НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ КПІ імені ІГОРЯ СІКОРСЬКОГО»
(КПІ ім. Ігоря Сікорського)
Кафедра автоматизації та систем неруйнівного контролю

КУРСОВА РОБОТА

з мікроконтролерів та мікропроцесорної техніки

на тему: Програмований пристрій сигналізації

Виконав:
студент ІІІ курсу, групи ПМ-11, денна форма навчання
Погорєлов Богдан Юрійович

Керівник:
ас. Самборська В.В

Залікова оцінка_____

Кількість балів:_____

Київ – 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ КПІ імені ІГОРЯ СІКОРСЬКОГО»

(КПІ ім. Ігоря Сікорського)

Кафедра автоматизації та систем неруйнівного контролю

Факультет Приладобудівний

Кафедра Автоматизації та систем неруйнівного контролю

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології

Освітня програма Комп'ютерно-інтегровані системи та технології в приладобудуванні

ЗАВДАННЯ

на курсову роботу

з мікроконтролерів та мікропроцесорної техніки

1. Тема роботи: Програмований пристрій сигналізації

2. Строк подання студентом роботи 2024.06.31.

3. Вихідні дані до курсової роботи: лекційний матеріал з дисципліни «Мікроконтролери та мікропроцесорна техніка», інтернет ресурси, документації, статті, конференції, наукові публікації.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) огляд існуючих систем програмованих пристроїв сигналізації, розробка схеми, підбір елементів, розробка блок-алгоритму, розробка комплексу програмного забезпечення, тестування та аналіз результатів

5. Перелік (ілюстративного) графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів, тощо) — кресленики:

1. КР ПМ-11.00.00 СХ (структурна схема);
2. КР ПМ-11.00.01 БС (блок-схема алгоритму роботи програми);
3. КР ПМ-11.00.02 БС (блок-схема алгоритму роботи програми 2 частина);
4. КР ПМ-11.00.00 ЕС (електрична система пристрою);
5. КР ПМ-11.00.00 ДП (друкована плата пристрою).

6. Дата видачі завдання 2024.01.18.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Строк виконання етапів роботи	Примітка
1	Аналітичний огляд. Визначення напрямку розробки	2024.01.18	
2	Проведення аналітичного огляду матеріалів	2024.02.10	
3	Підбір технологій та модулів	2024.02.20	
4	Розробка структурної схеми	2024.03.10	
5	Розробка блок схеми алгоритму, та комплексу програмного забезпечення	2024.03.20	
6	Оформлення текстової частини КР	2024.04.10	
7	Відправити КР на перевірку науковому керівнику	2024.05.30	
8	Захист КР	2024.10.03	

Студент _____ Б. Ю. Погорєлов

(підпис) (ініціали, прізвище)

Керівник роботи _____ Самборська В.В

(підпис) (ініціали, прізвище)

ЗМІСТ

стор.

ВСТУП

У сучасному світі безпека є одним з пріоритетних напрямків розвитку науково-технічного прогресу. Постійно зростаюча потреба у захисті майна, інформації та особистої безпеки стимулює розробку новітніх технологій у галузі охоронних систем.

Одним з найбільш поширених напрямків стала розробка систем охорони, що використовують різноманітні сенсори та пристрої для виявлення і попередження несанкціонованого доступу. Важливою складовою таких систем є датчики, що здатні реагувати на різні типи впливу, як-от відчинення дверей, рух у приміщенні та інші події. Завдяки цьому забезпечується комплексний підхід до захисту об'єктів різного призначення.

Цифрові технології займають панівне місце у розробці систем охорони, що зумовлено високою точністю, швидкодією та здатністю інтеграції з іншими інформаційними системами. Впровадження кодових панелей доступу дозволяє підвищити рівень безпеки, забезпечуючи надійний контроль доступу до приміщень та об'єктів.

Розвиток сенсорних технологій та мікроелектроніки значно покращив можливості сучасних охоронних систем. Використання датчиків відкриття дверей та датчиків руху дозволяє вчасно виявляти спроби проникнення і оперативно реагувати на них. Кодові панелі доступу, у свою чергу, забезпечують авторизований вхід до приміщень, що дозволяє знизити ризик несанкціонованого доступу.

Метою даної курсової роботи є розробка комп'ютеризованої системи охорони, що включає датчик відкриття дверей, датчик руху та кодову панель доступу. Система призначена для забезпечення високого рівня безпеки шляхом виявлення і обробки (збір, аналіз та візуалізація) даних про стан охоронюваного об'єкта. Завдяки такій системі можна проводити моніторинг подій у режимі реального часу, контролювати доступ до приміщень та оперативно реагувати на можливі загрози. Запропонована система може знайти застосування у житлових будинках, офісних приміщеннях, промислових підприємствах та інших об'єктах, де необхідний високий рівень безпеки.

1. ТЕХНІЧНЕ ЗАВДАННЯ

1.1. Підстава для розробки, призначення та галузь застосування

Підставою для розробки є завдання на курсове проектування. Система призначена для охорони приміщень і виявлення несанкціонованого доступу. Система повинна знаходитись на локальному сервері з графічним інтерфейсом користувача (GUI) та backend, а також мати панель на мікроконтролері, який відсилатиме стани датчиків руху та відкриття дверей.

1.2. Умови експлуатації

1.2.1. Температура зовнішнього середовища: від 0 до +45 °C

1.2.2. Відносна вологість: не більше 80%

1.2.3. Атмосферний тиск: 600-800 мм. рт. ст.

1.2.4. Хімічно активних речовин: немає

1.3. Вимоги до технічних характеристик

1.3.1. Об’єкт управління: комп’ютеризована система охорони

1.3.2. Параметр вимірювання: стан датчиків руху та відкриття дверей

1.3.3. Діапазон виявлення руху: до 10 метрів

1.3.4. Похибка виявлення руху: 0,5 метрів

1.3.5. Відображення отриманих значень на індикаторі та графічному інтерфейсі користувача

1.3.6. Джерело живлення мікроконтролера: + 5 В постійного струму

1.4. Вимоги до складових системи

1.4.1. Локальний сервер з встановленим програмним забезпеченням для обробки даних та GUI

1.4.2. Мікроконтролер з модулем бездротового зв'язку.

1.4.3. Датчики руху (PIR) та магнітні контакти для дверей

1.4.4. Програмне забезпечення для мікроконтролера для зчитування та передачі даних з датчиків

1.4.5. Програмне забезпечення для локального сервера для обробки та відображення даних на GUI

1.4.6. Мережеве обладнання для забезпечення стабільного з'єднання між мікроконтролером і сервером

1.5. Програмно-алгоритмічне забезпечення

1.5.1. Розробка коду для мікроконтролера для зчитування даних з датчиків та їх передача на сервер

1.5.2. Розробка backend-частини для прийому та обробки даних з мікроконтролера

1.5.3. Розробка GUI для відображення стану датчиків у реальному часі

1.6. Етапи розробки

1.6.1. Початок розробки – лютий 2023 р.

1.6.2. Кінець розробки – червень 2024 р.

1.7. Вимоги до безпеки

1.7.1. Забезпечити захист даних під час передачі між мікроконтролером і сервером

1.7.2. Забезпечити захист локального сервера від несанкціонованого доступу

2. ОГЛЯД ТА АНАЛІЗ МЕТОДІВ

2.1. Застосування інтелектуальних систем охорони в інформаційно-вимірювальних системах

Інформаційні технології глибоко проникають у всі сфери науково-технічної діяльності суспільства. Центральною ланкою таких систем є технології інженерних знань, тому значні успіхи багато в чому визначаються інтелектуальним рівнем і загальною ефективністю комп'ютерних систем.

Системи охорони, які базуються на мікроконтролерах, являють собою найважливіші частини сучасних мікропроцесорних систем управління безпекою об'єктів. Від однофункціональних засобів визначення поточних станів датчиків вони поступово перетворюються в багатофункціональні засоби автоматизації, які вирішують цілий ряд завдань з діагностики, обробки та виконання простих алгоритмів управління на основі вимірювальної інформації.

Неодмінною умовою таких систем є можливість самонавчання і самовідновлення при виникненні певних збоїв, що дозволяє функціонувати як в автономному, так і в інтерактивному режимі, забезпечуючи стабільну роботу протягом тривалого періоду навіть при зміні умов експлуатації.

Впровадження подібних інформаційно-вимірювальних пристроїв дозволяє не лише здійснювати моніторинг стану об'єктів, але й запам'ятовувати ряд подій, проводити їх обробку та аналіз. Ще однією особливістю є можливість обміну даними між зовнішніми (периферійними) пристроями або показ даних на цифрових та графічних індикаторах. При необхідності розширення функціональних можливостей, оперативного отримання інформації та візуалізації результатів вимірювань, це тягне за собою значне ускладнення алгоритму роботи, збільшення параметрів процесора, обсягу пам'яті тощо.

Рис. 1.1 Система охорони компанії Ajax [1]

2.2. Структура та компоненти систем охорони

Сучасні системи охорони, які базуються на мікроконтролерах і локальних серверах, включають наступні основні компоненти:

- Локальний сервер: Він відповідає за обробку даних, зібраних з мікроконтролера, і забезпечує графічний інтерфейс користувача (GUI) для моніторингу та управління системою. Сервер також зберігає історію подій і може генерувати звіти.
- Мікроконтролер: Мікроконтролери, такі як ESP32, використовуються для зчитування станів датчиків руху та відкриття дверей, а також для передачі цих даних на локальний сервер. Мікроконтролери можуть працювати в мережі Wi-Fi або використовувати інші бездротові технології.
- Датчики руху та відкриття дверей: Ці датчики є основними елементами системи охорони. Вони забезпечують виявлення руху та несанкціонованого відкриття дверей, передаючи сигнали на мікроконтролер.
- Програмне забезпечення: Програмне забезпечення включає код для мікроконтролера, який зчитує дані з датчиків і передає їх на сервер, а також backend для обробки даних на сервері і GUI для користувача.

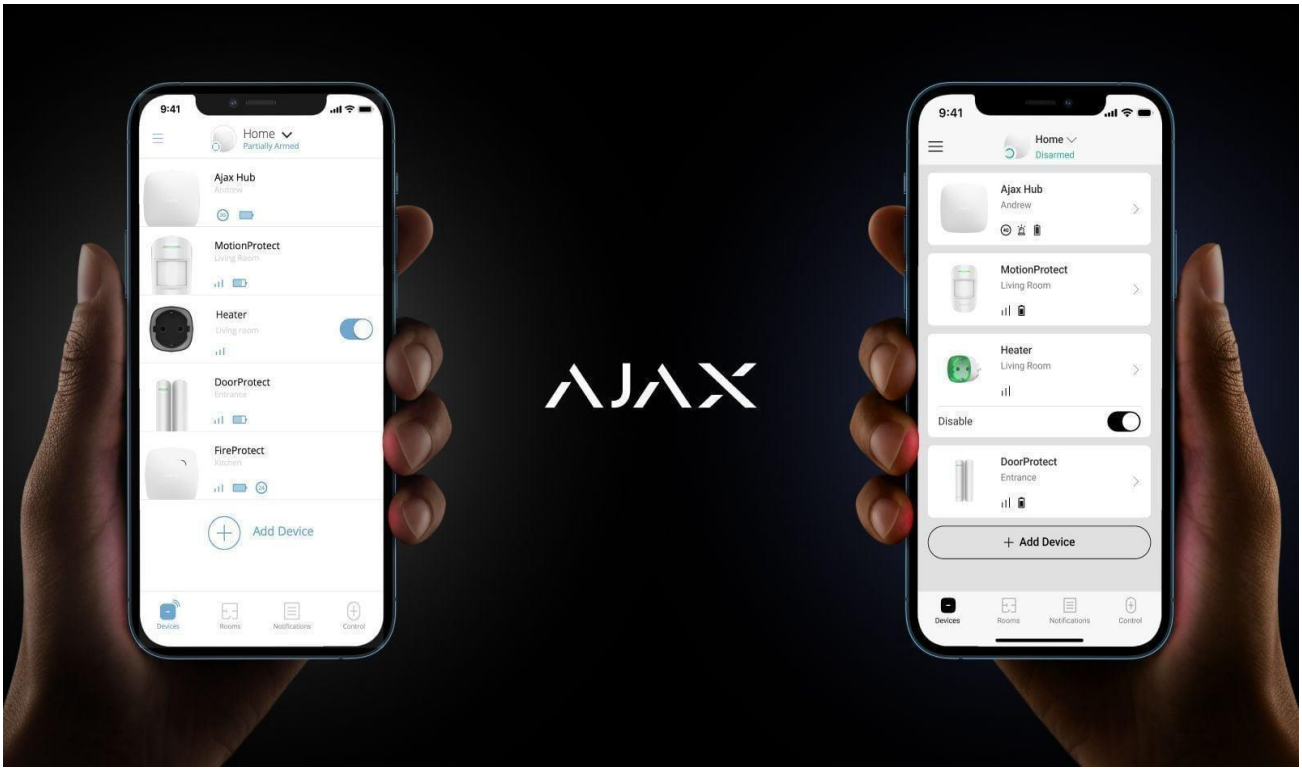


Рис. 1.2 Додаток системи охорони Ajax [1]

2.3. Принцип роботи системи

Принцип роботи системи охорони полягає в наступному:

- Датчики руху та відкриття дверей зчитують відповідні параметри та передають їх на мікроконтролер.
- Мікроконтролер обробляє отримані дані та передає їх на локальний сервер через бездротову мережу.
- Локальний сервер приймає дані, обробляє їх і відображає на GUI для користувача.

2.4. Переваги та виклики

Переваги:

- Високий рівень автоматизації та точності виявлення.
- Можливість дистанційного моніторингу та управління.
- Інтеграція з іншими системами безпеки та автоматизації.
- Збереження історії подій для аналізу та звітності.

Виклики:

- Забезпечення стабільного бездротового зв’язку між мікроконтролером та сервером.
- Захист від кібератак та несанкціонованого доступу до системи.
- Ускладнення алгоритмів роботи та збільшення вимог до обчислювальних ресурсів серверу.

2.5. Перспективи розвитку

Розвиток систем охорони, що базуються на мікроконтролерах та локальних серверах, спрямований на підвищення інтелектуальності систем, покращення інтеграції з іншими інформаційно-вимірювальними системами, а також на підвищення рівня безпеки і надійності системи в цілому. У майбутньому можливе впровадження більш складних алгоритмів обробки даних, використання машинного навчання для підвищення точності виявлення подій, а також розширення функціональних можливостей системи для задоволення потреб користувачів.

3. РОЗРОБКА СТРУКТУРНОЇ СХЕМИ

3.1. Аналіз існуючих систем охорони

Поширеним у застосуванні систем охорони є використання мікроконтролерів для збору даних з датчиків руху та відкриття дверей і передача цих даних на локальний сервер для подальшої обробки та відображення

Сучасні системи охорони, такі як українська Ajax [1], використовують мікроконтролери для збору даних з різноманітних датчиків, включаючи датчики руху та відкриття дверей. Ці дані передаються на локальний сервер для подальшої обробки та відображення. Структурна схема такої системи охорони може включати кілька ключових компонентів:

- Датчики: Різні типи датчиків, такі як датчики руху, відкриття дверей, розбиття скла тощо, розташовані по всій охоронюваній території. Вони фіксують будь-які підозрілі дії або порушення.
- Мікроконтролери: Вбудовані мікроконтролери в датчиках зчитують дані та відправляють їх на центральний блок управління. Ці мікроконтролери можуть бути оснащені бездротовими модулями для передачі даних (наприклад, Wi-Fi, Zigbee, Z-Wave).
- Центральний блок управління (хаб): Основний контролер системи, який отримує дані від датчиків, обробляє їх та, у разі виявлення загрози, активує сигнал тривоги або інші попереджувальні механізми. Хаб також може передавати дані на сервер.
- Локальний сервер: Зберігає та обробляє отримані дані, забезпечуючи інтерфейс для моніторингу і управління системою. Сервер може бути підключений до інтернету для віддаленого доступу та керування через мобільний додаток або веб-інтерфейс.
- Користувацький інтерфейс: Зазвичай представлений мобільним додатком або веб-інтерфейсом, що дозволяє користувачам отримувати повідомлення про події, переглядати історію подій, керувати налаштуваннями системи та здійснювати інші дії.

Система Ajax і подібні їй інтегрують усі ці компоненти для забезпечення надійної охорони об’єктів, використовуючи передові технології для швидкої та ефективної обробки даних і забезпечення безпеки.

Існують також і інші міжнародні системи охорони, розглянимо їх.

1. Honeywell

Honeywell [2] пропонує широкий спектр систем охорони, включаючи:

- Датчики руху: З технологією Dual Tec для зменшення помилкових спрацьовувань.
- Контроль доступу: Системи контролю доступу для обмеження входу в певні зони.
- Сигналізаційні системи: Централізовані пульти управління, що інтегруються з різними датчиками.
- Відеоспостереження: Камери високої роздільної здатності з можливістю запису та віддаленого доступу.

2. ADT

ADT [3] є одним з найбільших постачальників систем охорони в США:

- Моніторинг 24/7: Постійний моніторинг з швидким реагуванням на тривоги.
- Інтелектуальні датчики: Різні типи датчиків, включаючи датчики диму, CO, відкриття дверей і вікон.
- Смарт-управління: Інтеграція з домашніми системами автоматизації, такими як розумні термостати та освітлення.
- Відеоаналітика: Системи відеоспостереження з аналітикою для виявлення підозрілої активності.

3. Ring

Ring [4], відомий своїми розумними відеодомофонами, також пропонує комплексні системи безпеки:

- Відеодомофони: Високоякісні відеодомофони з функцією двостороннього аудіозв'язку.
- Смарт-сигналізація: Системи з різними датчиками та можливістю підключення до інтернету.
- Камери спостереження: Вуличні та внутрішні камери з функціями нічного бачення та рухомими оповіщеннями.

4. SimpliSafe

SimpliSafe [5] пропонує прості у встановленні системи охорони для будинків та квартир:

- Бездротові датчики: Легко встановлюються і підключаються до центрального блоку.
- Професійний моніторинг: Опційний професійний моніторинг з можливістю швидкого реагування.

- Інтуїтивний додаток: Зручний додаток для керування системою та отримання сповіщень.
- Гнучкі налаштування: Можливість додавання та налаштування різних датчиків за потреби.

5. Vivint

Vivint [6] пропонує інтегровані системи безпеки з акцентом на домашню автоматизацію:

- Розумні замки: Контроль доступу з можливістю віддаленого управління.
- Відеоспостереження: Камери з AI-аналітикою для розпізнавання облич та підозрілих рухів.
- Екологічні датчики: Датчики затоплення, вогню, диму та вуглекислого газу.
- Автоматизація будинку: Інтеграція з іншими розумними пристроями для повного контролю над будинком.

6. Bosch

Bosch Security Systems [7] пропонує комплексні рішення для безпеки:

- Інтегровані системи безпеки: Рішення для великих об'єктів та підприємств.
- Системи відеоспостереження: Камери з високою роздільною здатністю та інтелектуальними функціями.
- Системи контролю доступу: Рішення для контролю доступу з високим рівнем безпеки.
- Протипожежні системи: Комплексні системи для виявлення та боротьби з пожежами.

7. DSC (Digital Security Controls)

DSC [8] спеціалізується на сучасних охоронних системах:

- Панелі управління: Централізовані панелі з підтримкою різних типів датчиків.
- Датчики руху: Датчики з мінімізацією помилкових спрацьовувань.
- Комунікаційні модулі: Для підключення системи до інтернету або телефонних ліній.
- Інтеграція з іншими системами: Можливість інтеграції з системами відеоспостереження та контролю доступу.

Кожна з цих систем має свої переваги і може бути налаштована відповідно до специфічних потреб користувача. Вибір системи охорони залежить від багатьох факторів, включаючи розмір об'єкта, вимоги до безпеки, бюджет та інші особливості.

3.2. Опис структурної схеми системи охорони

Датчики руху (PIR) [14] та відкриття дверей:

- Датчики руху виявляють рух в охоронюваній зоні.
- Магнітні контакти встановлені на дверях, виявляють факт їх відкриття.

Мікроконтролер [15]:

- Приймає сигнали від датчиків руху та відкриття дверей.
- Обробляє ці сигнали і формує відповідні повідомлення.
- Передає повідомлення на локальний сервер через Wi-Fi з'єднання.

Локальний сервер [9]:

- Приймає дані від мікроконтролера.
- Обробляє отримані дані, визначає статус системи охорони.
- Зберігає історію подій в базі даних.

Графічний інтерфейс користувача (GUI) [13]:

- Відображає поточний статус системи охорони.
- Показує історію подій та сповіщення.
- Дозволяє користувачу налаштовувати параметри системи та отримувати звіти.

Мережеве обладнання:

- Забезпечує стабільне Wi-Fi з'єднання між мікроконтролером і локальним сервером.

3.4. Принцип роботи системи охорони

Збір даних:

- Датчики руху (PIR) фіксують рух в охоронюваній зоні і передають сигнал на мікроконтролер.
- Магнітні контакти на дверях фіксують факт їх відкриття і передають сигнал на мікроконтролер.

Обробка даних на мікроконтролері:

- Мікроконтролер обробляє отримані сигнали, визначаючи їх значущість та формуючи повідомлення.
- Передає дані на локальний сервер через Wi-Fi.

Обробка даних на локальному сервері:

- Сервер приймає дані, аналізує їх та оновлює статус системи охорони.
- Зберігає події в базі даних для подальшого аналізу.

Відображення та управління через GUI:

- GUI відображає поточний стан системи, включаючи активність датчиків та історію подій.
- Користувач може налаштовувати систему, переглядати звіти та отримувати сповіщення про події.

Структурна схема системи охорони забезпечує високу надійність, ефективність та зручність в експлуатації, дозволяючи своєчасно реагувати на загрози та зберігати повну історію подій для подальшого аналізу.

4. РОЗРОБКА ПРИНЦИПОВОЇ СХЕМИ

4.1. Вибір компонентів системи охорони

Для проектованого пристрою охорони, що розміщується на локальному сервері з GUI та backend, а також панеллю на мікроконтролері, який передає стани датчиків руху і відкриття дверей, необхідно вибрати відповідні компоненти. Основними елементами такої системи є датчики, мікроконтролер та програмне забезпечення для сервера.

- Датчик руху:

Рис. 4.1 Датчик руху (інфрачервоний) HC-SR501 [14]

Датчик руху (PIR-сенсор):

Модель: HC-SR501 [14].

Особливості: Працює при напрузі 5-20 В, має діапазон виявлення до 7 метрів, кут виявлення 120 градусів, регульований час затримки. Оскільки датчик з логічним рівнем 5В, а мікроконтролер 3.3В необхідно узгодження рівнів. Використаємо дільник напруги 1:2 з 2-ома резисторами на 1кОм.

- Датчик відкриття дверей (магнітний контакт):

Рис. 4.2 Геркон [16]

Модель: геркон нормально не замкнений [16].

Особливості: простота використання.

- Вибір мікроконтролера

Для збору даних з датчиків і передачі їх на локальний сервер доцільно використовувати мікроконтролери з вбудованою підтримкою Wi-Fi:

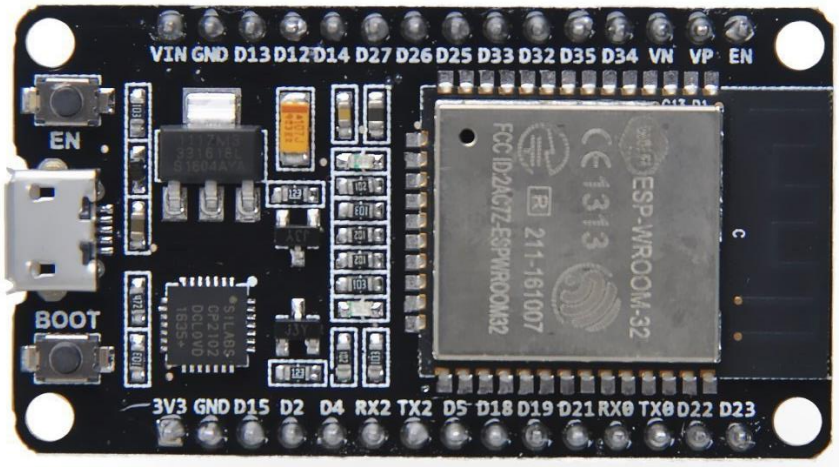


Рис. 4.3 ESP32 dev kit v1 [11]

Модель: Espressif [10] ESP32 dev kit v1 [11].

Особливості: Вбудований Wi-Fi модуль, працює при напрузі 3.3 В, має достатньо GPIO пінів для підключення датчиків, простий у програмуванні через Arduino IDE [12].

- Вибір пристрою наочно виведення інформації

Модель I2C LCD1602 [24]

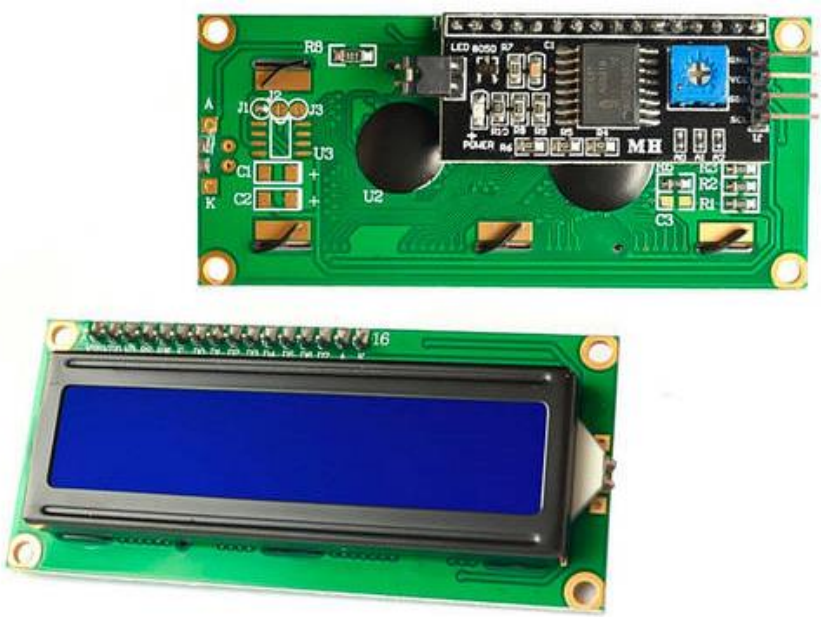


Рис. 4.5 I2C LCD1602 [24]

Переваги: простота підключення, роботи, надійність, доступність.

- Пристрій введення

Рис. Мембранна клавіатура 4*4 [25]

Модель: Мембранна клавіатура 4*4 [25]

Переваги: доступність, простота роботи.

Принципова схема

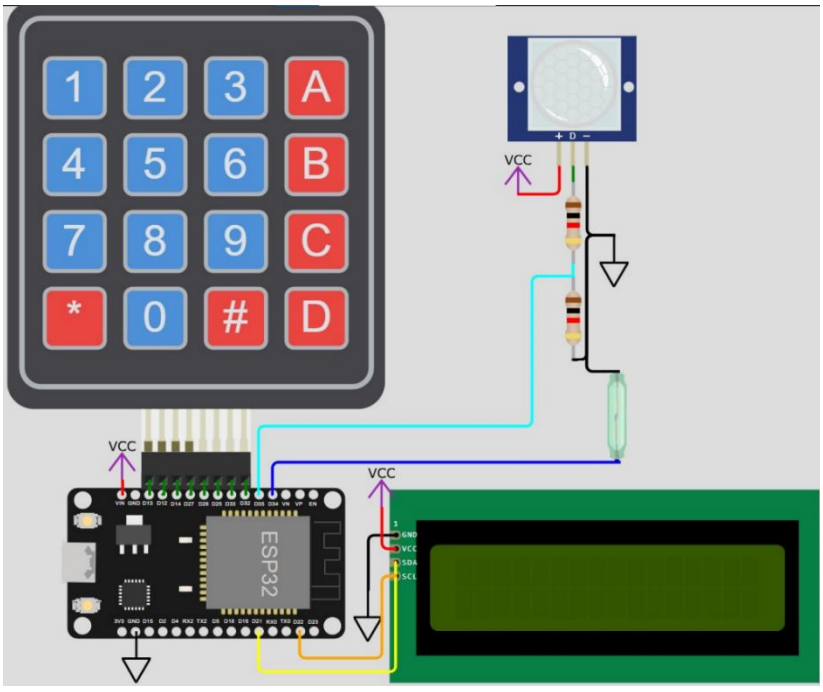


Рис. 4.1 Електрична схема панелі охоронної системи

На основі обраних компонентів можна розробити принципову схему системи охорони. Основні елементи схеми (див Додаток А):

- Датчики (PIR та магнітний контакт): Підключені до GPIO пінів мікроконтролера.
- Дисплей: Підключений до GPIO I2C пінів мікроконтролера.
- Клавіатура матрична: Підключена до GPIO пінів мікроконтролера.
- Мікроконтролер: Збирає дані з датчиків і передає їх на локальний сервер через Wi-Fi.
- Локальний сервер: Обробляє дані, зберігає їх у базі даних і відображає в GUI.

Мікроконтролер ESP збирає дані з датчиків.

- Обробляє сигнали та формує повідомлення.
- Передає дані на локальний сервер через Wi-Fi.

Ця схема забезпечує високу надійність і ефективність роботи системи охорони, дозволяючи своєчасно реагувати на загрози та надавати користувачам зручний інтерфейс для моніторингу і управління.

5. РОЗРОБКА АЛГОРИТМУ ТА ПРОГРАМИ РОБОТИ СИСТЕМИ

1. Вибір і підготовка апаратної платформи

На цьому етапі вирішується, який саме пристрій буде використовуватися для створення програмованої сигналізації. В даному випадку, це ESP32 - мікроконтролер, який є основою для реалізації функцій сигналізації. Також важливо обрати та підготувати різноманітні датчики, які використовуються для виявлення руху, стану дверей або вікон. Датчики руху можуть виявляти присутність людей в приміщенні, а датчики замків можуть визначати, чи зачинені двері або вікна.

2. Налаштування параметрів з'єднання

Однією з ключових функцій сучасної сигналізації є можливість сповіщати власників про події за допомогою мережі Інтернет. Для цього потрібно визначити параметри для підключення до мережі Wi-Fi (наприклад, ім'я мережі та пароль) та адресу сервера, на який будуть відправлятися дані з сигналізації.

3. Ініціалізація сенсорів і дисплея

Датчики та інші пристрої, які використовуються в сигналізації, повинні бути правильно підключені до мікроконтролера ESP32. Наприклад, датчики руху та замків можуть бути підключені через певні вихідні та вхідні піни на мікроконтролері. Для відображення інформації може використовуватися дисплей.

4. Ініціалізація та підключення до мережі Wi-Fi

Для того щоб сигналізація могла взаємодіяти з мережею Інтернет, необхідно налаштувати підключення до Wi-Fi. Це дозволить сигналізації взаємодіяти з сервером та іншими пристроями у мережі.

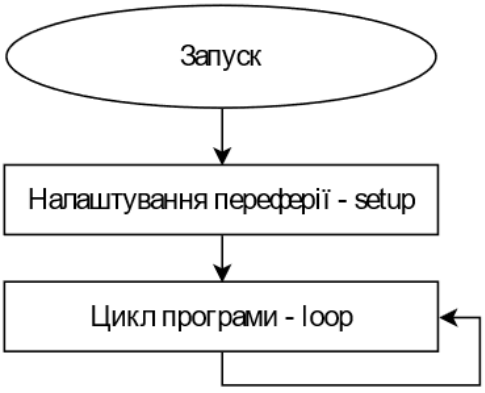


Рис. 5.1 Послідовність алгоритму

5. Основний цикл роботи

У цьому циклі програма постійно перевіряє стан сенсорів (наприклад, чи є рух у приміщенні або чи зачинені двері), та відображає інформацію на дисплеї. Якщо виявляються зміни у стані сенсорів, програма може виконати певні дії, наприклад, надіслати сповіщення на сервер.

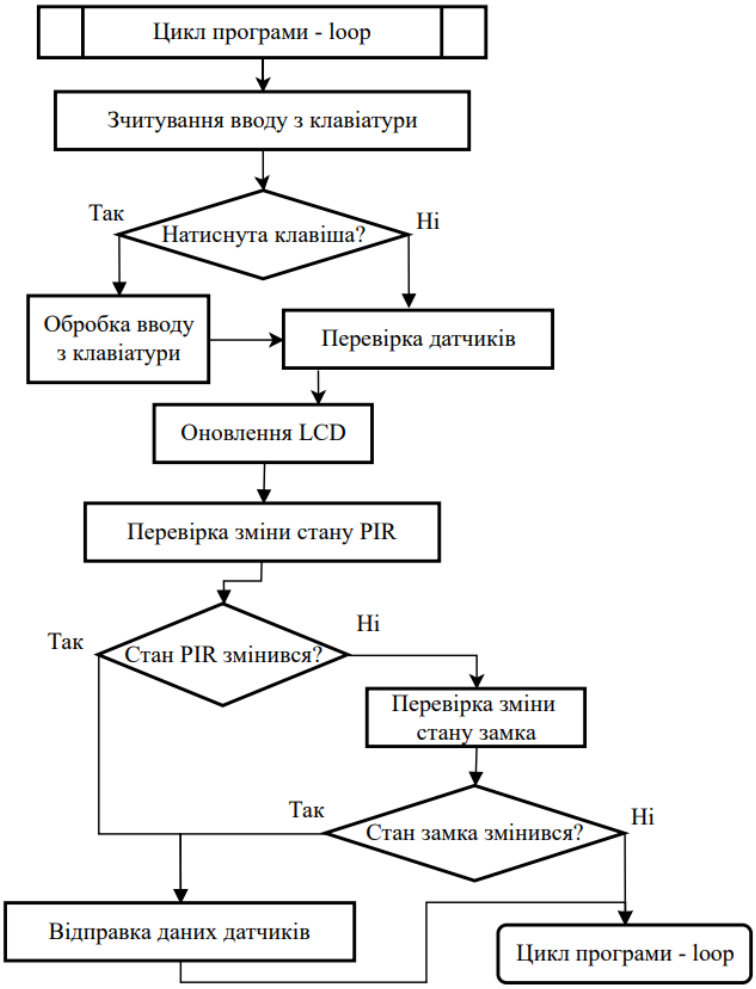


Рис. 5.2 Циклічний алгоритм опитування сенсорів

6. Обробка вводу з клавіатури

Клавіатура може бути використана для управління сигналізацією. Наприклад, власник може ввести спеціальну комбінацію клавіш для відправлення сигналу на сервер або зміни налаштувань.

7. Відправка даних на сервер

Однією з головних функцій сучасної сигналізації є можливість надсилання даних на сервер через Інтернет. Наприклад, коли спрацьовує датчик руху або замка, сигналізація може надіслати інформацію про це на сервер, щоб власник міг отримати сповіщення на свій смартфон або комп'ютер.

6. РОЗРОБКА GUI (графічного інтерфейсу користувача)

Home Assistant

Рис. 6.1 Демонстрація можливостей Home Assistant [20]

Home Assistant [20] – це популярна платформа для домашньої автоматизації з відкритим вихідним кодом, яка була створена для того, щоб зробити управління розумним будинком простим та ефективним. Вона дозволяє інтегрувати широкий спектр пристроїв та сервісів в одному інтуїтивно зрозумілому інтерфейсі. Основні характеристики Home Assistant включають:

- Відкритий вихідний код: Платформа є повністю безкоштовною та має відкритий вихідний код, що дозволяє користувачам модифікувати її під свої потреби. Це також сприяє активному розвитку та підтримці спільноти.
- Широка підтримка пристроїв та сервісів: Home Assistant підтримує більше 1000 різних інтеграцій, що дозволяє підключати різноманітні пристрої, такі як розумні лампи, термостати, камери безпеки, датчики руху, дверні замки та багато іншого.
- Автоматизація: Home Assistant підтримує створення складних сценаріїв автоматизації. Це дозволяє налаштовувати дії на основі подій, умов або часу. Наприклад, автоматичне вимкнення світла при виході з будинку або відправлення повідомлення при спрацюванні сигналізації.
- Безпека та конфіденційність: Home Assistant дозволяє зберігати всі дані локально, що підвищує рівень безпеки та конфіденційності. Платформа також підтримує SSL/TLS для захищених з'єднань.

Протокол MQTT

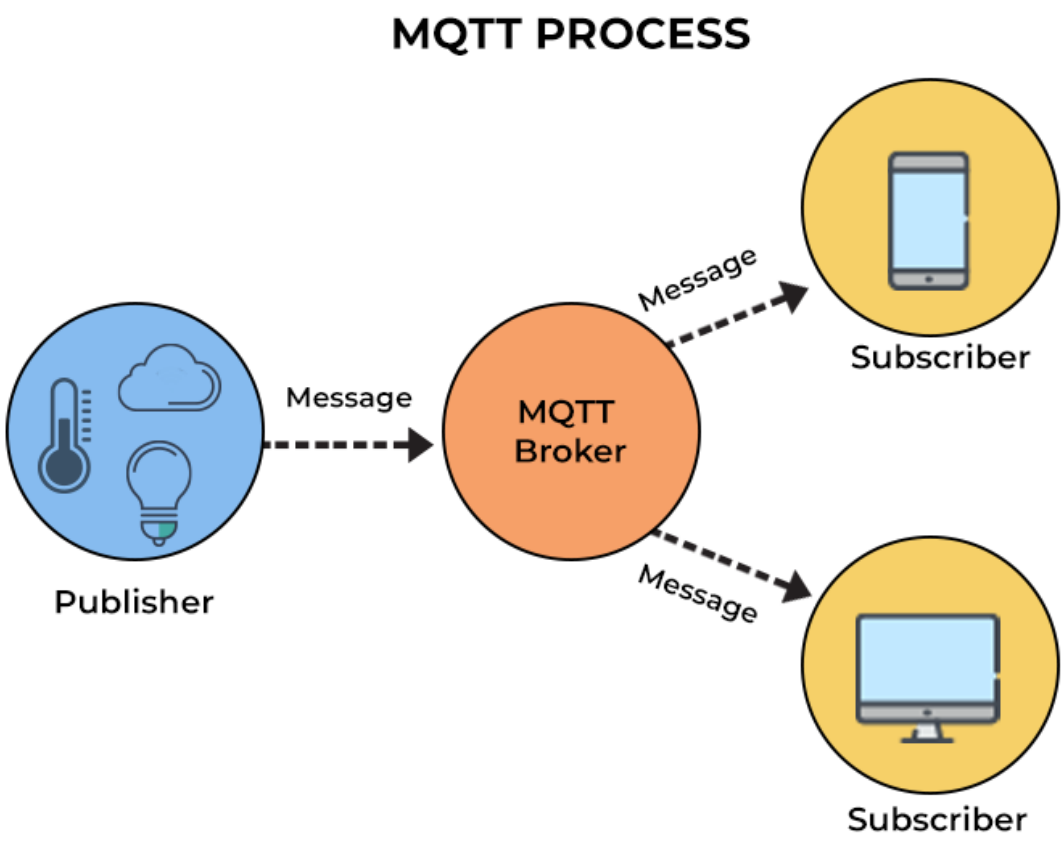


Рис. 6.2 Приклад MQTT мережі [21]

MQTT (Message Queuing Telemetry Transport) [21] – це легкий протокол обміну повідомленнями, оптимізований для пристроїв з обмеженими ресурсами і високошвидкісних мереж з низькою пропускною здатністю. Основні особливості MQTT включають:

- Принцип публікації/підписки: Протокол працює на основі моделі публікації/підписки, що дозволяє клієнтам публікувати повідомлення на певні теми і підписуватися на ці теми, щоб отримувати відповідні повідомлення. Це зменшує навантаження на мережу та спрощує обмін даними.
- Легкість та ефективність: MQTT був розроблений для роботи в умовах обмежених ресурсів (наприклад, обмеженої потужності процесора та пам'яті) і в ненадійних мережах. Це робить його ідеальним для використання в IoT-проектах.
- Надійність: Протокол підтримує три рівні якості обслуговування (QoS), які визначають надійність доставки повідомлень
- Масштабованість: MQTT-брокери можуть обробляти тисячі підключень одночасно, що робить протокол масштабованим і придатним для великих систем з багатьма пристроями.

Використання Home Assistant та MQTT для створення GUI

При створенні GUI для програмованого пристрою сигналізації на основі ESP та MQTT, Home Assistant виступає як центральний вузол для збору, обробки та відображення даних з датчиків. Ваша система складається з наступних компонентів:

- ESP-модуль: Цей мікроконтролер збирає дані з датчиків відкриття дверей, руху та клавіатури, а потім відправляє ці дані на MQTT-брокер.
- MQTT-брокер: MQTT-брокер, наприклад Mosquitto [22], приймає повідомлення від ESP і передає їх до Home Assistant.
- Home Assistant: Платформа підписується на відповідні теми MQTT, отримує дані та відображає їх у графічному інтерфейсі користувача.

Таким чином, поєднання Home Assistant та протоколу MQTT забезпечує високий рівень функціональності та зручності, дозволяючи ефективно управляти системою сигналізації та реагувати на події в режимі реального часу.

Створення GUI:

- Налаштування панелей управління: В Home Assistant є Lovelace Dashboard, де можна налаштувати панелі управління для відображення інформації з датчиків.
- Інтеграція пристроїв: Датчики руху, відкриття дверей та клавіатура додаються як окремі пристрої. Це дозволяє візуалізувати їхній стан та керувати ними через інтерфейс.
- Автоматизація: За допомогою автоматизації в Home Assistant можна налаштувати дії на основі даних з датчиків. Наприклад, відправка сповіщень на телефон при спрацюванні датчика руху або відкриття дверей.

7. ТЕСТУВАННЯ

Тестування системи охорони є для мене критично важливим етапом у процесі її впровадження та експлуатації. Основні причини, чому я вважаю тестування настільки важливим, включають:

1. Забезпечення безпеки: Система охорони покликана захищати моє майно та забезпечувати безпеку для мене та моєї родини. Виявлення і усунення помилок у системі дозволяє уникнути небажаних наслідків, таких як проникнення зловмисників.
2. Підвищення надійності: Регулярне тестування допомагає виявити потенційні слабкі місця в системі, що можуть призвести до відмови або некоректної роботи. Це забезпечує більш стабільну і надійну роботу системи.
3. Відповідність вимогам: Тестування дозволяє перевірити, чи відповідає система всім технічним і функціональним вимогам, які я встановив для своєї безпеки.
4. Забезпечення сумісності: Перевірка системи допомагає впевнитися, що всі її компоненти (датчики, клавіатура, сервер) коректно взаємодіють один з одним і передають дані без помилок.

Для проведення тестування системи охорони були створені наступні умови симуляції.

1. Панель управління в онлайн симуляторі Wokwi.com [23]

Для тестування системи була створена онлайн симуляція в Wokwi.com. Ця платформа дозволяє емуляцію роботи мікроконтролера ESP32 разом з підключеними датчиками та дисплеєм.

2. Локальний сервер з доступом через DMZ

Для обробки та зберігання даних використовується локальний сервер з налаштованим доступом через DMZ (для доступу з глобальної мережі інтернет).

3. Графічний інтерфейс Home Assistant

Графічний інтерфейс відкривається в браузері, забезпечуючи легкий доступ до всіх функцій системи.

Таким чином, тестування системи охорони проводиться в максимально наближених до реальних умовах, що дозволяє виявити та усунути потенційні проблеми до їх виникнення в реальному житті.

Для тестування моєї системи охорони , я виконую наступні кроки:

1. Ініціалізація та налаштування (згідно ДОДАТОК Г та Ґ):
    a. Підключаю та налаштовую всі датчики (інфрачервоний датчик руху, герконний датчик відкриття дверей, клавіатура).
    b. Переконуюсь, що MQTT-брокер налаштований і готовий до прийому даних.
    c. Налаштовую Home Assistant для прийому і відображення даних з датчиків через MQTT.

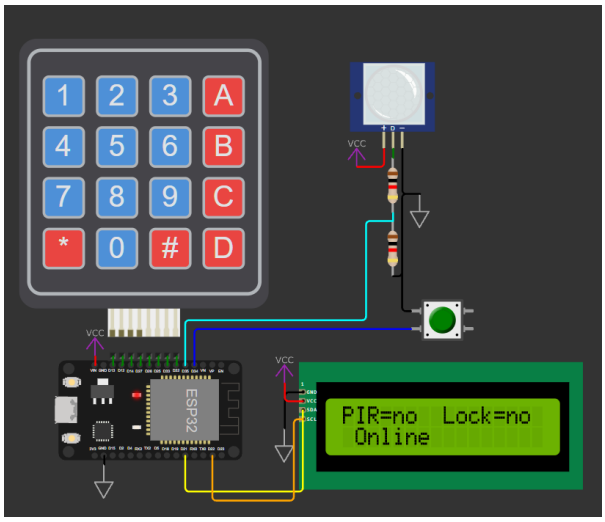


Рис. 7.1 Ввімкнення пристрою

2. Перевірка з'єднань:
    a. Перевіряю, чи всі датчики коректно підключені та працюють.
    b. Відправляю тестові повідомлення через MQTT для кожного датчика і переконуюсь, що вони відображаються в Home Assistant.

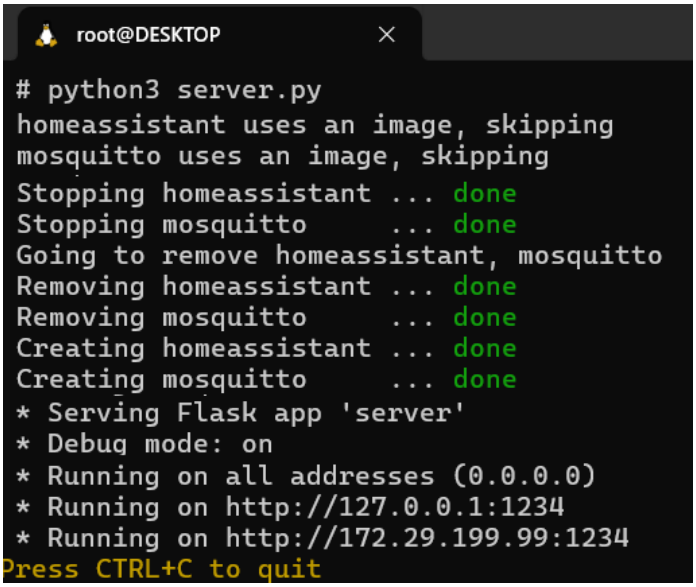


Рис. 7.2 Запуск серверу

3. Тестування інфрачервоного датчика руху:

a. Провожу перевірку виявлення руху в різних зонах покриття датчика.

b. Переконуюсь, що при виявленні руху надсилаються коректні повідомлення MQTT і вони правильно відображаються в Home Assistant.

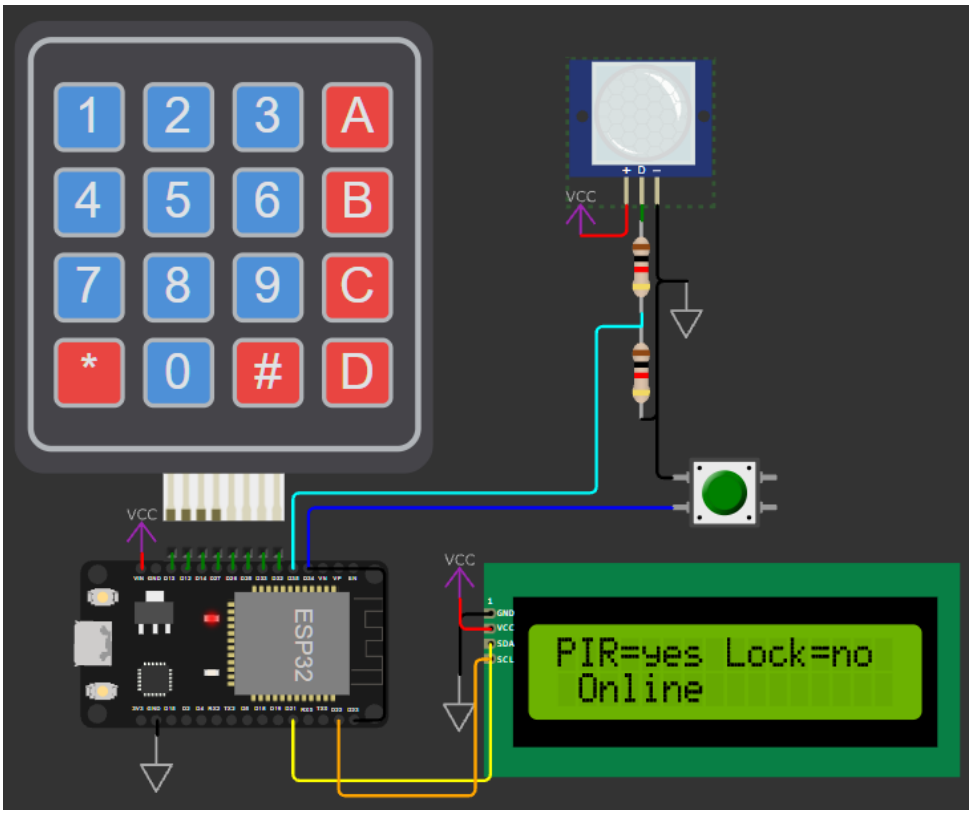


Рис. 7.3.1 Симуляція руху

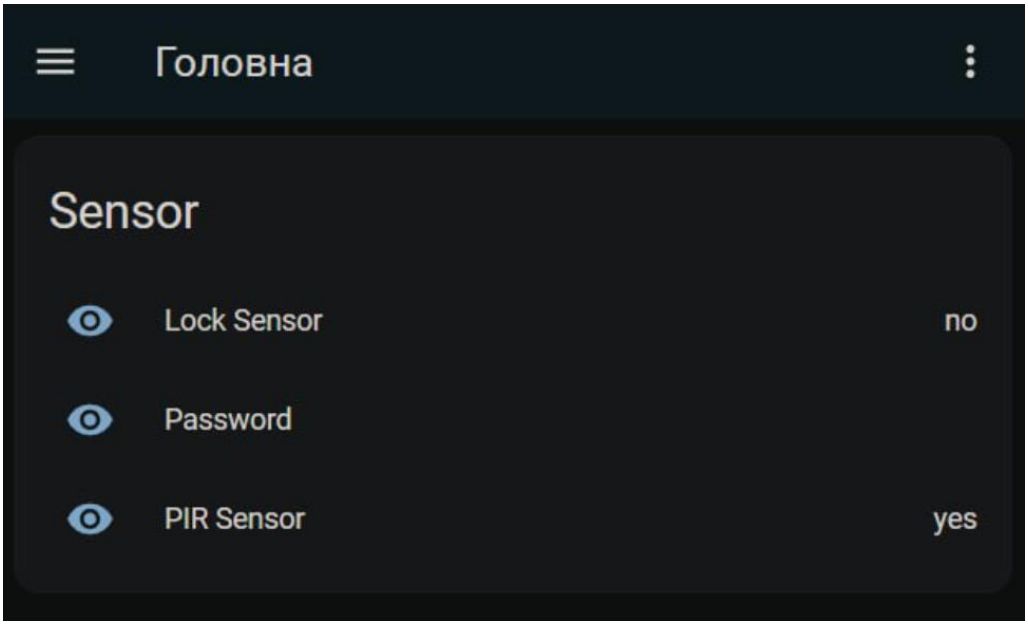


Рис. 7.3.2 Реакція Home Assistant на рух

4. Тестування герконного датчика відкриття дверей (кнопка емулює геркон):
    a. Відкриваю та закриваю двері, на яких встановлено герконний датчик.
    b. Перевіряю, що при відкритті та закритті дверей надсилаються коректні повідомлення MQTT і вони правильно відображаються в Home Assistant.

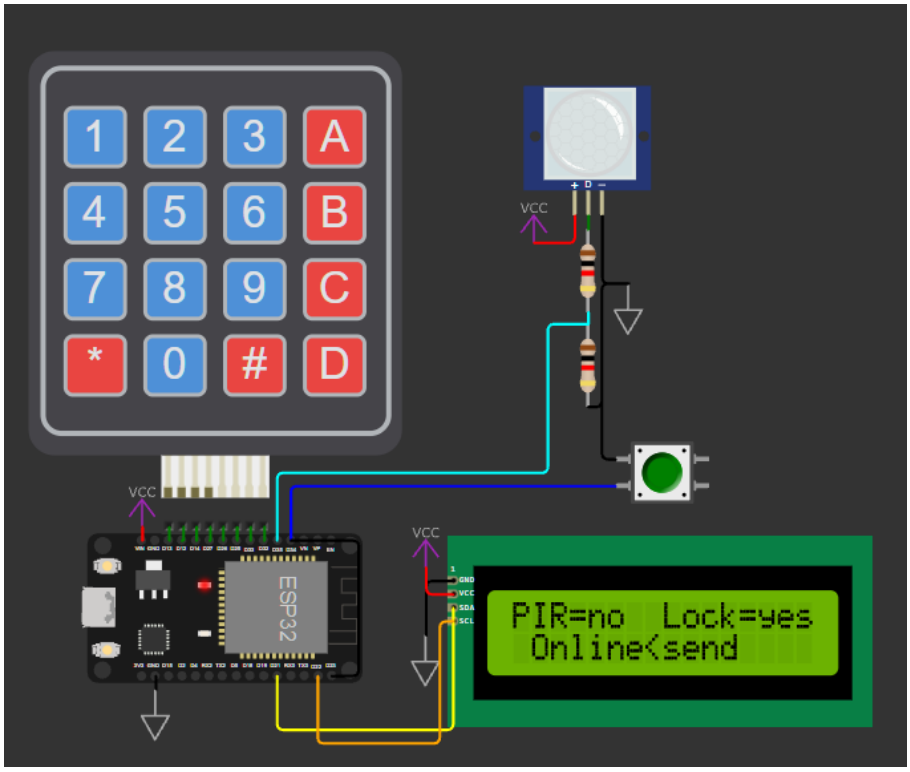


Рис. 7.4.1 Натискання кнопки емуляція спрацювання геркону)

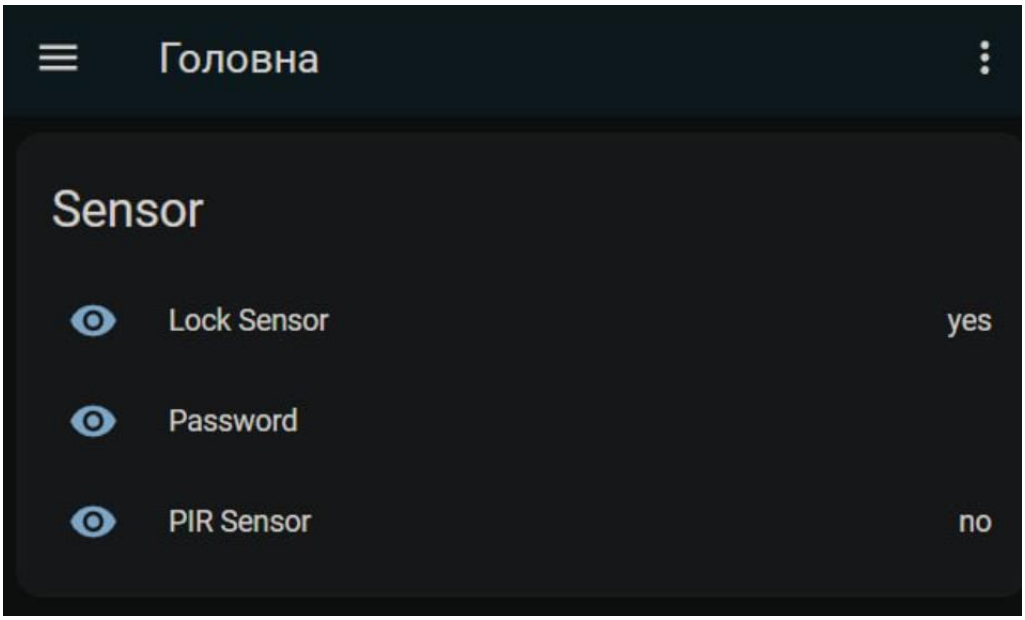


Рис. 7.4.2 Реакція Home Assistant на датчик відкриття дверей

5. Тестування клавіатури з паролем:
    a. Вводжу правильний та неправильний паролі.
    b. Переконуюсь, що при введенні паролю надсилаються коректні повідомлення MQTT і вони правильно відображаються в Home Assistant.

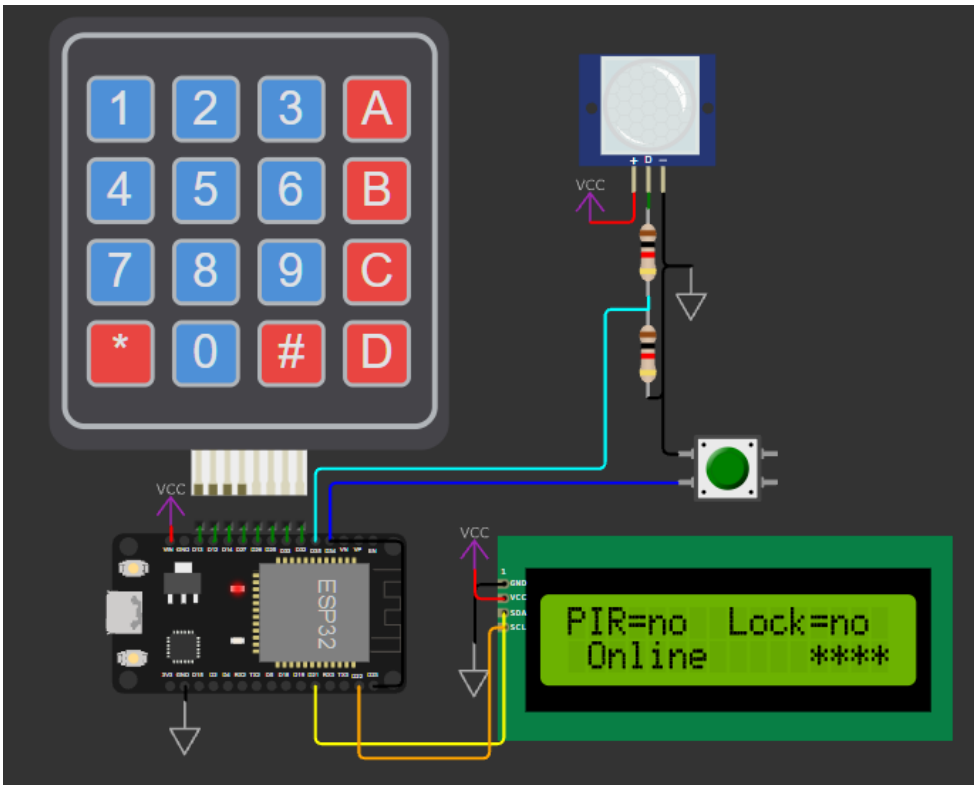


Рис. 7.5.1 Введеня 4-ох значного паролю

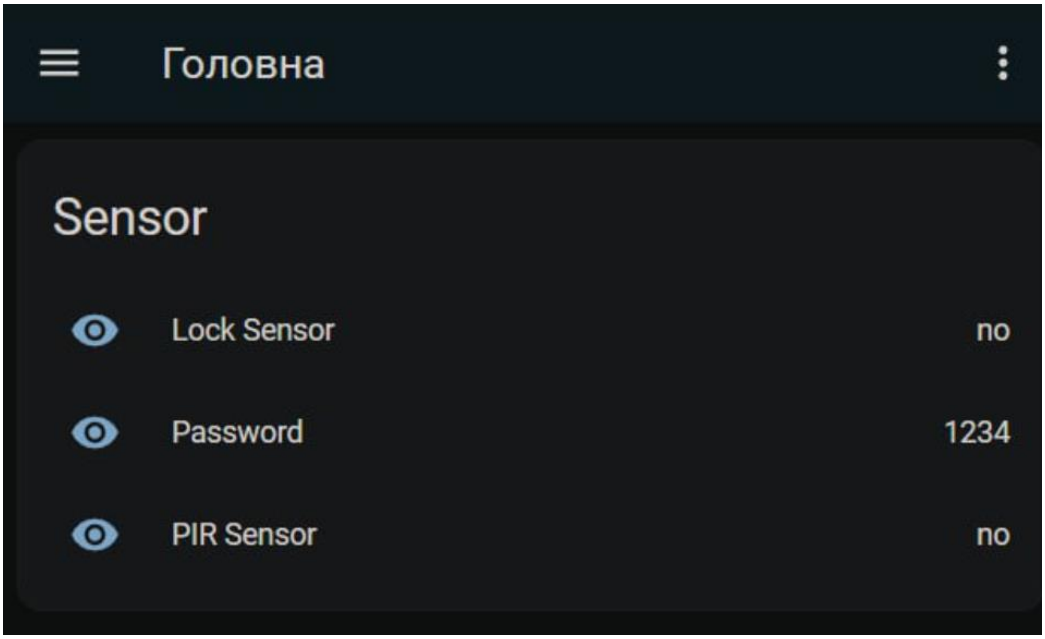


Рис. 7.5.2 Реакція Home Assistant на введеня 4-ох значного паролю (1234)

Після проведення комплексного тестування моєї системи охорони, що базується на інфрачервоному датчику руху, герконному датчику відкриття дверей та клавіатурі з паролем, з передачею даних через MQTT на локальний сервер Home Assistant, були виявлені наступні проблеми та недоліки:

Невелика Затримка:

- Виявлено невелику затримку в 1 секунду між реакцією датчиків на зміну показань і відображенням цієї інформації у Home Assistant. Це може бути критичним у випадку швидкої реакції на загрозу.

Інші Недоліки:

- Інколи спостерігалися випадки некоректного відображення стану датчиків у Home Assistant, зокрема, при одночасній активації декількох датчиків.
- Невеликий відсоток тестових повідомлень MQTT не досягав сервера Home Assistant, що призводило до втрати даних.

Для вирішення виявлених проблем я застосував наступні методи:

1. Оптимізація мережевих налаштувань:
   Налаштував пріоритетність передачі даних від охоронних датчиків у локальній мережі для зменшення затримки. Використав QoS (Quality of Service) в налаштуваннях роутера для забезпечення швидшої передачі важливих повідомлень.
2. Покращення налаштувань mqtt:
   Налаштував параметри з'єднання MQTT, зокрема, зменшив час очікування з'єднання та збільшив частоту передачі даних, що допомогло знизити затримку.
3. Оптимізація системи home assistant:
   Оновив Home Assistant до останньої версії, яка включає поліпшення продуктивності і оптимізації роботи з MQTT.

Результати

Після впровадження цих змін та повторного тестування система показала значно кращу продуктивність. Затримка реакції на зміну показань зменшилася до мінімуму, що не впливає на загальну функціональність системи. Проблеми з некоректним відображенням стану датчиків і втратою даних також були усунені.

Загалом, система охорони тепер працює стабільно і надійно.

ВИСНОВКИ

У результаті виконання курсової роботи розроблено комплексну систему безпеки для розумного будинку, яка включає в себе програмований пристрій сигналізації, який базується на мікроконтролері ESP32. До цього пристрою були підключені датчик відкриття дверей, датчик руху та панель вводу паролю. За допомогою технології Wi-Fi дані з цих датчиків передавалися на локальний сервер за допомогою протоколу MQTT, і значення датчиків відображалися у системі управління розумним будинком, такі як Home Assistant.

У ході роботи було розглянуто і реалізовано наступні етапи:

1. Підготовка та налаштування ESP32 для забезпечення з'єднання з мережею Wi-Fi та обробки даних з підключених датчиків.
2. Інтеграція датчиків відкриття дверей та руху з ESP32 для зчитування стану середовища.
3. Розробка та реалізація алгоритму обробки даних та прийняття рішень для активації сигналізації в залежності від отриманих від датчиків даних.
4. Створення панелі вводу паролю для активації та деактивації сигналізації.
5. Налаштування та тестування зв'язку з локальним сервером та інтеграція з системою Home Assistant для моніторингу та керування системою.

Отримані результати свідчать про успішну інтеграцію компонентів системи та їх коректну роботу. Під час тестування система продемонструвала надійність та ефективність у виявленні подій, які можуть свідчити про порушення безпеки в приміщенні.

У розробленій системі забезпечено повну незалежність від зовнішніх компаній, оскільки вона працює на локальному рівні. Це означає, що дані, зібрані датчиками та оброблені мікроконтролером ESP32, передаються лише на локальний сервер через Wi-Fi, а не через хмарні сервіси. Такий підхід забезпечує високий рівень конфіденційності та безпеки даних користувачів, оскільки інформація не опиняється у власності сторонніх постачальників послуг.

Такий локальний підхід також забезпечує більший контроль за системою та її функціональністю, оскільки користувач може самостійно налаштовувати та керувати системою без необхідності залучення зовнішніх постачальників послуг.

Розроблена система безпеки для розумного будинку може бути використана для моніторингу та захисту житлових приміщень, а також для інтеграції з іншими системами автоматизації будинку для забезпечення більш комплексного підходу до безпеки та комфорту користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ajax [Електронний ресурс]. Режим доступу: https://ajax.systems/ua/
2. Honeywell SECURITY SYSTEMS [Електронний ресурс]. Режим доступу: https://www.honeywellhome.com/us/en/products/security/security-systems/
3. ADT® Home Alarm Systems | Home Security Systems Provider [Електронний ресурс]. Режим доступу: https://www.adt.com/
4. Ring Home Security Products [Електронний ресурс]. Режим доступу: https://ring.com/collections/ring-bundles
5. SimpliSafe Home Security Systems | Wireless Home Security [Електронний ресурс]. Режим доступу: https://simplisafe.com/
6. Vivint® Smart Home Security & Alarm Systems [Електронний ресурс]. Режим доступу: https://www.vivint.com/
7. Bosch Security Systems [Електронний ресурс]. Режим доступу: https://www.boschsecurity.com/xc/en/
8. Digital Security Controls [Електронний ресурс]. Режим доступу: https://www.dsc.com/
9. Локальний сервер [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Сервер
10. Espressif Systems: Wireless SoCs, Software, Cloud and IoT [Електронний ресурс]. Режим доступу: https://www.espressif.com/
11. ESP-IDF Programming Guide - ESP32 [Електронний ресурс]. Режим доступу: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html
12. Arduino Documentation [Електронний ресурс]. Режим доступу: https://docs.arduino.cc/
13. GUI - Графічний інтерфейс користувача [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Графічний_інтерфейс_користувача
14. Датчик руху інфрачервоний (PIR Sensor) [Електронний ресурс]. Режим доступу: https://www.mini-tech.com.ua/ua/datchik-dvizheniya-infrakrasniy-pir-sensor-hc-sr501

15. Мікроконтролер [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Мікроконтролер

16. Геркон [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Геркон

17. Python [Електронний ресурс]. Режим доступу: https://www.python.org/

18. Flask [Електронний ресурс]. Режим доступу: https://flask.palletsprojects.com/en/3.0.x/

19. GPIO [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/GPIO

20. Home Assistant [Електронний ресурс]. Режим доступу: https://www.home-assistant.io/

21. Протокол MQTT [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/MQTT

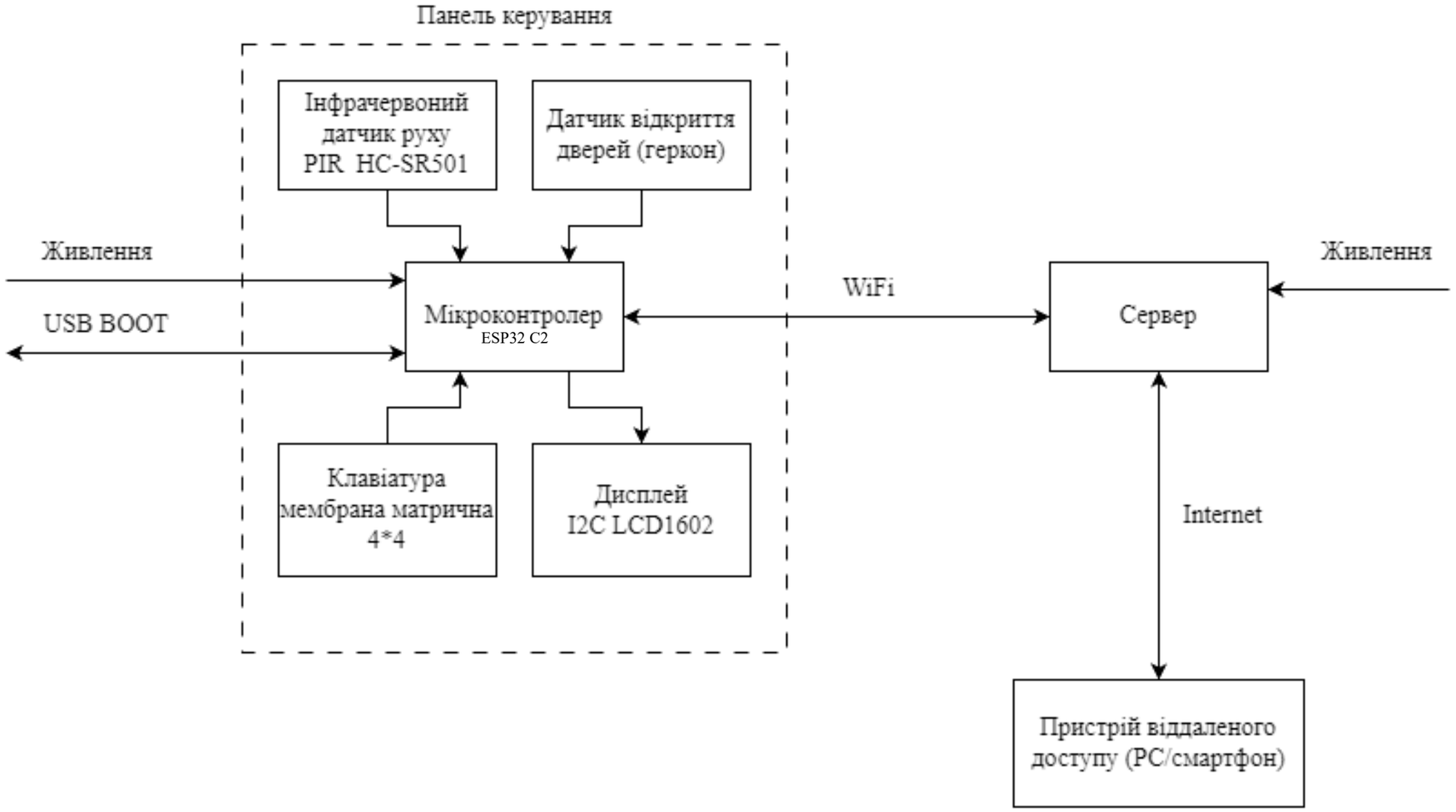
22. MQTT-брокер Mosquitto [Електронний ресурс]. Режим доступу: https://mosquitto.org

23. Wokwi - Online ESP32, STM32, Arduino Simulator [Електронний ресурс]. Режим доступу: https://wokwi.com/

24. LCD1602 I2C Module [Електронний ресурс]. Режим доступу: https://www.waveshare.com/wiki/LCD1602_I2C_Module

25. Keypad [Електронний ресурс]. Режим доступу: https://docs.oyoclass.com/unoeditor/Libraries/keypad/

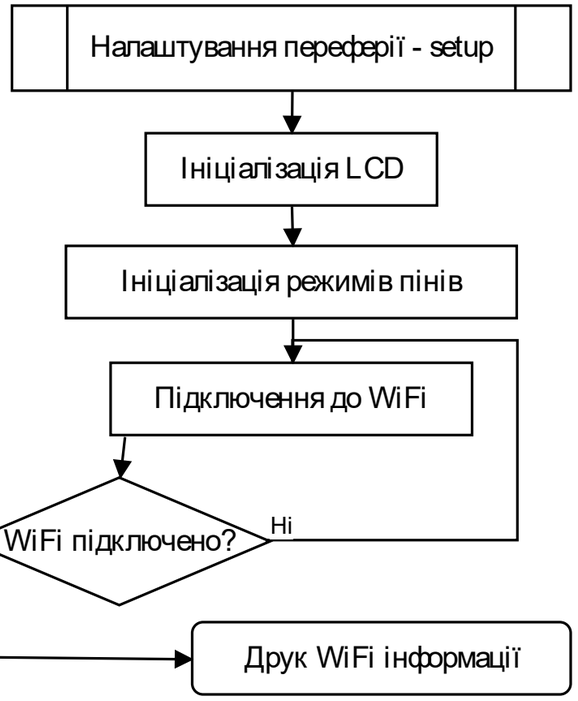
ДОДАТОК А

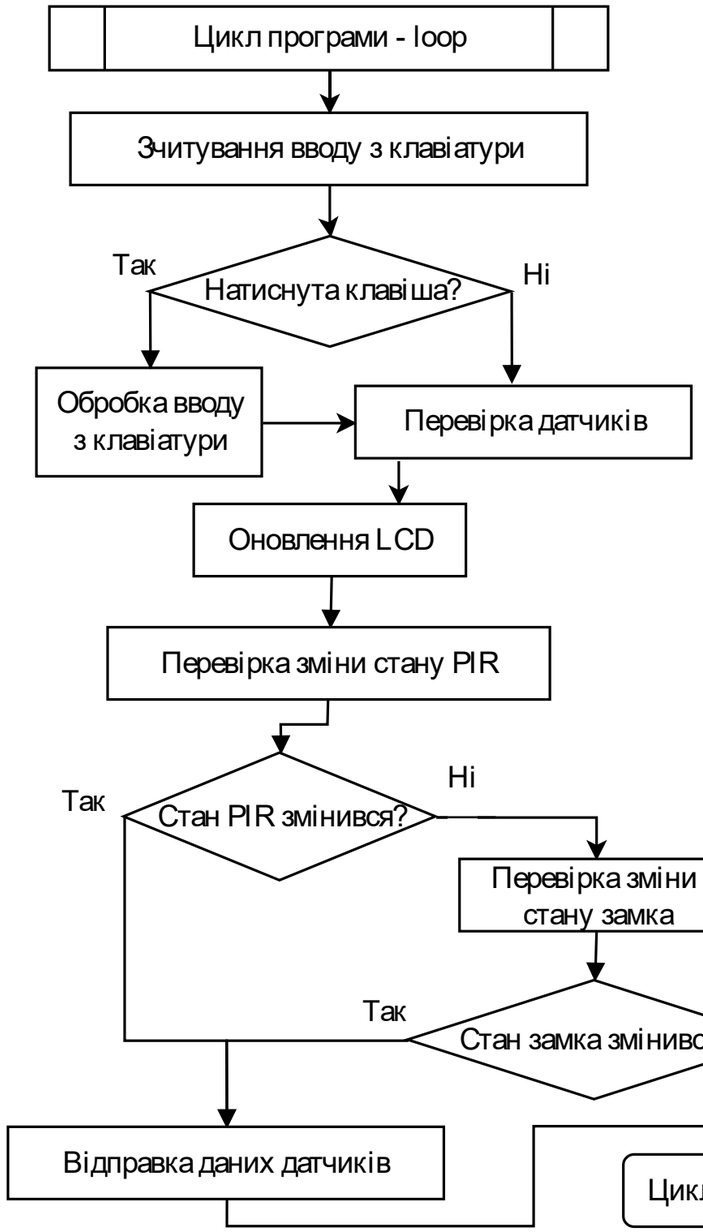


					КР ПМ-11.12.00.00 СХ	
Зм	Арк	N документа	Підпис	Дата	Програмований пристрій сигналізації Структурна схема	Літ / Маса / Масштаб
Розробив		Погорєлов Б.				
Перевірив		Самборська В.В				Аркуш / Аркушів
Т. контр.						
Н. контр.						КПІ, ПБФ, ПМ-11
Затвердив		Киричук Ю. В.				

Первинний приклад | Спр. № | Підпис і дата | Інв. № дубл. | Зам. Інв. № | Підпис і дата | Інв. № орг.

ДОДАТОК Б

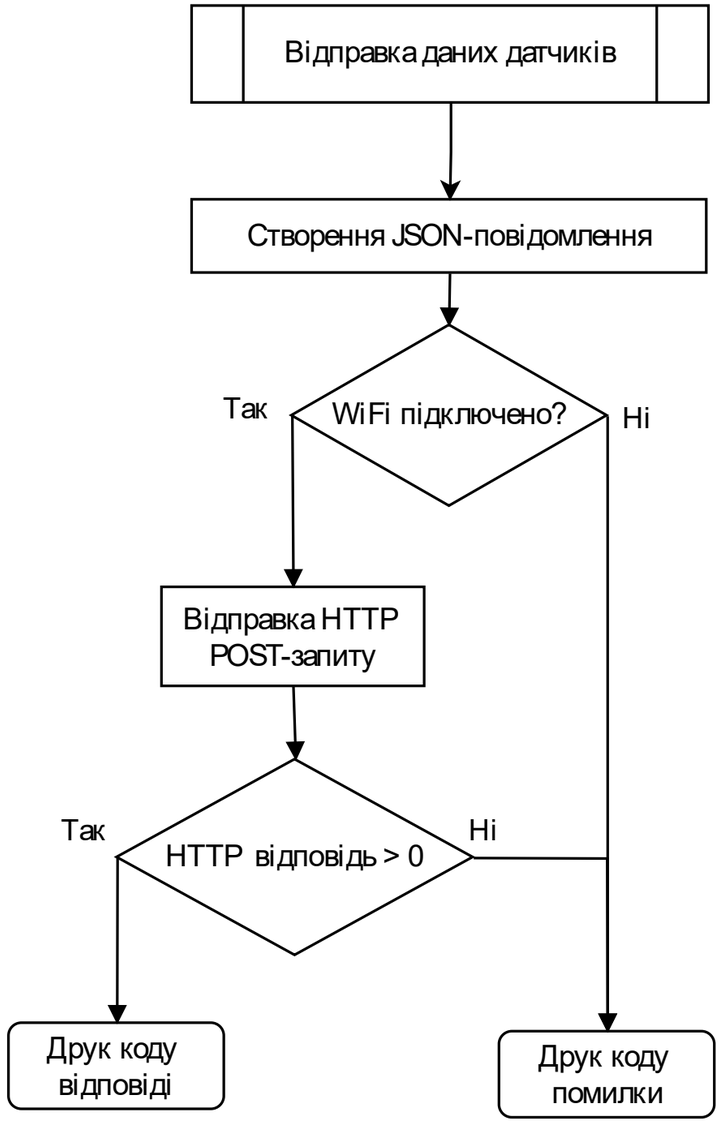


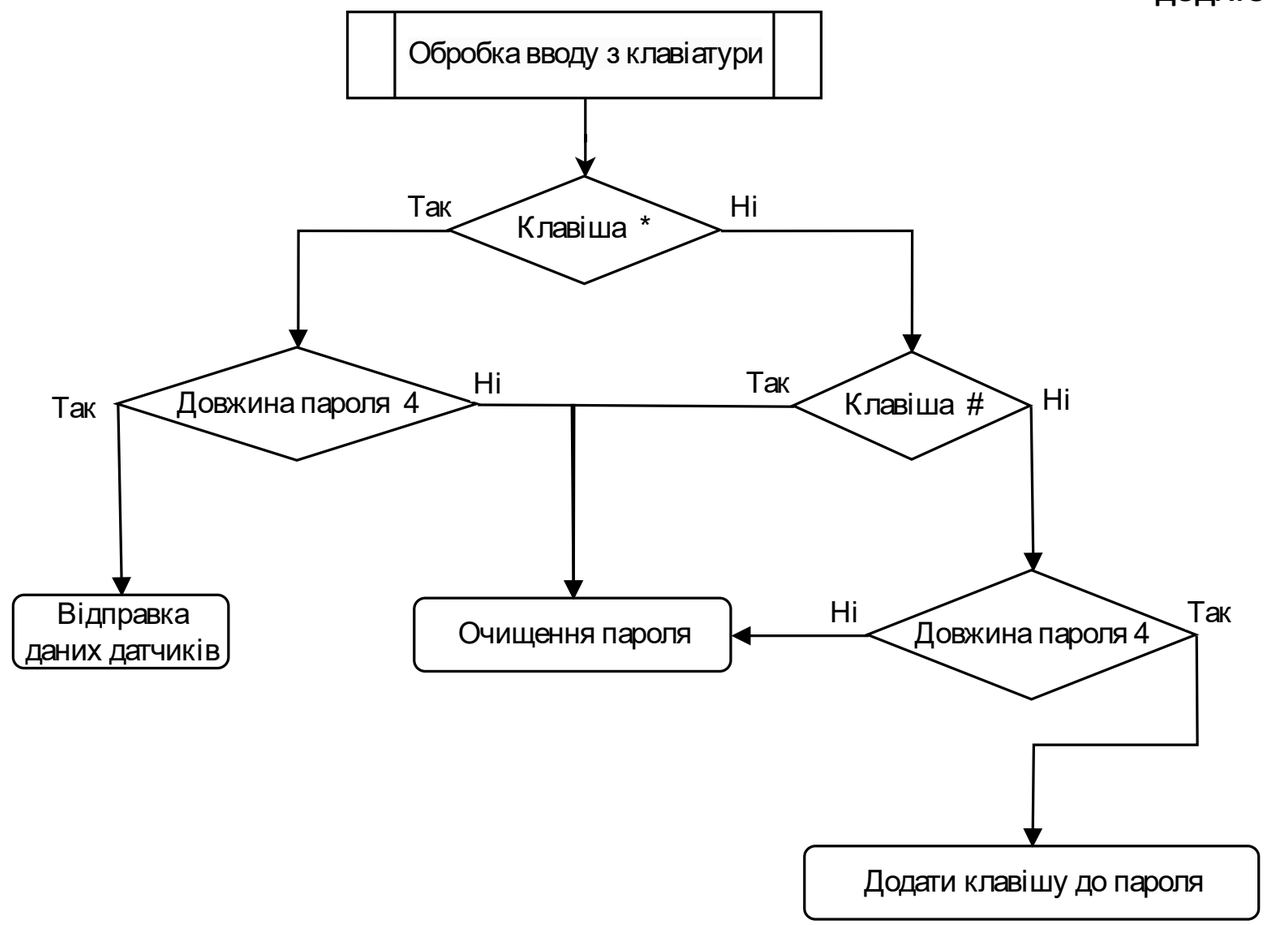


					КР ПМ-11.12.00.01 БС	
Зм	Арк	N документа	Підпис	Дата	Програмований пристрій сигналізації Блок- схема алгоритму роботи	Літ / Маса / Масштаб
Розробив		Погорєлов Б.				
Перевірив		Самборська В.В				Аркуш / Аркушів
Т. контр.						
Н. контр.						КПІ, ПБФ, ПМ-11
Затвердив		Киричук Ю. В.				

Первинний приклад | Спр. № | Підпис і дата | Інв № дубл. | Зам. Інв. № | Підпис і дата | Інв. № ор.

ДОДАТОК В





					КР ПМ-11.12.00.02 БС	
Зм	Арк	N документа	Підпис	Дата	Програмований пристрій сигналізації Блок- схема алгоритму роботи	Літ / Маса / Масштаб
Розробив		Погорєлов Б. Ю				
Перевірив		Самборська В.В				Аркуш / Аркушів
Т. контр.						
Н. контр.						КПІ, ПБФ, ПМ-11
Затвердив		Киричук Ю. В.				

Первинний приклад | Спр. № | Підпис і дата | Інв № дубл. | Взам. Інв. № | Підпис і дата | Інв. № ор.

ДОДАТОК Г

Код мікроконтролера esp32.ino

```
// ESP32 dev board
#include <WiFi.h>
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>
#include <HTTPClient.h>

////////////////////////////////////////////////
const char* WIFI_SSID     = " ";
const char* WIFI_PASSWORD = " ";
const char* server_address= " :1234/sensor";
const char* key           = "qwerty1234";
////////////////////////////////////////////////


// Sensor pins
const byte pin_sensor_pir  = 35;
const byte pin_sensor_lock = 23;

// Keypad settings
byte pin_rows[4]   = {13, 12, 14, 27};
byte pin_column[4] = {26, 25, 33, 32};
char keys[4][4] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
Keypad = Keypad(makeKeymap(keys), pin_rows, pin_column, 4, 4);

// LCD settings
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Variables
String password = "esp_test";
bool pirState   = false;
bool lockState  = false;
int sendCounter = 0;

bool pirStatePrev   = false;
bool lockStatePrev  = false;

void setup() {
  lcd.init();
  lcd.backlight();
```

```
  pinMode(pin_sensor_pir, INPUT);
  pinMode(pin_sensor_lock, INPUT_PULLUP);
  Serial.begin(115200);

  lcd.print("Try connect");
  lcd.setCursor(0, 1);
  lcd.print(WIFI_SSID);
  Serial.print("\nConnecting to WiFi SSID " + String(WIFI_SSID));
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.print("\nWiFi connected. IP address: " + String(WiFi.localIP()));
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("WiFi connected");

  // No need to setup MQTT
  sendSensorData();
}


void loop() {
  // Check keypad input

  char key = keypad.getKey();
  if (key) handleKeyInput(key);

  // Check PIR, lock sensor
  pirState  =  digitalRead(pin_sensor_pir);
  lockState = !digitalRead(pin_sensor_lock);

  // Display status
  lcd.setCursor(0, 0);
  lcd.print("PIR=" + String(pirState ? "yes" : "no ") +
            " Lock=" + String(lockState ? "yes" : "no ") );

  lcd.setCursor(0, 1);
  lcd.print(WiFi.status() == WL_CONNECTED ? " Online" : "Offline");

//  lcd.setCursor(7, 1);
//  lcd.print(sendCounter-- ? "<send" : "     ");
  lcd.setCursor(7, 1);
  lcd.print("     ");

  lcd.setCursor(12, 1);
  for (byte i = 0; i < 4; i++)
```

```
    lcd.print(i < password.length() ? '*' : ' ');

  if(pirState != pirStatePrev){
    pirStatePrev = pirState;
    if(pirState) sendSensorData();
   }
   if(lockState != lockStatePrev){
    lockStatePrev = lockState;
    sendSensorData();
   }
}


void handleKeyInput(char key) {
  if (key == '*' && password.length() == 4) {
    sendSensorData();
    password = "";
  }
  else if (key == '#') password = "";
  else if (password.length() < 4) password += key;
  else password = "";
}

void sendSensorData() {
  String json = "{ \"pass\":\"" + password + "\","
                "\"pir\":" + String(pirState ? "true" : "false") +
                ",\"lock\":" + String(lockState ? "true" : "false") +
                ",\"key\":\"" + String(key) +"\""+
               "}";
  Serial.println("JSON: " + json);

  lcd.setCursor(7, 1);
  lcd.print("<send");

  if (WiFi.status() == WL_CONNECTED) {
    sendCounter = 1;
    HTTPClient http;
    http.begin(server_address);
    http.addHeader("Content-Type", "application/json");
     int httpResponseCode = http.POST(json);
    if (httpResponseCode > 0)
      Serial.print("HTTP Response code: ");
    else
      Serial.print("Error code: ");
    Serial.println(httpResponseCode);
    http.end();
  }
```

ДОДАТОК Ґ

Код docker-compose.yaml

```yaml
version: '3'
services:
  homeassistant:
    container_name: homeassistant
    image: homeassistant/home-assistant
    volumes:
      - ./config/homeassistant/:/config/
    ports:
      - "8123:8123"
      - "1833:1833"
    networks:
      mynetwork:
        ipv4_address: 172.28.0.2
  mosquitto:
    container_name: mosquitto
    image: eclipse-mosquitto:1.6.9
    volumes:
      - ./config/mosquitto/:/mosquitto/data/
    ports:
      - "1883:1883"
    networks:
      mynetwork:
        ipv4_address: 172.28.0.3
networks:
  mynetwork:
    ipam:
      config:
        - subnet: 172.28.0.0/16
```

Код homeassistant/configuration.yaml

```yaml
mqtt:
  sensor:
    - name: "Password"
      unique_id: 5bc54f8b-7914-46fd-897b-fcee98e92409
      state_topic: "sensor/jsondata"
      value_template: "{{ value_json.pass }}"
    - name: "PIR Sensor"
      unique_id: 5bc54f8b-7914-46fd-897b-fcee98e9240
      state_topic: "sensor/jsondata"
      value_template: >
        {% if value_json.pir == 1 %}
          yes
        {% else %}
          no
        {% endif %}
    - name: "Lock Sensor"
      unique_id: 5bc54f8b-7914-46fd-897b-fcee98e9240b
      state_topic: "sensor/jsondata"
      value_template: >
        {% if value_json.lock == 1 %}
          yes
        {% else %}
          no
        {% endif %}
```

Код server.py

```python
# pip install paho-mqtt flask
from flask import Flask, request, jsonify
from datetime import datetime
import json
import paho.mqtt.client as mqtt
import subprocess

app = Flask(__name__)

# Ключ для перевірки
SECRET_KEY = "qwerty1234"

# MQTT налаштування
MQTT_BROKER = "localhost"
MQTT_PORT = 1883
MQTT_TOPIC = "sensor/jsondata"

# Функція для відправки даних в топік MQTT
def publish_to_mqtt(data):
    client = mqtt.Client()
    client.connect(MQTT_BROKER, MQTT_PORT, 60)
    client.publish(MQTT_TOPIC, json.dumps(data))
    client.disconnect()

# Маршрут для збереження JSON
@app.route('/sensor', methods=['POST'])
def save_to_db():
    try:
        data = request.json
        key = data.get('key')

        # Перевірка ключа
        if key != SECRET_KEY:
            return jsonify({"error": "Unauthorized"}), 403

        pass_str = data.get('pass')
        pir = data.get('pir')
        lock = data.get('lock')
        timestamp = datetime.now()

        # Підготовка даних для публікації
        mqtt_data = {
            "pass": pass_str,
            "pir": pir,
            "lock": lock,
            "timestamp": timestamp.isoformat()
        }
```

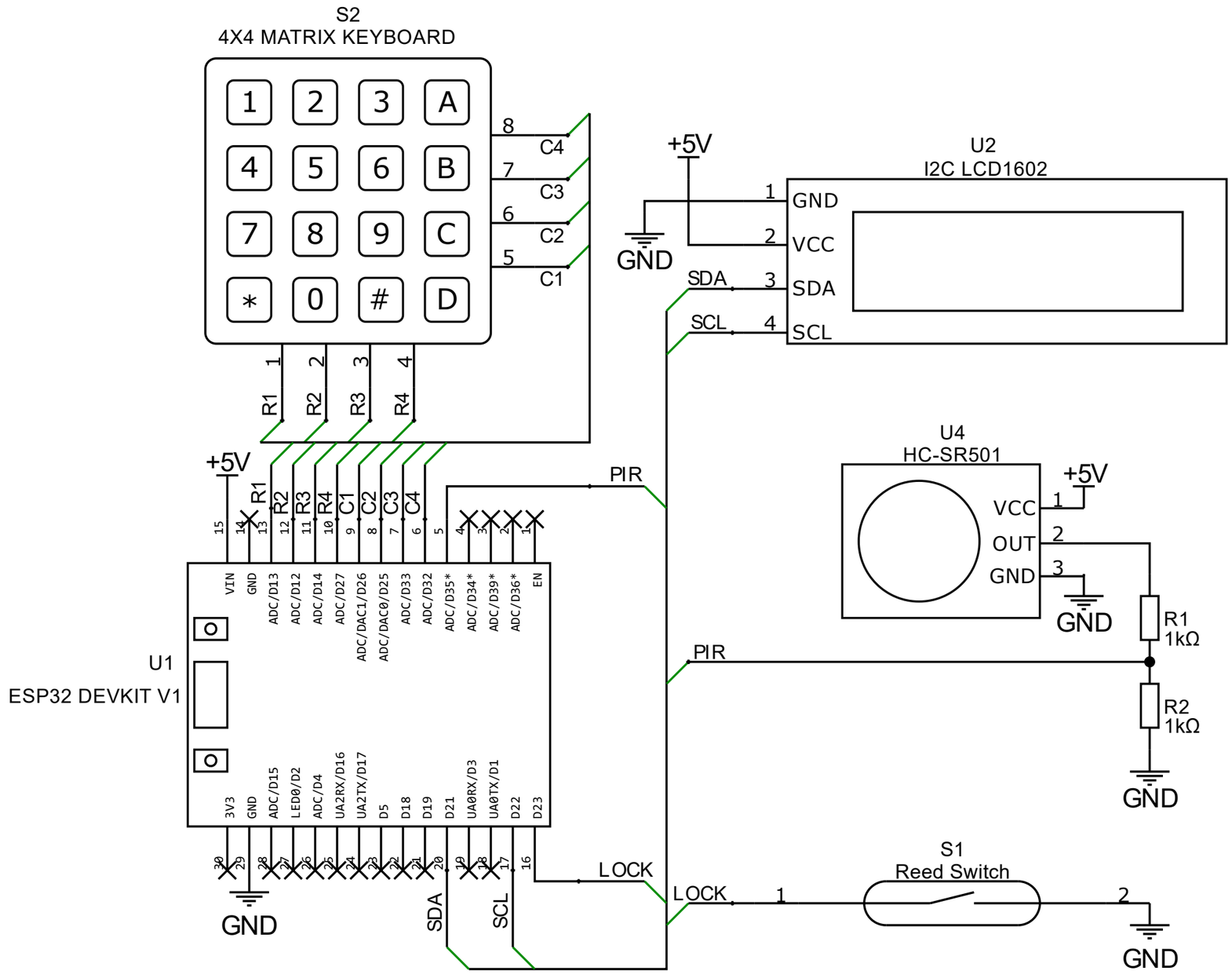
```
        # Публікація даних в MQTT
        publish_to_mqtt(mqtt_data)

        return jsonify({"message": "Data saved successfully"}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    # Виконання команд в консолі перед запуском Flask додатку
    try:
        subprocess.run(["docker-compose", "build"], check=True)
        subprocess.run(["./stop"], check=True)
        subprocess.run(["docker-compose", "up", "-d"], check=True)
    except subprocess.CalledProcessError as e:
        print(f"Error during subprocess execution: {e}")
        exit(1)

    app.run(host='0.0.0.0', port=1234)
```

ДОДАТОК Д

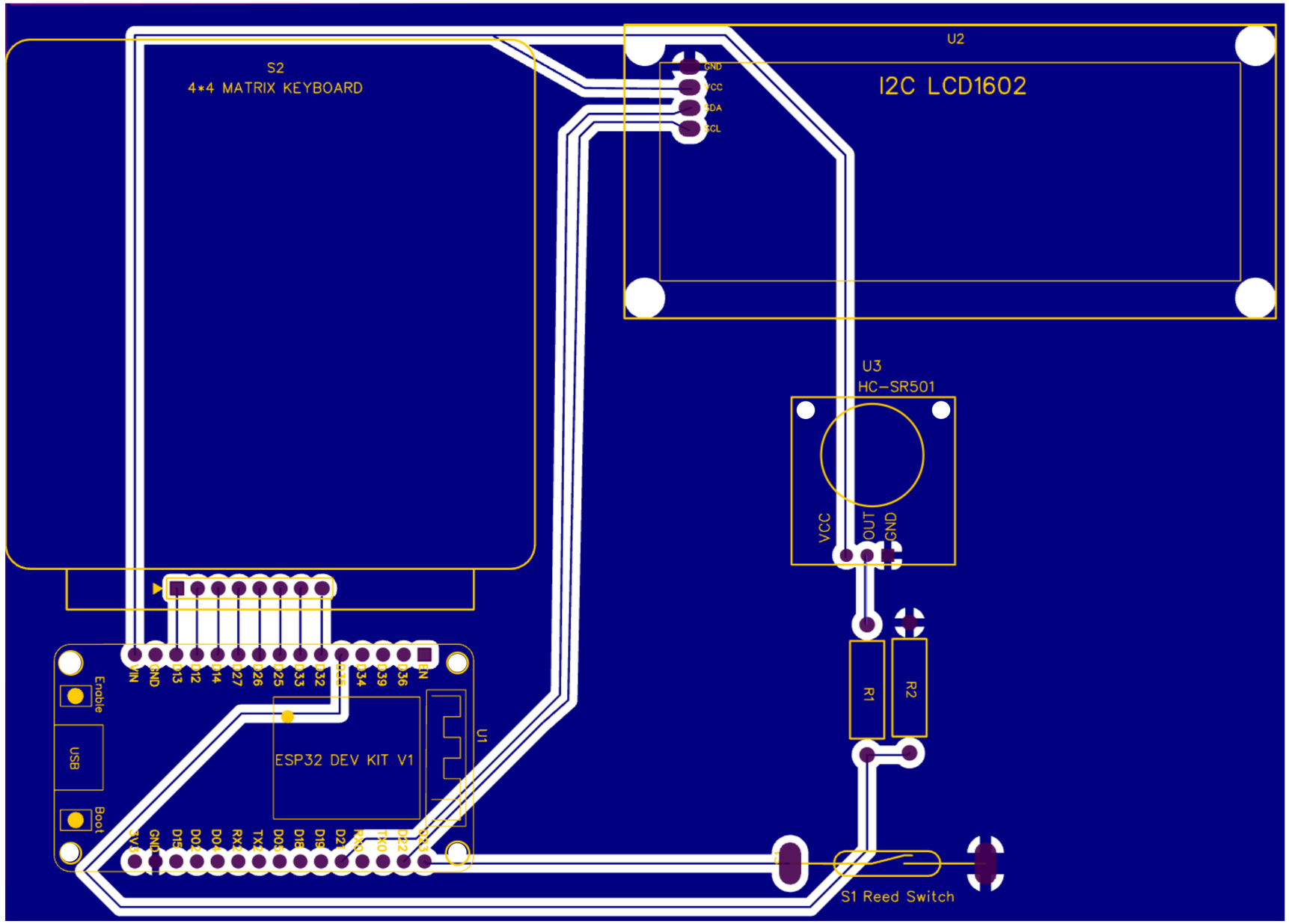


					КР ПМ-11.12.00.00 ЕС
Зм	Арк	N документа	Підпис	Дата	Програмований пристрій сигналізації Електрична схема
Розробив		Погорєлов Б. Ю			
Перевірив		Самборська В.В			
Т. контр.					
Н. контр.					
Затвердив		Киричук Ю. В.			

Літ	Маса	Масштаб
Аркуш	Аркушів	
КПІ, ПБФ, ПМ-11		

Первинний приклад | Спр. № | Підпис і дата | Інв № дубл. | Зам. інв. № | Підпис і дата | Інв. № орг.

ДОДАТОК Е



					КР ПМ-11.12.00.00 ДП		
					Програмований пристрій сигналізації Друкована плата	Літ	Маса / Масштаб
Зм	Арк	N документа	Підпис	Дата			2:1
Розробив		Погорєлов Б. Ю				Аркуш	Аркушів
Перевірив		Самборська В.В					
Т. контр.							
Н. контр.						КПІ, ПБФ, ПМ-11	
Затвердив		Киричук Ю. В.					

Первинний приклад | Спр. № | Підпис і дата | Інв № дубл. | Зам. Інв. № | Підпис і дата | Інв. № ор.