Міністерство освіти та науки України
Харківський національний університет радіоелектроніки
Кафедра програмної інженерії

Лабораторна робота № 5

З дисципліни: «Архітектура програмного забезпечення»

на тему: «Програмна система для автоматизації процесів прибирання приміщень»

Виконав
ст. гр. ПЗПІ-18-2
Кузнецов Роман Олександрович

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

Харків 2021

Мета: розробити IoT/Smart Device частину для програмної системи для автоматизації процесів прибирання приміщень.

Хід роботи:

В якості платформи, придатної для реалізації вбудованих систем (Embedded System), було обрано Arduino, а саме Arduino Uno R3. Програмне забезпечення для IoT написане на мові програмування для Arduino, синтаксис якої є полегшеною версією синтаксису мови C++. Для зв'язку з мережею Інтернет використано модуль Ethernet Shield 1. Використано Інтернет кабель для підключення цього модулю до мережі Інтернет, USB кабель для підключення плати Arduino до ПК для живлення та завантаження програмного коду. Використано датчик температури та вологості DHT-11 і датчик якості повітря та наявності у повітрі шкідливих речовин MQ-135.

Перед тим, як програмно реалізувати IoT частину, було створено Use Case діаграму, що описує сценарій поведінки застосунку у процесі взаємодії з його користувачами. Use Case діаграма для системи взагалі наведена у додатку А. Use Case діаграма для IoT частини системи зображена на рисунку 1.

Рисунок 1 – Use Case діаграма для IoT частини програмної системи «PurityService»

Для взаємодії з серверною частиною системи використовується HTTPS протокол та JSON формат транспортування даних, що у HTTP термінології позначається як application/json. Для реалізації цієї взаємодії використовуються бібліотеки Ethernet.h та ArduinoJson.h. Для роботи з датчиком вологості та температури використовувалось API бібліотеки DHT.h, для роботи з датчиком якості повітря використовувалось API бібліотеки MQ135.h.

Для відображення робочих компонентів IoT/Smart Device частини системи та відображення логіки їх взаємодії та інженерних рішень під час проектування було створено діаграму компонентів IoT/Smart Device частини програмної системи. Діаграма компонентів для програмної системи зображена на рисунку 2.

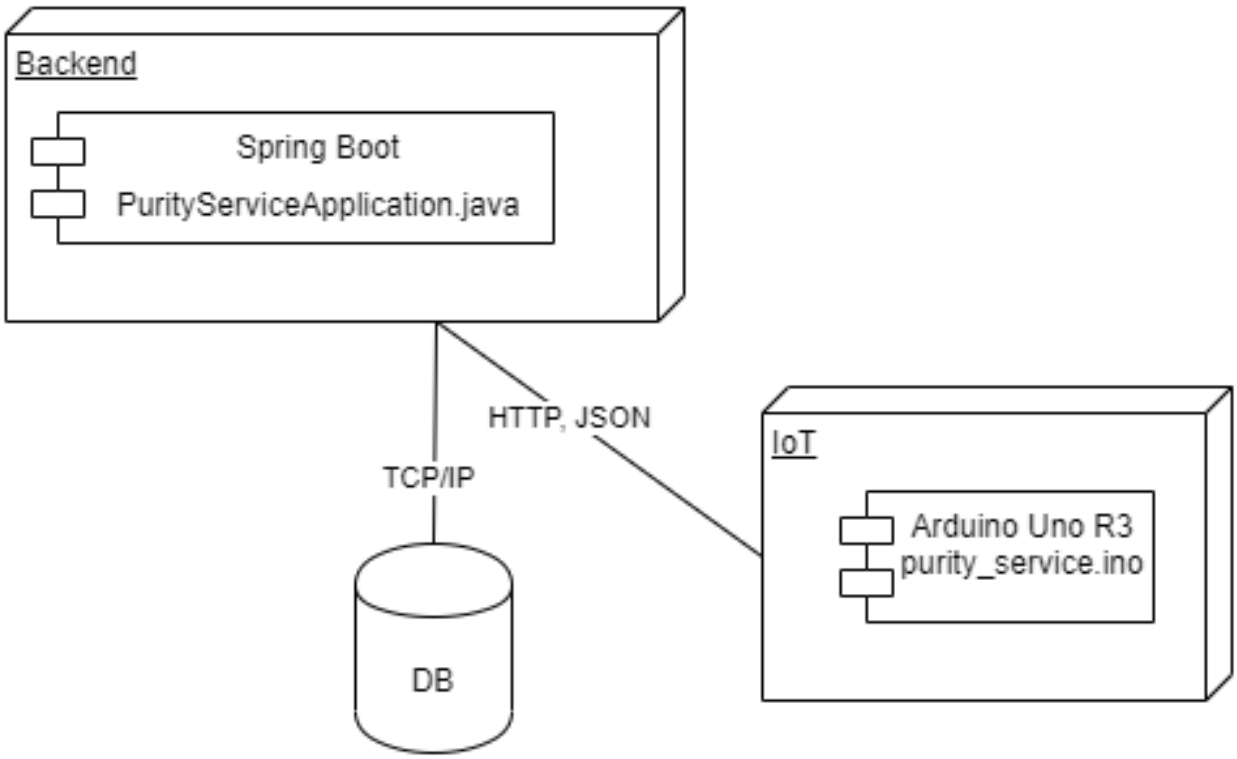

Рисунок 2 – Діаграма компонентів IoT частини програмної системи «Purity Service»

Для більш детального опису умов переходів системи з одного стану в інший також побудовано діаграму діяльності, що наведена на рисунку 3.

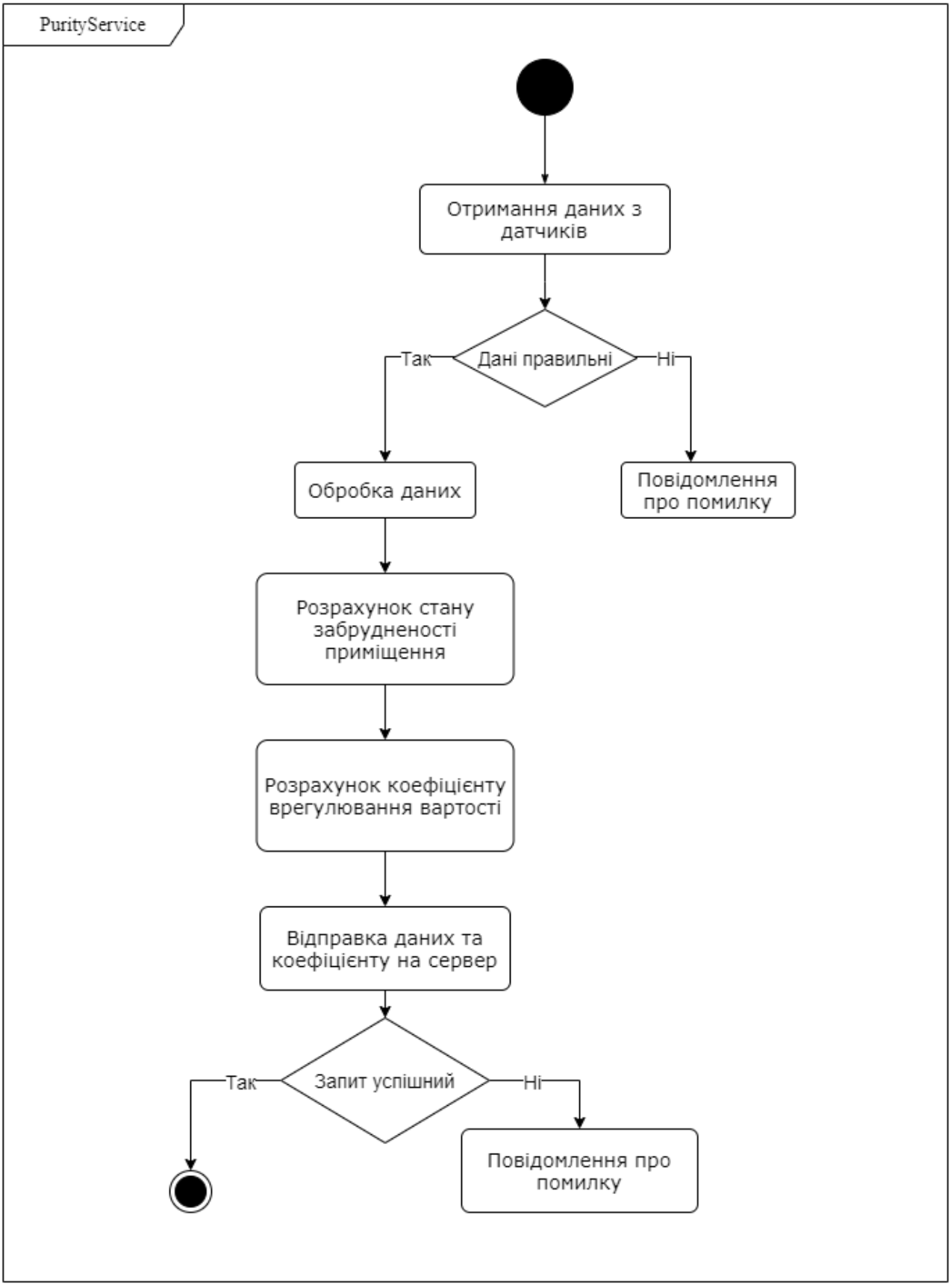

Рисунок 3 – Діаграма діяльності для IoT частини програмної системи «Purity Service»

Для опису поведінки тільки в межах одного варіанта використання було створено діаграму взаємодії, зображену на рисунку 4. На діаграмі відображено екземпляри об'єктів та повідомлення, якими ці об'єкти обмінюються один з одним в рамках даного варіанта використання.

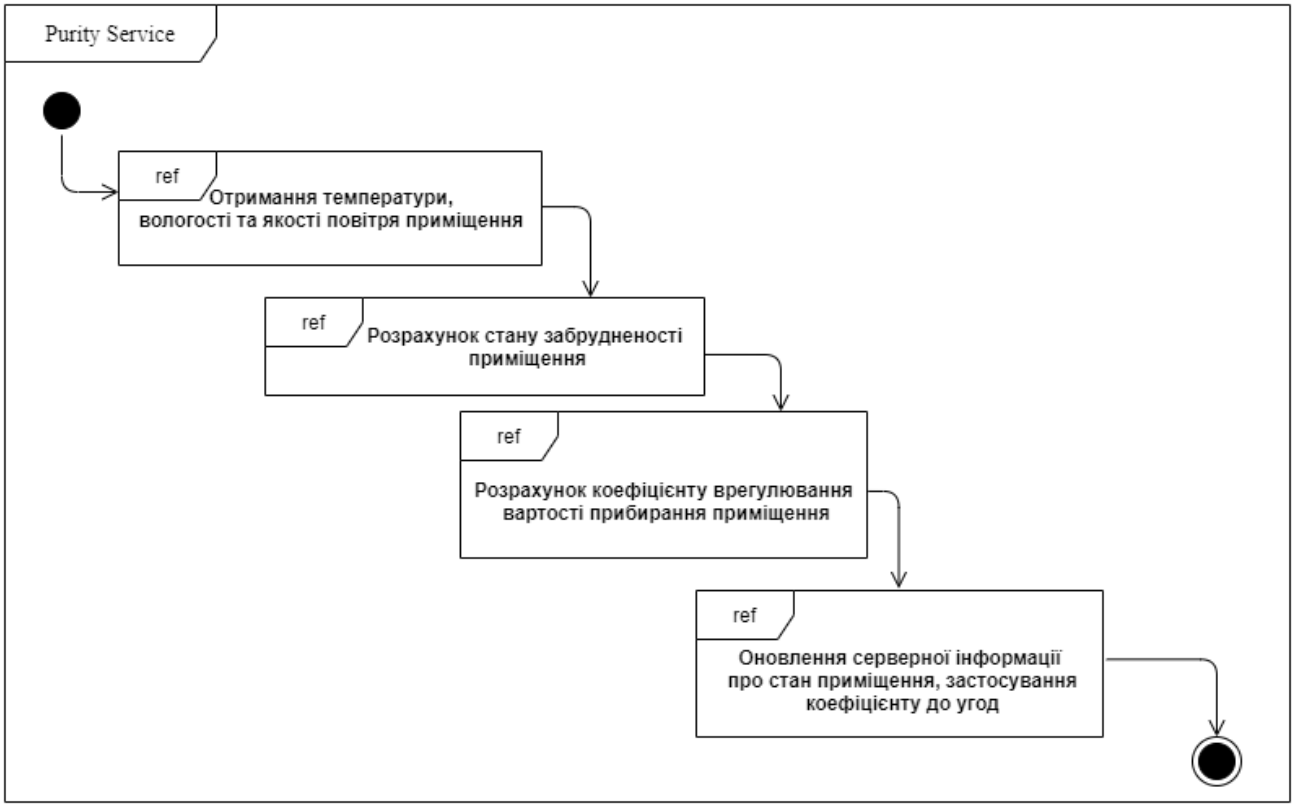

Рисунок 4 – Діаграма взаємодії для IoT частини системи «Purity Service»

У додатку Б наведено повну програмну реалізацію логіки IoT пристрою.

У додатку В наведено частину програмної реалізації серверної взаємодії з IoT пристроями.

Посилання на архів з програмним кодом та файл контрольної суми:

https://drive.google.com/drive/folders/1R73YAKXwjxAgCr7tVDh-xhRIDV9QMsmh?usp=sharing

Контрольна сума до архіву: 9ca59d926cad2751e0b4d74659222865

Висновок: під час виконання лабораторної роботи було розроблено IoT частину для програмної системи для автоматизації процесів прибирання приміщень.

ДОДАТОК А

Use Case діаграма програмної системи

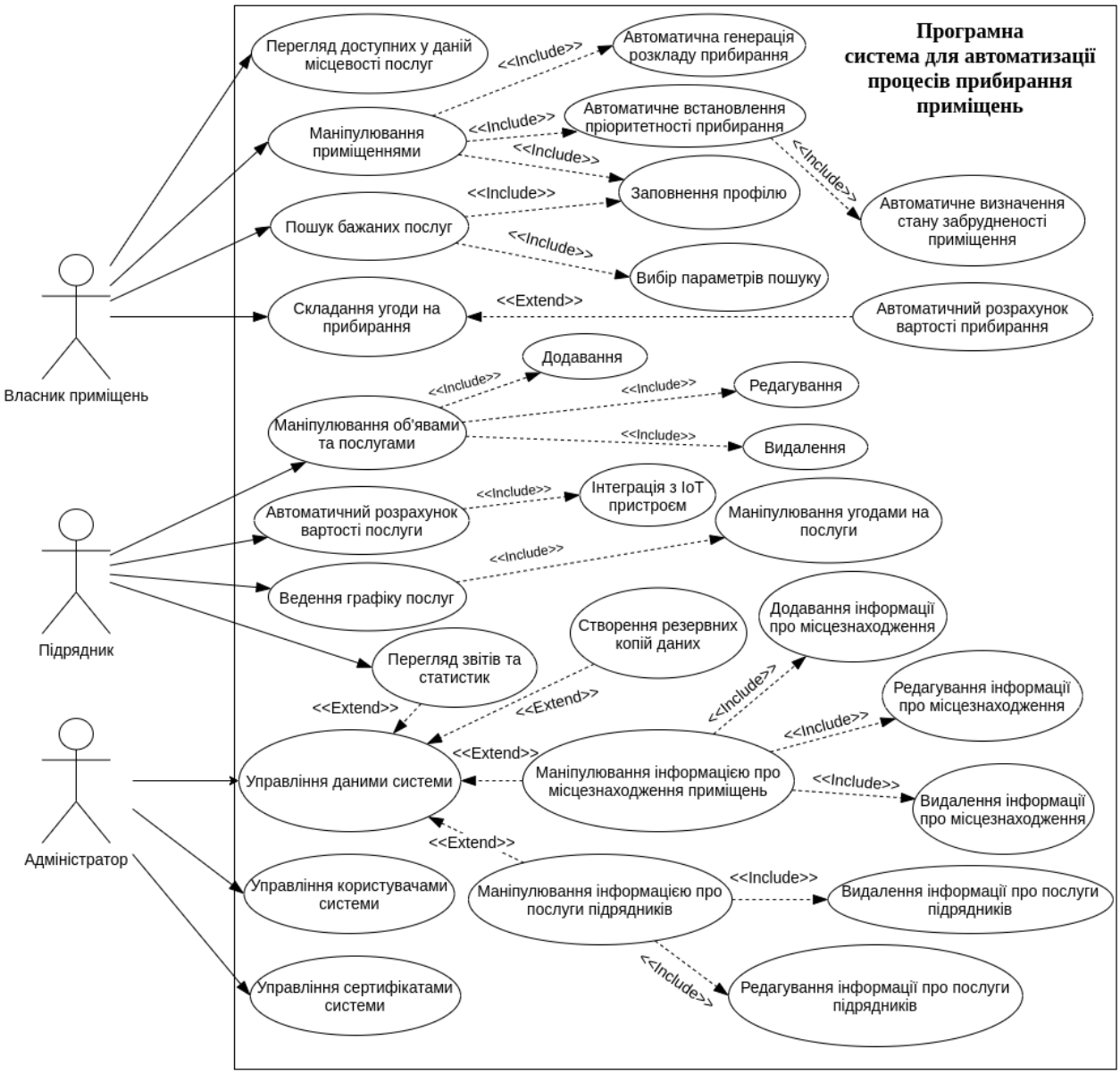

Рисунок А.1 - Діаграма варіантів використання для програмної системи автоматизації процесів прибирання приміщень

ДОДАТОК Б

Програмна реалізація логіки IoT пристрою

```
#include <SPI.h>
#include <Ethernet.h>
#include <ArduinoJson.h>
#include "DHT.h"
#include "MQ135.h"

#define MQ_PIN  A4
#define DHT_PIN A5

byte mac[] = {0x90, 0xA2, 0xDA, 0x00, 0x82, 0x7A};

IPAddress server(192, 168, 0, 103);

int    HTTP_PORT = 8080;
String HTTP_METHOD = "POST";
String HOST_NAME = "localhost";
String PATH_NAME = "/device";
int    id = 5;

IPAddress ip(192, 168, 0, 104);
IPAddress myDns(192, 168, 0, 1);

EthernetClient client;

DHT dht(DHT_PIN, DHT11);
MQ135 mq = MQ135(MQ_PIN);

float recHumidity = 50;
float recTemperature = 23;
float fullValue = 1;

float humidityDiff;
float temperatureDiff;
float totalDiff;

void setup() {
    Serial.begin(9600);

    while (!Serial) {}

    dht.begin();

    Serial.println("Ініціалізація з'єднання за допомогою DHCP:");

    if (Ethernet.begin(mac) == 0) {
        Serial.println("Не вдалося налаштувати Інтернет з DHCP");
        if (Ethernet.hardwareStatus() == EthernetNoHardware) {
            Serial.println("Інтернет модуль не знайдено");
        }
        if (Ethernet.linkStatus() == LinkOFF) {
            Serial.println("Інтернет кабель не під'єднано");
```

```
52          }
53          Ethernet.begin(mac, ip, myDns);
54      } else {
55          Serial.print("  DHCP встановлений IP ");
56          Serial.println(Ethernet.localIP());
57      }
58      delay(1000);
59      Serial.print("під'єднання до ");
60      Serial.print(server);
61      Serial.println("...");
62
63      if (client.connect(server, HTTP_PORT)) {
64          Serial.print("під'єднано ");
65          Serial.println(client.remoteIP());
66      } else {
67          Serial.println("зв'язок з сервером не встановлено");
68      }
69  }
70
71  void loop() {
72
73      if (client.connect(server, HTTP_PORT)) {
74
75          float humidity = dht.readHumidity();
76          float temperature = dht.readTemperature();
77          float airPollution = mq.getPPM() / 10;
78          Serial.println(
79                  "Вологість: " + String(humidity) + " %\t" +
80                  "Температура: " + String(temperature) + " *C\t");
81
82          humidityDiff = getPercentDifference(humidity, recHumidity);
83          temperatureDiff =
84              getPercentDifference(temperature, recTemperature);
85          totalDiff = temperatureDiff + humidityDiff;
86
87          printDifference();
88
89          const size_t capacity = JSON_OBJECT_SIZE(5);
90          DynamicJsonBuffer jsonBuffer(capacity);
91
92          float adjustmentFactor = fullValue + totalDiff;
93          float airQuality = fullValue - airPollution;
94
95          JsonObject &root = jsonBuffer.createObject();
96          root["adjustmentFactor"] = adjustmentFactor;
97          root["humidity"] = humidity;
98          root["id"] = id;
99          root["airQuality"] = airQuality;
100         root["temperature"] = temperature;
101
102         String data;
103         root.printTo(data);
104
105         client.println("POST " + PATH_NAME + " HTTP/1.1");
106         client.println("Host: " + HOST_NAME);
107         client.println("User-Agent: Arduino/1.0");
108         client.println("Connection: close");
```

```
109         client.println("Content-Type: application/json");
110         client.print("Content-Length: ");
111         client.println(data.length());
112         client.println();
113         client.println(data);
114 
115         printResponse(client);
116 
117     }
118 
119     delay(100000);
120 
121 }
122 
123 float getPercentDifference(float x, float y) {
124     float difference;
125     if (x > y) {
126         difference = 1 - (y / x);
127     } else {
128         difference = 1 - (x / y);
129     }
130     return difference;
131 }
132 
133 void printResponse(EthernetClient client) {
134     while (client.connected()) {
135         if (client.available()) {
136             char c = client.read();
137             Serial.print(c);
138         }
139     }
140 }
141 
142 void printDifference() {
143     Serial.println(
144             "Відхилення вологості від норми: " +
145             String(humidityDiff * 100) + " %\t");
146 
147     Serial.println(
148             "Відхилення температури від норми: " +
149             String(temperatureDiff * 100) + " %\t");
150 
151     Serial.println(
152             "Загальне відхилення: " +
153             String(totalDiff * 100) + " %\t");
154 }
```

ДОДАТОК В

Частина програмної реалізації серверної взаємодії з IoT

```
@Override
public PlacementDto updateSmartDevice(SmartDeviceDto smartDeviceDto) {
    Optional<Placement> placementDto = placementRepository
            .findById(smartDeviceDto.getId());

    if (placementDto.isPresent()) {
        Placement placement = placementDto.get();

        SmartDevice smartDevice = placement.getSmartDevice();
        double previousAdjustmentFactor =
                smartDevice.getAdjustmentFactor();
        double adjustmentFactor =
                round(smartDeviceDto.getAdjustmentFactor());
        double airQuality = round(smartDeviceDto.getAirQuality());

        Date currentDate = new Date();
        placement.getContracts().stream()
                .filter(contract -> contract.getDate()
                    .after(currentDate))
                .forEach(contract -> {
                    double price = contract.getPrice();
                    if (previousAdjustmentFactor != 0) {
                        price /= previousAdjustmentFactor;
                    }
                    if (adjustmentFactor != 0) {
                        price *= adjustmentFactor;
                    }
                    contract.setPrice(round(price));
                });

        double dirtinessFactor =
            round((1 - airQuality) + (adjustmentFactor - 1));

        smartDevice
                .setAirQuality(airQuality)
                .setTemperature(smartDeviceDto.getTemperature())
                .setHumidity(smartDeviceDto.getHumidity())
                .setAdjustmentFactor(adjustmentFactor)
                .setDirtinessFactor(dirtinessFactor)
                .setPriority(smartDeviceDto.getPriority());
        placement.setSmartDevice(smartDevice);
        System.out.println(smartDevice);
        return PlacementMapper
                .toPlacementDto(placementRepository.save(placement));
    }

    return null;
}
```