Лабораторна робота №4

«Програмування віртуальної плати ESP32: моніторинг датчика та віддалене керування виконавчим пристроєм через MQTT»

Мета роботи

1. Закріпити навички програмування мікроконтролера ESP32.
2. Опрацювати:
   - читання аналогового/цифрового датчика;
   - керування цифровим виходом (LED / «реле»);
   - підключення до Wi-Fi;
   - обмін даними через MQTT (publish/subscribe).
3. Реалізувати віддалене керування виконавчим пристроєм з зовнішнього клієнта (на іншому комп'ютері/браузері).

Хід роботи:

1. Підготовка та реєстрація

1. Відкрити сайт-симулятор **Wokwi**:
   https://wokwi.com wokwi.com
2. Зареєструватися або увійти (Sign up / Sign in).
3. Перевірити, що браузер дозволяє спливаючі вікна й не блокує JavaScript.

2. Створення проєкту ESP32 в Wokwi

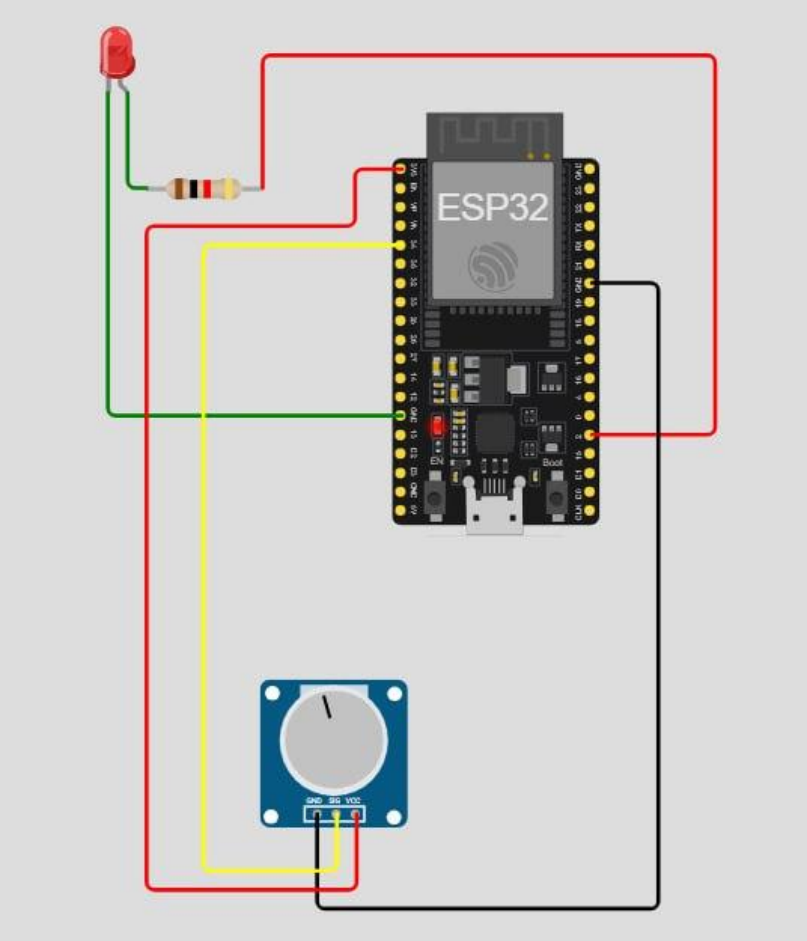
1. На головній сторінці натиснути **"Start new project"**.
2. Обрати шаблон **"ESP32"** (ESP32 DevKit v1).
3. Відкриється редактор з файлом sketch.ino і віртуальною платою ESP32.

3. Додавання компонентів на схему

У правій частині (або знизу) Wokwi є панель компонентів.

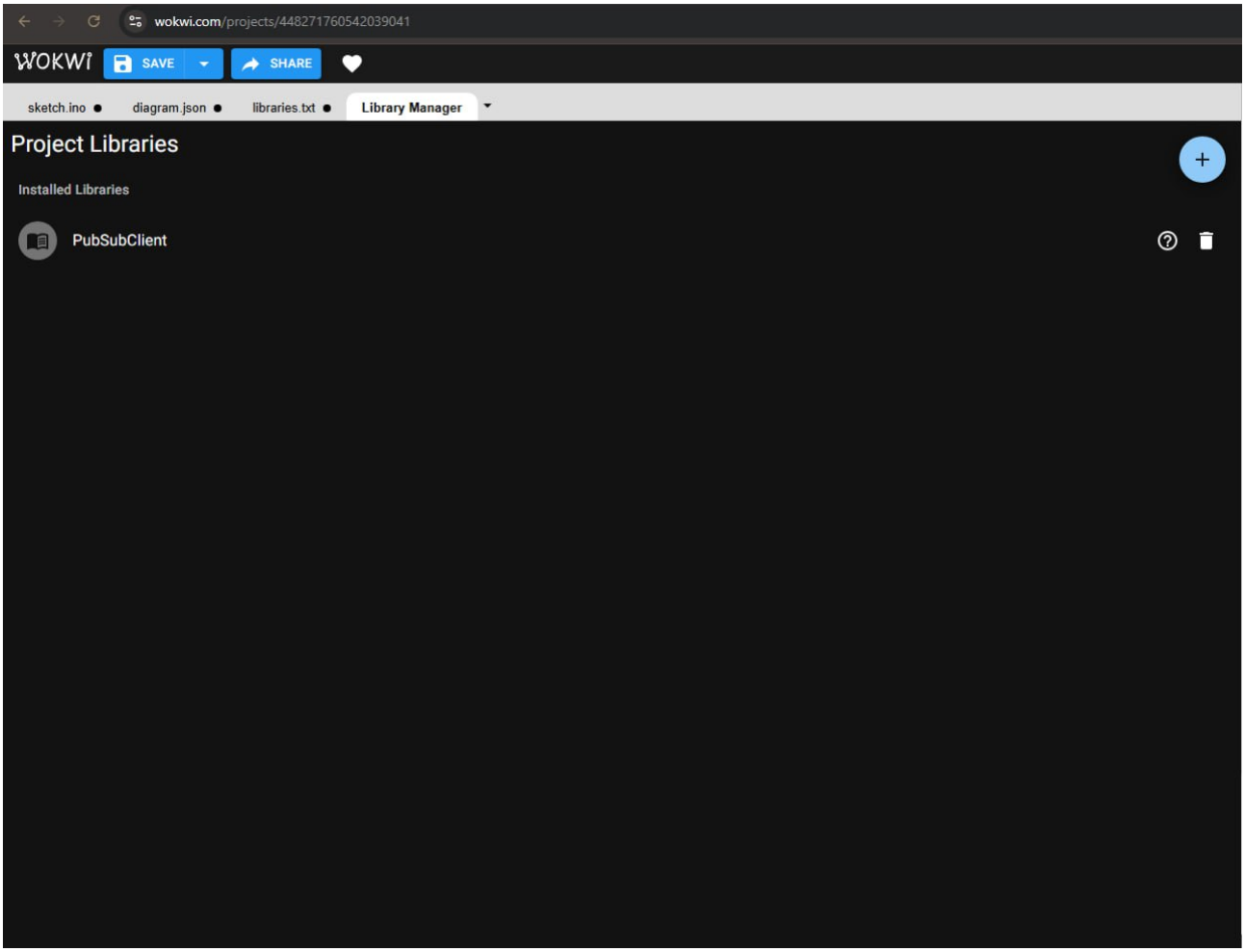
1. Додати **LED**:
   - Перетягнути компонент **LED** на поле схеми.
   - Додати **резистор** (220–330 Ом) послідовно з LED.
   - Підключити:
     - катод LED (коротка ніжка) → GND плати;
     - анод через резистор → пін **GPIO 2** (у Wokwi його позначають як D2 чи 2 — залежно від шаблону).
2. Додати **потенціометр** (імітація аналогового датчика):
   - Перетягнути компонент **Potentiometer**.
   - Крайні ніжки → **3.3V** та **GND** ESP32.
   - Середня ніжка (wiper) → **GPIO 34** (ADC-вхід ESP32).
3. Перевірити підпис пінів на схемі:
   - LED → GPIO 2
   - Потенціометр → GPIO 34

Зробити **скріншот** схеми для звіту.



4. Підключення бібліотеки MQTT (PubSubClient)

1. У Wokwi відкрити вкладку **“Library Manager”** (зазвичай зліва або зверху).
2. Знайти бібліотеку **PubSubClient**.wokwi.com
3. Натиснути **“Add to project”**.
4. Переконатися, що у списку файлів з’явився libraries.txt з записом про PubSubClient.



5. Створення базового скетчу

В sketch.ino видалити все й вставити мінімальний каркас:

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

// ---------- Налаштування Wi-Fi (Wokwi) ----------
const char* ssid     = "Wokwi-GUEST";
const char* password = "";   // порожній пароль

// ---------- Налаштування MQTT ----------
const char* mqttServer = "broker.emqx.io";
```

```
const int   mqttPort   = 1883;

// !!! ЗАМІНИТИ на свою групу/прізвище латиницею !!!
const char* topicSensor = "iot/kp-11/ivanov/sensor";
const char* topicCmd    = "iot/kp-11/ivanov/cmd";

// ---------- Піни ----------
const int ledPin    = 2;   // GPIO2 - вбудований або зовнішній LED
const int sensorPin = 34;  // потенціометр / сенсор

WiFiClient espClient;
PubSubClient client(espClient);

unsigned long lastPublish = 0;

// Прототипи
void reconnect();
void mqttCallback(char* topic, byte* payload, unsigned int length);

void setup() {
  Serial.begin(115200);
  delay(1000);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
```

```
  pinMode(sensorPin, INPUT);

  // --- Підключення до Wi-Fi ---
  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected!");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());

  // --- Налаштування MQTT-клієнта ---
  client.setServer(mqttServer, mqttPort);
  client.setCallback(mqttCallback);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  // тут далі буде публікація сенсора
}
```

6. Реалізувати підключення до MQTT

Додамо функцію **reconnect()** і callback для обробки команд.

6.1. Функція mqttCallback (обробка вхідних повідомлень)

```
void mqttCallback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");

  String msg;
  for (unsigned int i = 0; i < length; i++) {
    char c = (char)payload[i];
    msg += c;
  }
  Serial.println(msg);

  // Дуже проста "обробка JSON":
  // якщо в повідомленні є "led":"on" -> вмикаємо,
  // якщо "led":"off" -> вимикаємо
  if (msg.indexOf("\"led\"") != -1) {
    if (msg.indexOf("on") != -1) {
      digitalWrite(ledPin, HIGH);
      Serial.println("LED -> ON (by remote command)");
    } else if (msg.indexOf("off") != -1) {
      digitalWrite(ledPin, LOW);
      Serial.println("LED -> OFF (by remote command)");
    }
  }
}
```

6.2. Функція `reconnect` (відновлення з’єднання з брокером)

```
void reconnect() {
  // Цикл, поки не під’єднаємось
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection... ");

    String clientId = "esp32-";
    clientId += String(random(0xffff), HEX);

    // Спроба підключення без логіна/пароля
    if (client.connect(clientId.c_str())) {
      Serial.println("connected!");
      // Після підключення підписуємося на топік команд
      client.subscribe(topicCmd);
      Serial.print("Subscribed to: ");
      Serial.println(topicCmd);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 3 seconds");
      delay(3000);
    }
  }
}
```

7. Додавання публікації даних сенсора

Завдання: **раз на 5 секунд** читати значення потенціометра і надсилати в MQTT у форматі JSON.

Додайте в loop() після client.loop():

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long now = millis();
  if (now - lastPublish > 5000) { // кожні 5 секунд
    lastPublish = now;

    int raw = analogRead(sensorPin); // 0..4095 на ESP32
    // Перетворимо в "відсотки 0..100"
    float value = raw * 100.0 / 4095.0;

    String payload = "{";
    payload += "\"value\":";
    payload += value;
    payload += "}";

    Serial.print("Publishing to ");
    Serial.print(topicSensor);
    Serial.print(": ");
    Serial.println(payload);

    client.publish(topicSensor, payload.c_str());
  }
}
```

Тепер ESP32:

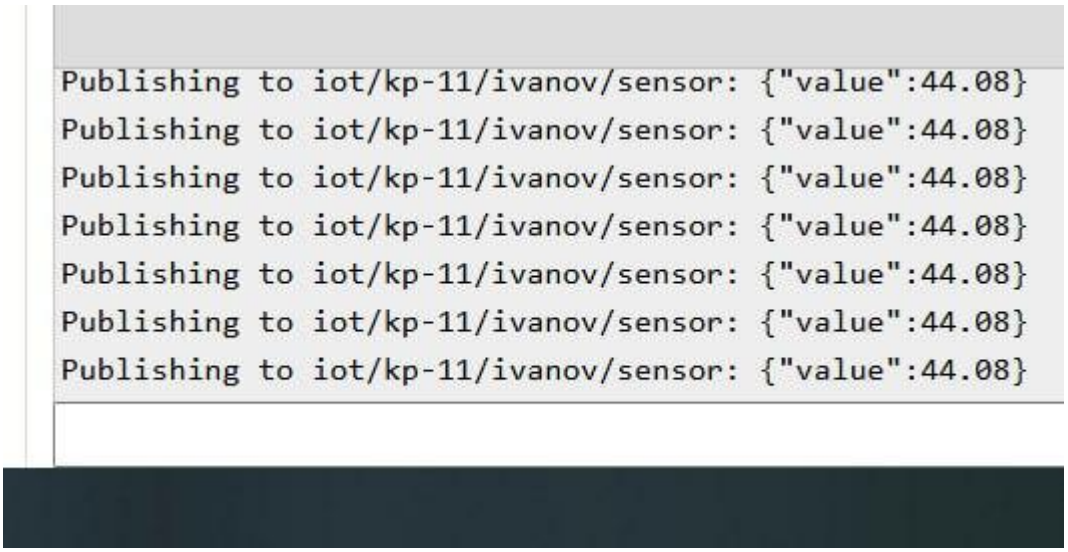
читає «сенсор» (потенціометр);

надсилає дані в MQTT кожні 5 секунд.

8. Перевірка роботи плати в симуляторі

1. Натиснути **“Play/Run”** у Wokwi (кнопка ► “Start simulation”).

2. Відкрити **Serial Monitor** (іконка монітора).
3. Переконатися, що:
   - з’являються повідомлення типу WiFi connected, Attempting MQTT connection... connected!;
   - кажні 5 секунд друкується рядок Publishing to ... { "value": ... }.



Якщо помилка підключення до MQTT — перевірити mqttServer, mqttPort і наявність інтернету.

9. Налаштування онлайн MQTT-клієнта (браузер)

Тепер покажемо **віддалене керування**: з браузера будемо надсилати команди, ESP32 — реагувати.

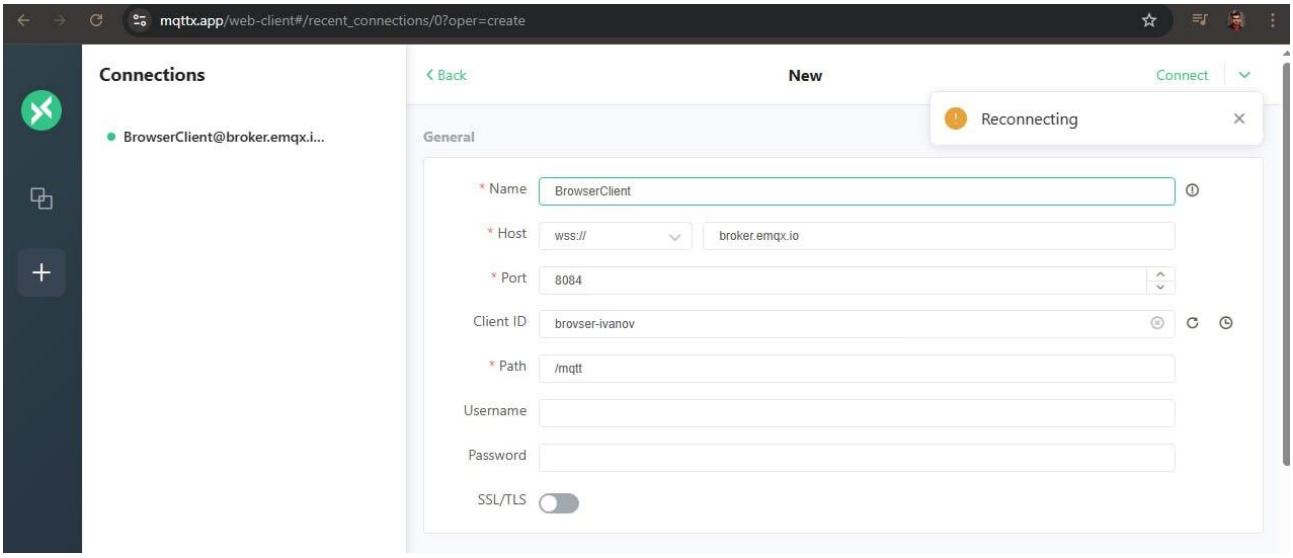
9.1. Відкрити EMQX Online MQTT Client

1. У новій вкладці браузера відкрити:
   https://www.emqx.io/online-mqtt-client
2. Натиснути **“New Connection”**.

9.2. Заповнити параметри з’єднання

Орієнтовні значення (може трохи відрізнятися в інтерфейсі, але суть та сама):

- **Name:** BrowserClient (будь-яка назва).
- **Client ID:** browser-ivanov (щось унікальне).
- **Host:** wss://broker.emqx.io:8084/mqtt
  - broker.emqx.io — адреса брокера;
  - 8084 — порт WebSocket;
  - /mqtt — шлях WebSocket для EMQX.www.emqx.com+1
- **Username / Password:** залишити порожніми.
- Інші параметри (Keep Alive, Clean Session) — за замовчуванням.



Натиснути **“Connect”**.
Статус має змінитися на **Connected**.

10. Перегляд даних сенсора з плати

1. У вкладці MQTT-клієнта натиснути **“New Subscription”** / “Subscribe”.

2. У поле **Topic** вписати свій топік сенсора, напр.:
   iot/kp-11/ivanov/sensor
3. QoS → залишити 0, натиснути **“Confirm/Subscribe”**.

Через кілька секунд ви повинні бачити повідомлення, які ESP32 публікує кожні 5 секунд:

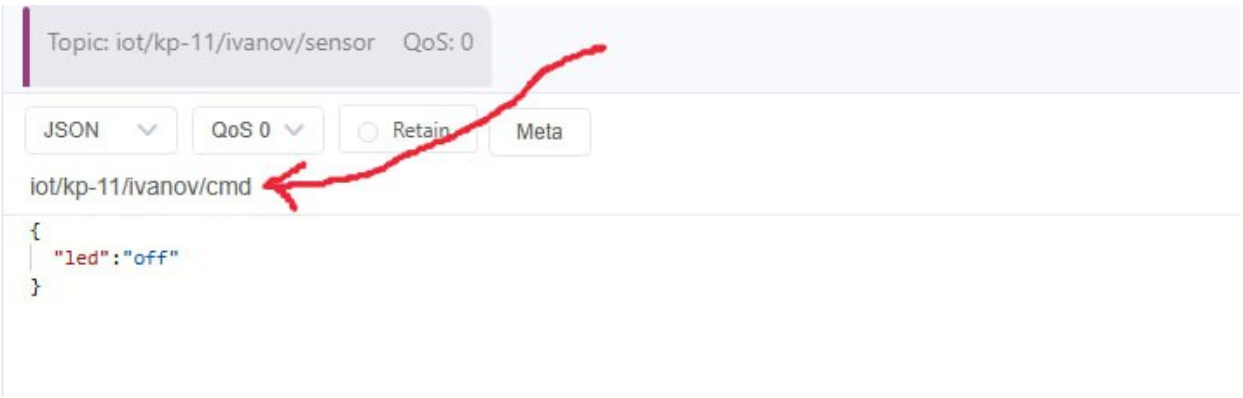
{"value": 42.3}

Повертаючись у Wokwi, покрутіть повзунок потенціометра — значення value у браузерному клієнті мають змінюватися.

11. Віддалене керування LED з браузера

Тепер перевіряємо топік **команд**.

1. У MQTT-клієнті у полі **Topic** вписати:
   iot/kp-11/ivanov/cmd



2. Тип повідомлення — JSON (якщо є опція) або просто текст.
3. У поле **Payload** ввести, наприклад:

   {"led":"on"}

Натиснути **“Publish”**.

У серійному моніторі Wokwi з’явиться:

Message arrived [iot/kp-11/ivanov/cmd] {"led":"on"}

LED -> ON (by remote command)

На платі ESP32 (віртуальній) **LED повинен загорітись**.

Аналогічно надішліть:

{"led":"off"}

LED має погаснути.

Це і є **керування на відстані**: ви зі стороннього клієнта (браузер) надсилаєте команду на MQTT-брокер, а плата ESP32, підписана на топік, її отримує й змінює стан виходу.

12. Експерименти для закріплення

1. **Змінити поріг увімкнення LED по сенсору:**
   - додати в код логіку: якщо value > 70, LED включається в автоматичному режимі.
2. **Зробити два режими роботи:**
   - mode = "auto" — плата сама керує LED по сенсору;
   - mode = "manual" — LED тільки по командам on/off.
   - режим можна змінювати через MQTT-команду:
   - {"mode":"auto"}
   - {"mode":"manual"}
3. **Додати гістерезис:**
   - увімкнення при value > 70,
   - вимкнення при value < 60,
     щоб LED не «миготів» біля порогу.