ДОДАТОК А

Код прошивки ESP-32

Лістинг А.1 – Підключення бібліотек

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESP32Servo.h>
```

Лістинг А.2 – Константи та змінні.

```
const char* ssid = "Chill out";
const char* password = "nekit";
const int sensorTemp1Pin = 36;
const int sensorTemp2Pin = 39;
const int sensorLight1Pin = 32;
const int sensorLight2Pin = 33;
const int sensorGas1Pin = 34;
const int sensorGas2Pin = 35;
const int buzzerPin = 27;
bool buzzerState = false;
const int led1Pin = 26;
const int led2Pin = 25;
int led1Brightness = 0;
int led2Brightness = 0;
const int servo1Pin = 12;
const int servo2Pin = 14;
bool servo1State = false;
bool servo1PreviousState = false;
bool servo2State = false;
bool servo2PreviousState = false;
Servo servo1;
Servo servo2;
const int pwn1Pin = 11;
const int pwn1Pin = 10;
bool cooler1State = false;
bool cooler2State = false;
WebServer server(80);  // Створення веб-сервера на порту 80
```

Лістинг А.3 – Ініціалізація потрібних модулів

```
void setup() {
  pinMode(buzzerPin, OUTPUT);
  digitalWrite(buzzerPin, LOW);
  delay(100);
  servo1.attach(servo1Pin);
  servo2.attach(servo2Pin);
  Serial.begin(115200);
  WiFiSetup();
  ledcSetup(0, 1000, 8);
```

```
    ledcAttachPin(buzzerPin, 0);
  }
```

Лістинг А.4 – Підключення до Wi-Fi

```
void WiFiSetup() {
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
      delay(1000);
      Serial.println("Connecting to WiFi...");
    }
    Serial.println("Connected to WiFi");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
    // Налаштування маршрутів веб-сервера
    server.on("/getValue", HTTP_GET, GetValueFromSensors);
    server.on("/setBuzzer", HTTP_POST, HandleBuzzer);
    server.on("/setLed", HTTP_POST, HandleLED);
    server.on("/setServo", HTTP_POST, HandleServo);
    server.on("/setFan", HTTP_POST, HandleFan);
    server.begin();
    Serial.println("HTTP server started");
  }
```

Лістинг А.5 – Функція loop()

```
void loop() {
    server.handleClient();
    SetCurrentBuzzer();
    SetCurrentLed();
    SetCurrentServo();
    SetCurrentFan();
  }
```

Лістинг А.6 – Функція GetValueFromSensors()

```
void GetValueFromSensors() {
    float sensor1TempValue = (analogRead(sensorTemp1Pin) * (3.3 /
4095.0)) * 100 * 1.65;
    float sensor2TempValue = (analogRead(sensorTemp2Pin) * (3.3 /
4095.0)) * 100 * 1.75;
    int gas1Value = analogRead(sensorGas1Pin);
    int gas2Value = analogRead(sensorGas2Pin);
    String response = "Tem 1: " + String(sensor1TempValue) + " |" +
"Tem 2: " + String(sensor2TempValue) + " |" + "Lux 1: " +
String(analogRead(sensorLight1Pin)) + " |" + "Lux 2: " +
String(analogRead(sensorLight2Pin)) + " |" + "Gas 1: " +
String(gas1Value / 2) + " |" + "Gas 2: " + String(gas2Value / 2);
    // Створення HTTP-відповіді
    server.send(200, "text/plain", response);
  }
```

Лістинг А.7 – Обробка кінцевої точки пискавки.

```
  void HandleBuzzer() {
```

```
  if (!server.hasArg("plain")) {
    server.send(400, "text/plain", "Missing body");
    return;
  }
  String body = server.arg("plain");
  if (body == "true") {
    buzzerState = true;
    server.send(200, "text/plain", "Buzzer ON");
  } else if (body == "false") {
    buzzerState = false;
    server.send(200, "text/plain", "Buzzer OFF");
  } else {
    server.send(400, "text/plain", "Invalid argument");
  }
}
```

Лістинг А.8 – Обробка кінцевої точки світлодіода.

```
void HandleLED() {
  if (!server.hasArg("plain")) {
    server.send(400, "text/plain", "Missing body");
    return;
  }
  String body = server.arg("plain");
  int underscoreIndex = body.indexOf('_');
  if (underscoreIndex == -1) {
    server.send(400, "text/plain", "Invalid format. Expected
value_room format.");
    return;
  }
  String brightnessStr = body.substring(0, underscoreIndex);
  String roomStr = body.substring(underscoreIndex + 1);
  int brightness = brightnessStr.toInt();
  int room = roomStr.toInt();
  if (brightness < 0 || brightness > 255) {
    server.send(400, "text/plain", "Invalid brightness value. Must
be between 0 and 255.");
    return;
  }
  if (room == 1) {
    led1Brightness = brightness;
    server.send(200, "text/plain", "LED 1 brightness set to " +
String(brightness));
  } else if (room == 2) {
    led2Brightness = brightness;
    server.send(200, "text/plain", "LED 2 brightness set to " +
String(brightness));
  } else {
    server.send(400, "text/plain", "Invalid room number. Must be 1
or 2.");
  }
}
```

Лістинг А.9 – Обробка кінцевої точки сервопривіда.

```
      void HandleServo() {
        if (!server.hasArg("plain")) {
          server.send(400, "text/plain", "Missing body");
          return;
        }
        String body = server.arg("plain");
        int underscoreIndex = body.indexOf('_');
        if (underscoreIndex == -1) {
          server.send(400, "text/plain", "Invalid format. Expected
value_room format.");
          return;
        }
        String stateServo = body.substring(0, underscoreIndex);
        String roomStr = body.substring(underscoreIndex + 1);
        int room = roomStr.toInt();
        bool isON = (stateServo == "True");
        if (room == 1) {
          servo1State = isON;
          server.send(200, "text/plain", "Servo 1 " + String(servo1State
? "ON" : "OFF"));
        } else if (room == 2) {
          servo2State = isON;
          server.send(200, "text/plain", "Servo 2 " + String(servo2State
? "ON" : "OFF"));
        } else {
          server.send(400, "text/plain", "Invalid room number. Must be 1
or 2.");
        }
        Serial.println("Servo1State: " + String(servo1State) + " |
Servo2State: " + String(servo2State));
      }
```

Лістинг А.10 – Обробка кінцевої точки вентиляторів.

```
void HandleFan() {
    if (!server.hasArg("plain")) {
      server.send(400, "text/plain", "Missing body");
      return;
    }
    String body = server.arg("plain");
    int underscoreIndex = body.indexOf('_');
    if (underscoreIndex == -1) {
      server.send(400, "text/plain", "Invalid format. Expected
value_room format.");
      return;
    }
    String stateFan = body.substring(0, underscoreIndex);
    String roomStr = body.substring(underscoreIndex + 1);
    int room = roomStr.toInt();
```

```
    bool isON = (stateFan == "True");
    if (room == 1) {
      fan1State = isON;
      server.send(200, "text/plain", "Fan 1 " + String(fan1State ? "ON"
: "OFF"));
    } else if (room == 2) {
      fan2State = isON;
      server.send(200, "text/plain", "Fan 2 " + String(fan2State ? "ON"
: "OFF"));
    } else {
      server.send(400, "text/plain", "Invalid room number. Must be 1 or
2.");
    }
    Serial.println("Fan1State: " + String(fan1State) + " | Fan2State: "
+ String(fan2State));
  }
```

Лістинг А.11 – Функція SetCurrentLed()

```
void SetCurrentLed() {
  analogWrite(led1Pin, led1Brightness);
  analogWrite(led2Pin, led2Brightness);
}
```

Лістинг А.12 – Функція змінення стану пискавки

```
void SetCurrentBuzzer(){
  if (buzzerState) {
    ledcWriteTone(0, 1000);
  } else {
    ledcWriteTone(0, 0);
  }}
```

Лістинг А.13 – Функція змінення стану сервоприводів

```
void SetCurrentServo() {
  if (servo1PreviousState != servo1State) {
    servo1PreviousState = servo1State;
    if (servo1State) {
      servo1.write(180);
      delay(1000);
    } else {
      servo1.write(0);
      delay(1000);
    }
  }
  if (servo2PreviousState != servo2State) {
    servo2PreviousState = servo2State;
    if (servo2State) {
      servo2.write(180);
      delay(1000);
    } else {
      servo2.write(0);
```

```
      delay(1000);
    }
  }
}
```

Лістинг А.14 – Функція змінення стану вентиляторів

```
void SetCurrentFan() {
if (cooler1State) {
      analogWrite(pwm1Pin, 255);
else {
      analogWrite(pwm1Pin, 0);
}
if (cooler2State) {
      analogWrite(pwm2Pin, 255);
else {
      analogWrite(pwm2Pin, 0);
}
}
```

ДОДАТОК Б

SQL запити до MSSQL

Лістинг Б.1 – SQL запит для створення БД та таблиць

```
-- Створення бази даних
CREATE DATABASE SensorHubDB;
-- Використання створеної бази даних
USE SensorHubDB;
-- Створення таблиць
CREATE TABLE Room (
    Id INT PRIMARY KEY,
    RoomName NVARCHAR(100) NOT NULL
);
CREATE TABLE HeatSensor (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    RoomId INT,
    TemperatureInCelsius FLOAT,
    DateRecord DATETIME,
    FOREIGN KEY (RoomId) REFERENCES Room(Id)
);
CREATE TABLE LightSensor (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    RoomId INT,
    LightInLux INT,
    DateRecord DATETIME,
    FOREIGN KEY (RoomId) REFERENCES Room(Id)
);
CREATE TABLE GasSensor (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    RoomId INT,
    AirQualityValue INT,
    DateRecord DATETIME,
    FOREIGN KEY (RoomId) REFERENCES Room(Id)
);
CREATE TABLE HeatSettings (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    RoomId INT NOT NULL,
    StartTime TIME NOT NULL,
    EndTime TIME NOT NULL,
    Value FLOAT NOT NULL,
);
CREATE TABLE LightSettings (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    RoomId INT NOT NULL,
    StartTime TIME NOT NULL,
    EndTime TIME NOT NULL,
    Value FLOAT NOT NULL,
);
CREATE TABLE GasSettings (
```

```
    Id INT IDENTITY(1,1) PRIMARY KEY,
    RoomId INT NOT NULL,
    StartTime TIME NOT NULL,
    EndTime TIME NOT NULL,
    Value FLOAT NOT NULL,
);
CREATE TABLE CompanyInformation (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    Region NVARCHAR(MAX) NOT NULL,
);
```

ДОДАТОК В

Код з проекту Avalonia

Лістинг В.1 – Програмний код класу Room

```
public class Room {
          public int Id;
    public string RoomName;
    public List<HeatSensor> HeatSensors { get; set; }
    public List<HeatSettings> HeatSettings { get; set; }
    public List<LightSensor> LightSensors { get; set; }
    public List<LightSettings> LightSettings { get; set; }
    public List<GasSensor> GasSensors { get; set; }
    public List<GasSettings> GasSettings { get; set; }   }
```

Лістинг В.2 – Програмний код класу LightSensor

```
public class LightSensor {
    public int Id { get; set; }
    public int RoomId { get; set; }
    public Room Room { get; set; }
    public int LightInLux { get; set; }
    public DateTime DateRecord { get; set; }
}
```

Лістинг В.3 – Програмний код класу LightSettings

```
public class LightSettings {
    public int Id { get; set; }
    public int RoomId { get; set; }
    public TimeOnly StartTime { get; set; }
    public TimeOnly EndTime { get; set; }
    public double Value { get; set; } }
```

Лістинг В.4 – Код класу GasSensor

```
public class GasSensor  {
    public int Id { get; set; }
    public int RoomId { get; set; }
    public Room Room { get; set; }
    public int AirQualityValue { get; set; }
    public DateTime DateRecord { get; set; }
}
```

Лістинг В.5 – Код класу GasSettings

```
public class GasSettings  {
    public int Id { get; set; }
    public int RoomId { get; set; }
    public TimeOnly StartTime { get; set; }
    public TimeOnly EndTime { get; set; }
    public double Value { get; set; }
}
```

Лістинг В.6 – Код класу HeatSensor

```
public class HeatSensor  {
    public int Id { get; set; }
    public int RoomId { get; set; }
    public Room Room { get; set; }
    public double TemperatureInCelsius { get; set; }
    public DateTime DateRecord { get; set; }
}
```

Лістинг В.7 – Код класу HeatSettings

```
public class HeatSettings  {
    public int Id { get; set; }
    public int RoomId { get; set; }
    public TimeOnly StartTime { get; set; }
    public TimeOnly EndTime { get; set; }
    public double Value { get; set; }
}
```

Лістинг В.8 – Код класу CompanyInformation

```
public class CompanyInformation  {
    public int Id { get; set; }
    public string Region { get; set; }
}
```

Лістинг В.9 – Код класу для взаємодії з БД’

```
public class AppDbContext : DbContext {
    public DbSet<Room> Room { get; set; }
    public DbSet<LightSensor> LightSensor { get; set; }
    public DbSet<HeatSensor> HeatSensor { get; set; }
    public DbSet<GasSensor> GasSensor { get; set; }
    public DbSet<HeatSettings> HeatSettings { get; set; }
    public DbSet<LightSettings> LightSettings { get; set; }
    public DbSet<GasSettings> GasSettings { get; set; }
    public DbSet<CompanyInformation> CompanyInformation { get; set; }
    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)  {
optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=Sen
sorHubDB;Trusted_Connection=True;");  }
    protected override void OnModelCreating(ModelBuilder modelBuilder)
{
        modelBuilder.Entity<Room>()
            .HasKey(room => room.Id);
        modelBuilder.Entity<Room>()
            .Property(room => room.RoomName)
            .IsRequired();
        modelBuilder.Entity<HeatSensor>()
            .HasKey(heatSensor => heatSensor.Id);
        modelBuilder.Entity<HeatSensor>()
            .HasOne(heatSensor => heatSensor.Room)
```

```
            .WithMany(room => room.HeatSensors)
            .HasForeignKey(heatSensor => heatSensor.RoomId);
        modelBuilder.Entity<LightSensor>()
            .HasKey(lightSensor => lightSensor.Id);
        modelBuilder.Entity<LightSensor>()
            .HasOne(lightSensor => lightSensor.Room)
            .WithMany(room => room.LightSensors)
            .HasForeignKey(lightSensor => lightSensor.RoomId);
        modelBuilder.Entity<GasSensor>()
            .HasKey(gasSensor => gasSensor.Id);
        modelBuilder.Entity<GasSensor>()
            .HasOne(gasSensor => gasSensor.Room)
            .WithMany(room => room.GasSensors)
            .HasForeignKey(gasSensor => gasSensor.RoomId); }  }
```

Лістинг В.10 – Інтерфейс для RoomService

```
public interface IRoomService {
    public List<string> GetAllNames();
    public Task<Dto.Room?> GetByNameWithHeatSensorAsync(string name);
    public Task<Dto.Room?> GetByNameWithLightSensorAsync(string name);
    public Task<Dto.Room?> GetByNameWithGasSensorAsync(string name);
    public Task<Dto.Room?> GetFirstRoomWithHeatSensorAsync();
    public Task<Dto.Room?> GetFirstRoomWithLightSensorAsync();
    public Task<Dto.Room?> GetFirstRoomWithGasSensorAsync(); }
```

Лістинг В.11 – Власний DI сервіс

```
    public static class CustomServiceCollection {
    private static readonly Dictionary<Type, object> Services = new ();
    static CustomServiceCollection()  {
        var dbContext = new AppDbContext();
        AddService(dbContext);
        AddService<IHeatSensorService>(new
HeatSensorService(dbContext));
        AddService<IHeatSettingsService>(new
HeatSettingsService(dbContext));
        AddService<ILightSensorService>(new
LightSensorService(dbContext));
        AddService<ILightSettingsService>(new
LightSettingsService(dbContext));
        AddService<IGasSensorService>(new
GasSensorSensorService(dbContext));
        AddService<IGasSettingsService>(new
GasSettingsService(dbContext));
        AddService<IRoomService>(new RoomService(dbContext));
        AddService<ICompanyInformationService>(new
CompanyInformationService(dbContext));
        AddService<IArduinoHandler>(new
ArduinoHandler(GetService<IHeatSensorService>(),GetService<ILightSensorS
ervice>(), GetService<IGasSensorService>()));
        AddService<IAirAlertHandler>(new AirAlertHandler());
        AddService<ILightHandler>(new
```

```
LightHandler(GetService<IArduinoHandler>(),GetService<ILightSettingsServ
ice>()));
        AddService<IBuzzerHandler>(new
BuzzerHandler(GetService<IArduinoHandler>(),GetService<ICompanyInformati
onService>(), GetService<IAirAlertHandler>()));
        }
    public static void AddService<TService>(TService service)  {
        Services[typeof(TService)] = service;
    }
    public static TService GetService<TService>()  {
        if (Services.TryGetValue(typeof(TService), out var service)) {
            return (TService)service;
        }
        else   {
            Console.WriteLine($"Service of type {typeof(TService)} not
found in the collection.");
            return default(TService);
     }  }   }
```

Лістинг В.12 – Приклад використання DI

```
public partial class LightSensorView : UserControl  {
    private IRoomService _roomService;
    public LightSensorView() {
        _roomService =
CustomServiceCollection.GetService<IRoomService>();
} }
```

Лістинг В.13 – Інтерфейс класу для взаємодії з ESP-32

```
public interface IArduinoHandler {
    public event EventHandler<IList<object>> OnlineDataUpdated;
    public Task Execute();
    public Task UpdateEvent();
    public Task SetBuzzerState(bool state);
    public Task SetLedBrightness(int brightness, int room);
    public Task SetServo(bool isOn, int room);
}
```

Лістинг В.14 – Клас для взаємодії з ESP-32

```
public class ArduinoHandler : IArduinoHandler {
    private readonly HttpClient _httpClient;
    private readonly DispatcherTimer _dataUpdateTimer;
    private readonly DispatcherTimer _onlineDataUpdated;
    private readonly IHeatSensorService _heatService;
    private readonly ILightSensorService _lightService;
    private readonly IGasSensorService _gasSensorService;
    public event EventHandler<IList<object>> OnlineDataUpdated;

    private readonly string _httpArduinoServerPath;

    public ArduinoHandler(IHeatSensorService heatService,
```

```
ILightSensorService lightService, IGasSensorService gasSensorService) {
        _heatService = heatService;
        _lightService = lightService;
        _gasSensorService = gasSensorService;
        _httpClient = new HttpClient();
        _dataUpdateTimer = new DispatcherTimer();
        _onlineDataUpdated = new DispatcherTimer();
        _httpArduinoServerPath = "http://192.168.1.4/";
    }
    public async Task Execute()  {
        _dataUpdateTimer.Interval = TimeSpan.FromSeconds(60);
        _dataUpdateTimer.Tick += UpdateAndSaveToDbDataAsync;
        _dataUpdateTimer.Start();
        _onlineDataUpdated.Interval = TimeSpan.FromSeconds(1);
        _onlineDataUpdated.Tick += UpdateDataAsync;
        _onlineDataUpdated.Start();
        UpdateAndSaveToDbDataAsync(new object(), EventArgs.Empty);
    }
    public async Task UpdateEvent()  {
        UpdateDataAsync(new object(), EventArgs.Empty);
    }
    private async void UpdateAndSaveToDbDataAsync(object? sender,
EventArgs eventArgs) {
        try   {
            var result = ArduinoOutputValueParser.Parse(await
GetArduinoResponse());
            await SaveToDbAsync(result);
            Console.WriteLine("ArduinoHandler.cs | method:
ArduinoHandler | Data Saved");
        }
        catch (Exception ex)   {
            Console.WriteLine("ArduinoHandler.cs | method:
UpdateAndSaveToDbDataAsync | Произошла ошибка: " + ex.Message);
        }
    }
  private async void UpdateDataAsync(object? sender, EventArgs
eventArgs) {
        try {
            var result = ArduinoOutputValueParser.Parse(await
GetArduinoResponse());
            OnlineDataUpdated?.Invoke(this, result);
        }
        catch (Exception ex)   {
            Console.WriteLine("ArduinoHandler.cs | method:
UpdateDataAsync | Произошла ошибка: " + ex.Message);
        }
    }
    private async Task<string> GetArduinoResponse()  {
        HttpResponseMessage response = await
_httpClient.GetAsync(_httpArduinoServerPath + "getValue");
        response.EnsureSuccessStatusCode();
```

```
        return await response.Content.ReadAsStringAsync();
    }
    public async Task SetBuzzerState(bool state)  {
        string stateValue = state ? "true" : "false";
        var content = new StringContent(stateValue, Encoding.UTF8,
"text/plain");
        await _httpClient.PostAsync(_httpArduinoServerPath +
"setBuzzer", content);
    }
    public async Task SetLedBrightness(int brightness, int room)  {
        if (brightness < 0 || brightness > 255)  {
            throw new ArgumentOutOfRangeException(nameof(brightness),
"Brightness must be between 0 and 255.");
        }
        string brightnessValue = brightness + "_" + room;
        var content = new StringContent(brightnessValue, Encoding.UTF8,
"text/plain");
        await _httpClient.PostAsync(_httpArduinoServerPath + "setLed",
content);
    }
    public async Task SetServo(bool isOn, int room)  {
        string stateServo = isOn+ "_" + room;
        var content = new StringContent(stateServo, Encoding.UTF8,
"text/plain");
        await _httpClient.PostAsync(_httpArduinoServerPath + "setServo",
content);
    }
    private async Task SaveToDbAsync(IList<object> data)  {
        foreach (var value in data)  {
            switch (value)  {
                case HeatSensor heatSensor:
                    await _heatService.AddAsync(heatSensor);
                    break;
                case LightSensor lightSensor:
                    await _lightService.AddAsync(lightSensor);
                    break;
                case GasSensor gasSensor:
                    await _gasSensorService.AddAsync(gasSensor);
                    break;
}  }  }  }
```

Лістинг В.15 – Клас для взаємодії API повітряної тривоги

```
 public class AirAlertHandler : IairAlertHandler {
    private readonly HttpClient _httpClient;
    private readonly DispatcherTimer _onlineDataUpdated;
    public event EventHandler<IList<AirAlert>> OnlineDataUpdated;
    public List<string> AllRegions { get; set; }
    private readonly string _httpAirAlertListServerPath;
    private readonly string _httpAirAlertMapServerPath;
    public AirAlertHandler() {
        _httpClient = new HttpClient();
```

```
        _onlineDataUpdated = new DispatcherTimer();

        _httpAirAlertListServerPath =
"http://ubilling.net.ua/aerialalerts/";
        _httpAirAlertMapServerPath =
"https://ubilling.net.ua/aerialalerts/?map=nightmode";
        Dispatcher.UIThread.Post(() => SetAllRegionsAsync(),
DispatcherPriority.Background);
    }
    public async Task Execute()  {
        _onlineDataUpdated.Interval = TimeSpan.FromSeconds(5);
        _onlineDataUpdated.Tick += UpdateDataAsync;
        _onlineDataUpdated.Start();
    }
    public async Task<IList<AirAlert>> GetAirAlertListAsync()  {
        try   {
            string json = await GetJsonFromServerAsync();
            IList<AirAlert> airAlertList = DeserializeAirAlerts(json);
            return airAlertList;
        }
        catch (HttpRequestException ex)    {
            Console.WriteLine($"\"AirAlertHandler.cs | method:
GetAirAlertListAsync | HTTP request failed: {ex.Message}");
            return new List<AirAlert>();
        }
        catch (Exception ex)  {
            Console.WriteLine($"AirAlertHandler.cs | method:
GetAirAlertListAsync | Error: {ex.Message}");
            return new List<AirAlert>();
        }
    }
    private async Task SetAllRegionsAsync() {
        try  {
            string json = await GetJsonFromServerAsync();
            IList<AirAlert> airAlertList = DeserializeAirAlerts(json);
            AllRegions = airAlertList.Select(airAlert =>
airAlert.Region).ToList();
        }
        catch (HttpRequestException ex)  {
            Console.WriteLine($"AirAlertHandler.cs | method:
SetAllRegionsAsync | HTTP request failed: {ex.Message}");
            AllRegions = new List<string>();
        }
        catch (Exception ex) {
            Console.WriteLine($"AirAlertHandler.cs | method:
SetAllRegionsAsync |Error: {ex.Message}");
            AllRegions = new List<string>();
        }
    }
    private void UpdateDataAsync(object? sender, EventArgs e)  {
        Task.Run(async () =>  {
```

```
            try  {
                string json = await GetJsonFromServerAsync();
                IList<AirAlert> airAlertList =
DeserializeAirAlerts(json);
                OnlineDataUpdated?.Invoke(this, airAlertList);
            }
            catch (HttpRequestException ex)  {
                Console.WriteLine($"AirAlertHandler.cs | method:
UpdateDataAsync | HTTP request failed: {ex.Message}");
            }
            catch (Exception ex) {
                Console.WriteLine($"AirAlertHandler.cs | method:
UpdateDataAsync | Error: {ex.Message}");
            }
        });
    }

    private async Task<string> GetJsonFromServerAsync() {
        return await
_httpClient.GetStringAsync(_httpAirAlertListServerPath);
    }
    private static IList<AirAlert> DeserializeAirAlerts(string
jsonString)  {
        dynamic jsonObject = JsonConvert.DeserializeObject(jsonString);
        var states = jsonObject.states;
        var airAlerts = new List<AirAlert>();
        foreach (var state in states)   {
            string region = state.Name;
            bool alertNow = state.Value.alertnow;
            DateTime changedState =
DateTime.ParseExact(state.Value.changed.ToString(), "yyyy-MM-dd
HH:mm:ss", System.Globalization.CultureInfo.InvariantCulture);
            airAlerts.Add(new AirAlert { Region = region, AlertNow =
alertNow, ChangedState = changedState });
        }
        return airAlerts;
    }
}
```

ДОДАТОК Г

Демонстраційний матеріал у вигляді презентації

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра КІТАР

КВАЛІФІКАЦІЙНА РОБОТА

На тему: Розробка системи автоматизації для управління приладами життєзабезпечення на підприємстві

Виконав:
ст. гр. АКТАКІТ-20-1
Седов М.А.

Керівник:
проф. каф. КІТАР
Сезонов І.К.

Актуальність теми

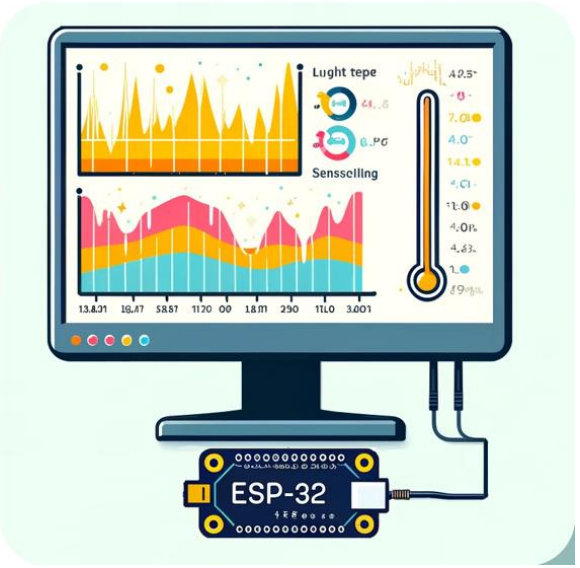
У сучасному світі значення автоматизації процесів на підприємствах зростає, адже її мета - це полегшення та оптимізації робочого процесу людини на підприємствах, це дозволяє покращити продуктивність, зменшити витрати на ресурси що позитивно впливає на його дохідність та підвищити рівень безпеки. Також це впливає на екологічність підприємства, адже автоматизація дозволяє вирішити надмірне споживання ресурсів, що позитивно пливає на екологічний стан. Дозволяє скоротити що зараз найважливіше це споживання електроенергії, адже через її дефіцит піднялась як її вартість, так і складність роботи через її обмеженість.

Мета атестаційної роботи

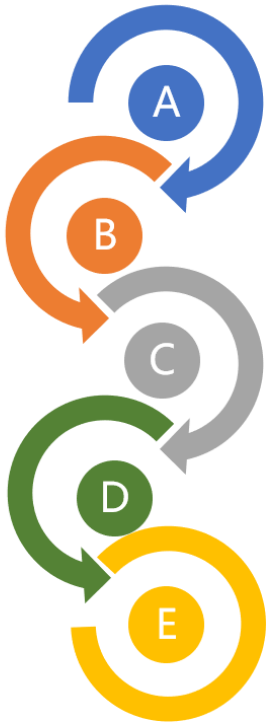
Метою кваліфікаційної роботи підвищення ефективності збору та аналізу даних з датчиків та управління виконавчими приладами через комп’ютерний додаток.

Об’єкт дослідження: процес управління приладами життєзабезпечення на підприємстві

Предмет розробки: Автоматизована система зчитування показників з датчиків, обробки та управління станом приладів.



Задачі роботи



Побудувати схему підключення апаратної частини

Обрати оптимальні компоненти для створення макету

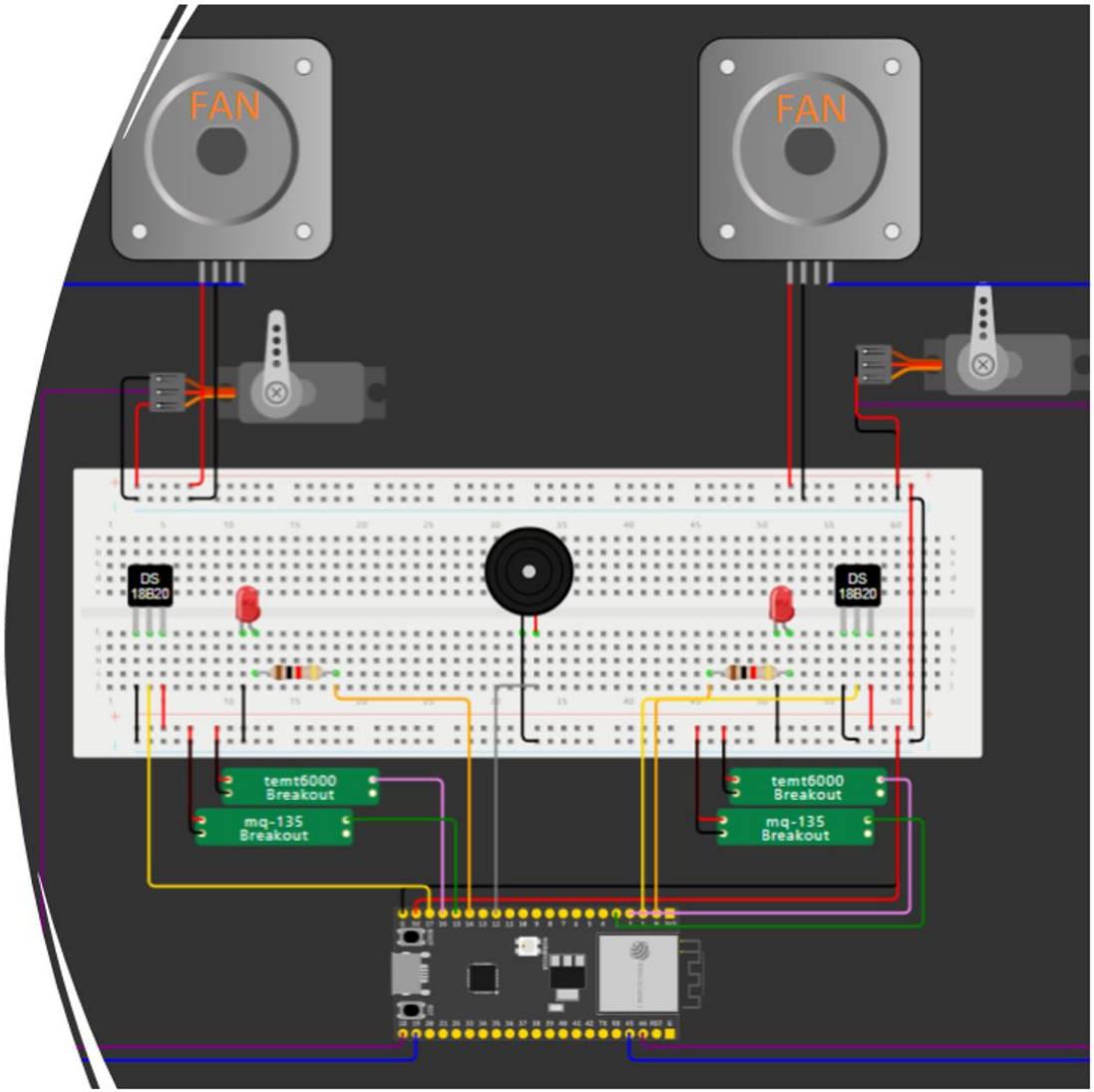
Написати код для обміну даними між макетом та ком’ ютерим додатком

Реалізувати серверну частину комп’ ютерного додатку та взаємдії з БД

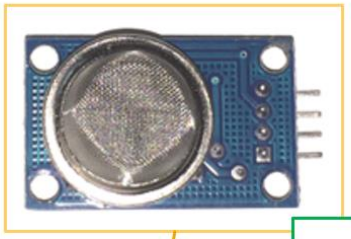
Написати інтерфейс для користувача

Схема підключення

Планування та розробка схеми підключення апаратної частини проекту вирішує кілька ключових аспектів. Вона допомагає визначити вимоги та компоненти, забезпечує гладку інтеграцію, дозволяє ідентифікувати потенційні проблеми та оптимізувати витрати, оскільки дозволяє точніше оцінити необхідні компоненти.

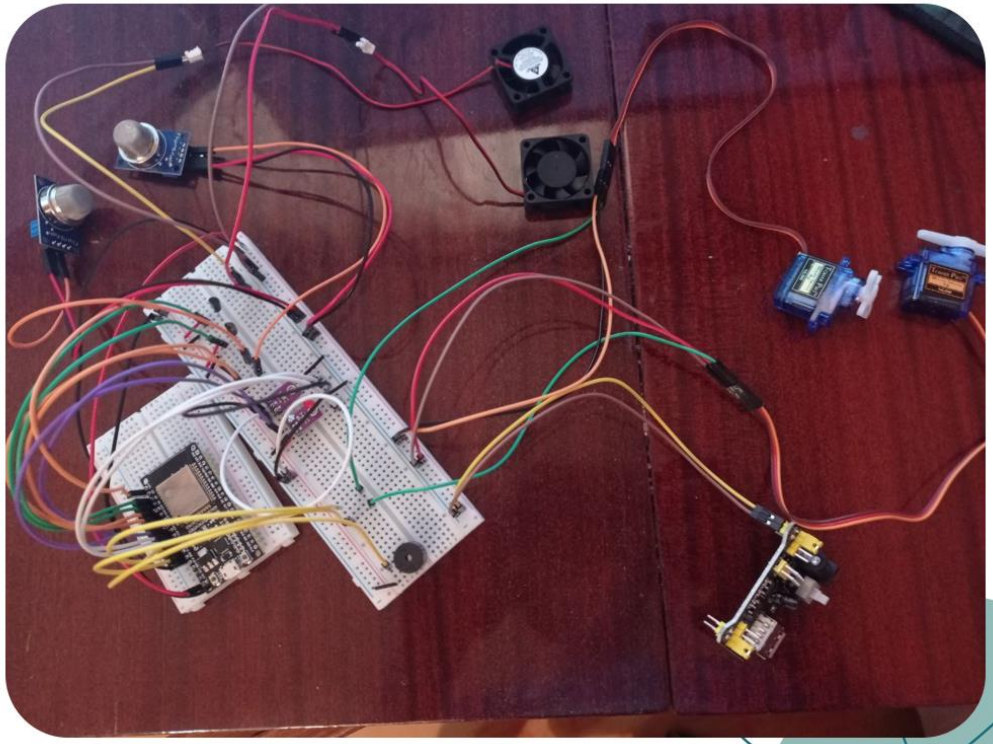


Компоненти системи



Складений макет

Основою макетної моделі системи слугує мікроконтролер ESP-32. Модель включає в себе різноманітні сенсори, та виконавчі прилади для демонстрації функціональності розробленої системи.



Програмування плати ESP-32

Код пишеться на мові C++

Ініціюються піні для датчиків та виконавчих приладів

Первичне налаштування пискавки та серводвигунів

Передача даних з датчиків

Підключаються бібліотеки для Wi-Fi, серводвигунів та ESP-32

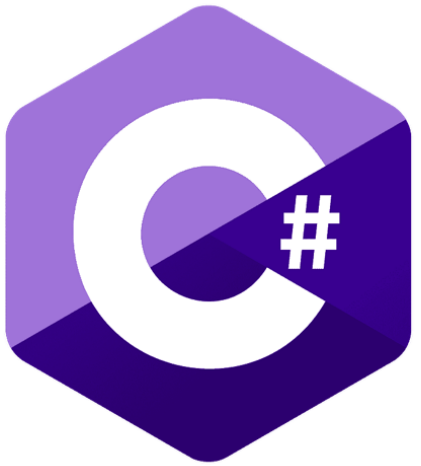
Ініціюються необхідні змінні

Налаштування Wi-Fi

Обробка налаштувань на виконавчі приладів

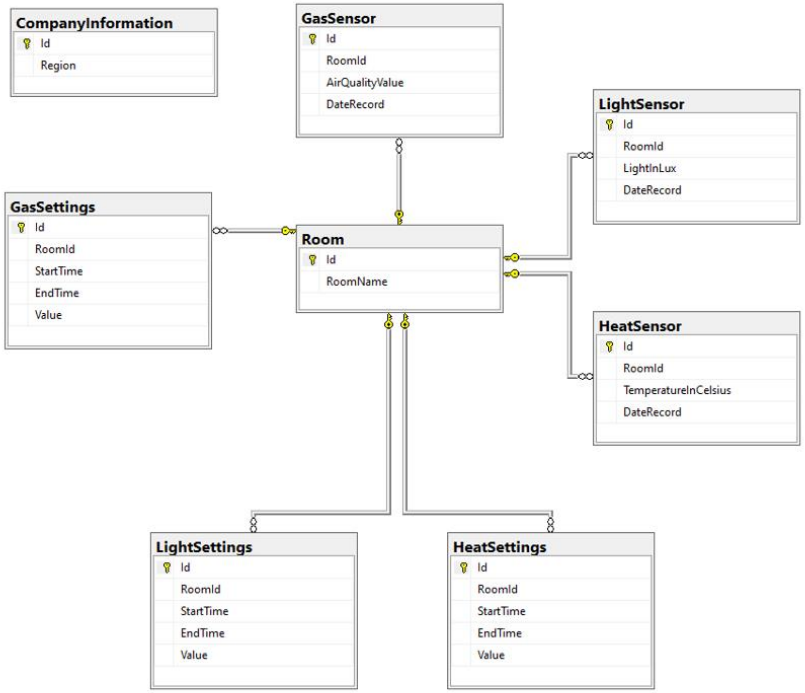
Серверна частина проекту

Серверна частина проекту будується на використанні БД MSSQL та мові програмування C#



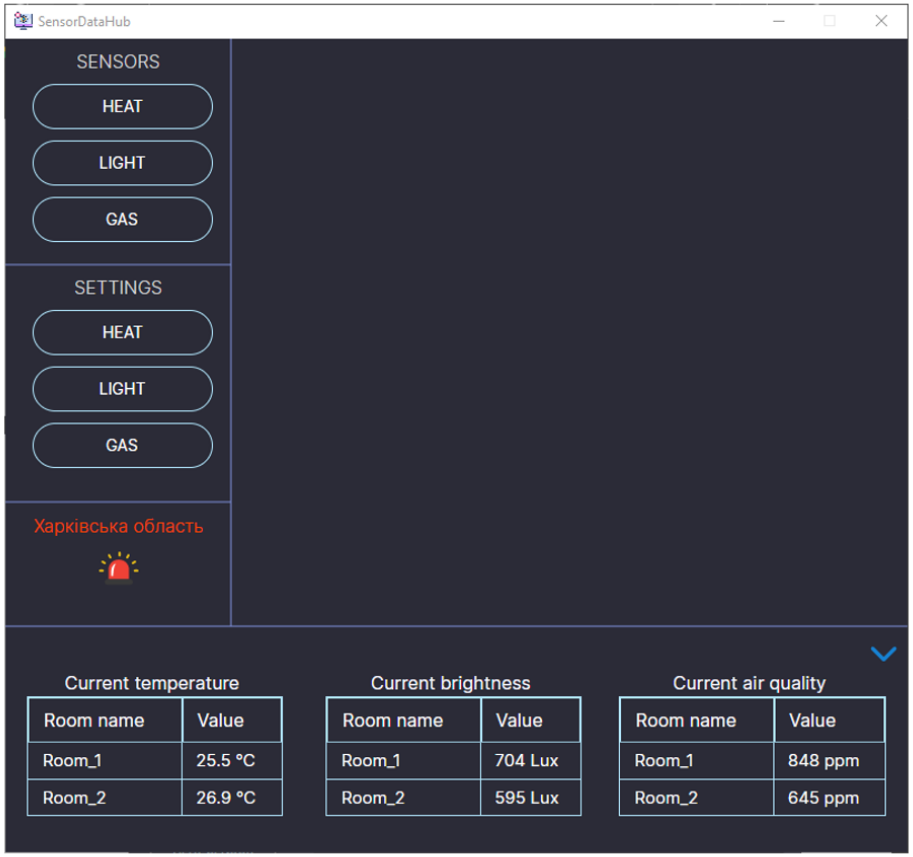
Серверна частина проекту

Графічне відображення структури БД



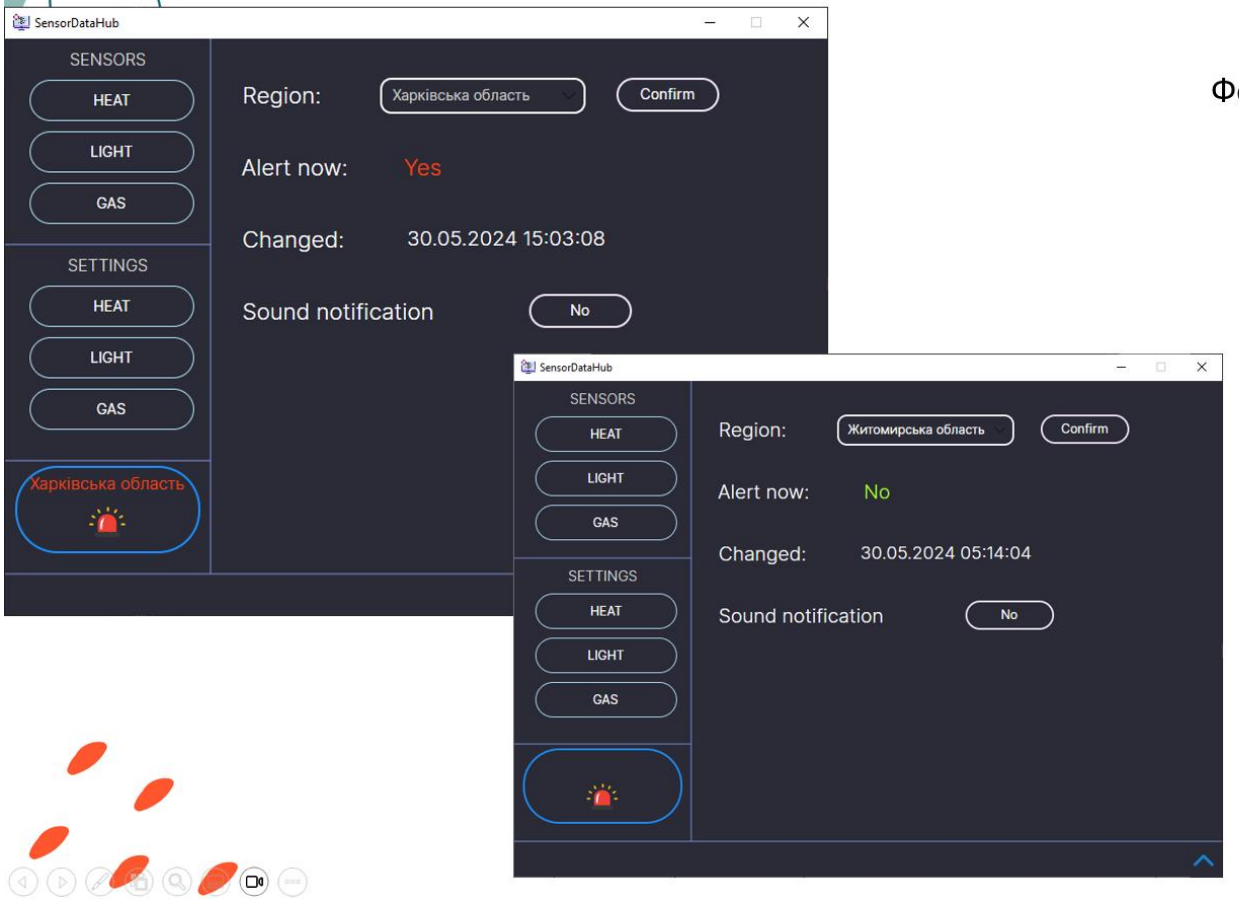
Microsoft® SQL Server®

Розробка інтерфейсу користувача



Avalonia

Розробка інтерфейсу користувача



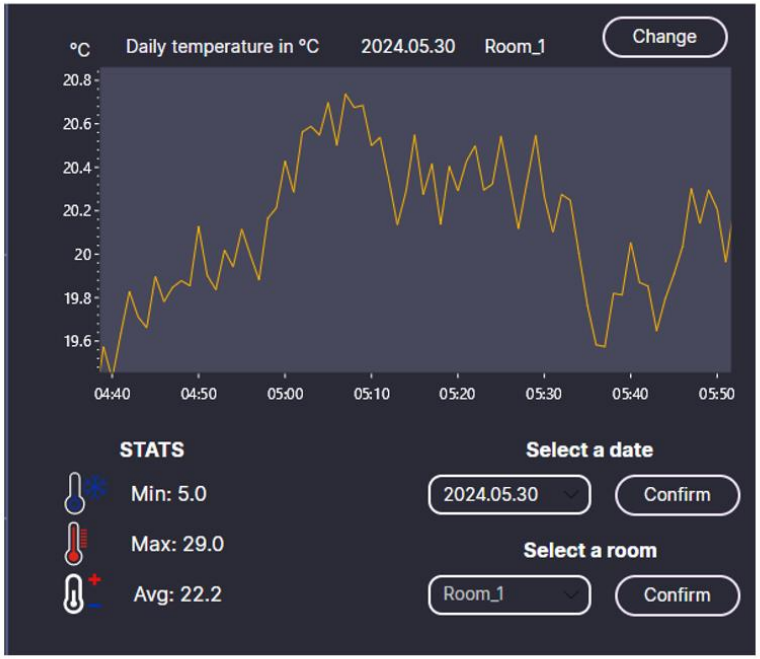
Формат відповідей від API повітряних тривог

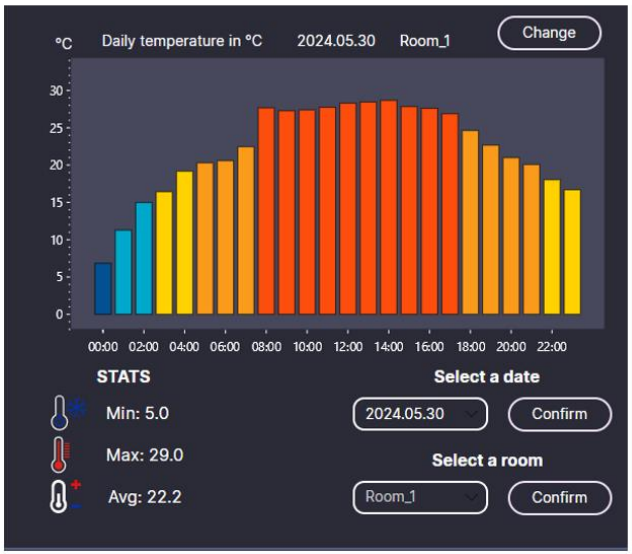
```
−<root>
   <source>Andrew Dunai API</source>
   <cachedat>2022-04-27 15:07:14</cachedat>
  −<states>
   −<state0>
      <name>Вінницька область</name>
      <changed>2022-04-27 10:40:07</changed>
      <alertnow>false</alertnow>
    </state0>
   −<state1>
      <name>Волинська область</name>
      <changed>2022-04-27 10:39:56</changed>
      <alertnow>false</alertnow>
    </state1>
```

Avalonia

Розробка інтерфейсу користувача

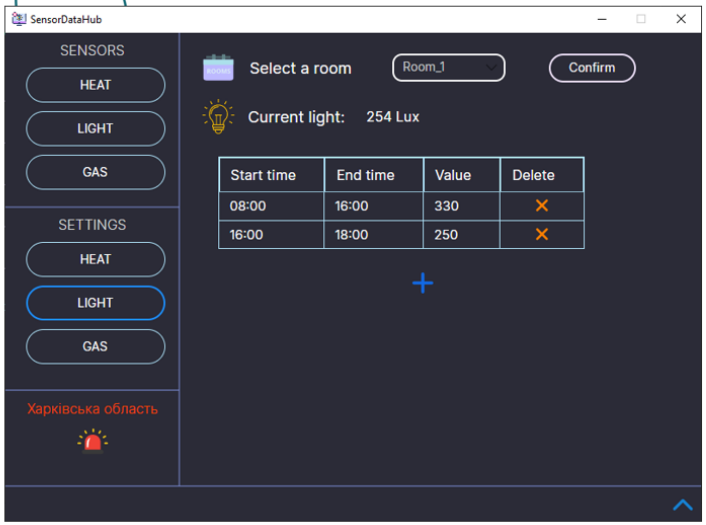
Розробка інтерфейсу користувача

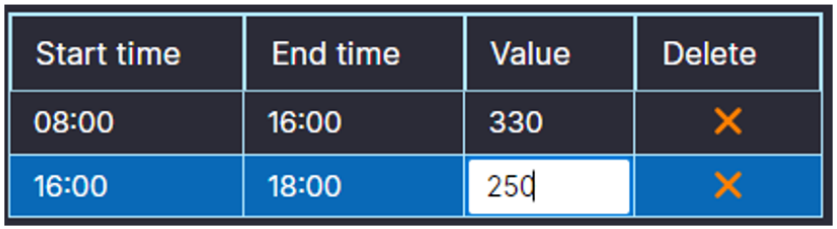


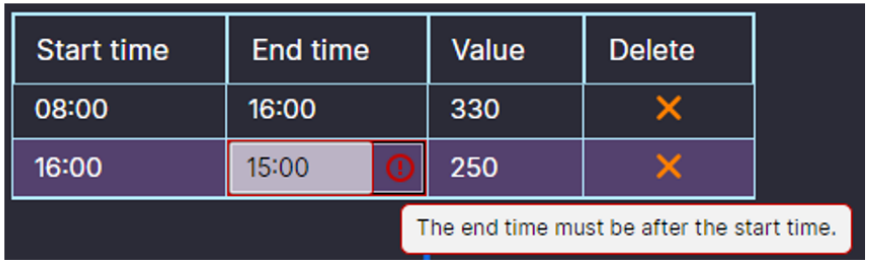


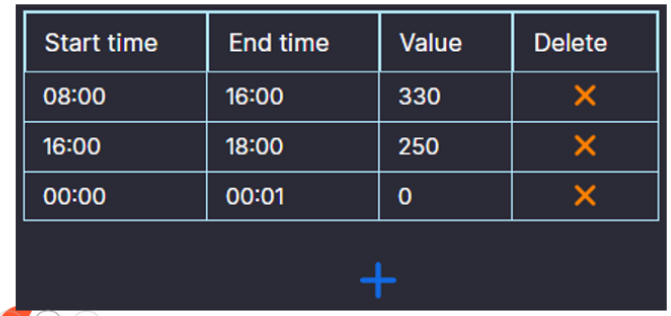
Avalonia

Розробка інтерфейсу користувача









Avalonia

Висновки

В ході виконання кваліфікаційної роботи було проведено аналіз існуючих систем автоматизації приладів. На основі існуючих систем було підкреслені важливі моменти, щоб покращити їх та зробити макет і десктоп додаток для взаємодії з датчиками та виконавчими приладами.

В ході розробки додатку та макету було досягнуто всі поставлені завдання. Була реалізована функціональність для відображення усіх даних по обраному дню та приміщенню які приходять з макету за допомогою фреймворку Avalonia, мові програмування C# та базі даних MSSQL. Відображення даних на різних графіків та статистики. Реалізована можливість робити настройку для виконавчих приладів для обраної кімнати та по обраному проміжку часу з можливістю редагування у додатку. Добавлене сповіщення про повітряні тривоги з можливостю налаштування.

Також були розглянуті питання з охорони праці та теорії автоматичного управління.

Система моніторингу вирізняється кількома перевагами над аналогами. Вона може працювати без електропостачання, використовуючи лише 5-ватне джерело живлення, та підтримує оповіщення про повітряні тривоги, забезпечуючи безпеку. Також система дозволяє індивідуально налаштовувати параметри для кожного приміщення відповідно до зазначених часових інтервалів, що надає більш гнучке та ефективне управління в порівнянні з іншими системами, які пропонують лише загальні налаштування.

Для подальшого розвитку додатку важливо зосередитись на розширенні функціональності. Додавання нових можливостей, таких як додавання різної кількості однакових датчиків на одне приміщення для більш точного вимірювання, додання нових видів сенсорів для вимірювання інших параметрів, та їх обробку. Також треба зосередитись на поліпшенні безпеки.

Дякую за увагу!

ст. гр. АКТАКІТ-20-1
Седов М.А.

ДОДАТОК Д

Відомість кваліфікаційної роботи

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 442351.011 ПЗ	Пояснювальна записка	78 с., А4
		Додаткові матеріали	
2		Демонстраційний матеріал у	17 с., А4
		вигляді презентації	

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК. 442351.011 ПЗ		
Розроб.		Седов М.А.			Розробка системи автоматизації для управління приладами життєзабезпечення на підприємстві	Літера	Аркуш	Аркушів
Перевір.		Сезонова І.К.				Н	1	1
						Кафедра КІТАМ ХНУРЕ		
Н.контр.		Стародубцев М.Г.						
Затв.		Невлюдов І.Ш.						