

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РАДІОЕЛЕКТРОНІКИ

ЗВІТ

з лабораторної роботи №4

з дисципліни

«Аналіз та рефакторинг коду»

Виконала

ст. гр. ПЗПІ-22-5

Черевко Марина Романівна

Перевірів:

Дашенков Д. С.

Харків 2024

4 РОЗРОБКА ІОТ КЛІЄНТА (БІЗНЕС-ЛОГІКИ ТА ФУНКЦІЙ НАЛАШТУВАННЯ)

4.1. Мета роботи

Розробити програмне забезпечення для IoT або SmartDevice пристрою, створеного на базі будь-якої поширеної на сьогодні платформи, придатної для реалізації вбудованих систем (Embedded System).

4.2. Порядок виконання роботи

4.2.1. Розробка будови програмного забезпечення IoT клієнта

Компоненти

1. ESP32

Потужний мікроконтролер із вбудованим Wi-Fi і Bluetooth. Виконує роль центрального контролера для збору даних з датчиків і управління відображенням інформації на дисплеї або подання сигналів через buzzer. Підтримка I2C, PWM, ADC, і цифрових сигналів.

2. DHT22 (AM2302)

Датчик для вимірювання температури і вологості. Збір даних про температуру і вологість повітря. Точність вимірювання температури: $\pm 0.5^{\circ}\text{C}$. Точність вимірювання вологості: $\pm 2-5\%$. Робочий діапазон температур: -40°C до $+80^{\circ}\text{C}$.

Підключення:

- Живлення (VCC): 3.3V або 5V.
- GND: Загальний мінус.
- Data: Передача даних до ESP32.

3. LCD-дисплей з I2C адаптером

ПК-дисплей із 16 символами на 2 рядки (16x2) із I2C інтерфейсом для підключення до мікроконтролера. Виконує роль відображення інформації, такої як поточна температура, вологість, чи попередження про аномальні значення.

Технічні характеристики:

- Живлення: 5V.
- I2C адреса: Зазвичай 0x27 або 0x3F.
- 4 контакти для підключення:
- GND – Мінус живлення.
- VCC – Живлення (зазвичай 5V).
- SDA – Лінія даних I2C.
- SCL – Лінія синхронізації I2C.

4. Buzzer (П'єзоелектричний динамік)

Маленький пристрій, який генерує звуковий сигнал. Видає попереджувальні звуки, наприклад, якщо температура чи вологість виходить за межі заданих параметрів.

Підключення:

- Один контакт підключається до GPIO ESP32.
- Інший контакт – до GND.

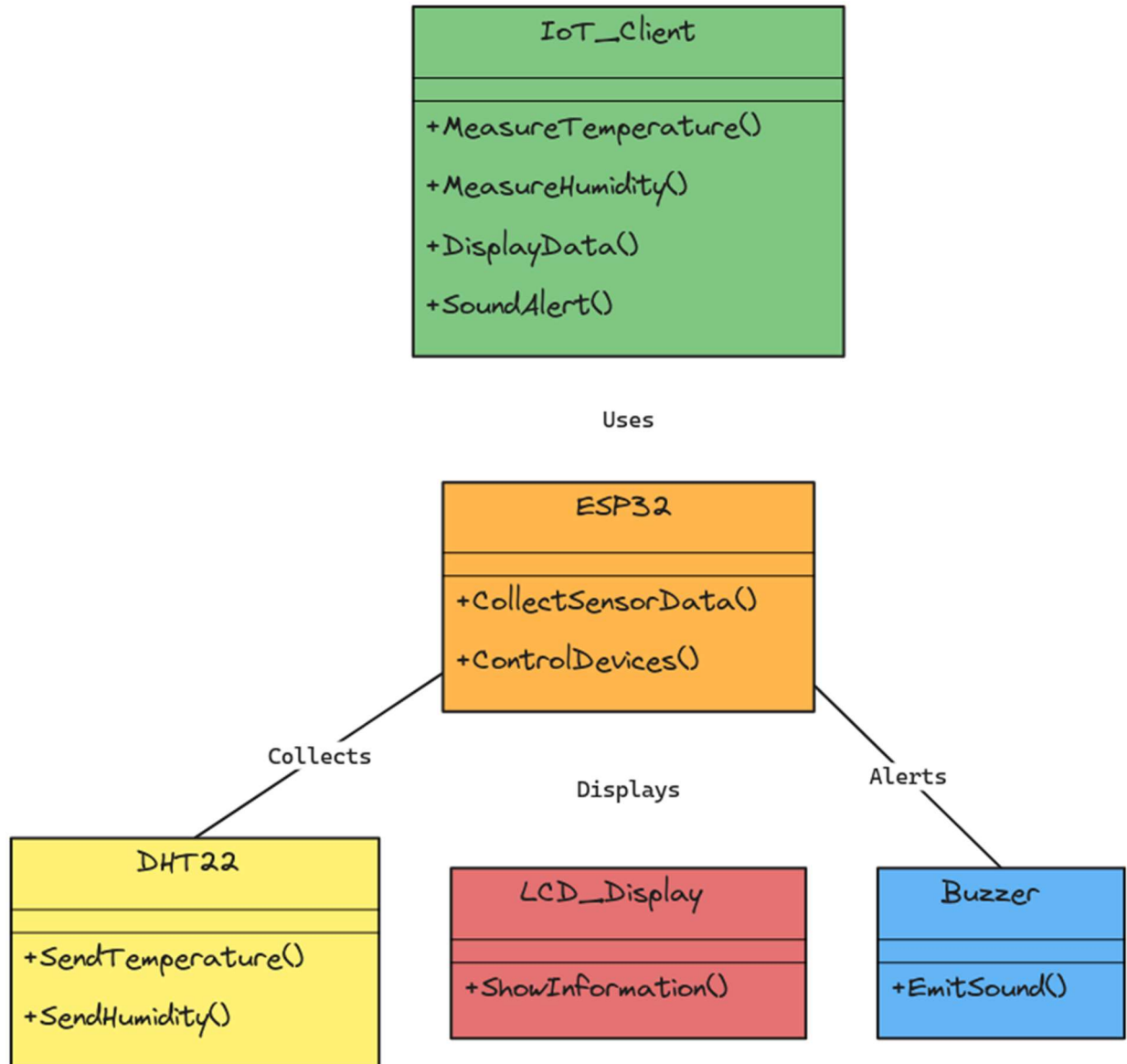
5. Проводи для з'єднання

Провідники для підключення компонентів до ESP32. Забезпечують передачу даних і живлення між компонентами.

Типи:

- Живлення: Для підключення VCC і GND.
- Сигнальні: Для передачі даних з датчиків на ESP32 і від ESP32 до дисплея чи бузера.

4.2.2. Створення UML діаграми прецедентів для IoT клієнта



4.2.3. Розробка бізнес-логіки (математична обробку пов'язаних із предметною областю даних) та функції налаштування IoT клієнта

- Фільтрація даних

Задача: обробляти отримані показники від датчиків, зменшувати вплив випадкових похибок або шумів.

Опис: забезпечує коректність і точність показників за допомогою алгоритмів математичного згладжування (наприклад, ковзне середнє).

Використовується для датчиків температури, вологості або інших вимірювальних пристроїв.

- Логування даних

Задача: зберігати отримані дані для подальшого аналізу або використання.

Опис: забезпечує запис показників у внутрішню пам'ять пристрою, на сервер або у хмару для подальшого перегляду історії даних.

- Підключення до Wi-Fi

Задача: забезпечити пристрій стабільним підключенням до бездротової мережі.

Опис: дозволяє IoT-клієнту отримувати доступ до сервера або хмарного середовища для передачі даних і прийому команд.

- Налаштування порогових значень

Задача: дозволити користувачу встановлювати допустимі діапазони показників (наприклад, мінімальну та максимальну температуру).

Опис: забезпечує персоналізацію роботи пристрою, яка відповідає умовам конкретного середовища.

- Система сповіщень

Задача: інформувати користувача про стан пристрою або порушення нормальних умов.

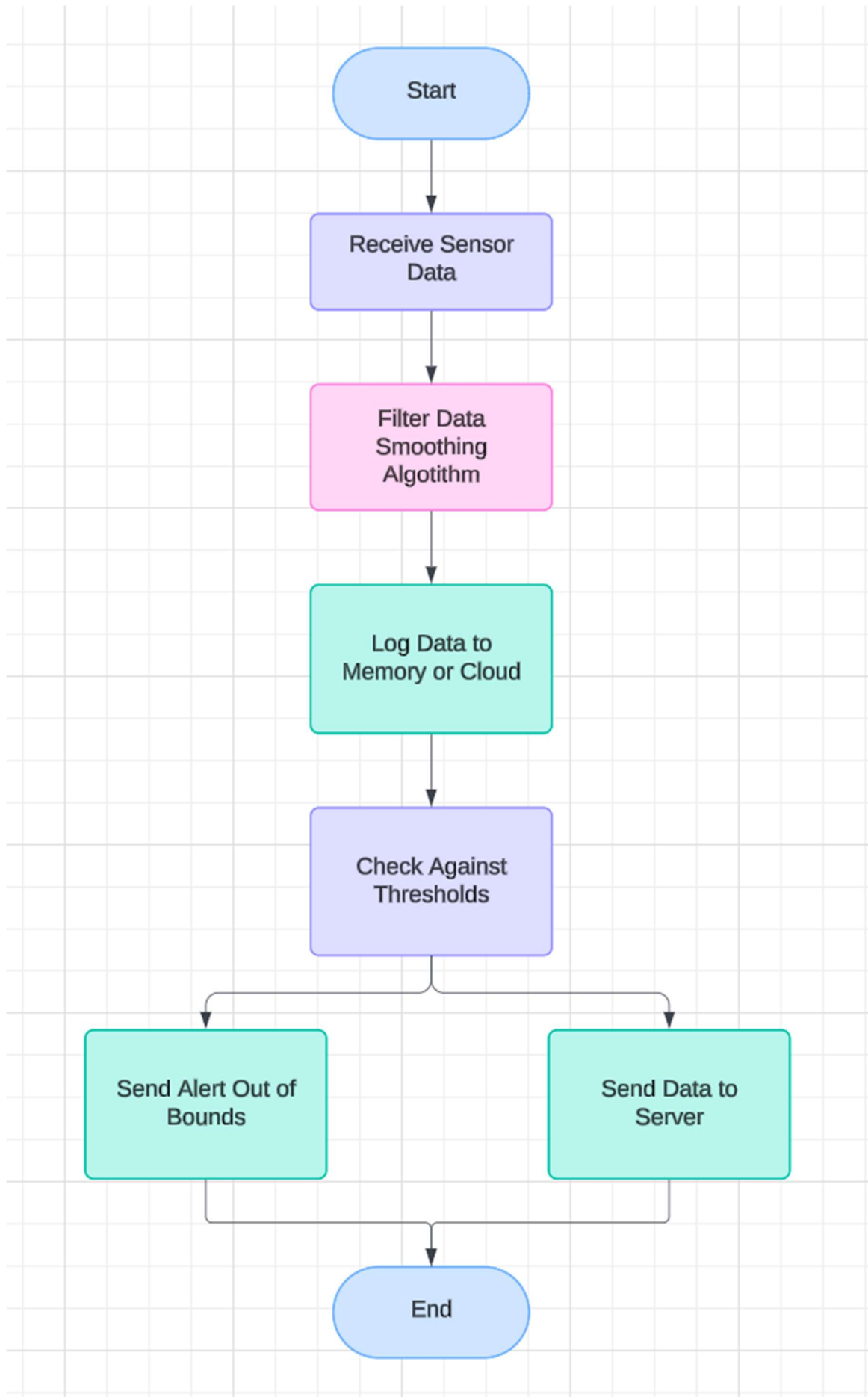
Опис: може включати повідомлення через мобільний застосунок, електронну пошту або звукові сигнали.

- Відправка даних на сервер

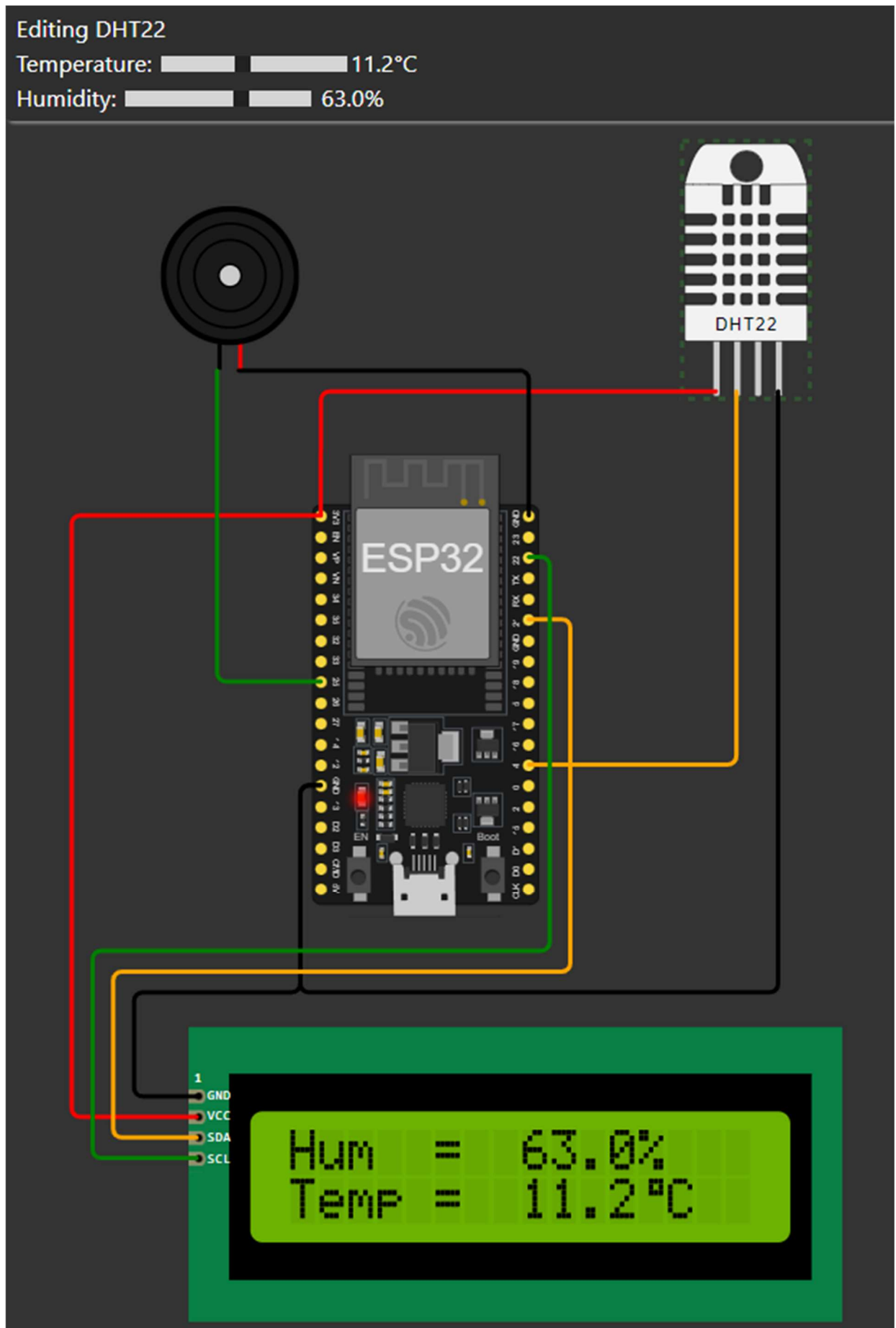
Задача: передавати зібрані дані для їх подальшого збереження або аналізу.

Опис: підтримує зв'язок між IoT-клієнтом та центральним сервером чи хмарним середовищем.

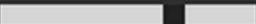
4.2.4. Створення діаграми діяльності для IoT клієнта



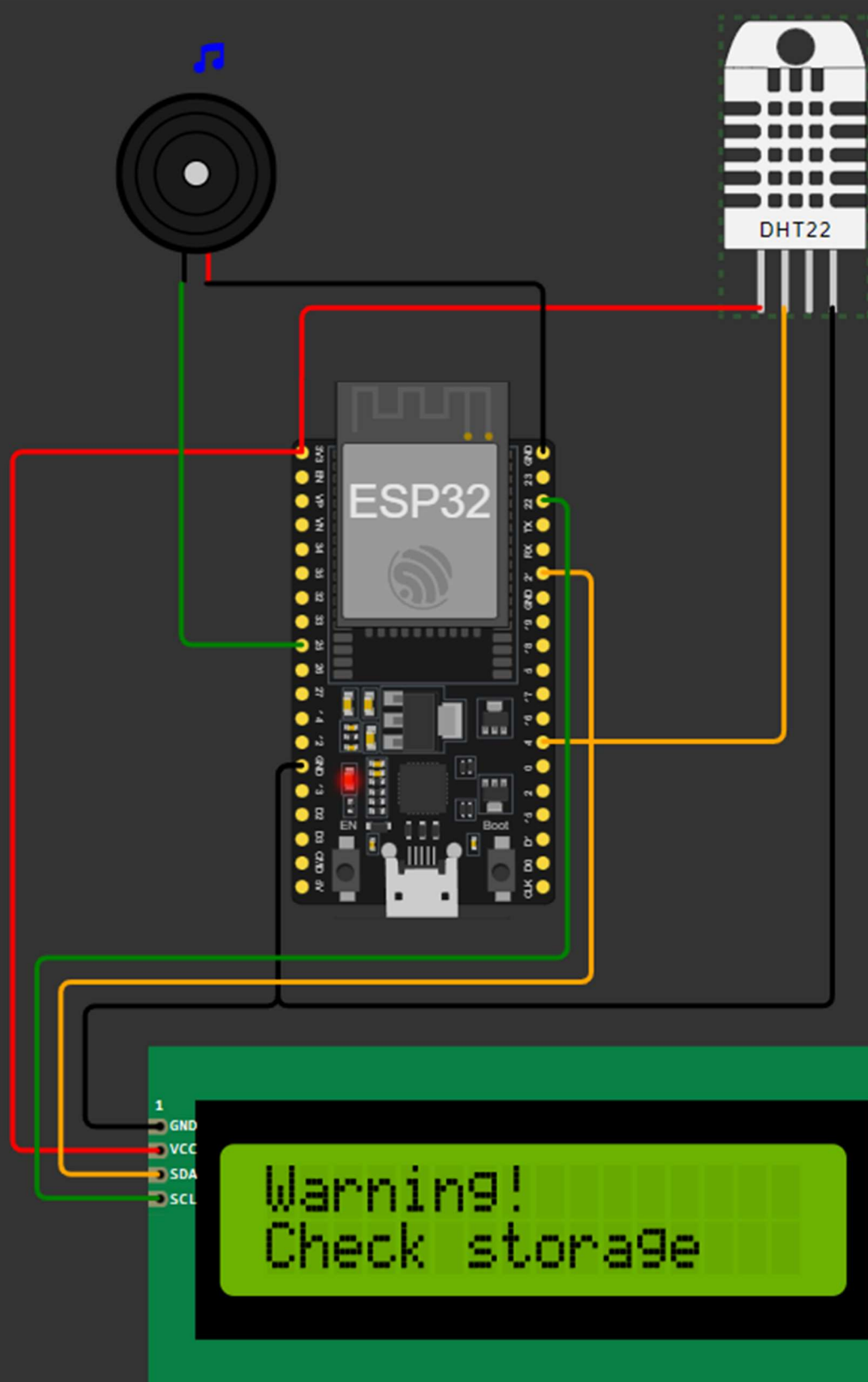
4.2.5. Перевірка роботи IoT клієнта



Editing DHT22

Temperature:  43.7°C

Humidity:  41.0%



HTTP Requests

```
-----
18:01:52.793 EET GET /api/notifications/getNotificationBySensorId/1 200 OK
18:01:41.577 EET PUT /api/sensors/updateSensorValues/1 200 OK
18:01:36.757 EET GET /api/notifications/getNotificationBySensorId/1 200 OK
18:01:33.493 EET PUT /api/sensors/updateSensorValues/1 200 OK
18:01:32.330 EET GET /api/notifications/getNotificationBySensorId/1 200 OK
18:01:29.284 EET PUT /api/sensors/updateSensorValues/1 200 OK
18:01:28.189 EET GET /api/notifications/getNotificationBySensorId/1 200 OK
18:01:24.396 EET PUT /api/sensors/updateSensorValues/1 200 OK
18:01:23.258 EET GET /api/notifications/getNotificationBySensorId/1 200 OK
18:01:19.704 EET PUT /api/sensors/updateSensorValues/1 200 OK
```

GET ▼ http://e7c9-188-163-52-155.ngrok-free.app/api/sensordata/getSensorDataBySensorId/1 Send ▼

Params Authorization Headers (8) Body ● Scripts Settings Cookies

Body Cookies Headers (6) Test Results 🔄 200 OK • 237 ms • 2.44 KB • 🌐 ⋮

Pretty Raw Preview Visualize JSON ▼ 🔗 📄 🔍

```
1  [
2    {
3      "time": "2025-01-07T00:23:56.736Z",
4      "sensor": "sensor1",
5      "temperature": 9,
6      "humidity": 51
7    },
8    {
9      "time": "2025-01-07T00:34:25.831Z",
10     "sensor": "sensor1",
11     "temperature": 8,
12     "humidity": 62
13   },
14   {
15     "time": "2025-01-07T15:51:42.121Z",
16     "sensor": "sensor1",
17     "temperature": 12,
18     "humidity": 67
19   },
20   {
21     "time": "2025-01-07T15:56:47.670Z",
22     "sensor": "sensor1",
23     "temperature": 12,
24     "humidity": 12
25   },
26   {
27     "time": "2025-01-07T15:56:56.920Z",
28     "sensor": "sensor1",
29     "temperature": 25.1,
30     "humidity": 61
31   },
32   {
33     "time": "2025-01-07T15:57:01.228Z",
34     "sensor": "sensor1",
35     "temperature": -2.5,
36     "humidity": 61
37   },
38   {
39     "time": "2025-01-07T15:57:05.587Z",
40     "sensor": "sensor1",
41     "temperature": -2.5,
42     "humidity": 61
43   },
44 ]
```

Додаток А

Код налаштування IoT-пристрою

```
#include <DHT.h>
#include <DHT_U.h>
#include "Wire.h"
#include "LiquidCrystal_I2C.h"
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* serverUrl = "http://e7c9-188-163-52-155.ngrok-free.app";

#define DHTPIN 4
#define DHTTYPE DHT22
#define BUZZER_PIN 25

DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);

byte degree[8] = {
    B00111,
    B00101,
    B00111,
    B00000,
    B00000,
    B00000,
    B00000,
    B00000
};

struct NotificationSettings {
    float temperature_min;
    float temperature_max;
    float humidity_min;
    float humidity_max;
};

void connectToWiFi() {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Connecting");
    lcd.setCursor(0, 1);
    lcd.print("to WiFi...");
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        lcd.print(".");
    }
}
```

```

}

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("WiFi Connected");
lcd.setCursor(0, 1);
lcd.print(WiFi.localIP());
delay(2000);
}

NotificationSettings fetchNotificationSettings() {
    NotificationSettings settings = { NAN, NAN, NAN, NAN };

    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        String url = String(serverUrl) +
"/api/notifications/getNotificationBySensorId/1";
        http.begin(url);
        int httpResponseCode = http.GET();

        if (httpResponseCode > 0) {
            String response = http.getString();
            Serial.println("Response from server: " + response);

            int tempMinIndex = response.indexOf("\"temperature_min\":");
            int tempMaxIndex = response.indexOf("\"temperature_max\":");
            int humMinIndex = response.indexOf("\"humidity_min\":");
            int humMaxIndex = response.indexOf("\"humidity_max\":");

            if (tempMinIndex >= 0) {
                String tempMinStr = response.substring(tempMinIndex + 18,
response.indexOf(",", tempMinIndex));
                settings.temperature_min = tempMinStr.equals("null") ? NAN :
tempMinStr.toFloat();
            }

            if (tempMaxIndex >= 0) {
                String tempMaxStr = response.substring(tempMaxIndex + 18,
response.indexOf(",", tempMaxIndex));
                settings.temperature_max = tempMaxStr.equals("null") ? NAN :
tempMaxStr.toFloat();
            }

            if (humMinIndex >= 0) {
                String humMinStr = response.substring(humMinIndex + 15,
response.indexOf(",", humMinIndex));
                settings.humidity_min = humMinStr.equals("null") ? NAN :
humMinStr.toFloat();
            }

            if (humMaxIndex >= 0) {

```

```

        String humMaxStr = response.substring(humMaxIndex + 15,
response.indexOf("}", humMaxIndex));
        settings.humidity_max = humMaxStr.equals("null") ? NAN :
humMaxStr.toFloat();
    }
} else {
    Serial.println("Error fetching notification settings: " +
String(httpResponseCode));
}

    http.end();
} else {
    Serial.println("WiFi disconnected. Cannot fetch notification settings.");
}

    return settings;
}

void playWarningTone() {
    for (int i = 0; i < 3; i++) {
        tone(BUZZER_PIN, 1000, 500);
        delay(500);
        noTone(BUZZER_PIN);
        delay(500);
    }
}

void sendDataToServer(float temperature, float humidity) {
    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        String url = String(serverUrl) + "/api/sensors/updateSensorValues/1";
        http.begin(url);
        http.addHeader("Content-Type", "application/json");

        String jsonPayload = "{";
        jsonPayload += "\"temperature\":" + String(temperature, 1) + ",";
        jsonPayload += "\"humidity\":" + String(humidity, 1);
        jsonPayload += "}";

        int httpResponseCode = http.PUT(jsonPayload);

        if (httpResponseCode > 0) {
            String response = http.getString();
            Serial.println("Server Response: " + response);
        } else {
            Serial.println("Error sending data to server. Code: " +
String(httpResponseCode));
        }

        http.end();
    } else {

```

```

        Serial.println("WiFi Disconnected. Cannot send data.");
    }
}

void setup() {
    pinMode(BUZZER_PIN, OUTPUT);
    lcd.init();
    lcd.backlight();
    lcd.createChar(1, degree);
    Serial.begin(9600);

    connectToWiFi();
    dht.begin();
    Serial.println("DHT22 initialized.");
}

void loop() {
    delay(2000);
    float t = dht.readTemperature();
    float h = dht.readHumidity();

    lcd.clear();

    if (isnan(t) || isnan(h)) {
        lcd.setCursor(0, 0);
        lcd.print("Sensor Error");
        Serial.println("Error reading DHT22 data");
        delay(3000);
        return;
    }

    NotificationSettings settings = fetchNotificationSettings();

    lcd.setCursor(0, 0);
    lcd.print("Hum = ");
    lcd.setCursor(8, 0);
    lcd.print(h, 1);
    lcd.print("%");
    lcd.setCursor(0, 1);
    lcd.print("Temp =");
    lcd.setCursor(8, 1);
    lcd.print(t, 1);
    lcd.print("\1C");

    bool warning = false;

    if (!isnan(settings.temperature_min) && t < settings.temperature_min) warning =
true;
    if (!isnan(settings.temperature_max) && t > settings.temperature_max) warning =
true;

```

```
if (!isnan(settings.humidity_min) && h < settings.humidity_min) warning = true;
if (!isnan(settings.humidity_max) && h > settings.humidity_max) warning = true;

if (warning) {
  Serial.println("Warning: Check storage conditions!");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Warning!");
  lcd.setCursor(0, 1);
  lcd.print("Check storage");
  playWarningTone();
  sendDataToServer(t, h);
  delay(5000);
} else {
  Serial.println("Conditions normal.");
  sendDataToServer(t, h);
}
}
```