**Національний технічний університет України**

**«Київський політехнчний 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»**

Варіант №16

Виконав:

студент групи ТР-23

Ровний Г.О.

Дата здачі: 19.10.2025

КИЇВ – 2025

**Мета роботи:** Ознайомлення з основами роботи з мікроконтролером ESP32 і реалізацією IoT-функціональності через протокол MQTT[6]. 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.

**Поставлене 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 dвімкнути пристрій. 104

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

**Результат виконання роботи**

Для роботи буде використано середовище Wokwi. Для побудови схеми необхідно розмістити та з’єднати на схемі мікроконтролер esp32, 112 — датчик відстані HC-SR04 та фоторезистор для відслідковування освітлення. Отримуємо наступну схему:

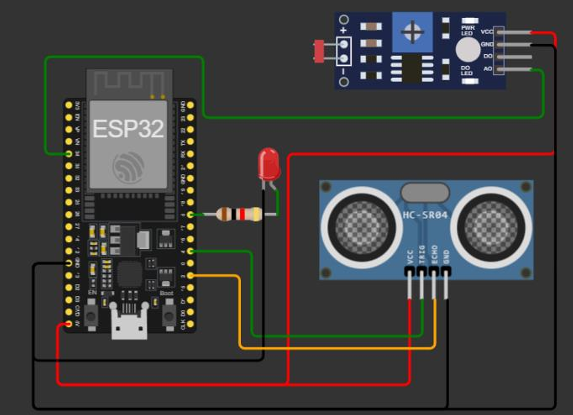


Рис.1.1. Схема для роботи

Для наступного етапу виконання лабораторної роботи необхідно установити з’єднання мікроконтролера з MQTT-брокером, який є онлайн клієнтом. На підготовчому етапі необхідно ініціалізувати змінні, які будуть використовуватися на подальших етапах і заповнити їх відповідними даними:

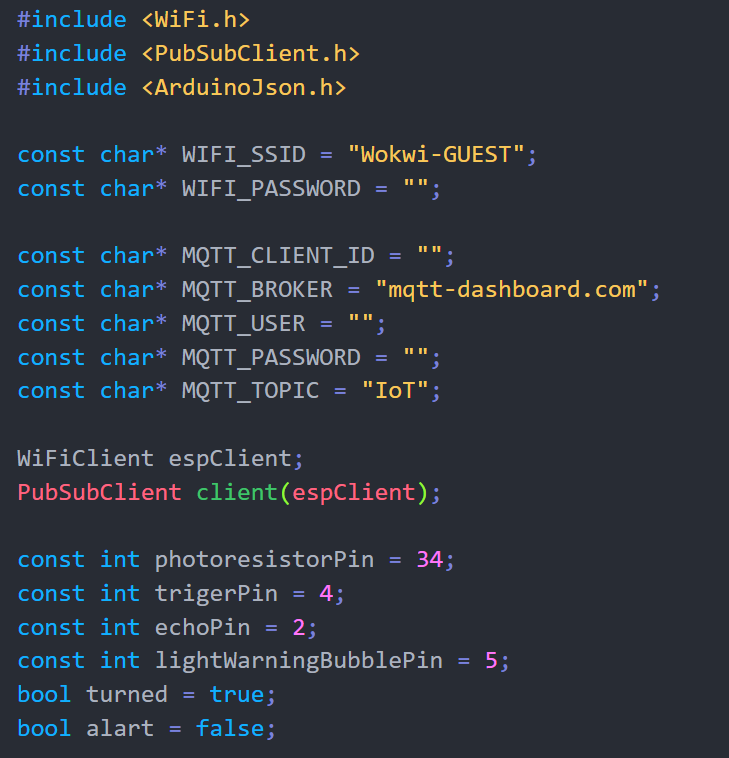


Рис.1.2 Частина коду для приєднання до мережі

Змінні для підключення до мережі Wi-Fi:

— SSID = "Ім’я Мережі";

— PASSWORD = "Пароль від мережі";

Змінні для підключення і налаштування до віддаленого MQTT-брокера:

— ID = "Унікальний ідентифікатор користувача";

— BROKER\_IP = "IP-адреса брокера";

— BROKER\_PORT = “Порт брокера”;

— USER = “Ім’я користувача”;

— PASSWORD = “Пароль користувача”;

— TOPIC = “Ім’я категорії для підпису на сповіщення”.

Наступним етапом буде ініціалізація об’єктів класів WiFiClient і PubSubClient, які мають інструментарій взаємодії. Для створеного екземпляра класу WiFiClient за допомогою методу класу потрібно ініціалізувати підключення до мережі. Це можна зробити шляхом виконання таких дій: — почати операцію з’єднання за допомогою методу класу begin(SSID, PASSWORD); — перевірити статус підключення за допомогою методу класу status().

Оскільки підключення до мережі вже завершено, переходимо до підключення до самого брокера:

void setup() {

  Serial.begin(115200);

  pinMode(photoresistorPin, INPUT);

  pinMode(trigerPin, OUTPUT);

  pinMode(echoPin, INPUT);

  pinMode(lightWarningBubblePin, OUTPUT);

  Serial.print("Connecting to WiFi");

  WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

  while (WiFi.status() != WL\_CONNECTED) {

    Serial.print(".");

    delay(100);

  }

  Serial.println(" Connected!");

  client.setServer(MQTT\_BROKER, 1883);

  client.setCallback(callback);

  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);

    } else {

      Serial.print("Failed, rc=");

      Serial.print(client.state());

      Serial.println(" Retrying in 5 seconds...");

      delay(5000);

    }

  }

}

Послідовні дії для успішного встановлення з’єднання включають:

— встановлення IP-адреси бажаного сервера: set\_server(BROKER, BROKER\_PORT);

— при необхідності встановлення функції підписки на оновлення брокера: setCallback(callback), де callback — ім’я функції, яка буде визиватися при викликові делегата оновлень брокера. Сама функція має приймати такі параметри: char\* topic, byte\* payload, unsigned int length;

— проведення підключення до брокера за допомогою методу класу: connect(ID, USER, PASSWORD);

— підписання на оновлення брокера: subscribe(TOPIC);

— для оновлення інформації про стан брокера використовується метод класу loop();

— для надсилання повідомлення до брокера використовується функція класу: publish(MQTT\_TOPIC, message), де message — це повідомлення. Після успішного підписання на оновлення брокера, кожного разу при внесенні нових даних буде викликана функція callback. Її мета — зчитати і використовувати дані брокера. Для отримання та обробки даних рекомендується використовувати конвертацію через JSON-розмітку.

Для налаштування логіки датчика спочатку необхідно зчитувати отримані дані з датчика освітлення. Слід пам’ятати, що ESP32 конвертує аналоговий сигнал до 12-бітного вигляду. Порівняти вхідне значення з допустимим граничним значенням освітлення і перевірити, чи доступне виконання мікроконтролеру. Наступним кроком буде взаємодія з датчиком відстані. Необхідно надіслати до датчика короткий сигнал і виміряти час відгуку за допомогою методу pulseIn(Pin, MODE), де pin — це номер порту, mode — шуканий рівень сигналу між LOW і HIGH. Конвертувати отримане значення часу до відстані за допомогою формули заданої у лаборатнорній роботі.

Отримаємо наступний результат:

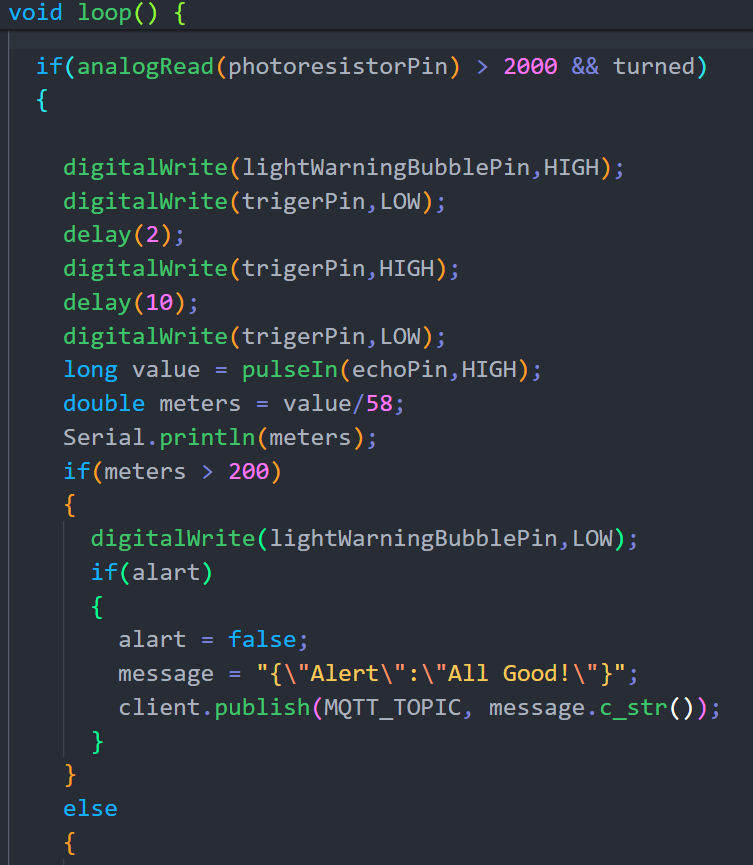


Рис. 1.3. Код для взаємодії з сенсорами

Для реалізації керування контролером створимо змінну, яка буде відповідати за дозвіл роботи мікроконтролера. При публікації брокером оновлення статусу «off» чи «on», контролер має зчитувати дані та перемикати змінну до відповідного режиму:

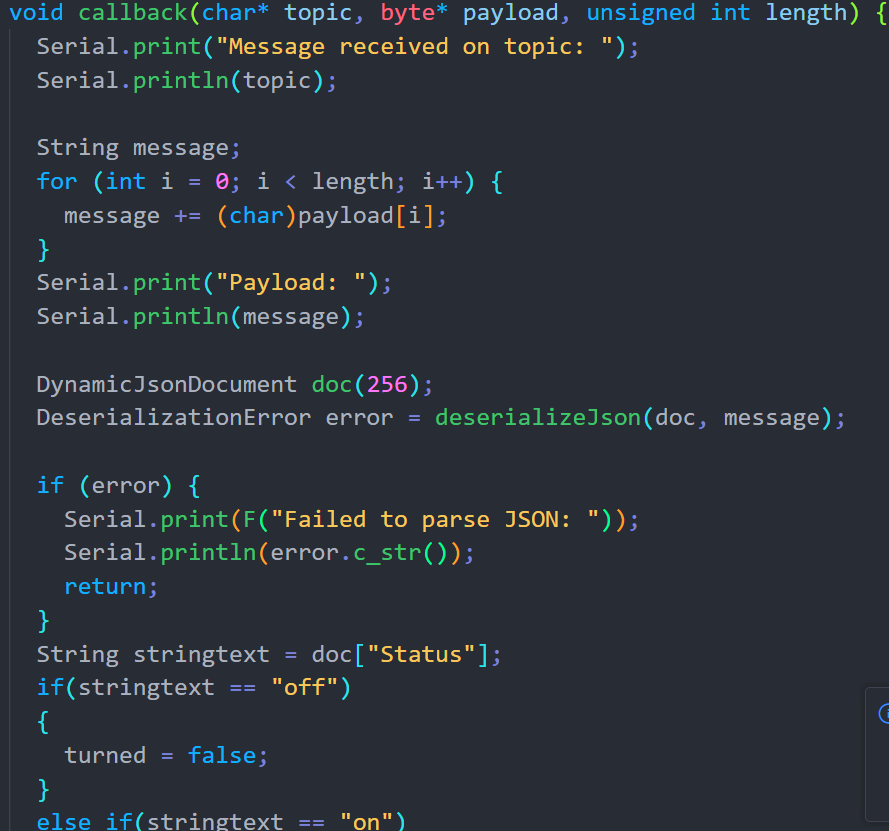
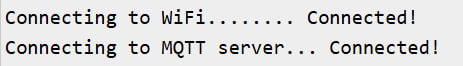


Рис. 1.4. Приклад написання callback функції

У консолі відбувається підключення до брокера та WiFi:



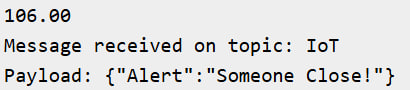


Рис. 1.5. Підключення до брокера з WIFI

На брокер надходять відповідні повідомлення:

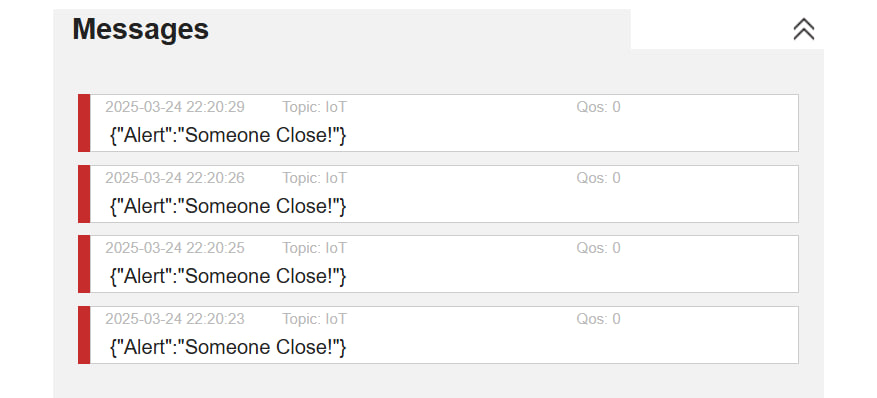


Рис. 1.6. Надходження повідомлень на брокер повідомлень

Якщо на датчику відстані значення більші за 200, у консоль та на брокер будуть надходити відповідні повідомлення:

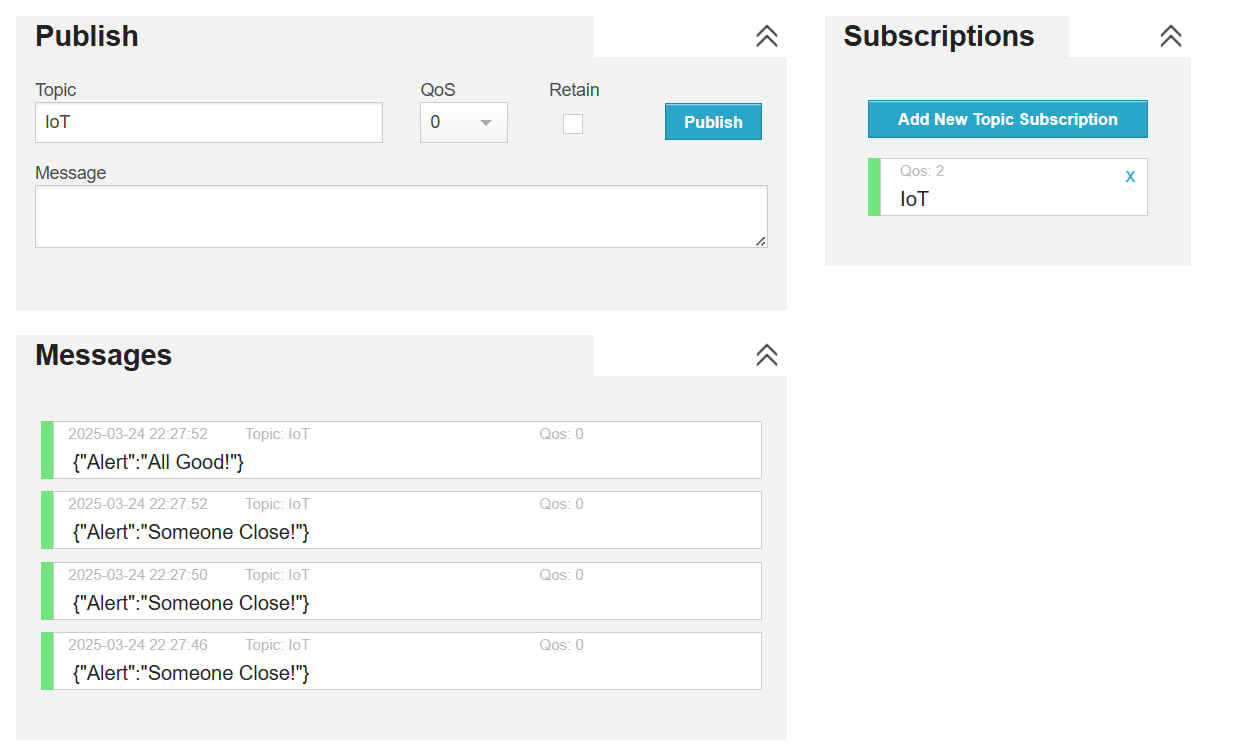
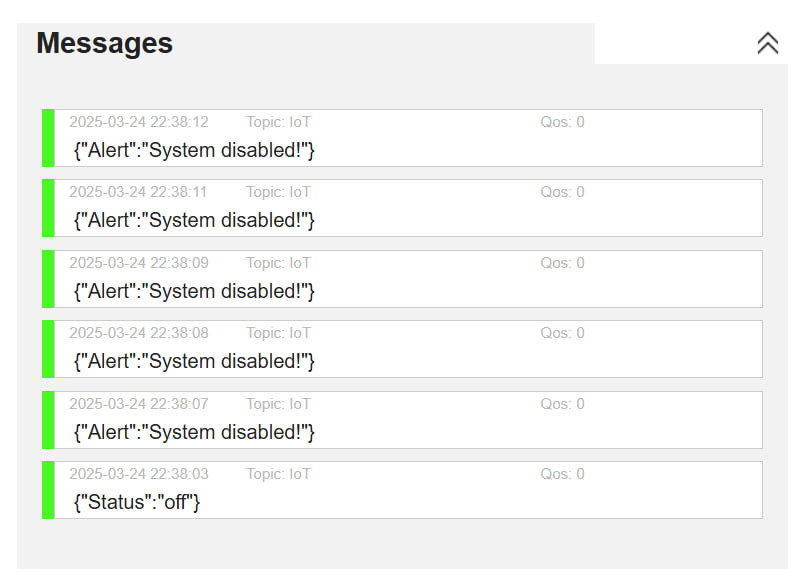


Рис. 1.7. Приклад роботи датчика при відповідних параметрах

Якщо вимкнути систему, надіславши відповідне повідомлення, будуть надходити відповідні повідомлення в консоль в тому числі:



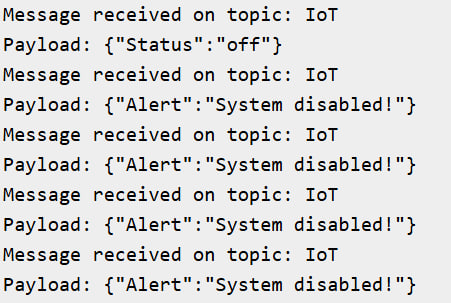


Рис. 1.8. Приклад роботи вимкненої системи

**Висновок:**

У результаті виконання лабораторної роботи було налаштовано IoT-пристрій на базі ESP32 з використанням протоколу MQTT. На основі цього було створено схему підключення датчиків руху, освітленості та відстані, а також реалізовано керування світлодіодом. Програмна частина забезпечила підключення мікроконтролера до мережі Wi-Fi, взаємодію з MQTT-брокером та обробку отриманих даних. Реалізований алгоритм дозволяє контролювати стан датчиків та реагувати на зміну умов, передаючи відповідні повідомлення на брокер.

**Додаток 1 «Лістинг програми»**

#include <WiFi.h>

#include <PubSubClient.h>

#include <ArduinoJson.h>

const char\* WIFI\_SSID = "Wokwi-GUEST";

const char\* WIFI\_PASSWORD = "";

const char\* MQTT\_CLIENT\_ID = "";

const char\* MQTT\_BROKER = "mqtt-dashboard.com";

const char\* MQTT\_USER = "";

const char\* MQTT\_PASSWORD = "";

const char\* MQTT\_TOPIC = "IoT";

WiFiClient espClient;

PubSubClient client(espClient);

const int photoresistorPin = 34;

const int trigerPin = 4;

const int echoPin = 2;

const int lightWarningBubblePin = 5;

bool turned = true;

bool alart = false;

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(trigerPin, OUTPUT);

  pinMode(echoPin, INPUT);

  pinMode(lightWarningBubblePin, OUTPUT);

  Serial.print("Connecting to WiFi");

  WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

  while (WiFi.status() != WL\_CONNECTED) {

    Serial.print(".");

    delay(100);

  }

  Serial.println(" Connected!");

  client.setServer(MQTT\_BROKER, 1883);

  client.setCallback(callback);

  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);

    } 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);

    digitalWrite(trigerPin,LOW);

    delay(2);

    digitalWrite(trigerPin,HIGH);

    delay(10);

    digitalWrite(trigerPin,LOW);

    long value = pulseIn(echoPin,HIGH);

    double meters = value/58;

    Serial.println(meters);

    if(meters > 200)

    {

      digitalWrite(lightWarningBubblePin,LOW);

      if(alart)

      {

        alart = false;

        message = "{\"Alert\":\"All Good!\"}";

        client.publish(MQTT\_TOPIC, message.c\_str());

      }

    }

    else

    {

      alart = true;

      message = "{\"Alert\":\"Someone Close!\"}";

      client.publish(MQTT\_TOPIC, message.c\_str());

    }

  }

  else if (analogRead(photoresistorPin) > 2000)

  {

    message = "{\"Alert\":\"System disabled!\"}";

    client.publish(MQTT\_TOPIC, message.c\_str());

  }

  client.loop();

  delay(1000);

}