МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ

УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Звіт

з лабораторної роботи №5

з дисципліни

«Аналіз та рефакторинг коду»

|  |  |
| --- | --- |
|  | Перевірила:  доц. каф. Програмної Інженерії  Побіженко І. О.  Виконала:  ст. гр. ПЗПІ-19-6  Бахмат Є. Ю. |

Харків 2021

**ВСТУП**

Темою даної лабораторної роботи є розробка програмного забезпечення для  IoT або SmartDevice пристрою, створеного на базі будь-якої поширеної на  сьогодні платформи, придатної для реалізації вбудованих систем (Embedded  System) системи за темою: «Програмна система універсального клімат-контролю».

Метою роботи є отримання навичок з проектування програмного забезпечення для  IoT або SmartDevice пристрою, створеного на базі будь-якої поширеної на  сьогодні платформи, придатної для реалізації вбудованих систем (Embedded  System) з використанням Use-Case діаграми, діаграми взаємодії, діаграми діяльності та діаграми компонентів. У ході роботи повинен бути розроблений робочий SmartDevice для програмної системи та відповідна супроводжувальна документація.

Хід даної лабораторної роботи має такий вигляд:

1. Розробити Use Case діаграму з зображеними на ній акторами та їх взаємодією із системою.
2. Розробити UML діаграму взаємодії.
3. Розробити UML діаграму діяльності;
4. Розробити діаграму компонентів.
5. Детально описати методи взаємодії програмних компонентів між собою та серверною частиною;
6. Зробити висновки по роботі.

**1 USE CASE**

Користувачами системи є 3 типи акторів: адміністратори системи, власники та співробітники. Власник та адміністратор взаємодіють із системою через веб-застосунок, а співробітники - за допомогою мобільного застосунку. Розумні пристрої контролюються як веб-застосунком так і мобільним застосунком. Виходячи з цього була створена UML діаграма прецедентів (див. додаток А. рис. А.1).

За допомогою  веб-застосунку адміністратори системи можуть отримати інформацію про стан усіх елементів системи, схеми систем контролю клімату підконтрольних будівель та їх складові частини, такі, як кімнати, пристрої, сенсори та їх дані, а також створювати резервні копії стану програмної платформи. Власник може використовувати веб-застосунок для створення, редагування та видалення організаційної інформації, наприклад: інформацію про будівлі, кімнати або співробітників, змінювати права доступу персоналу, реєструвати нових співробітників, тощо.

Крім того, власнику доступні перегляд статистики, усі CRUD операції системи, у рамках свого бізнесу, окрім зміни персональної інформації користувачів.

Усі операції потребують авторизації користувача. Усі користувачі мають можливість відновити пароль за допомогою електронної пошти. Авторизація через веб додаток доступна лише для робітників та адміністраторів, а для клієнтів – лише через мобільний додаток.

**2 СТРУКТУРА ПРОЕКТУ**

В ході лабораторної роботи було розроблено IoT пристрій на базі плати Arduino Nano, програмне забезпечення для IoT пристрою та програмне забезпечення для взаємодії із бекенд-частиною. Для розробки ПЗ для IoT пристрою було використано мову програмування C/C++, для взаємодії із бекенд-частиною було використано застосунок на NodeJS.

Пристрій складається із 4 частин:

1. Мікроконтролер Arduino Nano;
2. Сенсор DHT11 для виміру вологості та температури повітря;
3. Сенсор MQ-7 для виміру рівня чадного газу.
4. Сервопривід Tower Pro SG90 для демонстрації закриття заслонок.

Прилад власний серійний номер для комунікації з системою. Діаграма компонентів системи наведена у додатку Б.

Взаємодія із серверною частиною відбувається безпосередньо завдяки відповідним REST-запитам до endpoints серверу за певними URL-адресами. Дані передаються та повертаються у вигляді JSON-формату. Для виконання запитів використовується бібліотека Axios. Керування пристроєм реалізовано за допомогою технології WebSocket та бібліотеки Socket.io.

**3 ПОВЕДІНКА СИСТЕМИ**

Для розробки функціоналу систему були створені UML діаграми діяльності та взаємодії. Їх можна подивитися у додатках В та Г відповідно.

Діаграма діяльності відображає поведінку системи при різних діях користувача. Точкою старту на діаграмі є ввімкнення пристрою. Далі відбуваються відгалуження, які описують поведінку системи в основних ситуаціях її роботи.

Діаграма взаємодії описує поведінку тільки в межах одного [варіанта використання](https://uk.wikipedia.org/wiki/%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D0%BF%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82%D1%96%D0%B2). На такій діаграмі прийнято відображати екземпляри об'єктів та повідомлення, якими ці об'єкти обмінюються один з одним в рамках даного варіанта використання.

Основна бізнес логіка системи полягає у вимірі даних клімату навколишнього середовища, передача інформації до бекенд-частини, виконання команд бекенд-частини. Цей код можна переглянути у додатках Д та Е.

**ВИСНОВКИ**

У ході виконання лабораторної роботи були отримані навички з розробки програмного забезпечення для  IoT або SmartDevice пристрою, створеного на базі будь-якої поширеної на  сьогодні платформи, придатної для реалізації вбудованих систем (Embedded  System), розробки сценаріїв використання Use Case діаграми. Було проведене прогнозування поведінки системи (діаграми діяльності та взаємодії), визначені відношення компонентів у системі (діаграма компонентів). Результатом роботи є ІоТ-пристрій для розроблюваної програмної системи.

Проекти ПЗ IoT-пристрою знаходяться за посиланнями:

<https://drive.google.com/drive/folders/17NlIvUFzabdIwTfSeRAqNutCxiNscXUr?usp=sharing>

<https://drive.google.com/drive/folders/14H9ngv4Apas1QNQ6xEez-I6HTkm8CfcC?usp=sharing>

Контрольна хеш-сума архіву з проектом за алгоритмом md5: b6495a1b7c9b56dcda0197f266d3add8 \*climatizer-iot-reciver.rar

9ed34697fa3dba671862926221b1f04e \*climatizer-iot.rar

ДОДАТОК А

Use Case діаграма

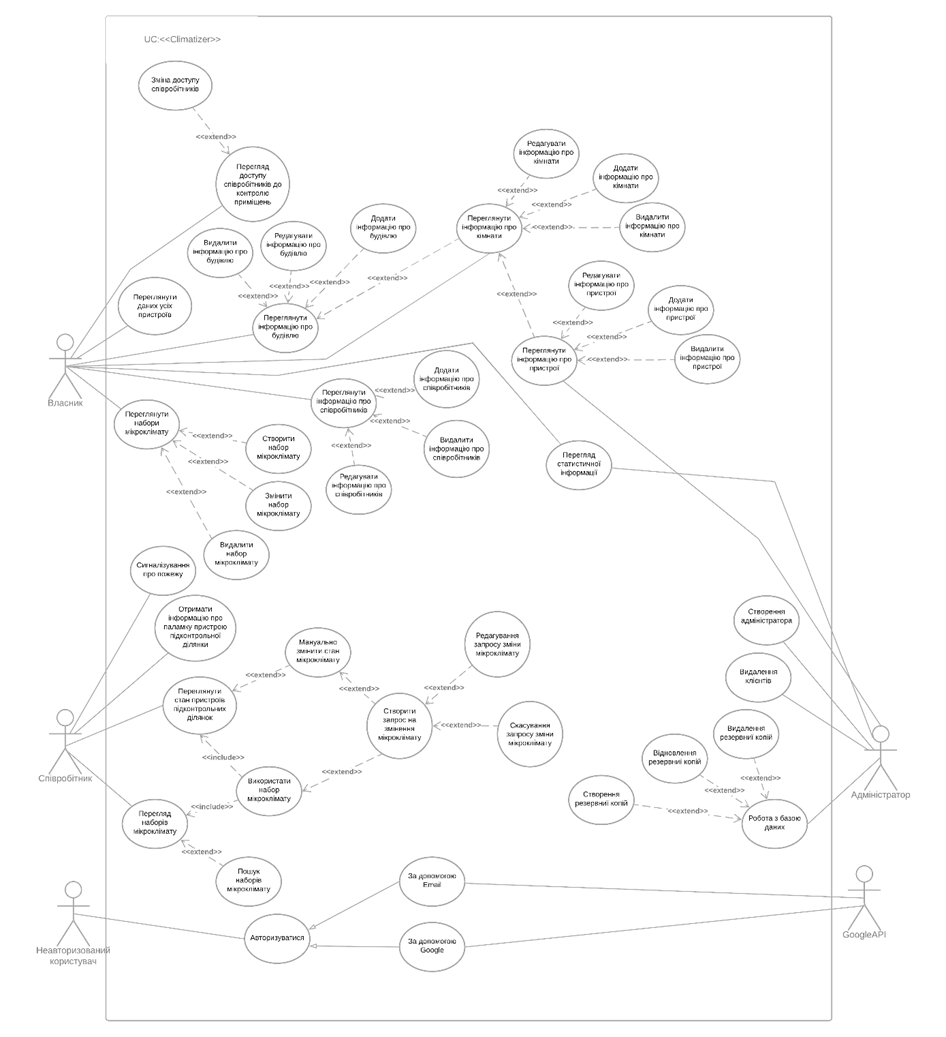


Рисунок А.1 – Use Case діаграма проекту «Програмна система універсального клімат-контролю»

ДОДАТОК Б

UML діаграма компонентів

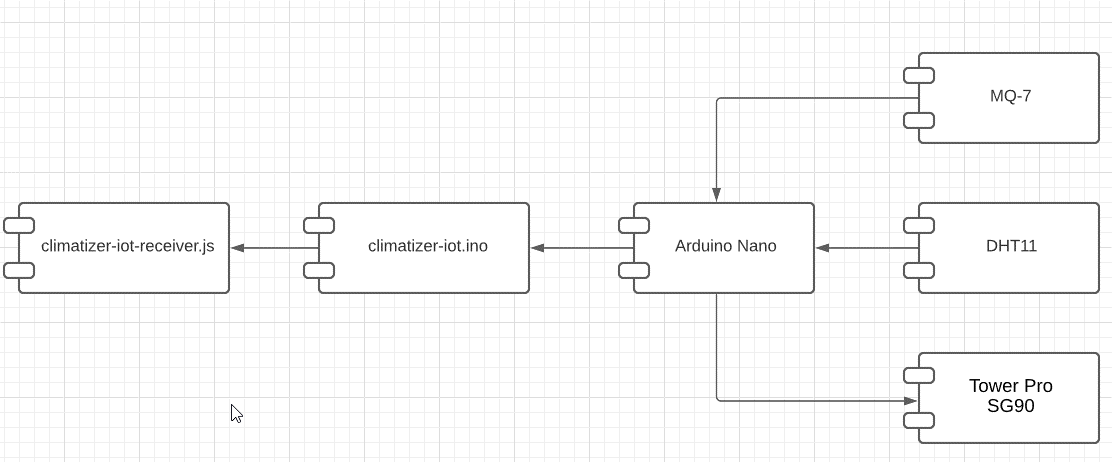


Рисунок Б.1 – UML діаграма компонентів ІоТ-пристрою проекту «Програмна система універсального клімат-контролю»

ДОДАТОК В

UML діаграма діяльності

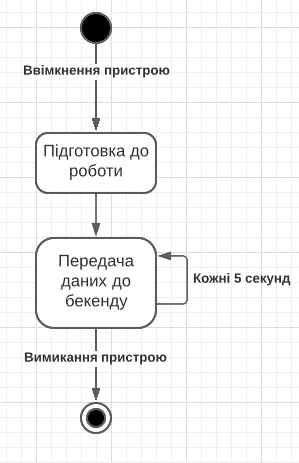


Рисунок В.1 – UML діаграма діяльності ІоТ-пристрою проекту «Програмна система універсального клімат-контролю»

ДОДАТОК Г

UML діаграма взаємодії

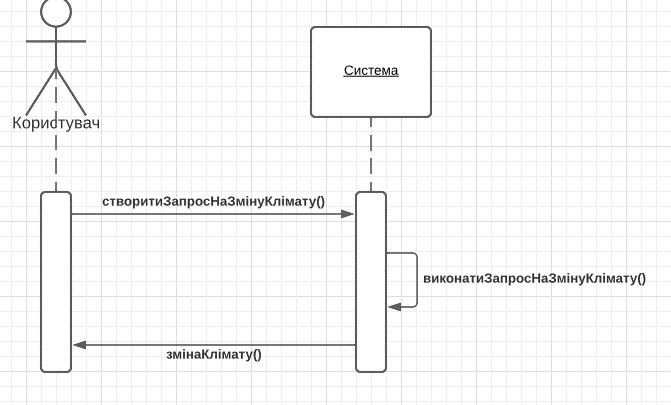


Рисунок Г.1 – UML діаграма взаємодії ІоТ-пристрою проекту «Програмна система універсального клімат-контролю»

ДОДАТОК Д

Код взаємодії із бекенд-частиною

const SerialPort = require('serialport')

const socket = require('socket.io')

const axios = require('axios')

const dotenv = require('dotenv')

const { io } = require('socket.io-client')

dotenv.config()

const host = process.env['HOST']

const deviceId = process.env['DEVICE\_ID']

const server = `http://${host}/device/data/`

const serialPort = new SerialPort('COM6', {

  baudRate: 9600,

  autoOpen: false,

})

const socket = io(server)

serialPort.open((err) => {

  if (err) {

    console.log('Error opening port: ', err.message)

  }

})

serialPort.on('open', () => {

  socket.on('close-gate', () => {

    serialPort.write(1)

  })

  socket.on('open-gate', () => {

    serialPort.write(2)

  })

  serialPort.on('data', (measurements) => {

    const data = measurements.toString()

    if (measurements.includes('CO')) {

      const humidity = data.split(';')[0].split(':')[1]

      const temperature = data.split(';')[1].split(':')[1]

      const carbonMonoxide = data.split(';')[2].split(':')[1]

      const body = {

        DeviceId: deviceId,

        H: humidity,

        T: temperature,

        CO: carbonMonoxide,

        DateTime: Date.now(),

      }

      axios

        .post(server, body)

        .catch((error) => console.error(error.data))

        .then((res) => console.log(res))

    }

  })

})

ДОДАТОК Е

Код IoT-пристрою

#include "DHT.h"

#include <Servo.h>

#define DHTTYPE DHT11

#define DHTPIN 2

#define MQPIN A0

void getDHTData();

void getMQData();

DHT dht(DHTPIN, DHTTYPE);

Servo servo;

float humidity = 0;

float temperature = 0;

int gasCount = 0;

int incomingByte = 0;

void closeTheGate(){

 //TODO close the gate;

  servo.write(90);

}

void openTheGate(){

  servo.write(0);

}

void setup()

{

  Serial.begin(9600);

  servo.attach(9);

  Serial.println(F("Warming up!"));

  delay(9000);

  Serial.println(F("Measuring"));

  dht.begin();

};

void loop()

{

  if(Serial.available()>0){

    incomingByte = Serial.read();

    if(incomingByte == '1'){

      Serial.print("Ok ill close the gate");

      closeTheGate();

    }

    if(incomingByte == '2'){

      Serial.print("Ok ill open the gate");

      openTheGate();

    }

    Serial.print("I recieved:");

    Serial.println(incomingByte);

  }

  servo.write(90);

  delay(100);

  servo.write(0);

   getDHTData();

   getMQData();

   Serial.print("\n");

  delay(5000);

};

void getDHTData()

{

  humidity = dht.readHumidity();

  temperature = dht.readTemperature();

  Serial.print(F("H:"));

  Serial.print(humidity);

  Serial.print(F(";T:"));

  Serial.print(temperature);

  Serial.print(F(";"));

}

void getMQData()

{

  gasCount = analogRead(MQPIN);

  Serial.print(F("CO:"));

  Serial.print(gasCount);

  Serial.print(F(";"));

}