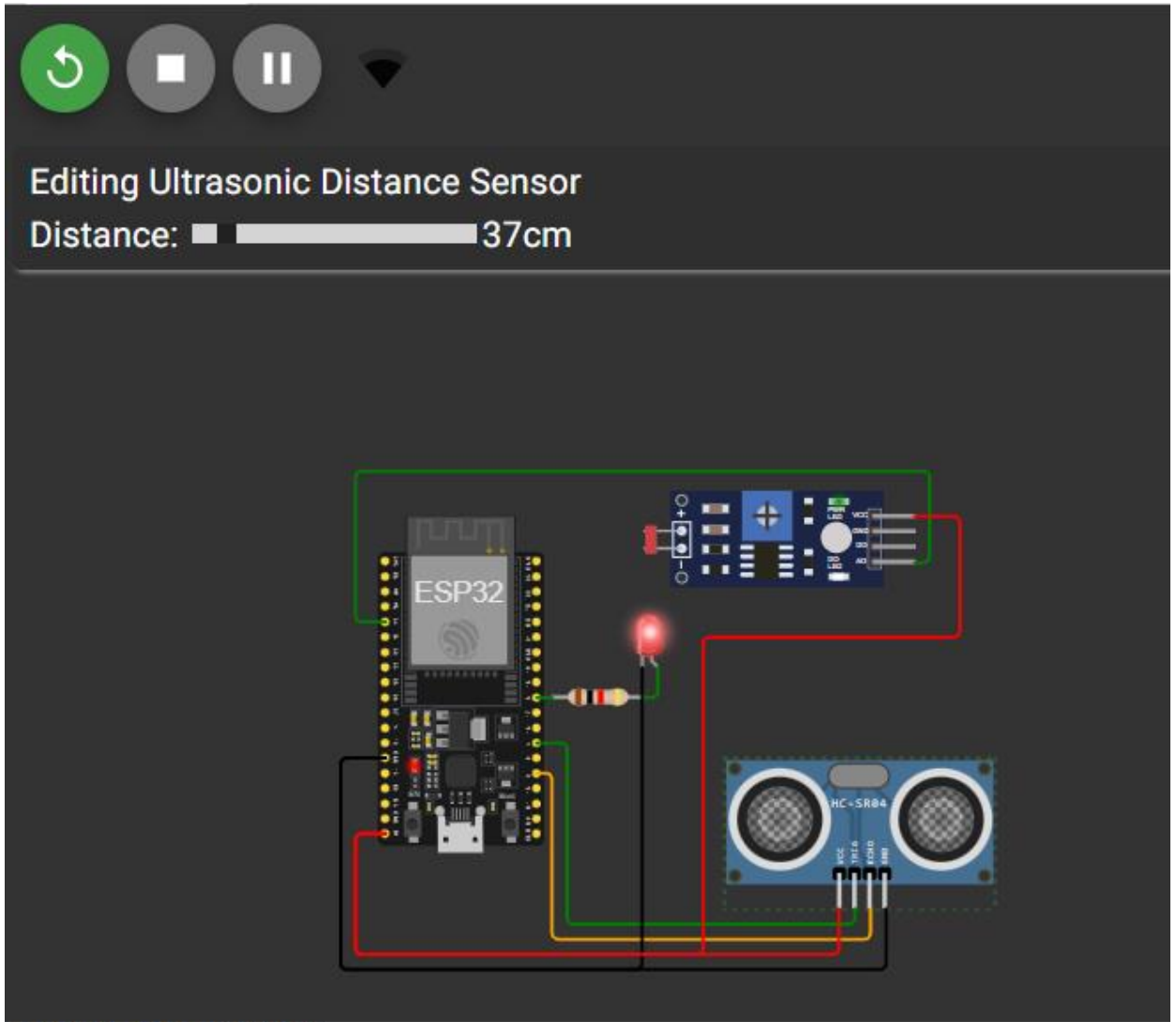
QoS: 0

Send Message

Отримане повідомлення:


```
Failed to parse JSON: InvalidInput  
Message received on topic: Lab5_Temperature_Topic  
Payload: {"Alert":"Hello World! Lab 5 Alex Kovalov"}
```

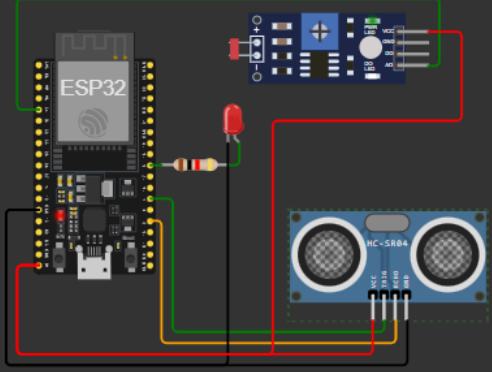
Перевірка роботи системи при освітленні 0.4 lux, 37 cm (вмикається лампочка, надсилається повідомлення що «хтось» близько):



```
entry 0x400805dc  
Connecting to WiFi..... Connected!  
Connecting to MQTT server... Connected!  
Message received on topic: Lab5_Temperature_Topic  
Payload: {"Alert":"All Good!"}  
Distance: 37.52 cm  
Broker Status:0
```

Те ж саме освітлення, «віддалення»:

Editing Ultrasonic Distance Sensor
Distance:  217cm



Message received on topic: Lab5_Temperature_Topic
Payload: {"Alert":"Someone Close!"}
Distance: 220.03 cm
Broker Status:0
Message received on topic: Lab5_Temperature_Topic
Payload: {"Alert":"All Good!"}
Distance: 219.97 cm

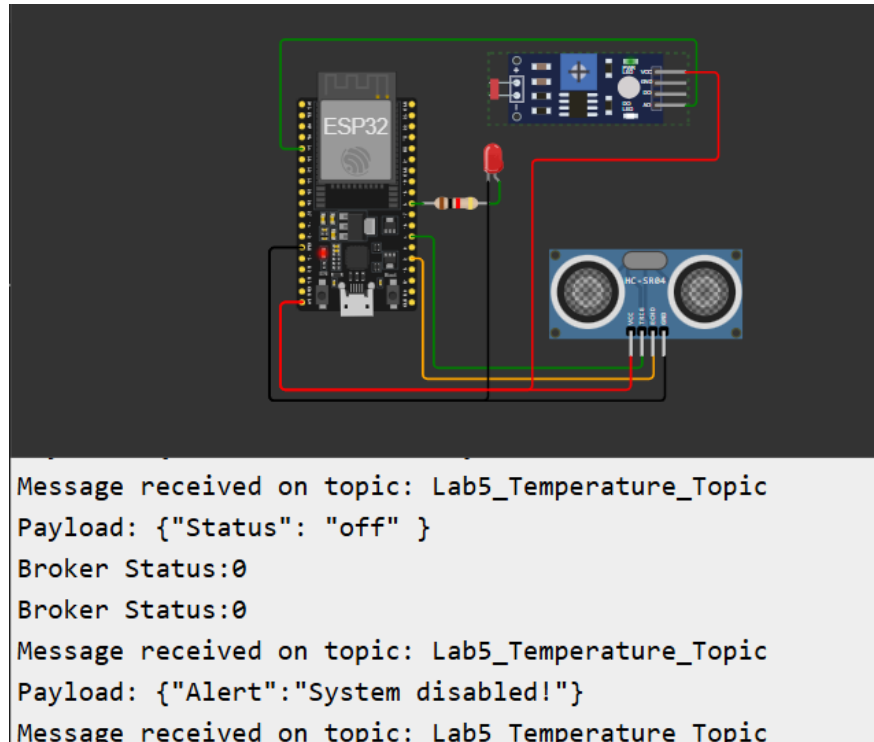
Повідомлення на клієнті:

4	Topic: Lab5_Temperature_Topic	QoS: 0
	{"Alert":"Someone Close!"}	
5	Topic: Lab5_Temperature_Topic	QoS: 0
	{"Alert":"All Good!"}	

Тестування «вимикання» системи. Перше – надіслане повідомлення, друге – отримане:

0	Topic: Lab5_Temperature_Topic	QoS: 0
	{"Status": "off" }	
1	Topic: Lab5_Temperature_Topic	QoS: 0
	{"Alert":"System disabled!"}	

Відповідно, система не спрацює при низькому освітленні та близькій відстані, бо вона вимкнута:



Висновок: У результаті виконання лабораторної роботи було здобуто практичні навички роботи з мікроконтролером ESP32, а також вивчено принципи інтеграції IoT-функціональності через протокол MQTT. Завдяки виконанню індивідуального завдання вдалося створити схему підключення мікроконтролера до різних пристроїв, таких як датчик руху, датчик освітленості та LED-лампа, а також налаштувати відповідні компоненти для забезпечення їх правильного функціонування.

Процес з'єднання мікроконтролера з MQTT-брокером через WebSocket протокол виявився ефективним для організації двостороннього зв'язку, що дозволяє здійснювати віддалене керування пристроєм. Програмування мікроконтролера для зчитування та аналізу даних від датчика відстані дало змогу здійснювати автоматичне відправлення тривоги при виявленні об'єктів на певній відстані, а також повідомлень про нормалізацію ситуації, коли об'єкт віддаляється.

Реалізація змін режимів роботи пристроїв за допомогою MQTT-клієнта дозволила підвищити зручність керування системою, забезпечивши можливість включати та вимикати систему через віддалений доступ. Усі поставлені завдання були виконані успішно, що свідчить про правильне освоєння основ роботи з ESP32 та налаштування IoT-систем на основі протоколу MQTT.

Контрольні питання:

1. Які основні функції мікроконтролера ESP32?

Основні функції мікроконтролера ESP32 включають обробку даних, збирання інформації з датчиків, управління периферійними пристроями, підключення до бездротових мереж Wi-Fi та Bluetooth, а також взаємодію з іншими пристроями через різноманітні протоколи, такі як MQTT. ESP32 є

високопродуктивним мікроконтролером з великою кількістю вбудованих функцій, що дозволяють використовувати його в різноманітних IoT-проектах.

2. *Що таке MQTT-протокол і для чого він використовується в IoT?*

MQTT (Message Queuing Telemetry Transport) – це протокол обміну повідомленнями, який використовується для з'єднання пристроїв в Інтернеті речей (IoT). Його основна мета – ефективна передача повідомлень між клієнтами (наприклад, мікроконтролерами) через брокера. MQTT забезпечує надійну і легку передачу даних навіть при слабких з'єднаннях, тому часто використовується для підключення датчиків і пристроїв, що передають невеликі обсяги даних на великі відстані.

3. *Які компоненти використовуються при підключенні датчиків до ESP32?*

Для підключення датчиків до ESP32 використовуються різноманітні компоненти, такі як резистори для налаштування рівнів сигналу, джерела живлення для забезпечення енергопостачання мікроконтролера та датчиків, а також перехідники і модулі для взаємодії з ESP32, наприклад, для датчиків руху, освітленості або відстані. Крім того, можуть бути використані різні роз'єми та плати для зручного з'єднання.

4. *Як налаштувати з'єднання між ESP32 і MQTT-брокером?*

Для налаштування з'єднання між ESP32 і MQTT-брокером необхідно налаштувати відповідну бібліотеку в програмному середовищі (наприклад, використовуючи бібліотеки для ESP32).

5. *Які параметри необхідні для створення підключення до MQTT-брокера?*

Для створення підключення до MQTT-брокера необхідно вказати кілька параметрів: адреса MQTT-брокера (IP-адреса або доменне ім'я), порт, через який відбувається підключення (найпоширеніший порт – 1883 для стандартного MQTT, або 8084 для WebSocket), а також дані для авторизації, такі як ім'я користувача та пароль, якщо вони використовуються.

6. *Як працює підписка на топик у MQTT?*

Підписка на топик в MQTT дозволяє пристрою отримувати повідомлення, що надсилаються на цей топик. Після підключення до MQTT-брокера, клієнт (наприклад, ESP32) може підписатися на один або кілька топиків, вказавши їх у налаштуваннях підключення. Коли брокер отримує повідомлення, воно автоматично надсилається всім клієнтам, що підписані на відповідний топик, що дозволяє зручно здійснювати віддалене керування або обмін даними між пристроями.