Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

СИСТЕМА ВІДДАЛЕНОГО МОНІТОРИНГУ ВІДКРИТТЯ ДВЕРЕЙ НА БАЗІ КОНТРОЛЕРА ESP

REMOTE MONITORING SYSTEM FOR DOOR OPENING BASED ON ESP CONTROLLER

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

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІз-41
Гайдук Ярослав Анатолійович

(підпис)

Керівник:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 04 » червня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Тарас ТЕРЛЕЦЬКИЙ
« 10 » 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Гайдуку Ярославові Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Система віддаленого моніторингу відкриття дверей на базі контролера ESP*

Керівник роботи *к.т.н., доц. Лавренчук Світлана Василівна*

затверджені наказом закладу вищої освіти від «04» січня 2025 року № 11/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 10.06.2025р.

3. Вихідні дані до роботи *джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз проблем та наявних рішень
Вибір апаратно-програмного бази для проєкту
Проєктування та тестування системи моніторингу відкриття дверей
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Принцип дії магніту на геркон
Схема під'єднання геркона до плати ESP
Монтаж системи моніторингу
Узагальнений алгоритм роботи програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Лавренчук С.В.., доцент*		
Підбір складових та монтаж апаратної частини	*Лавренчук С.В.., доцент*		
Розробка програмного забезпечення системи моніторингу стану дверей	*Лавренчук С.В.., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,ст.викладач*		

7. Дата видачі завдання 10.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	до 10.02.2025 р.	Виконано
2.	*Вибір апаратної та програмної бази для проєкту*	до 02.03.2025 р.	Виконано
3.	*Проєктування та тестування системи моніторингу відкриття дверей*	до 02.04.2025 р.	Виконано
4.	*Висновки та пропозиції*	до 10.04.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 15.04.2025 р.	Виконано
6.	*Формування додатків*	до 02.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 10.05.2025 р.	Виконано
8.	*Представлення остаточного варіанту кваліфікаційної роботи бакалавра керівникові*	до 15.05.2025 р.	Виконано
9.	*Нормоконтроль*	до 30.05.2025 р.	Виконано
10	*Інструментальна перевірка на академічний плагіат*	до 03.06.2025 р	Виконано
11.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	до 10.06.2025 р.	Виконано

Здобувач вищої освіти _____ Гайдук Я.С.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Лавренчук С.В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Гайдук Я.А. Система віддаленого моніторингу відкриття дверей на базі контролера ESP. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, трьох додатків.

У кваліфікаційній роботі представлено розробку системи віддаленого моніторингу стану дверей на базі мікроконтролера ESP8266 з використанням геркона та Telegram-бота.

У першому розділі здійснено огляд сучасних охоронних систем, як комерційних, так і самостійно зібраних (DIY), а також проаналізовано можливості інтеграції з Telegram API.

У другому розділі обґрунтовано вибір сенсорів, мікроконтролера та представлено схему апаратної частини.

Третій розділ присвячено розробці програмного забезпечення: описано логіку роботи, реалізовано базову та покращену версії чат-бота, включено підтримку кількох користувачів та inline-клавіатуру.

У результаті створено функціональну IoT-систему, здатну своєчасно інформувати користувачів про зміни стану дверей через Telegram.

Ключові слова: ESP, NodeMCU, геркон, Telegram API, Arduino IDE.

ANNOTATION

Haiduk Ya. Remote monitoring system for door opening based on ESP controller. Manuscript.

Qualifying work of a bachelor of EP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

The qualification work consists of an introduction, three chapters, a conclusion, a list of sources used, and three appendices.

The qualification work presents the development of a system for remote door status monitoring based on the ESP8266 microcontroller using a reed switch and a Telegram bot.

The first chapter reviews modern security systems, both commercial and self-assembled (DIY), and analyzes the possibilities of integration with the Telegram API.

The second chapter justifies the choice of sensors and a microcontroller, and presents a diagram of the hardware part.

The third chapter is devoted to software development: the logic of the work is described, the basic and improved versions of the chatbot are implemented, support for multiple users is provided, and an inline keyboard is included.

As a result, a functional IoT system has been created that can promptly inform users about changes in the status of doors via Telegram.

Keywords: ESP, NodeMCU, reed switch, Telegram API, Arduino IDE.

ЗМІСТ

ВСТУП

В наш час досить інтенсивно розвиваються технології IoT (Інтернету речей), які дозволяють об’єднувати фізичні пристрої з можливостями передачі інформації через Інтернет для автоматизованого збору та обробки даних. Один з перспективних напрямів застосування таких технологій – створення та підтримка систем моніторингу та безпеки. Подібні рішення впроваджують у побуті, офісах та промисловості.

Особливо це актуально для об’єктів, що передбачають обмеження фізичного доступу, таких як сейфи, шафи для зберігання зброї або цінностей, гаражі [3] чи автомобілі, склади або інші технічні приміщення. Моніторинг стану дверей таких об’єктів дасть можливість вчасно дізнаватися про стан дверей, та, як наслідок, оперативно реагувати на інциденти вторгнення (несанкціонованого відкриття), що підвищить загальний рівень безпеки об’єкту моніторингу.

Використання мікроконтролерів дає змогу створювати недорогі, енергоефективні та функціональні пристрої для віддаленого моніторингу, які легко можна інтегрувати з мобільними додатками чи хмарними сервісами. Тому практичний досвід розробки системи віддаленого моніторингу відкриття дверей на базі мікроконтролера ESP буде корисним як з технічної, так і з практичної точки зору.

Основною ідеєю кваліфікаційної роботи є розробка та реалізація системи віддаленого моніторингу відкриття дверей, в основі якої використано мікроконтролер ESP8266. Також створено бот для оперативного сповіщення про стан дверей в Telegram авторизованим користувачам. Основним елементом, що реагує на стан (відчинено чи зачинено) дверей є геркон, який реагує на відкриття або закриття.

Мета роботи – розробити функціональну, недорогу, але функціональну систему моніторингу стану дверей з можливістю надсилання сповіщень у Telegram у випадку зміни стану.

Об’єкт дослідження – процес моніторингу фізичного стану дверей за допомогою мікроконтролера та інтернет-сервісів.

Предмет дослідження – апаратні та програмні засоби реалізації системи моніторингу з використанням ESP8266 та Telegram API.

Завдання, які необхідно виконати:

– проаналізувати існуючі рішення для дистанційного моніторингу об’єктів;

– розробити схему апаратної частини проєкту;

– створити програмне забезпечення для ESP;

– розробити Telegram-бот для взаємодії з користувачами;

– здійснити тестування і налагодження системи.

РОЗДІЛ 1
ОГЛЯД І АНАЛІЗ СИСТЕМ ВІДДАЛЕНОГО МОНІТОРИНГУ ДОСТУПУ

1.1 Типовий склад охоронної системи

Охоронна система – це комплекс апаратних засобів та програмного забезпечення, що призначений для виявлення несанкціонованого доступу до об’єкта (приміщення, сейфу, автомобіля) та своєчасного інформування власника, уповноважених осіб або служби охорони про такі події.

Типова охоронна система складається з центрального блоку управління, датчиків (руху, відкриття, розбиття скла, тощо), звукового сповіщення (сирени), також можуть додатково бути камери відеоспостереження, тривожні кнопки та інші пристрої.

Центральний блок управління – головний елемент системи, він отримує сигнали від датчиків, опрацьовує їх та передає керуючі сигнали на інші компоненти – сирену або монітор охорони.

Датчики потрібні для виявлення несанкціонованого доступу чи підозрілих подій. Найчастіше в охоронних системах використовують датчики руху, відкриття дверей або вікон, розбиття скла.

Сирена сповіщає про інцидент за допомогою гучного звуку.

Якщо є відеокамери, то вони записують відео подій, що відбуваються в приміщенні або на території.

Тривожні кнопки дають можливість швидко повідомити про небезпеку.

Деякі охоронні системи також можуть містити дистанційне керування, GSM-модуль, освітлювальні та інші пристрої, залежно від потреб користувача.

«Головним завданням системи безпеки є захист входів в приміщення за допомогою детекторів та інших пристроїв, керованих концентратором. В основному детектори розміщуються на дверях, вікнах нижніх поверхів або на тих вікнах, в які легко проникнути» [22].

Моніторинг – невід’ємна частина будь-якої системи охоронної сигналізації.

«Одним із ключових елементів цієї системи є моніторинг, який відіграє вирішальну роль у своєчасному виявленні та реагуванні на загрози. Моніторинг охоронної сигналізації охоплює безперервне спостереження за станом системи та її компонентів, а також обробку й аналіз сигналів, що надходять» [24].

Існують два основних методи моніторингу [24]:

– централізований моніторинг – коли всі сигнали від датчиків передають на центральний пульт управління, де постійно стежать оператори за станом системи та, за потреби, сповіщують правоохоронні органи або службу безпеки про інциденти вторгнення;

– дистанційний моніторинг дає можливість власникові отримувати повідомлення про зміну стану системи через мобільний застосунок, веб-інтерфейс або бот.

1.2 Аналіз існуючих рішень систем моніторингу стану дверей

1.1.1 Комерційні системи контролю та управління доступом до дверей

На ринку представлено чимало готових систем моніторингу стану дверей від таких виробників, як Ajax Systems [14], Hikvision [12], Xiaomi [9], Ring [21], U-Prox WDC [11] та ін.

Ajax Systems [14] пропонують датчики відкриття в двох версіях (Baseline та Superior):

– бездротовий з одним герконом – Ajax DoorProtect Jeweller;

– дротовий з двома герконами – Superior DoorProtect Fibra;

– дротовий датчик відчинення, удару та нахилу – Superior DoorProtect G3 Fibra;

– бездротовий датчик відчинення з двома герконами, акселерометром, сенсором удару і нахилу – Superior DoorProtect Plus Jeweller;

– бездротовий датчик відчинення з герконом і сенсорами удару та нахилу – DoorProtect Plus Jeweller;

– дротовий датчик відчинення з герконом і сенсорами удару та нахилу – Superior DoorProtect Plus Fibra;

– бездротовий датчик відчинення з герконом – Superior DoorProtect Jeweller.

Функціональні елементи Superior DoorProtect Fibra наведено на рисунку 1.1.

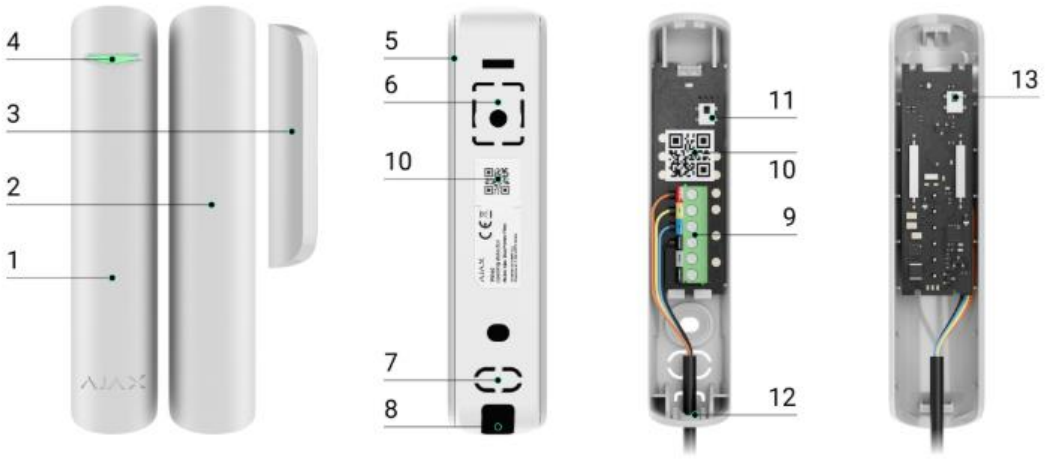

Рисунок 1.1 – Функціональні елементи Superior DoorProtect Fibra [16]

На рисунку 1.1 такі позначення:

– сам датчик Superior DoorProtect Fibra (1);

– великий магніт, спрацьовує на відстані до 2 см від датчика (2);

– малий магніт, спрацьовує на відстані до 1 см від датчика (3);

– індикатор-світлодіод (4);

– задня панель корпусу датчика, також виконує функцію кріплення (5);

– отвір для спрацьовування тампера у випадку спроби відірвати датчик від поверхні (6);

– отвір для виведення дротів крізь стіну (7);

– отвір для виведення дротів знизу датчика (8);

– клемна колодка під’єднання датчика (9);

– QR-код з ідентифікатором пристрою, для під’єднання до загальної системи Ajax (10);

– перша кнопка тампера (спрацьовує у випадку спроби відчинити корпус датчика) (11);

– отвір для кріплення (12);

– друга кнопка тампера (спрацьовує у випадку спроби відірвати датчик від поверхні) (13).

Всі датчики відкриття фірми Ajax є магнітними, входять до складу охоронної системи Ajax і зв’язуються з хабом Ajax за допомогою власного радіо-протоколу Jeweller. Він забезпечує захищений зв’язок з хабом Ajax (має шифрування з динамічним ключем, захист від глушіння на відстань до 1,2 км або більше – для версій Superior – завдяки технології стрибкоподібної зміни частоти). Вони також мають кнопку, яка повідомляє про спроби відриву датчика.

Деякі моделі (наприклад, DoorProtect Plus) додатково мають акселерометри для фіксації ударів та нахилів.

Ці датчики мають низьке енергоспоживання, здатні працювати до 5-7 років від однієї батареї типу CR123A.

Діапазон робочих температур: -10° С до +40° С при вологості до 75 %.

Hikvision [12] також пропонує магнітоконтактні датчики для виявлення відкриття, які інтегруються з охоронною системою AX PRO, керувати ними можна через додаток Hik-Connect. Приклад датчика фірми Hikvision наведено на рисунку 1.2.

Рисунок 1.2 – Магнітний детектор HIKVISION DS-PDMCS-EG2-WB

Фірма Xiaomi [9] теж пропонує датчики відкриття дверей та вікон, які можна інтегрувати з екосистемою розумного дому Mi Home чи Aqara. Основні моделі датчиків відкриття від Xiaomi: Xiaomi Mi Window and Door Sensor (працює за протоколом Zigbee, тому потрібен хаб Xiaomi Mi Smart Home Hub або інший Zigbee-сумісний хаб) та Xiaomi Mi Door and Window Sensor 2 (використовує протокол Bluetooth 5.1 BLE, можна підключити безпосередньо до смартфону для перегляду стану дверей чи вікон в межах дії Bluetooth, для віддаленого керування, отримання повідомлень та інтеграції з іншими розумними пристроями потрібен Bluetooth-шлюз).

Коли двері чи вікно, обладнані магнітоконтактним датчиком Ring [21], відчиняються, користувач отримує сповіщення на смартфон через додаток Ring. Вони підключаються до базової станції Ring Alarm за допомогою бездротового зв’язку.

У Луцьку працюють різні охоронні фірми, які надають послуги з встановлення та обслуговування охоронної сигналізації: Арсенал-СТ, Охоронні Системи, Алерт-Ком, Варта, Щит Волинь, Теплий Дім та інші. Всі вони використовують датчики відкриття та відповідні екосистеми, що описані вище.

Більшість із розглянутих вище систем доступу використовують:

– безпровідні магнітоконтактні датчики (геркони);

– центральні хаби, які приймають сигнал з датчика;

– мобільні застосунки для сповіщень та керування;

– шифрування та авторизацію користувачів для безпеки даних.

Переваги таких систем:

– висока надійність та захист;

– професійна технічна підтримка;

– інтеграція з іншими системами безпеки (відеоспостереження, сигналізація, тощо).

Недоліки:

– висока вартість;

– обмежена гнучкість налаштувань;

– закриті протоколи передачі даних.

1.1.2 DIY системи моніторингу

Абревіатура DIY (від Do It Yourself) означає сукупність того, що ми можемо зробити самі. Багато людей пробують розробляти системи моніторингу з використанням відкритих апаратних платформ, таких як ESP8266, ESP32, Arduino, Raspberry Pi. Такі рішення часто поєднуються з магнітними датчиками (герконами), Wi-Fi-з'єднанням, платформами повідомлень: Telegram, Blynk, MQTT, IFTTT.

Переваги DIY систем моніторингу: порівняно низька вартість компонентів, відкритий код та наявна велика база прикладів, велика гнучкість і здатність до масштабованості. Недоліки: необхідність самостійної розробки та налагодження, відсутність гарантії безпеки без додаткових заходів, потреба обслуговувати систему самостійно, залежність від стабільності Wi-Fi та сторонніх сервісів.

В роботі [1] розглядається система моніторингу стану дверей, створена на базі Інтернету речей та мікроконтролера ESP32. Запропонована система містить датчики руху, температури, модулі камер – все це під'єднано до мікроконтролера ESP32, роль блоку керування виконує Raspberry PI Zero, також є сенсорний екран та мембранна клавіатура (рисунок 1.3), дані передаються бездротовим способом, також використовується MQTT. GSM-модуль IoT-GA5-B забезпечує надсилання повідомлень на мобільний телефон власника. Скрипти, написані мовою Python, для зберігання даних автори використовують СУБД MariaDB. Вартість розробки (станом на 2021 рік) становила 230 євро.

Рисунок 1.3 – Апаратні компоненти системи моніторингу [1]

В статті [2] розглядається система моніторингу на основі технологій IoT. Автори створили додаток для ОС Android з назвою Door Security System, вони використовують MQTT для зв’язку між смартфоном та механізмом замка на дверях. Система містить також пасивний інфрачервоний датчик (PIR) та ESP32-CAM для фотографування. Зроблені фото потім надсилаються електронними листами мешканцям (використано протокол SMTP). Алгоритм роботи цієї системи моніторингу наведено на рисунку 1.4.

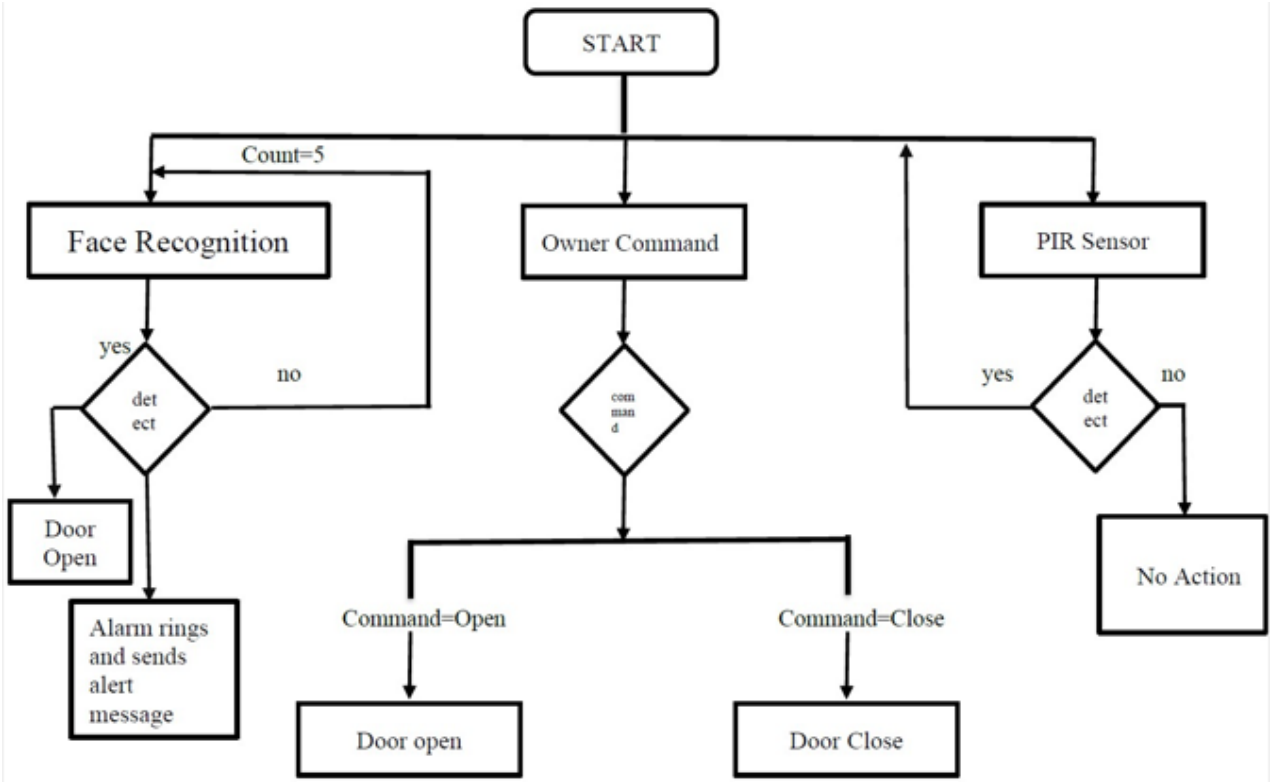

Рисунок 1.4 – Блок-схема функціонування системи моніторингу [2]

Власні розробки подібних систем можна знайти і в кваліфікаційних роботах здобувачів спеціальності комп’ютерна інженерія різних закладів освіти [17, 19].

Кійко Юрій [17]. створив електронний сейф на основі контролера Arduino, проте він має лише дві функції: зберігання цінностей та запис останнього відкриття сейфа. Сейф працює як автономний об’єкт на базі мікроконтролера, без під’єднання до IoT.

Лилик Іван [19] створив макет системи побутової охоронної сигналізації, структурна схема якого наведена на рисунку 1.5.

Система містить модуль GSM, який виконує фактично ті самі функції, що й мобільний телефон (надає змогу телефонувати та надсилати сповіщення), що дало можливість дистанційного моніторингу стану давачів. В якості мікроконтролера автор обрав модуль Arduino UNO.

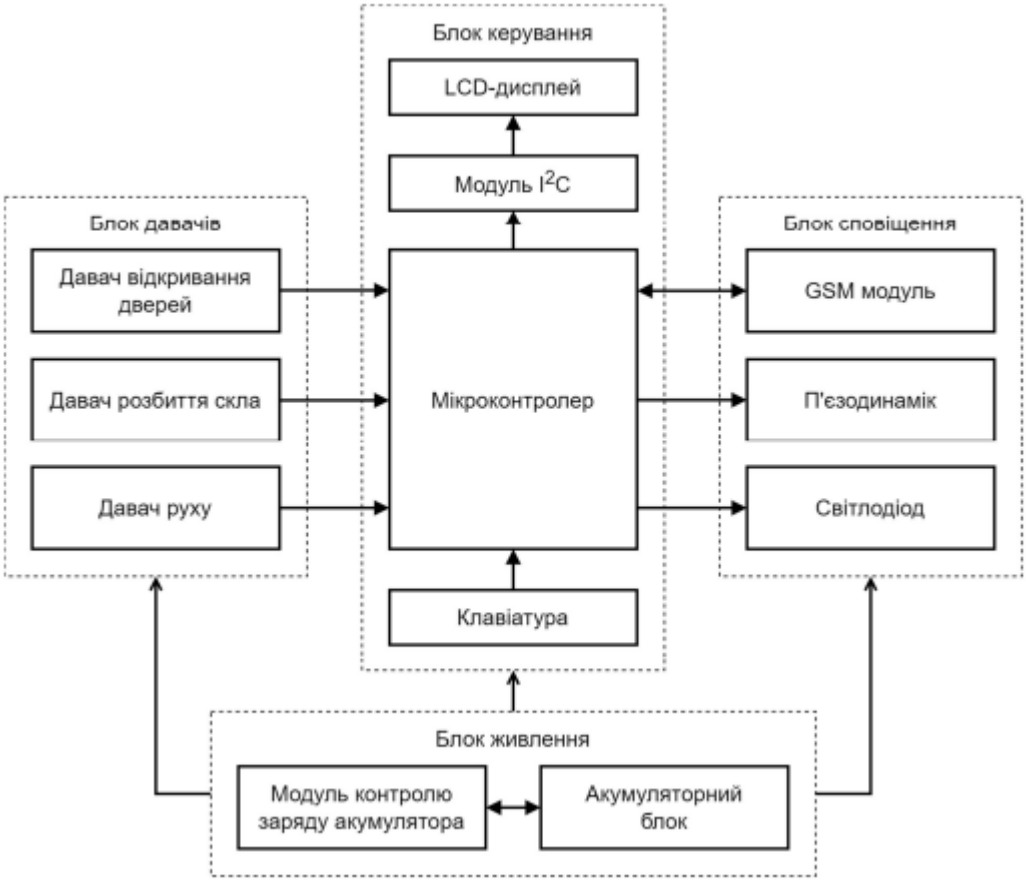

Рисунок 1.5 – Структурна схема системи побутової охоронної сигналізації [19]

В роботі [15] автори показують як можна використати мікроконтролер ESP8266 та PIR-датчик для створення простої системи моніторингу.

Індійські розробники [7] поєднали технології розпізнавання відбитків пальців з мікроконтролерами Arduino та модулем GSM для створення системи контролю доступу з дистанційним керуванням.

1.3 Функціональні вимоги до системи моніторингу

На основі аналізу комерційних та DIY-рішень можемо зробити висновок, що існують як дорогі професійні рішення, так і доступні саморобні проєкти з подібним функціоналом. Коли маємо обмежений бюджет та потребуємо простого й гнучкого рішення, доцільно використати мікроконтролер, який забезпечить підключення до мережі та дасть можливість взаємодіяти з API для надсилання сповіщень. Такий підхід дозволить реалізувати ефективну систему моніторингу з мінімальними витратами та достатньою функціональністю для особистих або навчальних цілей.

Функціонально система повинна:

– забезпечити моніторинг стану дверей у реальному часі;

– передавати сповіщення у Telegram при кожній зміні стану дверей типу: «Двері замкнено», «Двері відкрито»;

– підтримувати кількох користувачів (наприклад, авторизувати кількох членів сім’ї), при цьому не мати можливості доступу для всіх бажаючих;

– бот повинен вміти реагувати на команди, наприклад виводити поточний стан дверей за запитом користувача;

– архітектура системи повинна мати можливості для розширення в майбутньому: підключення нових датчиків, інтеграцію з системами розумного дому, логування подій в хмару, тощо.

1.4 Telegram API: можливості для інтеграції IoT-рішень

Завдяки Telegram API можна реалізувати зручну та надійну модель сповіщення про зміну стану об’єкта моніторингу, а також забезпечити інтерактивне керування.

Telegram Bot API дозволяє створювати ботів, які можуть: надсилати й отримувати повідомлення; обробляти команди користувачів; відправляти медіа,

локації, кнопки; взаємодіяти з webhooks; працювати з групами, каналами та приватними чатами.

Telegram-канали можуть використовуватися як централізовані майданчики для збору даних від великої кількості IoT-пристроїв.

Отримані через Telegram дані можуть бути інтегровані з іншими платформами для подальшого аналізу, візуалізації та прийняття рішень (наприклад, бази даних, системи аналітики).

Telegram надає механізми для ідентифікації користувачів, це можна використати для контролю доступу до керування IoT-пристроями. Всі повідомлення в Telegram зашифровані, що забезпечує певний рівень безпеки при передачі даних між IoT-пристроями та користувачами.

API Telegram дозволяє обробляти велику кількість запитів, що важливо для систем з великою кількістю підключених пристроїв.

Для інтеграції IoT-рішень з Telegram API необхідно:

– отримати API ID та API Hash у Telegram;

– використовувати одну з доступних бібліотек для роботи з Telegram API на потрібній мові програмування (наприклад, Telethon для Python, MadelineProto для PHP, Telegram Bot API для створення ботів);

– реалізувати логіку взаємодії між IoT-пристроями та Telegram API, включаючи обробку подій від пристроїв та надсилання команд керування.

Контролер надсилатиме запит до https://api.telegram.org/bot<token>/sendMessage, щоб повідомити користувача про зміну стану дверей.

Користувач може надсилати боту команди, наприклад:

– /status – отримати поточний стан дверей;

– /users – переглянути список авторизованих користувачів;

– /help – інструкція з використання бота.

Можна реалізувати фільтрацію користувачів за ID:

– лише авторизовані користувачі отримують повідомлення;

– небажані повідомлення від неавторизованих користувачів ігноруються.

Мікроконтролер може працювати через:

– періодичне опитування серверів Telegram для нових повідомлень (payload);

– системи з постійним підключенням до Інтернету можуть використовувати webhook-URL, який Telegram викликає при новому повідомленні.

Telegram API працює через HTTPS, що забезпечує захист даних при передачі. За потреби, можна реалізувати додаткове шифрування або хешування payload-запитів зі сторони мікроконтролера.

Переваги використання Telegram API у кваліфікаційній роботі:

– простий формат HTTP-запитів, доступна документація;

– Telegram API не потребує оплати за використання;

– хороша підтримка великої кількості бібліотек для різних мов: Python, Node.js, Arduino;

– працює на будь-якому пристрої з Telegram: Android, iOS, Web;

– інтерактивність – користувач може взаємодіяти з мікроконтролером через команди в чаті;

– можливість легко додати інші типи сенсорів або розширити функціонал.

Отже, Telegram API надає широкі можливості для інтеграції з IoT-рішеннями, дозволяє створювати різні сценарії взаємодії між фізичними пристроями та користувачами через месенджер.

РОЗДІЛ 2
ПІДБІР СКЛАДОВИХ АПАРАТНОЇ ЧАСТИНИ СИСТЕМИ МОНІТОРИНГУ ВІДКРИТТЯ ДВЕРЕЙ

2.1 Сенсори відкриття дверей

Датчик відкриття дверей – це пристрій, що фіксує фізичну зміну стану дверей (відчинено або зачинено) та передають сигнал контролеру чи іншим системам.

Існує досить багато різних видів сенсорів, які можна використовувати для фіксації стану відчиненості дверей, їх можна поділити на види за принципом дії (таблиця 2.1).

Таблиця 2.1 – Класифікація сенсорів для фіксації відкриття дверей

Вид	Принцип роботи	Переваги	Недоліки
Герконові	Складаються з двох частин: магніту та геркона. Коли двері зачинені – магніт знаходиться поряд із герконом, і контакт замкнений. Під час відкриття дверей магніт віддаляється – контакт розмикається, що фіксується як зміна стану	Дешеві й прості в підключенні. Надійна робота в побутових умовах. Не потребують живлення	Працюють лише на принципі замикання/розмикання (без проміжних станів). Потребують точного вирівнювання при монтажі
Інфрачервоні	Використовують інфрачервоне випромінювання для виявлення об’єктів. При перериванні ІЧ-променя (відкриття дверей) змінюється сигнал на виході	Можуть не потребувати фізичного контакту. Реагують на рух або присутність	Чутливість до зовнішнього освітлення та температури. Можливі хибні спрацювання

Продовження таблиці 2.1

Вид	Принцип роботи	Переваги	Недоліки
Ємнісні та індуктивні	Реагують на зміну електричного чи магнітного поля поблизу. Використовуються рідше для дверей, але можуть бути застосовані в специфічних рішеннях (наприклад, приховані монтажі)	Можуть працювати безконтактно. Довговічні	Дорожчі. Потребують налаштування чутливості
Гіроскопи, акселерометри	Встановлюються на дверне полотно. Фіксують зміну положення або кута нахилу, що свідчить про відкриття	Можуть виявляти нестандартні зміни (наприклад, силове відкривання). Працюють автономно	Складність в реалізації (потрібна обробка даних). Споживають більше енергії
Холлівські	Вимірюють магнітне поле. При наявності магніту поруч – змінюється вихідний сигнал	Більш точні й довговічні, ніж геркони. Є як аналогові, так і цифрові моделі	Потребують живлення. Трохи складніші в підключенні

Датчики розбиття скла хоч не є датчиками відчинення, проте їх часто встановлюють на дверях зі скляними вставками для виявлення спроб проникнення через розбиття скла. Вони реагують на характерний звук або вібрацію розбиття.

Приклад сенсора магнітного поля на основі геркона наведено на рисунку 2.1, ІЧ-сенсори – на рисунках 2.2–2.3.

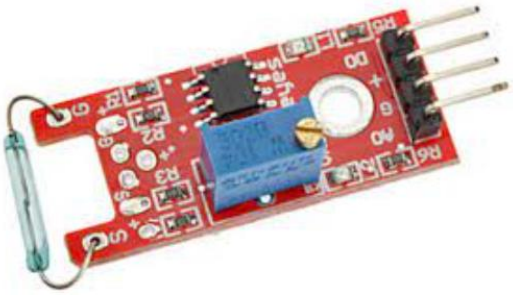

Рисунок 2.1 – Модуль з герконом KY-025 Magnetic Reed Switch [8]

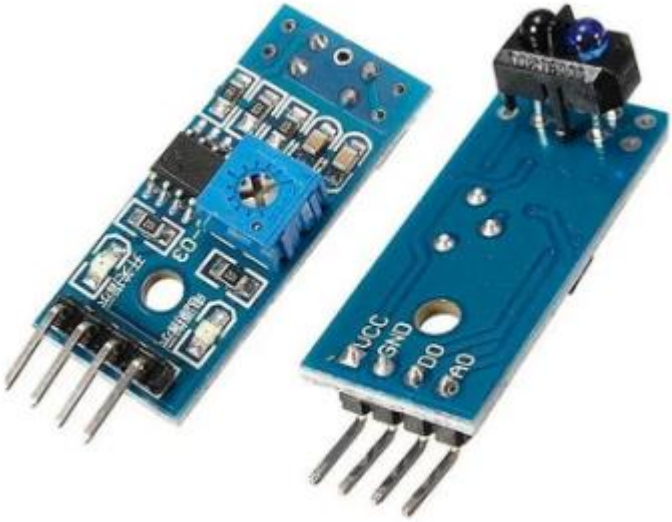

Рисунок 2.2 – Модуль ІЧ-відбивача TCRT5000 [20]

Рисунок 2.3 – HC-SR501 PIR [5]

На рисунку 2.4 наведено приклад акселерометра-гіроскопа – модуль позиціонування.

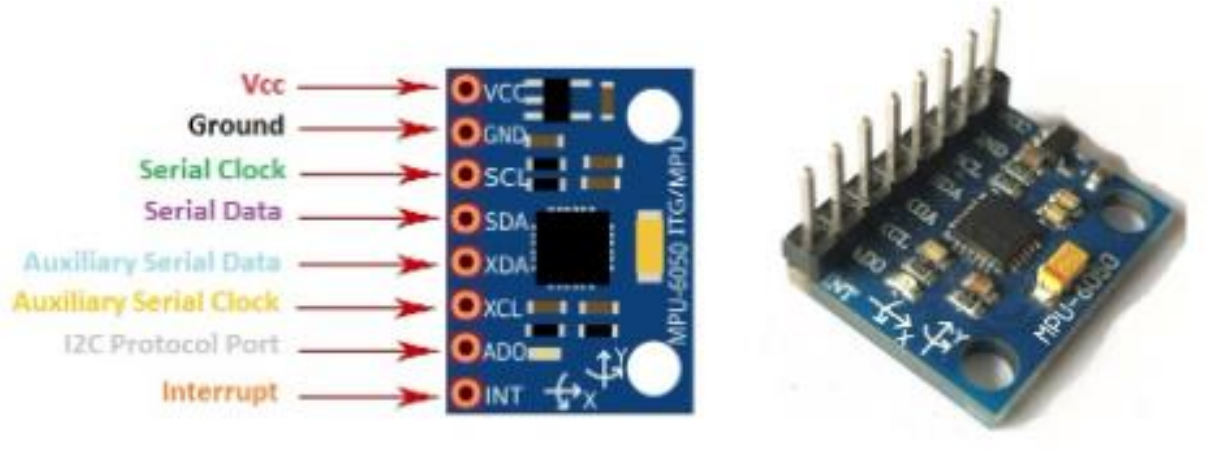

Рисунок 2.4 – MPU6050 [6]

На рисунку 2.5 представлено сенсор, що вимірює магнітне поле (датчик Холла). Цей датчик видає цифровий вихідний сигнал як високий, якщо він виявляє магнітне поле, інакше вихідний сигнал має низький рівень.

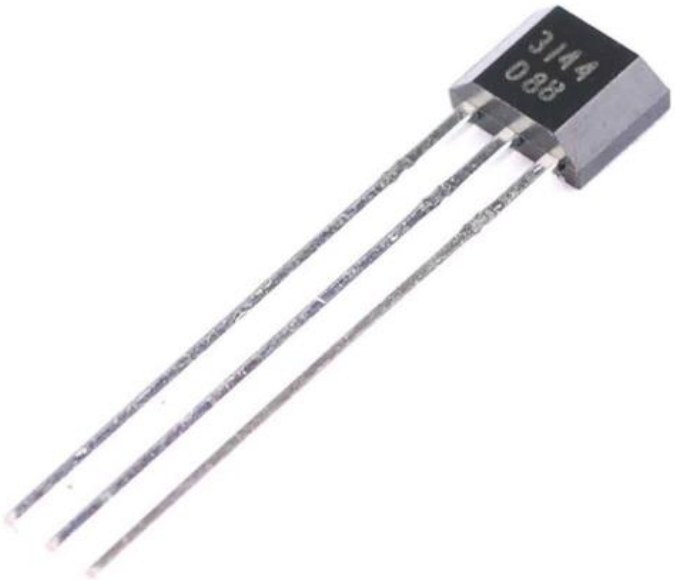

Рисунок 2.5 – Магнітний датчик Холла A3144 [13]

A3144 має невеликий розмір, чудову температурну стабільність і дуже швидку реакцію. Цей датчик також має вбудований захист від зворотної

полярності, щоб запобігти пошкодженню датчика у разі несправності. A3144 є енергоефективним і дуже стабільним датчиком, що робить його використання дуже економним. Для належного функціонування нам потрібно використовувати підтягувальний резистор на виході (рисунок 2.6).

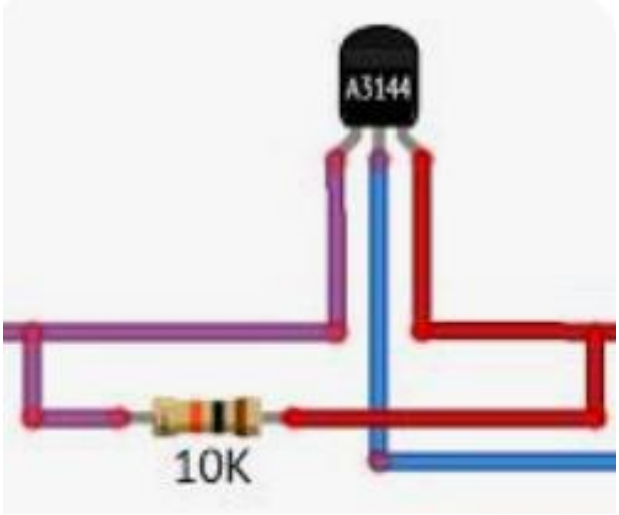

Рисунок 2.6 – Схема під’єднання сенсора A3144

Як бачимо з розділу 1, більшість датчиків відчинення дверей працюють за допомогою простого магнітного контакту. Вони складаються з двох частин:

– основний блок містить геркон (магнітокерований контакт) та передавач (у бездротових моделях). Він встановлюється на дверну раму або нерухому частину дверей;

– магніт встановлюється на самі двері, навпроти основного блоку.

Коли двері зачинені, магніт знаходиться близько до геркона, і магнітне поле утримує контакти геркона замкнутими, утворюючи замкнений електричний ланцюг (рисунок 2.7). Коли двері відчиняються, магніт віддаляється, магнітне поле зникає, і контакти геркона розмикаються, розриваючи ланцюг. Ця зміна стану фіксується датчиком і передається на центральний блок сигналізації або хаб розумного будинку.

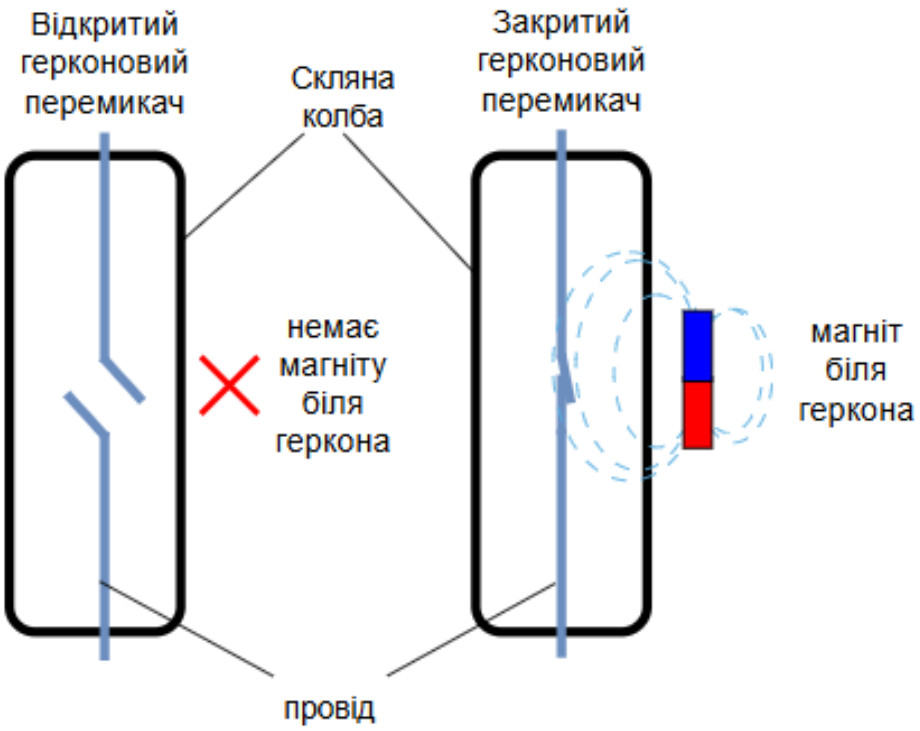

Рисунок 2.7 – Принцип дії магніту на герконовий сенсор

Отож, в ролі сенсора відкриття дверей було обрано геркон, яким має простий та ефективний принцип дії. Крім того, його легко можна поєднати з мікроконтролером.

2.2 Вибір мікроконтролера для системи моніторингу відкриття дверей

ESP8266 [23] – це недорогий мікроконтролер із вбудованим повним стеком протоколів TCP/IP, що дозволяє легко підключитися до Інтернету без використання додаткових модулів. Тому він дуже популярний в сфері Інтернету речей, адже він доступний за ціною, потребує не багато енергії та має вбудований Wi-Fi. Для нашого проєкту це дуже важливо, адже ми плануємо надсилати повідомлення у Telegram. Тому наявність вбудованого Wi-Fi-модуля на платі мікроконтролера спростить схему розробки, зменшить вартість, а також дасть можливість прямої інтеграції з хмарними API.

Основні характеристики:

– мікроконтролер містить 32-бітний процесор зі скороченим набором команд, який може працювати на частоті 80 МГц або 160 МГц;

– інтегрований Wi-Fi 2.4 ГГц (IEEE 802.11 b/g/n) з підтримкою аутентифікації WEP та WPA/WPA2. Може працювати в режимі станції (клієнта), точки доступу або обох режимів одночасно;

– зазвичай має 32 КБ оперативної пам’яті для команд, 32 КБ кеш-пам’яті для команд, 80 КБ оперативної пам’яті для даних користувача та зовнішню QSPI флеш-пам’ять (Б до 16 МБ);

– має ряд контактів загального призначення (GPIO), які можуть бути налаштовані для різних функцій, таких як цифровий вхід або вихід, ШІМ, I2C, SPI та UART. Точна кількість доступних GPIO пінів залежить від конкретного модуля ESP8266;

– більшість модулів ESP8266 мають один 10-бітний АЦП для зчитування аналогових рівнів напруги;

– модуль розроблений для пристроїв з живленням від батарей з різними режимами енергозбереження, включаючи глибокий сон.

Сам чіп ESP8266 зазвичай інтегрується в різні модулі та плати розробки для полегшення його використання. Поширені модулі та плати розробки ESP8266 [23]:

– ESP-01 – дуже простий та популярний модуль з невеликою кількістю GPIO пінів;

– ESP-12E/F – універсальніші модулі з більшою кількістю виведених GPIO пінів та часто інтегрованою чіп-антеною;

– NodeMCU – популярна плата розробки на базі ESP-12E з вбудованим USB-UART перетворювачем та зручним розташуванням для використання на макетній платі;

– WeMos D1 Mini – ще одна компактна та зручна для макетної плати розробки, схожа на NodeMCU.

Ці плати відрізняються за кількістю пінів, форм-фактором, функціональністю та зручністю використання, їх порівняння наведено в таблиці 2.2, яка була створена на основі документації до мікроконтролера [10].

Таблиця 2.2 – Порівняння ESP8266-модулів

Модуль / Параметр	ESP-01	ESP-12E / ESP-12F	NodeMCU (ESP-12E)	WeMos D1 Mini
Wi-Fi	так	так	так	так
Кількість GPIO	2 (GPIO0, GPIO2)	~9–11	~11	~11
Флеш-пам’ять	512 КБ / 1 МБ	до 4 МБ	до 4 МБ	до 4 МБ
Форм-фактор	дуже компактний	компактний, без плати	великий, з USB	компактний, з USB
Інтерфейси	UART	UART, SPI, I2C	UART, SPI, I2C	UART, SPI, I2C
USB-конвертер (UART)	немає, треба окремий	немає, треба окремий	вбудований	вбудований
Живлення	3,3 В	3,3 В	5 В через USB	5 В через USB
Зручність прототипування	ні	ні	так	так
Розміри	приблизно 14x24 мм	приблизно 24x16 мм	приблизно 50x25 мм	приблизно 34x25 мм
Призначення	прості проєкти / Wi-Fi-адаптер	IoT-модулі для вбудовування	прототипування або навчання	IoT-проєкти, де важлива компактність

На основі таблиці 2.2 можемо зробити висновки:

– ESP-01 має обмежені можливості (лише 2 GPIO), не підходить для складних проєктів, потребує додаткових компонентів;

– ESP-12E/F потребує пайки та окремого програматора, також він використовується як основа у NodeMCU/WeMos;

– NodeMCU має USB, стабілізатор 5 В→3,3 В, зручний для розробки;

– WeMos D1 Mini – компактний та повнофункціональний модуль, оптимальний для невеликих IoT-проєктів.

Отже, для нашого проєкту підійде модуль WeMos D1 Mini або ж NodeMCU. Було вирішено обрати той варіант, який є популярнішим в 2025 році. Для цього скористалися сервісом Google Trends [4], він показує, що плата NodeMCU набагато популярніша, ніж WeMos D1 Mini (рисунок 2.8).

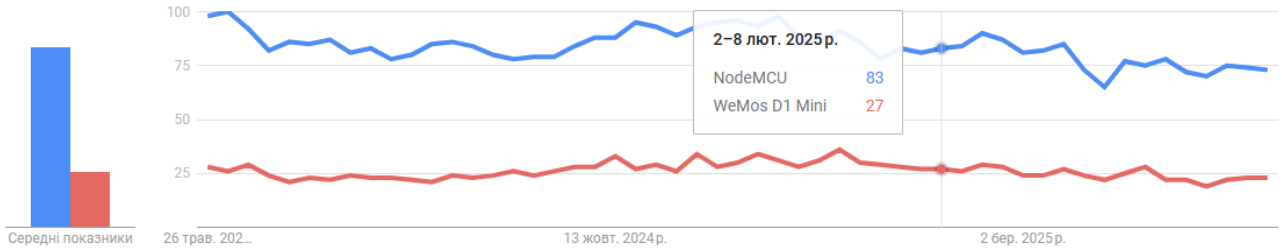

Рисунок 2.8 – Популярність модулів плати ESP8266 в 2024-2025 роках [4]

Тому було обрано повнорозмірний варіант плати, тобто модуль NodeMCU (рисунок 2.9).

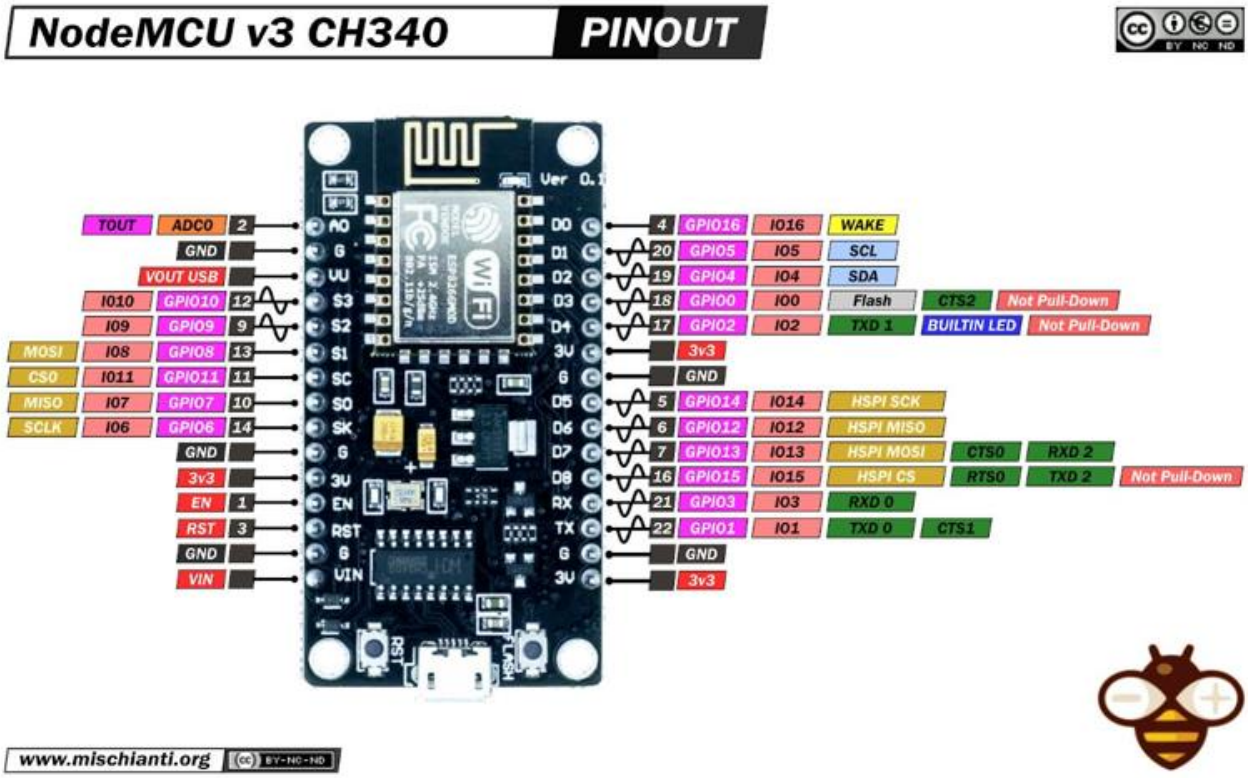

Рисунок 2.9 – Піни та їх функції плати NodeMCU [10]

Плата NodeMCU має:

а) піни для живлення:

– VIN – вхід для зовнішнього джерела живлення (5 В). Напруга проходить через вбудований регулятор та перетворюється до 3,3 В;

– VOUT USB – вихід 5 В, безпосередньо підключений до USB-роз'єму, доступний, коли плата живиться від USB;

– 3V3 – вихід стабілізованої напруги 3,3 В, використовується для живлення ESP8266 та зовнішніх 3,3 В пристроїв;

– GND – заземлення;

б) керуючі піни:

– EN (CH_PD) – вмикає та вимикає плату;

– RST (Reset) – перезавантажує ESP8266;

– WAKE – пін для виведення з глибокого сну;

в) піни UART (для обміну даними):

– TX (TX0, D10, GPIO1) – передача даних;

– RX (RX0, D9, GPIO3) – прийом даних;

г) піни SPI (для обміну даними):

– MOSI (D7, GPIO13);

– MISO (D6, GPIO12);

– SCLK (D5, GPIO14);

– CS (D8, GPIO15);

д) аналоговий вхід A0 – 10-бітний АЦП (діапазон 0-1V на чіпі ESP8266, але на платі NodeMCU v3 завдяки дільнику напруги може бути розширено до 0-3,3 В).

е) цифрові входи та виходи (GPIO Pins) – маркуються як D0-D10.

В таблиці 2.3 наведено відповідність між цими позначеннями на платі та реальними номерами GPIO ESP8266, які використовуються в програмному коді (наприклад, в Arduino IDE).

Таблиця 2.3 – Відповідність пнів на платі та в програмному середовищі

Позначення на платі	GPIO номер ESP8266	Функціональність
D0	GPIO16	може використовуватися для виходу з глибокого сну
D1	GPIO5	зазвичай використовується як SCL (Serial Clock Line) для I^2C
D2	GPIO4	зазвичай використовується як SDA (Serial Data Line) для I^2C
D3	GPIO0	важливий пін для режиму завантаження (boot mode), має бути HIGH під час нормальної роботи та LOW під час прошивки
D4	GPIO2	зазвичай підключений до вбудованого світлодіода на платі, також впливає на режим завантаження
D5	GPIO14	MOSI (Master Out Slave In) для SPI
D6	GPIO12	MISO (Master In Slave Out) для SPI
D7	GPIO13	SCK (Serial Clock) для SPI
D8	GPIO15	CS/SS (Chip Select/Slave Select) для SPI. Також може впливати на режим завантаження (повинен бути LOW під час нормальної роботи)
D9	GPIO3	RX0 – пін прийому UART0 (послідовний зв’язок). Використовується для налагодження та зв’язку з USB-UART перетворювачем, варто бути обережним при його використанні як звичайного GPIO
D10	GPIO1	TX0 – пін передачі UART0 (послідовний зв’язок). Використовується для налагодження та зв’язку з USB-UART перетворювачем, варто бути обережним при його використанні як звичайного GPIO

2.3 Проєктування апаратної частини системи моніторингу

На основі попередніх пунктів цього розділу можна стверджувати, що основними складовими апаратної частини пристрою є геркон, підключений до мікроконтролера ESP8266, який постійно приймає сигнали від геркона на піні A0 та, при змін стану, надсилає автоматичне сповіщення в Telegram усім заздалегідь визначеним авторизованим користувачам.

Схема підключення геркона до блоку ESP-12E, який лежить в основі модуля NodeMCU мікроконтролера ESP8266, наведена на рисунку 2.10.

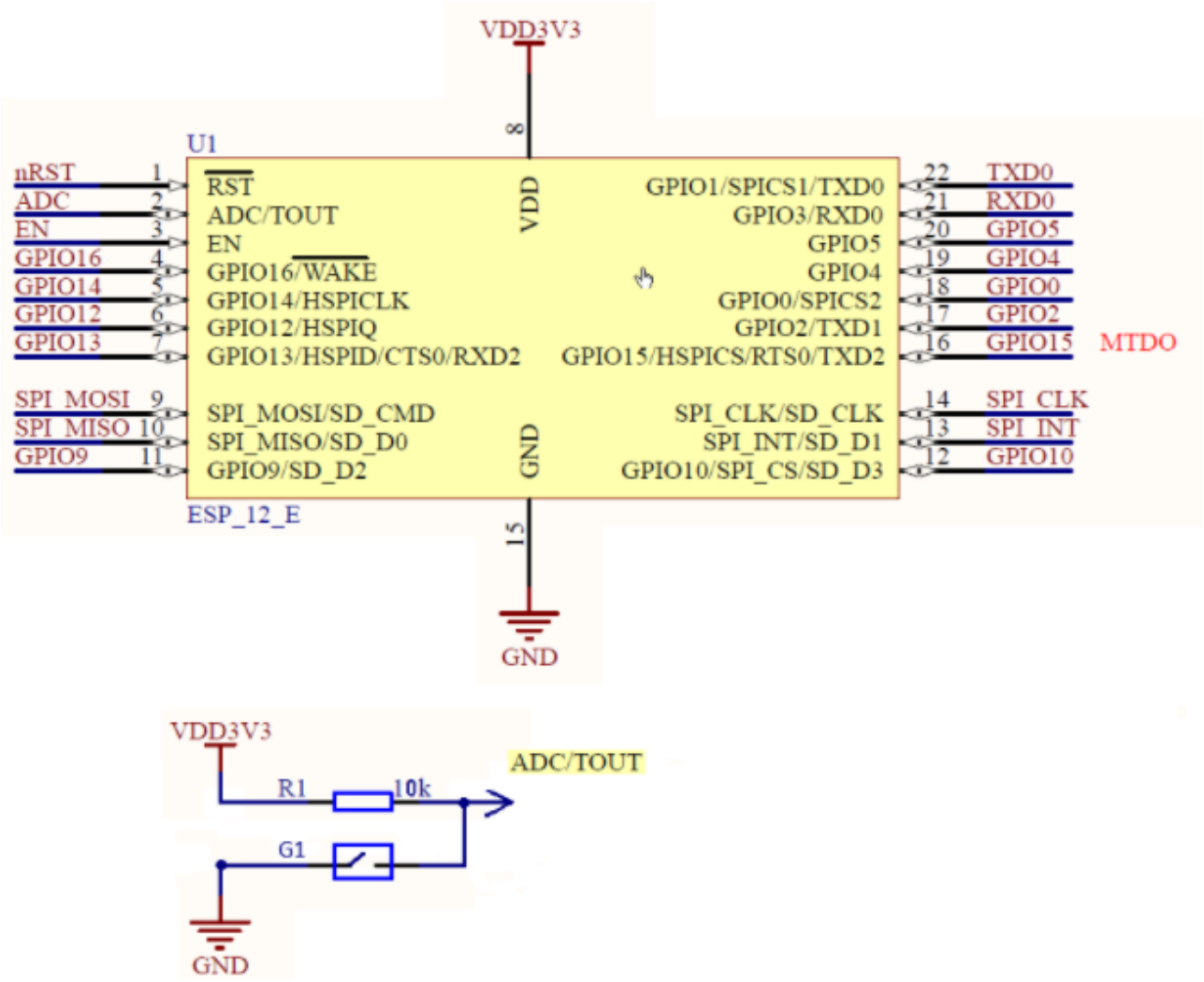

Рисунок 2.10 – Схема підключення геркона до блоку ESP-12E [10]

Для налагодження та симуляції (перевірки ідеї) було зібрано схему-прототип на макетній платі (рисунок 2.11).

Після того, як було перевірено функціональність прототипу, складові було інтегровано в двері іграшкового сейфу (рисунок 2.12). У верхній частині (над дверима) всередині сейфу прикріпили геркон (рисунок 2.13), в самих дверях розмістили магніт (рисунок 2.14).

Геркон з’єднано з мікроконтролером за схемою, наведеною вище (на рисунку 2.10).

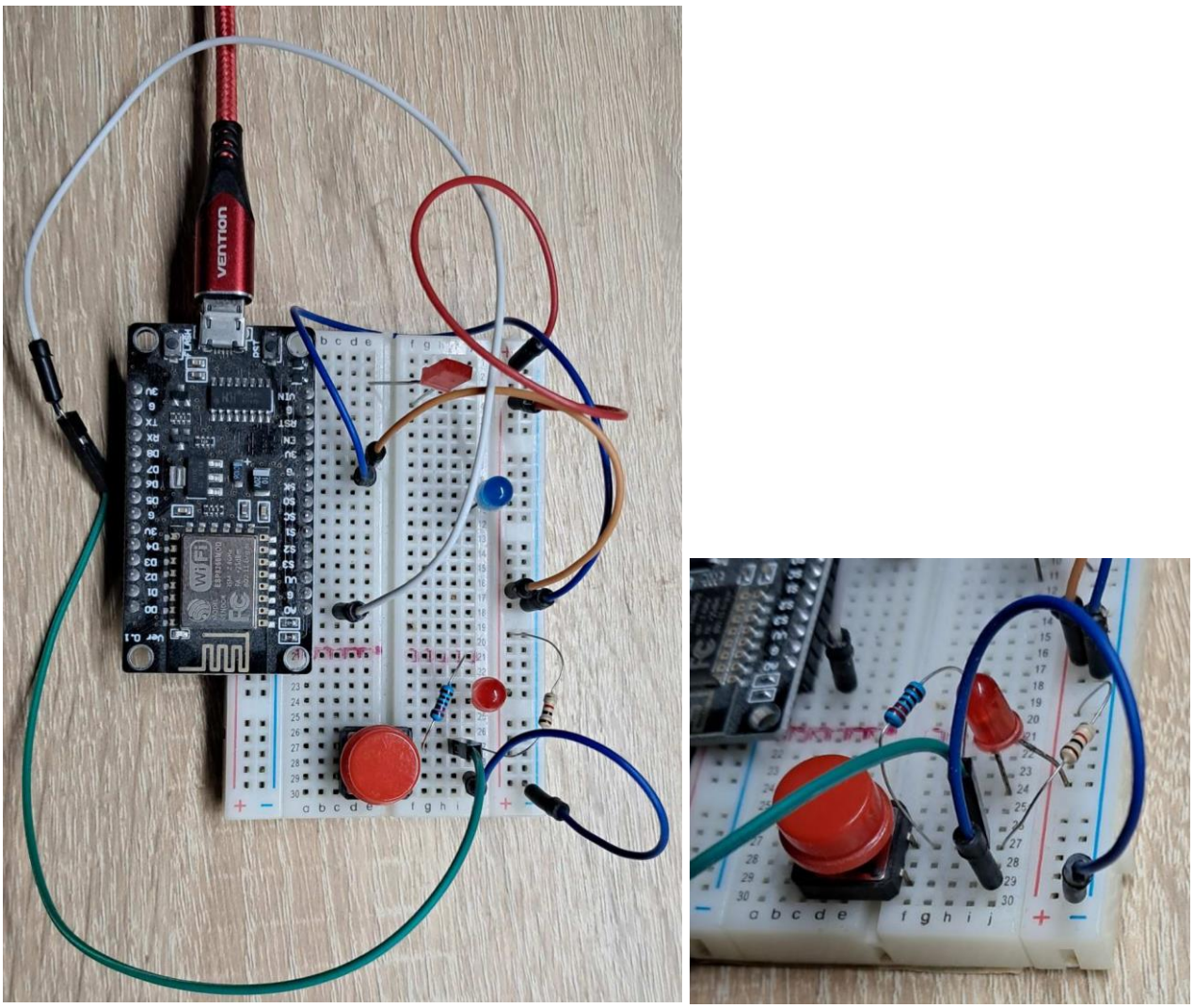

Рисунок 2.11 – Розробка та макетування пристрою

Рисунок 2.12 – Зовнішній вигляд сейфу

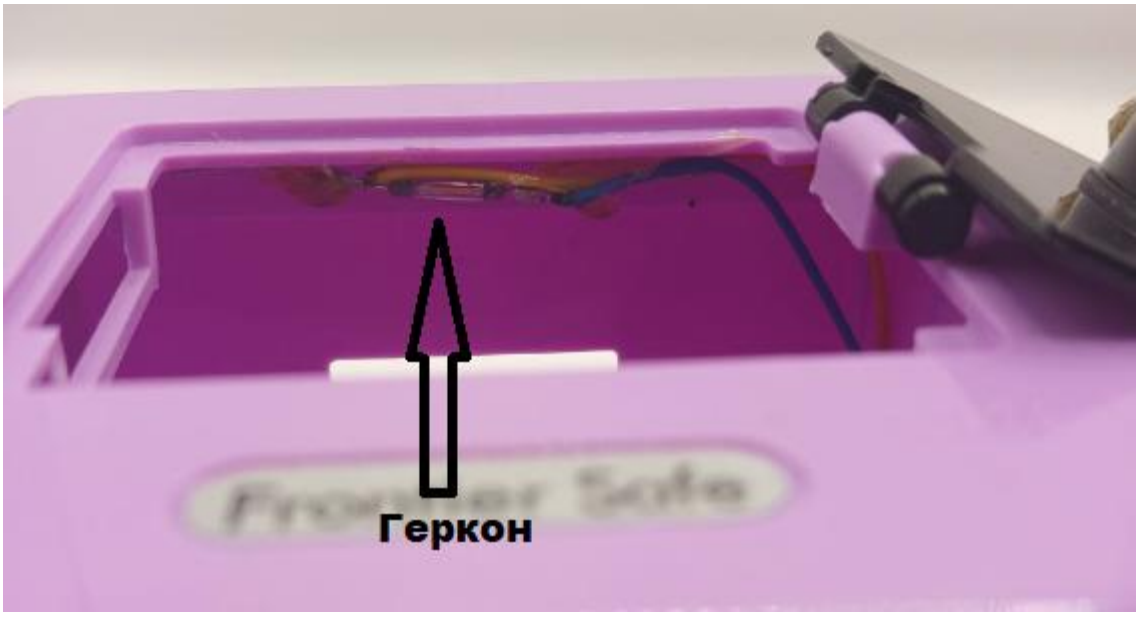

Рисунок 2.13 – Розміщення геркона в сейфі

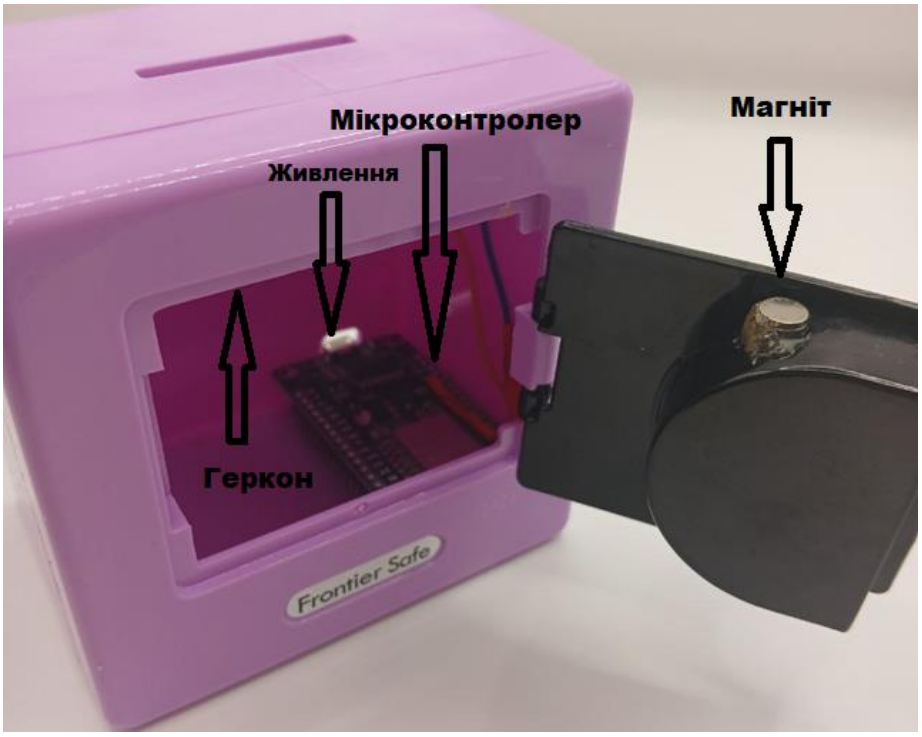

Рисунок 2.14 – Монтаж складових системи моніторингу стану дверей в сейфі

Для того, щоб система працювала, їй потрібне живлення (5 В), тому в задній стінці сейфу зроблено функціональний отвір – для підключення джерела живлення. Роль джерела живлення може виконувати будь-який пристрій, здатний видавати 5 В, найпростіше – повербанк.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ МОНІТОРИНГУ СТАНУ ДВЕРЕЙ

3.1 Вибір мови та середовища для програмування

Мікроконтролер ESP8266, в тому числі і на платі NodeMCU, програмують подібно до інших мікроконтролерів – за допомогою середовища розробки та підключення до комп'ютера через USB-порту.

Для підключення до комп'ютера використовуємо кабель USB – Micro USB. NodeMCU автоматично з'являється як COM-порт після встановлення драйверів.

ESP8266 можна програмувати кількома способами:

– використовуючи ядро ESP8266 для Arduino, ми можемо писати код мовою C++, використовувати бібліотеки Arduino та середовище розробки Arduino IDE. Це досить популярний вибір через наявну широку підтримку спільноти та доступні бібліотеки;

– використовуючи офіційний SDK (Espressif IoT Development Framework або скорочено ESP-IDF) – це потужніше та гнучкіше середовище для розробки застосунків мовою C/C++ з операційною системою реального часу;

– використовуючи MicroPython – легку та ефективну реалізацію мови Python 3, яка оптимізована для мікроконтролерів (часто використовується для різних IoT-проєктів);

– прошивка NodeMCU дозволяє також програмувати ESP8266 за допомогою скриптової мови Lua;

– ESP8266 також можна використовувати як Wi-Fi модем, що управляється іншим мікроконтролером за допомогою AT-команд (назва походить від агнл. ATtention) через послідовний інтерфейс.

Проаналізувавши можливі середовища розробки, стало зрозуміло що варто використовувати мову C/C++, адже навіть Lua написана на C. Для нашого завдання підійде середовище Arduino IDE. Такий вибір зроблений тому, що це

середовище дозволить легко взаємодіяти з периферією (сенсорами) та має готові бібліотеки для роботи з Telegram API.

Arduino IDE для створення прошивки в цій роботі має такі переваги:

– має інтуїтивно зрозумілий інтерфейс, чудово підходить для початківців;

– середовище повністю підтримує NodeMCU (після встановлення плати через Boards Manager);

– існує багато готових прикладів для ESP8266, які можна адаптувати під свої потреби;

– має багато готових бібліотек;

– не потрібно складних налаштувань конфігурації;

– Arduino IDE підтримує бібліотеки для роботи з Telegram (наприклад, UniversalTelegramBot);

– працює на Windows, macOS, Linux, що заберпечує кросплатформеність для розробника.

3.2 Загальна логіка роботи програми

Програма-бот DoorSafeBot призначена для моніторингу стану дверей (двері іграшкового сейфу) за допомогою геркона, підключеного до мікроконтролера ESP8266, та надсилання сповіщень про зміни стану в Telegram.

ESP8266 постійно приймає сигнали від геркона через пін A0. Щойно стан геркона зміниться (замкнено/розімкнено, що еквівалентно зачинено/відчинено для дверей сейфу) і ця зміна стабілізується (після debounce-затримки), ESP8266 надсилає автоматичне сповіщення в Telegram усім заздалегідь визначеним авторизованим користувачам.

Також бот у Telegram постійно перевіряє нові повідомлення та команди від користувачів. Якщо авторизований користувач надсилає команду (наприклад, /status або натискає кнопку «СТАН ДВЕРЕЙ»), бот негайно відповідає поточним станом дверей. Це створює систему моніторингу дверей з інтеграцією в Telegram.

Для автоматичного надсилання змін статусу геркона дверей (ON/OFF) в Telegram через ESP8266, нам потрібно:

1) створити бота в Telegram через BotFather і отримати токен;

2) отримати chat ID користувача, якому потрібно надсилати повідомлення;

3) написати код для ESP8266, який буде моніторити стан геркона і надсилати повідомлення до Telegram при зміні.

Алгоритм створення Telegram-бота:

1) відкрити Telegram і знайти BotFather – @BotFather (рисунок 3.1);

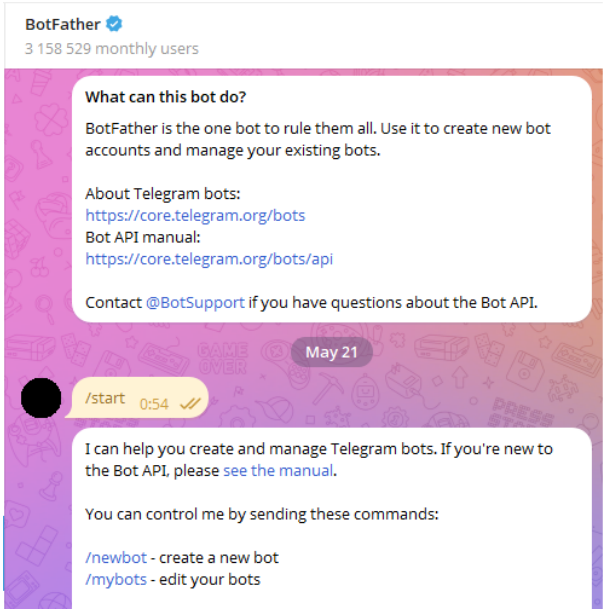

Рисунок 3.1 – Створення бота через @BotFather

2) виконати команду /newbot і дотримуватися інструкцій;

3) після створення бота отримано токен (рисунок 3.2), який потім використаємо в прошивці плати (лістинг3.1).

Лістинг 3.1 – Використання токену з Telegram в прошивці плати

```
#define BOT_TOKEN «7552727215:AAHZ7OjGILwG0PAn_3eMT4ttLz1 _mqPzcG8» // Slava-бот
…
UniversalTelegramBot bot(BOT_TOKEN, client);
```

Кінець лістингу 3.1

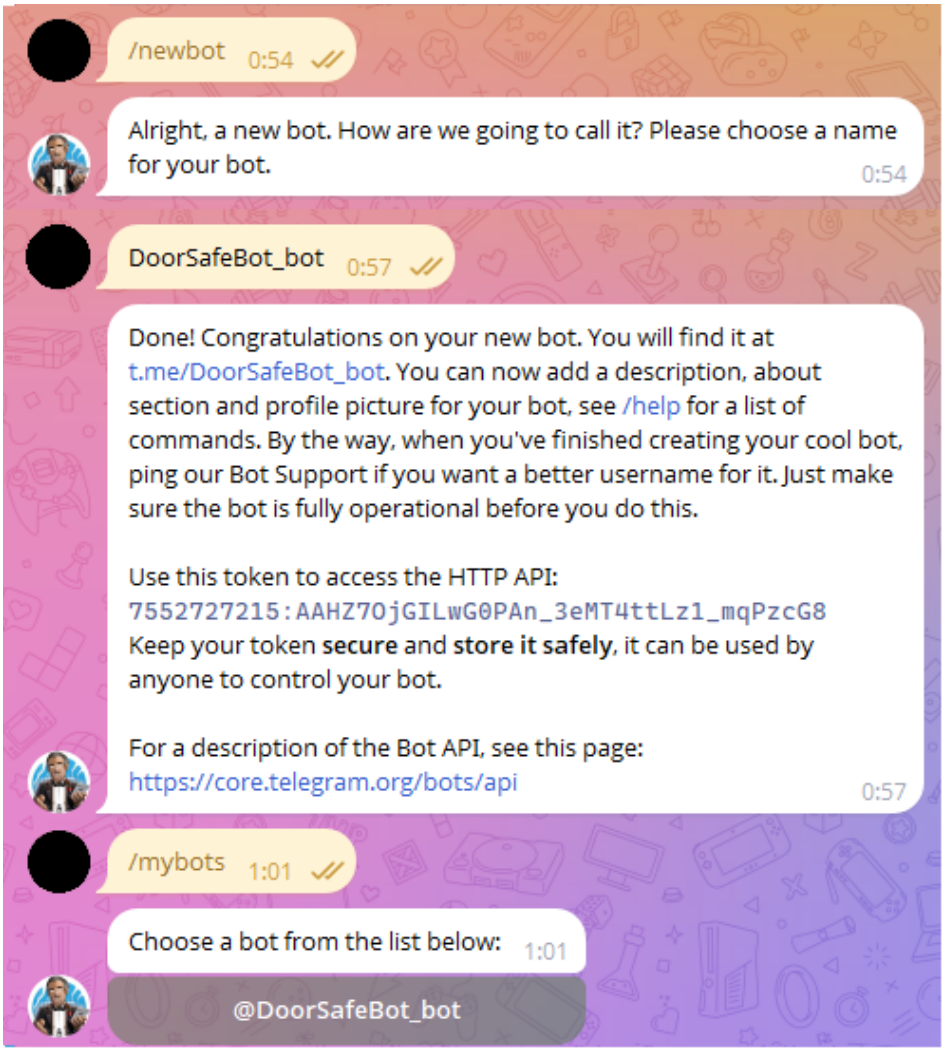

Рисунок 3.2 – Отримання токена для бота

Для того, щоб сторонні не могли дізнатися про стан дверей, система моніторингу повинна надавати інформацію лише зареєстрованим користувачам, для цього нам потрібно знати chat ID. Це можна зробити двома способами.

Спосіб 1. Знайти в телеграм створений нами бот (рисунок 3.3) та надіслати йому спочатку /start, а потім /id (рисунок 3.4).

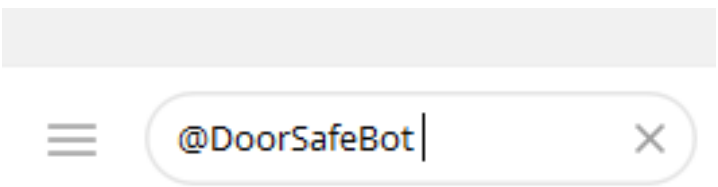

Рисунок 3.3 – Пошук свого бота

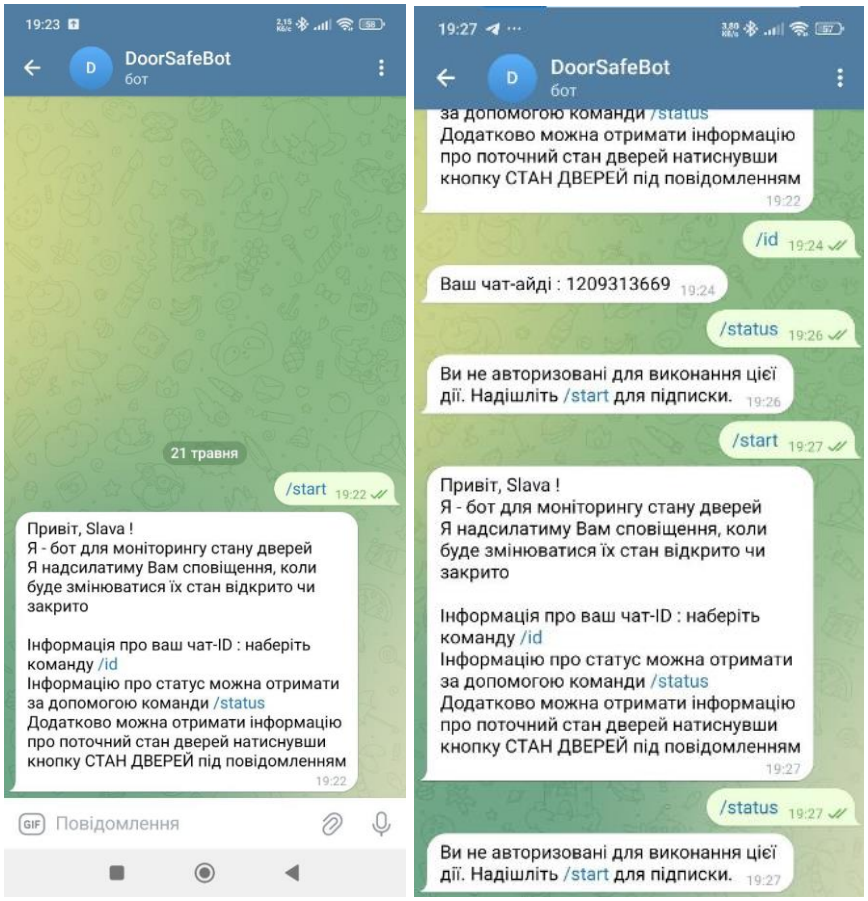

Рисунок 3.4 – Пошук свого chat ID

Спосіб 2 – використати метод Telegram API getUpdates, щоб отримати chat ID, для цього слід відкрити браузер і перейти за посиланням https://api.telegram.org/bot7552727215:AAHZ7OjGILwG0PAn_3eMT4ttLz1_mqPzcG8/getUpdates, тоді надіслати будь-яке повідомлення своєму боту, оновити сторінку браузера і побачимо відповідь у json-форматі, з якого можна дізнатися свій chat ID (поле chat.id у відповіді).

З рисунку 3.4 ми бачимо, що кнопка /status поки не повертає нам інформації про стан дверей, адже користувач з потрібним chat ID ще не відомий мікроконтролеру. Таким чином, не всі бажаючі зможуть отримати інформацію про стан дверей, навіть якщо знайдуть наш бот (див. рисунок 3.3).

Узагальнений алгоритм роботи програми наведено в додатку А.

3.3 Створення простого бота для одного користувача

Початкові налаштування для геркона (рисунок 3.5) здійснено в глобальній області пам’яті мікроконтролера.

```
// налаштування для геркона
const int reedSwitchPin = A0; // АЦП-вивід A0
int reedSwitchState = -1;     /* змінна для зберігання поточного стану геркона (-1 для першого читання)*/
int lastReedSwitchState = -1; /* змінна для зберігання попереднього стану геркона*/

/* поріг для визначення стану геркона
якщо геркон замкнений (магніт є) -> GND -> AnalogRead буде близьким до нуля
якщо геркон розімкнений (магніту немає) -> VCC (через PULLUP) -> AnalogRead буде близьким до 1023*/
const int analogThreshold = 500; // типовий початковий поріг

unsigned long lastSendTime = 0;   // час останнього надсилання повідомлення
const long debounceDelay = 50;    /* затримка для уникнення "стрибків" контактів геркона*/
```

Рисунок 3.5 – Початкові налаштування параметрів та змінних для геркона

Як відомо, будь-який мікроконтролер має як мінімум дві функції: початкова ініціалізація – функція setup() та основний цикл програми – функція loop().

На етапі початкової ініціалізації (функція setup()) модуль підключається до мережі Wi-Fi, використовуючи задані ім’я та пароль мережі, тут також встановлюється безпечне з’єднання для Telegram (ми тимчасово відключаючи перевірку сертифікатів, на етапі початкового тестування), визначається початковий стан геркона (замкнено/розімкнено) на основі зчитування значення ADC (рисунок 3.6).

```
1  void setup() {
2      delay(10);
3
4      // підключення до Wi-Fi
5      WiFi.begin(wifi_ssid, wifi_password);
6      client.setInsecure(); // відключаємо перевірку сертифікатів
7
8      while (WiFi.status() != WL_CONNECTED) {
9          delay(500);
10     }
11
12     // перше читання стану геркона
13     int analogValue = analogRead(reedSwitchPin);
14     reedSwitchState = (analogValue < analogThreshold) ? LOW : HIGH;
15     lastReedSwitchState = reedSwitchState;
16 }
```

Рисунок 3.6 – Початкова ініціалізація модуля, перше читання стану геркона

В основному циклі постійно здійснюється моніторинг стану геркона (рисунок А.2). Якщо виявлено зміну стану (після фільтрації «стрибків» та «дрижання» контактів за допомогою debounceDelay), надсилається відповідне сповіщення. Наприклад, при зміні стану дверей (відкриті/закриті), програма-бот надсилає просте текстове повідомлення («ДВЕРІ ЗАМКНЕНО» або «ДВЕРІ ВІДКРИТО») поки єдиному авторизованому користувачеві Telegram відповідно до прописаного chat ID користувача (рисунок 3.7).

```
void loop() {
...
// читаємо поточне аналогове значення з геркона
    int currAnalogValue = analogRead(reedSwitchPin);
    int currReedSwitchState = (currAnalogValue < analogThreshold) ? LOW : HIGH;

    // перевіряємо, чи змінився стан геркона і чи пройшла затримка
    if (currReedSwitchState != lastReedSwitchState && millis() - lastSendTime > debounceDelay) {
        delay(debounceDelay); // додаткова затримка для стабілізації
        currAnalogValue = analogRead(reedSwitchPin);
        currReedSwitchState = (currAnalogValue < analogThreshold) ? LOW : HIGH;

        if (currReedSwitchState != lastReedSwitchState) {
            lastReedSwitchState = currReedSwitchState;
            lastSendTime = millis(); // оновлюємо час останнього надсилання

            String messageText;
            if (lastReedSwitchState == LOW) {
                messageText = "ДВЕРІ ЗАМКНЕНО";
            } else {
                messageText = "ДВЕРІ ВІДКРИТО";
            }
            bot.sendMessage(AUTHORIZED_CHAT_ID, messageText, "");
        }
    }
...
}
```

Рисунок 3.7 – Моніторинг стану геркона в основному циклі прошивки мікроконтролера

Також постійно здійснюється перевірка на наявність нових повідомлень від Telegram та передача їх на обробку функції handleNewMessages() – рисунок А.3. Для кожного отриманого повідомлення спочатку перевіряється чи воно надійшло від єдиного авторизованого користувача (визначеного AUTHORIZED_CHAT_ID). Якщо ні, надсилаємо повідомлення про відмову в доступі.

Чат реагує на такі текстові команди:

– /start – надсилає привітальне повідомлення з описом функціоналу бота та доступними командами;

– /stop – повідомляє про завершення дії;

– /status – надсилає поточний стан дверей користувачеві («ДВЕРІ ЗАМКНЕНО» або «ДВЕРІ ВІДКРИТО»);

– /id – відповідає користувачеві його власний Chat ID.

На етапі тестування прошивки спочатку потрібно було переконатися, що система реагує на геркон (рисунок 3.8), а потім вже робити «правильний» інтерфейс (рисунок 3.9).

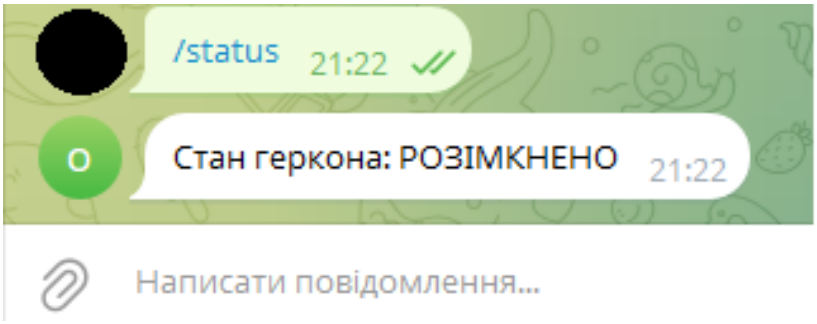

Рисунок 3.8 – Повідомлення в Telegram про стан геркона

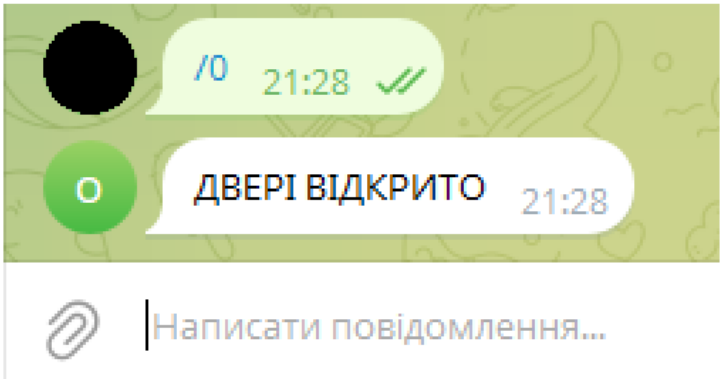

Рисунок 3.9 – Повідомлення в Telegram про стан дверей

Код прошивки повністю (з коментарями) наведений в додатку Б, він є спрощеним, призначений для тестування самої ідеї і слугує кістяком для подальшої розробки.

3.4 Розширення функціональності телеграм-бота

3.4.1 Створення inline-клавіатури

Для покращення взаємодії із ботом варто під кожним згенерованим на подію відкриття дверей повідомленням додати візуальну кнопку, яка виконуватиме дію, еквівалентну команді /status. Щоб створити візуальну кнопку в чаті Telegram нам потрібно скористатися функцією надсилання повідомлень з inline-клавіатурою та обробляти так звані «callback queries» (запити зворотного виклику), які генеруються при натисканні такої кнопки.

Це можна реалізувати з використанням бібліотеки Universal Arduino Telegram Bot.

Перший крок – створення повідомлення в форматі JSON для inline-клавіатури (оскільки Inline-клавіатура в Telegram задається саме у форматі JSON). Для кнопки «СТАН ДВЕРЕЙ» з даними зворотного виклику «get_status» JSON виглядатиме так, як наведено в лістингу 3.2.

Лістинг 3.2 – JSON для кнопки inline-клавіатури

```
JSON
[[{"text":"СТАН ДВЕРЕЙ", "callback_data":"get_status"}]]
```

Кінець лістингу 3.2

Це масив масивів, де внутрішній масив представляє рядок кнопок, а об’єкт у ньому – саму кнопку з текстом, який відображається (text), та даними, які будуть надіслані боту при натисканні (callback_data).

Другий крок – надсилання повідомлень з inline-клавіатури. Замість звичайного методу bot.sendMessage(), використаємо метод bot.sendMessageWithInlineKeyboard(). Ця функція приймає додатковий аргумент – рядок з JSON представленням клавіатури.

На третьому кроці здійснимо зворотній зв’язок – обробку callback queries у handleNewMessages(). Для цього у функцію handleNewMessages потрібно додати перевірку типу повідомлення. Якщо тип повідомлення callback_query, ми

обробляємо його як натискання кнопки. Дані зворотного виклику доступні через bot.messages[i].text.

Тепер, коли бот надсилатиме повідомлення про зміну стану геркона (дверей), під повідомленням буде кнопка «СТАН ДВЕРЕЙ». Натискання на цю кнопку викличе функцію handleNewMessages, яка розпізнає callback query з даними «get_status» і надішле користувачеві поточний стан геркона, тобто дверей (рисунок 3.10).

Рисунок 3.10 – Кнопка як елемент меню в Telegram

Отже, тепер ми можемо не лише надсилати текстові повідомлення ботом, але й маємо елемент управління – кнопку.

3.4.2 Багатокористувацький режим роботи

Розширимо функціональність, щоб бот міг працювати з багатьма користувачами та надсилати їм сповіщення (наприклад, для випадку, коли кілька членів сім'ї хочуть знати інформацію про факт відчинення сейфу).

Для цього нам буде потрібно:

– зберігати ідентифікатори чатів (Chat ID) користувачів, які хочуть отримувати сповіщення у масиві даних;

– змінити логіку надсилання повідомлень, щоб вони відправлялися всім підписаним користувачам.

У цьому прикладі ми зберігаємо Chat ID у масиві, який знаходиться в оперативній пам'яті. Це означає, що список підписаних користувачів буде кожного разу перезавантажено у пам'ять при перезавантаженні плати. Зауважимо, що ESP8266 має обмежену оперативну пам'ять. Зберігання великої кількості Chat ID безпосередньо в пам'яті може призвести до проблем, разом з тим сама специфіка проєкту не передбачає масової підписки на бот. Тому для конкретно цього проєкту такий підхід себе виправдовує, а для інших задач потрібно буде шукати інший спосіб зберігання даних. Наприклад, для збереження списку між перезавантаженнями можна використовувати файлову систему (SPIFFS/LittleFS) або EEPROM, але це значно ускладнить робочий код без додавання функціональності. У цьому прикладі ми обмежимося зберіганням масиву у RAM (лістинг 3.3).

Лістинг 3.3 – Масив для зберігання ID авторизованих користувачів

```
// масив для зберігання усіх Chat IDs для підписаних користувачів
const String authorized_chat_ids[] = {
 // "**********" , // ID користувача /
 "1209313669", // ID першого користувача / Slava
 "744703240"  // ID другого користувача / Svitlana
};
```

Кінець лістингу 3.3

Тепер вже два користувачі зможуть використовувати бот (рисунок 3.11).

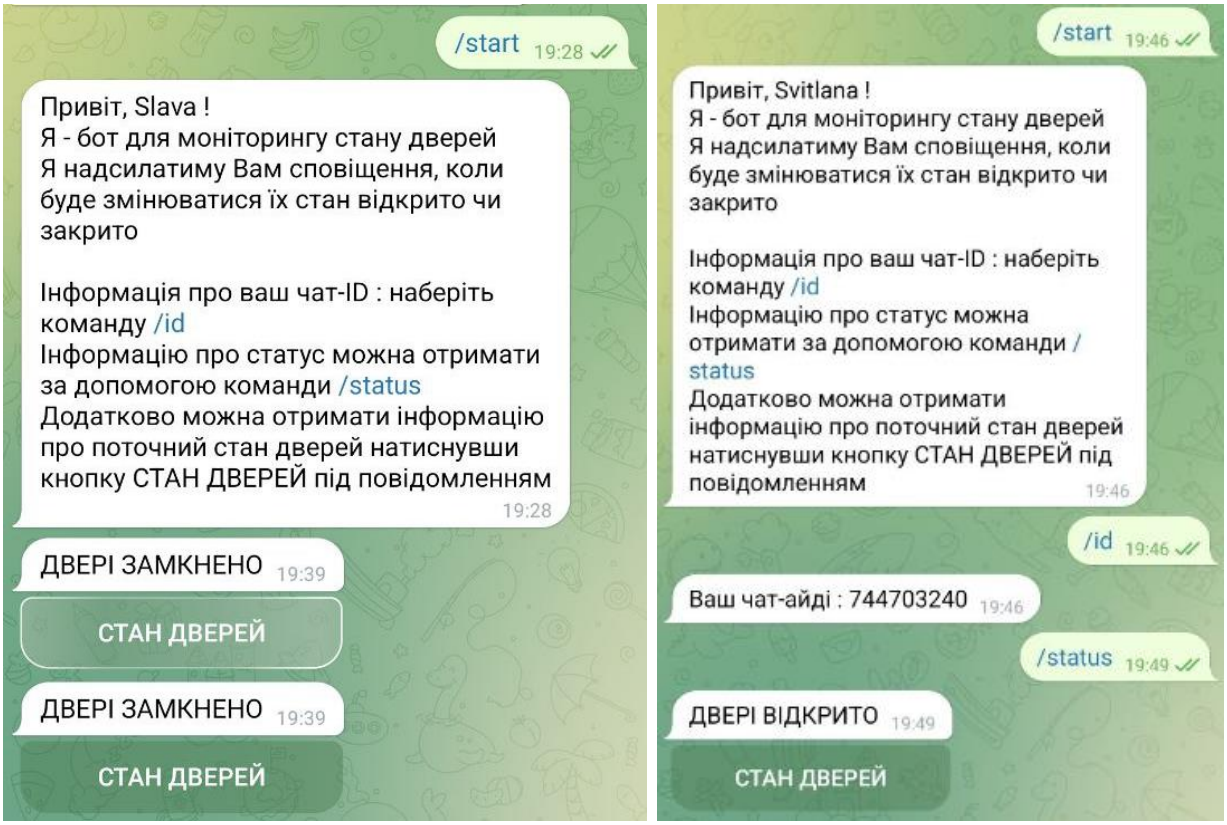

Рисунок 3.11 – Тестування роботи бота зі сповіщенням двох користувачів

3.4.3 Оновлення прошивки мікроконтролера

Код оновленої прошивки повністю (з коментарями) наведений в додатку В.

На початку своєї роботи, у секції налаштувань (setup()) підключаємо усі необхідні бібліотеки: для роботи з Wi-Fi, для спілкування з Telegram-ботом, для роботи із складними даними (JSON). Також маємо надати для бота унікальний «ключ» (BOT_TOKEN), щоб він міг спілкуватися з Telegram.

Дана версія програми «знає» своїх користувачів заздалегідь. Вона зберігає список авторизованих Chat ID Telegram, забезпечуючи лише дозволеним користувачам отримувати сповіщення та керувати ботом.

Геркон як і раніше підключений до аналогового входу A0.

Для зручності користувача у бот додано інтерактивну клавіатуру в Telegram з кнопкою «СТАН ДВЕРЕЙ», це реалізовано за допомогою структур JSON. Кнопка дозволяє миттєво дізнатися про стан дверей сейфу.

На етапі тестування прошивки також програма виводить проміжну важливу інформацію на консоль через послідовний порт.

Можливим недоліком може бути «небезпечний» режим перевірки сертифікатів (client.setInsecure()), що робить бот менш захищеним, але зручнішим для налагодження та тестування.

Аналогічно до першої версії бота, розширений варіант теж має нескінченний цикл (loop()), у якому відбувається опитування давача геркона. Кожні кілька мілісекунд він зчитує аналогове значення і перевіряє, чи змінився стан дверей. Якщо двері відчинилися або зачинилися, і ця зміна є стабільною (без «тремтіння»), бот негайно надсилає сповіщення усім авторизованим користувачам Telegram, інформуючи їх про новий стан.

Паралельно з цим, бот уважно «слухає», що говорять йому користувачі Telegram, на відміну від спрощеної версії, це створює додаткове навантаження на ESP8266 та викликає можливе обмеження на кількість користувачів.

За обробку повідомлень відповідає спеціальна функція handleNewMessages(). Функція handleNewMessages() – це своєрідний «мозок» бота для спілкування (рисунок А.3). Вона перевіряє, чи кожен, хто надсилає повідомлення, є авторизованим користувачем. Якщо ні, бот ввічливо відмовляє у доступі. Далі функція розпізнає тип вхідного повідомлення. Якщо це натискання кнопки на клавіатурі («СТАН ДВЕРЕЙ»), бот надсилає у відповідь поточний статус дверей. Якщо ж це текстова команда, то діє залежно від змісту:

– /start – бот надсилає привітання та розповідає, що він вміє;

– /stop – бот інформує про завершення дії;

– /status – бот негайно повідомляє про поточний стан дверей;

– /id або /0 – бот повідомляє користувачеві його власний Chat ID.

Важливим є те, що виклики bot.getUpdates() у циклі loop() можуть іноді затримувати роботу програми, якщо надходить багато нових повідомлень. Проте у нашому випадку це не є критичною проблемою.

Прівняння функціоналу базової прошивки та оновленої наведено в таблиці 3.1.

Таблиця 3.1 – Основні зміни та нові функції оновленої прошивки

Характеристика	Покращений чат-бот	Чат-бот (базовий)
Кількість користувачів	підтримка кількох авторизованих користувачів (масив authorized_chat_ids)	підтримка лише одного авторизованого користувача (AUTHORIZED_CHAT_ID)
Змінна для користувачів	const String authorized_chat_ids[] та authorized_chat_ids_count	const String AUTHORIZED_CHAT_ID
Функції надсилання сповіщень про стан дверей	sendStatusMessageToUser(String chat_id) та sendStatusMessageToAll()	функціональність інтегрована безпосередньо в loop() та handleNewMessages()
Обробка авторизації	функція isChatIdAuthorizedIgnoreCase() перевіряє вхідний ID на відповідність будь-якому елементу масиву	функція isChatIdAuthorized() перевіряє вхідний ID на відповідність єдиному AUTHORIZED_CHAT_ID
Telegram-клавіатура	використання inline-клавіатури (statusKeyboardJson) з кнопкою «СТАН ДВЕРЕЙ» та обробка callback_query	відсутність inline-клавіатур, повідомлення надсилаються без кнопок
Обробка Telegram-повідомлень	обробка як текстових повідомлень, так і callback_query	обробка тільки текстових повідомлень
Стартове повідомлення (/start)	можливість отримати статус дверей за допомогою кнопки	немає кнопки
Вивід в Serial-порт	активне використання Serial.print() та Serial.println() для налагоджувального виводу (стани підключення, отримані повідомлення, стани геркона)	повністю відсутні виклики Serial.print() та Serial.println(). Код в консолі нічого не виводить, що ускладнює відладку
Структура коду	має окремі функції для надсилання повідомлень про стан дверей	примітивна структура коду в частині надсилання повідомлень, функціональність яких інтегрована

Як бачимо з таблиці 3.1, базовий бот є спрощеним, підтримує лише одного користувача, не використовує кнопок і не забезпечує зручної діагностики, що обмежує його застосування в реальних IoT-сценаріях. Покращений чат-бот

значно функціональніший порівняно з базовим варіантом. Він підтримує кількох авторизованих користувачів, має розділення логіки на окремі функції, зручну Telegram-клавіатуру та розширене логування в Serial Monitor, що спрощує налагодження.

Архітектура та реалізація система моніторингу стану дверей дають хороші перспективи для її подальшого розвитку. В майбутньому її можна доповнити іншими сенсорами, наприклад, датчиком руху або температури, що дозволить створити більш комплексну систему безпеки. Можна також подумати в напрямку зберігання історії подій на сервері чи в хмарному сховищі для подальшого аналізу. Інтеграція з платформами розумного дому, такими як MQTT чи Home Assistant, відкриє можливості для автоматизованого керування та взаємодії з іншими IoT-пристроями (наприклад ввімкнення світла при відчиненні дверей, тощо). Отож, на даному етапі розробки маємо основу для масштабованої системи віддаленого моніторингу.

Цю роботу можна також розвинути для великої кількості користувачів, наприклад для моніторингу доступу до укриттів (якщо є зміна стану геркона, значить доступ до укриття є). Але тоді варто подумати про зберігання великих масивів Chat ID всіх бажаючих користувачів, крім того на монтаж системи в укритті потрібен дозвіл.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено систему DoorSafeBot для віддаленого моніторингу стану дверей на базі ESP8266 з використанням геркона та Telegram-бота. Система дозволяє оперативно отримувати сповіщення про відкриття чи закриття дверей, підтримує кількох користувачів та забезпечує зручне керування через Telegram. Реалізація проєкту підтвердила доцільність використання ESP8266 та Arduino IDE для побудови доступних і функціональних IoT-рішень.

Проаналізовано існуючі рішення для дистанційного моніторингу відкриття дверей – промислові датчики від таких фірм, як Ajax Systems, Hikvision, Xiaomi, Ring, U-Prox WDC, а також DIY-розробки на базі недорогих мікроконтролерів.

Підібрано деталі для реалізації апаратної складової проєкту: геркон та мікроконтролер ESP8266 на платі NodeMCU. Функціональність даного вибору та тестування ідеї виконано на макетній платі, потім систему інтегровано в двері іграшкового сейфу для точного моделювання ситуації відкриття та зачинення дверей.

Створено бот в Telegram за допомогою BotFather, отримано токен, який використано для з’єднання мікроконтролера з ботом. Описано логіку роботи програми (прошивки мікроконтролера). Відтестовано базовий функціонал, після чого виконано перепрошивку (з додаванням inline-клавіатури та багатокористувацького режиму роботи бота).

Весь процес створення системи віддаленого моніторингу відкриття дверей супроводжувався тестуванням та перевіркою функціональності, з цією метою було також додано в проєкт розширене логування в Serial Monitor.

Наведено приклад роботи системи на базі іграшкового сейфу, проте дану розробку можна використовувати в реальних IoT-сценаріях, де потрібно знати про стан (відчинено чи зачинено) дверей. На основі цієї роботи можна також розробити систему моніторингу стану дверей в укриттях, що є досить актуальним на даний час.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Babiuch M., Postulka J. Smart Home Monitoring System Using ESP32 Microcontrollers. Internet of Things. IntechOpen, Aug. 18, 2021. doi: 10.5772/intechopen.94589.

2. Enhancing Home Security through an Android-based Door Security System using IoT Technology. URL: https://www.researchsquare.com/article/rs-3163015/v1 (дата звернення: 12.04.2025).

3. Garage Door Open/Closed Monitor via ESP-NOW or Domestic LAN. *Arduino Forum*. URL: https://forum.arduino.cc/t/garage-door-open-closed-monitor-via-esp-now-or-domestic-lan/698424/13 (дата звернення: 21.03.2025).

4. Google Тренди. *Google Тренди.* URL: https://trends.google.com.ua/trends/explore?q=NodeMCU,WeMos%20D1%20Mini&hl=uk (дата звернення: 20.05.2025).

5. HC-SR501 Регульований інфрачервоний датчик руху. *Радіомагазин «Електроніка» – інтернет магазин радіодеталей та електроніки*. URL: https://electronica.in.ua/ua/p1530387497-sr501-reguliruemyj-infrakrasnyj.html (дата звернення: 01.05.2025).

6. Introduction to MPU6050. *The Engineering Projects*. URL: https://www.theengineeringprojects.com/2019/02/introduction-to-mpu6050.html (дата звернення: 01.05.2025).

7. Kanagamalliga S., et al. Biometric and IoT Integration for Secure and Remote Door Access Control Using Fingerprint Recognition and GSM Technology. In: *E3S Web of Conferences*. EDP Sciences, 2025. p. 03013.

8. KY-025 Reed Switch Module. *ArduinoModulesInfo*. URL: https://arduinomodules.info/ky-025-reed-switch-module/ (дата звернення: 24.04.2025).

9. Mi Window and Door Sensor. *MI.UA*. URL: https://mi.ua/ua/datchiki-dlja-signalizacij/mi-window-and-door-sensor.html (дата звернення: 10.02.2025).

10. NodeMCU v3 high resolution pinout and specs. *Renzo Mischianti*. URL: https://mischianti.org/nodemcu-v3-high-resolution-pinout-and-specs/ (дата звернення: 15.04.2025).

11. Pipl – Системи відеоспостереження, джерела живлення, сигналізація. *Pipl*. URL: https://pipl.ua/ (дата звернення: 11.02.2025).

12. Аксесуари для СКУД. *HIK.UA.* URL: https://hik.ua/control-access-systems/acs-accessories/ (дата звернення: 12.03.2025).

13. Датчик Холла A3144. *Електрочіп – магазин електроніки.* URL: https://surl.li/grbiwz (дата звернення: 06.05.2025).

14. Датчики відчинення вікон та дверей. *Ajax Systems*. URL: https://ajax.systems/ua/groups/open-detectors/ (дата звернення: 24.05.2025).

15. Євсюков Є. А., Дарнапук, Є. С. Використання системи «розумний дім» на базі ESP8266 як частини системи безпеки оселі. *ОЛЬВІЙСЬКИЙ ФОРУМ–2024: стратегії країн Причорноморського регіону в геополітичному просторі*, 2024, 212.

16. Інструкція з використання Superior DoorProtect Fibra. URL: https://support.ajax.systems/uk/manuals/superior-doorprotect-fibra/ (дата звернення: 28.03.2025).

17. Кійко Ю. Ю. Електронний сейф на базі контролера Arduino: випускна робота молодшого спеціаліста за спеціальністю «123 – Комп’ютерна інженерія». Запоріжжя, 2022. 74 с.

18. Кваліфікаційна робота Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп’ютерна інженерія» галузі знань 12 Інформаційні технології спеціальності 123 Комп’ютерна інженерія денної та заочної форм навчання/ уклад. С.В. Лавренчук, Н.В. Багнюк. Луцьк: ЛНТУ, 2025. 56 с.

19. Лилик І. В. Комп’ютерна система побутової охоронної сигналізації: кваліфікаційна робота бакалавра за спеціальністю «123 – Комп’ютерна інженерія». Тернопіль: ТНТУ, 2022. 75 с.

20. Модуль TCRT5000 – інфрачервоний датчик відображення. *Радіомагазин «Електроніка» – інтернет магазин радіодеталей та електроніки.* URL: https://surl.li/tnoqma (дата звернення: 01.05.2025).

21. Огляд охоронної сигналізації Ring Alarm Security Kit. *Smarty.ninja*. URL: https://www.smarty.ninja/hard/security/ohliad-okhoronnoi-syhnalizatsii-ring-alarm-security-kit/ (дата звернення: 10.02.2025).

22. Охоронна система: з чого вона складається і як функціонує. *bezpeka.club*. URL: https://bezpeka.club/ohoronna-systema-z-chogo-vona-skladayetsya-i-yak-funktsionuye (дата звернення: 08.02.2025).

23. Плати розробника ESP8266 (ESP32) NanoV3. *Мій Проект*. URL: https://surl.li/cwzhmc (дата звернення: 07.04.2025).

24. Складові частини охоронної сигналізації та їхні функції. *АЛЬЯНС-БЕЗПЕКА*. URL: https://alliancesafety.com.ua/uk/blog-ukr/skladovi-chastini-ohoronnoi-signalizatsii/ (дата звернення: 08.02.2025).

ДОДАТКИ

Додаток А

Узагальнений алгоритм роботи програми

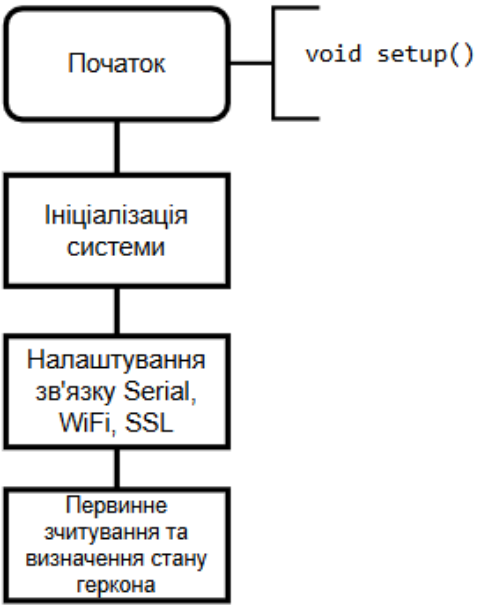

Рисунок А.1 – Узагальнений алгоритм функції setup()

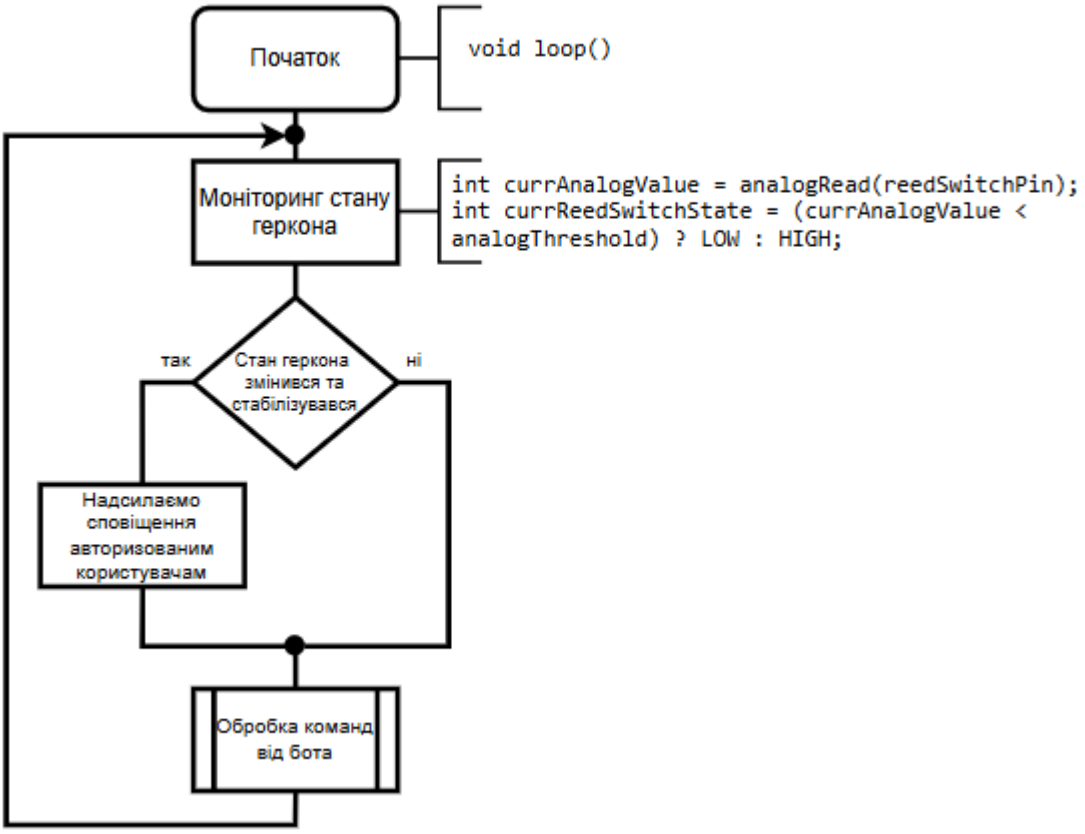

Рисунок А.2 – Узагальнений алгоритм функції loop()

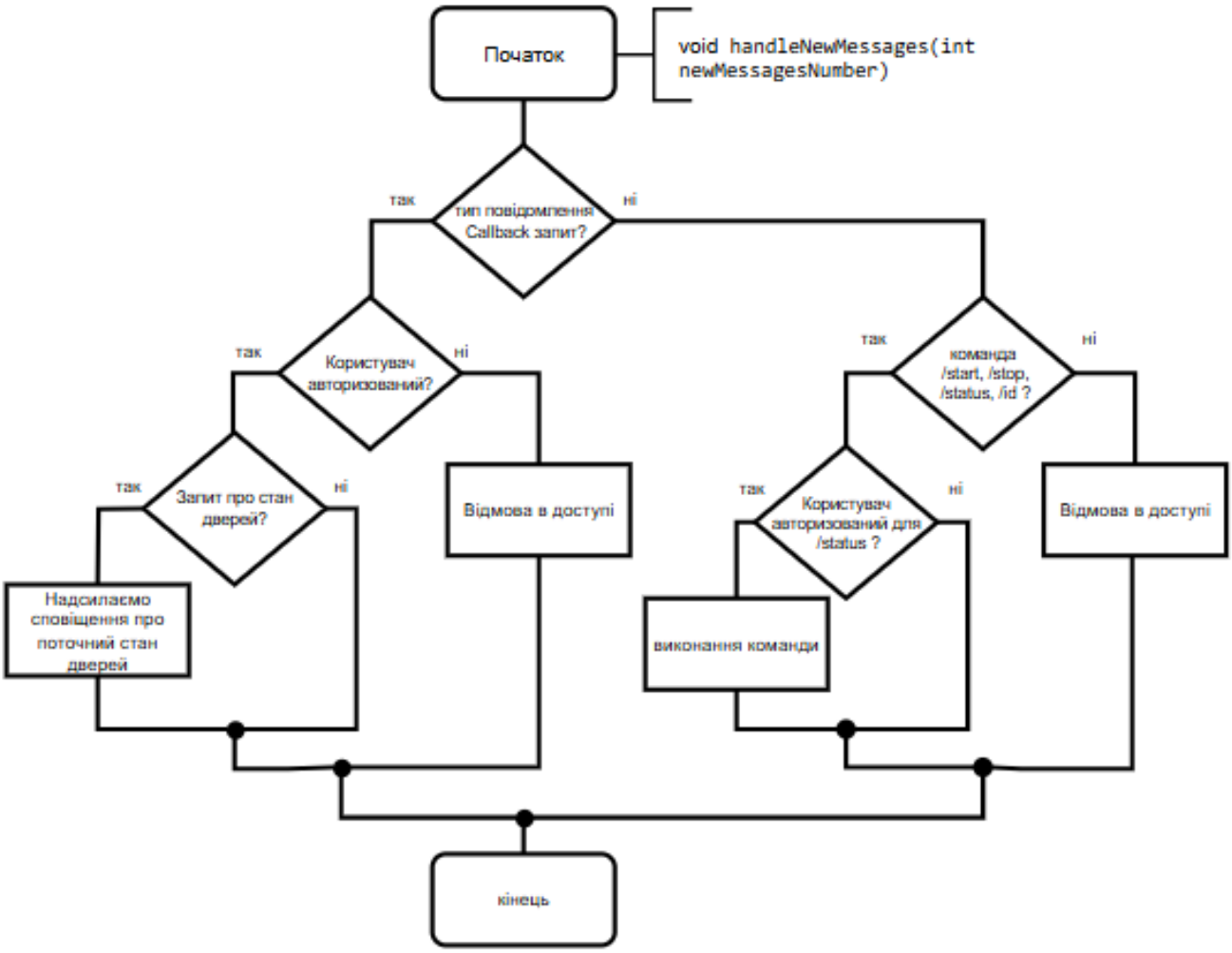

Рисунок А.3 – Узагальнений алгоритм обробки команд

Додаток Б

Базовий код бота відкритих дверей для одного користувача

```
/* БОТ: DoorSafeBot (спрощений для одного користувача, без Serial виводу, без кнопок)*/
// імпортуємо сюди коди бібліотек
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
// налаштовуємо Wi-Fi - мережу
const char* wifi_ssid = "def";  // назва Wi-Fi мережі до якої підключаємося
const char* wifi_password = "bot1"; // пароль від Wi-Fi мережі
// базові налаштування Telegram-бота
#define BOT_TOKEN "7552727215:AAHZ7OjGILwG0PAn_3eMT4ttLz1_mqPzcG8" /* Slava-бот – DoorSafeBot */
// ЄДИНИЙ АВТОРИЗОВАНИЙ CHAT ID
// прописуємо CHAT ID користувача (поки свій)
const String AUTHORIZED_CHAT_ID = "1209313669";

WiFiClientSecure client;
UniversalTelegramBot bot(BOT_TOKEN, client);

// налаштування для геркона
const int reedSwitchPin = A0; // АЦП-вивід A0
int reedSwitchState = -1;     /* змінна для зберігання поточного стану геркона (-1 для першого читання)*/
int lastReedSwitchState = -1; /* змінна для зберігання попереднього стану геркона*/

/* поріг для визначення стану геркона
якщо геркон замкнений (магніт є) -> GND -> AnalogRead буде близьким до нуля
якщо геркон розімкнений (магніту немає) -> VCC (через PULLUP) -> AnalogRead буде близьким до 1023*/
const int analogThreshold = 500; // типовий початковий поріг

unsigned long lastSendTime = 0;   // час останнього надсилання повідомлення
const long debounceDelay = 50;    /* затримка для уникнення "стрибків" контактів геркона*/

/* функція - перевірка на збіг ідентифікатор користувача (для одного користувача)*/
bool isChatIdAuthorized(String chat_id_to_check) {
    return AUTHORIZED_CHAT_ID.equalsIgnoreCase(chat_id_to_check);
}

void handleNewMessages(int newMessagesNumber) {
    for (int i = 0; i < newMessagesNumber; i++) {
        String chat_id = bot.messages[i].chat_id;
```

```
        // Перевіряємо, чи повідомлення від авторизованого користувача
        if (!isChatIdAuthorized(chat_id)) {
            bot.sendMessage(chat_id, "Ви не авторизовані для взаємодії з
цим ботом.", "");
            continue; // Пропускаємо обробку для неавторизованих
        }

        // обробка звичайних текстових повідомлень
        String text = bot.messages[i].text;
        text.toLowerCase();
        String from_name = bot.messages[i].from_name;
        if (from_name == "") from_name = "Guest";

        // обробка команд
        if (text == "/start") {
            String welcomeMessage = "Привіт, " + from_name + " !\n";
            bot.sendMessage(chat_id, welcomeMessage, "");
        } else if (text == "/stop") {
            bot.sendMessage(chat_id, "Завершіть дію за допомогою
інструментів телеграм", "");
        } else if (text == "/status") {
            if (lastReedSwitchState == LOW) {
                bot.sendMessage(chat_id, "ДВЕРІ ЗАМКНЕНО", "");
            } else {
                bot.sendMessage(chat_id, "ДВЕРІ ВІДКРИТО", "");
            }
        } else if (text == "/id" || text == "/0") { // Додано /0 як
альтернатива для /id, якщо треба
            bot.sendMessage(chat_id, "Ваш чат-айді : " + chat_id, "");
        }
    }
}

void setup() {
    delay(10);

    // підключення до Wi-Fi
    WiFi.begin(wifi_ssid, wifi_password);
    client.setInsecure(); // відключаємо перевірку сертифікатів

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
    }

    // перше читання стану геркона
    int analogValue = analogRead(reedSwitchPin);
    reedSwitchState = (analogValue < analogThreshold) ? LOW : HIGH;
    lastReedSwitchState = reedSwitchState;
}

void loop() {
    // читаємо поточне аналогове значення з геркона
```

```
    int currAnalogValue = analogRead(reedSwitchPin);
    int currReedSwitchState = (currAnalogValue < analogThreshold) ? LOW :
HIGH;

    // перевіряємо, чи змінився стан геркона і чи пройшла затримка
    if (currReedSwitchState != lastReedSwitchState && millis() -
lastSendTime > debounceDelay) {
        delay(debounceDelay); // додаткова затримка для стабілізації
        currAnalogValue = analogRead(reedSwitchPin);
        currReedSwitchState = (currAnalogValue < analogThreshold) ? LOW :
HIGH;

        if (currReedSwitchState != lastReedSwitchState) {
            lastReedSwitchState = currReedSwitchState;
            lastSendTime = millis(); // оновлюємо час останнього надсилання

            String messageText;
            if (lastReedSwitchState == LOW) {
                messageText = "ДВЕРІ ЗАМКНЕНО";
            } else {
                messageText = "ДВЕРІ ВІДКРИТО";
            }
            bot.sendMessage(AUTHORIZED_CHAT_ID, messageText, "");
        }
    }

    // обробка вхідних повідомлень
    int newMessagesNumber = bot.getUpdates(bot.last_message_received + 1);
    while (newMessagesNumber) {
        handleNewMessages(newMessagesNumber);
        newMessagesNumber = bot.getUpdates(bot.last_message_received + 1);
    }
    delay(9); // Невелика затримка для стабільної роботи
}
```

Додаток В

Розширений код бота для багатьох користувачів та кнопковим інтерфейсом

```
// БОТ: DoorSafeBot /
// імпортуємо коди бібліотек
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
// налаштовуємо Wi-Fi - мережу
const char* wifi_ssid = "def"; // назва Wi-Fi мережі до якої підключаємося
const char* wifi_password = "bot1"; // пароль від Wi-Fi мережі
// базові налаштування Telegram-бота
#define BOT_TOKEN "7552727215:AAHZ7OjGILwG0PAn_3eMT4ttLz1_mqPzcG8"
/* Slava-бот – DoorSafeBot */
// CHAT_ID більше не використовуємо як єдиний ідентифікатор
WiFiClientSecure client;
UniversalTelegramBot bot(BOT_TOKEN, client);
// масив для зберігання усіх Chat IDs для підписаних користувачів
const String authorized_chat_ids[] = {
  // "**********" , // ID користувача /
  "1209313669", // ID першого користувача  / Slava
  "744703240"   // ID другого користувача /
};
// обчислимо розмір масиву користувачів ***
const int authorized_chat_ids_count = sizeof(authorized_chat_ids) /
sizeof(authorized_chat_ids[0]);
// налаштування для геркона
const int reedSwitchPin = A0; // АЦП-вивід A0
int reedSwitchState = -1;   /* змінна для зберігання поточного стану
геркона (-1 для першого читання)*/
int lastReedSwitchState = -1; /* змінна для зберігання попереднього стану
геркона*/
/* поріг для визначення стану геркона (залежить від геркона та підключення)
при підтягуванні до +3,3 вольти або використанні внутрішнього PULLUP:
якщо геркон замкнений (магніт є) -> GND -> AnalogRead буде близьким до нуля
якщо геркон розімкнений (магніту немає) -> VCC (через PULLUP) -> AnalogRead
буде близьким до 1023*/
/* потрібно вибрати поріг десь посередині діапазону. Значення 500 мілівольт
є типовим початковим порогом.*/
const int analogThreshold = 500;
unsigned long lastSendTime = 0; // час останнього надсилання повідомлення
const long debounceDelay = 50; /* затримка для уникнення "стрибків"
контактів геркона*/
// JSON для inline-клавіатури з кнопкою "СТАН ДВЕРЕЙ"
const char* statusKeyboardJson = "[[{\"text\":\"СТАН ДВЕРЕЙ\",
\"callback_data\":\"get_status\"}]]";
// функція - перевірка на збіг ідентифікатор користувача
bool isChatIdAuthorizedIgnoreCase(String chat_id_to_check) {
  for (int i = 0; i < authorized_chat_ids_count; i++) {
```

```
    if (authorized_chat_ids[i].equalsIgnoreCase(chat_id_to_check)) {
      return true; // знайдено збіг (незалежно від регістру)
    }
  }
  return false; // не знайдено збігів
}
/* функція для надсилання повідомлення про поточний стан геркона/дверей
конкретному користувачеві*/
void sendStatusMessageToUser(String chat_id) {
  if (lastReedSwitchState == LOW) {
    Serial.print("DOR CLOSE 0 ");
    bot.sendMessageWithInlineKeyboard(chat_id, "ДВЕРІ ЗАМКНЕНО", "",
statusKeyboardJson);
  } else {
    Serial.print("DOR OPEN 0 ");
    bot.sendMessageWithInlineKeyboard(chat_id, "ДВЕРІ ВІДКРИТО", "",
statusKeyboardJson);
  }
}
/* функція для надсилання повідомлення про поточний стан геркона ВСІМ
підписаним користувачам*/
void sendStatusMessageToAll() {
    String messageText;
    if (lastReedSwitchState == LOW) {
        messageText = "ДВЕРІ ЗАМКНЕНО";
        Serial.print("DOR CLOSE 1 ");
    } else {
        messageText = "ДВЕРІ ВІДКРИТО";
        Serial.print("DOR OPEN 1 ");
    }

    Serial.print("Sending status to ");
    Serial.print(authorized_chat_ids_count);
    Serial.println(" users:");

    for (int i = 0; i < authorized_chat_ids_count; i++) {
      // надсилаємо повідомлення з inline-клавіатурою кожному користувачу
      bot.sendMessageWithInlineKeyboard(authorized_chat_ids[i],
messageText, "", statusKeyboardJson);
      Serial.println(authorized_chat_ids[i]);
      delay(50); /* невелика затримка між надсиланням повідомлень різним
користувачам */
    }
}

void handleNewMessages(int newMessagesNumber) {
  Serial.print("handleNewMessages: ");
  Serial.println(newMessagesNumber);
  for (int i=0; i<newMessagesNumber; i++) {
    String chat_id = bot.messages[i].chat_id;
    Serial.print("chat_id = "); Serial.println(bot.messages[i].chat_id);
    // перевіряємо тип повідомлення
```

```
    if (bot.messages[i].type == "callback_query") {
      String callback_data = bot.messages[i].text; /* У цій бібліотеці
callback_data знаходиться в полі text*/
      String from_id = bot.messages[i].from_id; /* ID користувача, який
натиснув кнопку */
      //
      Serial.print("Received callback query from ");
      Serial.print(from_id);
      Serial.print(": ");
      Serial.println(callback_data);
      // перевіряємо, чи користувач, який натиснув кнопку, авторизований
      if (isChatIdAuthorizedIgnoreCase(from_id)) {
         if (callback_data == "get_status") {
            /* якщо дані зворотного виклику "get_status", надсилаємо стан
конкретному користувачу*/
            sendStatusMessageToUser(from_id);
         }
         /* можна буде додати обробку інших callback_data тут, якщо
потрібно*/
      } else {
        bot.sendMessage(from_id, "Ви не авторизовані для виконання цієї
дії", "");
      }
    } else {
      // обробка звичайних текстових повідомлень
      String text = bot.messages[i].text;
      text.toLowerCase(); // або text.toUpperCase();
      String from_name = bot.messages[i].from_name;
      if (from_name == "") from_name = "Guest";

      Serial.print("Received text message from ");
      Serial.print(chat_id);
      Serial.print(" (");
      Serial.print(from_name);
      Serial.print("): ");
      Serial.println(text);

      // обробка команд підписки та відписки
      if (text == "/start") {
         // надсилаємо вітальне повідомлення
         String welcomeMessage = "Привіт, " + from_name + " !\n";
         welcomeMessage += "Я - бот DoorSafeBot для моніторингу стану
дверей\n";
         welcomeMessage += "Я надсилатиму Вам сповіщення, коли буде
змінюватися їх стан відкрито чи закрито\n\n";
         welcomeMessage += "Інформація про ваш чат-ID : наберіть команду
/id \n";
         welcomeMessage += "Інформацію про статус можна отримати за
допомогою команди /status \n";
         welcomeMessage += "Додатково можна отримати інформацію про
поточний стан дверей натиснувши кнопку СТАН ДВЕРЕЙ під повідомленням";
         bot.sendMessage(chat_id, welcomeMessage, "");
```

```
      } else if (text == "/stop") {
         bot.sendMessage(chat_id, "Завершіть дію за допомогою інструментів
телеграм", "");
      }
      /* обробка команди /status (для сумісності або якщо користувач не
використовує кнопку)*/
      else if (text == "/status") {
         if (isChatIdAuthorizedIgnoreCase(chat_id)) {
             sendStatusMessageToUser(chat_id);
         } else {
            bot.sendMessage(chat_id, "Ви не авторизовані для виконання цієї
дії. Надішліть /start для підписки.", "");
         }
      }
      // додайте обробку інших текстових команд тут, якщо потрібно
      else if (text == "/id" || text == "/0") {
         bot.sendMessage(chat_id, "Ваш чат-айді : " + chat_id, "");
      }
    }
  }
}

void setup() {
  Serial.begin(115200);
  delay(10);
  // підключення до Wi-Fi
  Serial.println();
  Serial.print("Connecting to WiFi: ");
  Serial.println(wifi_ssid);
  WiFi.begin(wifi_ssid, wifi_password);
  // налаштування SSL для Telegram
  /* для старіших версій, або якщо потрібна вища безпека, може знадобитися
root certificate */
  client.setInsecure(); // відключаємо перевірку сертифікатів

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  //
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  /* ініціалізація бота відбувається при створенні об'єкта
UniversalTelegramBot*/
  /* налаштування виводу геркона як входу з внутрішнім підтягуючим
резистором - у нашому випадку 10к резистор зовні*/
  // pinMode(reedSwitchPin, INPUT_PULLUP); // Активуємо внутрішній PULLUP
  // перше читання стану геркона -- усуваємо дрижання контактів
  int analogValue = analogRead(reedSwitchPin);
```

```
  /* визначаємо початковий стан: LOW = замкнено (магніт є), HIGH =
розімкнено (магніту немає)*/
  reedSwitchState = (analogValue < analogThreshold) ? LOW : HIGH;
  lastReedSwitchState = reedSwitchState;
  Serial.println("Bot started. Send /start in Telegram to subscribe.");
}

void loop() {
  // читаємо поточне аналогове значення з геркона
  int currAnalogValue = analogRead(reedSwitchPin);
  // визначаємо поточний цифровий стан на основі порогу
  int currReedSwitchState = (currAnalogValue < analogThreshold) ? LOW :
HIGH;
  /* перевіряємо, чи змінився стан геркона і чи пройшла затримка для
уникнення "стрибків"*/
  if (currReedSwitchState != lastReedSwitchState && millis() - lastSendTime
> debounceDelay) {
    delay(debounceDelay); // додаткова затримка для стабілізації
    // повторно читаємо стан після затримки, щоб переконатися у зміні
    currAnalogValue = analogRead(reedSwitchPin);
    currReedSwitchState = (currAnalogValue < analogThreshold) ? LOW : HIGH;
    //
    if (currReedSwitchState != lastReedSwitchState) {
      lastReedSwitchState = currReedSwitchState;
      lastSendTime = millis(); // оновлюємо час останнього надсилання
      /* надсилаємо повідомлення про зміну стану ВСІМ підписаним
користувачам*/
      sendStatusMessageToAll();
       if (currReedSwitchState == LOW) {
         Serial.println("Геркон: ЗАМКНЕНО");
      /*bot.sendMessageWithInlineKeyboard(CHAT_ID, "ДВЕРІ ЗАМКНЕНО", "",
statusKeyboardJson);*/
       } else {
         Serial.println("Геркон: РОЗІМКНЕНО");
      /*   bot.sendMessageWithInlineKeyboard(CHAT_ID, "ДВЕРІ ВІДКРИТО", "",
statusKeyboardJson);*/
       }
    }
  }
  // обробка вхідних повідомлень та callback queries
  int newMessagesNumber = bot.getUpdates(bot.last_message_received + 1);
  while(newMessagesNumber) {
    handleNewMessages(newMessagesNumber);
    newMessagesNumber = bot.getUpdates(bot.last_message_received + 1);
  }
  delay(9); // Невелика затримка для стабільної роботи
}
```