Міністерство науки та освіти України
Харківський національний університет радіоелектроніки

Звіт до лабораторної роботи №5
з дисципліни «Аналіз та рефакторінг коду програмного забезпечення»
На тему: «Програмна система для виявлення задимлення у приміщенні та допомоги при евакуації»

Виконав:
ст. гр. ПЗПІ-19-3
Логвінов О. В.

Перевірив:
ст. викл. каф. ПІ
Сокорчук І.П

Харків 2021

Тема роботи: Розробка розумного пристрою програмної системи для виявлення задимлення у приміщенні та допомоги при евакуації.

Мета роботи: Розробити розумний пристрій на платформі esp8266, що буде вимірювати показники вмісту повітря та у разі перевищення цих показників відправляти повідомлення на сервер за допомогою API. Також організувати оповіщення людей про небезпеку, а також їх пропуск у приміщення шляхом отримання відповідних команд з серверу за допомогою MQTT протоколу.

Посилання на файл з кодом на гугл-диску:

https://drive.google.com/file/d/1XdrTFW4AGF1NLyzI0xFHH2np0NQ2zDwi/view?usp=sharing

MD5 хеш для pzpi-19-3-lohvinov-oleksandr-lab5.zip:

1C2393BAE25A57AE78C8FEACD4F59D99

Хід роботи:

1) Створення пристрою на основі плати esp8266 та датчику диму MQ-135.
2) Написання коду, що забезпечує під'єднання пристрою до локальної мережі wifi, встановлення з'єднання з сервером використовуючи технологію MQTT.
3) Побудова UML діаграм: UML діаграму прецедентів (Use case diagram), UML діаграму компонентів (Component Diagram), UML діаграму взаємодії (Interaction Overview Diagram), UML діяльності(Activity Diagram);

Для відображення того, як пристрій може взаємодіяти з системою була розроблена UseCase діаграма (див. рисунок 1). Згідно з цією діаграмою, видно, що актором є сам пристрій. До його обов'язків відноситься зчитування даних з датчику диму MQ-135, відправки отриманих даних на сервер у разі їх підвищення та реагування на команди, що надходять з серверу, а саме:

- Пропустити людину у приміщення у разі успішного проходження перевірочного тесту.

- Відкрити двері під час надзвичайної ситуації у будівлі.
- Повернутися у початковий стан при закінченні надзвичайної ситуації.

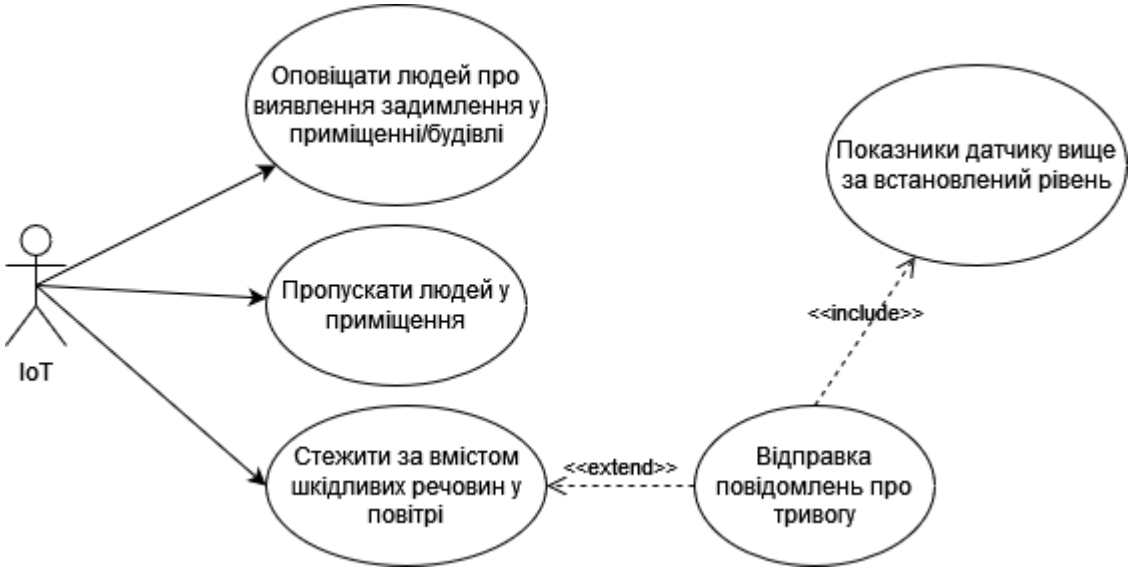

Рисунок 1 – Діаграма прецедентів розумного пристрою програмної системи для виявлення задимлення у приміщенні та допомоги при евакуації

Для зображення того, з яких елементів складається система, була розроблена діаграма компонентів (див. рисунок 2).

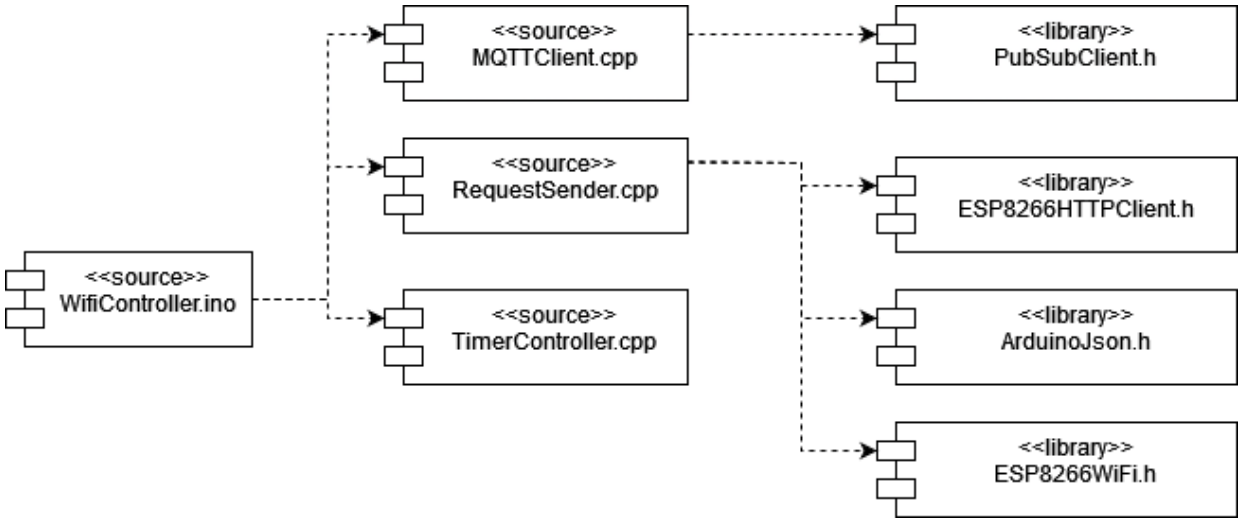

Рисунок 2 – Діаграма компонентів розумного пристрою програмної системи для виявлення задимлення у приміщенні та допомоги при евакуації

Зі схеми видно, що головним файлом у програмі є WifiController.ino. Під час роботи коду даного файлу ініціалізується підключення до серверу, робиться автентифікація пристрою, налаштовується з'єднання по протоколу MQTT та виконується основний цикл програми. Автентифікація та відправка показників на сервер виконує код класу RequestSender.cpp. Так як для відправки даних використовується JSON, то для перетворення даних у потрібний формат використовується бібліотека ArduinoJson. Код функцій автентифікації та відправки даних наведений у додатку В. За зв'язок пристрою з сервером по протоколу MQTT та обробку запитів від сервера відповідає клас MQTTClient.cpp. MQTT (Message Queue Telemetry Transport) - спрощений мережевий протокол, що працює на TCP/IP. Використовується для обміну повідомленнями між пристроями за принципом видавець-підписник. У своїй роботі використовую функціонал бібліотеки PubSubClient. Код даного класу наведений у додатку Г.

Для опису взаємодії пристрою з сервером була розроблена діаграма взаємодії (див. додаток А), що описую основний цикл роботи пристрою у разі отримання високих показників із датчику диму MQ-135.

Для зменшення навантаження на сервер пристрій відправляє данні з періодичністю 10 секунд. Спочатку відправляється POST запит до серверу з даними датчику а також описом пристрою, що їх надсилає. На сервері дані перевіряються ще раз та у разі підтвердження загрози відправляються відповідні повідомлення про загрозу клієнтській частині, мобільному застосунку, використовуючи технологію SignalR, та іншим розумним пристроям у приміщенні через MQTT протокол. Після даної операції розумний пристрій перевіряє показники датчику ще раз, та якщо загроза не зникла, повторює описаний цикл ще раз.

Для опису життєвого циклу пристрою була розроблена діаграма діяльності (див. додаток Б). Згідно з цією діаграмою, спочатку пристрій під'єднується до мережі, після чого проводиться автентифікація. Якщо пристрій не

зареєстрований, то він не може взаємодіяти із сервером. Далі виконується основна робота пристрою, а саме: зчитування даних з датчику диму та обробка запитів, що надходять із серверу.

Висновки: під час лабораторної роботи був розроблений пристрій, що стежить за показниками повітря у приміщенні та сповіщаю сервер та персонал будівлі у разі виявлення надзвичайної ситуації. Був налаштований зв'язок із сервером за допомогою протоколу MQTT та выдправки даних, використовуючи бібліотеки платформи esp8266 ESP8266HTTPClient та ESP8266WiFi.

Додаток А

Діаграма взаємодії – зображує процес взаємодії пристрою із сервером під час виявлення надзвичайної ситуації

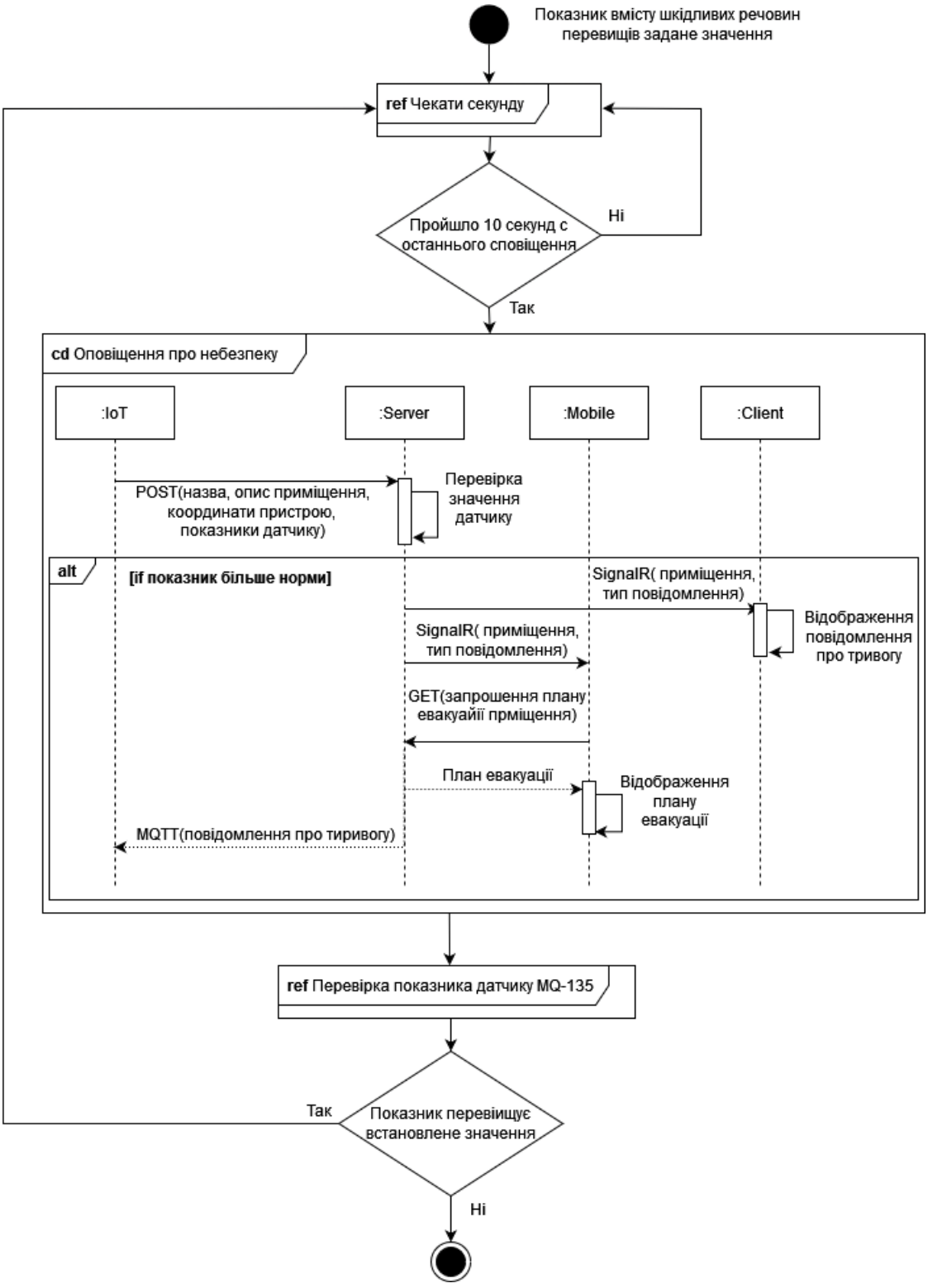

Рисунок А.1 – Діаграма взаємодії пристрою із сервером

Додаток Б

Діаграма діяльності – зображує алгоритм роботи пристрою програмної системи для виявлення задимлення у приміщенні та допомоги при евакуації

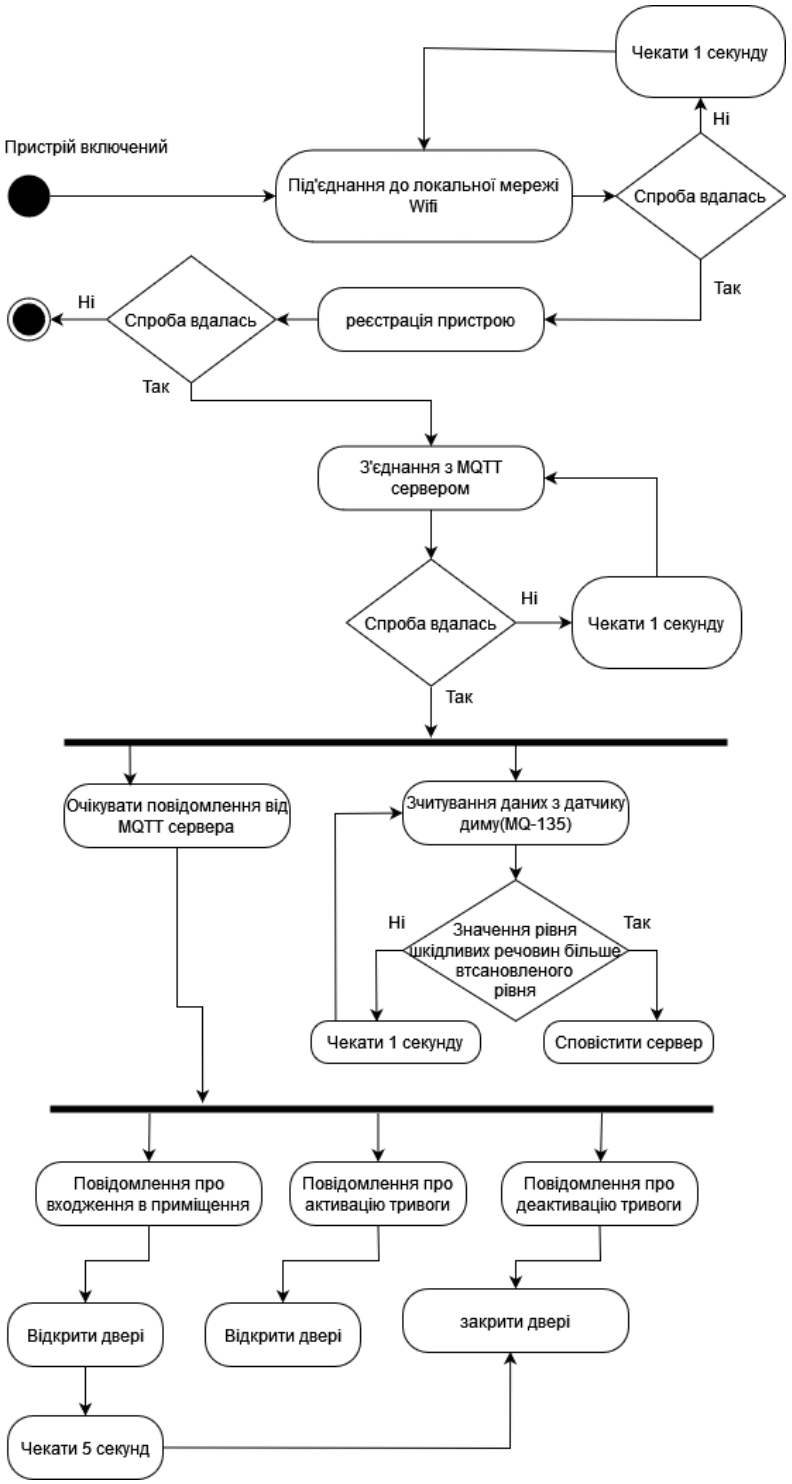

Рисунок Б.1 – Життєвий цикл пристрою програмної системи

Додаток В

Функція авторизації пристрою

```
bool RequestSender::RegisterIoT(int &iotId)
{
    HTTPClient http;
    StaticJsonDocument<300> loginDataJs;
    StaticJsonDocument<300> responseJs;
    loginDataJs["Identifier"] = iotIdentifier;
    char loginData[100];
    serializeJson(loginDataJs, loginData);

    String dataToSend = String(loginData);
    Serial.println("Sending data: " + dataToSend);
    http.begin(client, "http://192.168.0.109:5000/IoT/loginIot");
    http.addHeader("Content-Type", "application/json");

    int httpCode = http.POST(dataToSend);
    String payload = http.getString();
    Serial.println(payload);

    if (payload == "" || httpCode <= 0)
    {

        Serial.println(httpCode);
        Serial.println("Response payload: " + payload);
        return false;
    }

    deserializeJson(responseJs, payload);

    iotId = responseJs["userId"].as<int>();
    this->bearerToken = responseJs["token"].as<String>();

    Serial.println("iotId: " + String(iotId));
    Serial.println(responseJs["token"].as<String>());
    return true;
}
```

Додаток Г

Код, що відповідає за встановлення MQTT з'єднання з сервером

Функція обробки запиту, шо приходить з серверу

```
1  void callback(char *topic, uint8_t *payload, unsigned int length)
2  {
3    Serial.print("Message arrived in topic: ");
4    Serial.println(topic);
5    Serial.print("Message:");
6
7    string message = std::string((char *)payload);
8    if (message == "open" && !isSetAlarm)
9    {
10     openCloseDoorTimer.SetTimer(5, CloseDoor, OpenDoor);
11   }
12   else if (message == "alarm" && !isSetAlarm)
13   {
14     Serial.println("Activating alarm....");
15     if (openCloseDoorTimer.IsTimerOn())
16     {
17       openCloseDoorTimer.FinishTimer();
18       delay(1000);
19     }
20     isSetAlarm = true;
21     Serial.println("Alarm is on");
22     alarmTimer.SetTimer(
23     std::numeric_limits<int>::max() / 2, OffAlarm, SetAlarm);
24   }
25   else if (message == "close" && isSetAlarm)
26   {
27     isSetAlarm = false;
28     alarmTimer.FinishTimer();
29     Serial.println("Alarm is off");
30   }
31   Serial.println(message.c_str());
32   Serial.println("----------------------");
33   }
34
35 static void callback_function(char *topic, uint8_t *payload, unsigned
36                                                              int length)
37 {
38   Serial.println("Call back is called");
39   MQTTClient::current->callback(topic, payload, length);
40 }
```

Функція підключення пристрою до серверу використовуючи MQTT протокол

```
void initMqttClient()
{
  mqttClient.setServer(mqtt_broker, mqtt_port);
  MQTTClient::current = this;
  mqttClient.setCallback(callback_function);

  while (!mqttClient.connected())
  {
    String client_id = "esp8266-client-";
    client_id += String(WiFi.macAddress());
    Serial.printf("The client %s connects to the public mqtt broker\n",
                                                        client_id.c_str());
    if (mqttClient.connect(client_id.c_str(), mqtt_username,
                                                         mqtt_password))
    {
      Serial.println("Public emqx mqtt broker connected");
    }
    else
    {
      Serial.print("failed with state ");
      Serial.print(mqttClient.state());
      delay(2000);
    }
  }
}
```