БАКАЛАВРСЬКА РОБОТА

БР.КІ-13.00.00.000 ПЗ

Група КІ-21-1

Мацюк Іван

2025

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Інститут інформаційних технологій
Кафедра комп'ютерних систем і мереж

Мацюк Іван Васильович

УДК
004.738.5:004.056.5:004.383.6

БАКАЛАВРСЬКА РОБОТА

Розробка мікроконтролерної web-системи управління бар'єром на платформі ESP32 WiFi

Комп'ютерна інженерія
(назва освітньої програми)

123 - Комп'ютерна інженерія
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня *Мацюк І. В.*
(підпис, ініціали та прізвище здобувача)

Науковий керівник *Мойсеєнко О. В., к.т.н., доцент*
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри КСМ

д.т.н.,проф. *С.І. Мельничук*
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025 рік

Івано-Франківський національний технічний університет нафти і газу
(повне найменування вищого навчального закладу)

Інститут *інформаційних технологій*

Кафедра *комп'ютерних систем і мереж*

Освітній ступінь *бакалавр*

Спеціальність *123 – Комп'ютерна інженерія*
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КСМ
(С.І. Мельничук)
« 25 » травня 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Мацюку Івану Васильовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка мікроконтролерної web-системи управління бар'єром на платформі ESP32 WiFi

керівник проекту (роботи) Мойсеєнко Олена Володимирівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "05" травня 2025року№275/7

2. Строк подання студентом роботи 12 червня 2025р

3. Вихідні дані до роботи Методичні вказівки,технічна література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз існуючих систем для управління бар'єром на основі ESP32. 2. Розробка апаратних та програмних рішень інформаційного супроводу шахової дошки на базі мікроконтролера ESP-WROOM-32. 3. Розробка програмного забезпечення та перевірка працездатності розробленої системи.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання – 29 січня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання етапів роботи	Примітка
1	Аналіз існуючих систем для управління бар'єром на основі ESP32	08.03.2025 – 16.03.2025	виконано
2	Розробка та збір системи	17.03.2025 – 05.04.2025	виконано
3	Розробка програмного забезпечення та перевірка працездатності розробленої системи	06.04.2025 – 30.04.2025	виконано
4	Оформлення пояснювальної записки	01.05.2025 – 30.05.2025	виконано
5	Підготовка до здачі бакалаврської роботи	31.05.2025 – 07.06.2025	виконано

Студент _____ *Мацюк І.В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Мойсеєнко О.В.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У роботі розроблено мікроконтролерну web-систему для управління електромеханічним бар’єром з використанням плати ESP32. Система забезпечує створення автономної Wi-Fi точки доступу, авторизацію користувачів через веб-інтерфейс, поділ прав доступу між адміністраторами та звичайними користувачами, логування дій та збереження даних на SD-карті. Основною перевагою реалізованого рішення є автономність, простота у використанні, можливість розширення функціоналу та гнучкість у налаштуваннях.

Функціонування пристрою не потребує підключення до зовнішньої мережі Інтернет. Користувач отримує доступ до системи через IP-адресу, що надається ESP32, та проходить авторизацію. Після цього здійснюється відкриття шлагбаума за допомогою сервоприводу, перемикання світлодіодних індикаторів та запис події в log-файл. Адміністратор має додаткові можливості: перегляд журналу, додавання нових користувачів, контроль доступу. Усі дані зберігаються на microSD-карті, що забезпечує автономність і надійність.

Запропоновану систему протестовано в умовах, наближених до реальних, підтверджено її працездатність, стабільність та відповідність заданому алгоритму роботи.

Ключові слова: ESP32, керування шлагбаумом, Wi-Fi точка доступу, веб-інтерфейс, авторизація, журнал подій, SD-карта, сервопривід.

ABSTRACT

This work presents the development of a microcontroller-based web system for controlling an electromechanical barrier using the ESP32 board. The system operates as an autonomous Wi-Fi access point, providing user authentication via a web interface, differentiated access rights for administrators and regular users, action logging, and data storage on an SD card. The main advantages of the implemented solution include autonomy, ease of use, scalability, and flexible configuration.

The system functions without requiring an external Internet connection. A user accesses the system through the IP address provided by the ESP32 and completes authentication. Upon successful login, the system activates a servo to open the barrier, switches LED indicators, and logs the event to a file. Administrators have extended capabilities, such as viewing the log, adding new users, and managing access rights. All data is stored on a microSD card, ensuring system autonomy and reliability.

The proposed system was tested under conditions close to real operation. Its functionality, stability, and compliance with the specified algorithm were successfully confirmed.

Keywords: ESP32, barrier control, Wi-Fi access point, web interface, authentication, event logging, SD card, servo motor.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ESP32 – Енергозберігаючий мікроконтролер із вбудованим Wi-Fi та Bluetooth, що використовується для реалізації IoT-рішень.

Wi-Fi – Wireless Fidelity – бездротова технологія передачі даних, яка забезпечує зв’язок мікроконтролера з іншими пристроями.

LED – Light Emitting Diode – світлодіод, використовується для індикації станів системи.

SD-карта – Secure Digital Card – носій пам’яті для зберігання користувачів та журналів у розробленій системі.

GPIO – General Purpose Input/Output – універсальні порти введення/виведення, що використовуються на платі ESP32 для підключення зовнішніх пристроїв.

HTML – HyperText Markup Language – мова розмітки гіпертексту, що використовується для створення веб-інтерфейсу.

HTTP – HyperText Transfer Protocol – протокол передачі гіпертексту, який використовується веб-браузером для комунікації з веб-сервером ESP32.

IP-адреса – Ідентифікатор пристрою в мережі, за яким користувач може звернутись до веб-інтерфейсу системи.

IDE – Integrated Development Environment – інтегроване середовище розробки, яке використовується для програмування мікроконтролерів.

NTP – Network Time Protocol – протокол синхронізації системного часу через інтернет.

API – Application Programming Interface – інтерфейс прикладного програмування.

ВСТУП

Стрімкий розвиток мікроконтролерних технологій та бездротових засобів зв’язку відкрив нові можливості для створення доступних, функціональних та автономних систем автоматизації. Одним з актуальних напрямів є впровадження систем контролю доступу, які активно застосовуються у приватному секторі, житлово-комунальному господарстві, на підприємствах і в організаціях. Серед таких рішень бар’єрні системи відіграють важливу роль у регулюванні в’їзду транспортних засобів та забезпеченні безпеки.

Особливу увагу привертають рішення, побудовані на базі мікроконтролера ESP32, який має вбудований модуль Wi-Fi, підтримку веб-технологій та можливість взаємодії з зовнішніми накопичувачами. Застосування такого підходу дозволяє створювати системи, що не потребують підключення до Інтернету чи зовнішніх серверів, залишаючись автономними, простими в налаштуванні та зручними у використанні.

Актуальність обраної теми полягає у необхідності розробки недорогих і водночас ефективних рішень для контролю доступу, що забезпечують авторизацію користувачів, розмежування прав доступу, ведення журналу подій, а також збереження даних на SD-карті.

Об’єкт дослідження — автоматизовані системи контролю доступу на основі мікроконтролера.

Предмет дослідження — мікроконтролерна web-система управління бар’єром на основі плати ESP32 з функціоналом авторизації, адміністрування, логування дій та керування фізичними компонентами.

Мета роботи — розробити автономну систему керування електромеханічним бар’єром з веб-інтерфейсом, побудовану на базі ESP32, яка підтримує авторизацію користувачів, адміністрування, індикацію станів та ведення журналу подій із збереженням даних на SD-карті.

Завдання дослідження:

– провести аналіз існуючих систем керування доступом з web-інтерфейсами;

– здійснити вибір компонентів та розробити схему підключення;

– реалізувати програмну логіку на мові Arduino C++ з використанням середовища Arduino IDE;

– створити web-інтерфейс з підтримкою авторизації та адміністрування;

– реалізувати механізм збереження даних користувачів і логів на SD-карті;

– протестувати систему на предмет стабільності, функціональності та відповідності поставленим вимогам.

Методи дослідження. Під час дослідження застосовано методи аналізу та синтезу технічних рішень, моделювання структури системи, проєктування електричних схем, програмування з використанням мови C++ у середовищі Arduino IDE, а також експериментальне тестування функціональності розробленої системи. Верифікація роботи проводилась шляхом реального підключення ESP32 до фізичних компонентів і емпіричної перевірки працездатності програмного коду в умовах, близьких до експлуатаційних.

Практичне значення отриманих результатів. Результати дипломної роботи мають практичну цінність, оскільки реалізована система може бути застосована як готове технічне рішення для організації автономного доступу в офісних центрах, житлових дворах, автостоянках або на приватних територіях. Система не потребує зовнішньої інфраструктури, просто налаштовується, підтримує журналювання дій, що робить її зручною для малих підприємств та домашнього використання. Крім того, розроблений проект може бути використаний як приклад у навчальному процесі для студентів технічних спеціальностей.

1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ДЛЯ УПРАВЛІННЯ БАР'ЄРАМИ НА ОСНОВІ ESP32 ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Характеристика та функціональні вимоги до інформаційної системи керування доступом

Інформаційна система — це програмно-апаратний комплекс, що реалізує автоматизовану обробку інформації про події доступу, зберігання облікових даних користувачів, адміністрування та керування фізичним пристроєм (бар'єром) через веб-інтерфейс [1].

У межах бакалаврської роботи розроблена система відповідає таким ключовим характеристикам:

– Автономна робота. Система не потребує підключення до мережі Інтернет і функціонує як локальна точка доступу.

– Авторизація користувачів. Підтримується логін/парольна автентифікація з розмежуванням прав (користувач / адміністратор).

– Керування фізичним пристроєм. Відкриття шлагбаума здійснюється через сервопривід.

– Журналювання подій. Вся інформація про доступ зберігається у лог-файлі на SD-карті.

Таблиця 1.1 – Основні характеристики інформаційної системи

Параметр	Опис
Тип системи	Автономна web-система керування доступом
Платформа	ESP32 DevKit V1
Мова програмування	C++ (Arduino IDE)
Інтерфейс	Веб-інтерфейс через Wi-Fi точку доступу
Авторизація	Логін/пароль, ролі (користувач / адміністратор)
Апаратні компоненти	Серводвигун SG90, SD-карта, світлодіоди, макетна плата
Журналювання подій	Збереження у log.txt на microSD
Облік користувачів	Збереження у users.txt на microSD
Функціональність	Відкриття шлагбаума, ведення логів, адміністрування, автозакриття
Безпека	Веб-автентифікація, локальне зберігання даних, ізоляція доступу

Нормативна та технічна документація, що регламентує розробку У процесі проектування та реалізації інформаційної системи враховано положення таких нормативно-правових актів та технічної документації:

– Закон України «Про інформацію». Визначає поняття інформаційних систем, порядок і режими доступу до інформації [2].

– Закон України «Про захист персональних даних». Регламентує обробку облікових даних користувачів [3].

– Закон України «Про електронні комунікації». Визначає вимоги до використання бездротових комунікаційних засобів [4].

– Закон України «Про наукову і науково-технічну діяльність». Актуальний для використання результатів дослідження у наукових або навчальних цілях [5].

Методичні вказівки до виконання бакалаврської роботи. Визначають загальні вимоги до структури, змісту, оформлення дипломного проєкту.

1.2 Аналіз існуючих систем контролю доступу

Системи контролю доступу є невід’ємною частиною сучасної інфраструктури безпеки як у приватному, так і в громадському секторі. Основним завданням таких систем є регулювання входу/виїзду осіб або транспортних засобів через певні об’єкти, забезпечуючи обмеження доступу для неавторизованих користувачів.

До традиційних засобів контролю доступу належать механічні замки, електромагнітні замки, турнікети, а також бар’єри з електроприводом. З переходом до цифрових технологій дедалі більшого поширення набувають рішення з мікроконтролерами та вбудованими web-інтерфейсами.

Рішення на основі ESP32 є особливо популярними серед розробників, завдяки вбудованому Wi-Fi, високій продуктивності, доступній вартості та підтримці середовища Arduino. Приклади існуючих реалізацій систем управління бар’єрами включають:

Системи з RFID-ідентифікацією: користувачі мають RFID-картки, які зчитуються модулем, підключеним до ESP32. У разі збігу даних із базою – бар’єр відкривається. Такі системи зручні, але потребують зовнішніх зчитувачів [6]. На рисунку 1.1 подано приклад ESP32-системи з RFID-модулем і електрозамком, яка дозволяє автоматизувати контроль доступу через фізичний бар’єр:

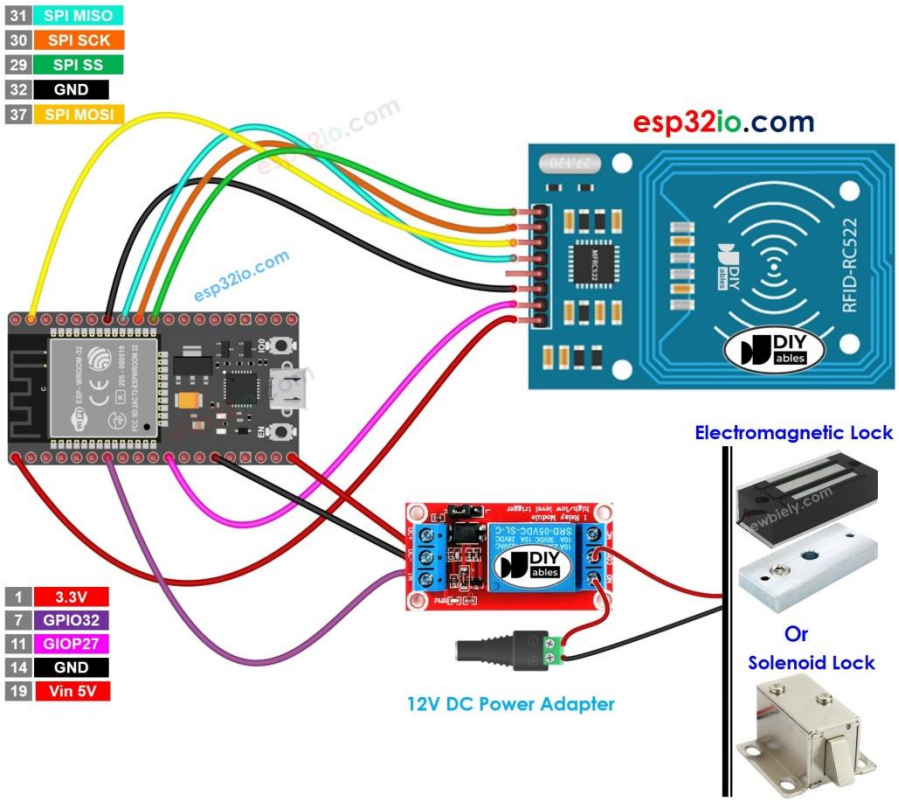

Рисунок 1.1 – Приклад системи контролю доступу з ESP32 і RFID-модулем

Системи з веб-інтерфейсом: ESP32 виступає як точка доступу або клієнт у Wi-Fi мережі, надаючи через браузер доступ до інтерфейсу авторизації. Після входу користувач отримує можливість керувати бар’єром [7]. Такі рішення підвищують зручність і не потребують фізичних ключів або карток. На рис. 1.2 подано приклад ESP32-системи з веб-інтерфейсом:

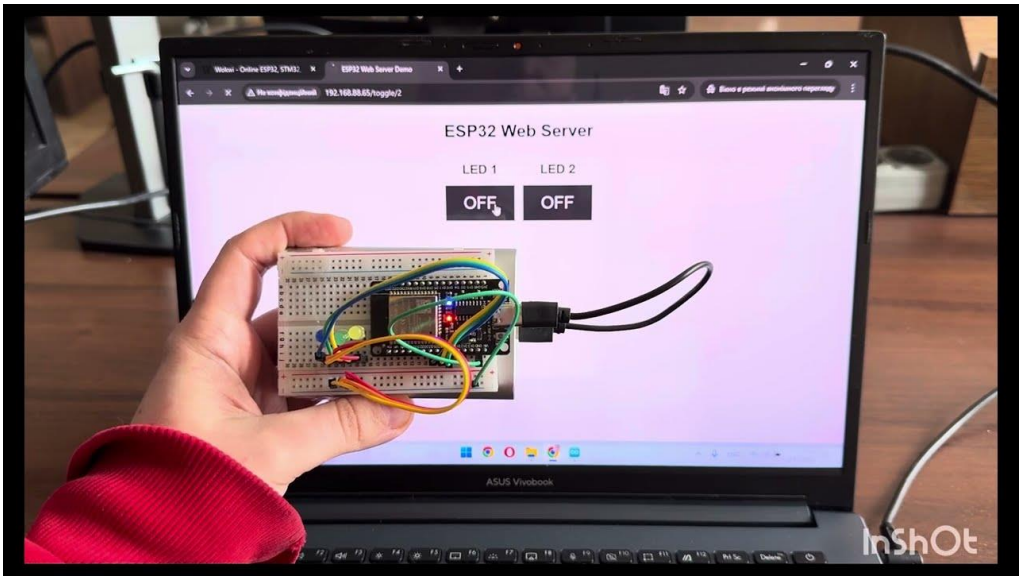

Рисунок 1.2 – Системи з веб-інтерфейсом на базі ESP32

Системи з мобільним керуванням: реалізуються через мобільні застосунки, які надсилають HTTP-запити до ESP32. Хоча такі варіанти зручні, вони складніші в реалізації та потребують підтримки застосунку [8].

Системи з Telegram-ботами або MQTT-протоколами: використовуються для віддаленого доступу, але потребують стабільного підключення до Інтернету та складнішої інфраструктури [9].

У загальному, тенденції розвитку систем керування бар'єрами спрямовані на мінімізацію використання фізичних ключів, підвищення гнучкості керування, забезпечення журналювання подій та віддаленого адміністрування. Водночас серед вимог — збереження безпеки, стійкість до несанкціонованого доступу, автономність та простота обслуговування.

Найбільш ефективними серед недорогих і універсальних рішень вважаються системи, які поєднують:

– автономну Wi-Fi точку доступу;

– веб-інтерфейс з авторизацією;

– локальне зберігання користувачів та журналів на SD-карті;

– можливість розширення прав доступу (адмін/користувач);

– простий електропривід (наприклад, сервомотор) для фізичного керування бар’єром.

1.3 Постановка задачі

У рамках розробки мікроконтролерної системи керування бар’єром на базі ESP32 було поставлено наступні задачі, які необхідно реалізувати для досягнення мети бакалаврської роботи:

Забезпечити доступ до системи через Wi-Fi без потреби у зовнішньому інтернет-з’єднанні, створивши автономну точку доступу на основі модуля ESP32.

Реалізувати веб-інтерфейс користувача, який дозволяє здійснювати вхід у систему, керувати шлагбаумом та взаємодіяти з іншими функціональними модулями.

Розробити механізм авторизації з підтримкою двох типів користувачів: звичайного та адміністратора. Адміністратор має розширені права доступу.

Інтегрувати фізичне керування бар’єром: відкриття/закриття за допомогою сервоприводу, індикацію стану за допомогою світлодіодів.

Забезпечити ведення журналу подій з фіксацією дати, часу та типу дії. Дані зберігаються у текстовому файлі на SD-карті.

Організувати облік користувачів шляхом зберігання їхніх облікових даних на SD-карті у форматі текстового файлу.

Реалізувати безпечне автоматичне закриття шлагбаума після певного інтервалу часу (30 секунд), у разі неактивності користувача або виходу із системи.

Впровадити логіку безпеки, що відключає доступ до керування у випадку помилкової авторизації, а також блокує інтерфейс у разі недоступності зовнішніх ресурсів (наприклад, SD-карти).

Створити адаптивну архітектуру, яка дозволяє в майбутньому додавати нових користувачів, змінювати права доступу, редагувати інтерфейс та

розширювати функціональні можливості без кардинального переписування коду.

Забезпечити сумісність із сучасними браузерами, що дозволить використовувати систему на будь-якому пристрої без додаткового програмного забезпечення на стороні клієнта.

Таким чином, сформульовані задачі охоплюють як апаратну, так і програмну частини, а також враховують вимоги до безпеки, зручності користування, автономності та масштабованості інформаційної системи.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ ДЛЯ СИСТЕМИ УПРАВЛІННЯ БАР'ЄРОМ НА ESP32

2.1 Розробка структурної схеми

Розробка структурної схеми є першим етапом створення мікроконтролерної web-системи керування бар'єром. Структурна схема дозволяє сформувати загальне уявлення про логіку функціонування системи та взаємозв'язок між її компонентами.

Структурна схема включає такі основні функціональні блоки:

– ESP32 – центральний обчислювальний елемент, який виконує функції Wi-Fi точки доступу, веб-сервера, обробника команд та контролера периферійних пристроїв;

– веб-інтерфейс – забезпечує взаємодію з користувачем через браузер, реалізуючи авторизацію, керування бар'єром, перегляд логів;

– SD-карта – використовується для збереження облікових даних користувачів та журналів подій;

– сервопривід – механізм, що фізично відкриває або закриває бар'єр за командою від ESP32;

– блок індикації – складається з червоного та зеленого світлодіодів для візуального відображення стану бар'єра;

– користувач – підключається до веб-інтерфейсу через Wi-Fi, виконує авторизацію та керує пристроєм.

На рисунку 2.1 представлено структурну схему системи управління бар'єром на ESP32.

Рисунок 2.1 – Структурна схема системи управління бар’єром на ESP32

Як видно зі схеми, основні інформаційні потоки сходяться до контролера ESP32, який приймає запити через Wi-Fi інтерфейс, виконує логіку перевірки прав доступу, а також фізично керує бар’єром і забезпечує збереження даних на карті пам’яті. У разі введення неправильних облікових даних система блокує доступ до функцій керування.

2.2. Вибір мікроконтролера та інших компонентів

Система, яка розробляється у межах даної дипломної роботи, — це мікроконтролерна web-система управління бар’єром. Основне її призначення — забезпечення автоматизованого відкриття та закриття бар’єра на основі авторизації користувача через Wi-Fi мережу.

Система повинна виконувати наступні функції:

– створення локальної точки доступу для з'єднання користувачів;

– надання веб-інтерфейсу для авторизації та керування;

– відкриття та автоматичне закриття бар’єра з використанням сервоприводу;

– відображення статусу системи за допомогою світлодіодної індикації;

– ведення журналу подій та управління обліковими записами користувачів через SD-карту.

Для реалізації цих функцій система потребує мікроконтролера з підтримкою Wi-Fi, достатньою кількістю GPIO-виводів, можливістю підключення SD-карти, а також підтримкою роботи в середовищі Arduino IDE.

Для реалізації даної системи було проаналізовано декілька мікроконтролерних платформ. Основні критерії порівняння: тактова частота, кількість доступних GPIO, підтримка Wi-Fi та Bluetooth, обсяг пам’яті, підтримка файлової системи та вартість системи.

В доступі є багато рішень, які задовільняють вище вказані потреби, але найпопулярнішими є ESP32 DevKit V1, NodeMCU ESP8266 та Arduino UNO.

ESP32 DevKit V1 (рис.2.2) — це компактна розробницька плата, створена компанією Espressif, яка базується на потужному мікроконтролері ESP32 [10]. Вона має:

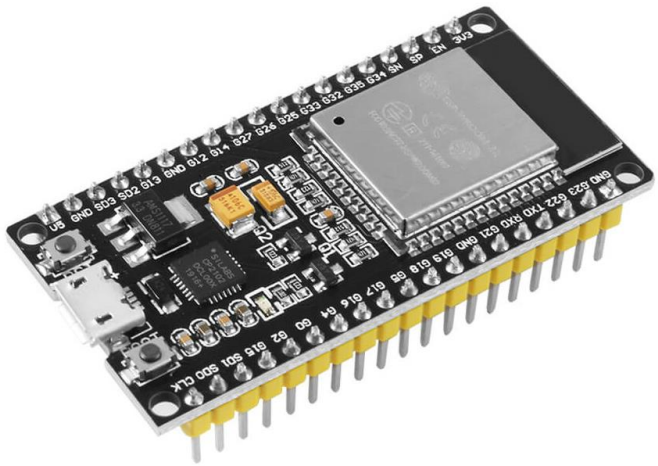

Рисунок 2.2 – ESP32 DevKit V1

Вона має:

– Двоядерний процесор Xtensa LX6 до 240 МГц;

– Вбудовану підтримку Wi-Fi (802.11 b/g/n) та Bluetooth 4.2 (включаючи BLE);

– До 34 GPIO-виводів з можливістю використання для SPI, I2C, UART, ADC, DAC, PWM;

– Пам’ять: 520 КБ SRAM, до 4 МБ Flash;

– Підтримку MicroSD через SPI;

– Напругу логічного рівня 3.3 В;

– Живлення через microUSB (5 В);

– Повна сумісність з Arduino IDE, PlatformIO, ESP-IDF

– Ціна ~250 грн [10].

NodeMCU — це відкрита платформа для IoT-проєктів, яка базується на чипі ESP8266 (рис.2.3) [11].

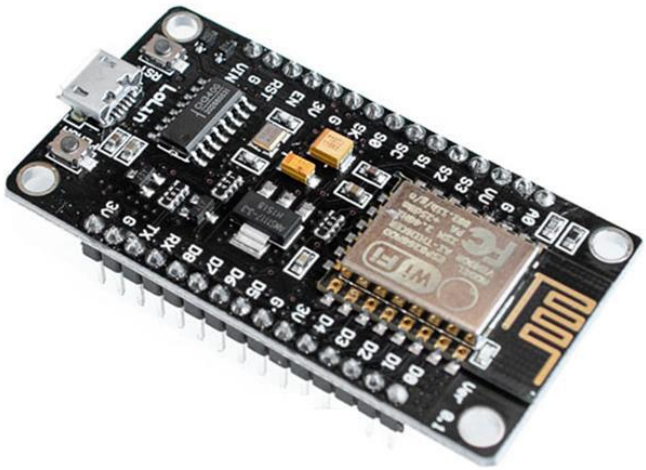

Рисунок 2.3 – ESP8266

Основні характеристики:

– Одноядерний процесор Tensilica L106 з частотою 80 МГц;

– До 11 доступних GPIO-виводів;

– Вбудований Wi-Fi (2.4 ГГц), без Bluetooth;

– Пам’ять: до 4 МБ Flash, 80 КБ SRAM;

– Робоча напруга логічного рівня — 3.3 В;

– Інтерфейси: UART, SPI, I2C, PWM;

– Живлення через microUSB;

– Підтримка Arduino IDE та Lua;

– ціна ~150 грн [11].

Arduino UNO (рис.2.4) — це одна з найпопулярніших плат серед початківців [12]. Вона базується на мікроконтролері ATmega328P.

Рисунок 2.4 – Arduino UNO

Основні технічні параметри:

– Одноядерний процесор з частотою 16 МГц;

– 14 цифрових виводів GPIO, з яких 6 підтримують PWM;

– Пам’ять: 32 КБ Flash, 2 КБ SRAM, 1 КБ EEPROM;

– Немає вбудованого Wi-Fi або Bluetooth;

– Підтримка інтерфейсів: UART, SPI, I2C;

– Напруга логічного рівня — 5 В;

– Живлення через USB або зовнішнє джерело (7–12 В);

– Підтримка середовища Arduino IDE;

– ціна ~300 грн [12].

Після детального порівняльного аналізу трьох популярних мікроконтролерних платформ — Arduino UNO, NodeMCU ESP8266 та ESP32 DevKit V1 — було сформовано таблицю 2.1, яка демонструє ключові технічні характеристики кожної з плат.

Таблиця 2.1 – Основні характеристики плат мікроконтролерів

Параметр	Arduino UNO	NodeMCU ESP8266	ESP32 DevKit V1
Тактова частота (МГц)	16	80	240
GPIO-виводи	14	11	30+
Flash (кБ)	32	4096	4480
SRAM (кБ)	2	80	520
Wi-Fi	Немає	Є	Є
Bluetooth	Немає	Немає	BLE 4.2
Підтримка SD	Обмежена	Є	Повна
Ціна	~300грн	~150грн	~250грн

Arduino UNO є надійною, але застарілою платформою без підтримки бездротових інтерфейсів, з обмеженими ресурсами пам’яті та повільною тактовою частотою. Її використання у сучасних web-орієнтованих системах є недоцільним.

NodeMCU ESP8266 має вбудований Wi-Fi та невелике енергоспоживання, але поступається ESP32 як за кількістю виводів GPIO, так і за обчислювальною потужністю. Крім того, у неї відсутній Bluetooth та нижчий обсяг пам’яті, що зменшує масштабованість проєкту.

ESP32 DevKit V1 має більші габарити, але при цьому забезпечує найкращі показники за всіма параметрами: двоядерний процесор, найбільший обсяг оперативної пам’яті, підтримка Wi-Fi та Bluetooth, можливість підключення зовнішньої пам’яті (SD), велику кількість GPIO-виводів, що робить її універсальним рішенням для розробки складних автоматизованих систем. Саме тому вона була обрана для реалізації даної системи керування бар’єром на платформі ESP32.

Після вибору мікроконтролерної плати наступним важливим кроком стало визначення типу виконавчого елемента для фізичного управління бар’єром. Основним критерієм вибору виступала потреба у точному позиціонуванні та компактності. Серед доступних варіантів були такі серводвигуни: SG90, MG90S та MG996R.

SG90 — це найпопулярніший мікросервопривід, який має крутний момент до 1.8 кг·см, працює при напрузі 4.8–6 В, з максимальним кутом

повороту приблизно 180°. Його розміри (23×12.2×29 мм) і маса близько 9 г роблять його надзвичайно зручним для компактних систем. Швидкість обертання становить 0.1 с/60° при 4.8 В. Матеріал шестерень — пластик, що є достатнім для легких навантажень (рис.2.5) [13].

Рисунок 2.5 – Серводвигун SG90

MG90S має металеві шестерні, вищий крутний момент (2.2–2.5 кг·см), що забезпечує надійнішу роботу при вищих навантаженнях. Проте збільшена вага, габарити та вартість зробили цей варіант менш доцільним для нашої конструкції (рис.2.6) [14].

Рисунок 2.6 – Серводвигун MG90S

MG996R — потужний сервомотор з крутним моментом понад 10 кг·см, призначений для великих механічних систем. Через його розміри та споживання струму він не підходив для даного проекту (рис.2.7) [15].

Рисунок 2.7 – Серводвигун MG996R

Основні характеристики розглянутих сервоприводів наведено нижче у таблиці 2.2, що дозволяє наочно оцінити їх технічні параметри відповідно до потреб системи.

Таблиця 2.2 – Основні характеристики серводвигунів

Параметр	SG90	MG90S	MG996R
Крутний момент	1.8 кг·см	2.2–2.5 кг·см	10–12 кг·см
Напруга живлення	4.8–6 В	4.8–6 В	4.8–7.2 В
Кут повороту	~180°	~180°	~120–180°
Матеріал шестерень	Пластик	Метал	Метал
Розміри (мм)	21×12.2×28	22.8×12.2×28.5	40.7×19.7×42.9
Маса	~9 г	~13.4 г	~55 г
Ціна	~65грн	~100грн	~200грн

З таблиці 2.2 видно, що SG90 має менші розміри та масу, що ідеально підходить для вбудованих автоматизованих систем з обмеженим простором. Хоча MG90S та MG996R мають вищу потужність і надійність, їх параметри перевищують вимоги системи, а підвищене енергоспоживання та розміри є недоліками. Тому SG90 обрано як найоптимальніший варіант. для управління невеликим шлагбаумом, оскільки забезпечує необхідну точність, швидкість та є енергоефективним.

Для збереження журналів системи та даних користувачів була необхідна підтримка зовнішнього накопичувача. Найбільш розповсюдженим рішенням є модулі для читання microSD через SPI. Основні варіанти включали SPI SD Adapter (на 5В), TF Card Reader (3.3В) та модуль Robotdyn.

Модуль Robotdyn SD Module підтримує як 3.3 В, так і 5 В живлення, завдяки вбудованому стабілізатору напруги та логічному перетворювачу рівнів. Це робить його універсальним варіантом як для 5-вольтових, так і для 3.3-вольтових систем. Крім того, він має надійний конденсаторний фільтр для стабільної роботи при зчитуванні та записі даних, а також виводи з маркуванням для швидкого підключення. Така архітектура забезпечує повну сумісність з ESP32 і спрощує інтеграцію в систему (рис.2.8) [16].

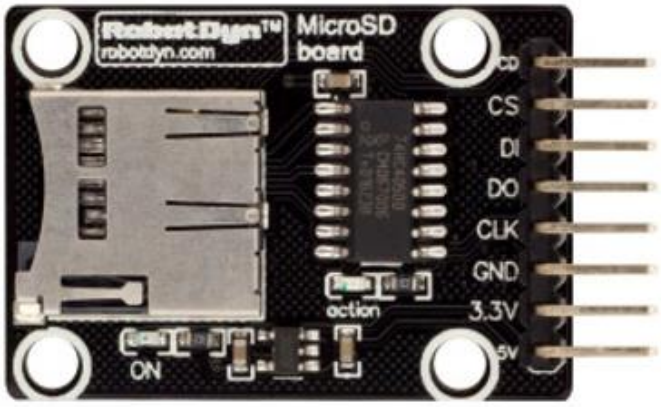

Рисунок 2.8 – Модуль Robotdyn SD Module

SPI SD Adapter (5 В) — це простий модуль для зчитування карт пам’яті через SPI-інтерфейс. Він також містить регулятор напруги, але не має логічного перетворювача рівнів. При використанні з 3.3 В логікою ESP32 потребує зовнішнього захисту або рівнетворення для уникнення перевантаження. Відсутність фільтрації може спричинити проблеми при роботі на високих швидкостях(рис.2.9).

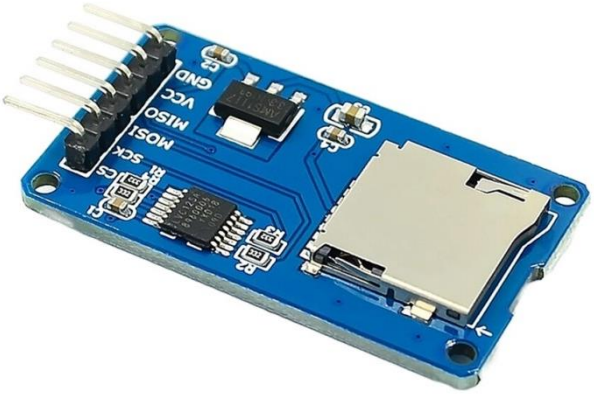

Рисунок 2.9 – SPI SD Adapter (5 В)

TF Card Reader (3.3 В) — компактний адаптер для microSD, який не містить жодних стабілізаторів або захисних елементів. Він розрахований на логіку 3.3 В, тому сумісний з ESP32, але потребує окремого джерела стабільного 3.3 В живлення. Через спрощену конструкцію може виникати нестабільна робота при довготривалому записі чи читанні великих обсягів даних(рис.2.10) [16].

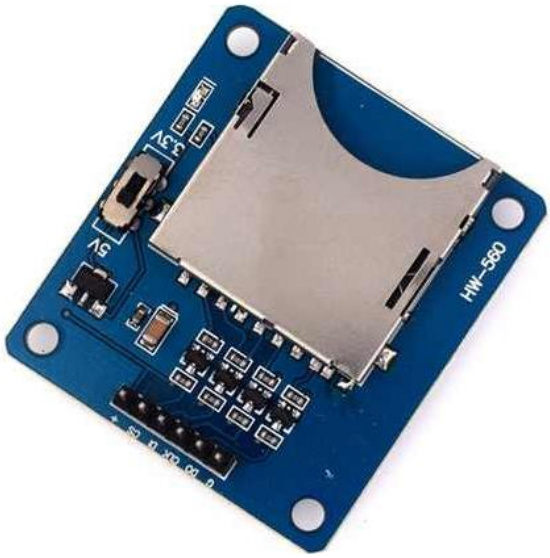

Рисунок 2.10 – TF Card Reader (3.3 В)

Основні характеристики розглянутих SD-модулів представлені у таблиці 2.3.

Таблиця 2.3 – Порівняння модулів для роботи з microSD-картами

Параметр	Robotdyn SD Module	SPI SD Adapter (5B)	TF Card Reader (3.3B)
Напруга живлення	5 В / 3.3 В	5 В	3.3 В
Регулятор напруги	Є	Є	Немає
Підтримка логіки ESP32	Повна (3.3 В)	Часткова	Повна
Наявність фільтрації	Є	Обмежена	Обмежена
Стабільність при записі	Висока	Середня	Низька
Простота інтеграції	Висока	Середня	Складна (може потребувати стабілізації живлення)
Ціна	~150 грн	~60 грн	~60 грн

Порівняння демонструє, що модуль Robotdyn має найбільшу стабільність і найвищу сумісність з ESP32, оскільки містить вбудований регулятор напруги, працює з логікою 3.3 В та має необхідні фільтруючі компоненти. Це забезпечує безпечну та стабільну роботу при довготривалому зберіганні даних. У результаті саме цей адаптер був інтегрований у систему.

Окрім основних компонентів, у системі також були використані допоміжні елементи, які забезпечують індикацію та з’єднання між модулями.

Для індикації стану бар’єра застосовуються світлодіоди червоного та зеленого кольорів. Вони сигналізують про відкритий або закритий стан відповідно (рис.2.11).

Рисунок 2.11 – Червоний світло діод

Живлення здійснюється через резистори номіналом 220–330 Ом для обмеження струму, що є стандартним рішенням для ESP32, яка працює з логічним рівнем 3.3В (рис.2.12).

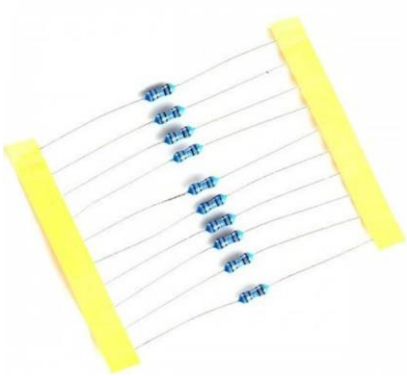

Рисунок 2.11 – Резистори

З’єднання між модулями здійснюється за допомогою з’єднувальних проводів для макетної плати. Це дає змогу гнучко формувати схему, протестувати всі елементи й адаптувати розводку залежно від потреб системи (рис.2.12).

Рисунок 2.12 – З’єднувальні проводи

Для зберігання логів та даних про користувачів було обрано карту пам’яті microSD об'ємом 1 ГБ, відформатовану у файлову систему FAT16. Вона повністю сумісна з бібліотекою SD.h, яку використовує ESP32 для роботи з SPI (рис.2.13).

Рисунок 2.13 – MicroSD об'ємом 1 ГБ

Всі компоненти системи зібрані на макетній платі (breadboard), що забезпечує можливість швидкого тестування, зміни конфігурації та налагодження пристрою без необхідності пайки (рис2.14).

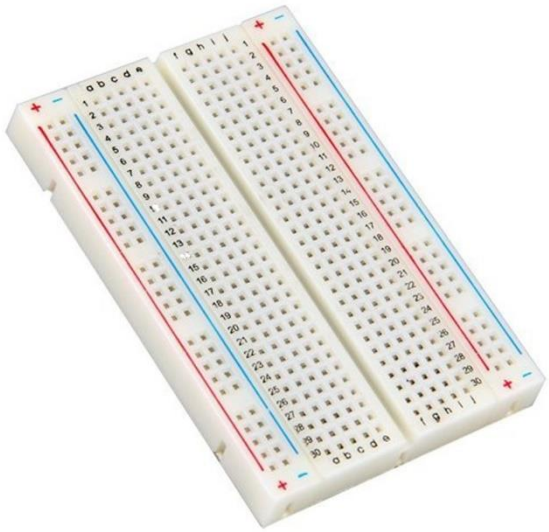

Рисунок 2.14 – Макетна плата

Таким чином, остаточний набір апаратних засобів був сформований таблиця 2.3 з урахуванням сумісності, простоти інтеграції, вартості та відповідності вимогам системи.

Таблиця 2.3 – Вартість апаратного забезпечення

№	Назва компонента	Кількість	Орієнтовна ціна (грн)	Загальна вартість (грн)
1	ESP32 DevKit V1	1	250	200
2	Серводвигун SG90	1	65	65
3	Модуль Robotdyn SD	1	150	150
4	Світлодіоди (червоний, зелений)	2	3	6
5	Резистори 220 Ом	2	1	2
6	Проводи з’єднувальні	10	1	10
7	MicroSD об'ємом 1 ГБ	1	100	100
8	Макетна плата	1	50	50

Загальна вартість системи: приблизно 583 грн сумісності, простоти інтеграції, вартості та відповідності вимогам системи.

2.3 Розробка електричної принципової схеми

У даному розділі розглядається принципова електрична схема системи управління бар’єром, реалізованої на базі ESP32. Принципова схема є основою для розробки функціонального з’єднання усіх апаратних компонентів, з урахуванням особливостей електричних сигналів, рівнів напруги та логіки взаємодії елементів. Основні компоненти, які необхідно з’єднати:

ESP32 DevKit V1 — головний контролер системи;

Серводвигун SG90 — виконавчий пристрій для фізичного відкриття/закриття бар’єру;

Модуль Robotdyn SD — забезпечує запис журналів подій та збереження користувачів на microSD карту;

Світлодіоди — сигналізують стан системи (червоний — закрито, зелений — відкрито);

Резистори — обмежують струм для безпечної роботи світлодіодів;

З’єднувальні проводи — фізичні з’єднання між контактами;

Макетна плата — базова основа для створення електричної схеми без пайки.

Рисунок 2.15 демонструє структурну схему підключення всіх основних елементів до ESP32.

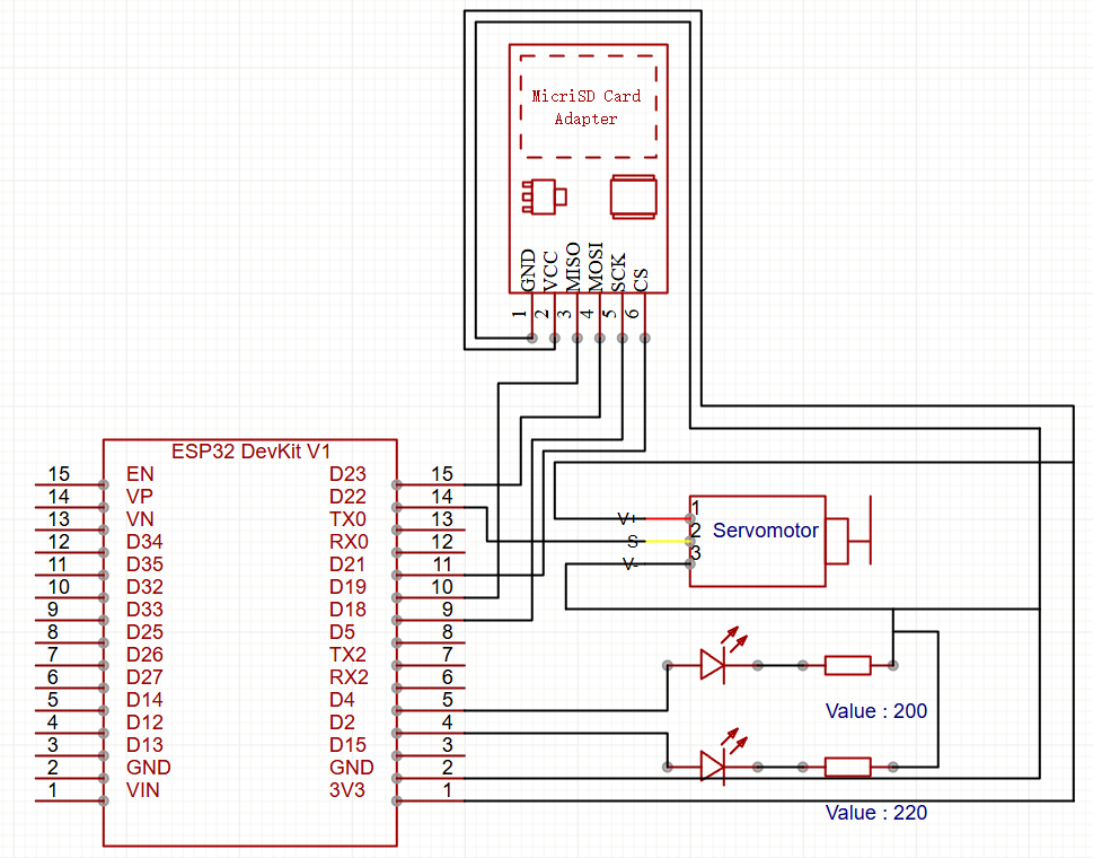

Рисунок 2.15 – Схема підключення компонентів до ESP32

Опис підключень:

Серводвигун SG90 підключається до GPIO 22 (сигнальний пін), VCC до 3,3В, GND до землі.

Модуль Robotdyn SD підключається по SPI-шині: MOSI (GPIO 23), MISO (GPIO 19), SCK (GPIO 18), CS (GPIO 21).

Світлодіоди: зелений до GPIO 4, червоний до GPIO 2 через резистори по 220 Ом.

Живлення ESP32 — через microUSB від зовнішнього джерела живлення або комп’ютера.

Усі з’єднання реалізовані з дотриманням електробезпеки та рекомендацій по логічних рівнях напруг. Таке підключення забезпечує стабільну роботу системи як у тестовому, так і у реальному середовищі експлуатації.

2.4 Розробка алгоритму роботи системи

Для забезпечення коректної логіки роботи системи управління бар’єром на базі ESP32 було розроблено програмний алгоритм, що охоплює всі етапи взаємодії користувача із пристроєм — від підключення до точки доступу до відкриття/закриття бар’єра.

На рисунку 2.16 наведено блок-схему основного алгоритму роботи системи.

Алгоритм реалізує такі основні етапи:

1. Ініціалізація компонентів: встановлення з’єднань із SD-картою, серводвигуном, веб-сервером та світлодіодами.
2. Запуск точки доступу (AP mode) та підключення клієнтів до мережі ESP32.
3. Виведення веб-інтерфейсу з формою авторизації.
4. Обробка введених логіна і пароля та перевірка їх у файлі users.txt.
5. Якщо облікові дані вірні:

– бар’єр відкривається (серводвигун повертається у відкрите положення);

– запис у файл log.txt із фіксацією користувача, дати та часу;

– запускається таймер на 30 секунд;

– після тайм-ауту бар’єр автоматично закривається.

6. Якщо користувач — адміністратор, надається доступ до окремої панелі для перегляду логів та додавання нових користувачів.
7. У разі невірної авторизації доступ блокується.
8. Користувач може вийти із сесії, після чого повертається на стартову сторінку.

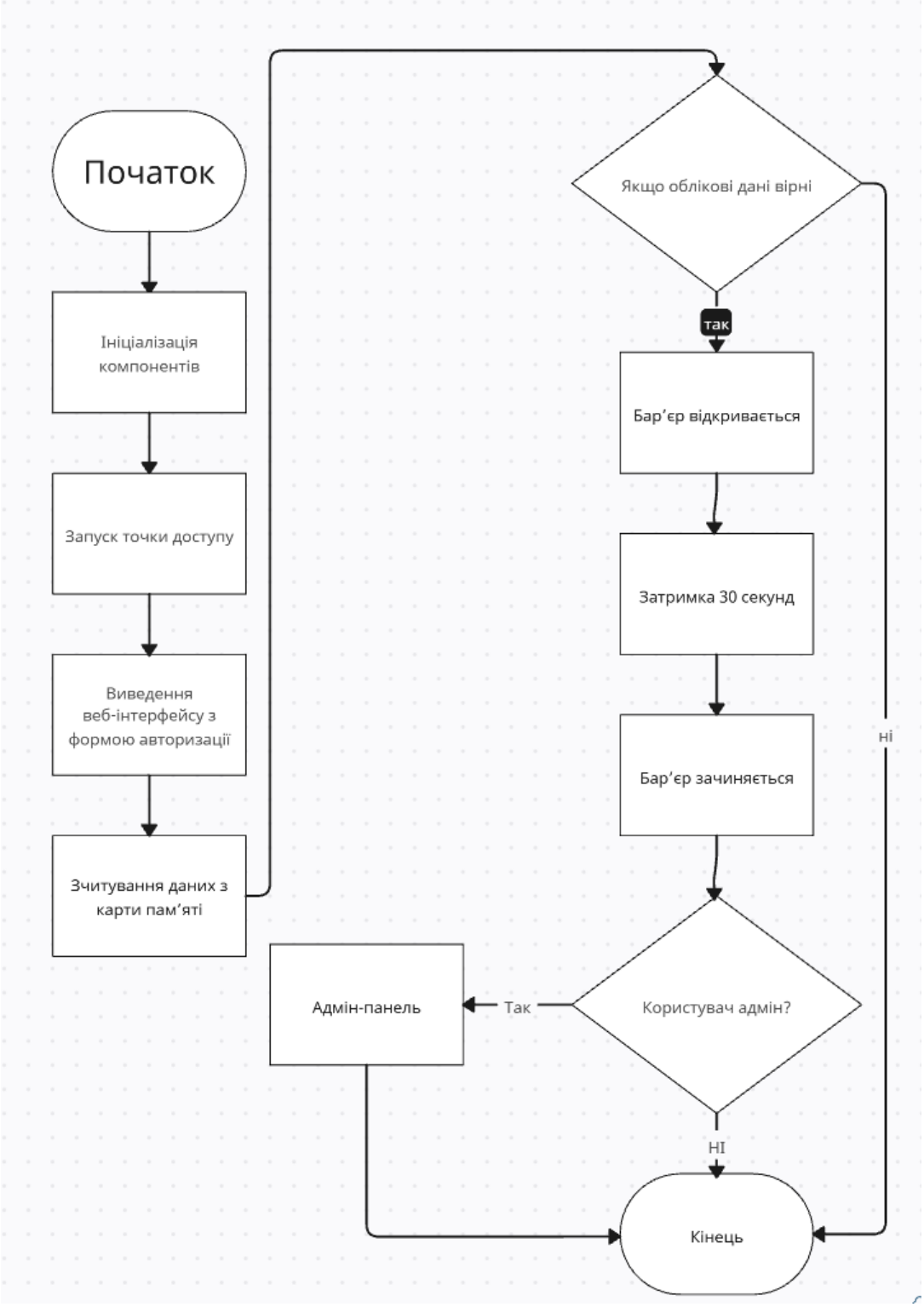

Рисунок 2.16 – Алгоритм роботи системи керування бар'єром на ESP32

Такий алгоритм забезпечує безпечне, гнучке та інтуїтивно зрозуміле керування фізичним бар'єром із веб-браузера, а також дозволяє вести облік дій усіх користувачів у системі [17]. Таке підключення забезпечує стабільну роботу системи як у тестовому, так і у реальному середовищі експлуатації.

2.5 Вибір мови програмування та середовища розробки програмного забезпечення

У процесі створення програмного забезпечення для системи управління бар'єром було проаналізовано декілька мов програмування та середовищ розробки. Основними критеріями при виборі стали: сумісність із платформою ESP32, підтримка периферійних бібліотек, швидкість розробки та налагодження, а також доступність документації та активної спільноти.

Для реалізації проєкту було обрано мову програмування **C++** у середовищі Arduino IDE. Такий вибір обумовлений тим, що Arduino IDE має повну підтримку мікроконтролерів ESP32 через офіційне ядро Espressif, містить велику кількість готових бібліотек, підтримує роботу з файловими системами, веб-серверами, сервоприводами та іншими периферійними пристроями [18].

Arduino IDE дозволяє:

– швидко завантажувати програму на плату ESP32 через USB;

– переглядати вивід у моніторі порту для налагодження;

– використовувати бібліотеки WiFi.h, WebServer.h, SD.h, Servo.h для реалізації ключових функцій системи [19];

– інтегрувати HTML-сторінки та обробники запитів всередині прошивки [20];

Мова C++ забезпечує низькорівневий контроль над ресурсами мікроконтролера, що дозволяє ефективно реалізовувати логіку обробки подій, взаємодії з пам'яттю, файлами та периферією. Крім того, мова підтримує об'єктно-орієнтований підхід, що полегшує структурування коду та повторне використання програмних модулів [21].

Розробка у середовищі Arduino IDE значно спростила імплементацію всієї логіки системи: від створення Wi-Fi точки доступу до контролю серводвигуна, взаємодії з SD-картою, обробки веб-запитів і авторизації користувачів.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ АПАРАТНИХ ТА ПРОГРАМНИХ РІШЕНЬ УПРАВЛІННЯ БАР'ЄРОМ НА ОСНОВІ ESP32

3.1 Загальна структура програмного забезпечення

Програмне забезпечення системи управління бар’єром реалізовано з використанням мікроконтролера ESP32, який виконує роль центрального вузла керування. Основна логіка включає створення точки доступу Wi-Fi, до якої підключаються користувачі для доступу до веб-інтерфейсу системи (рис. 3.1):

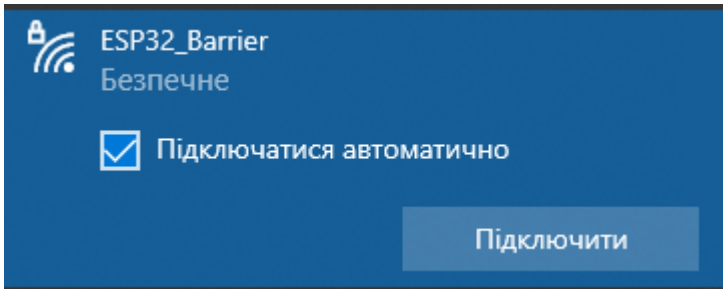

Рисунок 3.1 – Точка доступа «ESP32_Barrier»

Після запуску пристрою створюється точка доступу з іменем "ESP32_Barrier" та паролем "87654321". Підключившись до цієї точки доступу, користувач може перейти за IP-адресою, виданою ESP32 (192.168.4.1), у веб-браузері для взаємодії з головною сторінкою інтерфейсу системи(рис.3.2-3.3).

Рисунок 3.2 – Пошукове поле з IP-адресою

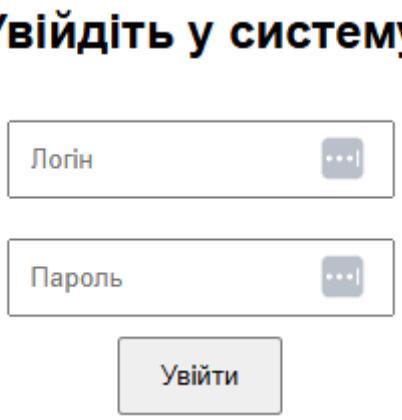

Рисунок 3.3 – Головна сторінка

Головна сторінка пропонує авторизацію користувача. Після успішного входу відкривається головна панель з кнопками керування, а для адміністраторів — додаткові функції, такі як доступ до журналу дій та додавання нових користувачів. Всі сторінки веб-інтерфейсу реалізовані безпосередньо в коді прошивки за допомогою HTML-шаблонів.

Реалізація взаємодії з компонентами системи (світлодіоди, сервопривід, SD-карта) здійснюється через відповідні бібліотеки Arduino, що дозволяє здійснювати читання та запис файлів, зчитування поточного часу через NTP, контроль сервомеханізму та збереження даних користувачів.

3.2 Ініціалізація апаратних компонентів у коді

У коді, реалізованому для ESP32 у проєкті керування бар’єром, ініціалізація апаратних компонентів проводиться поетапно в межах функції *setup()*. Нижче подано розбір реального коду з документа:

Ініціалізація серійного порту:

Serial.begin(115200);

Цей рядок активує серійну комунікацію з комп'ютером для налагодження та виведення повідомлень.

Підключення серводвигуна:

myservo.attach(servoPin);

myservo.write(90);

Встановлюється початкова позиція сервоприводу, що відповідає закритому положенню бар'єру. Змінна *servoPin* попередньо оголошена як пін керування.

Підключення світлодіодів:

pinMode(redLed, OUTPUT);

pinMode(greenLed, OUTPUT);

digitalWrite(redLed, HIGH);

digitalWrite(greenLed, LOW);

Позамовчуванням включений червоний світло діод та зелений вимкнений. Змінні *redLed* та *greenLed* попередньо оголошені як піни керування.

Ініціалізація SD-карти:

```
if (!SD.begin(SD_CS)) {
  Serial.println("✗ Не вдалося ініціалізувати SD-карту!");
  return;
}
```

Використовується бібліотека *SD.h*. У разі помилки ініціалізації з'являється повідомлення, що спрощує діагностику.

Створення Wi-Fi точки доступу:

```
WiFi.softAP(ssid, password);
IPAddress IP = WiFi.softAPIP();
Serial.print("AP IP address: ");
Serial.println(IP);
```

ESP32 ініціює власну мережу Wi-Fi, до якої підключаються користувачі для доступу до веб-інтерфейсу.

Ініціалізація веб-сервера та обробників запитів:

server.on("/", HTTP_GET, handleRoot);

server.on("/login", HTTP_POST, handleLogin);

server.on("/admin", HTTP_GET, handleAdmin);

server.on("/open", HTTP_POST, handleOpen);

server.on("/logout", HTTP_GET, handleLogout);

server.begin();

Встановлюються маршрути для основних сторінок інтерфейсу: головна, логін, адміністративна панель, команда відкриття бар’єру та вихід.

Усі вищенаведені фрагменти забезпечують базову функціональність і запуск усіх компонентів одразу після включення пристрою. Їхній коректний порядок ініціалізації гарантує безперебійну роботу системи.

3.3 Програмна логіка авторизації користувачів

Програмна логіка авторизації користувачів реалізована через веб-інтерфейс, доступний при підключенні до точки доступу ESP32. Авторизація потрібна для контролю доступу до системи керування бар'єром та розмежування прав користувачів і адміністраторів.

Реалізація зчитування користувачів із SD-карти

У функції *loadUsersFromSD()* реалізовано зчитування зареєстрованих користувачів із файлу */users.txt* (рис.3.4) SD-карті.

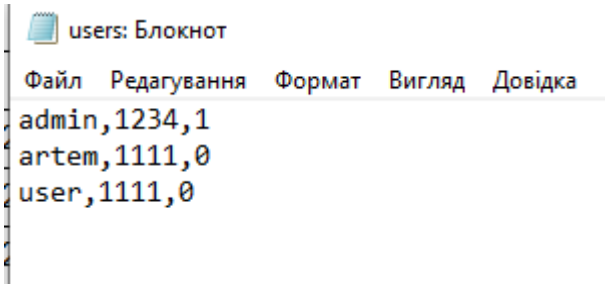

Рисунок 3.4 – Текстовий документ з даними користувача

Кожен рядок файлу містить логін, пароль та ознаку адміністратора:

```
if (SD.exists("/users.txt")) {
```

```
  File file = SD.open("/users.txt", FILE_READ);
  while (file.available() && userCount < MAX_USERS) {
    String line = file.readStringUntil('
');
    // Обробка рядка, розділення на логін, пароль, isAdmin
  }
  file.close();
}
```

У разі відсутності файлу створюється новий файл з обліковими даними адміністратора за замовчуванням.

Перевірка облікових даних

Користувач вводить логін і пароль у HTML-формі(рис3.5).

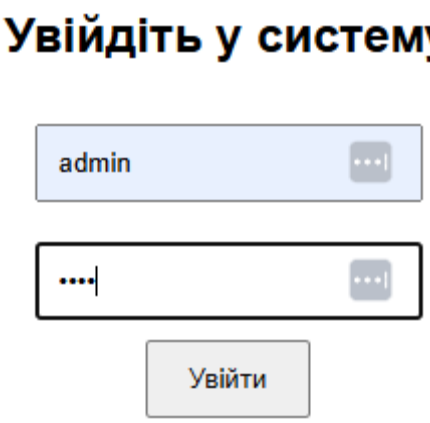

Рисунок 3.5 – Авторизація користувача

Дані передаються на сервер за методом *POST*, який обробляється функцією *handleLogin():*

```
void handleLogin() {
 String u = server.arg("username");
 String p = server.arg("password");
 for (int i = 0; i < userCount; i++) {
  if (users[i].username == u && users[i].password == p) {
   isAuthorized = true;
```

```
      isAdmin = users[i].isAdmin;
      currentUser = u;
      break;
    }
  }
  if (isAuthorized) {
    server.sendHeader("Location", "/");
    server.send(303);
  } else {
    server.send(200, "text/html", getFailPage());
  }
}
```

Головна сторінка та адмін-панель

Функція *handleRoot()* повертає головну сторінку системи для авторизованих користувачів(рис.3.6):

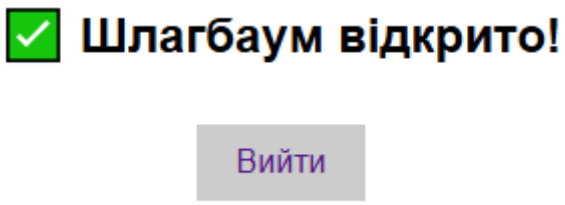

Рисунок 3.6 – Сторінка після успішної авторизації

Якщо користувач є адміністратором, йому додатково надається доступ до *handleAdmin()* — адмін-панелі (рис.3.7-3.8), де реалізовано:

– додавання нових користувачів;
– перегляд журналу відкриттів бар’єру;
– кнопки «Назад» та «Вийти».

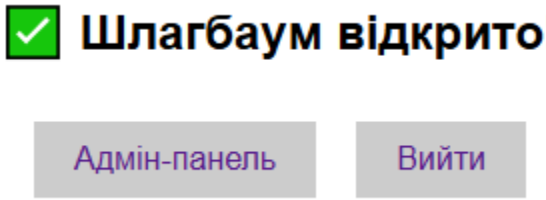

Рисунок 3.7 – Сторінка після успішної авторизації адміна

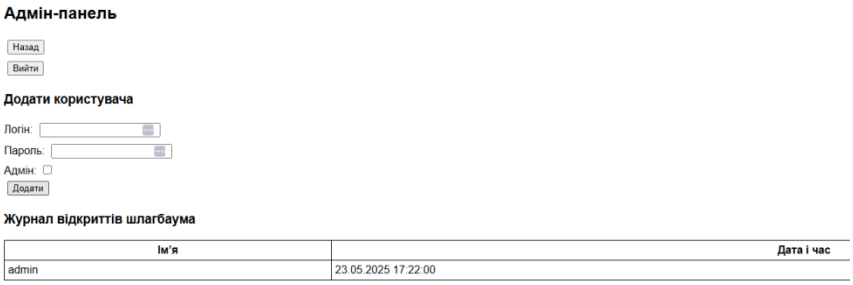

Рисунок 3.8 – Адмін-панель

Збереження логів

Ім’я користувача та час відкриття бар’єру записується у */log.txt* показано на рисунку 3.9:

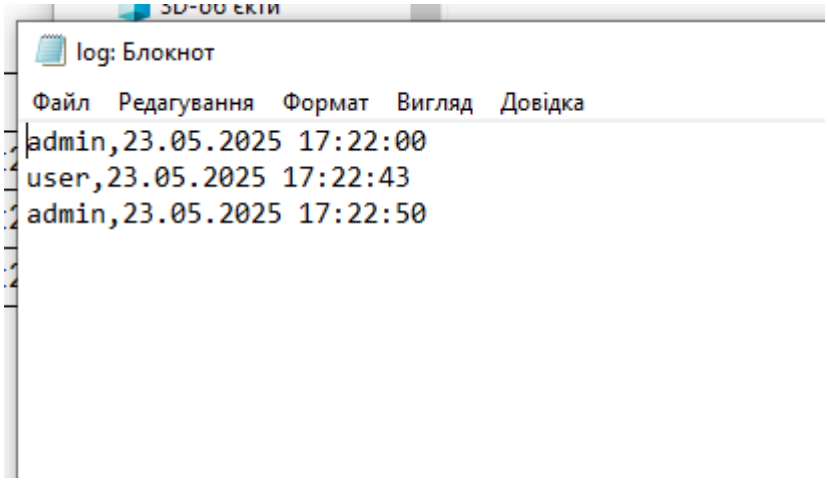

Рисунок 3.9 – Файл «*log.txt*»

Код:

appendLogToSD(currentUser, timestamp);

Це дозволяє адміністратору переглядати історію доступу через HTML-таблицю в адмін-панелі.

Вихід із системи

При виході користувача викликається *handleLogout(),* яка скидає всі змінні автентифікації та повертає систему в початковий стан.

3.4 Реалізація журналу доступу та адмін-панелі

Журнал доступу до системи керування бар'єром реалізується за допомогою зберігання даних у текстовий файл */log.txt* на SD-карті.

Після кожного успішного входу користувача та активації шлагбаума створюється запис у журналі, який містить ім'я користувача та дату з часом дії. Це виконується функцією *appendLogToSD()*:

```
void appendLogToSD(String name, String datetime) {
  File file = SD.open("/log.txt", FILE_APPEND);
  file.printf("%s,%s\n", name.c_str(), datetime.c_str());
  file.close();
}
```

Записи журналу виводяться у табличному вигляді в HTML-форматі через адмін-панель:

```
for (int i = 0; i < logCount; i++) {
  html += "<tr><td>" + logs[i].username + "</td><td>" + logs[i].datetime
+ "</td></tr>";
}
```

Функціональність адміністративної панелі

Реалізовано окрему веб-сторінку *getAdminPage()*, яка доступна лише авторизованим адміністраторам. Адмін може додати нового користувача за допомогою форми (рис.3.10), що надсилається методом *POST* на */adduser*.

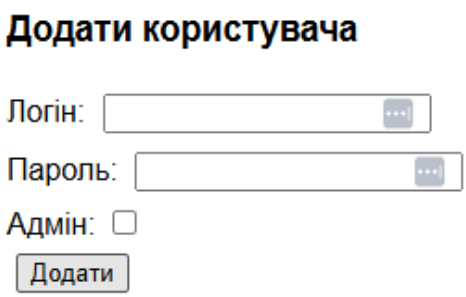

Рисунок 3.10 – Форма дял додавання нового користувача

Обробник *handleAddUser()* перевіряє права доступу, додає користувача до масиву *users* і записує його у файл */users.txt* на SD-карті:

```
void handleAddUser() {
  if (!isAuthorized || !isAdmin || userCount >= MAX_USERS) {
    server.send(403, "text/plain", "Помилка додавання");
    return;
  }
  String newuser = server.arg("newuser");
  String newpass = server.arg("newpass");
  bool adminFlag = server.hasArg("isadmin");
  users[userCount++] = {newuser, newpass, adminFlag};
  saveUserToSD(users[userCount - 1]);
  server.sendHeader("Location", "/admin");
  server.send(303);
}
```

Захист доступу

Для захисту панелі використовується перевірка ролі користувача. Якщо він не є адміністратором, система повертає код 403 «Доступ заборонено»:

```
if (!isAuthorized || !isAdmin) {
  server.send(403, "text/plain", "Доступ заборонено");
  return;
}
```

3.5 Реалізація алгоритму керування компонентами бар’єра

Даний розділ зосереджений на реалізації фізичної взаємодії з компонентами: сервоприводом, світлодіодами та таймером. Після успішної авторизації користувача у функції *handleRoot()* виконується безпосереднє відкриття шлагбаума та зміна стану індикаторів(рис.3.11-3.12):

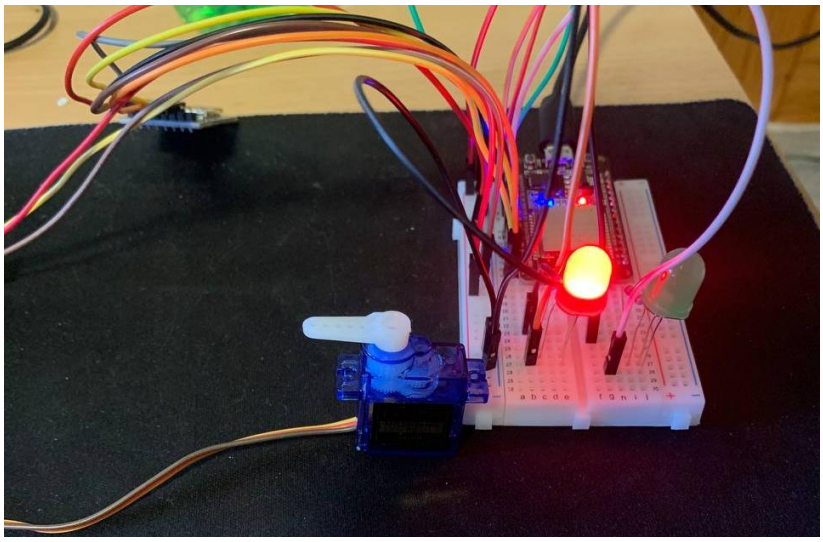

Рисунок 3.11 – Початковий стан системи

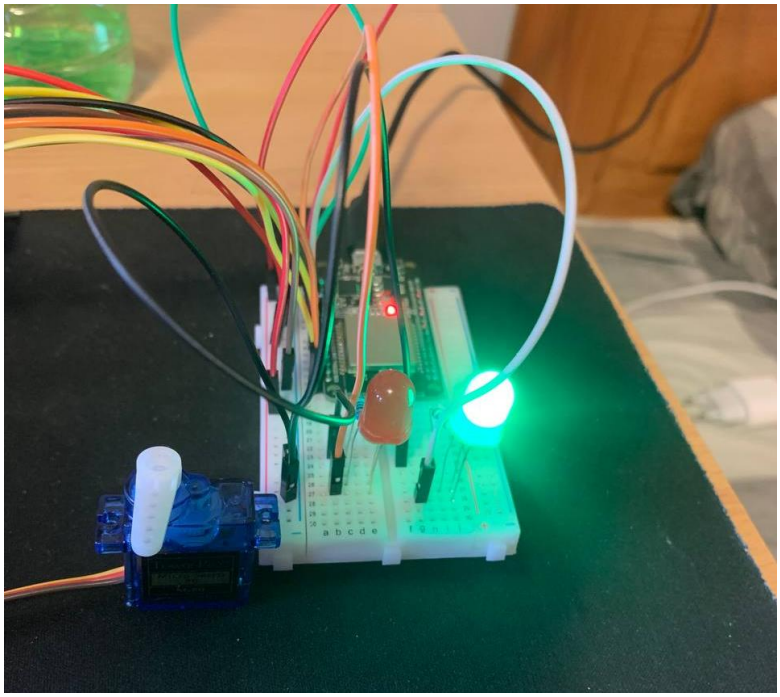

Рисунок 3.12 – Стан системи після успішної авторизації

Код:

```
barrier.write(90);              // Відкриття бар'єра
 digitalWrite(redLed, LOW);     // Вимкнути червоний світлодіод
 digitalWrite(greenLed, HIGH);  // Увімкнути зелений світлодіод
```

Ці дії сигналізують про дозвіл проїзду. Після цього фіксується час відкриття для подальшого контролю:

```
barrierOpenTime = millis();
barrierIsOpen = true;
```

Через 30 секунд у головному циклі *loop()* автоматично виконується закриття бар'єра та зворотна зміна індикації:

```
if (barrierIsOpen && millis() - barrierOpenTime >= 30000) {
 barrier.write(0);              // Закриття бар'єра
 digitalWrite(redLed, HIGH);    // Вмикаємо червоний світлодіод
 digitalWrite(greenLed, LOW);   // Вимикаємо зелений
 barrierIsOpen = false;}
```

3.6 Перевірка працездатності розробленої системи

Для перевірки працездатності системи управління бар'єром, реалізованої на базі ESP32, було здійснено низку тестів як апаратної, так і програмної частини. Основною метою перевірки було переконатися, що всі компоненти функціонують відповідно до заданого алгоритму та взаємодіють між собою без збоїв.

На першому етапі система була підключена до комп'ютера або джерела живлення. Після подачі напруги живлення ESP32 автоматично створює точку доступу з назвою «ESP32_Barrier» та паролем «87654321». Це підтверджується повідомленням у серійному моніторі, а також доступністю точки доступу на пристроях користувача(рис.3.13):

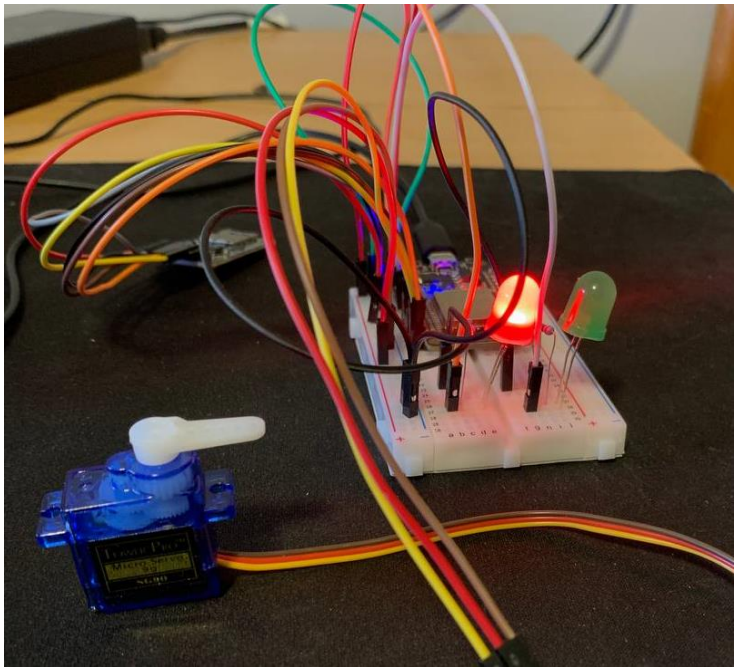

Рисунок 3.13 – Початковий стан системи при підключенні до ноутбука

Після підключення до мережі, у браузері необхідно ввести IP-адресу, (192.168.4.1) для переходу на головну сторінку веб-інтерфейсу (рис.3.14-3.15):

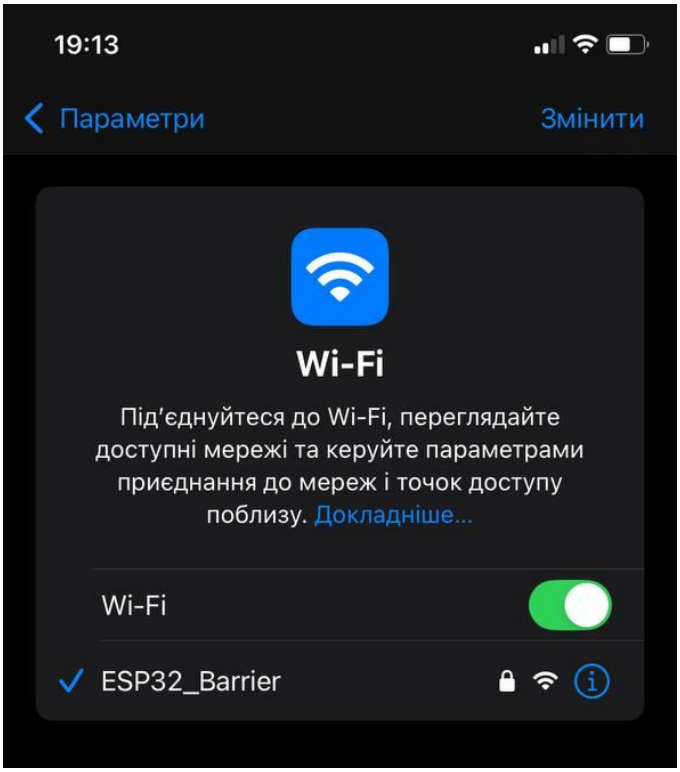

Рисунок 3.14 – Точка доступу ESP32 на телефоні

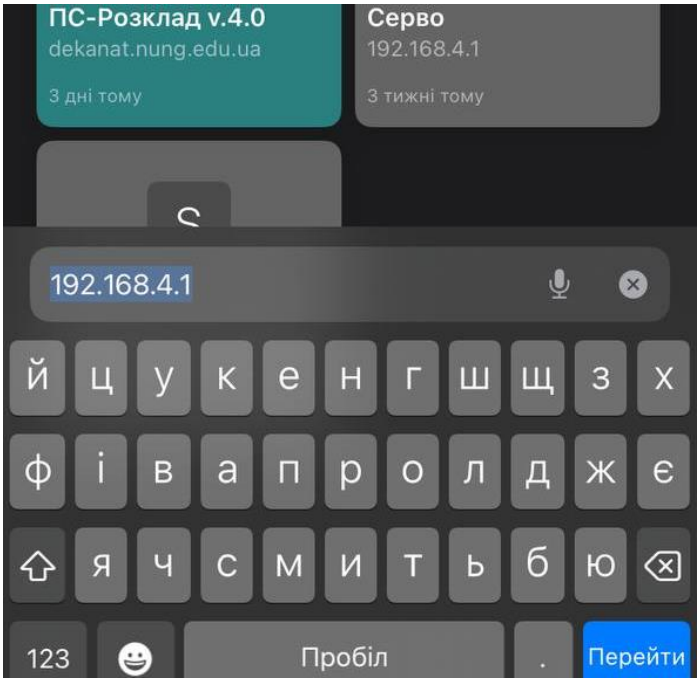

Рисунок 3.15 – Введення IP-адреси в браузері смартфона

Далі користувач потрапляє на сторінку авторизації, де повинен ввести логін і пароль. У разі правильного введення даних система підтверджує авторизацію та виконує логіку доступу (рис.3.16-3.17):

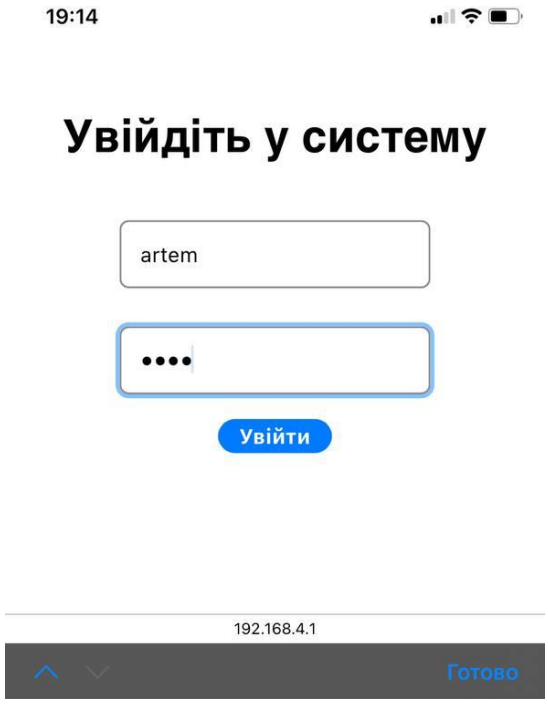

Рисунок 3.16 – Сторінка авторизація

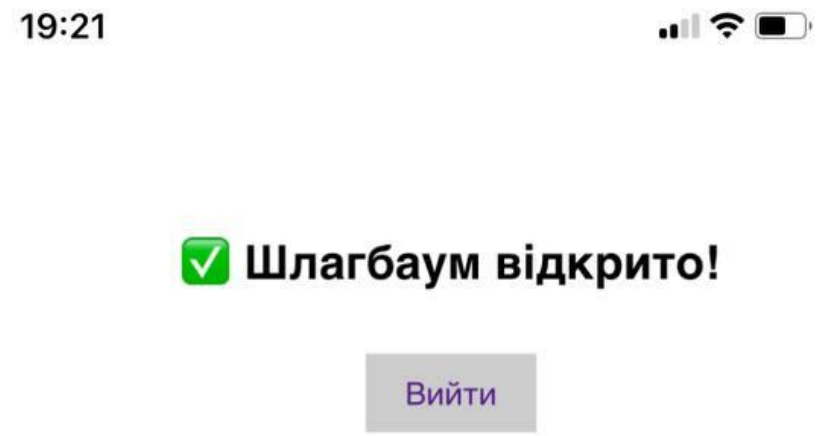

Рисунок 3.17 – Успішна авторизація

В разі введення неправильних даних система про це повідомляє:

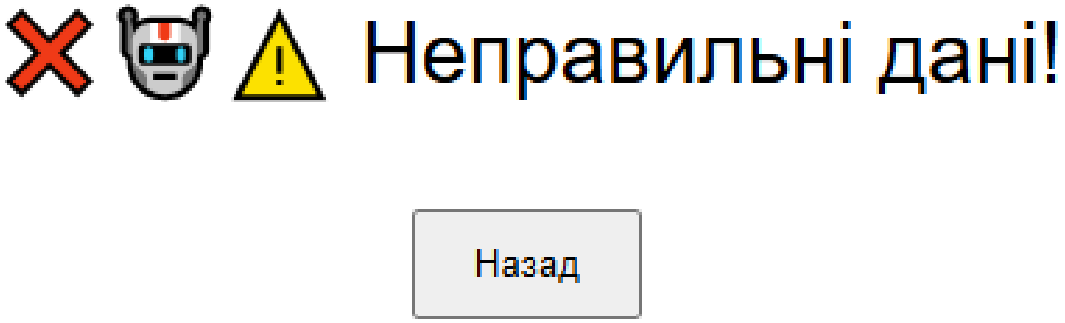

Рисунок 3.18 – Неуспішна авторизація

Успішна авторизація супроводжується фізичним відкриттям шлагбаума (сервопривід повертається у кут 90°), гасінням червоного та вмиканням зеленого світлодіода. Через 30 секунд шлагбаум автоматично закривається (рис. 3.19):

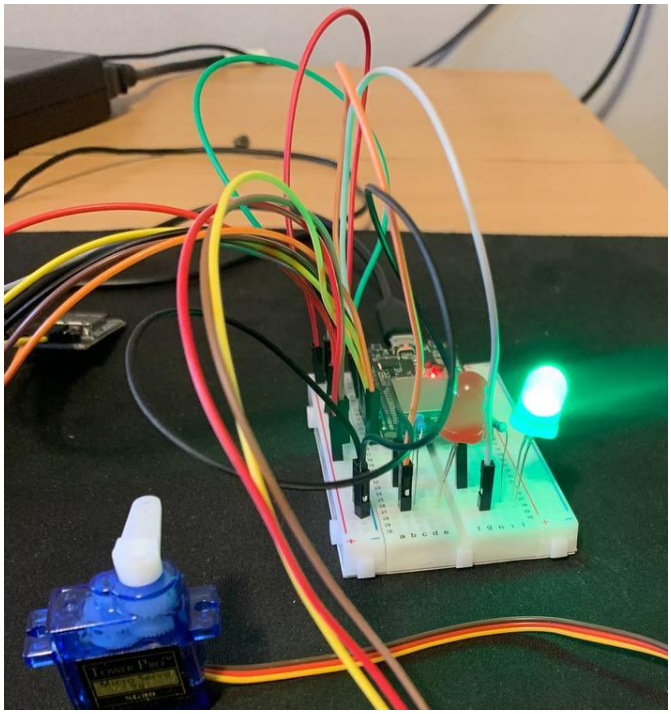

Рисунок 3.19 – Відкриття шлагбауна

Користувач з адміністративними правами має доступ до додаткової сторінки – адмін-панелі, де доступні журнал подій, а також інтерфейс додавання нових користувачів (рис.3.20-3.22):

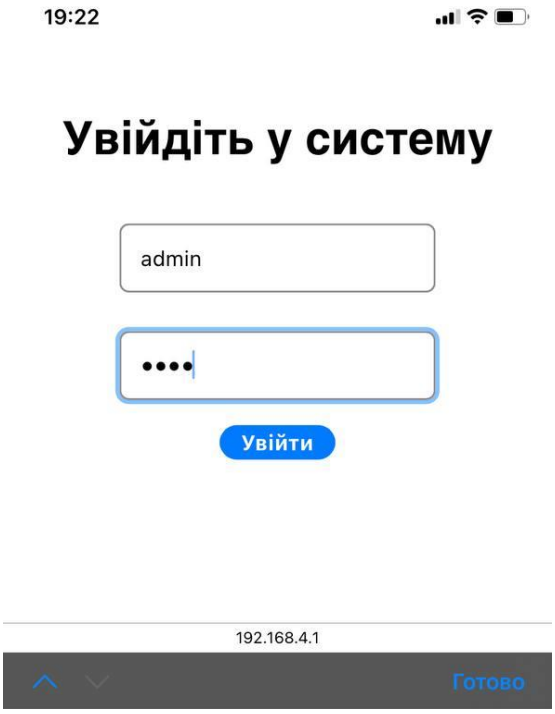

Рисунок 3.20 – Авторизація адміністратора

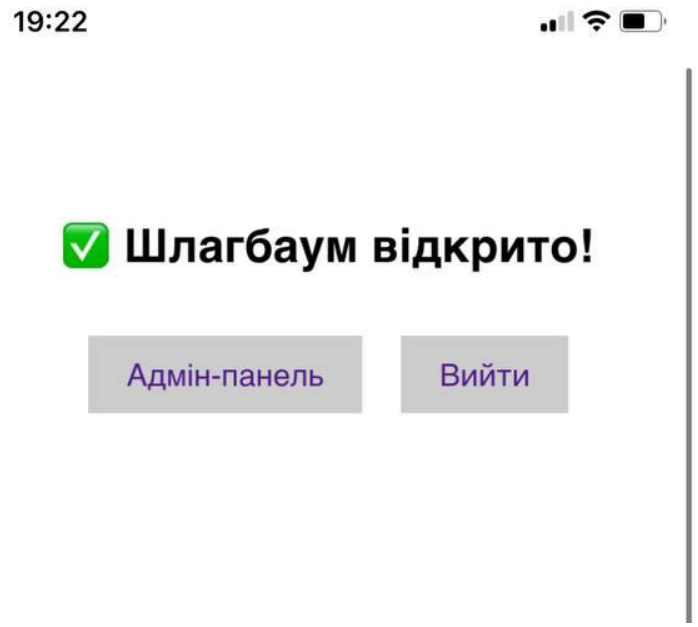

Рисунок 3.21 – Успішна авторизація

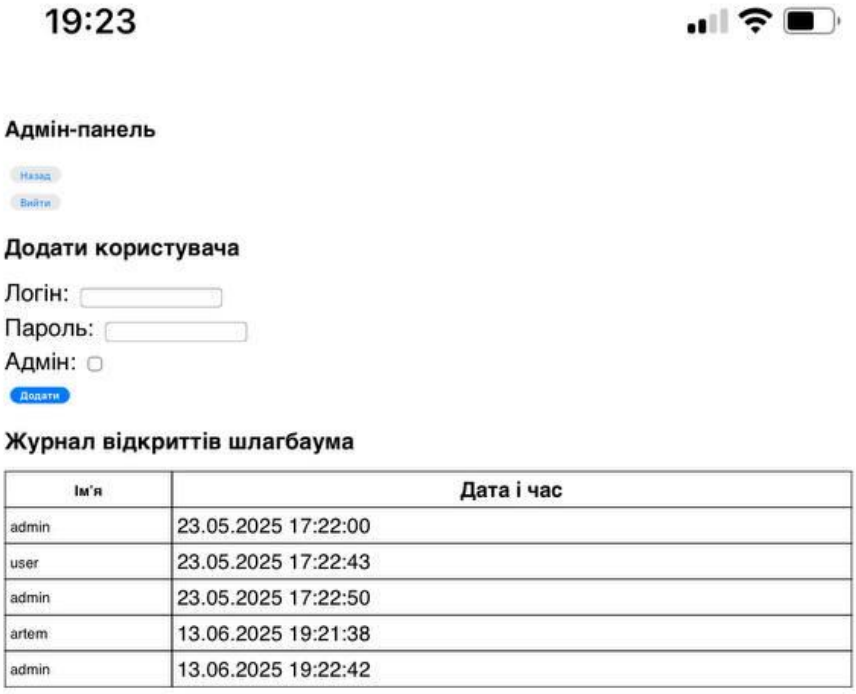

Рисунок 3.22 – Інтерфейс адміністратора

Адміністратор додає нового користувача за допомогою відповідної форми. Цей користувач з’являється у файлі *users.txt* на SD-карті, а також може авторизуватись і отримати базовий функціонал доступу до системи (рис.3.23-3.26):

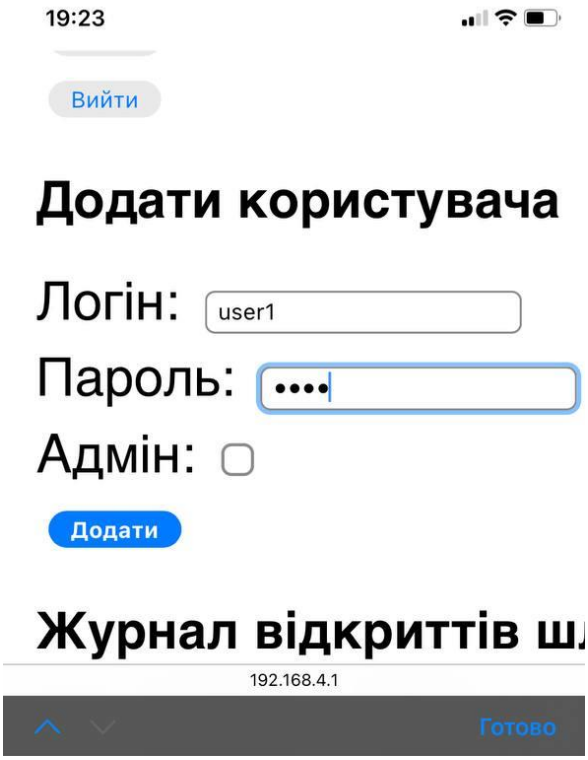

Рисунок 3.23 – Добавлення нового користувача

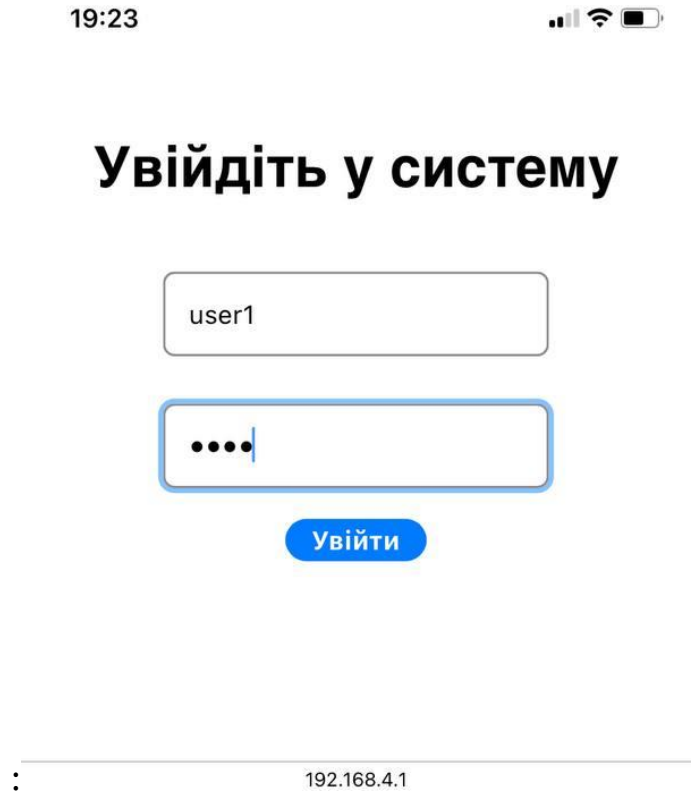

Рисунок 3.24 – Авторизація за даними нового користувача

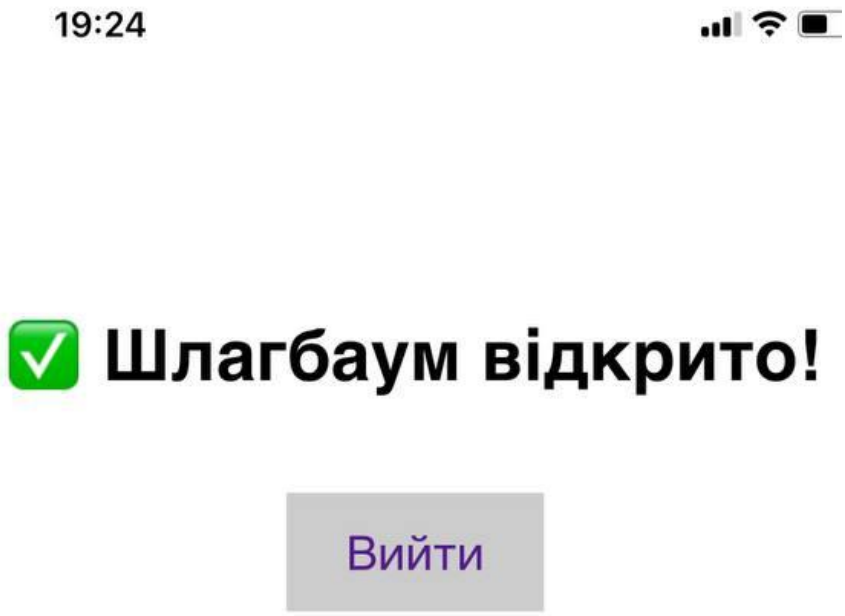

Рисунок 3.25 – Успішна авторизація

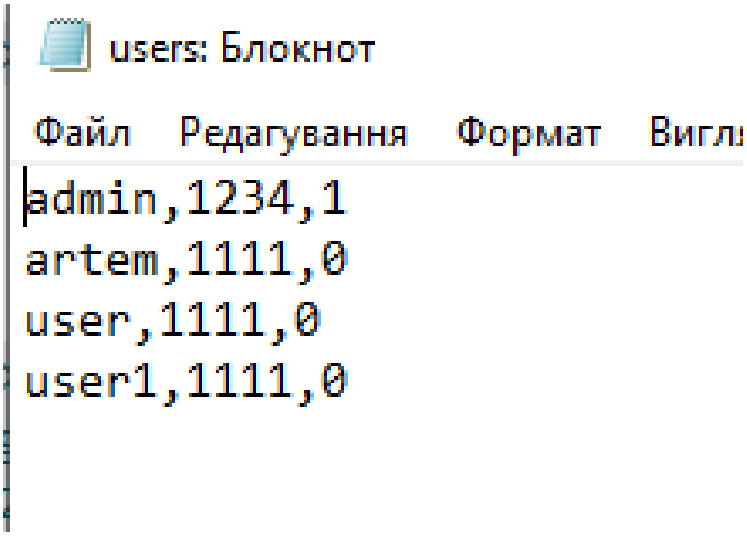

Рисунок 3.26 – Файл «user.txt»

Журнал входів формується автоматично – кожен факт відкриття шлагбаума зберігається як у динамічному масиві, так і в файлі *log.txt (*рис. 3.27-3.28):

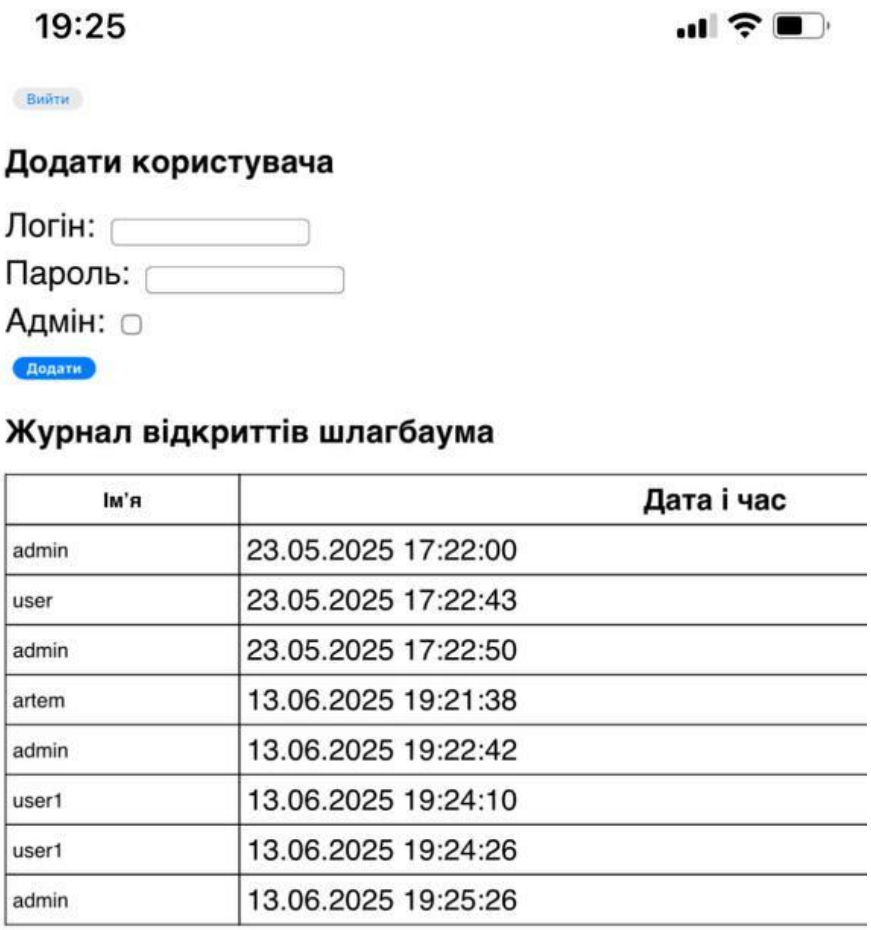

Рисунок 3.27 – Журнал

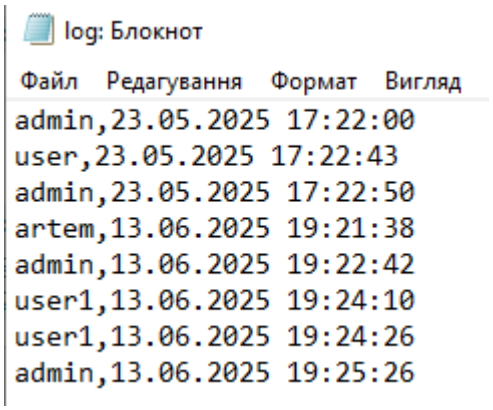

Рисунок 3.28 – Журнал у файлі «log.txt»

Висновок

Проведена перевірка довела працездатність запропонованої системи. Усі її функції виконуються коректно: точка доступу створюється стабільно, авторизація працює з урахуванням ролей, шлагбаум фізично реагує на команди,

а всі дії логуються та зберігаються. Система може бути впроваджена як у навчальних, так і в прикладних задачах контролю доступу без підключення до зовнішнього Інтернету.

ВИСНОВКИ

У ході виконання дипломної роботи було здійснено повний цикл розробки мікроконтролерної системи управління шлагбаумом з вбудованим веб-інтерфейсом на базі плати ESP32. Проведено аналіз аналогічних рішень, визначено вимоги до функціональності системи та інформативних ознак, які мають бути реалізовані для забезпечення ефективної роботи системи.

На основі цього було розроблено апаратну архітектуру системи, зокрема структурну та принципову електричну схеми, а також обґрунтовано вибір апаратних компонентів на основі порівняльного аналізу технічних характеристик і можливостей.

Було обрано мову програмування C++ та середовище розробки Arduino IDE для написання програмного забезпечення, що дозволило забезпечити простоту розробки та широкі можливості взаємодії з апаратними модулями. Розроблений програмний код реалізує створення точки доступу Wi-Fi, авторизацію користувачів, управління сервоприводом, індикацію за допомогою світлодіодів, запис журналу подій та управління користувачами через інтерфейс адміністратора.

Система пройшла повне тестування, в ході якого підтверджено її працездатність, відповідність вимогам і стабільну роботу в умовах реального використання. Результати реалізації продемонстрували функціональність таких можливостей, як автоматичне закриття шлагбаума, авторизація з обмеженням прав доступу, збереження логів на SD-карту, адаптивність інтерфейсу під різні сценарії використання.

Таким чином, розроблена система є надійним, зручним у використанні та технологічно ефективним рішенням для автоматизованого контролю доступу. Вона має прикладну цінність, може бути використана в навчальних цілях, а також має потенціал для подальшої комерціалізації або впровадження у смарт-інфраструктуру.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Wikipedia. Information system. URL: https://en.wikipedia.org/wiki/Information_system (дата звернення: 28.05.2025).

2. Закон України «Про інформацію». URL: https://zakon.rada.gov.ua/laws/show/2657-12 (дата звернення: 28.05.2025).

3. Закон України «Про захист персональних даних». URL: https://zakon.rada.gov.ua/laws/show/2297-17 (дата звернення: 28.05.2025).

4. Закон України «Про електронні комунікації». URL: https://zakon.rada.gov.ua/laws/show/1089-20 (дата звернення: 28.05.2025).

5. Закон України «Про наукову і науково-технічну діяльність». URL: https://zakon.rada.gov.ua/laws/show/848-19 (дата звернення: 28.05.2025).

6. Random Nerd Tutorials. ESP32 with RFID RC522 – Access Control System. URL: https://randomnerdtutorials.com/esp32-rfid-nfc-rc522-door-lock/ (дата звернення: 28.05.2025).

7. W3Schools. HTML Forms. URL: https://www.w3schools.com/html/html_forms.asp (дата звернення: 30.05.2025).

8. Maker Advisor. ESP32 Web Server using WiFi Station (STA) mode. URL: https://makeradvisor.com/esp32-web-server-wifi-station/ (дата звернення: 28.05.2025).

9. Random Nerd Tutorials. ESP32 Telegram: Control Outputs and Monitor Sensors (Arduino IDE). URL: https://randomnerdtutorials.com/esp32-telegram-control/ (дата звернення: 28.05.2025).

10. Espressif Systems. ESP32-WROOM-32 Datasheet. URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (дата звернення: 03.06.2025).

11. NodeMCU Documentation. ESP8266 NodeMCU DevKit. URL: https://nodemcu.readthedocs.io/en/release/ (дата звернення: 03.06.2025).

12. Arduino. Arduino Uno Rev3. URL: https://store.arduino.cc/products/arduino-uno-rev3 (дата звернення: 04.06.2025).

13. Components101. SG90 Servo Motor Specifications. URL: https://components101.com/motors/sg90-servo-motor (дата звернення: 04.06.2025).

14. Electropeak. MG90S Micro Servo Motor Datasheet. URL: https://www.electropeak.com/learn/mg90s-servo-motor-tutorial/ (дата звернення: 04.06.2025).

15. ArduinoModulesInfo. MG996R Servo Motor. URL: https://arduinomodules.info/mg996r-servo-motor/ (дата звернення: 04.06.2025).

16. Robotdyn. MicroSD Card Adapter for Arduino. URL: https://robotdyn.com/microsd-card-module.html (дата звернення: 05.06.2025).

17. Arduino Forum. How to Secure Web Interface on ESP32. URL: https://forum.arduino.cc/t/esp32-secure-web-login/870095 (дата звернення: 05.06.2025).

18. Arduino IDE для ESP32 – офіційна документація. URL: https://docs.espressif.com/projects/arduino-esp32/en/latest/ (дата звернення: 05.06.2025).

19. GitHub. Arduino Libraries for ESP32. URL: https://github.com/espressif/arduino-esp32/tree/master/libraries (дата звернення: 05.06.2025).

20. Rui Santos. ESP32 Web Server – Arduino IDE. URL: https://randomnerdtutorials.com/esp32-web-server-arduino-ide/ (дата звернення: 05.06.2025).

21. C++ Language Reference – The C++ Programming Language. URL: https://cplusplus.com/doc/tutorial/ (дата звернення: 05.06.2025).

Додатки

Код програми:

```
#include <WiFi.h>
#include <WebServer.h>
#include <ESP32Servo.h>
#include <time.h>
#include <SPI.h>
#include <SD.h>

#define SD_CS 21  // Пін CS для SD-карти

const char* ssid = "ESP32_Barrier";
const char* password = "87654321";

const char* login = "netis";
const char* pasword = "password";

WebServer server(80);

Servo barrier;
const int servoPin = 22;
const int redLed = 2;
const int greenLed = 4;

struct User {
String username;
String password;
bool isAdmin;
};
#define MAX_USERS 10
User users[MAX_USERS];
int userCount = 0;
void loadUsersFromSD() {
```

```
if (SD.exists("/users.txt")) {
File file = SD.open("/users.txt", FILE_READ);
while (file.available() && userCount < MAX_USERS) {
String line = file.readStringUntil('\n');
line.trim();
if (line.length() == 0) continue;
int sep1 = line.indexOf(',');
int sep2 = line.lastIndexOf(',');
if (sep1 > 0 && sep2 > sep1) {
users[userCount].username = line.substring(0, sep1);
users[userCount].password = line.substring(sep1 + 1, sep2);
users[userCount].isAdmin = line.substring(sep2 + 1) == "1";
userCount++;
}
}
file.close();
} else {
// Створити файл з адміном за замовчуванням
users[0] = {"admin", "1234", true};
userCount = 1;
File file = SD.open("/users.txt", FILE_WRITE);
if (file) {
file.printf("%s,%s,%d\n", users[0].username.c_str(), users[0].password.c_str(),
users[0].isAdmin ? 1 : 0);
file.close();
}
}
}

void saveUserToSD(User u) {
File file = SD.open("/users.txt", FILE_APPEND);
if (!file) {
Serial.println("_Error_Не вдалося відкрити users.txt для запису");
return;
}
```

```
file.printf("%s,%s,%d\n", u.username.c_str(), u.password.c_str(), u.isAdmin ? 1 : 0);
file.close();
}

struct LogEntry {
String username;
String datetime;
};
#define MAX_LOGS 50
LogEntry logs[MAX_LOGS];
int logCount = 0;

void appendLogToSD(String name, String datetime) {
File file = SD.open("/log.txt", FILE_APPEND);
if (!file) {
Serial.println("_Error_Не вдалося відкрити log.txt для запису");
return;
}
file.printf("%s,%s\n", name.c_str(), datetime.c_str());
file.close();
}

void loadLogsFromSD() {
if (SD.exists("/log.txt")) {
File file = SD.open("/log.txt", FILE_READ);
while (file.available() && logCount < MAX_LOGS) {
String line = file.readStringUntil('\n');
line.trim();
if (line.length() == 0) continue;
int sep = line.indexOf(',');
if (sep > 0) {
logs[logCount].username = line.substring(0, sep);
logs[logCount].datetime = line.substring(sep + 1);
logCount++;
}
```

```
}
file.close();
}
}

bool isAuthorized = false;
bool isAdmin = false;
String currentUser = "";
unsigned long barrierOpenTime = 0;
bool barrierIsOpen = false;

String getCurrentTime() {
time_t now = time(nullptr);
struct tm* timeinfo = localtime(&now);
char buffer[30];
strftime(buffer, sizeof(buffer), "%d.%m.%Y %H:%M:%S", timeinfo);
return String(buffer);
}

String getLoginPage() {
  String html = R"rawliteral(
                    <!DOCTYPE html><html><head><meta charset='UTF-
8'><title>Авторизація</title>
    <style>body{text-align:center;padding-top:50px;font-family:sans-serif;}
    input{padding:10px;margin:10px;}button{padding:10px 20px;}</style>
    </head><body><h2>Увійдіть у систему</h2>
    <form method='POST' action='/login'>
      <input name='username' placeholder='Логін'><br>
      <input name='password' type='password' placeholder='Пароль'><br>
      <button type='submit'>Увійти</button>
    </form>
  )rawliteral";
  html += "</body></html>";
  return html;
}
```

```
String getFailPage() {
  return R"rawliteral(
    <!DOCTYPE html><html><head><meta charset='UTF-8'><title>Помилка</title>
    <style>body{text-align:center;padding-top:50px;font-family:sans-serif;font-size:30px;}
    button{padding:10px 20px;}</style>
    </head><body>
    ❌ ⚠️ Неправильні дані!<br><br>
    <form action="/"><button>Назад</button></form>
    </body></html>
  )rawliteral";
}

String getMainPage() {
  String html = R"rawliteral(
    <!DOCTYPE html><html><head><meta charset='UTF-8'><title>Шлагбаум</title>
    <style>
      body{text-align:center;padding-top:50px;font-family:sans-serif;}
          a{display:inline-block;margin:10px;padding:10px 20px;background:#ccc;text-
decoration:none;}
    </style>
    </head><body>
    <h2>🚧 Шлагбаум відкрито!</h2>
  )rawliteral";

  // 🔒 Додаємо кнопку лише якщо користувач — адмін
  if (isAdmin) {
    html += "<a href='/admin'>Адмін-панель</a>";
  }

  html += "<a href='/logout'>Вийти</a></body></html>";
  return html;
}

String getAdminPage() {
```

```
String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'><title>Admin</title><style>body{font-family:sans-serif;padding:20px;}table{border-collapse:collapse;width:100%;}td,th{border:1px solid #000;padding:5px;}input{margin:5px;}button{margin:5px;}</style></head><body>";
  html += "<h2>Адмін-панель</h2>";
  html += "<form action='/'><button>Назад</button></form>";
  html += "<form action='/logout'><button>Вийти</button></form>";

  html += "<h3>Додати користувача</h3>";
  html += "<form method='POST' action='/adduser'>";
  html += "Логін: <input name='newuser'><br>";
  html += "Пароль: <input name='newpass' type='password'><br>";
  html += "Адмін: <input type='checkbox' name='isadmin'><br>";
  html += "<button type='submit'>Додати</button></form>";

  html += "<h3>Журнал відкриттів шлагбаума</h3>";
  html += "<table><tr><th>Ім’я</th><th>Дата і час</th></tr>";
  for (int i = 0; i < logCount; i++) {
    html += "<tr><td>" + logs[i].username + "</td><td>" + logs[i].datetime + "</td></tr>";
  }
  html += "</table></body></html>";
  return html;
}

void handleRoot() {
if (!isAuthorized) {
server.send(200, "text/html", getLoginPage());
return;
}
barrier.write(90);
digitalWrite(redLed, LOW);
digitalWrite(greenLed, HIGH);
barrierIsOpen = true;
barrierOpenTime = millis();
```

```
if (logCount < MAX_LOGS) {
String timestamp = getCurrentTime();
logs[logCount++] = {currentUser, timestamp};
appendLogToSD(currentUser, timestamp);
}
server.send(200, "text/html", getMainPage());
}

void handleLogin() {
String u = server.arg("username");
String p = server.arg("password");
isAuthorized = false;
isAdmin = false;
for (int i = 0; i < userCount; i++) {
if (users[i].username == u && users[i].password == p) {
isAuthorized = true;
isAdmin = users[i].isAdmin;
currentUser = u;
break;
}
}
if (isAuthorized) {
server.sendHeader("Location", "/");
server.send(303);
} else {
server.send(200, "text/html", getFailPage());
}
}

void handleLogout() {
isAuthorized = false;
isAdmin = false;
currentUser = "";
barrier.write(0);
digitalWrite(redLed, HIGH);
```

```
digitalWrite(greenLed, LOW);
barrierIsOpen = false;
server.sendHeader("Location", "/");
server.send(303);
}

void handleAdmin() {
if (!isAuthorized || !isAdmin) {
server.send(403, "text/plain", "Доступ заборонено");
return;
}
server.send(200, "text/html", getAdminPage());
}

void handleAddUser() {
if (!isAuthorized || !isAdmin || userCount >= MAX_USERS) {
server.send(403, "text/plain", "Помилка додавання");
return;
}
String newuser = server.arg("newuser");
String newpass = server.arg("newpass");
bool adminFlag = server.hasArg("isadmin");
users[userCount++] = {newuser, newpass, adminFlag};
saveUserToSD(users[userCount - 1]);
server.sendHeader("Location", "/admin");
server.send(303);
}

void setup() {
Serial.begin(115200);
WiFi.softAP(ssid, password);
Serial.println("IP адреса: " + WiFi.softAPIP().toString());
delay(100);

Serial.print("Підключення до Wi-Fi: ");
```

```
  Serial.println(login);
  WiFi.begin(login, pasword);

  // Очікування підключення
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // Успіх
  Serial.println("");
  Serial.println("□ Підключено!");
  Serial.print("IP адреса: ");
  Serial.println(WiFi.localIP());

pinMode(redLed, OUTPUT);
pinMode(greenLed, OUTPUT);
digitalWrite(redLed, HIGH);
digitalWrite(greenLed, LOW);
barrier.attach(servoPin);
barrier.write(0);
configTime(3 * 3600, 0, "pool.ntp.org", "time.nist.gov");
if (!SD.begin(SD_CS)) {
Serial.println("X Не вдалося ініціалізувати SD-карту!");
} else {
Serial.println("□ SD-карта готова");
loadUsersFromSD();
loadLogsFromSD();
}
server.on("/", handleRoot);
server.on("/login", handleLogin);
server.on("/logout", handleLogout);
server.on("/admin", handleAdmin);
server.on("/adduser", handleAddUser);
server.begin()
```

```
void loop() {
        server.handleClient();
        if (barrierIsOpen && millis() - barrierOpenTime >= 30000) {
        barrier.write(0);
        digitalWrite(redLed, HIGH);
        digitalWrite(greenLed, LOW);
        barrierIsOpen = false;
        }
        }
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: Розробка мікроконтролерної web-системи управління бар’єром на платформі ESP32 WiFi

Обсяг пояснювальної записки 53 аркуші.

5 таблиці;

44 рисунка;

1 додатків.

Дата завершення роботи: ____ *червня 2025р*.

Підпис студента-дипломника _____Мацюк І.В.