

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій

(повне найменування факультету)

Комп'ютерні системи та мережі

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему **РОЗРОБКА СИСТЕМИ АВТОМАТИЧНОГО ДОСТУПУ ДО АУДИТОРІЇ
НА ОСНОВІ ARDUINO**

Виконав: студент 4 курсу, групи КНТ-527
спеціальності

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Комп'ютерна інженерія»

Кострикін М.М.

(прізвище та ініціали)

Керівник Зеленьова І.Я.

(прізвище та ініціали)

Рецензент Неласа Г.В.

(прізвище та ініціали)

м. Запоріжжя
2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти (освітній ступінь) бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри Кудерметов Р.К.

“ ” 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Кострикiна Максима Максимовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка системи автоматичного доступу до аудиторії в умовах пандемії на основі Arduino

керівник проєкту (роботи) Зеленьова Ірина Яковлівна, к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “17” березня 2021 року № 81

2. Строк подання студентом проєкту (роботи) 06 травня 2021 року

3. Вихідні дані до проєкту (роботи) наявні методи та алгоритми реалізації функціональних вузлів під платформу Arduino

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз технічного завдання та інструментів для реалізації проєкту;

2) Розробка архітектури проєкту, апаратних та програмних модулів;

3) Тестування пристрою.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

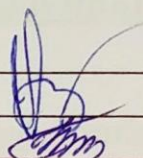
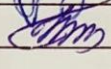
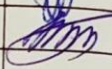
Структурна схема реалізації системи;

Принципова схема пристрою;

Алгоритм роботи системи;

Алгоритм роботи веб-додатку.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	приймає виконав завдання
1-3	Зеленьова І.Я., к. т. н., доцент		
Нормоконтроль	Щербак Н.В., ст. викл.		

7. Дата видачі завдання 01.03.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Прим.
1	Аналіз наявних методів реалізації, порівняння існуючої елементної бази, виділення недоліків та переваг	01.03.2021 р.	
2	Розробка архітектури проєкту	15.03.2021 р.	
3	Розробка програмних модулів	25.03.2021 р.	
4	Розробка веб-інтерфейсу проєкту	01.04.2021 р.	
5	Розробка структурної схеми системи	05.04.2021 р.	
6	Розробка функціональних вузлів	13.04.2021 р.	
7	Тестування пристрою	15.04.2021 р.	
8	Оформлення отриманих результатів у ПЗ	20.04.2021 р.	
9	Оформлення графічного матеріалу	25.04.2021 р.	
10	Оформлення допоміжного матеріалу	01.05.2021 р.	

Студент

(підпис)

М.М. Кострикін
(ініціали та прізвище)

Керівник проєкту (роботи)

(підпис)

І.Я. Зеленьова
(ініціали та прізвище)

РЕФЕРАТ

ПЗ: 90 с., 37 рис., 10 таблиць, 16 джерел, 4 додатки.

ARDUINO, NODEMCU, PROTEUS VSM, APACHE HTTP SERVER,
МІКРОКОНТРОЛЕР, ВЕБ-ДОДАТОК

Об'єкт дослідження – система контролю доступу до приміщень.

Предмет дослідження – засоби моделювання та проектування електронних схем.

Мета дослідження – розробка систем автоматичного доступу на базі платформи Arduino з підтримкою технології Інтернет.

Для досягнення поставленої мети було розроблено функціональну схему пристрою; розроблено клієнтський веб-додаток для керування системою та отримання вихідних даних з неї; здійснено моделювання системи у програмі для автоматичного проектування електронних схем; проведено тестування системи.

В результаті виконання роботи було розроблено систему автоматичного доступу на базі платформи Arduino, яка контролюється за допомогою клієнтського веб-додатка.

ЗМІСТ

Вступ.....	8
1 Аналіз інструментів розробки.....	9
1.1 Огляд керуючих контролерів.....	9
1.2 Система контролю доступу	13
1.3 Система контролю температури.....	16
1.4 Додаткові модулі системи	18
1.5 Програмні засоби.....	19
1.6 Постановка задачі.....	23
2 Розробка програмної частини	24
2.1 Структура системи	24
2.2 Налаштування середовищ розробки.....	25
2.3 Модуль зв'язку з сервером.....	27
2.4 Модуль аутентифікації користувачів.....	28
2.4 Модуль визначення температури та дезінфекції рук	32
2.5 Розробка веб-інтерфейсу для відстеження даних о користувачах.....	35
3 Розробка апаратної частини	60
3.1 Розробка функціонального вузла аутентифікації користувачів.....	61
3.2 Розробка функціонального вузла визначення температури	63
3.3 Розробка функціонального вузла дезінфекції рук	64
3.4 Розробка загальної схеми пристрою	65
3.5 Тестування схеми пристрою	67
Висновки	72
Перелік джерел посилання	73
Додаток А.....	75
Додаток Б.....	77
Додаток В	82
Додаток Г.....	84

Перелік графічних матеріалів:

Креслення 1 – Структурна схема реалізації системи

Креслення 2 – Принципова схема пристрою

Креслення 3 – Алгоритм роботи пристрою

Креслення 4 – Алгоритм роботи веб-додатку

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

OЗУ	—	оперативний запам'ятовуючий пристрій
ПЗУ	—	постійний запам'ятовуючий пристрій
COM	—	послідовний порт, Communication Port
CSS	—	каскадні таблиці стилів, Cascading Style Sheets
DTIM	—	повідомлення про індикацію трафіку доставки, Delivery Traffic Indication Message.
GPIO	—	інтерфейс введення/виведення загального призначення, General-Purpose Input/Output
HTML	—	мова гіпертекстової розмітки, HyperText Markup Language
HTTP	—	протокол передачі гіпертексту, HyperText Transfer Protocol
I ² C	—	інтерфейс Inter-Integrated Circuit
IP	—	міжмережевий протокол, Internet Protocol
MISO	—	вхід ведучого, вихід веденого, Master In Slave Out
MOSI	—	вихід ведучого, вхід веденого, Master Out Slave In
PHP	—	препроцесор гіпертексту, Hypertext Preprocessor
PSK	—	ключ Pre-Shared Key
RFID	—	радіочастотна ідентифікація, Radio Frequency Identification
SCK	—	послідовний тактовий сигнал, Serial Clock
SDA	—	послідовна лінія даних, Serial Data
SPI	—	послідовний периферійний інтерфейс, Serial Peripheral Interface
SSID	—	ідентифікатор набору послуг, Service Set Identifier
UART	—	універсальний асинхронний приймач-передавач Universal Asynchronous Receiver-Transmitter
UID	—	ідентифікатор користувача, User Identifier
URL	—	уніфікований локатор ресурсів, Uniform Resource Locator

ВСТУП

У зв'язку з погіршенням епідемічної ситуації у країні та світі, підвищується гостра потреба у нових мірах по протидії загрозі пандемії. В нинішніх умовах, проведення освітнього процесу у навчальних закладах частково або повністю унеможлиблюється через небезпеку, яка пов'язана з великим скупченням людей, тому важливо вживати заходи, які були б спрямовані на протидію розповсюдженню захворювання.

Задачею даної роботи є розробка системи, метою використання якої є надання можливості автоматизації процесу проходження аутентифікації, введення обліку відвідувань та надання можливості проведення профілактичних заходів для мінімізації негативного впливу від очного відвідування навчальних закладів. Попри це, система має надавати можливість дистанційного відстеження інформації про відвідуваність та статус здоров'я відвідувачів.

Перед системою також ставиться завдання надати зручний спосіб для отримання та обробки даних, для чого необхідно використовувати комплекс технологій, що надають користувачам легкий у використанні інтерфейс.

Найкращим рішенням для побудови подібних пристроїв є системи, що ґрунтуються на базі платформи Arduino, у якій в якості головного керуючого пристрою використовується мікроконтролер. Головною особливістю платформи є можливість застосування широкого переліку різноманітних сенсорів та датчиків, що надає потужний інструмент для створення різноманітних пристроїв.

В якості інтерфейсу, з яким буде контактувати користувач, необхідно розробити додаток, через який контролер матиме змогу контактувати з користувачем за допомогою мережі Інтернет. Тому, попри створення прототипу системи відстеження користувачів, задачею є налаштування серверного додатку, через який буде можливість керувати одразу групою подібних систем.

1 АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБКИ

Важливою задачею, яка стоїть перед початком виконання будь-якого проєкту є огляд та аналіз програмних та апаратних засобів, які будуть використовуватися при розробці. Цей етап має велике значення, оскільки заздалегідь визначені умови запобігають виникненню помилок в подальшій роботі та допомагають побудувати цілісну та надійну структуру пристрою. Тому в поточному розділі постає задача огляду, аналізу і вибору апаратно-програмного комплексу для виконання роботи.

Серед основних компонентів розроблюваної системи можна виділити наступні:

- керуючий контролер, задачею якого є управління підключеними до нього модулями;
- електричні пристрої, датчики та модулі, що підключені до контролеру;
- програмне забезпечення – це як набір команд та інструкцій, що має виконувати контролер, так і серверне забезпечення, задачею якого є керування та аналіз даних з цих контролерів.

Отже, метою цього розділу є визначення компонентів системи, які найкращим чином підходять для використання у встановлених умовах, визначення можливим нюансів і недоліків, а також сценаріїв використання. Тим самим необхідно створити базу для подальшої розробки.

1.1 Огляд керуючих контролерів

На сьогодні існує велике різноманіття пристроїв, призначенням яких є використання для проєктування, моделювання та тестування проєктів, що ґрунтуються на базі мікроконтролерів або мікропроцесорів. Такі пристрої отримали широке розповсюдження як серед професіоналів, які використовують їх

для побудови прототипів апаратів, так і серед нефахівців, які конструюють різноманітні саморобні пристрої. Тому важливо розглянути найважливіших представників подібних контролерів, проаналізувати особливості та зробити вибір керуючої апаратної платформи.

Arduino – це готова для використання програмно-апаратна платформа, призначення якої полягає у взаємодії з підключеними сенсорами та модулями, а також обробка сигналів з них.

Апаратно Arduino являє собою плату, основу на базі мікроконтролеру ATmega. До її складу входять усі елементи, що необхідні для роботи з мікроконтролером, а саме: цифрові та аналогові входи/виходи, кварцовий резонатор, USB-UART перетворювач, роз'єм для внутрішньосхемного програмування, кнопка скидання, роз'єм живлення, USB-порт, лінійний стабілізатор напруги [1].



Рисунок 1.1 – Опис елементів плати Arduino Uno Rev3 [2]

Серцем будь-якої аналогічної платформи є мікроконтролер, тому важливо зрозуміти, що він собою представляє. Отже, мікроконтролер – це спеціальна мікросхема, яка виконана на одному кристалі та призначенням якої є управління різними електронними пристроями. Мікроконтролер містить одне або кілька ядер процесора, пам'ять (ПЗУ та ОЗУ), аналого-цифровий перетворювач, таймери і програмовані периферійні пристрої введення / виводу [1].

Особливістю мікроконтролерів в порівнянні з мікропроцесорами є простота цих пристроїв. Мікроконтролери не володіють достатньою обчислювальною потужністю для виконання складних завдань, проте і робота з ними значно простіша ніж з мікропроцесорами. Тому для реалізації даного проєкту вибір припав саме на мікроконтролер, бо низька продуктивність повністю компенсується вартістю і простотою використання, а також універсальністю їх застосування.

Написання, налагодження і завантаження прошивок відбувається за допомогою середовищі розробки – Arduino IDE або у безлічі інших аналогічних спеціалізованих програмних продуктах. Безпосередньо програмування відбувається фактично на мові C/C++, яка доповнена бібліотекою AVR Libc, у складі якої є функції та об'єкти, необхідні для роботи з Arduino. Компілювання і збірка проєкту відбувається за допомогою компілятора AVR-GCC.

Через те, що Arduino є платформою з відкритим вихідним кодом, це дозволяє створювати іншим виробникам свої аналоги цієї плати, які будуть повністю сумісними з програмною та апаратною частиною оригінальної плати Arduino. Це дає можливість більш гнучко підбирати готові рішення під свої потреби та задачі, або навіть створити особисту плату під потреби конкретного проєкту. Тому, завдяки різноманітності плат на ринку, є можливість вибирати готові рішення з необхідним функціоналом, відсутнім на платі Arduino. Наприклад, важливим компонентом для виконання цього проєкту, відсутнім на оригінальній платі Arduino є Wi-Fi модуль, для використання якого на Arduino необхідно встановлювати додаткові компоненти, тим самим підвищуючи кінцеву вартість, складність та фізичні розміри головної керуючої плати. Саме тому, при виборі контролеру варто звернути увагу, в тому числі, на рішення від інших виробників, технічні характеристики яких можуть бути більш придатними для використання у конкретних умовах.

Отже, необхідно зробити вибір модифікації Arduino. Важливим критерієм вибору контролеру для цього проєкту є наявність інтерфейсу бездротової мережі. Тому далі необхідно розглянути найбільш придатні для цього плати, серед яких будуть Arduino Uno та Arduino Nano з окремим Wi-Fi модулем ESP8266, платою

Wemos D1 Mini та платою NodeMCU. Жодна з цих плат не має значної переваги перед іншою. Тому далі усі вони будуть розглянуті більш детально.

Перед тим як почати порівняння безпосередньо самих плат, необхідно розглянути мікроконтролери, що використовуються в них. Порівняємо мікроконтролери ESP8266 та ATmega328p. Головною особливістю першого є те, що він надає користувачу можливість використовувати технологію Wi-Fi, важливу в даному проєкті, як в ролі основного контролера системи, так і в ролі окремого модуля. Оскільки ATmega328p не має підтримки ніяких мережових технологій, використання плати розширення на базі ESP8266 у будь-якому разі є необхідним. Попри це, мікроконтролер ESP8266 має більшу енергоефективність. Контролер не володіє особистою пам'яттю, тому працює виключно із зовнішнім Flash накопичувачем, об'єм котрого може бути до 4 Мбайт. Але значним недоліком цього мікроконтролеру є те, що він має лише один аналоговий контакт, що обмежує перелік можливих ситуацій використання плати.

Для наочного порівняння самих плат у таблиці 1.1 будуть наведені найважливіші для цієї роботи характеристики контролерів.

Таблиця 1.1 – Порівняльна характеристика плат [3]

Плата	NodeMCU V3	WeMos D1 Mini	Arduino UNO R3	Arduino NANO 3
Мікроконтролер	ESP8266	ESP8266	ATmega328p	ATmega328p
Робоча напруга	3.3В	5В	5В	5В
Напруга живлення	7В – 12В	7В – 12В	7В – 12В	7В – 12В
Споживання струму	15 μ A – 400 mA	15 μ A – 400 mA	19 mA – 180 mA	45 mA – 80 mA
Цифрові контакти	16	11	6	6
Аналогові контакти	1	1	6	8
Flash накопичувач	4МБ	4МБ	32КБ	32КБ
SRAM	64КБ	64КБ	2КБ	2КБ
Wi-Fi	+	+	-	-

З таблиці 1.1 можна відзначити, що плати, які основані на базі одного й того ж мікроконтролеру не мають значних відмінностей і все зводиться до вибору між одним з двох мікроконтролерів.

Єдиною серйозною перевагою плат на базі ATmega328p є наявність більшого числа аналогових контактів, але це не важливо в рамках цього проєкту. Усі компоненти систему підключаються до цифрових контактів, що нівелює цей недолік. В свою чергу потрібно пам'ятати, що при використанні цього мікроконтролеру необхідно буде користуватися зовнішнім Wi-Fi модулем саме на базі ESP8266. Тому більш логічним є вибір в сторону плат, що базуються на цьому мікроконтролері.

Отже, залишається визначитися між платами NodeMCU V3 та WeMos D1 Mini. Хоча вони й дуже схожі, але кожна з них має свої особливості. WeMos D1 Mini має компактніші розміри, через що і кількість контактів менша, але робоча напруга становить 5 В, що робить її більш універсальною. Але вибір буде зроблений все ж в сторону NodeMCU через більшу кількість контактів.

1.2 Система контролю доступу

Коли стоїть задача контролю доступу до деякого приміщення та ідентифікація особи, тоді можна звернути увагу на пристрої безконтактного контролю. Взагалі, системи контролю доступу являються програмним та апаратним комплексом, головною задачею яких є обмеження доступу до деякої території. Усі такі пристрої за функціональним призначенням можна поділити на дві категорії – перегороджуючі пристрої та пристрої, які ідентифікують особу.

Задачею пристроїв, що перешкоджають прохід є фізично огородити доступ до приміщень, тобто не дати змогу особам без певних прав потрапити на захищену територію. Серед основних подібних пристроїв можна виділити різноманітні електрозамки, смартзамки, електрозаціпки, кодові клавіатури та турнікети.

Недоліком використання подібних пристроїв у цій роботі є те, що для їх встановлення довелося б вбудовувати їх у двері аудиторій, що значно б підвищило вартість та складність всієї системи в загалом.

Серед ідентифікаторів особи можна виділити безліч видів пристроїв, які відрізняються за принципом роботи.

Пристроєм розпізнавання в таких системах виступають електронні ключі, картки, брелоки, мітки, що несуть в пам'яті пристроїв інформацію про персональні права доступу володаря. Також в якості ідентифікатора можуть виступати біометричні дані людини чи паролі для кодових замків.

Наведемо декілька типів серед найпоширеніших ідентифікаторів: ідентифікатори з перфораційним кодуванням, з вбудованими пасивними радіoeлементами або магнітами, ідентифікатори зі штриховим кодуванням, ідентифікаційні карти з магнітним кодуванням, безконтактні ідентифікатори RFID, електронні ключі [4].

Для використання в цьому проєкті підходить кілька видів серед перерахованих пристроїв. Найважливішими критеріями, що впливають на вибір пристрою є безконтактність, надійність системи при невеликих витратах, висока відмовостійкість.

Серед найбільш придатних варіантів можна виділити радіочастотні та штрихові ідентифікатори. Недоліками інших систем є ряд факторів, які унеможливають їх використання у проєкті. Це як і необхідність фізичного контакту, так і дорожнеча обладнання, але головна причина – відсутність модулів, що реалізують ці технології на платформі Arduino.

Отже, розглянемо детальніше обидва види пристроїв контролю доступу. Сканери штрих-кодів являють собою пристрої, що мають у своїй конструкції цифрову камеру і модуль обробки отриманого зображення. Пристрій має змогу розпізнавати штрих-коди та QR-коди, обробляти та передавати отримані дані до контролера. Але із недоліків цього підходу є те, що пристрій має високу вартість, а також є не дуже зручним у використанні, бо має низьку швидкість роботи, високу вибагливість до зовнішніх умов, таких як освітленість, кут і положення

ідентифікатора в просторі [5]. Все це сильно знижує швидкість розпізнавання осіб, що неприпустимо в умовах пандемії.

Наступним варіантом є використання радіочастотних RFID зчитувачів. Принцип роботи цих пристроїв базується на використанні в якості каналу передачі даних радіосигналів. Кожна така мітка має свій особистий унікальний ідентифікатор та пам'ять, що відправляється на запит від зчитувача, який використовує електромагнітні хвилі певної частоти.

Серед переваг використання саме технології RFID, найважливішим є те, що вона є безконтактною, має високу швидкість роботи та надійність, оскільки RFID мітки складно підробити.

Поширеним зчитувачем, який часто використовують на Arduino являється модуль RC522, який наведено на рисунку 1.2.

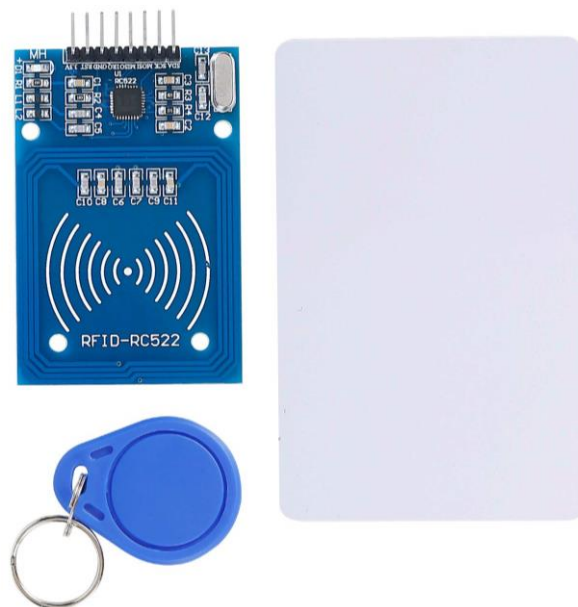


Рисунок 1.2 – RFID зчитувач RC522

Загальні характеристики RFID модуля RC522 наступні: напруга живлення 3,3 В; споживання струму 30 мА; частотна смуга 13,55-13,57 МГц; відстань зчитування до 25 мм [6]. В якості зчитувача може використовуватися як брелоки, так і картки – принцип дії у них однаковий.

Отже, кінцевим вибором буде RFID зчитувач із-за перерахованих переваг.

1.3 Система контролю температури

Важливою функцією розроблюваної системи є вимірювання температури студентів. Відомо, що саме підвищена температура є важливою ознакою того, що людина могла захворіти коронавірусом. Тому необхідно розглянути можливості реалізації вимірювання температури за допомогою Arduino.

На платформу Arduino існує безліч датчиків, спроможних вимірювати температуру. Виділити можна дві основні категорії датчиків, які працюють за різним принципом, та сфера використання яких відрізняється кардинально. Отже, існують датчики виміру температури повітря чи рідин, принцип дії яких базується, як правило, на зміні фізичних параметрів в залежності від температури. В залежності від типу пристрою робота може ґрунтуватися на термоелектричному ефекті, на зміні опору провідника чи на залежності зміни напруги на р-п переході. Також існують датчики виміру температури конкретного об'єкта, робота яких заснована на використанні інфрачервоного випромінювання [7].

Коли стоїть задача безконтактного виміру температури поверхні деякого об'єкта, тоді єдиним можливим варіантом залишається використовувати інфрачервоні датчики. Принцип роботи цих пристроїв заснований на законі Стефана-Больцмана, який говорить, що усі тіла з температурою вище абсолютного нуля випромінюють енергію в інфрачервоному діапазоні, а її інтенсивність прямо пропорційна температурі цього тіла [8]. Термоелемент датчику поглинає інфрачервону енергію, яка випромінюється об'єктом, після чого обчислювальний блок конвертує отримані значення енергії в температуру.

Існує чимало модулів, придатних для використання в цьому проєкті. Відрізняються вони за сферою застосування, ціною категорією, конструкцією та матеріалом виготовлення. Важливими критеріями, які впливають на вибір саме для цього проєкту є якість та точність виміру температури, поле зору датчику, вартість, а також, що не менш важливо, вихідний сигнал, представлений у цифровому виді.

Отже, необхідно розглянути модулі, які найкраще підходять під визначенні вимоги. Серед найкращих варіантів виділяються датчики A2TPMI [9] та MLX90614 [10]. Порівняння основних характеристик цих датчиків наведено у таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика датчиків температури

Датчик	A2TPMI	MLX90614
Робоча напруга	4.5-5.5В	3.6-5.5В
Вимірюваний діапазон температур	-20-250°C	-70-382.2 ° C
Точність вимірювання температури	-	0.5°C

Завдяки порівняльній таблиці та тому, що датчик MLX90614 має 17-розрядний аналого-цифровий перетворювач у порівнянні з 8-розрядним в A2TPMI, що позитивно впливає на точність вимірювання, можна говорити про те, що датчик MLX90614 є значно кращим вибором для використання.

На рисунку 1.3 наведено датчик MLX90614.

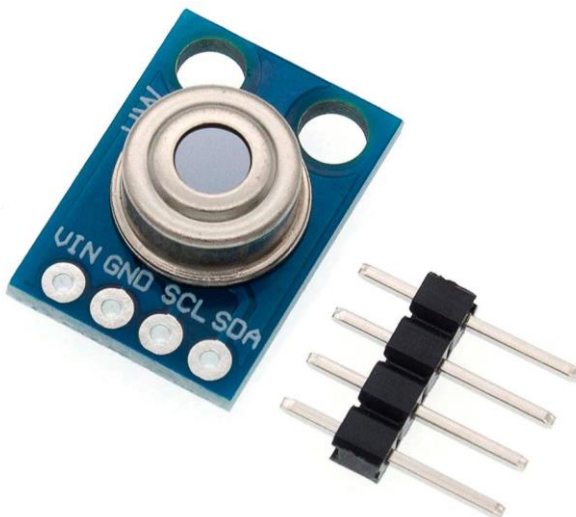


Рисунок 1.3 – Датчик MLX90614

1.4 Додаткові модулі системи

Далі будуть розглянуті елементи системи, які не є занадто комплексними та вибір яких не є настільки багатограним, щоб вимагати глибокого занурення в суть їх роботи. Тому в загальних рисах розглянемо принцип їх функціонування.

Водяна помпа є пристроєм, який обладнаний електродвигуном для перекачування рідин. Моделі розрізняють за зануренням у воду [11]. Також важливо враховувати, що помпа є достатньо потужним пристроєм, тому для її роботи необхідно використовувати достатньо потужний блок живлення.

Для управління помпою з мікроконтролера необхідно використовувати комутатор, такий як силовий ключ, транзистор або релейний модуль.

Для цієї роботи буде використовуватися помпа, яку не потрібно занурювати у воду, тому як це сприятливо впливає на надійність та довговічність пристрою. На рисунку 1.4 наведено помпу, яку можна використовувати у цій роботі.



Рисунок 1.4 – Водяна помпа

Загальні характеристики обраного насосу: робоча напруга 12 В постійного струму, мінімальна напруга 6 В, діаметр патрубків: внутрішній 6 мм, зовнішній 8 мм, продуктивність перекачування 1.5-2л/хв.

Інфрачервоний датчик перешкод застосовується у випадках, коли необхідно отримати інформацію про наявність або відсутність деякого об'єкта на заданій від нього відстані. У конструкції датчику міститься джерело інфрачервоного випромінювання та фотоприймач. Принцип дії полягає у тому, що датчик посилає промінь, який відбивається від об'єкта і фіксується фотоприймачем.

У цій роботі буде використовуватися датчик FC-51, оскільки він достатньо надійний і має високу чутливість роботи. Датчик показано на рисунку 1.5.

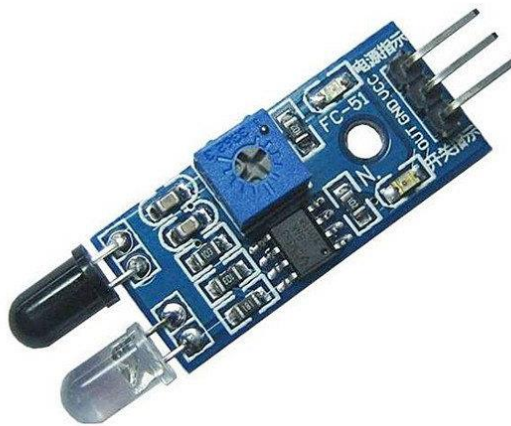


Рисунок 1.5 – ІЧ-датчик перешкод FC-51

1.5 Програмні засоби

Інструмент, який найчастіше використовують для написання «скетчів» під платформу Arduino, є інтегральне середовище розробки Arduino IDE. Представляє воно з себе комплекс програмних модулів, необхідних для комфортної взаємодії, налаштування та відладки мікроконтролерів.

Інтерфейс програми складається з текстового редактора з функціями форматування та підсвічування синтаксису, панелі інструментів, вікна консольного виводу.

На рисунку 1.6 зображено вікно програми Arduino IDE.

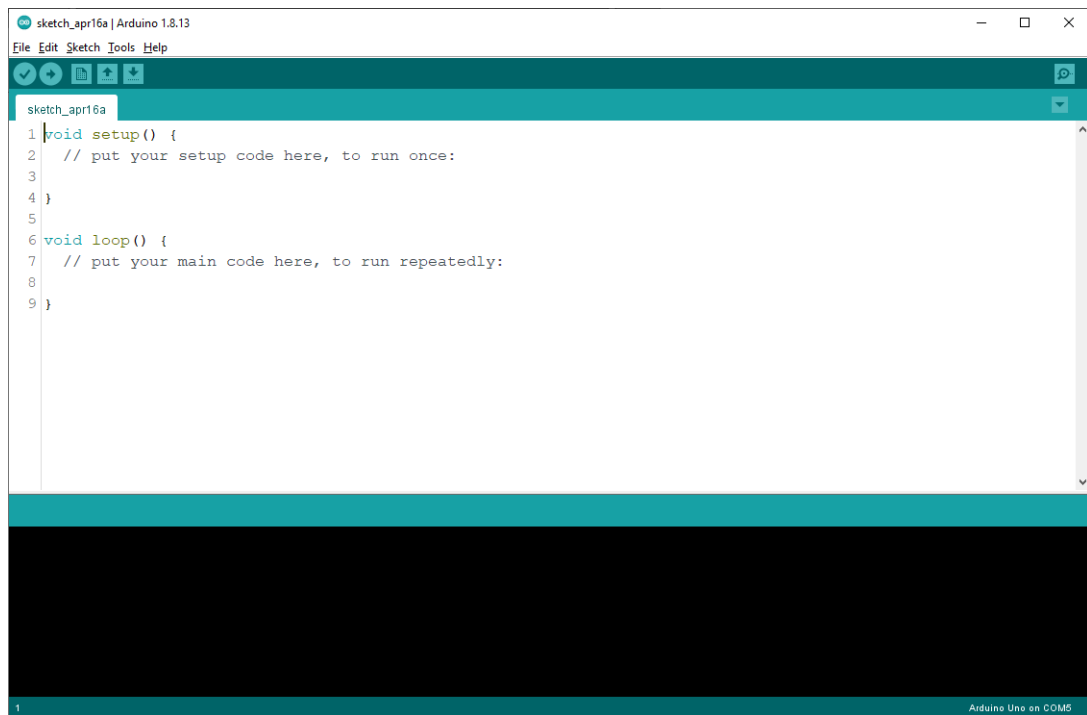


Рисунок 1.6 - Головне вікно програми Arduino IDE

Завдяки вбудованим функціям налаштування можна без зайвих проблем встановити усі необхідні параметри платформи, такі як модель плати, контролера, програматора, вибрати COM-порт та встановити необхідні бібліотеки для керування модулями та завантажити приклади вже готових програм для тестування окремих компонентів системи.

Arduino IDE є зручним інструментом для роботи з Arduino-сумісними платами, який надає увесь необхідний функціонал для комфортної роботи з платою.

Для задач, пов'язаних з моделюванням виконуваної системи зручно використовувати програми для автоматизованого проєктування електронних схем. Одним з найпотужніших подібних інструментів є Proteus VSM. Ця програма обладнана усіма необхідними функціями для зручної та швидкої розробки систем, в тому числі створення на базі Arduino.

Програма дозволяє як розробляти принципіальні схеми, так і моделювати їх, тестувати та відлагоджувати, при цьому отримуючи достовірні результати роботи системи.

Також, серед важливих функцій є можливість розробляти свої моделі пристроїв або додавати їх за допомогою користувацьких бібліотек та використовувати фізичні пристрої, які підключені через USB та пов'язані зі схемою за допомогою віртуального COM-порта.

На рисунку 1.7 зображено вікно програми Arduino IDE.

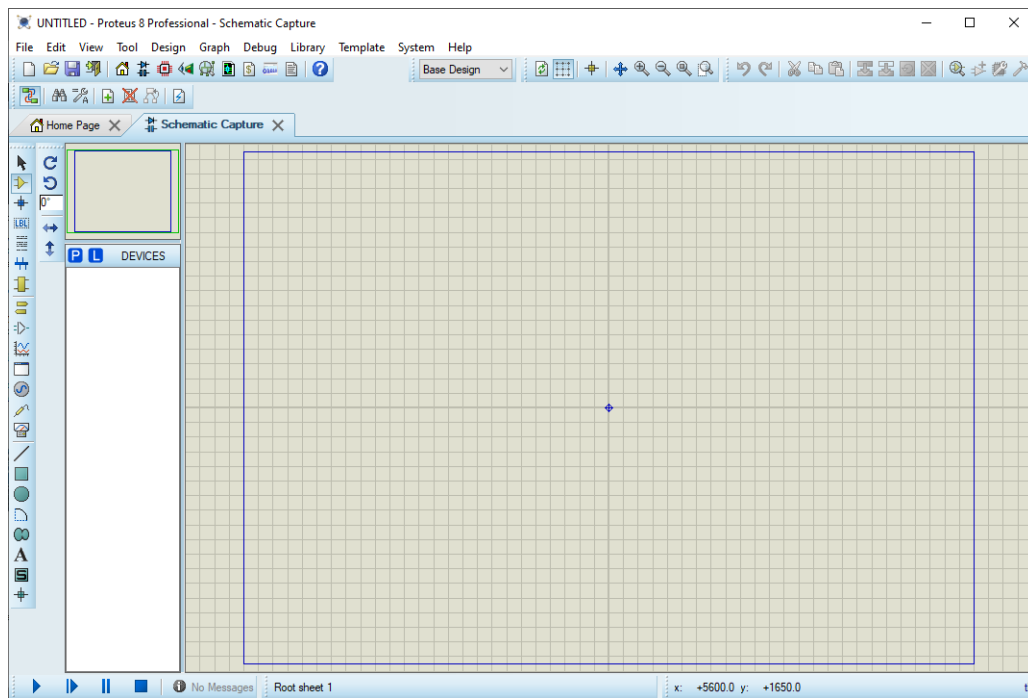


Рисунок 1.7 - Вікно програми Proteus VSM

Інтерфейс складається з редактора електронних схем, панелі інструментів та приладів, кнопок управління процесом моделювання та всього проєкту в цілому.

Але серед недоліків є те, що програма має значні вимоги до апаратного забезпечення системи, на якій виконується моделювання схеми.

Головною особливістю, якою володіє мікроконтролер ESP8266 є можливість використовувати веб-технології. Це дає зручний інструментарій для виконання задач, пов'язаних з керуванням, збереженням та обробкою даних з контролера. Тому для цих задач у проєкті буде використовуватися веб-сервер для встановлення контролю над системою.

Apache HTTP Server є найпоширенішим веб-сервером серед усіх існуючих на сьогоднішній день варіантів. Значна популярність робить цю платформу універсальною та

зручною для використання у різноманітних ситуаціях. Сервер має детальну документацію, є дуже гнучким у налаштуванні, що дозволяє використовувати різні модулі та виконувати програми без необхідності встановлення додаткового зовнішнього програмного забезпечення [12].

Але окрім самого сервера для роботи знадобляться й інші модулі. Для швидкого та зручного налаштування та розробки існує програмна платформа Denwer. Хоча це програмне забезпечення і не є надійною середою для створення повноцінних робочих серверів, Denwer надає зручний інструментарій для створення та налаштування серверів у локальній мережі. Серед найважливіших компонентів, які включено до складу цієї платформи є:

- Apache HTTP Server – безпосередньо основа всієї платформи;
- PHP – скриптова мова програмування, яка широко використовується для розробки веб-додатків;
- MySQL – програмне забезпечення для створення та керування реляційними базами даних;
- Система керування віртуальними хостами - інструмент, що дозволяє додаванням директорії в каталозі «./home» створити новий хост;
- phpMyAdmin – зручний інструмент, який застосовується для керування MySQL через веб-інтерфейс [13].

Таким чином використання Denwer значно спрощує процес установки та налаштування веб-сервера, надаючи змогу швидко розпочати використовувати усі необхідні інструменти, встановив лише одну програму.

Серед основних компонентів, які будуть активно використовуватись при роботі з сервером необхідно окремо звернути увагу на PHP та MySQL. Це найважливіші компоненти для розробки, бо на основі цих технологій будується уся логіка роботи та взаємозв'язок різних рівнів системи.

PHP – це скриптова, інтерпретована мова програмування, головною задачею якої є створення скриптів для виконання на стороні сервера та створення динамічного змісту HTML-сторінок [14]. Синтаксично PHP бере початок з таких мов як C, Java і Perl, що робить користування ним зручним та зрозумілим. Отже,

PHP являє собою безперечного лідера у сфері веб-розробки, що надає розробнику широкий спектр можливостей для реалізації необхідних функцій.

MySQL – це система, що надає можливості для створення та керування реляційними базами даних. Синтаксично ця система використовує мову SQL як основу для створення запитів, хоча сама вона написана на мові C та C++ .

Серед функцій, які має виконувати MySQL можна виділити наступне:

- запит конкретних даних з сервера;
- додавання, редагування та видалення даних;
- контроль доступу та забезпечення захисту даних.

Серед переваг саме цієї системи можна виділити гнучкість і простоту використання, високу швидкість роботи та високу безпечність.

1.6 Постановка задачі

В ході роботи необхідно вирішити наступні задачі:

- провести вибір програмного забезпечення для реалізації системи;
- провести вибір апаратного забезпечення для моделювання системи;
- реалізувати програмну частину системи та розробити веб-додаток;
- реалізувати схему пристрою у середовищі моделювання;
- провести тестування пристрою.

Головна задача, яка стоїть перед цією системою є підвищення безпеки та поліпшення епідемічної ситуації серед співробітників та студентів університету. Також, задачею є оптимізація та прискорення процесу авторизації для виділення більшого часу для проведення навчальної діяльності.

Таким чином, розроблювальна система повинна покращити життя усіх, залучених в навчальному процесі, людей.

2 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ

Першим етапом розробки пристрою є написання та тестування програмної частини пристрою, визначення задач, які повинна виконувати система, а також пошук оптимальних способів вирішення поставлених цілей.

У цьому розділі буде розглянуто процес проєктування та розробки програмної частини проєкту на основі визначених у попередньому розділі компонентів та модулів, з використанням розглянутих програмних засобів.

2.1 Структура системи

Розроблювана система об'єднує у собі як апаратні, так і програмні рішення для реалізації заданих функцій. Поділити її можна на три основні частини, а саме: безпосередньо контролер, що забезпечує роботу та взаємодію окремих модулів, збір та обробку даних; сервер, задачею якого є віддалене управління контролерами усіх копій систем, отримання даних о користувачах та представлення їх у зручній формі необхідним особам; компоненти та модулі, які забезпечують функціонування системи.

На рисунку 2.1 зображено структуру розроблюваної системи.

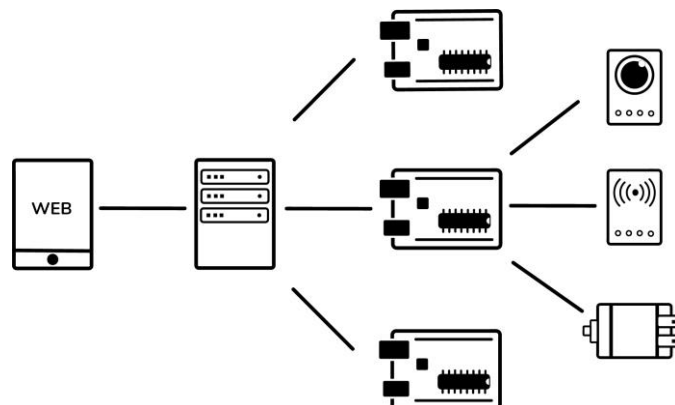


Рисунок 2.1 – Структура системи

Сутність функціонування системи полягає у тому, що кожний окремий контролер, який підключено до єдиного сервера, що має змогу керувати ними та отримувати інформацію. Певні користувачі володіють правом підключатися за допомогою будь-якого пристрою з підтримкою Інтернет-технології до сервера за допомогою веб-інтерфейсу та отримувати необхідні дані. Студент безконтактним чином взаємодіє з системою, ідентифікує себе та проводить протиепідемічні заходи, після чого має змогу потрапити до аудиторії, а отримані дані відправляються на сервер.

Алгоритм роботи системи:

- 1) користувач проходить контроль за допомогою RFID зчитувача;
- 2) контролер відправляє GET запит для порівняння ідентифікатора користувача з базою допущених до аудиторії користувачів;
- 3) при успіху RGB світлодіод загоряється зеленим та користувач підносить руки до термометра. При невдачі світлодіод загоряється червоним;
- 4) при піднесенні рук спрацьовує інфрачервоний датчик перешкод, який вмикає термометр, після нього помпу з антисептиком. Якщо користувач прибрав руки, тоді ІЧ-датчик вимикається, тим самим завершує роботу пов'язаних з ним пристроїв;
- 5) UID та температура користувача відправляється на сервер, де оброблюється та виводиться у вигляді таблиці.

2.2 Налаштування середовищ розробки

Перш ніж почати розробку системи необхідно налаштувати для роботи середовища, у яких ця розробка буде проходити – Arduino IDE та Proteus VSM.

Для комфортного використання Arduino IDE, до неї необхідно приєднати усі потрібні бібліотеки, встановити драйвер для CH340G та додати до менеджера плат сімейство плат на базі контролера ESP8266.

Почнемо саме з встановлення драйверу для мікросхеми CH340G. Ця мікросхема використовується на різних версіях плат Arduino в якості перетворювача інтерфейсу USB в UART, тобто він необхідний для підключення плат до комп'ютера за допомогою USB та створення віртуального COM-порту. Сама установка є простою, необхідно лише завантажити драйвер та натиснути кнопку «Install».

Далі необхідно додати нові плати. Для цього слід натиснути: File => Preferences та у полі «Additional Boards Manager URLs» вписати посилання на додаткові плати:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

Після цього з'явиться можливість вибрати необхідну плату, для чого слід перейти: Tools => Board => ESP8266 Boards та обрати плату NodeMCU 1.0 (ESP-12E Module). На цьому налаштування плати завершується.

Щоб додати бібліотеки для середовища розробки можна скористуватися одним з двох способів. Перший – скористуватися інтерфейсом програми та перейти: Sketch => Include Library => Manage Libraries, де в полі для введення тексту вписати назву необхідної бібліотеки. Другий спосіб – завантажити файли окремо та розмістити їх у директорії «.\Documents\Arduino\libraries».

Для виконання цього проекту знадобляться наступні бібліотеки Arduino IDE:

- SPI – бібліотека для послідовного периферійного інтерфейсу, яка використовується для обміну даними з пристроями, які підключені до плати. У цій роботі це RFID-зчитувач;
- MFRC522 – бібліотека для роботи RFID-зчитувача RC522, за допомогою якої відбувається зчитування та запис даних з карт;
- MLX90614 - бібліотека, яка використовується для взаємодії з інфрачервоним датчиком температури MLX90614.;
- ESP8266WiFi – бібліотека, яка необхідна для використання Wi-Fi технології на базі контролера ESP8266;
- ESP8266HTTPClient – бібліотека для використання протоколу передачі даних HTTP.

Наступним етапом налаштування є додавання необхідних бібліотек до середовища Proteus VSM. Використання цього програмного пакета ускладнюється тим, що він не є профільним програмним забезпеченням для моделювання контролерів Aduino, тому не кожний необхідний компонент можливо знайти. Тому в ролі моделей пристроїв будуть застосовуватися еквівалентні аналоги та реальні модулі, які поєднані з моделлю через віртуальний COM-порт.

Додавання бібліотек відбувається за допомогою розміщення файлів у директорії «.\Proteus 8 Professional\DATA\LIBRARY».

Таким чином будемо використовувати наступні бібліотеки:

- Arduino - бібліотека додає модель Arduino Uno, яка має змогу виконувати звичайні скетчі, призначенні для реальної плати;
- InfraredSensorsTEP - бібліотека інфрачервоний датчик перешкод FC-51.

2.3 Модуль зв'язку з сервером

Робота розроблюваної системи міцно пов'язана з використанням бездротової мережі Wi-Fi. Для повноцінного функціонування модулів перевірки достовірності та прав доступу особи, а також для відправлення результатів вимірювань показників температури необхіден доступ до мережі. Тому, перш ніж почати реалізовувати інші модулі системи, необхідно налагодити її функціонування.

При роботі з платами на базі мікроконтролера ESP8266 використовуються особисті бібліотеки для реалізації функцій, пов'язаних з Інтернетом. Використовуватися у цій роботі будуть дві наступні бібліотеки:

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
```

Наступним кроком є визначення параметрів, які використовуються для підключення до WiFi-мережі. Тому необхідно вказати SSID та PSK наступним чином:

```
#define STASSID                "your-ssid"
```

```
#define STAPSK                "your-password"
```

Останнім кроком налаштування мережеских функцій є визначення підключення до Wi-Fi. Виконується це у два кроки, де перший – це безпосередньо ініціалізація об'єкта встановленими раніше параметрами, а другий – очікування підключення до мережі. Реалізація має наступний вигляд:

```
WiFi.begin(STASSID, STAPSK);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
```

2.4 Модуль аутентифікації користувачів

Першим етапом, який необхідно пройти користувачу є аутентифікація. Спочатку алгоритм реалізації виконує перевірку прав піднесеної мітки, та в залежності від результату відбудеться одна з двох реакцій – це, по-перше, дозвіл на проходження наступного етапу взаємодії з системою, або відмова в доступі.

Підключення бібліотек для роботи з SPI та RFID-зчитувачем відбувається наступним чином:

```
#include <SPI.h>
#include <MFRC522.h>
```

Далі необхідно визначити контакти SS та RST. Перший необхіден для вибору з яким підлеглим пристроєм контролеру необхідно встановити зв'язок, а другий – це вивід для скидання та відключення живлення пристрою. Встановлюються значення контактів наступним чином:

```
#define SS_PIN                D4
#define RST_PIN                D3
```

Тим самим було визначено, що контакт SDA відповідає контакту D4 керуючої плати, а RST контакту D2.

Після налаштування контактів для роботи зчитувача, їх слід використовувати в якості параметрів, які використовуються для створення екземпляра класу MFRC522:

```
MFRC522 mfrc522(SS_PIN, RST_PIN);
```

Наступним кроком є визначення шини SPI та створення об'єкту класу. Зробити це слід в функції `setup()` і на цьому кроці завершується налаштування зчитувача. Визначення має наступний вигляд:

```
SPI.begin();  
mfrc522.PCD_Init();
```

Безпосередньо виконання алгоритму відбувається в головній циклічній функції `loop()`. Першим кроком викликається функція отримання ідентифікатора користувача з радіочастотної мітки. Код функції наведено у лістингу 2.1.

Лістинг 2.1 – Функція отримання ідентифікатора мітки користувача

```
unsigned int getUID() {  
    if (mfrc522.PICC_IsNewCardPresent()) {  
  
        if (mfrc522.PICC_ReadCardSerial()) {  
            mfrc522.PICC_HaltA();  
            unsigned int UID;  
            UID = mfrc522.uid.uidByte[0] << 24;  
            UID += mfrc522.uid.uidByte[1] << 16;  
            UID += mfrc522.uid.uidByte[2] << 8;  
            UID += mfrc522.uid.uidByte[3];  
            return UID;  
        }  
    }  
}
```

У тілі функції спочатку відбувається перевірка наявності нової картки та чи були прочитані які-небудь данні з неї. Далі викликається функція припинення роботи картки, після чого починається зчитування отриманих байтів ідентифікатора у форматі `unsigned int`.

Наступним етапом є порівняння отриманого ідентифікатора з базою даних через протокол HTTP. При успішному завершенні користувач матиме змогу надалі

взаємодіяти з системою, в іншому випадку робота програми завершиться. У лістингу 2.2 наведено код функції `checkUser()`.

Лістинг 2.2 – Функція перевірки прав доступу користувача

```
boolean checkUser(unsigned int UID) {
    if ((WiFi.status() == WL_CONNECTED)) {
        WiFiClient client;
        HTTPClient http;

        Serial.print("[HTTP] begin...\n");
        String url = "http://" SERVER_IP "/arduino" CHECK_USER_PAGE
        "?UID=" + String(UID) + "&aud=" AUD_NUM;
        http.begin(client, url);

        int httpCode = http.GET();
        if (httpCode == HTTP_CODE_OK) {
            const String& payload = http.getString();
            if (payload == "FOUND")
                return true;
            else
                return false;
            http.end();
        }
    }
}
```

Перед початком виконання запиту у функції виконується перевірка підключення контролеру до Wi-Fi мережі. Далі формується GET-запит, до якого додається отриманий раніше ідентифікатор користувача та номер аудиторії, у якій встановлена ця система. Потім виконується запит та отримується відповідь від сервера. Якщо відповідь відповідає слову «FOUND», тоді користувача успішно знайдено, про що і повідомляється головна циклічна функція `loop()`.

Відправлений запит у лістингу 2.2 оброблюється на стороні серверу за допомогою PHP-сторінки. Ця сторінка не має графічного інтерфейсу, тому як існує лише заради обробки вхідних даних з контролерів, та виводить лише результат своєї роботи. Код сторінки представлений у лістингу 2.3.

Лістинг 2.3 – Перевірка користувача з боку серверу

```
<?php
$UID = $_GET['UID'];
$aud = $_GET['aud'];

$link = mysqli_connect("localhost", "maxim", "1234", "arduino");

$sql = 'SELECT * FROM `aud` WHERE `UID` = "'. $UID. '" AND `aud` =
"'. $aud. '"';
$result = mysqli_query($link, $sql);
if (mysqli_num_rows($result) > 0) {
    print("FOUND");
} else {
    print("NOT FOUND");
}
?>
```

На початку роботи виконується отримання параметрів запиту. Відбувається це при обробці масиву змінних `$_GET`, які були передані скрипту через параметри URL. Далі відбувається підключення до бази даних MySQL, для чого слід вказати IP адресу серверу, ім'я та пароль користувача, а також назву необхідної бази даних. Потім формулюється запит до бази, після чого він виконується з вище зазначеними параметрами. Далі результат роботи поміщається до масиву і відбувається перевірка наявності яких-небудь даних у цьому масиві. Результат виводиться на HTML-сторінку, після чого зчитується методом `setString()` у функції `checkUser`, яка зазначена у лістингу 2.2.

Наведемо структуру таблиць баз даних, важливих для подальшої роботи. Структуру таблиці `user` подано у таблиці 2.1, у якій зберігаються основні дані про користувачів.

Таблиця 2.1 – Структура таблиці `user`

Поле	Тип	Значення за замовчуванням	Ключ
UID	bigint(20)	NOT NULL	UNIQUE
name	varchar(32)	NOT NULL	-
group	varchar(10)	NOT NULL	-

У таблиці aud, що наведено у таблиці 2.2, зберігається інформація про права доступу користувачів до аудиторій.

Таблиця 2.2 – Структура таблиці aud

Поле	Тип	Значення за замовчуванням	Ключ
UID	bigint(20)	NOT NULL	-
aud	int(3)	NOT NULL	-

2.4 Модуль визначення температури та дезинфекції рук

Після перевірки прав доступу користувача починається етап зняття показників температури та обробки рук антисептиком. Це є найголовнішим етапом усієї системи, який дозволяє забезпечити відносну безпечність знаходження студентів та викладачів у приміщенні.

Розглянемо етапи процесу налаштування та розробки зазначених модулів. Отже, почати слід з підключення бібліотеки для управління модулем MLX90614 та створення екземпляру цього класу. Має це наступний вигляд:

```
#include <MLX90614.h>
MLX90614 temperature;
```

Далі можна визначити контакти для інфрачервоного датчика, помпи та RGB світлодіоду:

```
#define IR          2
#define PUMP        7
#define GREEN_PIN   4
#define RED_PIN     5
```

Наступним кроком необхідно налаштувати режим роботи вказаних вище контактів. Зробити це слід в головній функції setup() наступним чином:

```
pinMode(IR, INPUT);
pinMode(PUMP, OUTPUT);
pinMode(GREEN_PIN, OUTPUT);
pinMode(RED_PIN, OUTPUT);
```

Також слід ініціювати бібліотеку Wire, яка дозволяє взаємодіяти з I²C пристроями, а в даному випадку з термометром:

```
Wire.begin();
```

На цьому усі налаштування завершуються та можна приступати до реалізації необхідного алгоритму. Як було зазначено раніше, алгоритм роботи має дві можливі реакції на проходження етапу контролю доступу – допуск та відмову. Для зручності розглянемо спочатку реакцію системи на відмову:

```
digitalWrite(RED_PIN, HIGH);
delay(3000);
digitalWrite(RED_PIN, LOW);
```

Програма лише вмикає на деякий час світлодіод, даючи зрозуміти, що в доступі відказано.

Тепер можна детально розглянути роботу основної частини алгоритму. Першим кроком, який виконується після налаштування усіх необхідних для роботи локальних змінних є запуск циклу, в якому будуть виконуватися усі необхідні дії:

```
uint32_t tmr = millis();
do {
    ...
} while (millis() - tmr <= 5000);
```

Перед початком роботи запускається таймер, а після досягнення різниці у часі рівній п'яти секундам, сеанс роботи припиняється. В свою чергу, першим, що виконується у цьому циклі є отримання середніх показників температури. Фрагмент коду наведено у лістингу 2.4.

Лістинг 2.4 – Отримання середніх показників температури

```
if (!get_temperature) {
    for (int i = 0; i < 10; i++) {
        average_temperature += temperature.readTemperatureC() / 10;
        delay(50);
    }
    get_temperature = true;
}
```

Температурні показники беруться як середнє значення декількох вимірювань для підвищення кінцевої точності. Для запобігання повторних вимірів після завершення роботи встановлюється прапор стану, який перешкоджає повторній роботі датчика.

Слід за цим з затримкою виконується увімкнення помпи, яка буде працювати до самого завершення сеансу з користувачем:

```
if (digitalRead(PUMP) == LOW)
    delay(2000);
digitalWrite(PUMP, HIGH);
```

На цьому робота модулів завершується і отримані дані відправляються до сервера. Відбувається це схожим чином як і перевірка наявності користувача у функції `checkUser()` (дивись лістинг 2.2), але з тією відмінністю, що запит до сервера формується з додаванням температури користувача.

У лістингу 2.5 наведено функцію відправлення температури до серверу:

Лістинг 2.5 – Функція відправлення температури до серверу

```
void postTemp(unsigned int UID, float temp) {
    if ((WiFi.status() == WL_CONNECTED)) {

        WiFiClient client;
        HTTPClient http;

        Serial.print("[HTTP] post temp begin...\n");

        String url = "http://" SERVER_IP "/arduino" POST_TEMP_PAGE
"/?UID=" + String(UID) + "&temp=" + temp + "&aud=" AUD_NUM;
        http.begin(client, url); //HTTP

        int httpCode = http.GET();
        if (httpCode == HTTP_CODE_OK)
            const String& payload = http.getString();
        http.end();

    }
```

В свою чергу код сторінки отримання показників мало відрізняється від коду сторінки перевірки прав користувача (дивись лістинг 2.3). Помітною відмінністю є безпосередньо запит до бази даних. Наведемо сам запит:

```
$sql = 'INSERT INTO `user_temp`(`UID`, `temp`, `aud`) VALUES
("'.$UID.$UID.'", '.$temp.$temp.', "'.$aud.$aud.'")';
```

Наведемо структуру таблиці `user_temp` у таблиці 2.4, до якої зберігаються дані про відвідування та температуру користувачів.

Таблиця 2.4 – Структура таблиці user_temp

Поле	Тип	Значення за замовчуванням	Ключ
UID	bigint(20)	NOT NULL	-
temp	float	NOT NULL	-
aud	int(3)	NOT NULL	-
date	timestamp	CURRENT_TIMESTAMP	-

На цьому закінчується реалізація програмної частини роботи самого контролера та налаштування інтерфейсу взаємодії його з базою даних MySQL, розташованої на сервері.

2.5 Розробка веб-інтерфейсу для відстеження даних о користувачах

У цій частині наведена реалізація веб-інтерфейсу, розташованого на сервері. Серед задач, які користувач повинен мати змогу виконувати через сервер є управління усіма користувачами, в тому числі тими, хто також має доступ до додатка. Також необхідно мати можливість керувати роботою системи та отримувати зібрані дані з контролера, а саме відвідуваність та показники температури.

На цьому етапі необхідно розробити структуру розроблюваного додатка для того, щоб визначити необхідні сторінки, які повинні міститись у веб-інтерфейсі, та принцип взаємодії з ними. Додаток повинен мати три основні сторінки для виконання кожної з поставлених задач, тому на рисунку 2.2 наведено схему, що зображує структуру додатку.

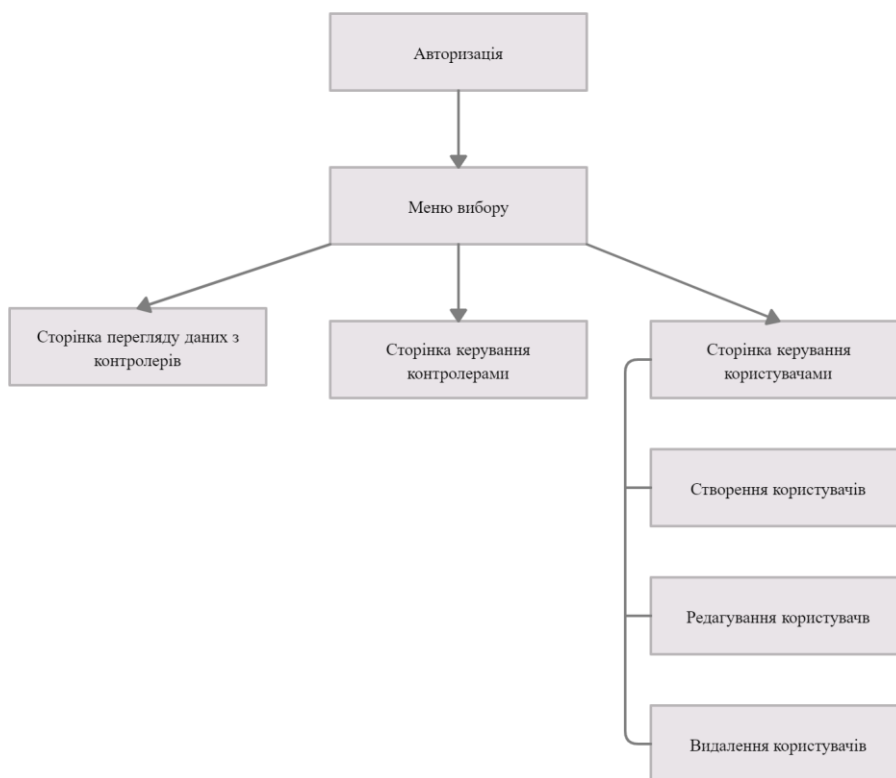


Рисунок 2.2 – Структура веб-додатку

Першою частиною розробки є створення веб-сторінки авторизації користувачів. Користувачі додаються у ручному режимі відповідальною особою, тому усі вони визначені заздалегідь та мають необхідні дані для авторизації.

На рисунку 2.3 зображено користувацький інтерфейс авторизації.

Sign In

Login

Lecturer

Password

.....

Sign in

Рисунок 2.3 – Сторінка авторизації

Розглянемо процес створення та принцип роботи сторінки. Сторінка складається з двох частин – PHP та HTML коду. Перша частина відповідає за логіку роботи сторінки, підключення до бази даних та перевірку прав доступу користувача. Друга частина – за створення форм для введення даних та підключення файлу опису зовнішнього вигляду CSS. Отже, спочатку розглянемо HTML частину сторінки. Код розмітки наведено у лістингу 2.6:

Лістинг 2.6 – HTML-код сторінки авторизації

```
<head>
  <link rel="stylesheet" href="styles/default.css">
</head>
<H1> Sign In </H1>
<div>
  <form method="POST">
    Login <input name="login" type="text" required><br>
    Password <input name="pass" type="password" required><br>
    <input class="default" name="submit" type="submit"
value="Sign in">
  </form>
</div>
```

У середині тега `<head>` розташовується тег `<link>`, який підключає зовнішній CSS-файл, необхідний для встановлення зовнішнього вигляду сторінки. У середині тегу `<div>` розташовується форма для передачі даних, які були введені користувачем, на сервер за допомогою методу POST. Таким чином організується взаємодія між клієнтом та сервером, і PHP частина сторінки має змогу оброблювати вхідні дані.

Далі розглянемо серверну частину сторінки. На початку роботи відбувається перевірка, чи були дані відправлені до серверу:

```
if(isset($_POST['submit']))
```

Якщо перевірка була вдалою, тоді відбувається підключення до бази даних, що знадобиться для перевірки наявності введених даних у базі створених облікових записів:

```
$link = mysqli_connect("localhost", "login", "password",
"arduino");
```

У наступному кроці відбувається отримання зашифрованого пароля, де введений логін відповідає запису у таблиці logins, структура якої наведена у таблиці 2.5. Код отримання паролю наведено у лістингу 2.7.

Таблиця 2.5 – Структура таблиці logins

Поле	Тип	Значення за замовчуванням	Ключ
RightId	varchar(3)	NOT NULL	-
UID	bigint(20)	UNIQUE	-
login	varchar(32)	NOT NULL	-
password	varchar(32)	NOT NULL	-

Лістинг 2.7 – Фрагмент коду отримання паролю

```
$login = $_POST['login'];
$sql = 'SELECT password FROM logins WHERE login = "'.$login.'"';
$result = mysqli_query($link, $sql);
$data = mysqli_fetch_assoc($result);
```

Далі відбувається порівняння паролю з базою:

```
if($data['password'] === md5(md5($_POST['pass'])))
```

Якщо пароль співпав, створюється новий сеанс для поточного користувача, яка запобігає несанкціонованому доступу до захищених сторінок та дозволяє зберігати персональні дані, які необхідні для коректної роботи з веб-інтерфейсом.

Далі запускається нова сесія та в якості параметру сесії передається ім'я користувача:

```
session_start();
$_SESSION["login"] = $login;
```

Для переходу між сторінками у проєкті використовується функція redirect(), яка знаходиться у зовнішньому файлі та виконує код на мові javascript, який динамічно створюється за допомогою команди echo. Код функції наведено у лістингу 2.8.

Лістинг 2.8 – Функція переходу між сторінками

```
<?php
function redirect($link){
```

```

echo "<script type='text/javascript'>
    window.location = ".$link.";
</script>";
}
?>

```

Для підключення зовнішніх файлів використовують вираз include:

```
include 'functions/switchPage.php';
```

До функції передається адрес сторінки, до якої необхідно перейти.

Викликається вона одразу після створення сесії наступним чином:

```
redirect("'selectAction.php'");
```

Після переходу користувач опиняється на сторінці вибору наступних дій.

Серед доступних можливостей присутні пункти, призначенням яких є пересилання користувача на наступні сторінки:

- View tables – перегляд даних, зібраних з контролера;
- Manage controllers – керування роботою кожного окремого контролера;
- Manage users – додавання, видалення і редагування користувачів;
- Sign out – вихід з облікового запису;

Користувачський інтерфейс зображено на рисунку 2.4.

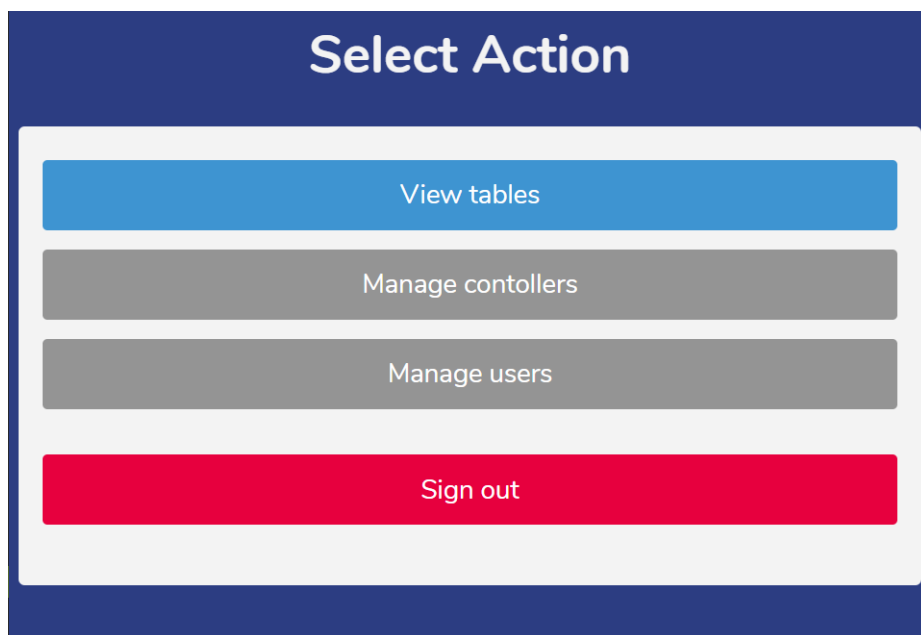


Рисунок 2.4 – Сторінка вибору наступних дій

Можливі варіанти дій користувача залежать від прав, якими він володіє. Існує дві групи прав, а саме: звичайні користувачі та адміністратори. Звичайні користувачі мають можливість лише переглядати дані з доступних їм аудиторій, а в свою чергу, адміністратори можуть володіти усіма можливими правами. Отже, на рисунку 2.4 зображені можливості найзвичайнішого користувача, який не володіє ніякими особливими правами.

На початку роботи скрипту виконується підключення необхідних зовнішніх функцій та запуск авторизованої сесії:

```
include 'functions/switchPage.php';
include 'functions/buttons.php';
session_start();
```

Файл `buttons.php` зберігає у собі функції з кнопками. Розглянемо функцію `exitSession()`, яка створює форму `<input>`, що виконує роботу по закінченню сеансу сесії та виходу з облікового запису. При наявності переданого значення про вихід, виконується знищення змінних та сеансу сесії, після чого відбувається переспрямування користувача на головну сторінку за допомогою функції `redirect()` (дивись лістинг 2.8). Код функції наведено у лістингу 2.9.

Лістинг 2.9 – Функція виходу

```
<?php
function exitSession(){
    echo '<form method="POST">
    <input class="default" name="exit" type="submit" value="Sign
out" style="background-color: #E7003E">
    </form>';

    $exit = $_POST['exit'];
    if(isset($exit)){
        session_unset();
        session_destroy();
        redirect("'index.php'");
    }
}
?>
```

Захист від несанкціонованого доступу виконується перевіркою змінної сесії:

```
if(isset($_SESSION["login"]))
```

Якщо такої змінної не існує, тоді викликається повідомлення про відмову у доступі, що наведено на рисунку 2.5. Така реакція дійсна для будь-якої сторінки веб-інтерфейсу.

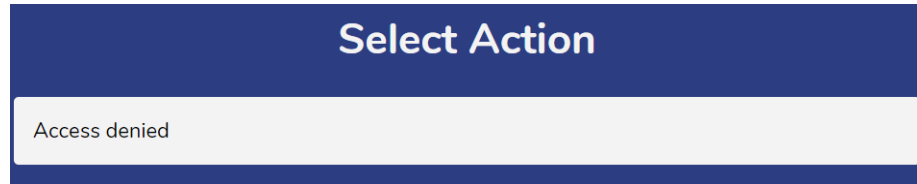


Рисунок 2.5 – Реакція на несанкціонований доступ

Перевірка прав доступу до окремих розділів веб-інтерфейсу реалізована прив'язкою функції `checkInputRights()` (наведено у лістингу 2.11) до елементу форми `<input>`, яка виводить атрибут доступу в залежності від прав користувача. Реалізація наведена у лістингу 2.10.

Лістинг 2.10 – створення форми вибору пункту меню

```
<form method="POST">
  <input class="default" name="table" type="submit" value="View
tables" <?php checkInputRights("t") ?>>
  <input class="default" name="contoller" type="submit"
value="Manage contollers" <?php checkInputRights("c") ?>>
  <input class="default" name="user" type="submit" value="Manage
users" <?php checkInputRights("u") ?>>
</form>
```

Лістинг 2.11 – Функція перевірки прав користувача

```
function checkInputRights($right) {
  $link = mysqli_connect("localhost", "login", "password",
"arduino");
  $sql = 'SELECT RightId FROM logins WHERE login =
"' . $_SESSION["login"] . '"';
  $result = mysqli_query($link, $sql);
  $data = mysqli_fetch_array($result);
  $permission = stripos($data['RightId'], $right);
  if($permission !== false) {
    echo 'enabled';
  } else {
    echo 'disabled';
  }
}
```

Функція виконує підключення до бази даних для отримання прав авторизованого користувача. Далі відбувається перевірка наявності необхідного дозволу. В залежності від його наявності виводиться один з атрибутів поля.

Реалізація переадресації при натисканні кнопок наведена у лістингу 2.12.

Лістинг 2.12 – переадресація користувача на наступну сторінку

```
if (isset($_POST['table'])) {
    redirect("'table.php'");
} else if (isset($_POST['contoller'])) {
    redirect("'manageControllers.php'");
} else if (isset($_POST['user'])) {
    redirect("'manageUsers.php'");
}
```

Тепер необхідно розглянути сторінку перегляду даних про користувачів. На ній користувач має змогу вибрати параметри за яким буде сформована таблиця даних. Серед критеріїв для фільтрації можна вибрати серед доступних аудиторій, груп та за певну дату. Таким чином, за вказаними критеріями динамічно формується таблиця, що містить необхідні дані. Інтерфейс сторінки з виведеною таблицею вказано на рисунку 2.6.

The screenshot shows a web interface titled "Table". It contains a table with two columns: "Name" and "Temperature". The table lists two users: Petrenko O. O. with a temperature of 37.1, and Ivanov I. I. with a temperature of 37.8. Below the table, there are three filter sections: "Group:" with a dropdown menu showing "CSN-510", "Auditorium:" with a dropdown menu showing "101", and "Date:" with a dropdown menu showing "2021-04-28". At the bottom, there are three buttons: "Send" (blue), "Back" (gray), and "Sign out" (red).

Name	Temperature
Petrenko O. O.	37.1
Ivanov I. I.	37.8

Group:

Auditorium:

Date:

Рисунок 2.6 – Сторінка перегляду даних про користувачів

Налаштування сторінки відбувається таким же чином як на минулій сторінці. Відбувається підключення зовнішніх функцій, налаштування сесії, додається додатковий файл стилів table.css, налаштовується підключення до бази даних.

Зміст сторінки можна поділити на дві частини, де перша – це створення форм для введення параметрів формування таблиці та ініціація функцій з кнопками повернення та виходу. Друга частина – безпосередньо створення таблиці та заповнення її вибраними даними.

Отже, розглянемо першу частину веб-сторінки. Розпочинається вона зі створення та виконання усіх необхідних запитів до бази даних. Вибірка виконується по наступним параметрам: групах; аудиторіях, до яких користувач має доступ; датах проведення занять. Таким чином формується три групи запитів, які наведені у лістингу 2.13.

Лістинг 2.13 – Запити до бази даних

```
$sql_group = 'SELECT DISTINCT `group` FROM `user` WHERE `group` !=
""';
$arr_group = mysqli_query($link, $sql_group);

$sql_aud = 'SELECT aud FROM aud JOIN logins ON aud.UID =
logins.UID WHERE logins.login = "' . $_SESSION["login"] . "'';
$arr_aud = mysqli_query($link, $sql_aud);

$sql_date = 'SELECT DISTINCT DATE(date) FROM user_temp';
$arr_date = mysqli_query($link, $sql_date);
```

Далі створюється безпосередньо форма, у складі якої будуть три елементи типу <select> та елемент <input>, який посилає зібранні дані до таблиці. У лістингу 2.14 наведено код цієї форми.

Лістинг 2.14 – Створення форми для отримання параметрів сортування

```
echo '<form method="POST">';
createSelect('Group', 'groupList', $arr_group, 'group');
createSelect('Auditorium', 'audList', $arr_aud, 'aud');
createSelect('Date', 'dateList', $arr_date, 'DATE(date)');
echo '<INPUT TYPE="submit" name="Set" value="Send" />
</form>';
```

Створення кожного окремого меню відбувається у функції `createSelect()`. Всередині неї відбувається перебір усіх значень, які були отримані з бази даних, та на їх основі створюються пункти меню. Код функції наведено у лістингу 2.15.

Лістинг 2.15 – Функція створення меню вибору

```
function createSelect($title, $select_name, $sql_arr, $sql_row){
    echo " ".$title." :
    <select name='".$select_name."'>";
    while ($result = mysqli_fetch_array($sql_arr)) {
        echo "<option value='" . $result[$sql_row] . "'>" .
$result[$sql_row] . "</option>";
    }
    echo "</select>";
}
```

На цьому перший етап створення сторінки завершується, після чого необхідно формувати таблицю. Початок її створення є реакцією на відправлення запиту формою з лістинга 2.14. Виглядає це наступним чином:

```
$selectedDate = $_POST['Set'];
if(isset($selectedDate))
```

Далі виконується запит до бази даних, метою якого є отримання імені та температури усіх відвідувачів, які підходять за заданими раніше критеріями. Відбір полів відбувається з двох таблиць бази даних, що пов'язані між собою полем UID, яке відповідає унікальному ідентифікатору RFID-мітки користувача. Запит має наступний вигляд:

```
$sql_table = 'SELECT user.name, user_temp.temp FROM user_temp
JOIN user ON user.UID = user_temp.UID WHERE aud = '.$_POST['audList'].'
AND DATE(date) = "'.$_POST['dateList'].'" AND user.group =
'.$_POST['groupList'].'" ORDER BY date desc';
```

За результатом вибірки формується таблиця, основною частиною якої є цикл, що виконує безпосереднє встановлення отриманих даних до таблиці. Код цієї частини таблиці наведено у лістингу 2.15.

Лістинг 2.15 – Заповнення таблиці даними

```
$arr_table = mysqli_query($link, $sql_table);
while($res_table = mysqli_fetch_array($arr_table)){
    echo '<tr>
        <td>'.$res_table['name'].'</td>
```

```

        <td>'.$res_table['temp'].'</td>
    </tr>';
}

```

На цьому формування сторінки завершується. Користувач може надалі виконувати різноманітні операції, пов'язані з таблицею, або вийти із системи чи повернутися до попередньої сторінки вибору дій. Виконання цієї дії реалізовано у функції `returnBack()`, яка розташована у зовнішнього файлу `buttons.php`. Реалізація функції наведено у лістингу 2.16.

Лістинг 2.16 – Функція повернення

```

function returnBack(){
    echo '<form method="POST">
        <input      class="default"      name="return"      type="submit"
value="Return back" style="margin-bottom: -10px">
    </form>';

    $return = $_POST['return'];
    if(isset($return)){
        redirect("'selectAction.php'");
    }
}

```

Наступним розглянемо сторінку взаємодії з контролерами. Для того, щоб отримати можливість перейти до цієї та сторінки керування користувачами, необхідно мати особливі права доступу. Тому наступні дії будуть розглядатися від імені адміністратора.

Сторінка керування контролерами повинна містити інформацію про стан роботи кожної окремої системи. Серед задач, які користувач повинен мати можливість виконувати на цій сторінці користувач – це можливість виконувати перевантаження контролеру та переходу у режим економії енергії, а також отримувати інформацію про те, чи знаходиться контролер у робочому стані, або він вимкнен.

Отримання інформації про стан системи відбувається за допомогою викликів бази даних MySQL контролером через GET-запити до сервера, де і виконується запит з повідомленням. Отже, у таблиці 2.6 наведено структуру цієї таблиці.

Таблиця 2.6 – Структура таблиці controller_status

Поле	Тип	Значення за замовчуванням	Ключ
name	int(3)	NOT NULL	UNIQUE
reset_status	varchar(10)	NOT NULL	-
power_status	varchar(10)	NOT NULL	-
online_status	varchar(10)	NOT NULL	-
update_time	time	NOT NULL	-

У таблиці 2.7 розглянуто значення, що можуть приймати поля. Ці значення безпосередньо впливають на роботу системи та пов'язують веб-додаток з контролерами.

Таблиця 2.7 - значення полів таблиці

Поле	Значення	Пояснення
name	*	Відповідає номеру аудиторії, де встановлен контролер
reset_status / power_status	status_on	Відповідає звичайній роботі
	status_off	Відповідає виконанню операції, при якій контролер припиняє свою роботу
online_status	online	Підключен до живлення
	offline	Вимкнен
update_time	*	Зберігає час останньої активності системи

Таким чином, в залежності від значень полів таблиці, відбувається визначення стану роботи контролерів на користувацькій сторінці керування. Розглянемо яким чином відбувається взаємодія контролеру з базою даних.

Для початку необхідно визначити ідентифікатори для заміщення назв сторінок, через які відбуватиметься взаємодія з сервером:

```
#define CHECK_STATUS_PAGE      "/checkStatus.php"
#define ENABLE_STATUS_PAGE    "/enableStatus.php"
```

Далі у головній функції setup() викликається функція postStatus(), яка повідомляє сервер про увімкнення системи. Код функції наведено у лістингу 2.17.

Лістинг 2.17 – Функція повернення

```
void postStatus() {
    if ((WiFi.status() == WL_CONNECTED)) {

        WiFiClient client;
        HTTPClient http;

        Serial.print("[HTTP] power on begin...\n");
        String url = "http://" + SERVER_IP + "/arduino"
ENABLE_STATUS_PAGE "?aud=" AUD_NUM;
        http.begin(client, url); //HTTP

        int httpCode = http.GET();
        http.end();
    }
}
```

З боку серверу по заданій адресі виконується оновлення статусу перевантаження, що дозволяє автоматично відновлювати стан цього поля після увімкнення. Код скрипту наведено у лістингу 2.18.

Лістинг 2.18 – Код відправлення запиту до бази даних

```
<?php
$name = $_GET['aud'];
$link = mysqli_connect("localhost", "maxim", "1234", "arduino");
$sql = 'UPDATE `controller_status` SET `reset_status`="status_on"
WHERE name = '.$name;
mysqli_query($link, $sql);
?>
```

У головній циклічній функції loop() відбувається дві перевірки на те, чи надійшла команда про перевантаження або перехід у режим глибокого сну. За виконання перевірки відповідає функція checkStatus(), що приймає прапор перевірки, який говорить про те, на яку команду необхідно дати реакцію. Функція приймає значення «1» чи «0», де «1» відповідає переходу у режим сну, а «0» – перевантаженню. Це значення передається у якості параметру до сервера, де, в залежності від значення виконується необхідний запит до бази даних. Отже, код функції checkStatus() наведено у лістингу 2.19.

Лістинг 2.19 – Код відправлення запиту до бази даних

```
boolean checkStatus(String code) {
    if ((WiFi.status() == WL_CONNECTED)) {

        WiFiClient client;
        HTTPClient http;

        Serial.print("[HTTP] [status " + code + "] check reset/power
status begin...\n");
        String url = "http://" SERVER_IP "/arduino" CHECK_STATUS_PAGE
"/?code=" + code + "&aud=" AUD_NUM;
        http.begin(client, url); //HTTP

        int httpCode = http.GET();
        if (httpCode == HTTP_CODE_OK) {
            const String& payload = http.getString();
            if (payload == "TRUE")
                return true;
            else
                return false;
            http.end();
        }
    }
}
```

У лістингу 2.20 наведено код сторінки, яка виконує обробку серверної частини програми.

Лістинг 2.20 – Код відправлення запиту до бази даних

```
<?php
$code = $_GET['code'];
$aud = $_GET['aud'];

if (isset($code)) {
    $link = mysqli_connect("localhost", "login", "password",
"arduino");
    if ($code == 0) {
        $sql = 'SELECT `reset_status` FROM `controller_status` WHERE
`name` = "'. $aud. "'';
        $result = mysqli_query($link, $sql);
        $row = mysqli_fetch_array($result);
        if ($row['reset_status'] == "status_off") {
            print("TRUE");
        }
    } elseif ($code == 1) {
```

```

        $sql = 'SELECT `power_status` FROM `controller_status` WHERE
`name` = "'. $aud.'";
        $result = mysqli_query($link, $sql);
        $row = mysqli_fetch_array($result);
        if ($row['power_status'] == "status_off") {
            print("TRUE");
        }
    }
    $sql_online = 'UPDATE `controller_status` SET `online_status` =
"online", `update_time` = NOW() WHERE `name` = "'. $aud.'";
    mysqli_query($link, $sql_online);
}
?>

```

Скрипт повертає «TRUE», якщо. Також, незважаючи на запити до бази даних, у кінці роботи скрипт завжди повідомляє базу про те, що контролер активний та працює справно. Це необхідне повідомлення, так як від нього залежить статус роботи системи у базі. Механізм задання онлайн стану роботи реалізується за допомогою використання подій MySQL.

У проєкті подія використовується для того, щоб автоматично виконувати зміну стану активності системи, тим самим повідомляє користувача про її вимкнення. Код події наведено у лістингу 2.18.

Лістинг 2.21 – Подія MySQL

```

CREATE EVENT set_offline_status
ON SCHEDULE EVERY 1 MINUTE
STARTS NOW()
DO
    UPDATE controller_status SET `online_status` = "offline" WHERE
`update_time` < (NOW() - INTERVAL 1 MINUTE)

```

Принцип роботи події полягає у тому, що кожну хвилину виконується запит, який оновлює значення полів, у яких останнє повідомлення прийшло пізніше заданого інтервалу також у одну хвилину. Тим самим відбувається автоматичне оновлення усіх полів, що відповідають заданій умові.

Розглянемо реакцію контролеру на повідомлення, які надходять від сервера. В першу чергу розглянемо процес переходу системи у режим сну. Існує декілька режимів, у яких контролер може знаходитись, серед них виділяють наступні:

- No-sleep – режим при якому працюють усі блоки мікроконтролеру;
- Modem-sleep – стандартний режим роботи, який вмикається, коли контролер підключено до точки доступу. Цей режим вимикає Wi-Fi між інтервалами DTIM;
- Light-sleep – попри виконання тієї ж функції, що й Modem-sleep, вимикає системний годинник та припиняє роботу CPU;
- Deep-sleep – вмикається усі модулі, окрім годиннику реального часу RTC;
- Power-down – вимикаються усі модулі, тому для пробудження системи необхідно подати зовнішній сигнал [15].

Найкращим режимом, що можна використовувати в рамках цього проекту є Deep-sleep, через те, що він пропонує можливість майже повного вимкнення системи, але при можливості увімкнення без зовнішнього впливу, що ідеально підходить для використання його разом з веб-додатком.

При виконанні запиту на перехід системи у режим сну відбувається виклик методу `deepSleep()` стандартної бібліотеки систем на базі сімейства мікроконтролерів ESP. Значення задаються у мікросекундах і на вказаний період контролер переходить у режим зниженого енергоспоживання. Для пробудження модуля необхідно під'єднати контакт GPIO16 до виводу RST. Таким чином, контролер буде періодично пробуджуватись та перевіряти стан необхідного поля таблиці, поки користувач не увімкнить систему за допомогою веб-інтерфесу. У лістингу 2.22 наведено код переходу системи у режим сну.

Лістинг 2.22 – Код переходу у режим сна

```
if (checkStatus("1")) { //power
  Serial.println("Enter sleep mode...");
  ESP.deepSleep(10 * 1000000);
}
```

Тепер розглянемо алгоритм роботи при надходженні запиту на перевантаження системи. Як і у минулому випадку, виконуватись ця подія буде за допомогою стандартної бібліотеки ESP, у якій присутній метод `restart()`, при

виклику якого і виконує перевантаження. Але, на відміну від минулого випадку, повернення стану перевантаження у таблиці відбувається автоматично при завантаженні системи та виклику функції `postStatus()` (дивись лістинг 2.17). Код перевірки стану перевантаження наведено у лістингу 2.23.

Лістинг 2.23 – Код перевантаження системи

```
if (checkStatus("0")) { //reset
    Serial.println("Restarting...");
    ESP.restart();
}
```

На цьому налаштування контролера завершується, далі розглянемо сторінку користувацького веб-інтерфейсу для взаємодії з системою. На рисунку 2.7 зображено інтерфейс цієї сторінки.

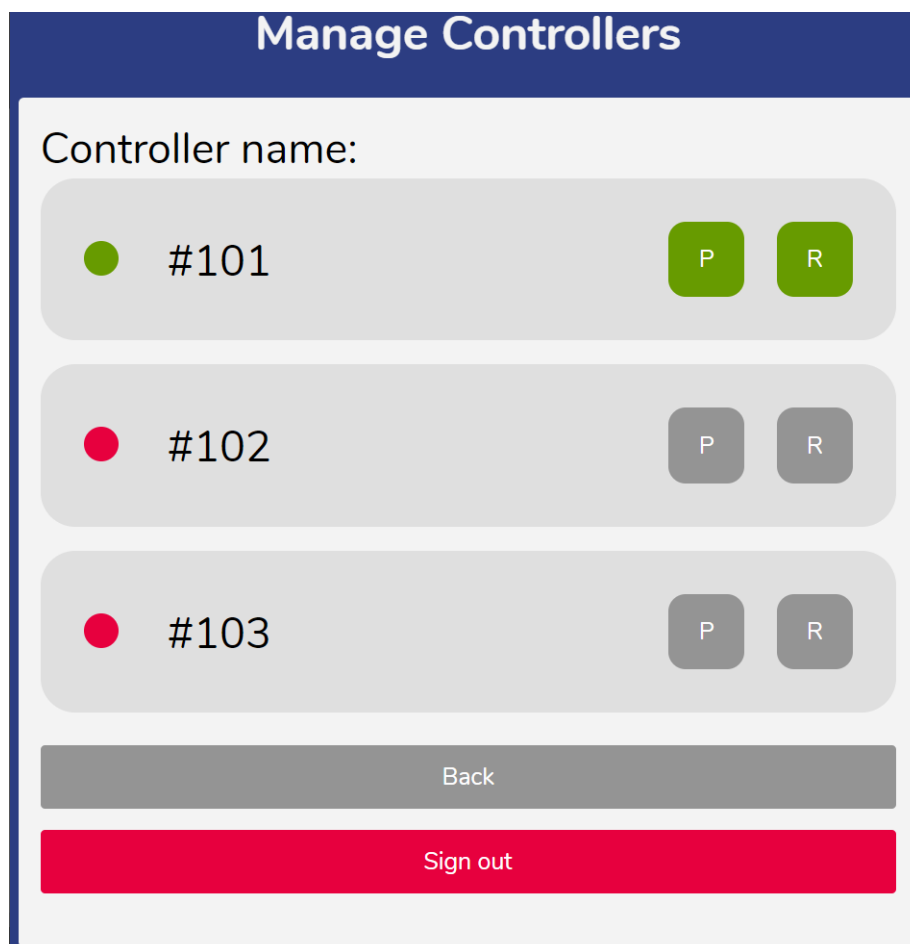


Рисунок 2.7 – Сторінка керування контролерами

Сторінка складається з переліку усіх контролерів, які були визначені у базі даних. Кожна окрема комірка складається зі стану роботи системи, назви, кнопок вимкнення та перевантаження. Розглянемо код цієї сторінки.

Далі відбувається стандартне налаштування сторінки та перевірка наявності сесії та прав користувача на доступ до неї. Для цього виконується зовнішня функція `checkRight()`, яка наведена у лістингу 2.24.

Лістинг 2.24 – Функція перевірки прав користувача

```
<?php
function checkRight($right){
    $link    =    mysqli_connect("localhost",    "login",    "password",
"arduino");
    $sql      =    'SELECT    `RightId`    FROM    `logins`    WHERE
`login`="' . $_SESSION["login"] . "'';
    $result = mysqli_query($link, $sql);
    $data = mysqli_fetch_array($result);

    $permission = stripos($data['RightId'], $right);
    if($permission !== false)  {
        return true;
    }
}
?>
```

Після проходження усіх перевірок відбувається запит на отримання полів, що необхідні для заповнення інформацією сторінки. Відбувається формування масивів даних, які відповідають елементам кожної комірки. Форма заповнення наведена у лістингу 2.25.

Лістинг 2.25 – Код отримання масивів даних

```
$sql = 'SELECT * FROM controller_status';
$result = mysqli_query($link, $sql);
while($row = mysqli_fetch_array($result, MYSQLI_NUM))  {
    $name[] = $row[0];
    $status_r[] = $row[1];
    $status_p[] = $row[2];
    $status[] = $row[3];
}
```

Далі відбувається ініціація кнопок кожної комірки. Циклічним викликом виконуються запити до бази даних функцією `control_btns()`, що наведена у лістингу 2.26, поки не закінчатся елементи масиву даних. Має це наступний вигляд:

```
for ($i=0; $i < sizeof($name); $i++) {
    control_btns($link, $name[$i], $status_r[$i], $status_p[$i]);
}
```

Лістинг 2.26 – Функція створення

```
function control_btns($link, $name, $status_r, $status_p){

    if (isset($_POST['reset-'. $name])) {
        $sql_update      =      'UPDATE      `controller_status`      SET
`reset_status`="status_off" WHERE name = ' . $name;
        mysqli_query($link, $sql_update);
    }

    if (isset($_POST['power-'. $name])) {
        if ($status_p == "status_off") {
            $sql_update      =      'UPDATE      `controller_status`      SET
`power_status` = "status_on" WHERE name = ' . $name;
        } else {
            $sql_update      =      'UPDATE      `controller_status`      SET
`power_status` = "status_off" WHERE name = ' . $name;
        }
        mysqli_query($link, $sql_update);
    }

}
```

Функція складається з двох частин, які відповідають кнопкам взаємодії з контролером. При натисканні будь-якої з них виконується запит до бази даних з оновленням стану поля, що відповідає натиснутій кнопці. Отже, перша кнопка відповідає перевантаженню системи, а друга, в свою чергу, відповідає кнопці переходу у режим глибокого сну. На відміну від минулого випадку, кнопка має два стану, які відповідають увімкненому та вимкненому стану роботи. Таким чином виконується взаємодія кнопок керування з базою даних та самим контролером.

Далі відбувається формування самої комірки. За допомогою циклу виконується створення і заповнення відповідних елементів сторінки визначеними раніше даними. Далі виконується перевірка активності контролеру на те, чи знаходиться він вимкненому стані. Якщо результат позитивний, тоді кнопки

управління контролеру стають неактивними, що запобігає помилковому натисканню на ці елементи комірки. Неактивний стан зображено на рисунку 2.8.

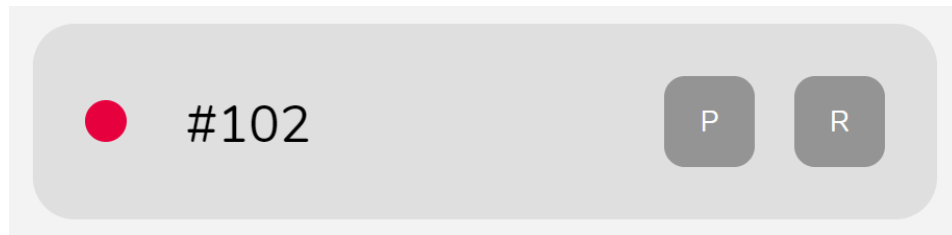


Рисунок 2.8 – Зображення неактивного стану системи

Код циклу створення комірок наведено у лістингу 2.27.

Лістинг 2.27 – Цикл створення комірок

```
for ($i=0; $i < sizeof($name) ; $i++) {
    if($status[$i] == "offline"){
        $disabled = "disabled";
    }
    echo '<form method="POST" style="padding: 40px; background-
color:#DFDFDF; border-radius: 32px; height: 70px">
    <input class="'. $status[$i].'" name="online-'. $name[$i].'"
type="submit" value="" disabled >
    <p style="font-size: 40px; margin-top: 10px; margin-left:
40px; display: inline-block">#'. $name[$i]. '</p>
    <input '. $disabled.' class="'. $status_r[$i].'" name="reset-
'. $name[$i].'" type="submit" value="R" style="float: right; margin-
left: 30px" >
    <input '. $disabled.' class="'. $status_p[$i].'" name="power-
'. $name[$i].'" type="submit" value="P" style="float: right">
    </form>';
}
```

При натисканні на кнопки керування відбувається зміна стану їх роботи, що відбувається за допомогою функції `control_btns()` (дивись лістинг 2.26), і візуальне відображення також залежить від значень полів таблиці. Візуальне змінення відбувається за допомогою заміни класу стилю CSS, де назва класу відповідає станам, що зазначені у таблиці бази даних (дивись таблицю 2.7). Таким чином реалізується залежність стилю від значень полів бази даних та стану роботи контролеру. Візуальне відображення можливих станів роботи кнопок наведено на рисунку 2.9.

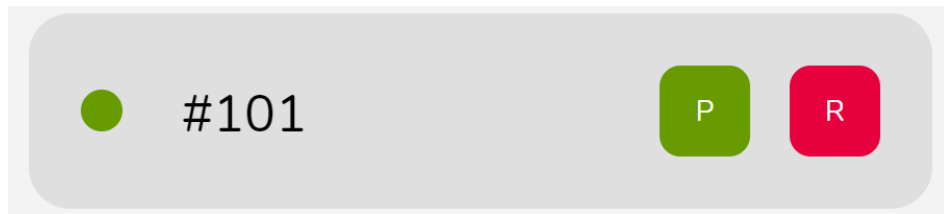


Рисунок 2.9 – Зображення станів кнопок

Останньою сторінкою, яку необхідно розглянути є сторінка керування користувачами. Як було зазначено раніше, на цій сторінці користувач повинен мати можливість виконувати операції над обліковими записами користувачів, а саме: додавання, видалення і редагування облікових записів. При переході на цю сторінку з'являється меню вибору дії, яку хоче виконати користувач. Зображення меню наведено на рисунку 2.10.

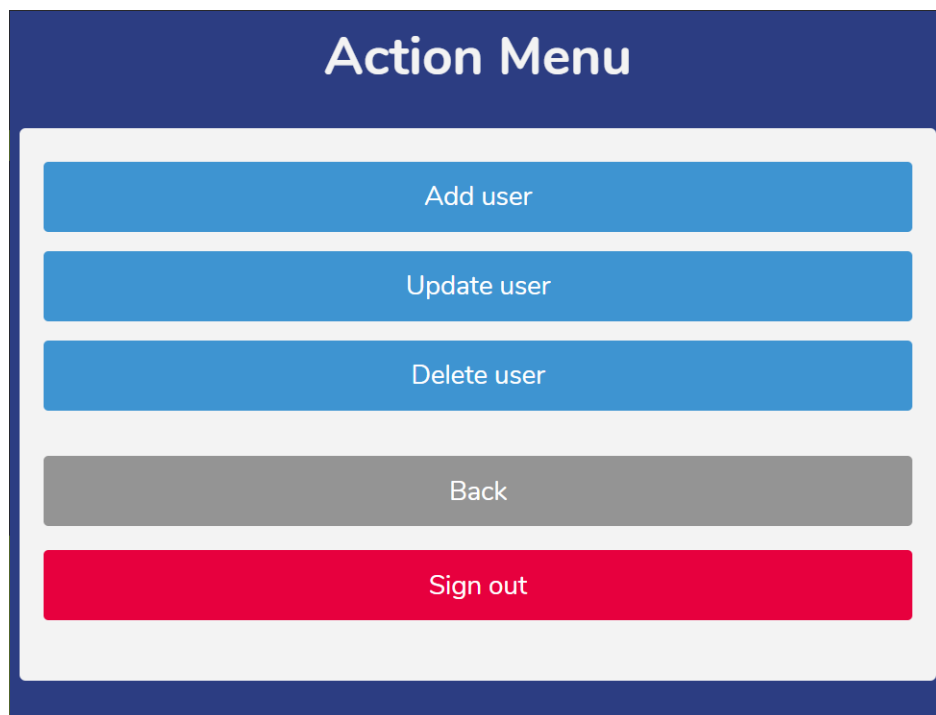


Рисунок 2.10 – Меню вибору операції

Розглянемо реалізацію цього меню. Створення його відбувається, якщо до сервера не надійшов жоден запит від елементів меню, тобто воно з'являється за умовчанням при відсутності будь-яких дій зі сторони користувача. Наведемо

фрагмент коду у лістингу 2.28, що реалізує створення цього меню та реакцію переходу при натисканні на його елементи.

Лістинг 2.28 – Меню вибору дій над користувачами

```
if($_GET['action'] == "Add user") {
    setUserData($link, "Add");
} elseif ($_GET['action'] == "Update user") {
    setUserData($link, "Update");
} elseif ($_GET['action'] == "Delete user") {
    deleteUser($link);
} else {
    ?>
    <H1> Action Menu </H1>
    <div>
    <form method="GET">
    <input class="default" name="action" type="submit" value="Add
user">
    <input class="default" name="action" type="submit"
value="Update user">
    <input class="default" name="action" type="submit"
value="Delete user">
    </form>
    <?
    returnBack();
    exitSession();
    }
```

Таким чином перехід до наступних розділів сторінки реалізується за допомогою перевірки наявності GET-запиту, що надійшов від створеного меню. При успіху перевірки виконується функція `setUserData()` для створення або оновлення користувачів, а функція `deleteUser()` для їх видалення.

Розглянемо функцію `setUserData()`. При її виконанні створюється форма для введення даних про користувача, його унікальний ідентифікатор, ім'я та пароль, а також права доступу. На рисунку 2.11 наведено інтерфейс форми для створення нових користувачів. Форма для оновлення даних відрізняється лише тим, що на ній відсутнє поле для введення ідентифікатору, а визначення користувача відбувається за допомогою його ім'я.

Рисунок 2.11 – Форма для створення нового користувача

Розглянемо те, як утворюється ця форма. Розділити функцію `setUserData()` можна на три частини – створення форми для введення даних, отримання значень полів форми та виконання запитів до бази даних.

Почнемо зі створення форми. Код її формування наведено у лістингу 2.29.

Лістинг 2.29 – Створення користувацького інтерфейсу

```

?><H1> <?echo $query;?> Menu </H1>
<div>
  <form method="POST">
    <?if ($query == "Add") {
      echo 'UID <input name="UID" type="text" required><br>';
    }?>
    Login <input name="login" type="text" required><br>
    Password <input name="pass" type="text" required><br>
    <input type="checkbox" name="t"> table
    <input type="checkbox" name="c"> controller
    <input type="checkbox" name="u"> user
    <input class="default" name="submit" type="submit"
value="<?echo $query;?> user" style="margin-top: 30px">
  </form><?
  userBackBtn();
  echo "</div>";

```

Після введення даних виконується їх обробка для використання у запитах до бази даних. Код визначення наведено у лістингу 2.30.

Лістинг 2.30 – Формування даних для запитів

```

if (isset($_POST['t'])) {
    $right_t = "T";
}
if (isset($_POST['c'])) {
    $right_c = "C";
}
if (isset($_POST['u'])) {
    $right_u = "U";
}
$rights = $right_t.$right_c.$right_u;

$password = md5(md5(trim($_POST['pass'])));
$login = $_POST['login'];
$UID = $_POST['UID'];

```

Далі виконується відправлення запиту на створення або редагування користувача. Визначення того, який запит необхідно відправити відбувається за допомогою переданої до функції змінної `$query`, яка і вказує на необхідну дію. У лістингу 2.31 наведено формування та виконання запитів.

Лістинг 2.31 – Виконання запиту до бази даних

```

if ($query == "Add") {
    $sql = 'INSERT INTO `logins` (`RightId`, `UID`, `login`,
`password`) VALUES ("'.$rights.'", "'.$UID.'", "'.$login.'",
"'.$password.'")';
} elseif ($query == "Update") {
    $sql = 'UPDATE `logins` SET `RightId` = "'.$rights.'',
`password` = "'.$password.'" WHERE `login` = "'.$login.'"'';
}
mysqli_query($link, $sql);

```

Наступним кроком розглянемо принцип роботи функції `deleteUser()`. Вона також створює форму, у якій присутнє поле для введення логіну облікового запису, за допомогою якого відбувається видалення користувача з бази. На рисунку 2.12 зображено інтерфейс цієї сторінки.

Рисунок 2.12 – Форма для видалення користувачів

Функція складається з двох частин – створення користувацького інтерфейсу та відправлення запиту на видалення з бази даних. У лістингу 2.32 наведено формування користувацького інтерфейсу.

Лістинг 2.32 – Створення інтерфейсу

```
?>
<H1> Delete Menu </H1>
  <div>
    <form method="POST">
      Login <input name="login" type="text" required><br>
      <input class="default" name="submit" type="submit"
value="Drop user" style="margin-top: 30px">
    </form>
  <?
  userBackBtn();
  echo "</div>";
```

Далі виконується отримання даних з веденого поля та виконання запиту. Код наведено у лістингу 2.33.

Лістинг 2.33 – Виконання запиту на видалення користувача

```
if (isset($_POST['submit'])) {
  $login = $_POST['login'];
  $sql = 'DELETE FROM `logins` WHERE `login` = "'.$login.'";
  mysqli_query($link, $sql);
}
```

3 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ

Наступним етапом розробки є проектування та тестування апаратної частини системи. У цьому розділі буде виконуватись складання схеми пристрою, моделювання її роботи та кінцеве тестування готової системи. Виконання цього етапу дозволяє заздалегідь визначити усі зв'язки між елементами, їх взаємодію та призначення, визначити принцип роботи, а також дізнатись про можливі помилки та недоліки у проектуванні.

Розробка спирається на дані, які були отримані у першому розділі, де розглядалися компоненти та модулі, що застосовуються при розробці системи.

При виконанні роботи будуть використовуватись послідовні інтерфейси передачі даних, тому важливо заздалегідь визначити які контакти підтримують необхідні технології. Тому наведемо розпіновку плати NodeMCU на рисунку 3.1.

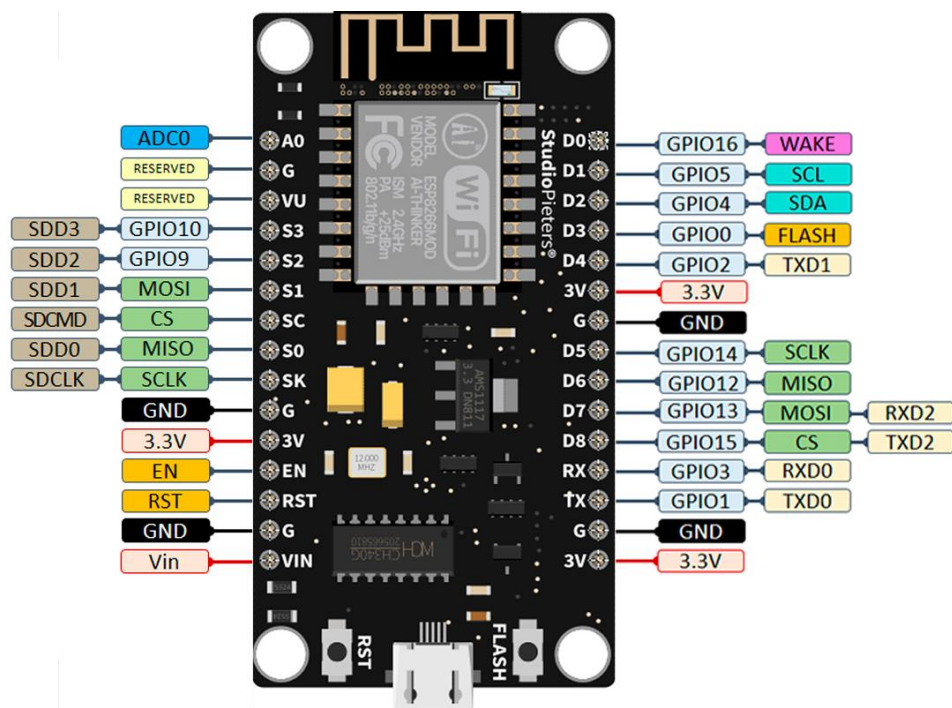


Рисунок 3.1 – Розпіновка плати NodeMCU

3.1 Розробка функціонального вузла аутентифікації користувачів

Розпочнемо налаштування системи з підключення радіочастотного ідентифікатора MFRC522 до плати. Модуль підтримує декілька інтерфейсів передачі даних, серед яких є: SPI, I²C та UART. На цьому модулі заздалегідь вибрано інтерфейс SPI, тому для підключення необхідно використовувати його.

Робота інтерфейсу SPI заснована на принципі «master-slave», тобто це модель взаємодії, у якій головний пристрій виконує керування одним чи групою підлеглих пристроїв. Взаємодія між пристроями відбувається за допомогою чотирьох ліній:

- MISO (Master In Slave Out) – передача даних від підлеглого пристрою до головного;
- MOSI (Master Out Slave In) – передача даних від головного до підлеглого пристрою;
- SCK (Serial Clock) – тактові імпульси для синхронізації передачі даних;
- SS (Slave Select) – вивід, який призначений для вибору головним пристроєм конкретного підлеглого [16].

Наведемо розпіновку контактів плати MFRC522 на рисунку 3.2, які необхідно під'єднати до контролера.

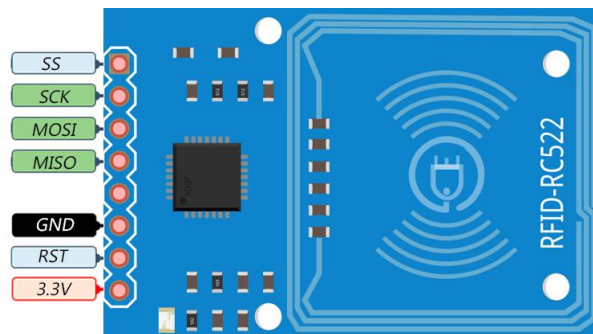


Рисунок 3.2 – Необхідні контакти модулю

Таким чином, підключення модулю буде виконуватися за способом, що наведено у таблиці 3.1. При підключенні використовуються контакти з підтримкою інтерфейсу SPI, а також контакти для виводу SS та RST.

Таблиця 3.1 – Підключення модулю

Плата NodeMCU	Модуль RFID-RC522
D4	SS
D5	SCK
D7	MOSI
D6	MISO
GND	GND
D3	RST
3.3V	3.3V

Остаточне з'єднання модулю з платою зображено на рисунку 3.3.

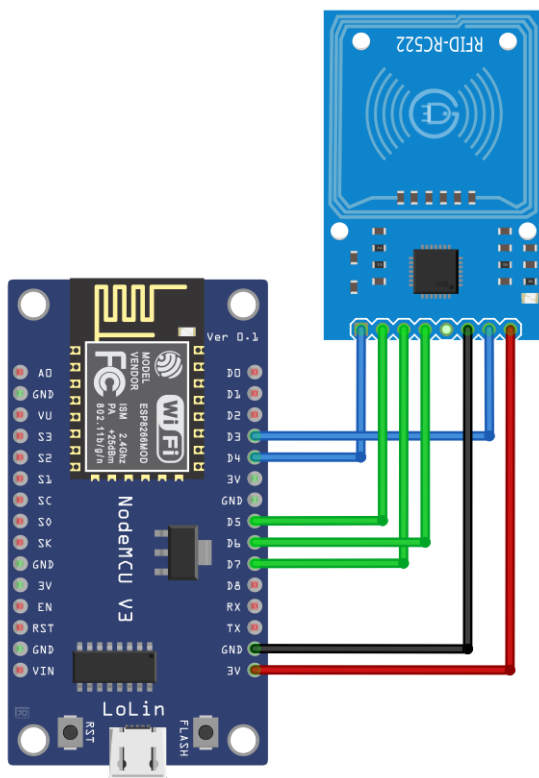


Рисунок 3.3 – Підключення модулю до плати

3.2 Розробка функціонального вузла визначення температури

Далі розглянемо процес підключення інфрачервоного датчика температури MLX90614. На відміну від попереднього модуля, цей датчик використовує інтерфейс I²C для зв'язку з контролером. Він також працює за принципом «master-slave», але для роботи цьому інтерфейсу необхідно лише дві двонаправлені лінії зв'язку – це SDA, що використовується для послідовної передачі даних, а також SCL – лінія синхронізації.

Отже, наведемо розпіновку датчику на рисунку 3.4.

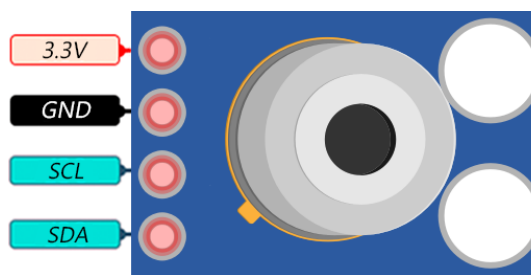


Рисунок 3.4 – Підключення модулю до плати

У таблиці 3.2 наведено з'єднання контактів датчику з платою. Для цього використовується виключно інтерфейс I²C і ніяких інших контактів для виводу не потрібно.

Таблиця 3.2 – Підключення модулю

Плата NodeMCU	Модуль MLX90614
D2	SDA
D1	SCL
GND	GND
3.3V	3.3V

З'єднання датчику з платою зображено на рисунку 3.5.

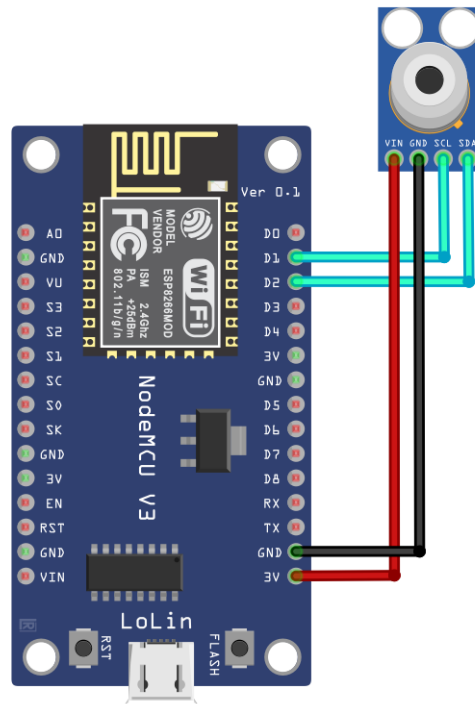


Рисунок 3.5 – Підключення датчика до плати

3.3 Розробка функціонального вузла дезінфекції рук

Наступним кроком є налаштування роботи водяної помпи. Через те, що помпа є, по суті, електродвигуном постійного току, тому для керування нею необхідно використовувати реле. Необхідність використання обумовлюється тим, реле захищає плату від перенавантажень.

Реле складається з двох ланцюгів: керуючого та керованого. В якості керуючого ланцюга в схемі виступає струм з плати, який виконує управління струмом із зовнішнього джерела живлення на 12 В. Такий підхід дозволяє контролювати значний струм за допомогою слабкого джерела.

З метою керування помпою у схемі буде використовуватись контакт S1, який підключається до біполярного транзистора, а вже потім йде на реле. Схема підключення наведена на рисунку 3.6.

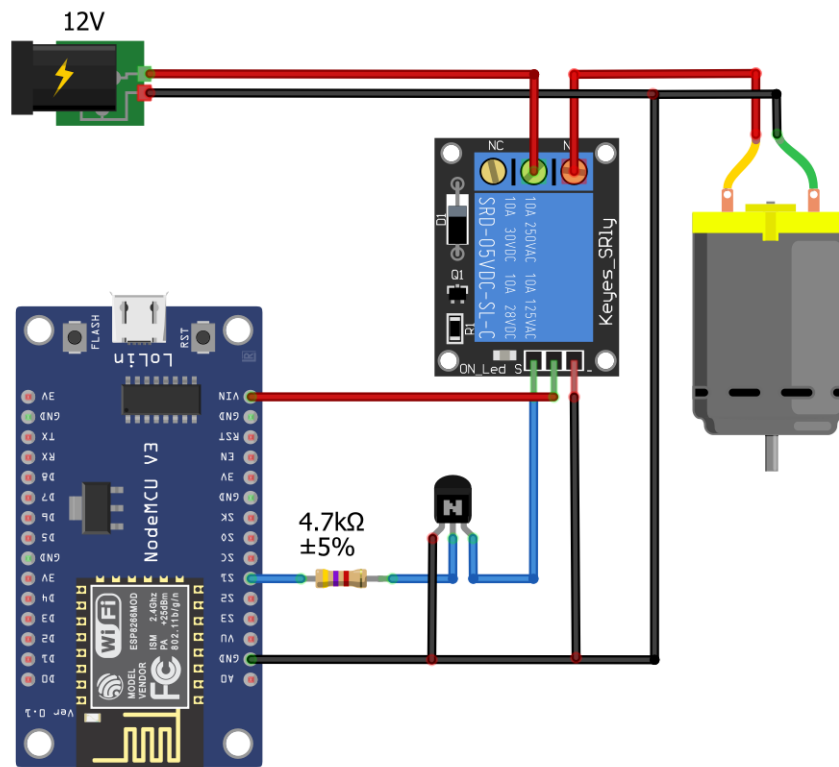


Рисунок 3.6 – Підключення помпи

3.4 Розробка загальної схеми пристрою

Перед завершальним етапом збірки остаточної схеми пристрою розглянемо підключення додаткових модулів системи, а саме інфрачервоного датчика перешкод та RGB світлодіода. Також на цьому етапі підключаємо контакт для прокидання системи від глибокого сну. Для цього необхідно з'єднати контакт D0 з RST, ІЧ-датчик підключається до контакту S0, світлодіод використовує контакт S2 для зеленого кольору, а S3 – для червоного. Також, при підключенні світлодіода необхідно використовувати резистори для запобігання перегорання діода. Схема підключення наведена на рисунку 3.7.

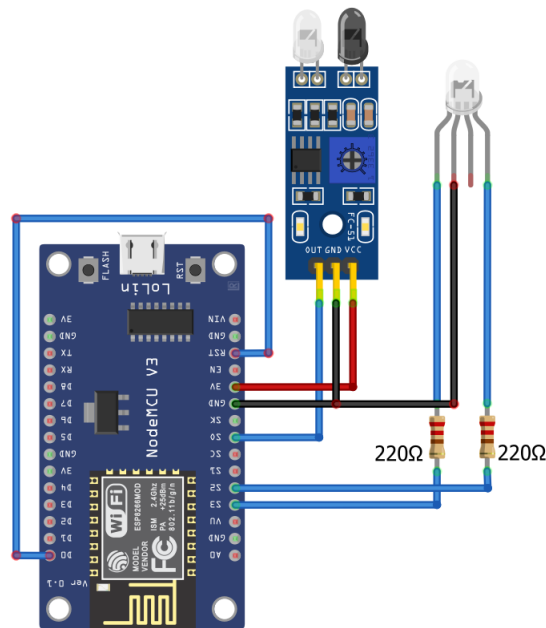


Рисунок 3.7 – Підключення додаткових модулів

Розібравшись з підключенням усіх компонентів системи до контролера, можна приступати до кінцевого етапу налаштування схеми. На цьому етапі необхідно з'єднати усі розроблені раніше функціональні вузли в єдиний пристрій. Загальна схема пристрою наведена на рисунку 3.8.

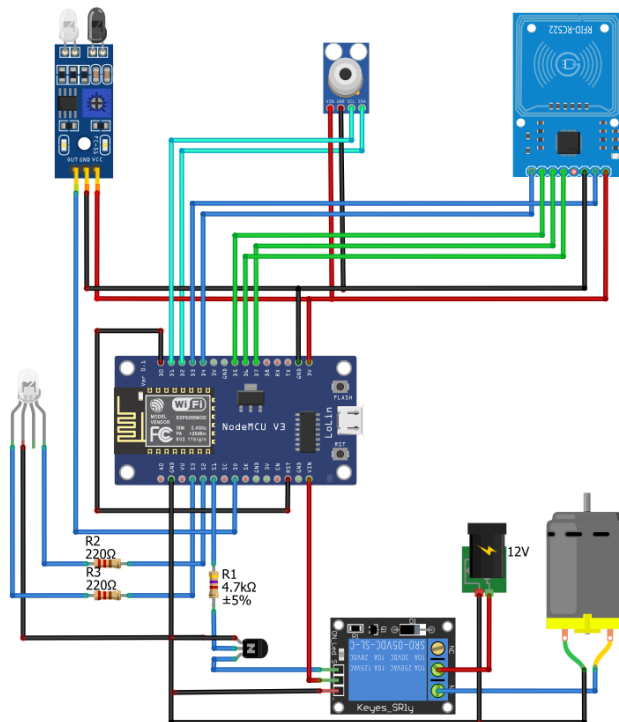


Рисунок 3.8 – Загальна схема пристрою

3.5 Тестування схеми пристрою

Завершаючий етап розробки складається з тестування розроблюваної системи. Для виконання цього етапу необхідно розробити схему пристрою у середовищі моделювання Proteus VSM та поєднати її із зовнішньою частиною системи.

Модель пристрою у Proteus VSM буде поєднувати у собі усі функціональні вузли, окрім радіочастотного зчитувача та Wi-Fi модуля системи. Пов'язано це з тим, що бібліотек, які б реалізовували цей функціонал не існує, тому будуть використовуватись реальні компоненти, які використовуватимуть віртуальний послідовний порт для взаємодії з моделлю.

Загальна схема пристрою у середовищі моделювання Proteus VSM зображена на рисунку 3.9

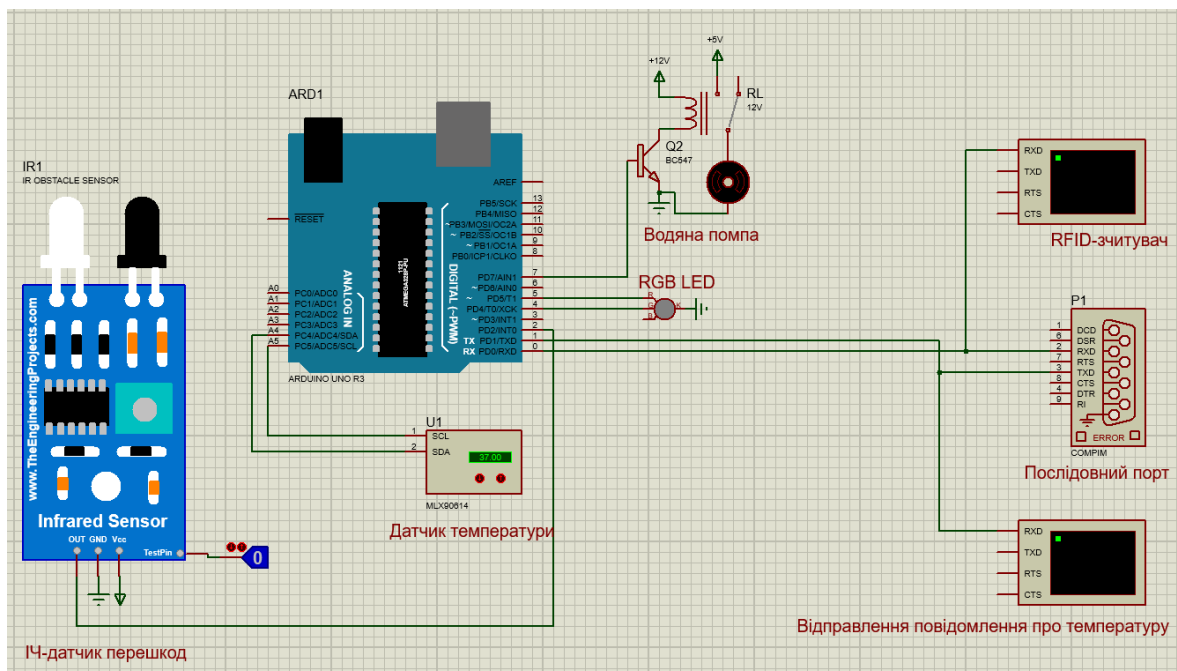


Рисунок 3.9 – Загальна схема пристрою

Для коректної взаємодії віртуальний послідовний порт необхідно налаштувати на роботу з фізичним портом по UART інтерфейсу, до якого

під'єднана реальна плата NodeMCU. У даному випадку це порт COM3 зі швидкістю передачі даних у 9600 біт/сек.

Таким чином, через віртуальний порт буде надходити унікальний ідентифікатор користувача, який зчитується підключеним до фізичної плати модулем, потім модель отримує з датчика температуру і повертає її через послідовний порт назад до плати, де і виконується уся взаємодія системи з сервером.

Також необхідно налаштувати модель контролеру Arduino та інфрачервоного датчика. Виконується це за допомогою вказання шляху до скомпільованого скетчу, який вказується у полі «Program File» в меню налаштування приладів.

Для початку тестування необхідно увімкнути сервер та під'єднати плату до необхідного порту комп'ютеру, після чого вже запускати симуляцію у Proteus VSM.

Після піднесення мітки відбувається перевірка прав користувача на проходження, якщо вони підтверджуються, тоді у порт відправляється повідомлення «TRUE», яке запускає роботу системи у Proteus VSM. На рисунку 3.10 зображено термінал з отриманими повідомленнями.

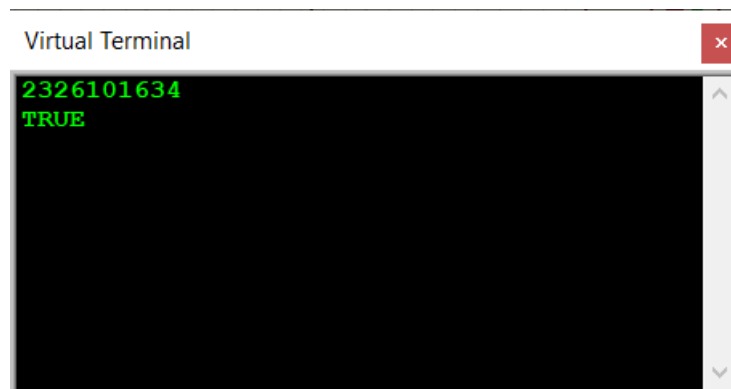


Рисунок 3.10 – Вікно терміналу RFID-зчитувача

Якщо на інфрачервоний датчик передається логічний сигнал «1», тоді це відповідає піднесенню рук до нього та запуску процесу зчитування температури та дезінфекції рук.

На рисунку 3.11 зображено активний стан роботи інфрачервоного датчику.

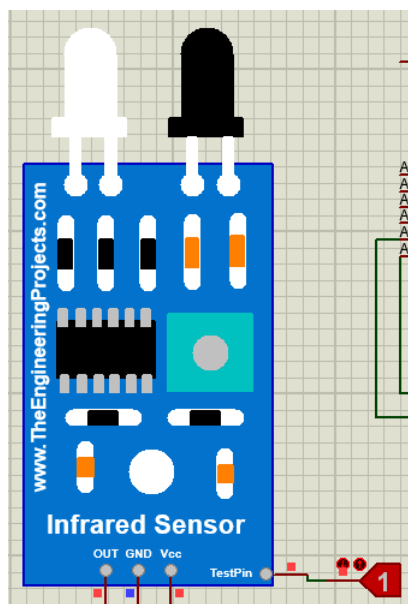


Рисунок 3.11 – Активний стан роботи ІЧ-датчика

Наступним вмикається датчик температури, значення якого задаються за допомогою інтерфейсу. На рисунку 3.12 зображено цей датчик.

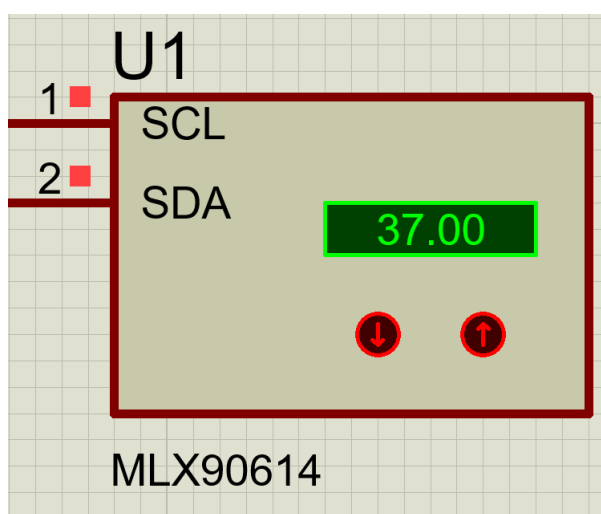


Рисунок 3.12 – Активний стан роботи ІЧ-термометра

На рисунку 3.13 зображено активний стан роботи водяної помпи та RGB світлодіода.



Рисунок 3.13 – Активний стан роботи помпи та ІЧ-датчика

Після завершення виконання усіх заданих процесів відбувається відправлення отриманої температури через віртуальний порт назад до реальної плати, де відбувається відправлення зібраної інформації на сервер. На рисунку 3.14 зображено термінал відправлених повідомлень.

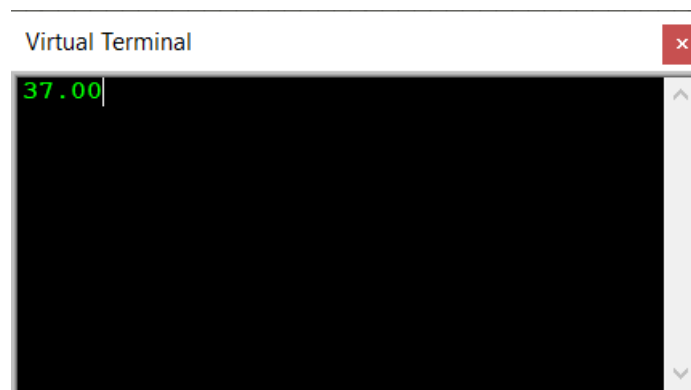


Рисунок 3.14 – Вікно терміналу відправлених повідомлень

Розглянемо далі взаємодію контролеру з веб-інтерфейсом. Контролер повинен мати можливість перевантажуватись та входити в режим глибокого сну при натисканні відповідних кнопок інтерфейсу.

Отже, при натисканні кнопки перевантаження у порт відправляється повідомлення про це, після чого відбувається перепідключення системи до Wi-Fi

мережі. На рисунку 3.15 зображено вікно з реакцією на натискання кнопки перевантаження.

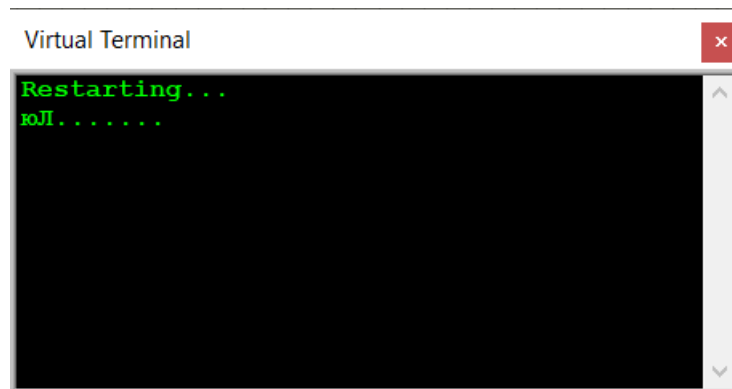


Рисунок 3.15 – Вікно терміналу з реакцією на перевантаження

На рисунку 3.16 зображено вікно з реакцією на натискання кнопки переходу контролеру у режим енергоощадження.

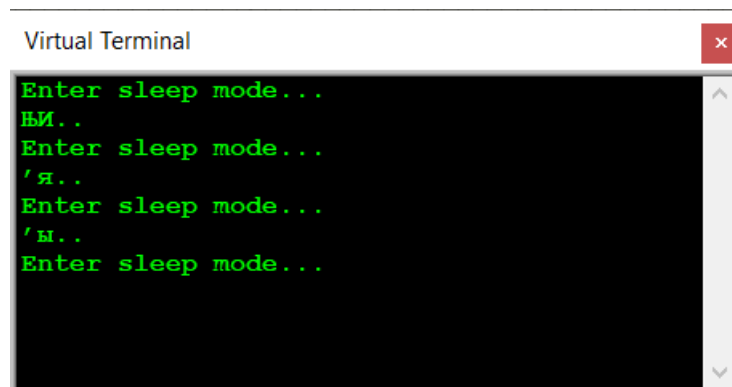


Рисунок 3.16 – Вікно терміналу з реакцією на перехід у режим глибоко сну

ВИСНОВКИ

В результаті виконання роботи було створено систему автоматичного доступу до аудиторії на основі платформи Arduino, яка поєднала у собі три основні функціональні частини, а саме: контроль доступу, профілактика захворювання і веб-інтерфейс.

Встановлена мета роботи з виконання розробки та моделювання пристрою була успішно досягнена. Було розроблено функціональні блоки, написано програмний код та складено схему кінцевого пристрою. При виконанні роботи було реалізовано усі необхідні функції для комфортного та автоматичного проходження контролю доступу та проведення профілактичних заходів. В додаток до цього, було розроблено веб-інтерфейс, який надав можливість виконувати обробку отриманих даних та керувати станом роботи розроблюваного пристрою.

Працездатність системи була перевірена за допомогою тестування у середовищі моделювання Proteus VSM. В результаті була перевірена коректність роботи основних модулів системи та правильність взаємодії пристрою з веб-додатком. Було визначено повну працездатність розробленої системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1 Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino / Улли Соммер., 2012. – 256 с.
- 2 A tour of the Arduino UNO board [Электронный ресурс] – Режим доступа до ресурсу: <https://www.hackerearth.com/blog/developers/a-tour-of-the-arduino-uno-board/>.
- 3 Arduino vs ESP8266 vs ESP32 Comparison [Электронный ресурс] – Режим доступа до ресурсу: <https://diyi0t.com/technical-datasheet-microcontroller-comparison/>.
- 4 Комплексный подход при идентификации личности [Электронный ресурс] – Режим доступа до ресурсу: http://conf.secuteck.ru/articles2/sys_ogr_dost/kompleksnyy-podhod-pri-identifikatsii-lichnosti/.
- 5 IoT сканер штрих-кода с отправкой результатов в облако [Электронный ресурс] – Режим доступа до ресурсу: <https://cxem.net/arduino/arduino257.php>.
- 6 RC522 Datasheet (PDF) - NXP Semiconductors [Электронный ресурс] – Режим доступа до ресурсу: <https://pdf1.alldatasheet.com/datasheet-pdf/view/227840/NXP/RC522.html>.
- 7 Датчики температуры Arduino [Электронный ресурс] – Режим доступа до ресурсу: <https://arduinomaster.ru/datchiki-arduino/podkluchenie-datchik-temperature-arduino/>.
- 8 Закон Стефана-Больцмана [Электронный ресурс] – Режим доступа до ресурсу: https://elementy.ru/trefil/21209/Zakon_StefanaBoltsmana.
- 9 A2TPMI Datasheet (PDF) - PerkinElmer Optoelectronics [Электронный ресурс] – Режим доступа до ресурсу: <https://pdf1.alldatasheet.com/datasheet-pdf/view/174009/PERKINELMER/A2TPMI.html>.
- 10 MLX90614 Datasheet (PDF) - List of Unclassified Manufacturers [Электронный ресурс] – Режим доступа до ресурсу: <https://pdf1.alldatasheet.com/datasheet-pdf/view/218975/ETC2/MLX90614.html>.

- 11 Миниатюрный водяной насос-помпа RS-360SH [Электронный ресурс] – Режим доступа до ресурсу: <https://3d-diy.ru/wiki/arduino-mechanics/miniatorynyj-vodyanoj-nasos-pompa-rs-360sh/>.
- 12 Что такое Apache [Электронный ресурс] – Режим доступа до ресурсу: <https://eternalhost.net/blog/hosting/web-server-apache>.
- 13 Denwer – Возможности проекта [Электронный ресурс] – Режим доступа до ресурсу: <http://www.denwer.ru/about.html>.
- 14 Веллинг Л. Разработка веб-приложений с помощью PHP и MySQL / Л. Веллинг, Л. Томсон., 2010. – 848 с.
- 15 ESP8266 Datasheet (PDF) Download - ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. [Электронный ресурс] – Режим доступа до ресурсу: <https://pdf1.alldatasheet.com/datasheet-pdf/download/1132995/ESPRESSIF/ESP8266.html>.
- 16 Болл С. Аналоговые интерфейсы микроконтроллеров / Болл Р. Стюарт., 2010. – 354 с.

ДОДАТОК А

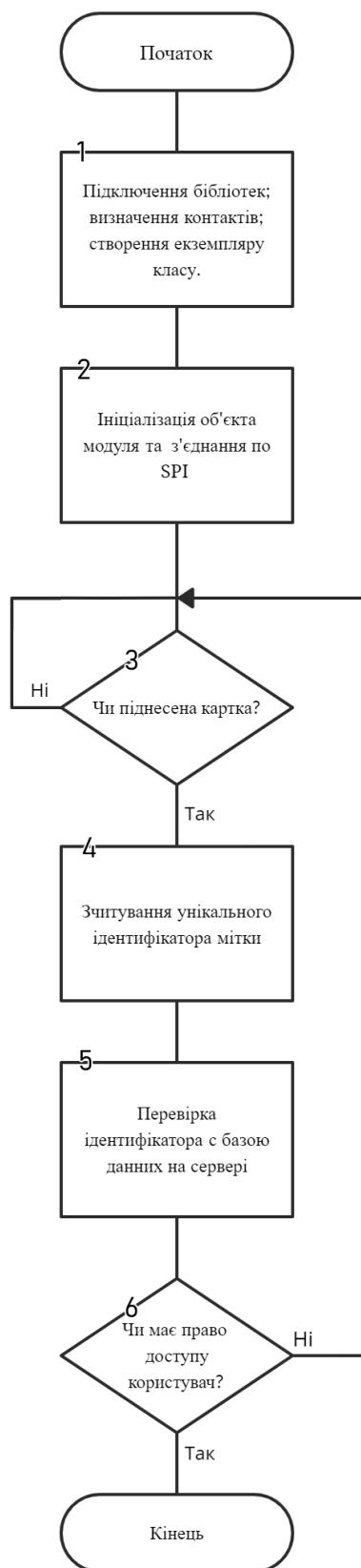


Рисунок А.1 – Алгоритм роботи системи ідентифікації



Рисунок А.2 – Алгоритм роботи системи контролю температури

ДОДАТОК Б

Лістинг Б.1 – Код основного мікроконтролеру

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN          D4
#define RST_PIN         D3

#define AUD_NUM          "101"

#define SERVER_IP        "192.168.0.10"
#define POST_TEMP_PAGE   "/postTemperature.php"
#define CHECK_USER_PAGE  "/checkUser.php"

#define CHECK_STATUS_PAGE "/checkStatus.php"
#define ENABLE_STATUS_PAGE "/enableStatus.php"

#define STASSID          "TP-Link_D776"
#define STAPSK           "54754225"

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance

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

    SPI.begin(); // Init SPI bus
    mfrc522.PCD_Init(); // Init MFRC522

    WiFi.begin(STASSID, STAPSK);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");

    postStatus(); //setstatus
}

void loop() {

    if (checkStatus("1")) { //power
        Serial.println("Enter sleep mode...");
        ESP.deepSleep(10 * 1000000);
    }
}
```

```

if (checkStatus("0")) { //reset
    Serial.println("Restarting...");
    ESP.restart();
}

unsigned int UID =  getUID();
if (UID != NULL) {
    Serial.println(UID);
    if (checkUser(UID)) { //Запуск скетча в протееyse
        Serial.print("TRUE");

        float temp = 0;
        do {
            temp = Serial.parseFloat();
            delay(50);
        } while (temp == 0);

        postTemp(UID, temp);
    } else
        Serial.print("FALSE");
}

delay(1000);
}

unsigned int getUID() {
    if (mfrc522.PICC_IsNewCardPresent()) {
        if (mfrc522.PICC_ReadCardSerial()) {
            mfrc522.PICC_HaltA();
            unsigned int UID;
            UID = mfrc522.uid.uidByte[0] << 24;
            UID += mfrc522.uid.uidByte[1] << 16;
            UID += mfrc522.uid.uidByte[2] << 8;
            UID += mfrc522.uid.uidByte[3];
            return  UID;
        }
    }
}

boolean checkUser(unsigned int UID) {
    if ((WiFi.status() == WL_CONNECTED)) {

        WiFiClient client;
        HTTPClient http;

        String url = "http://" SERVER_IP "/arduino" CHECK_USER_PAGE
"/?UID=" + String(UID) + "&aud=" AUD_NUM;
        http.begin(client, url); //HTTP

        int httpCode = http.GET();
    }
}

```

```

    if (httpCode == HTTP_CODE_OK) {
        const String& payload = http.getString();
        if (payload == "FOUND")
            return true;
        else
            return false;
        http.end();
    }
}

void postTemp(unsigned int UID, float temp) {
    if ((WiFi.status() == WL_CONNECTED)) {

        WiFiClient client;
        HTTPClient http;

        String url = "http://" SERVER_IP "/arduino" POST_TEMP_PAGE
        "/?UID=" + String(UID) + "&temp=" + temp + "&aud=" AUD_NUM;
        http.begin(client, url); //HTTP

        int httpCode = http.GET();
        if (httpCode == HTTP_CODE_OK)
            const String& payload = http.getString();
        http.end();

    }
}

boolean checkStatus(String code) {
    if ((WiFi.status() == WL_CONNECTED)) {

        WiFiClient client;
        HTTPClient http;

        String url = "http://" SERVER_IP "/arduino"
CHECK_STATUS_PAGE "/?code=" + code + "&aud=" AUD_NUM;
        http.begin(client, url); //HTTP

        int httpCode = http.GET();
        if (httpCode == HTTP_CODE_OK) {
            const String& payload = http.getString();
            if (payload == "TRUE")
                return true;
            else
                return false;
            http.end();
        }
    }
}

void postStatus() {

```

```

    if ((WiFi.status() == WL_CONNECTED)) {

        WiFiClient client;
        HTTPClient http;

        String url = "http://" + SERVER_IP + "/arduino"
ENABLE_STATUS_PAGE "?aud=" AUD_NUM;
        http.begin(client, url); //HTTP

        int httpCode = http.GET();
        http.end();
    }
}

```

Лістинг Б.2 – Код для програми Proteus VSM

```

#include <MLX90614.h>

MLX90614 temperature;

#define IR          2
#define PUMP        7

#define GREEN_PIN   4
#define RED_PIN     5

void setup() {
    Serial.begin(9600);
    pinMode(IR, INPUT);
    pinMode(PUMP, OUTPUT);

    pinMode(GREEN_PIN, OUTPUT);
    pinMode(RED_PIN, OUTPUT);
}

void loop() {
    if (Serial.available() > 0) {
        if (Serial.find("TRUE")) {
            digitalWrite(GREEN_PIN, HIGH);
            boolean IR_was_enabled = false;
            boolean get_temperature = false;
            float average_temperature = 0.0;

            uint32_t tmr = millis();
            do {
                if (digitalRead(IR) == HIGH) {
                    IR_was_enabled = true;

```

```

        if (!get_temperature) {
            for (int i = 0; i < 10; i++) {
                average_temperature += temperature.readTemperatureC()
/ 10;

                delay(50);
            }
            get_temperature = true;
        }
        if (digitalRead(PUMP) == LOW)
            delay(2000);
        digitalWrite(PUMP, HIGH); // Увімкнення помпи

    } else if (IR_was_enabled && digitalRead(IR) == LOW) {
        break;
    }

    delay(500);
} while (millis() - tmr <= 5000);

digitalWrite(PUMP, LOW); // Вимкнення помпи
digitalWrite(GREEN_PIN, LOW);

Serial.print(average_temperature);

} else if (Serial.find("FALSE")) {
    digitalWrite(RED_PIN, HIGH);
    delay(3000);
    digitalWrite(RED_PIN, LOW);
}

}

}

```

ДОДАТОК В

Лістинг В.1 – Файл checkStatus.php

```
<?php
$code = $_GET['code'];
$aud = $_GET['aud'];

if (isset($code)) {
    $link = mysqli_connect("localhost", "login", "password",
"arduino");
    if ($code == 0) {
        $sql = 'SELECT `reset_status` FROM `controller_status` WHERE
`name` = "'. $aud. "'';
        $result = mysqli_query($link, $sql);
        $row = mysqli_fetch_array($result);
        if ($row['reset_status'] == "status_off") {
            print("TRUE");
        }
    } elseif ($code == 1) {
        $sql = 'SELECT `power_status` FROM `controller_status` WHERE
`name` = "'. $aud. "'';
        $result = mysqli_query($link, $sql);
        $row = mysqli_fetch_array($result);
        if ($row['power_status'] == "status_off") {
            print("TRUE");
        }
    }

    $sql_online = 'UPDATE `controller_status` SET `online_status` =
"online", `update_time` = NOW() WHERE `name` = "'. $aud. "'';
    mysqli_query($link, $sql_online);
}

?>
```

Лістинг В.2 – Файл enableStatus.php

```
<?php
$name = $_GET['aud'];
$link = mysqli_connect("localhost", "login", "password",
"arduino");
$sql = 'UPDATE `controller_status` SET `reset_status`="status_on"
WHERE name = '.$name;
mysqli_query($link, $sql);
?>
```

Лістинг В.3 – Файл checkUser.php

```
<?php
$UID = $_GET['UID'];
$aud = $_GET['aud'];

$link = mysqli_connect("localhost", "login", "password",
"arduino");

$sql = 'SELECT * FROM `aud` WHERE `UID` = "'. $UID. '" AND `aud` =
"' . $aud. '"';
$result = mysqli_query($link, $sql);
if (mysqli_num_rows($result) > 0) {
    print("FOUND");
} else {
    print("NOT FOUND");
}
?>
```

Лістинг В.4 – Файл postTemperature.php

```
<?php
$UID = $_GET['UID'];
$temp = $_GET['temp'];
$aud = $_GET['aud'];

$link = mysqli_connect("localhost", "login", "password",
"arduino");

$sql = 'INSERT INTO `user_temp`(`UID`, `temp`, `aud`) VALUES
("' . $UID. '", "' . $temp. '", "' . $aud. '")';
$result = mysqli_query($link, $sql);
if ($result) {
    print("Temperature posted");
} else {
    print("An error occurred while executing of the request");
}
?>
```

ДОДАТОК Г

Лістинг Г.1 – Файл index.php

```
<?
include 'functions/switchPage.php';
if(isset($_POST['submit'])){
    $link = mysqli_connect("localhost", "login", "password",
"arduino");

    $login = $_POST['login'];
    $sql = 'SELECT password FROM logins WHERE login = "'. $login. "'';
    $result = mysqli_query($link, $sql);
    $data = mysqli_fetch_assoc($result);

    if($data['password'] === md5(md5($_POST['pass']))) {
        session_start();
        $_SESSION["login"] = $login;
        redirect("'selectAction.php'");
    } else {
        print("You entered the wrong login/password");
    }
}
?>

<head>
<link rel="stylesheet" href="styles/default.css">
</head>
<H1> Sign In </H1>
<div>
<form method="POST">
    Login <input name="login" type="text" required><br>
    Password <input name="pass" type="password" required><br>
    <input class="default" name="submit" type="submit"
value="Sign in">
</form>
</div>
```

Лістинг Г.2 – Файл selectAction.php

```
<?
include 'functions/switchPage.php';
include 'functions/buttons.php';
session_start();

function checkInputRights($right) {
    $link = mysqli_connect("localhost", "login", "password",
"arduino");
    $sql = 'SELECT RightId FROM logins WHERE login =
"' . $_SESSION["login"] . "'';
```

```

$result = mysqli_query($link, $sql);
$data = mysqli_fetch_array($result);

$permission = stripos($data['RightId'], $right);
if($permission !== false) {
    echo 'enabled';
} else {
    echo 'disabled';
}
}

?>
<head>
<link rel="stylesheet" href="styles/default.css">
</head>
<H1> Select Action </H1>
<?

if(isset($_SESSION["login"])){

if (isset($_POST['table'])) {
    redirect("'table.php'");
} else if (isset($_POST['contoller'])) {
    redirect("'manageControllers.php'");
} else if (isset($_POST['user'])) {
    redirect("'manageUsers.php'");
}

?>
<div>
    <form method="POST">
        <input class="default" name="table" type="submit"
value="View tables" <?php checkInputRights("t") ?>>
        <input class="default" name="contoller" type="submit"
value="Manage contollers" <?php checkInputRights("c") ?>>
        <input class="default" name="user" type="submit"
value="Manage users" <?php checkInputRights("u") ?>>
    </form>
    <?
    exitSession();
    echo "</div>";
} else {
    echo "<div>";
    echo "Access denied";
    echo "</div>";
}

?>

```

Лістинг Г.3 – Файл table.php

```

<?
include 'functions/switchPage.php';
include 'functions/buttons.php';
session_start();

function createSelect($title, $select_name, $sql_arr,
$sql_row){
    echo " ".$title." :
    <select name='".$select_name."'>";
    while ($result = mysqli_fetch_array($sql_arr)) {
        echo "<option value='" . $result[$sql_row] . "'>" .
$result[$sql_row] . "</option>";
    }
    echo "</select>";
}

?>
<head>
<link rel="stylesheet" href="styles/default.css">
<link rel="stylesheet" href="styles/table.css">
</head>
<H1> Table </H1>
<?php

if(isset($_SESSION["login"])){

    $link = mysqli_connect("localhost", "login", "password",
"arduino");

    echo "<div>";
    $selectedDate = $_POST['Set'];
    if(isset($selectedDate)){

        $sql_table = 'SELECT user.name, user_temp.temp FROM
user_temp JOIN user ON user.UID = user_temp.UID WHERE aud =
' . $_POST['audList'] . ' AND DATE(date) = "' . $_POST['dateList'] . '" AND
user.group = "' . $_POST['groupList'] . '" ORDER BY date desc';

        echo '<table border="1">
        <thead>
        <tr>
        <th>Name</th>
        <th>Temperature</th>
        </tr>
        </thead>
        <tbody>';

        $arr_table = mysqli_query($link, $sql_table);
        while($res_table = mysqli_fetch_array($arr_table)){
            echo '<tr>
                <td>' . $res_table['name'] . '</td>

```

```

        <td>'.'.$res_table['temp'].'</td>
    </tr>';
    }

    echo '</tbody>
    </table>';
}

$sql_group = 'SELECT DISTINCT `group` FROM `user` WHERE
`group` != ""';
$arr_group = mysqli_query($link, $sql_group);

$sql_aud = 'SELECT aud FROM aud JOIN logins ON aud.UID =
logins.UID WHERE logins.login = "' . $_SESSION["login"] . "'';
$arr_aud = mysqli_query($link, $sql_aud);

$sql_date = 'SELECT DISTINCT DATE(date) FROM user_temp';
$arr_date = mysqli_query($link, $sql_date);

echo '<form method="POST">';
createSelect('Group', 'groupList', $arr_group, 'group');
createSelect('Auditorium', 'audList', $arr_aud, 'aud');
createSelect('Date', 'dateList', $arr_date, 'DATE(date)');
echo '<input class="default" type="submit" name="Set"
value="Send"/>
</form>';

returnBack();
exitSession();

echo "</div>";
} else{
echo "<div>
    Access denied
    </div>";
}
?>

```

Лістинг Г.4 – Файл manageControllers.php

```

<?php
include 'functions/switchPage.php';
include 'functions/buttons.php';
include 'functions/checkRight.php';
session_start();

function control_btns($link, $name, $status_r, $status_p){

    if (isset($_POST['reset-'.$.$name])) {
        $sql_update = 'UPDATE `controller_status` SET
`reset_status`="status_off" WHERE name = '.$.$name;
        mysqli_query($link, $sql_update);
    }
}

```

```

    }

    if (isset($_POST['power-'. $name])) {
        if ($status_p == "status_off") {
            $sql_update = 'UPDATE `controller_status` SET
`power_status` = "status_on" WHERE name = ' . $name;
        } else {
            $sql_update = 'UPDATE `controller_status` SET
`power_status` = "status_off" WHERE name = ' . $name;
        }
        mysqli_query($link, $sql_update);
    }
}

?>
<meta http-equiv='refresh' content='1'>
<head>
<link rel="stylesheet" href="styles/default.css">
</head>
<H1> Manage Controllers </H1>
<?php

if(isset($_SESSION["login"]) && checkRight("c")){

    $link = mysqli_connect("localhost", "login", "password",
"arduino");
    $sql = 'SELECT * FROM controller_status';
    $result = mysqli_query($link, $sql);
    while($row = mysqli_fetch_array($result, MYSQLI_NUM)) {
        $name[] = $row[0];
        $status_r[] = $row[1];
        $status_p[] = $row[2];
        $status[] = $row[3];
    }

    for ($i=0; $i < sizeof($name); $i++) {
        control_btns($link, $name[$i], $status_r[$i],
$status_p[$i]);
    }

    echo '<div>
<span style="font-size: 40px;">Controller name:</span>';

    for ($i=0; $i < sizeof($name) ; $i++) {
        if($status[$i] == "offline"){
            $disabled = "disabled";
        }
        echo '<form method="POST" style="padding: 40px;
background-color:#DFDFDF; border-radius: 32px; height: 70px">
        <input class="'. $status[$i].'" name="online-'. $name[$i].'"
type="submit" value="" disabled >

```

```

        <p style="font-size: 40px; margin-top: 10px; margin-left:
40px; display: inline-block">#'. $name[$i]. '</p>
        <input '.$disabled.' class="'. $status_r[$i]. '"
name="reset-'. $name[$i]. '" type="submit" value="R" style="float:
right; margin-left: 30px" >
        <input '.$disabled.' class="'. $status_p[$i]. '"
name="power-'. $name[$i]. '" type="submit" value="P" style="float:
right">
        </form>';
    }

    returnBack();
    exitSession();
    echo "</div>";
} else {
    echo "<div>";
    echo "Access denied";
    echo "</div>";
}
?>

```

Лістинг Г.5 – Файл manageUsers.php

```

<?php
include 'functions/switchPage.php';
include 'functions/buttons.php';
include 'functions/checkRight.php';
session_start();

?>
<head>
<link rel="stylesheet" href="styles/default.css">
</head>

<?

function setUserData($link, $query){
if(isset($_POST['submit'])){
    if (isset($_POST['t'])) {
        $right_t = "T";
    }
    if (isset($_POST['c'])) {
        $right_c = "C";
    }
    if (isset($_POST['u'])) {
        $right_u = "U";
    }
    $rights = $right_t.$right_c.$right_u;

    $password = md5(md5(trim($_POST['pass'])));
    $login = $_POST['login'];
    $UID = $_POST['UID'];
}

```

```

        if ($query == "Add") {
            $sql = 'INSERT INTO `logins` (`RightId`, `UID`,
`login`, `password`) VALUES ("'. $rights. '", "'. $UID. '",
"'. $login. '", "'. $password. '")';
        } elseif ($query == "Update") {
            $sql = 'UPDATE `logins` SET `RightId` =
"'. $rights. '", `password` = "'. $password. '" WHERE `login` =
"'. $login. '""';
        }
        mysqli_query($link, $sql);
    }
    ?>
<H1> <?echo $query;?> Menu </H1>
<div>
    <form method="POST">
        <?if ($query == "Add") {
            echo 'UID <input name="UID" type="text"
required><br>';
        }?>
        Login <input name="login" type="text" required><br>
        Password <input name="pass" type="text" required><br>
        <input type="checkbox" name="t"> table
        <input type="checkbox" name="c"> controller
        <input type="checkbox" name="u"> user
        <input class="default" name="submit" type="submit"
value="<?echo $query;?> user" style="margin-top: 30px">
    </form>
    <?
    userBackBtn();
    echo "</div>";
}

function deleteUser($link){
    if (isset($_POST['submit'])) {
        $login = $_POST['login'];
        $sql = 'DELETE FROM `logins` WHERE `login` =
"'. $login. '""';
        mysqli_query($link, $sql);
    }
    ?>
<H1> Delete Menu </H1>
<div>
    <form method="POST">
        Login <input name="login" type="text"
required><br>
        <input class="default" name="submit"
type="submit" value="Drop user" style="margin-top: 30px">
    </form>
    <?
    userBackBtn();
    echo "</div>";
}

```

```

    }

    if(isset($_SESSION["login"]) && checkRight("u")){
        $link = mysqli_connect("localhost", "login",
"password", "arduino");
        if($_GET['action'] == "Add user") {
            setUserData($link, "Add");
        } elseif ($_GET['action'] == "Update user") {
            setUserData($link, "Update");
        } elseif ($_GET['action'] == "Delete user") {
            deleteUser($link);
        } else {
            ?>
            <H1> Action Menu </H1>
            <div>
                <form method="GET">
                    <input class="default" name="action"
type="submit" value="Add user">
                    <input class="default" name="action"
type="submit" value="Update user">
                    <input class="default" name="action"
type="submit" value="Delete user">
                </form>
                <?
                returnBack();
                exitSession();
            }
        } else {
            echo "<div>";
            echo "Access denied";
            echo "</div>";
        }
    }
    ?>

```

Лістинг Г.6 – Файл buttons.php

```

<?php
function exitSession(){
    echo '<form method="POST">
        <input class="default" name="exit" type="submit"
value="Sign out" style="background-color: #E7003E">
    </form>';

    $exit = $_POST['exit'];
    if(isset($exit)){
        session_unset();
        session_destroy();
        redirect("'index.php'");
    }
}

function returnBack(){
    echo '<form method="POST">

```

```

        <input class="default" name="return" type="submit"
value="Back" style="margin-bottom: -10px; background-color:
#949494">

        </form>';

        $return = $_POST['return'];
        if(isset($return)){
            redirect("'selectAction.php'");
        }
    }
    function userBackBtn(){
        echo '<form method="POST">
        <input class="default" name="back" type="submit"
value="Back" style="margin-top: -10px; background-color: #949494">
        </form>';
        $back = $_POST['back'];
        if(isset($back)){
            redirect("'manageUsers.php'");
        }
    }
?>

```

Лістинг Г.7 – Файл checkRight.php

```

<?php
function checkRight($right){
    $link = mysqli_connect("localhost", "login", "password",
"arduino");
    $sql = 'SELECT `RightId` FROM `logins` WHERE
`login`="' . $_SESSION["login"] . "'';
    $result = mysqli_query($link, $sql);
    $data = mysqli_fetch_array($result);

    $permission = stripos($data['RightId'], $right);
    if($permission !== false) {
        return true;
    }
}
?>

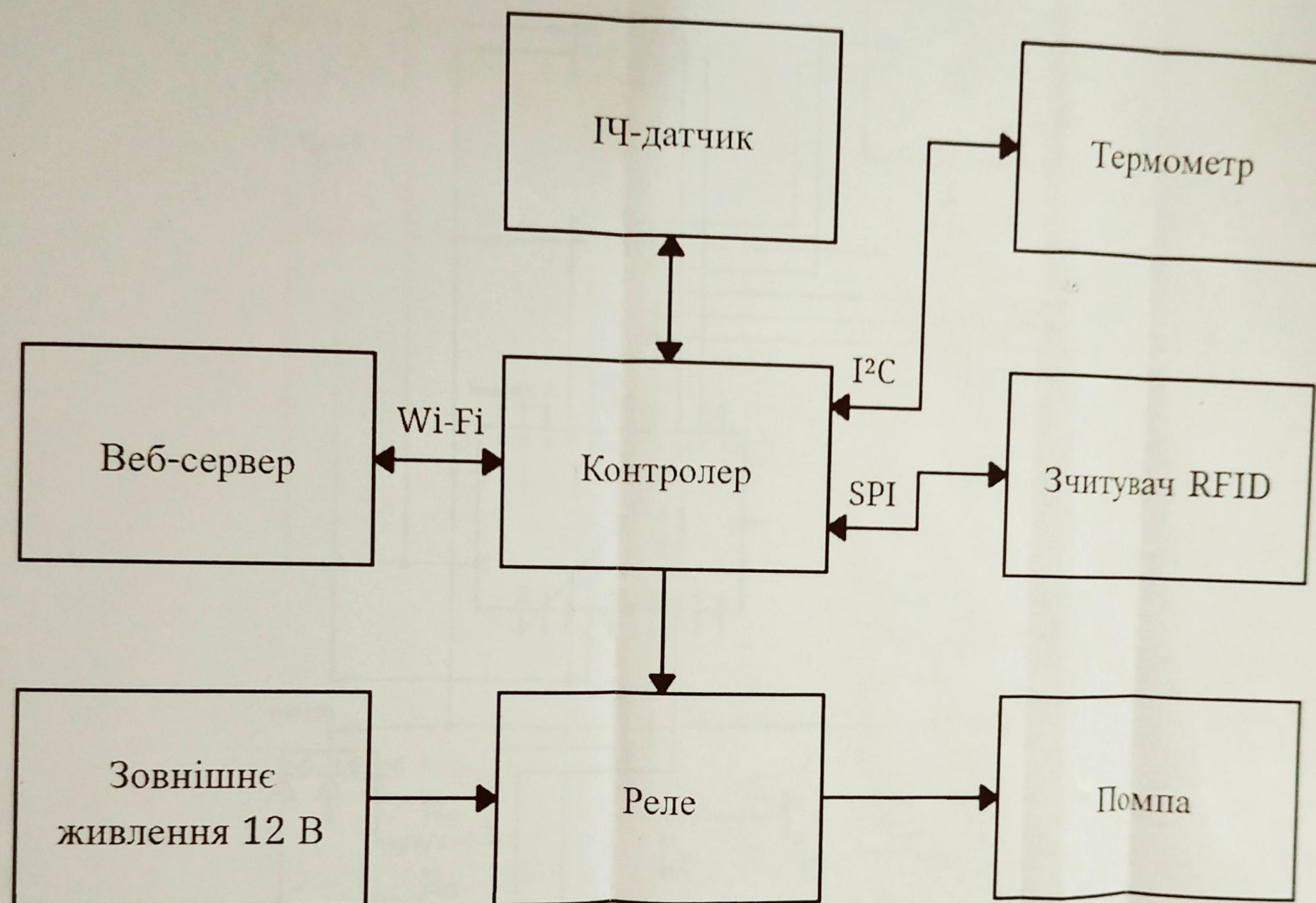
```

Лістинг Г.8 – Файл switchPage.php

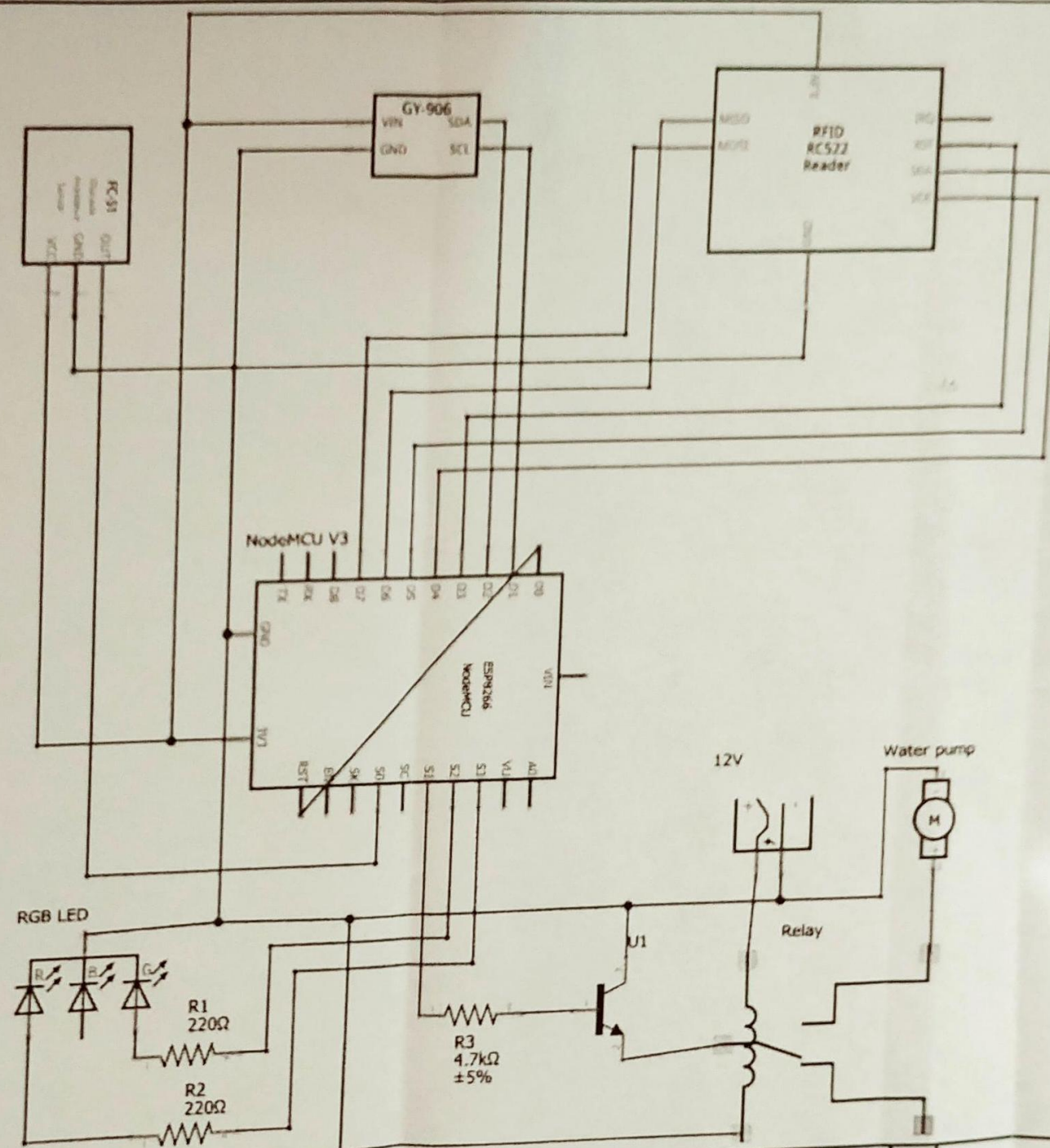
```

<?php
function redirect($link){
    echo "<script type='text/javascript'>
        window.location = ".$link.";
    </script>";
}
?>

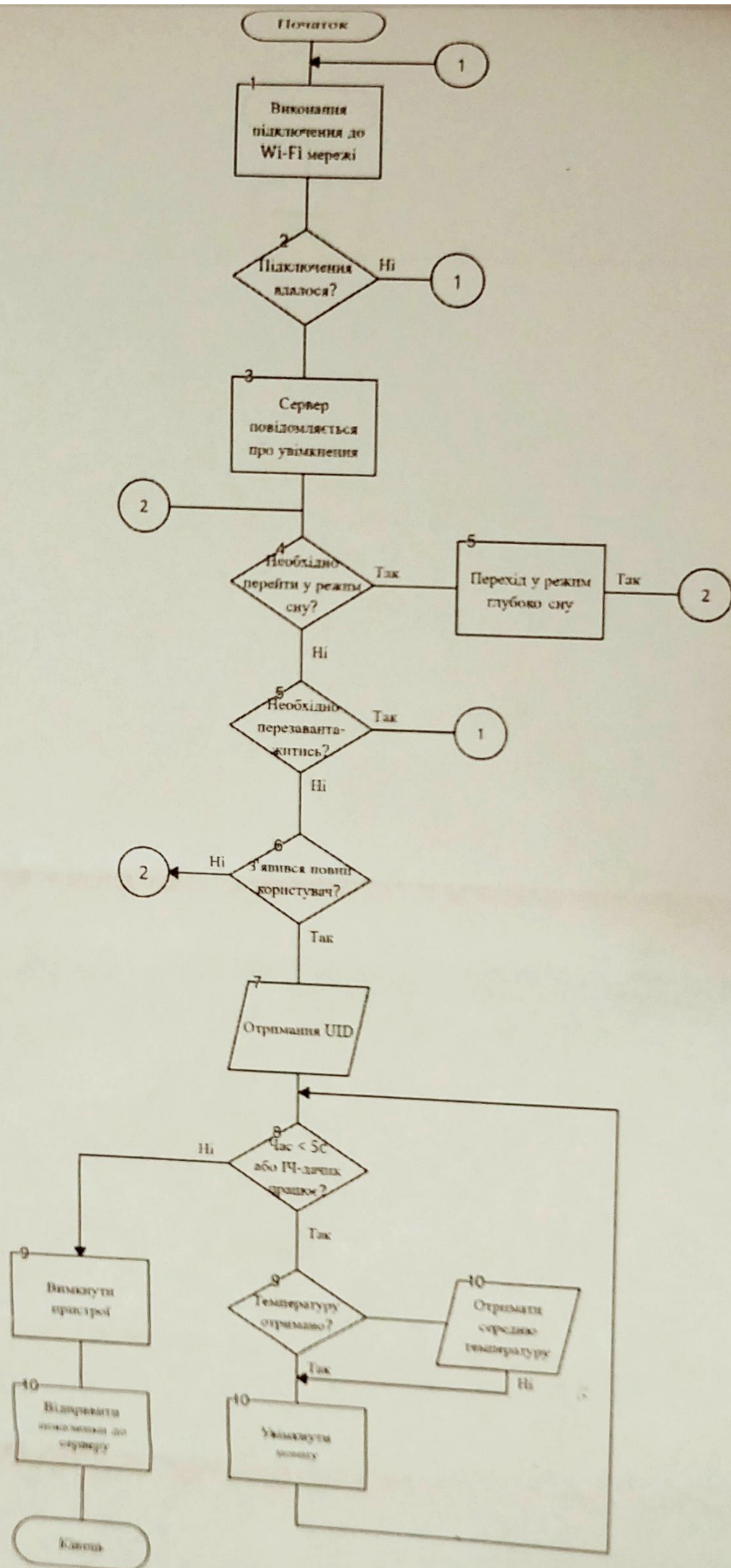
```



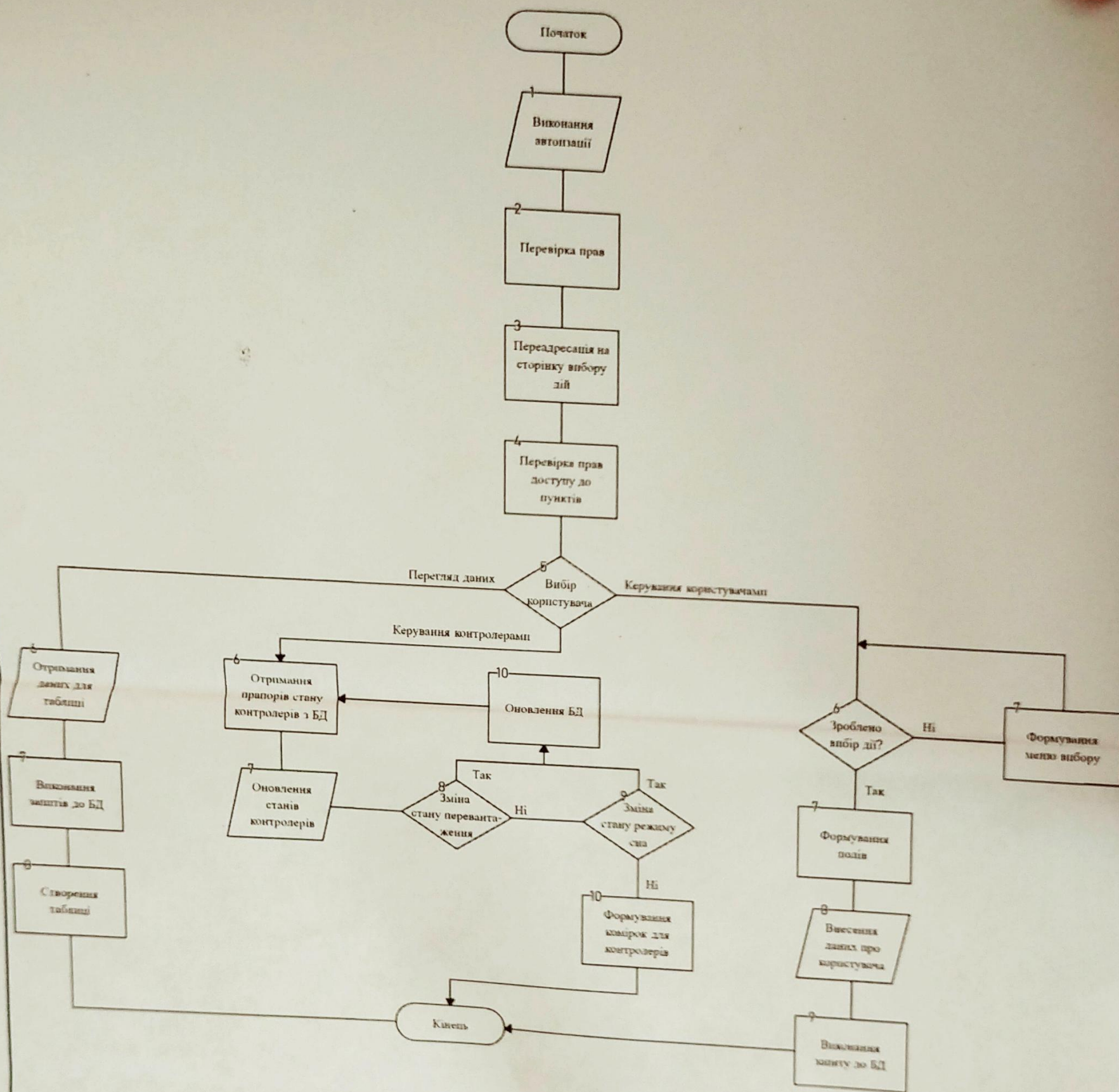
13.02070849 00033 E1				
Змін.	Арх.	№ документа	Підпис	Дата
		Костришин М.М.	<i>[Signature]</i>	
		Зеленцова І.Я.	<i>[Signature]</i>	
		Щербак Н.В.	<i>[Signature]</i>	
		Кудряшова Р.К.	<i>[Signature]</i>	
Розробка системи автоматичного доступу до аудиторії на основі Arduino. Структурна схема реалізації системи				
Лист		Місяць		Рік
Архив		Архив		Архив
Національний університет «Запорізька політехніка» КНТ-527				



13.02070849.00033 E3					Розробка системи автоматичного доступу до аудиторії на основі Arduino.		
Змін.	Арх.	№ документа	Підпис	Дата	Принципова схема пристрою		
Розробив		Костришин М.М.			Дат.		
Перевір.		Зеленцова І.Я.			Місяц		
Т. Контр.					Рік		
Н. Контр.		Щербак Н.В.			Арх. 1		
Утвердж.		Кулиничев Р.К.			Арх. 1		
					Національний університет «Запорізька політехніка»		
					КНТ-527		



13.02070849.00033 A1					
Вип.	Впр.	№ документа	Питом.	Внес.	Розробка системи заповнення даних до аудиторії на основі Алгоритму Алгоритм роботи пристрою
Варіант	Користувач	Варіант	Внес.	Внес.	
Варіант	Внес.	Внес.	Внес.	Внес.	
Варіант	Внес.	Внес.	Внес.	Внес.	
Варіант	Внес.	Внес.	Внес.	Внес.	
Національний університет «Дніпрова академія» 6367-329					



					13 02070849.00033 A1		
Вип.	Дзв.	№ документа	Підпис	Дата	Розробка системи автоматизованого доступу до аудиторій на основі Arduino. Алгоритм роботи веб-додатку		
Розробив	Асистент М.М.						
Проверив	Допоміжник І.Я.				Національний університет «Закарпатський національний університет імені Якова Галанича» КНУТ-ЗНУ		
П. Інженер							
М. Інженер	Міністр М.В.				Архив 1 Архив 1		
Завідувач	Бюджету І.К.						