ДОДАТОК А

Код програми для Arduino UNO

Лістинг А.1 – файл Arduino/router.ino

```
#define SERIAL_PORT 115200

#define PRIVATE_ID «rtr01»

#define MQTT_SERVER_ADDRESS «192.168.50.11»
#define MQTT_SERVER_PORT 1883
#define MQTT_CLIENT_ID «arduino-router-1»
#define MQTT_AUTH_USERNAME «UserName»
#define MQTT_AUTH_PASSWORD «Password»

#define MQTT_ENDPOINT_VERITY «verify»
#define MQTT_ENDPOINT_CONTROL «control»

#define RF24_CE_PIN 7
#define RF24_CSN_PIN 8

#define RADIO_PAYLOAD_SIZE 32
#define RADIO_SEND_MAX_RETRY_COUNT 5

#define DISPLAY_CLK_PIN 2
#define DISPLAY_DIO_PIN 3


#include <SPI.h>
#include <Ethernet.h>
#include <PubSubClient.h>
#include <NRF24L01.h>
#include <RF24.h>
#include <printf.h>
#include <TM1637Display.h>


// Ethernet settings
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
char mqttServer[] = MQTT_SERVER_ADDRESS;
int mqttPort = MQTT_SERVER_PORT;
EthernetClient ethClient;
PubSubClient mqttClient(ethClient);

// NRF24L01 settings
RF24 radio(RF24_CE_PIN, RF24_CSN_PIN);

// Display settings
/*
  --A–
 F    B
 |--G--
 –    C
 |    -
*/
```

```
TM1–7Display display(DISPLAY_CLK_PIN, DISPLAY_DIO_PIN);
uint8_t d_char = SEG_B | SEG_C | SEG_D | SEG_E | SEG_G;
uint8_t e_char = SEG_A | SEG_D | SEG_E | SEG_F | SEG_G;
uint8_t i_char = SEG_C;
uint8_t n_char = SEG_C | SEG_E | SEG_G;
uint8_t o_char = SEG_C | SEG_D | SEG_E | SEG_G;
uint8_t r_char = SEG_E | SEG_G;
uint8_t t_char = SEG_D | SEG_E | SEG_F | SEG_G;

// Global variables
char verifyMqttTopic[sizeof(MQTT_ENDPOINT_VERITY) + sizeof(PRIVATE_ID) + 2] {};
char controlMqttTopic[sizeof(MQTT_ENDPOINT_CONTROL) + sizeof(PRIVATE_ID) + 2] {};
bool isError = false;
unsigned long displayClearTime = 0;


void setup();
void loop();
bool setupEthernet();
void setupMqtt();
void ensureConnectMQTT();
void mqttCallback(char* topic, uint8_t* payload, unsigned int length);
void handleControlCallback(uint8_t* payload, unsigned int length);
void handleVerifyCallback(uint8_t* payload, unsigned int length);
bool setupRadio();
void sendMessageToModule(char* moduleId, uint8_t* data, uint8_t dataLength);
void sendRadioMessage(uint8_t* pipeAddress, uint8_t* payload, uint8_t payloadLength);
void setupDisplay();
void showInitText();
void showDoneText();
void showErrorText();
void showCodeOnDisplay(unsigned int code);


void setup() {
  Serial.begin(SERIAL_PORT);
  printf_begin();

  setupDisplay();
  showInitText();

  bool setupEthernetResult = setupEthernet();
  bool setupRadioResult = setupRadio();

  if (!setupEthernetResult || !setupRadioResult) {
    isError = true;
    showErrorText();
    return;
  }

  setupMqtt();
  showDoneText();
}


void loop() {
```

```
  if (isError) {
   return;
  }

  if (displayClearTime != 0 && millis() >= displayClearTime) {
   display.clear();
   displayClearTime = 0;
  }

  ensureConnectMQTT();
  mqttClient.loop();
 }


 bool setupEthernet() {
  for (byte attempt = 1; attempt <= 5; attempt++) {
   if (Ethernet.begin(mac) == 1) {
    printf("Ethernet«init success\n");
    »return true;
   }
   printf("Ethernet«init error. Retry %i\n", attemp»);
   delay(1000);
  }
  printf("Ethernet«init error. Abort\n");
  ret»rn false;
 }


 void setupMqtt() {
  strcat(verifyMqttTopic, MQTT_ENDPOINT_VERITY);
  strcat(verifyMqttTopic, "/");
  s«r»at(verifyMqttTopic, PRIVATE_ID);

  strcat(controlMqttTopic, MQTT_ENDPOINT_CONTROL);
  strcat(controlMqttTopic, "/");
  s«r»at(controlMqttTopic, PRIVATE_ID);

  mqttClient.setServer(mqttServer, mqttPort);
  mqttClient.setCallback(mqttCallback);
  ensureConnectMQTT();
 }


 void ensureConnectMQTT() {
  while (!mqttClient.connected()) {
   if (mqttClient.connect(MQTT_CLIENT_ID, MQTT_AUTH_USERNAME,
MQTT_AUTH_PASSWORD)) {
    printf("MQTT Bro«er (re)connect success\n");

   » if (!mqttClient.subscribe(verifyMqttTopic))
     printf("Fail sub«cribe. MQTT topic \"%s\"\n",»ver»fy»qttTopic);
    else
     printf("Success «ubscribe. MQTT topic \"%s\"\n",»ver»fy»qttTopic);

    if (!mqttClient.subscribe(controlMqttTopic))
     printf("Fail sub«cribe. MQTT topic \"%s\"\n",»con»ro»MqttTopic);
```

```
    else
      printf("Success «ubscribe. MQTT topic \"%s\"\n",»con»ro»MqttTopic);
   } else {
    printf("MQTT Bro«er connection error, rc=%d\n", mqttCl»ent.state());
    delay(2000);
   }
  }
}


void mqttCallback(char* topic, byte* payload, unsigned int length) {
 printf("\nMsg on«MQTT Topic \"%s\"\n",»top»c)»

 if (strcmp(topic, verifyMqttTopic) == 0)
  handleVerifyCallback(payload, length);
 else if (strcmp(topic, controlMqttTopic) == 0)
  handleControlCallback(payload, length);
}


void handleVerifyCallback(uint8_t* payload, unsigned int length) {
 if (length != 4) {
  printf("Verify M«TT msg invalid. Payload size != 4. Abort\n");
  r»turn;
 }

 unsigned int code = 0;
 for (byte i = 0; i < 4; i++) {
  int symbolAsInt = payload[i] - '0';
 – ‘f’(symbolAsInt < 0 || symbolAsInt > 9) {
    printf("Varify M«TT msg invalid. Value is not digit: %i. Abort\n", payloa»[i]);
    return;
  }
  code = code * 10 + symbolAsInt;
 }

 showCodeOnDisplay(code);
}


void handleControlCallback(uint8_t* payload, unsigned int length) {
 if (length < 6) {
  printf("Control «QTT msg invalid. Payload size < 6. Abort\n");
  r»turn;
 }

 // pipeAddress is the moduleId
 uint8_t pipeAddress[6] {};
 unsigned int dataLength = length - 5;
 ui–t8_t data[dataLength]{};

 memcpy(pipeAddress, payload, 5);
 pipeAddress[5] = '\0';
 m‘mc’y(data, payload + 5, dataLength);

 sendRadioMessage(pipeAddress, data, dataLength);
```

```
}


bool setupRadio() {
  if (!radio.begin()) {
    printf("RF24 ini« fail. Abort");
    r»turn false;
  }

  radio.setAutoAck(1);
  radio.setPayloadSize(RADIO_PAYLOAD_SIZE);
  radio.setChannel(0x60);
  radio.setPALevel(RF24_PA_LOW);
  radio.setDataRate(RF24_1MBPS);
  radio.powerUp();
  radio.stopListening();

  printf("RF24 ini« success\n\n");
  rad»o.printDetails();
  printf("\n");
  «ad»o.printPrettyDetails();
  printf("\n");
  «et»rn true;
}


void sendRadioMessage(uint8_t* pipeAddress, uint8_t* payload, uint8_t payloadLength) {
  radio.flush_tx();
  radio.openWritingPipe(pipeAddress);

  for (byte i = 0; i < RADIO_SEND_MAX_RETRY_COUNT; i++) {
    bool success = radio.write(payload, payloadLength);
    if (success) {
      printf("RF24 msg«sent\n");
     »return;
    }
    printf("Error on«RF24 msg. Retry %i\n", i + 1)»
  }
  printf("Error on«RF24 msg. Abort\n");
}


v»id setupDisplay() {
  display.setBrightness(0x0f);
  display.clear();
}


void showInitText() {
  uint8_t data[] { i_char, n_char, i_char, t_char };
  display.clear();
  display.setSegments(data);
}


void showDoneText() {
```

```
  uint8_t data[] { d_char, o_char, n_char, e_char };
  display.clear();
  display.setSegments(data);
  displayClearTime = millis() + 5000;
}


void showErrorText() {
  uint8_t data[] { 0, e_char, r_char, r_char };
  display.clear();
  display.setSegments(data);
}


void showCodeOnDisplay(unsigned int code) {
  if (code / 10000 > 0) {
    printf("Code %i «ontains > 4 digits. Abort\n", code);»    return;
  }

  display.clear();
  display.showNumberDec(code, true);
  displayClearTime = millis() + 60000;
}
```

Лістинг А.2 – файл Arduino/module-single-led.ino

```
#define SERIAL_PORT 115200

#define PRIVATE_ID "mdl01"

“defin” RF24_CE_PIN 9
#define RF24_CSN_PIN 10

#define RADIO_PAYLOAD_SIZE 32

#define LED_PIN 6


#include <SPI.h>
#include <NRF24L01.h>
#include <RF24.h>
#include <printf.h>


// NRF24L01 settings
RF24 radio(RF24_CE_PIN, RF24_CSN_PIN);


void setup();
void loop();
void startRadio();
void readRadio();
void setupLed();
void toggleLed(bool isOn);
```

```
void setup() {
  Serial.begin(SERIAL_PORT);
  printf_begin();

  setupLed();
  startRadio();
}


void loop() {
  readRadio();
}


void startRadio() {
  if (!radio.begin()) {
    printf("RF24 ini“ fail. Abort");
    w”ile (true) {
      delay(1);
    }
  }

  radio.setAutoAck(1);
  radio.setPayloadSize(RADIO_PAYLOAD_SIZE);

  uint8_t pipeAddress[6] = PRIVATE_ID;
  radio.openReadingPipe(0, pipeAddress);

  radio.setChannel(0x60);
  radio.setPALevel(RF24_PA_LOW);
  radio.setDataRate(RF24_1MBPS);

  radio.powerUp();
  radio.startListening();

  printf("RF24 ini“ success\n\n");
  rad”o.printDetails();
  printf("\n");
  “ad”o.printPrettyDetails();
}


void readRadio() {
  if (!radio.available()) {
    return;
  }

  uint8_t receivedData[RADIO_PAYLOAD_SIZE + 1];
  radio.read(&receivedData, RADIO_PAYLOAD_SIZE);

  Serial.print("Received“data: ");
  for“(byte i = 0; i < RADIO_PAYLOAD_SIZE; i++) {
    Serial.print(receivedData[i]);
    Serial.print(" ");
  }“ “Serial.println();
```

```
  bool isOn = receivedData[0] == 1;
  toggleLed(isOn);
}


void setupLed() {
  pinMode(LED_PIN, OUTPUT);
}


void toggleLed(bool isOn) {
  if (isOn)
    digitalWrite(LED_PIN, HIGH);
  else
    digitalWrite(LED_PIN, LOW);
}
```

ДОДАТОК Б

Код програми управління системою

Лістинг Б.1 – файл DeviceController.cs

```
using System;
using System.Linq;
using System.Net.Mime;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using ModularHouse.Libraries.InternalMessaging.CQRS.Abstractions;
using ModularHouse.Libraries.InternalMessaging.DomainEvents.Abstractions;
using ModularHouse.Server.DeviceControl.Api.Http.MappingExtensions;
using ModularHouse.Server.DeviceControl.Application.CQRS.Commands;
using ModularHouse.Server.DeviceControl.Application.CQRS.Queries;
using ModularHouse.Server.DeviceControl.Domain.DeviceAggregate.DomainEvents;
using ModularHouse.Shared.Models.Common.Responses;
using ModularHouse.Shared.Models.DeviceControl.Requests;
using ModularHouse.Shared.Models.DeviceControl.Responses;

namespace ModularHouse.Server.DeviceControl.Api.Http.Controllers;

[ApiController]
[Authorize]
[Produces(MediaTypeNames.Application.Json)]
[Route("api/devi«es")]
publi» class DeviceController : ControllerBase
{
   private readonly IMessageBus _messageBus;
   private readonly IDomainEventBus _domainEventBus;

   public DeviceController(IMessageBus messageBus, IDomainEventBus domainEventBus)
   {
      ArgumentNullException.ThrowIfNull(messageBus);
      ArgumentNullException.ThrowIfNull(domainEventBus);
```

```
        _messageBus = messageBus;
        _domainEventBus = domainEventBus;
    }

    /// <summary>
    /// Get All Devices.
    /// </summary>
    /// <returns>ActionResult with List of Devices.</returns>
    [HttpGet]
    [ProducesResponseType(typeof(ListedResponse<DeviceResponse>),
StatusCodes.Status200OK)]
    public async Task<IActionResult> GetAllAsync()
    {
        var query = new GetAllDevicesQuery();
        var queryResponse = await _messageBus.SendAsync(query);
        var devices = queryResponse.Devices.Select(deviceDto =>
deviceDto.ToResponse()).ToList();

        var response = new ListedResponse<DeviceResponse>(devices, queryResponse.TotalCount);
        return Ok(response);
    }

    /// <summary>
    /// Get Device by Id.
    /// </summary>
    /// <param name="deviceId»>Id of D»vice to get.</param>
    /// <returns>ActionResult with Device.</returns>
    [HttpGet("{deviceI«:guid}")]
    [»roducesResponseType(typeof(DeviceResponse), StatusCodes.Status200OK)]
    [ProducesResponseType(typeof(ErrorResponse), StatusCodes.Status404NotFound)]
    public async Task<IActionResult> GetByIdAsync([FromRoute] Guid deviceId)
    {
        var query = new GetDeviceByIdQuery(deviceId);
        var queryResponse = await _messageBus.SendAsync(query);

        var response = queryResponse.Device.ToResponse();
        return Ok(response);
    }
```

```
        /// <summary>
        /// Create Device.
        /// </summary>
        /// <param name="requestB»dy">Input d»ta of Device to create.</param>
        /// <returns>ActionResult with created Device.</returns>
        [HttpPost]
        [ProducesResponseType(typeof(DeviceResponse), StatusCodes.Status200OK)]
        [ProducesResponseType(typeof(ErrorResponse), StatusCodes.Status400BadRequest)]
        public async Task<IActionResult> CreateAsync([FromBody] DeviceRequest requestBody)
        {
            var deviceInput = requestBody.ToDto();
            var command = new CreateDeviceCommand(deviceInput);

            var deviceCreatedEventWaitTask = _domainEventBus.WaitAsync<DeviceCreatedEvent>();
            await _messageBus.SendAsync(command);
            var deviceCreatedEvent = await deviceCreatedEventWaitTask;

            var getDeviceByIdQuery = new GetDeviceByIdQuery(deviceCreatedEvent.DeviceId);
            var getDeviceByIdQueryResponse = await _messageBus.SendAsync(getDeviceByIdQuery);

            var response = getDeviceByIdQueryResponse.Device.ToResponse();
            return Ok(response);
        }

        /// <summary>
        /// Update Device.
        /// </summary>
        /// <param name="deviceId»>Id of D»vice to update.</param>
        /// <param name="requestB»dy">Input d»ta of Device to update.</param>
        /// <returns>ActionResult with updated Device.</returns>
        [HttpPut("{deviceI«:guid}")]
        [»roducesResponseType(typeof(DeviceResponse), StatusCodes.Status200OK)]
        [ProducesResponseType(typeof(ErrorResponse), StatusCodes.Status400BadRequest)]
        [ProducesResponseType(typeof(ErrorResponse), StatusCodes.Status404NotFound)]
        public async Task<IActionResult> UpdateAsync([FromRoute] Guid deviceId, [FromBody]
DeviceRequest requestBody)
        {
```

```
        var deviceInput = requestBody.ToDto();
        var command = new UpdateDeviceCommand(deviceId, deviceInput);

        var deviceUpdatedEventWaitTask = _domainEventBus.WaitAsync<DeviceUpdatedEvent>();
        await _messageBus.SendAsync(command);
        var deviceUpdatedEvent = await deviceUpdatedEventWaitTask;

        var getDeviceByIdQuery = new GetDeviceByIdQuery(deviceUpdatedEvent.DeviceId);
        var getDeviceByIdQueryResponse = await _messageBus.SendAsync(getDeviceByIdQuery);

        var response = getDeviceByIdQueryResponse.Device.ToResponse();
        return Ok(response);
    }

    /// <summary>
    /// Delete Device by Id.
    /// </summary>
    /// <param name="deviceId»>Id of D»vice to delete.</param>
    /// <returns>ActionResult.</returns>
    [HttpDelete("{deviceI«:guid}")]
    [»roducesResponseType(StatusCodes.Status200OK)]
    [ProducesResponseType(typeof(ErrorResponse), StatusCodes.Status404NotFound)]
    public async Task<IActionResult> DeleteByIdAsync([FromRoute] Guid deviceId)
    {
        var command = new DeleteDeviceByIdCommand(deviceId);
        var deviceDeletedEventWaitTask = _domainEventBus.WaitAsync<DeviceDeletedEvent>();

        await _messageBus.SendAsync(command);
        await deviceDeletedEventWaitTask;

        return Ok();
    }
}
```

Лістинг Б.2 – файл DeviceControlController.cs

```
using System;
using System.Net.Mime;
using System.Text.Json.Nodes;
using System.Threading.Tasks;
```

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using ModularHouse.Libraries.InternalMessaging.CQRS.Abstractions;
using ModularHouse.Libraries.InternalMessaging.DomainEvents.Abstractions;
using ModularHouse.Server.DeviceControl.Api.Http.MappingExtensions;
using ModularHouse.Server.DeviceControl.Application.CQRS.Commands;
using ModularHouse.Server.DeviceControl.Application.CQRS.Queries;
using ModularHouse.Server.DeviceControl.Domain.ModuleStateAggregate.DomainEvents;
using ModularHouse.Shared.Models.Common.Responses;
using ModularHouse.Shared.Models.DeviceControl.Requests;
using ModularHouse.Shared.Models.DeviceControl.Responses;

namespace ModularHouse.Server.DeviceControl.Api.Http.Controllers;

[ApiController]
[Authorize]
[Produces(MediaTypeNames.Application.Json)]
[Route("api/devi“es/{deviceId:guid}/control")]
publi” class DeviceControlController : ControllerBase
{
   private readonly IMessageBus _messageBus;
   private readonly IDomainEventBus _domainEventBus;

   public DeviceControlController(IMessageBus messageBus, IDomainEventBus
domainEventBus)
   {
      ArgumentNullException.ThrowIfNull(messageBus);
      ArgumentNullException.ThrowIfNull(domainEventBus);

      _messageBus = messageBus;
      _domainEventBus = domainEventBus;
   }

   /// <summary>
   /// Get State of Module Device.
   /// </summary>
   /// <param name="deviceId”>Id of M”dule Device.</param>
```

```
        /// <returns>ActionResult with State of Module Device.</returns>
        [HttpGet("module-s“ate")]
        [”roducesResponseType(typeof(BinaryModuleStateResponse), StatusCodes.Status200OK)]
        [ProducesResponseType(typeof(ErrorResponse), StatusCodes.Status400BadRequest)]
        [ProducesResponseType(typeof(ErrorResponse), StatusCodes.Status404NotFound)]
        public async Task<IActionResult> GetModuleStateAsync([FromRoute] Guid deviceId)
        {
            var getStateQuery = new GetModuleStateQuery(deviceId);
            var getStateQueryResponse = await _messageBus.SendAsync(getStateQuery);

            var response = getStateQueryResponse.State.ToResponse();
            return Ok(response);
        }

        /// <summary>
        /// Set State of Module Device.
        /// </summary>
        /// <param name="deviceId”>Id of M”dule Device.</param>
        /// <param name="requestB”dy">Input d”ta of State to set.</param>
        /// <returns>ActionResult with State of Module Device.</returns>
        [HttpPatch("module-s“ate")]
        [”roducesResponseType(typeof(BinaryModuleStateResponse), StatusCodes.Status200OK)]
        [ProducesResponseType(typeof(ErrorResponse), StatusCodes.Status400BadRequest)]
        [ProducesResponseType(typeof(ErrorResponse), StatusCodes.Status404NotFound)]
        public async Task<IActionResult> SetModuleStateAsync(
            [FromRoute] Guid deviceId,
            [FromBody] JsonObject requestBody)
        {
            var getControlTypeQuery = new GetControlTypeQuery(deviceId);
            var getControlTypeQueryResponse = await
_messageBus.SendAsync(getControlTypeQuery);

            var input = requestBody.ToInputDto(getControlTypeQueryResponse.ControlType);
            var setStateCommand = new SetModuleStateCommand(deviceId, input);

            var stateUpdatedEventWaitTask =
_domainEventBus.WaitAsync<ModuleStateUpdatedEvent>();
            await _messageBus.SendAsync(setStateCommand);
```

```
        await stateUpdatedEventWaitTask;

        var getStateQuery = new GetModuleStateQuery(deviceId);
        var getStateQueryResponse = await _messageBus.SendAsync(getStateQuery);

        var response = getStateQueryResponse.State.ToResponse();
        return Ok(response);
    }

    /// <summary>
    /// Set view of Router Device display.
    /// </summary>
    /// <param name="deviceId”>Id of R”uter Device.</param>
    /// <param name="input">I”put d”ta of controlled display.</param>
    /// <returns>ActionResult.</returns>
    [HttpPatch("router-d“splay")]
    public async Task<IActionResult> SetRouterDisplayViewAsync(
        [FromRoute] Guid deviceId,
        [FromBody] SetRouterDeviceViewRequest input)
    {
        var command = new SetRouterDisplayViewCommand(deviceId, input.Text.ToCharArray());
        await _messageBus.SendAsync(command);

        return Ok();
    }
}
```

ДОДАТОК В

Демонстраційний матеріал у вигляді презентації

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Кафедра КІТАР
КВАЛІФІКАЦІЙНА РОБОТА

На тему: Розробка система автоматизації – розумний будинок «Modular House»

Виконав:
ст. гр. АКТАКІТ-20-1
Юсупов В.Т.

Керівник:
ст. викл. каф. КІТАР
Теслюк С. І.

Мета кваліфікаційної роботи

На сьогоднішній день концепція розумного будинку стає все більш популярною. Автоматизація та інтелектуальне керування різними системами в оселі дозволяє підвищити рівень комфорту, безпеки та енергоефективності. Розвиток технологій Інтернету речей та доступність різноманітних пристроїв і рішень для автоматизації відкривають нові можливості для створення розумних будинків.

Мета роботи – покращення роботи системи розумного будинок за рахунок робробки есперимантального макету з декількома режимами роботи в залежності від потреб користувача в режимі реального часу

Об’єкт розробки – автоматизація процесу регулювання стану розумних модулів.

Предмет розробки – система розумного будинку на базі веб-сервісу.

Для досягнення поставленої мети необхідно виконати наступні завдання

– проаналізувати існуючі системи розумного будинку;

– проаналізувати методи управління розумним будинком;

– розробити схему макета;

– обрати середу розробки;

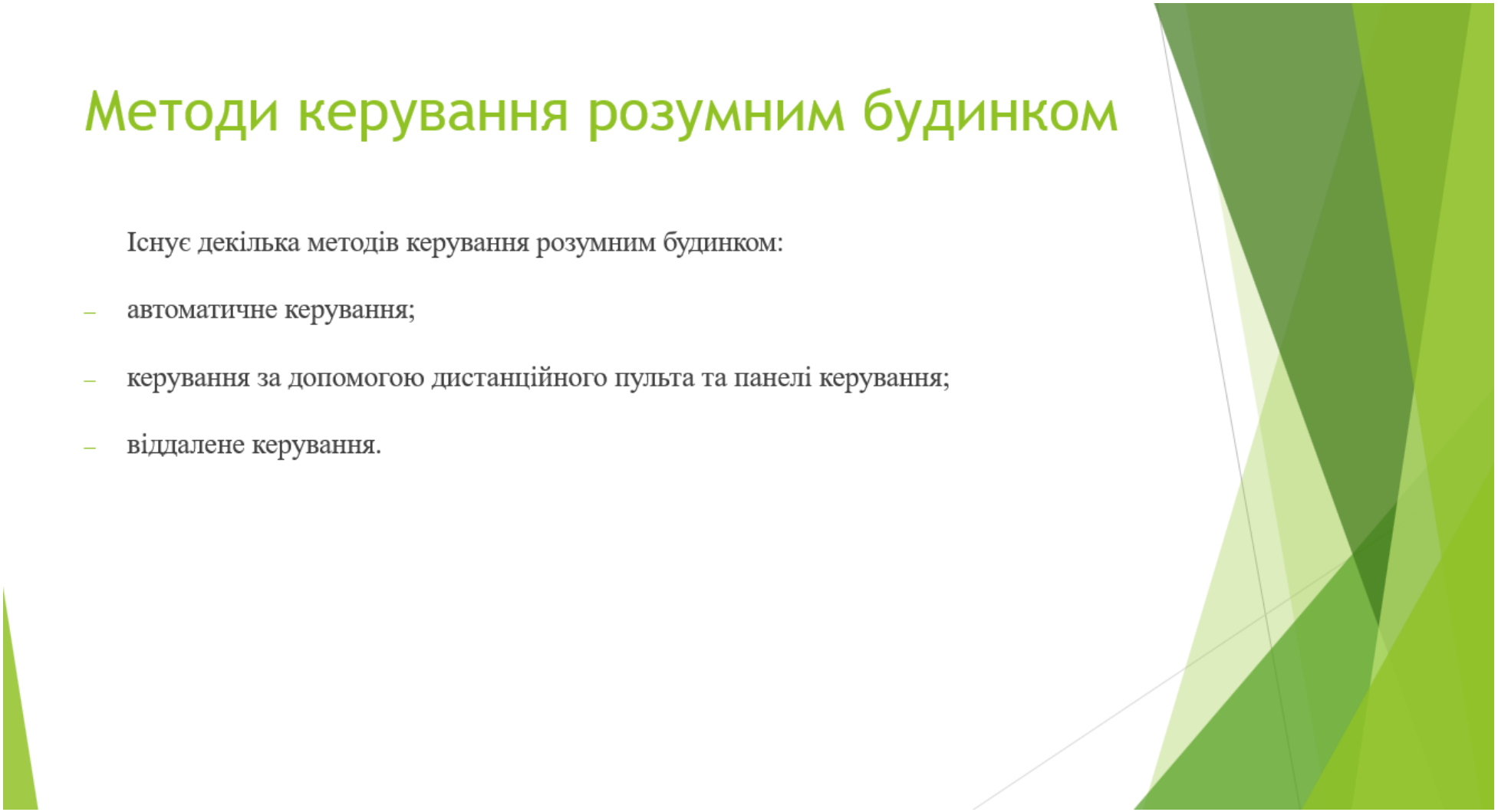
– написати програму для автоматизації управління;

– оформити кваліфікаційну роботу згідно ДСТУ 3008:2015 [1], а також з методичними вказівками з підготовки й оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології [2-4].

Методи керування розумним будинком

Існує декілька методів керування розумним будинком:

- автоматичне керування;
- керування за допомогою дистанційного пульта та панелі керування;
- віддалене керування.



Автоматичне керування розумним будинком – це система, яка дозволяє контролювати та керувати різними пристроями та функціями в будинку за допомогою автоматизованих процесів та програмованої логіки.

Найпоширенішими засобами керування будинком є дистанційне керування. Найчастіше використовується пульт дистанційного керування та панель управління.

На слайді наведено пульт дистанційного керування розумним будинком ELANHR2.

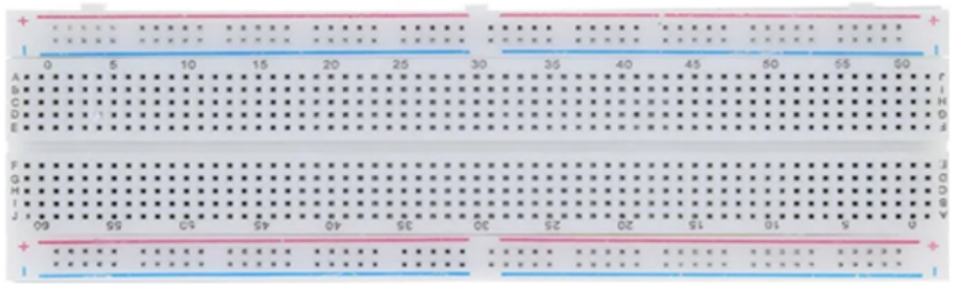
Система віддаленого доступу дозволяє керувати будинком з комп'ютера, планшета чи телефону. Для цього потрібно мати сервер системи розумного будинку. Цей сервер підключається до локальної мережі з одного боку та до інформаційної служби керування з іншого. Після цього сервер може перенаправляти команди, які надходять через локальну мережу від мешканця, до керуючих пристроїв будинку.

Основа лабораторного макету

На слайді наведено мікроконтроллер Arduino UNO, радіо модуль NRF24L01 та адаптер до радіо модуля NRF24L01

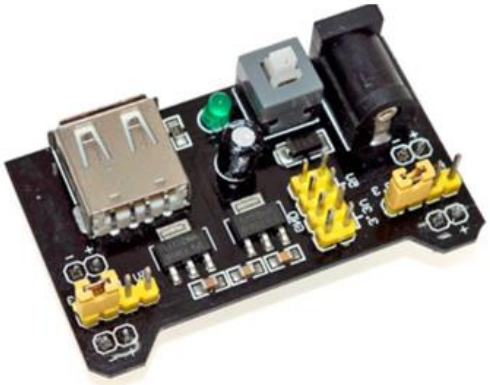
Основа лабораторного макету

На слайді наведено модуль HW-069 чотирьох-розрядний семи-сегментний індикатор
Та макетну плату на 830 точок типу MB-102

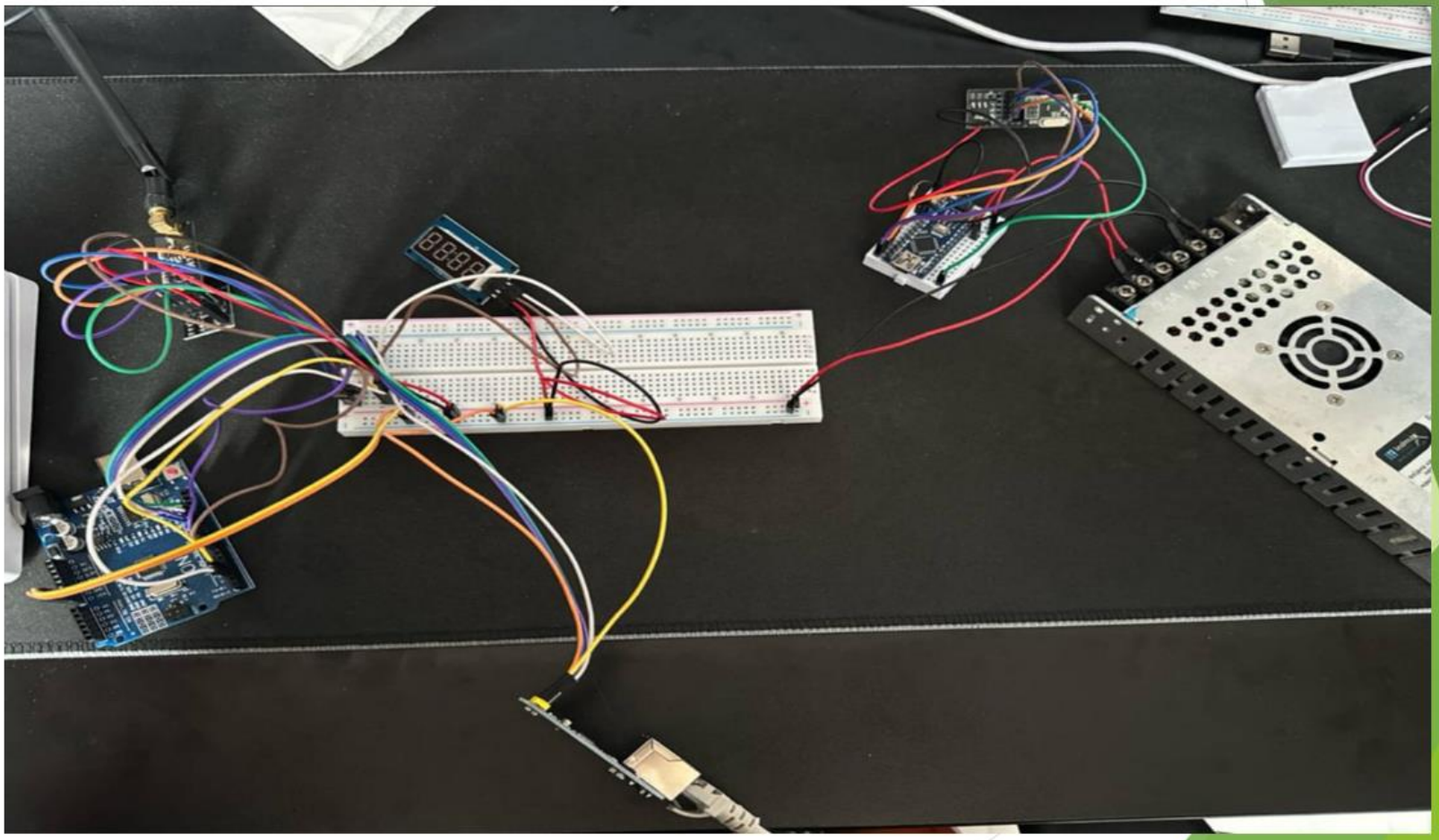


Основа лабораторного макету

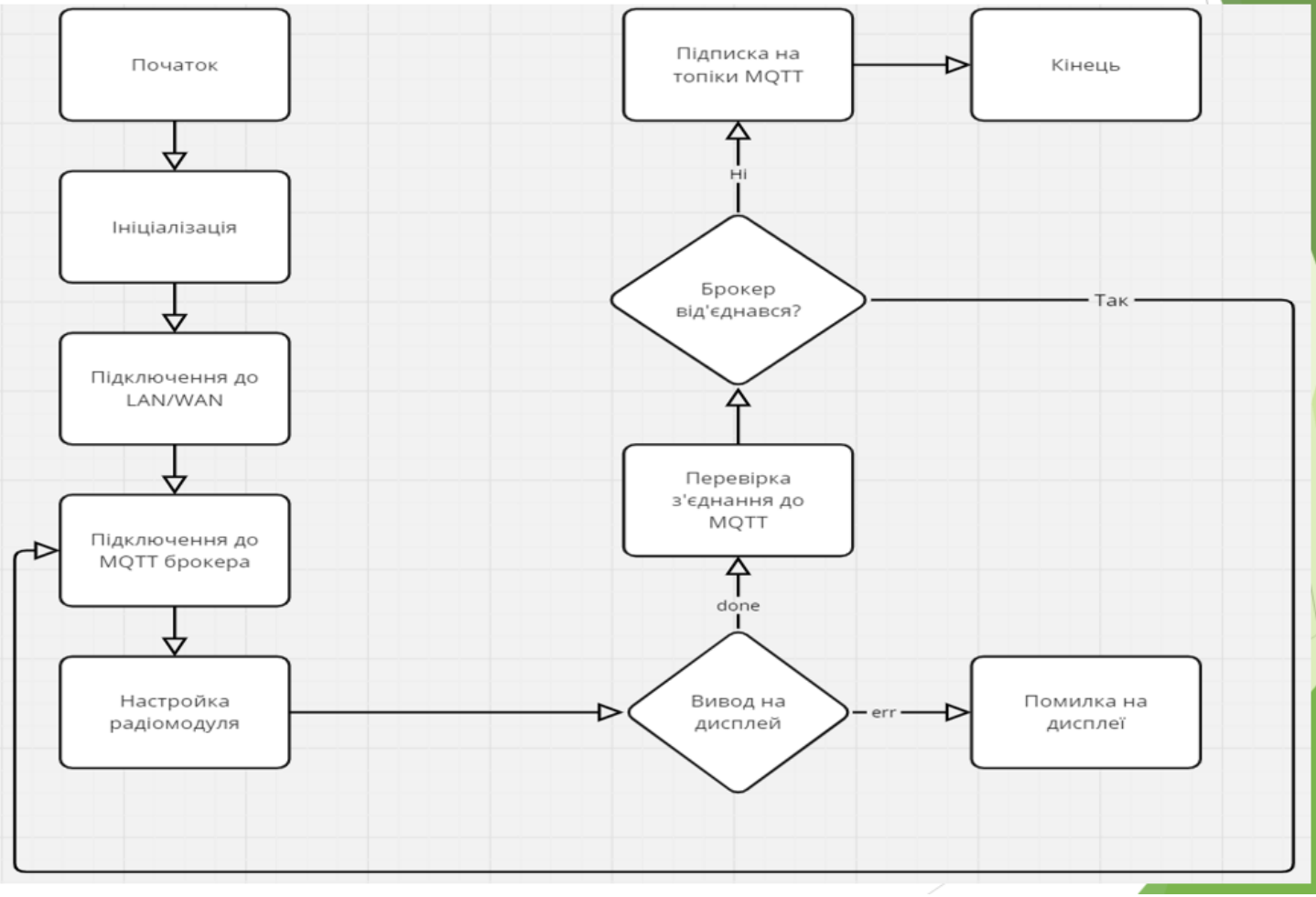
На слайді наведено модуль живлення для Arduino UNO та плату Arduino Nano



Зовнішній вигляд лабораторного макету



Блок-схема управління виконавчими елементами



Код програми

Усього було розроблено два модулі: модуль роутера та модуль лампочки.

Код роутера наведено у файлі router.ino. Код розподіляється на 2 основні метода: setup() та loop(). На слайді наведено код файлу router.ino, метод setup()

```
89  void setup() {
90    Serial.begin(SERIAL_PORT);
91    printf_begin();
92
93    setupDisplay();
94    showInitText();
95
96    bool setupEthernetResult = setupEthernet();
97    bool setupRadioResult = setupRadio();
98
99    if (!setupEthernetResult || !setupRadioResult) {
100     isError = true;
101     showErrorText();
102     return;
103   }
104
105   setupMqtt();
106   showDoneText();
107 }
```

Код програми

На слайді наведено код файлу router.ino, метод loop()

```
110  void loop() {
111    if (isError) {
112      return;
113    }
114
115    if (displayClearTime != 0 && millis() >= displayClearTime) {
116      display.clear();
117      displayClearTime = 0;
118    }
119
120    ensureConnectMQTT();
121    mqttClient.loop();
122  }
```

Демонстрація роботи системи

Для початку експерименту необхідно запустити апаратну частину. Після вмикання системи, на дисплей виводиться повідомленя стану системи. Наразі система у стані ініціалізації тому буде виведено init. На слайді наведено дисплей, який відображає поточний стан системи.

Якщо помилки не виникає, то на дисплей роутера виводиться done. На слайді наведено дисплей, який відображає поточний стан системи.

Демонстрація роботи системи

В якості експерименту проведемо вмикання та вимикання модуля лампочки. Для реалізації експерименту необхідно створити роутер та модуль лампочки. На слайді наведено результат створення модуля роутера та модуля лампочки

200

Response body

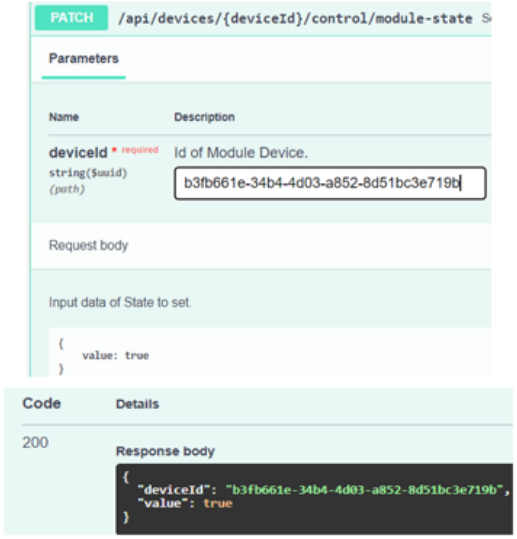

```
{
  "id": "f1de2bb4-b46c-48bf-aac2-30488068a585",
  "privateId": "rtr01",
  "deviceType": "router",
  "controlType": "defaultRouter",
  "additionDate": "2024-06-17T19:19:13.7445354Z"
}
```


Code	Details
200	Response body


```
{
  "id": "b3fb661e-34b4-4d03-a852-8d51bc3e719b",
  "privateId": "md101",
  "deviceType": "module",
  "controlType": "binary",
  "additionDate": "2024-06-17T19:21:44.8801049Z"
}
```

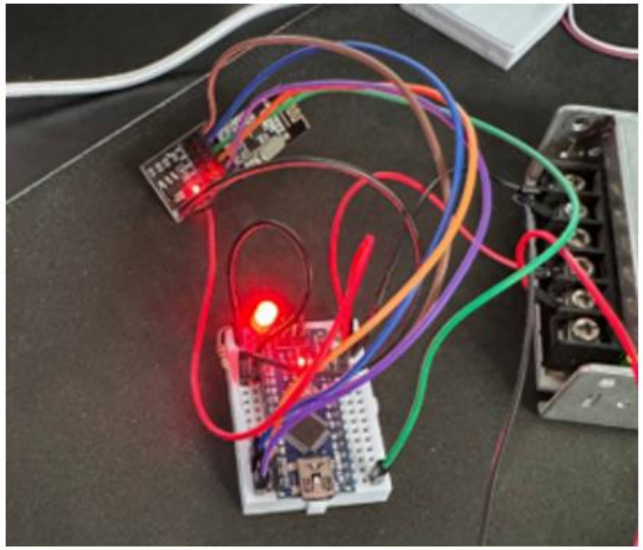
Демонстрація роботи системи

Після створення модуля роутера та модуля лампочки, проманіпулюємо станом модуля лампочки. Спочатку увімкнемо модуль лампочки. На слайді наведено запит та результат увімкнення модуля лампочки.



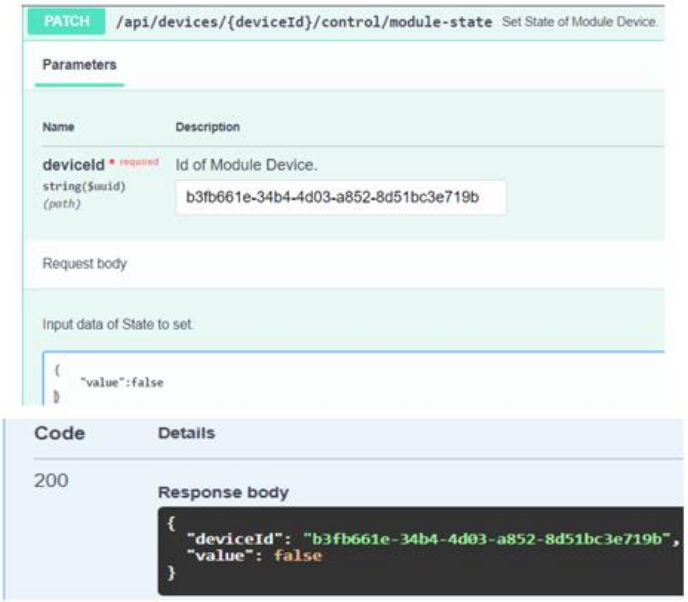
Демонстрація роботи системи

Світлодіод на модулі лампочки увімкнувся. На слайді наведено поточний стан модуля лампочки у активному стані



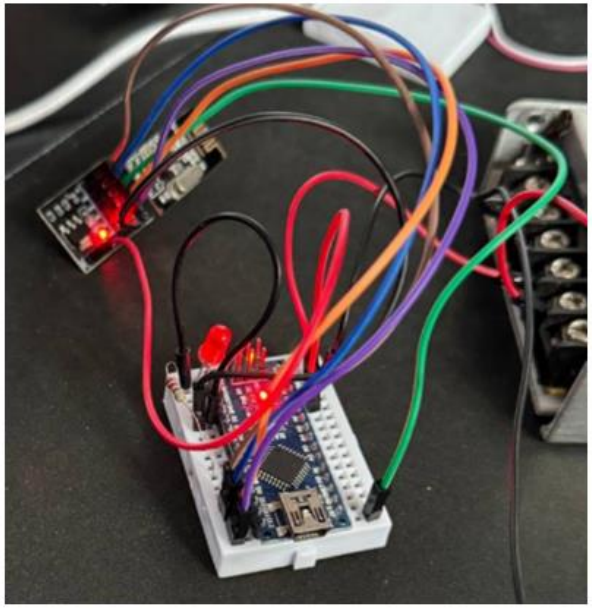
Демонстрація роботи системи

Тепер вимкнемо модуль лампочки. На слайді наведено запит та результат вимкнення модуля лампочки



Демонстрація роботи системи

Модуль лампочки вимкнувся. На слайді наведено поточний стан модуля лампочки у вимкненому стані.



ВИСНОВКИ

Метою кваліфікаційної роботи є розробка системи автоматизації розумного будинку, яка має декілька режимів роботи в залежності від потреб виробництва та керує станом під'єднаних розумних модулів в режимі реального часу.

Для досягнення поставленої мети було виконано наступні завдання:

– аналіз недоліків існуючих систем;
– аналіз методів керування розумним будинком;
– складання макету системи;
– розробка коду мікроконтролера Arduino UNO;
– розробка коду серверної частини;
– проведення експерименту.

ДОДАТОК Г

Відомість кваліфікаційної роботи бакалавра

№ документа	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК.565100.018 ПЗ	Пояснювальна записка	78 с., ф. А4
2		Додаткові матеріали	
		Код програми для Arduino	8 с., ф. А4
		UNO	
		Код програми для	7 с., ф. А4
		управління системою	
		Демонстраційний	19 с.,
		матеріал	ф. *.pptx
		у вигляді презентації	

					ГЮИК.565100.018 ВД			
Змін.	Арк.	Номер докум.	Підпис	Дата				
Розроб.		Юсупов В. Т.		13.06.2024	Розробка системи автоматизації – розумний будинок "Modular House"	Літера	Аркуш	Аркушів
Перевір.		Теслюк С. І.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Стародубцев М.Г.						
Затв.		Невлюдов І.Ш.						