Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Система охорони приміщення
(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-8
Ванін Е. С.
(Прізвище, ініціали)

спеціальності *123 – Комп'ютерна інженерія*
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма *Комп'ютерна інженерія*
(повна назва освітньої програми)

Керівник ас. Корнієнко В. Р.

Допускається до захисту
Зав кафедри _____ Чумаченко С. В.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проєктування обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 Комп'ютерна інженерія

Тип програми Освітньо-професійна

Освітня програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ваніну Ельдару Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система охорони приміщення

затверджена наказом університету від 21 травня 2025 р. № 403 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 11 ____червня____ 2025 р.

3. Вихідні дані до роботи
мікроконтролер ESP32-CAM з підключеними датчиком руху,
датчиком диму,
GSM-модуль, камера,
мобільний пристрій (смартфон),
бездротова мережа Wi-Fi

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз та обґрунтування вибору компонентів системи,
розробка апаратної та програмної частини,
реалізація підключення мікроконтролера до мобільної мережі та взаємодія з нею,
розробка поведінкової та схеми підключення пристрою.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____
13 слайдів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Видача теми проекту, узгодження і затвердження теми	06.05.2025 – 09.05.2025	
2	Аналіз та вибір методу вирішення поставленої задачі поставленої задачі	10.05.2025 – 15.05.2025	
3	Ознайомлення з літературними джерелами аналіз та вибір методу вирішення поставленої задачі вибір системних засобів вирішення	16.05.2025 – 20.05.2025	
4	Проектування апаратної програмної частини	21.05.2025 – 25.05.2025	
5	Налагодження та тестування комп'ютерної системи	26.05.2025 – 31.05.2025	
6	Оформлення пояснювальної записки	01.06.2025 – 08.06.2025	
7	Перевірка виконаного проекту керівником, допуск до захисту	09.06.2025 – 11.06.2025	
8	Захист проекту	12.06.2025 – 24.06.2025	

Дата видачі завдання 06 травня 2025 р.

Здобувач  Ванін Е.С.
(підпис)

Керівник роботи _____ ас. Корнієнко В. Р
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи містить: 47 сторінок, 19 рисунків, 6 джерела за переліком посилань

СИСТЕМА ОХОРОНИ, ESP32, GSM, ДАТЧИК РУХУ, МІКРОКОНТРОЛЕР, КОНТРОЛЬНА ПАНЕЛЬ.

Метою кваліфікаційної роботи є огляд систем охорони, їх характеристик, складових частин, їх різновидів. Наведена інформація з історії розвитку сфери систем охорони. Розглянуті компоненти системи охорони. Детально описані плата розробника, мікроконтролер, сенсори, GSM-модуль.

ABSTRACT

The explanatory note contains 47 pages, 19 figures and 6 reference sources

SECURITY SYSTEM, ESP32, GSM, MOVEMENT DETECTOR, MICROCONTROLLER, CONTROL PANEL

The purpose of the qualification works is to review security systems, their characteristics, components, and types. The information provided includes a brief history of the development of security systems. The components of a security system are examined. The development board, microcontroller, sensors, and GSM module are described in detail.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

UART – Universal Asynchronous Receiver Transmitter – інтерфейс передачі даних;

IDE – Integrated Development Environment – інтегроване середовище розробки;

GSM – Global System for Mobile Communications – міжнародний стандарт для мобільного цифрового стільникового зв’язку;

SMS – Short Message Service – послуга обміну короткими текстовими повідомлення;

API – Application Programming Interface – інтерфейс програмування застосунків;

Бібліотека – набір готових функцій, класів, ресурсів для використання у коді;

Framework – фреймворк – інфраструктура програмних рішень, яка полегшує розробку складних систем;

Абстракція – принцип в програмуванні, який дозволяє приховати складні деталі реалізації та надати розробнику більш простий та зрозумілий інтерфейс для використання.

ВСТУП

Питання розробки охоронних систем є актуальним на поточний момент через необхідність забезпечення безпеки, доступності для більшої кількості споживачів та простоти користування і налаштування. Сучасні рішення дозволяють вирішити спектр проблем, але запропоновані на ринку рішення не є доступними для великої кількості споживачів.

Проблема розробки вбудованих систем для захисту приміщення є актуальною на поточний момент у зв’язку зі складністю сучасних рішень та збільшенням факторів небезпеки, які треба враховувати для видачі сигналів про небезпеку.

Першою розробкою в сфері охоронних систем є сигналізація, що була винайдена в 1853 році Августусом Поупом[3]. Для сповіщення про небезпеку цей пристрій бив у дзвін молотком, коли двері або вікно відкривались, використовуючи електромагнітні поля. Після Першої світової війни збільшився попит на охоронні системи. Люди, для яких ці системи були недоступні, наймали людей, які перевіряли, чи зачинені двері. Технологія відеоспостереження з’явилася в 1940 році, проте вона не була доступною для великої кількості людей. Системи відеоспостереження набули більшого поширення в 1980 році[3].

На даний момент однією з актуальних способів охорони є охоронні системи. Ці системи є найбільш універсальними в питаннях виявлення загроз та сповіщення про них. Вони можуть виявляти різноманітні загрози, такі як зловмисники, витоки газу, пожежі, затоплення та інші. Звісно, не всі охоронні системи можуть виявляти всі загрози одночасно, проте вони часто можуть підтримувати можливість заміни та доповнення модулів, які виявляють певні небезпечні чинники.

1 СИСТЕМИ ОХОРОНИ. СКЛАДОВІ ТА КЛАСИФІКАЦІЯ

1.1 Загальна інформація про системи охорони

Система охорони — комплекс засобів, які призначені для підвищення рівня безпеки об’єкта, що охороняється. До засобів, які можуть бути включені в систему охорони, належать замки, контрольна панель, датчики, відеокамери та інші пристрої.

Системи охорони набули широкого розповсюдження в різних повсякденних сферах: у супермаркетах, у метро, у кафе, у навчальних закладах і так далі. Саме вони забезпечують нам безпеку. Ці системи миттєво реагують на загрози та сповіщають про них, що дозволяє людям реагувати на них певним чином.

Вони набули такого розповсюдження, що можуть використовуватися для домашнього користування. Користувачі цих систем можуть в реальному часі отримувати зображення з камер відеоспостереження і отримувати сповіщення у випадку, якщо майну цього користувача загрожує небезпека.

Завдяки розвитку галузі систем охорони, ці системи користувач може встановити та налаштувати сам, без допомоги спеціалістів. Компоненти цих систем можна легко прибрати, якщо в них немає необхідності, або їх можна додати.

Дані системи можуть виявляти зловмисників, пожежі, витоки газу, підтоплення, розбиття вікон, злом замка та відправляти сповіщення про це користувачеві.

Також існують охоронні фірми, які встановлюють охоронні системи. У випадку, коли охоронна система або сам користувач виявляє зловмисника, дані про це передаються охоронній фірмі, і вона реагує на це.

1.2 Класифікація систем охорони

За підключенням системи охорони класифікують на дротові та бездротові[2].

В дротових системах всі компоненти, підключені за допомогою дротів. Дані системи є стабільними та надійними, в них не може виникнути збій при передачі сигналу, якщо дроти, що використовувались при передачі даних, не піддавались механічному впливу.

Ці системи потребують професійного встановлення, оскільки дроти цієї системи повинні проходити через усе приміщення. Необхідність прокладки дротів також є причиною, чому цей тип систем охорони складно модифікувати.

Бездротові системи використовують бездротові засоби передачі даних. Це рішення є більш зручним для середньостатистичного користувача, проте має свої недоліки.

Бездротові охоронні системи є вразливими до перешкод. Проте покращені протоколи шифрування забезпечують стабільний зв’язок між компонентами системи.

Бездротові системи є набагато простішим варіантом для встановлення користувачем, оскільки ця система не потребує прокладки дротів по всьому приміщенню. Бездротові системи легко піддаються модифікаціям. Через відсутність дротів компоненти системи можна легко встановити в інше місце в приміщенні або додати ще компоненти при виникненні такої необхідності.

Особливістю цих систем віддалене є керування. За допомогою мобільного або веб-застосунку можна отримувати сповіщення про загрози, переглядати зображення з камер спостереження в режимі реального часу, ввімкнути або вимкнути охорону[2].

Компоненти бездротових систем охорони використовують автономні джерела живлення, тому виникає необхідність заміна цих джерел після деякого часу користування цією системою.

1.3 Складові частини

Системи охорони можуть складатися з багатьох компонентів, проте вони обов’язково мають три наступні компоненти[6].

1. Контрольна панель (рис. 1.1). Це пристрій, до якого під’єднуються інші пристрої, і який контролює їх. Нею можна керувати за допомогою графічного інтерфейсу користувача, якщо вона має його, або віддалено за допомогою мобільного застосунку або веб-застосунку[6]. До неї також підключають датчики та пристрої сповіщення. У випадку отримання від датчиків сигналів про виявлення певної загрози контрольна панель відправить користувачу сповіщення про загрозу та сформує сигнал на пристрої сповіщення, щоб вони почали свою роботу.

Рисунок 1.1 – Контрольна панель.

2. Датчики. Це пристрої, які визначають присутність певних явищ та сповіщають про них за допомогою електронного сигналу. В охоронних системах саме вони виявляють загрози. Вони можуть виявляти пожежі, підтоплення, злом дверей, розбиття вікон, рух, витоки газу, шум, тощо. У

випадку, якщо вони виявили певну загрозу, вони передають сигнал контрольній панелі[6]. Нижче наведено зображення датчика руху (рис. 1.2).

Рисунок 1.2 – Датчик руху

3. Пристрої сповіщення. Вони мають на меті сповіщати про загрози. У випадку загрози, яку створює не людина, вони сповіщають про загрозу. Якщо загрозу створює людина, то вони сповіщають про загрозу та мають певний психологічний ефект на людину, яка створює загрозу. Нижче наведено зображення пристрою сповіщення (рис. 1.3)

Рисунок 1.3 – Пристрій світлозвукового оповіщення «Сигнал-М»

Охоронні системи можуть мати інші компоненти:

1. Камери відеоспостереження (рис. 1.4). Вони транслюють в режимі реального часу зображення частини приміщення. Деякі системи мають можливість робити запис з цих камер для подальшого перегляду.

Рисунок 1.4 – Камера відеоспостереження

2. Тривожна кнопка (рис. 2.5). Пристрій, який при натисканні сповіщає про тривогу. Зазвичай не використовується в домашніх системах охорони, проте має широке застосування в інших сферах.

Рисунок 1.5 – Тривожна кнопка

3. Розумні замки (рис. 2.6). Пристрій, який виконують функції замка та керується за допомогою певних електронних пристроїв[6]. Цими

пристроями можуть бути ключ-карта, смартфон, спеціальний електронний ключ.

Рисунок 1.6 – Розумний дверний замок

2 ТЕХНОЛОГІЧНА ПЛАТФОРМА РЕАЛІЗАЦІЇ

2.1 Технічні характеристики компонентів системи охорони приміщення

Для розробки пристрою системи охорони приміщення було використано:

- плата розробки ESP32-CAM;
- інтерфейс UART;
- GSM модуль на SIM800L;
- інфрачервоний датчик руху HC-SR501;
- датчик газу та диму MQ-2;
- зарядний модуль TP4056;
- перетворювач напруги XL6019E1.

2.2 Плата розробки ESP32-CAM

ESP32-CAM є платою розробки з чипом ESP32-S. Дана плата розробки підтримує бездротові протоколи передачі даних WiFi, Bluetooth та BLE, а також використовує два високопродуктивних 32-бітних процесора LX6. Також вона підтримує читання та запис даних з SD-картки та має цифрову камеру OV2640.

Для реалізації цього проєкту було вирішено використовувати плату розробки ESP32-CAM (рис. 2.1).

Рисунок 2.1 – Плата розробки ESP32-CAM з камерою OV2640

Дана плата розробки має 9 GPIO портів, підтримує інтерфейс UART та має 7 пінів, які підтримують приймання аналогового сигналу. ESP32-CAM не має власного програматора, тому для завантаження прошивки на цій платі потрібна спеціальна плата ESP32-CAM-MB (рис. 2.2), інший мікроконтролер, що підтримує UART, або адаптер USB-UART (рис. 2.3).

Рисунок 3.2 – ESP32-CAM-MB.

ESP32-CAM-MB – це плата, яка застосовується для завантаження прошивки та налагодження плати розробки ESP32-CAM-MB. Ця плата спрощує завантаження прошивки на ESP32-CAM, оскільки дозволяє простим натисканням кнопок перезавантажити плату в режимі завантаження прошивки та перезавантажити плату, коли прошивка була завантажена. Це особливо корисно для початківців.

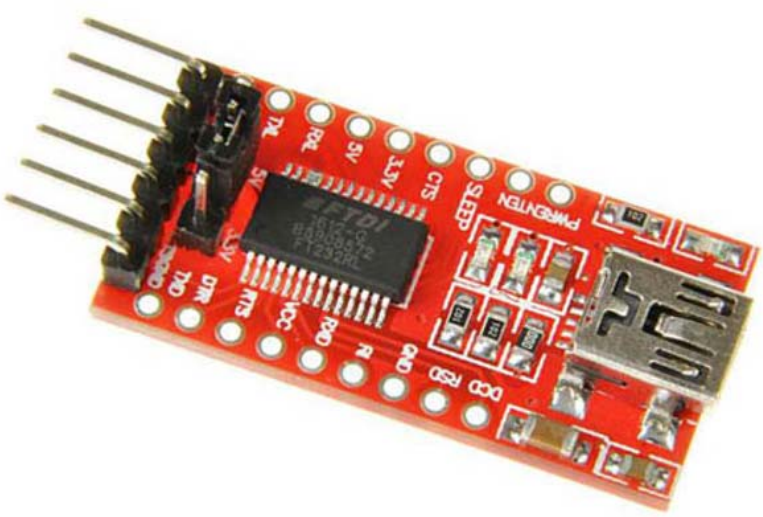

Рисунок 2.3 – Адаптер USB-UART

Адаптер USB-UART – це плата, яка перетворює сигнали USB в сигнали UART, що використовується для налагоджування та завантаження прошивки на мікроконтролери, в тому числі і ESP32-CAM. Це більш універсальний варіант для завантаження прошивки. Проте процес завантаження прошивки за допомогою адаптера USB-UART на ESP32-CAM є більш складним, ніж при використанні ESP32-CAM-MB.

2.3 Інтерфейс UART

UART – асинхронний послідовний інтерфейс для комунікації між цифровими пристроями[5]. Цей інтерфейс застосовується для передачі даних між пристроями, завантаження прошивки та налагодження. Дані в цьому

інтерфейсі можуть відправлятися паралельно за рахунок двох незалежних ліній передачі даних. Проте цей інтерфейс підтримує зв'язок тільки між двома цифровими пристроями.

Пристрої, що підтримують UART мають виходи Rx та Tx. Вихід Rx відповідає за отримання даних, а вихід Tx – за передачу. Схему підключення двох пристроїв, що підтримують UART зображено на рис. 3.x

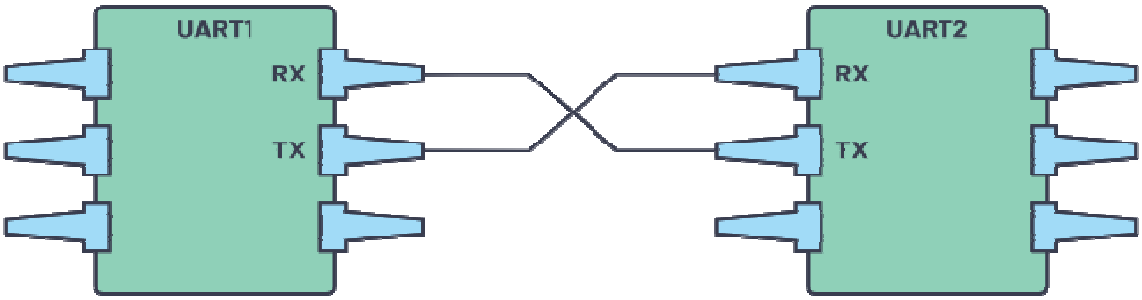

Рисунок 2.4 – Схема підключення двох пристроїв,
що підтримують UART

UART передає дані в вигляді пакетів. Кожен пакет обов'язково складається зі стартового біта, бітів даних, та стоп-бітів. Також UART може мати біт парності. Пакет може мати розмір від 7 до 13 бітів.

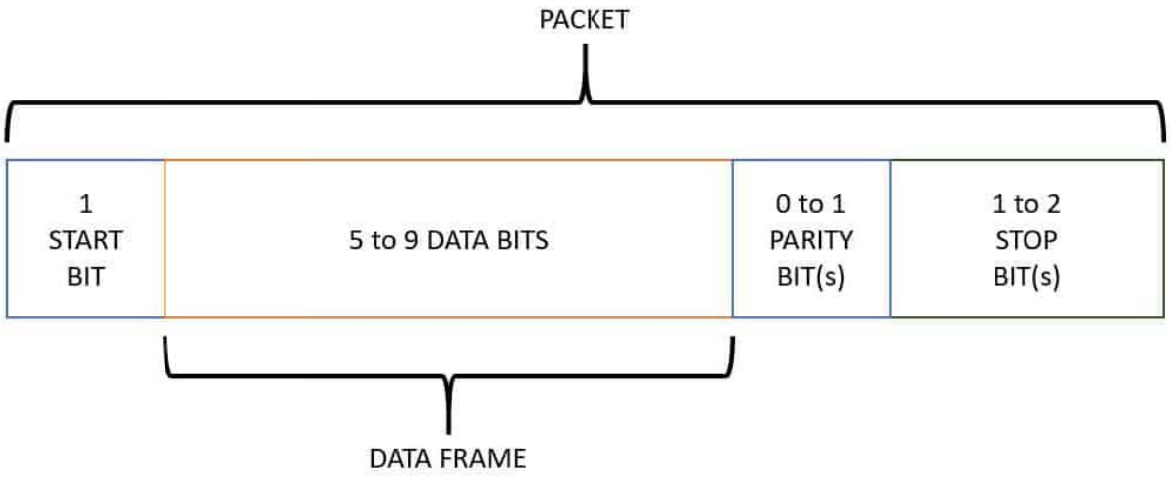

Рисунок 2.5 – Структура пакету UART

Стартовий біт відповідає за початок передачі даних. Лінії UART зазвичай під’єднані до високого рівня напруги, коли дані не передаються. Для початку відправки даних пристрій UART, який відправляє дані, підтягує лінію передачі даних до низького рівня на один тактовий цикл.

Біти даних складаються з даних, що передаються. Вони мають від 5 до 9 бітів, якщо не використовується біт парності, або від 5 до 8 бітів, якщо використовується біт парності. Зазвичай використовується порядок бітів де спершу йде наймолодший біт.

Біт парності вказує на непарну кількість встановлених бітів даних. Якщо встановлених бітів парна кількість, то біт парності буде скинутий; якщо встановлених бітів непарна кількість, то біт парності буде встановлений. Цей біт використовується для виявлення одиночних помилок при передачі даних, оскільки є ймовірність пошкодження даних при їх передачі під впливом зовнішніх чинників.

Біти зупинки сигналізують про завершення передачі пакета даних. Кількість бітів зупинки може бути від одного до двох. Біти зупинки встановлені на високий рівень[5].

Для коректної передачі даних між пристроями, які використовують UART, швидкість передачі даних та структура пакетів мають співпадати.

2.4 GSM модуль на SIM800L

GSM модуль підтримує стільниковий зв'язок, або мобільний зв'язок. Був використаний саме цей вид бездротового зв’язку через широке покриття, стабільний зв'язок у більшості регіонів світу, мобільність та підтримку роумінгу.

Стільниковий зв'язок забезпечує не лише голосові дзвінки, але й передачу текстових повідомлень (SMS), мультимедійних повідомлень (MMS), доступ до інтернету та різноманітні онлайн-сервіси. Підтримка

технологій 3G, 4G та 5G дозволяє використовувати високошвидкісний інтернет для різних потреб.

Крім того, сучасні стандарти шифрування забезпечують високий рівень безпеки передачі даних.

Не менш важливим фактором є доступність для широкого кола користувачів та економність. Велика кількість операторів та тарифних планів дозволяє користувачам обирати найбільш вигідні та зручні умови.

Завдяки цим перевагам стільниковий зв'язок є важливою складовою сучасної комунікаційної інфраструктури, забезпечуючи зв'язок, доступ до інформації та безліч інших можливостей для людей по всьому світу.

GSM модуль SIM800L - це компактний та економічний модуль для здійснення бездротового зв'язку в мережах GSM/GPRS. Цей модуль дозволяє додати можливість передачі даних та SMS-повідомлень до мікроконтролерних систем, таких як ESP32.

Цей модуль працює в діапазонах GSM 850/900/1800/1900 МГц, що дає можливість модулю SIM800L працювати практично в будь-якій країні світу, підтримує багатоканальне GPRS з’єднання з класом 12, забезпечуючи високу швидкість передачі даних (до 85,6 кбіт/с), має можливість приймати та здійснювати голосові дзвінки, SMS-повідомлення, та підключатись до інтернету за допомогою GPRS. Керування модулем здійснюється за допомогою набору стандартних AT-команд, що забезпечує користувачу можливість легкого налаштування та інтеграції цього модуля.

Модуль SIM800L підтримує протокол UART, що дає можливість використання цього модуля з мікроконтролерами, може працювати при напрузі від 3.4В до 4.4В, має можливість підключення зовнішньої антени для покращення якості сигналу.

Рисунок 2.6 – GSM модуль на SIM800L

Використання модуля SIM800L у системах охорони дає можливість сповіщати про виявлені загрози за допомогою SMS-повідомлень або дзвінків. Це є дуже корисним для миттєвого реагування на потенційні загрози.

При ввімкненні модуля SIM800L світлодіод, що знаходиться на платі, починає блимати раз на секунду. При встановленні зв’язку з мобільним оператором світлодіод мерехтить раз на три секунди. При втраті зв’язку світлодіод починає блимати раз на секунду. Також для налагодження використовується UART. Проте, оскільки цей UART цього модуля працює при напрузі до 2.8 В, використання адаптера USB-UART для налагодження не є можливим.

2.5 Інфрачервоний датчик руху HC-SR501

Інфрачервоні датчики руху (PIR-датчики) використовуються для виявлення руху в системах охорони, автоматизації освітлення, автоматичних дверях та розумних будинках. Вони працюють, виявляючи інфрачервоне випромінювання, яке випромінюють теплові об'єкти, такі як люди чи тварини.

Датчик містить піроелектричні сенсори, що реагують на зміни рівня цього випромінювання. Коли тепловий об'єкт рухається, змінюється рівень випромінювання, який реєструється сенсором. Вбудована електроніка аналізує ці зміни, і якщо вони виявляються результатом руху, датчик генерує сигнал для активації інших пристроїв або систем(рис. 2.7).

Наприклад, якщо датчик встановлено у порожній кімнаті, кожен чутливий елемент отримує постійну дозу випромінювання, а отже, і напруга на них має постійне значення.

Як тільки в кімнату заходить людина, вона потрапляє спочатку до зони реагування першого елемента, що призводить до появи позитивного електричного імпульсу на ньому.

Людина рухається, і її теплове випромінювання через лінзи потрапляє вже другий PIR-елемент, який генерує негативний імпульс. Електронна схема датчика руху реєструє ці різноспрямовані імпульси та робить висновки про те, що в поле зору датчика потрапила людина. На виході датчика генерується позитивний імпульс (рис. 2.7).

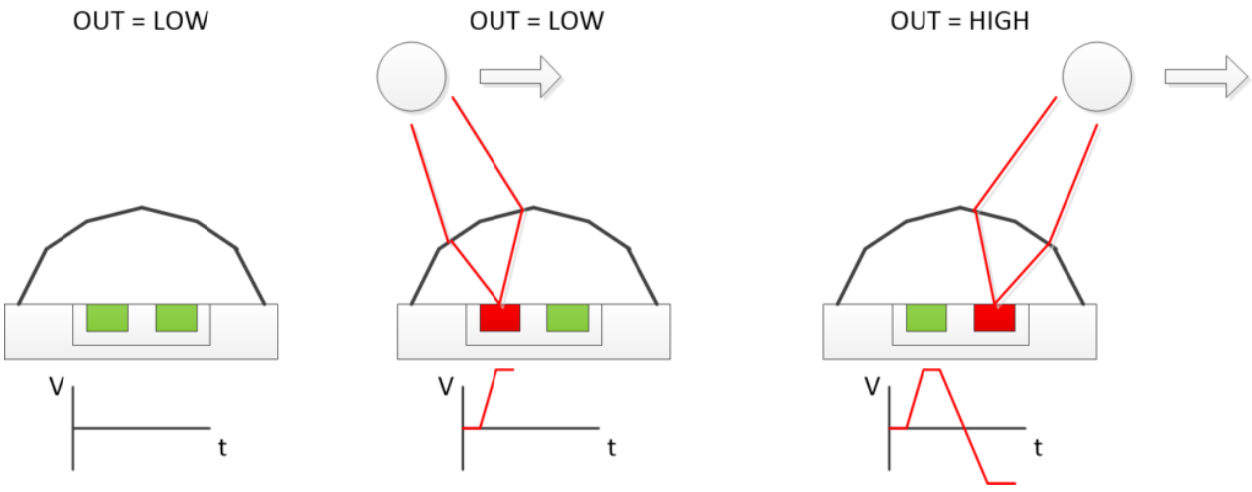

Рисунок 2.7 – Принцип роботи PIR датчиків

Інфрачервоний датчик руху HC-SR501 є популярним та доступним рішенням для виявлення руху в системах охорони та автоматизації. Цей датчик здатен виявляти рух у радіусі до 7 метрів і має кут огляду близько 120 градусів.

Датчик оснащений регульованими параметрами затримки часу та чутливості, що дозволяє налаштувати його під конкретні потреби.

HC-SR501 використовується у різних додатках, включаючи системи охорони, автоматичне освітлення, "розумні" будинки та інші проекти з мікроконтролерами.

Характеристики обраного датчика:

– дальність виявлення: 0 - 7 м;

– кут спрацьовування: 110 ° на дистанції до 7 м;

– напруга живлення (рекомендована): 4.5 - 12 В;

– вихідна напруга логічного рівня: 0 - 3.3 В;

– час затримки: 0.3 – 300 секунд (регулюється);

– метод спрацьовування: L неповторне перемикання; H повторюване перемикання;

– максимальний вихідний струм: 65 мА;

– робочі температури: -20 – +50 град. Цельсія;

– розміри: 32x24 мм.

2.6 Датчик газу та диму MQ2

Датчик MQ2 (рис. 2.8) призначений для виявлення хімічних сполук в повітрі. Цей датчик може виявляти водень, зріджений нафтовий газ, метан, чадний газ, пари алкоголю, пропан та дим. Він має високу чутливість та швидкий час відгуку, що дозволяє швидко вимірювати концентрацію газів в повітрі.

Датчик газу MQ2 має чотири виходи: VCC, GND, A0, D0.

Входи VCC та GND відповідають за живлення. Даний датчик працює при напрузі 5В. Вихід A0 є аналоговим виходом, що змінює своє значення в залежності від концентрації в повітрі газів, які датчик здатний виявляти. Вихід D0 є дискретним виходом, який скидається в випадку, коли виявлена концентрація газів перевищує певний рівень, який встановлюється за допомогою потенціометра.

Рисунок 2.8 – Датчик газу MQ2.

При ввімкненні датчик починає розігрівати напівпровідниковий шар діоксиду олова (SnO_2) до температури, при якій молекули кисню з повітря адсорбуються на поверхні діоксиду олова. Ці адсорбовані молекули кисню забирають вільні електрони з шару діоксиду олова, створюючи зону виснаження електронів на поверхні діоксиду олова, що збільшує електричний опір шару діоксиду олова[4].

Коли в цей датчик входить горючий газ, він реагує з адсорбованими молекулами кисню, вивільняючи захоплені ними електрони, при цьому зменшуючи зону виснаження електронів, що зменшує опір напівпровідника.

2.7 Модель індикації руху у приміщені

Модель системи охорони призначена для виявлення руху та надсилання повідомлень або викликів у разі спрацювання датчика.

На рис. 2.9 зображено схему роботи моделі системи охорони.

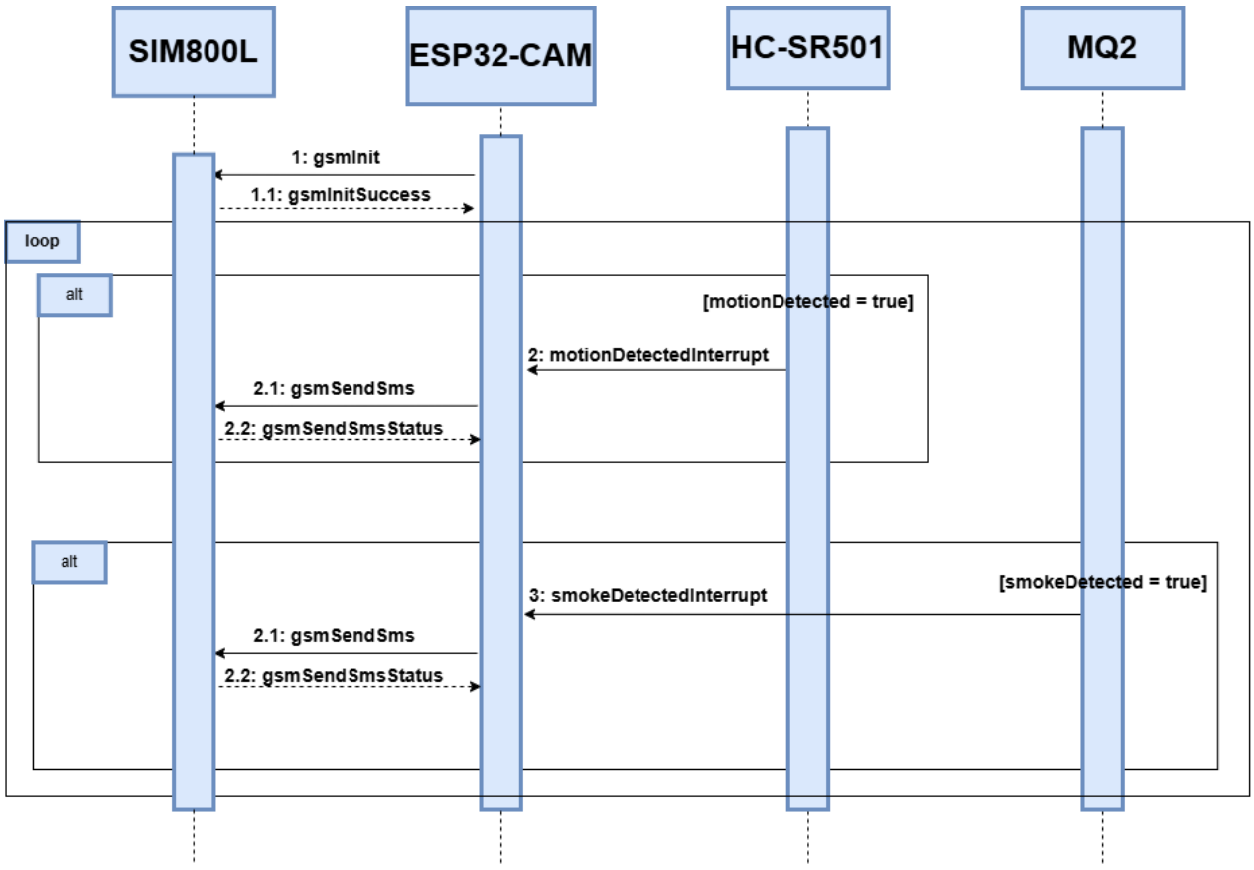

Рисунок 2.9 – Схема роботи моделі системи охорони.

Робота системи охорони відбувається наступним чином:

1: gsmInit – ESP32-CAM ініціалізує SIM800L та надсилає AT-команди для входу в мережу.

1.1: gsmInitSuccess – SIM800L відправляє на ESP32 відповідь зі статусом ініціалізації.

2: motionDetectedInterrupt – датчик руху HC-SR501 посилає сигнал при виявленні руху та активує переривання на ESP32-CAM.

2.1: gsmSendSms – ESP32-CAM формує список AT-команд для SIM800-L для відправки повідомлення про виявлення руху на певний номер телефону.

2.2: gsmSendSmsStatus -SIM800L відправляє на ESP32-CAM відповідь зі статусом відправлення повідомлення.

3: smokeDetectedInterrupt – датчик MQ2 надсилає сигнал при виявленні диму та активує переривання на ESP32-CAM.

3.1: gsmSendSms – ESP32-CAM формує список AT-команд для SIM800-L для відправки повідомлення про виявлення диму на певний номер телефону.

3.2: gsmSendSmsStatus – SIM800L відправляє на ESP32-CAM відповідь зі статусом відправлення повідомлення.

Схема реалізації системи індикації руху без використання бездротового з’єднання представлена на рис. 2.10.

Ця модель системи охорони може бути використана у житлових будинках, офісах, складах та інших об'єктах, де необхідний контроль за рухом та оперативне сповіщення власника про потенційні загрози. Завдяки використанню GSM модуля, система забезпечує віддалений моніторинг і управління, що підвищує її ефективність та зручність використання.

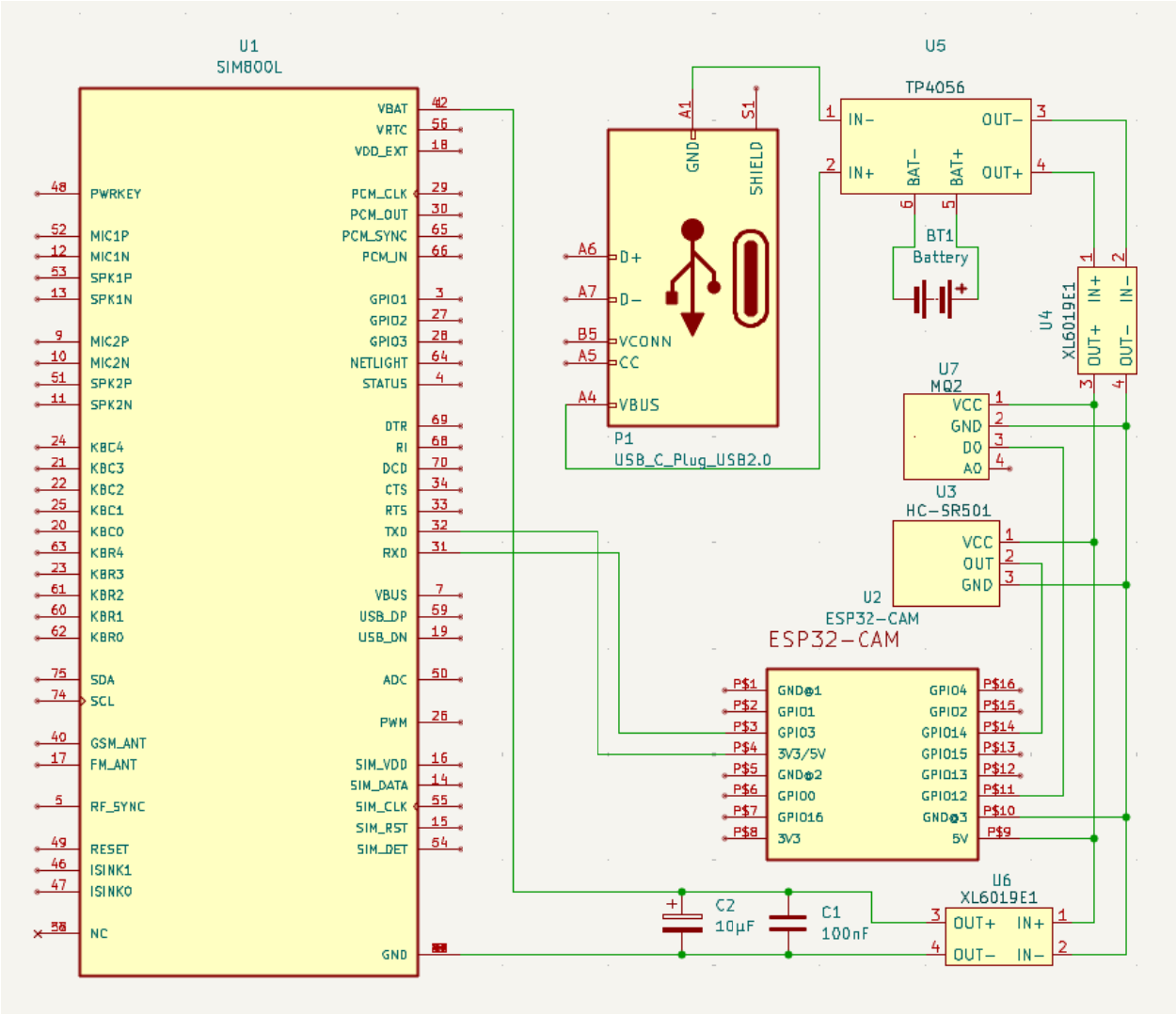

Рисунок 2.10 – Схема реалізації індикації руху та передачі показників на мобільний пристрій

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1 Вибір засобів розробки для плати розробки ESP32-CAM

Для розробки програмного забезпечення для плати розробки ESP32-CAM було розглянуто наступні засоби розробки програмного забезпечення:

- Arduino IDE;
- ESP-IDF;
- PlatformIO.

3.1.1 Arduino IDE

Arduino IDE – офіційне середовище розробки для плат Arduino, що підтримує розробку програмного забезпечення для інших плат. Воно має простий та зрозумілий інтерфейс, що дає можливість редагувати код, завантажувати прошивку на плату та налагоджувати її[1]. Також це середовище розробки має широкий вибір бібліотек, які створює та підтримує спільнота. Ці бібліотеки спрощують розробку коду або розширюють можливості для розробки коду.

Перевагами цього середовища розробки є інтуїтивно зрозумілий інтерфейс, що дозволяє користуватись функціями програми без зайвих проблем, та велика спільнота, що створює бібліотеки для різних модулів та допомагає вирішити проблеми, які можуть виникати під час користування програмою.

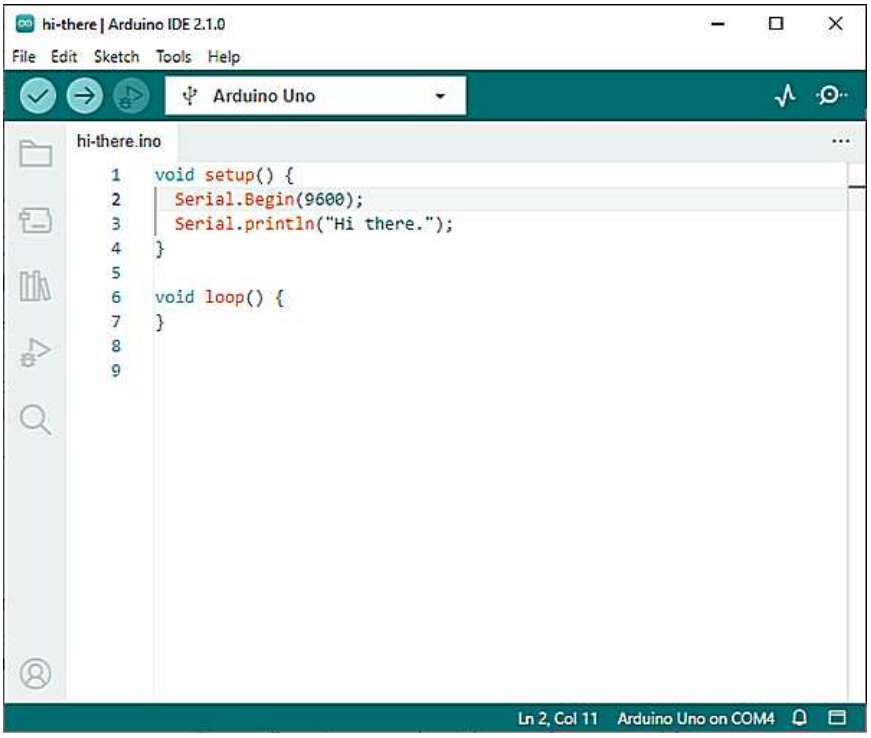

Рисунок 3.1 – Інтерфейс Arduino IDE

Корисною функцією цієї програми є Serial Monitor. Вона дозволяє в режимі реального часу отримувати та передавати дані, використовуючи UART. Це корисно при налагодженні програми, оскільки мікроконтролер може передавати повідомлення про помилку.

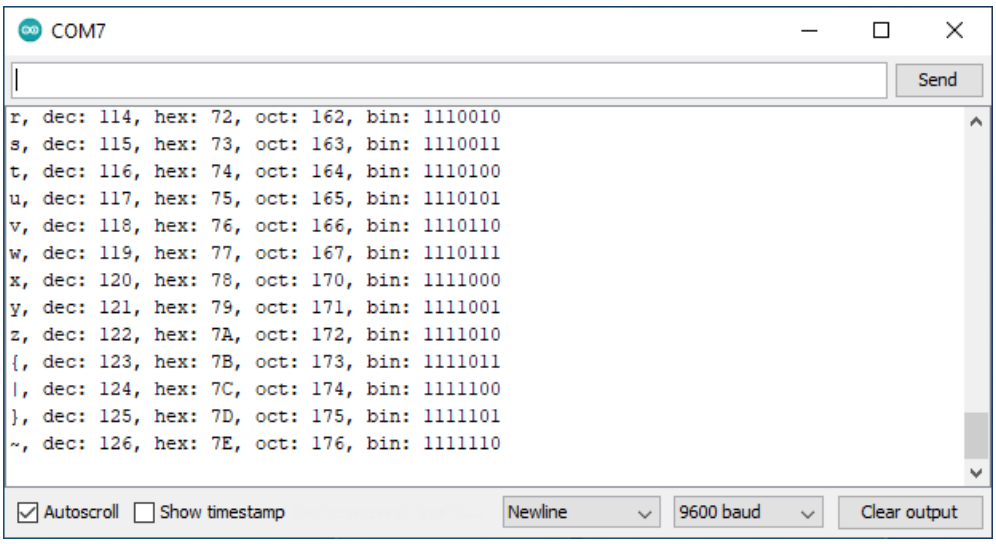

Рисунок 3.2 – Serial Monitor в Arduino IDE

Arduino IDE використовує фреймворк Arduino, що створений для полегшення розробки проєктів на базі мікроконтролерів. Він стандартизує структуру програм, спрощує доступ до апаратних ресурсів плати, для якої розробляється програмне забезпечення. Цей фреймворк реалізований як набір бібліотек C/C++ з чітко визначеним API.

Перевагами використання Arduino IDE є підтримка різноманітних плат розробника на різних мікроконтролерах, простота для початківців за рахунок малої кількості налаштувань та приховування від користувача деталей роботи на нижчому рівні, велика кількість бібліотек та велика спільнота. Проте, Arduino IDE має обмежений функціонал текстового редактора, відсутні функції автодоповнення, перейменування або пошуку по проєкту, повільна компіляція коду, приховування від користувача деталей роботи на нижчому рівні та слабка підтримка зовнішніх інструментів, таких як Git, Valgrind.

3.1.2 ESP-IDF

ESP-IDF – офіційний фреймворк від компанії Espressif Systems для розробки програмного забезпечення для мікроконтролерів ESP32. Він дозволяє розробляти програмне забезпечення для мікроконтролерів ESP32, включаючи ESP32-CAM, на нижчому рівні абстракції в порівнянні з Arduino IDE.

ESP-IDF дозволяє користувачу мати повний контроль над периферією мікроконтролера та доступ до апаратних можливостей мікроконтролера. Але на відміну від Arduino IDE тут відсутній графічний інтерфейс, тому всі функції ESP-IDF доступні лише при використанні командного рядка.

Перевагами ESP-IDF є максимальний контроль над апаратною частиною мікроконтролера, що дозволяє розробляти більш швидкі рішення в порівнянні з проєктами Arduino, модульність проєктів і багатозадачність.

Проте коду написаного на ESP-IDF зазвичай більше, ніж в аналогічних проєктах написаних на Arduino Framework. Також освоєння цього фреймворка є більш складним.

Отже, це потужний інструмент, що призначений для розробників, яким потрібні висока стабільність проєкту, максимальна швидкодія та максимальний контроль за ресурсами мікроконтролера.

3.1.3 PlatformIO

PlatformIO – екосистема розроблена українською компанією PlatformIO Labs для розробки програмного забезпечення для вбудованих систем. Ця екосистема має більший функціонал ніж Arduino IDE. Вона дозволяє розробляти програмне забезпечення для більш ніж 200 платформ, включаючи Arduino, ESP32, STM32, керувати бібліотеками, збіркою, прошивкою та автоматизованими тестуваннями, та підтримку зовнішніх інструментів.

PlatformIO є більш просунутим в питаннях налагодження програми в порівнянні з Arduino IDE. PlatformIO має підтримку багатьох пристроїв налагодження, таких як ST-Link V2, J-Link, CMSIS-DAP та інших. Також, PlatformIO має Serial Monitor, що дає можливість отримувати та передавати дані з UART мікроконтролера.

Також існує PlatformIO IDE – графічна оболонка, що інтегрує PlatformIO в текстові редактори, наприклад, Visual Studio Code. Це середовище розробки забезпечує користувачу графічний інтерфейс до функцій PlatformIO, автодоповнення тексту, перейменування, пошук по проєкту, дебагу та керування бібліотеками, платформами та платами без використання командного рядка.

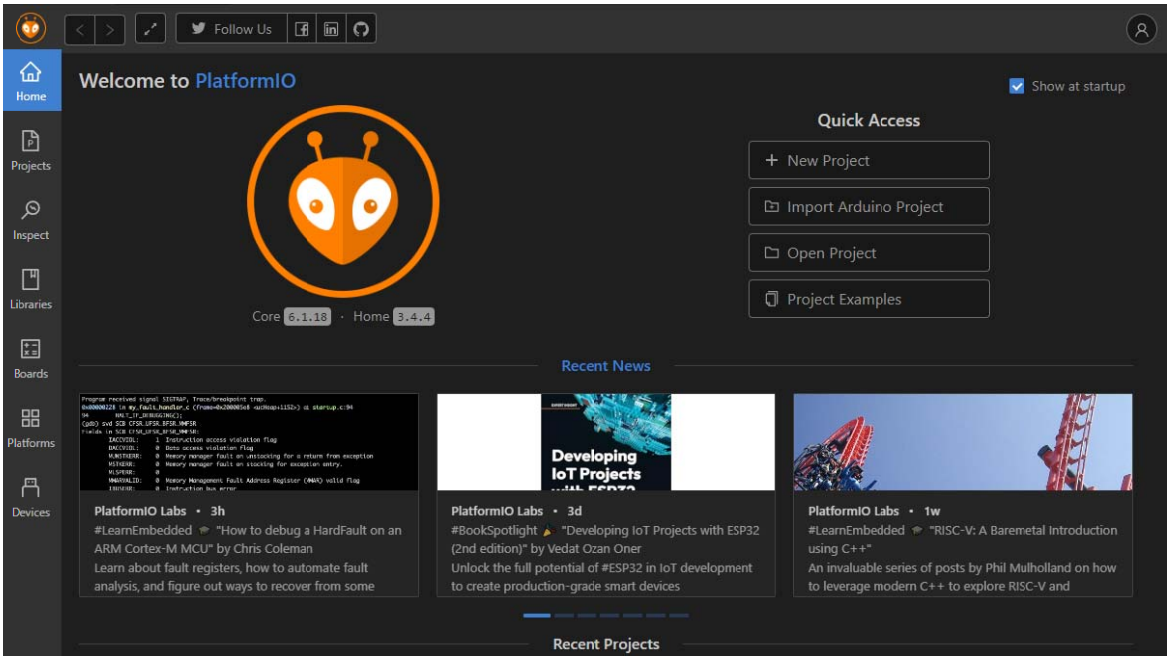

Рисунок 3.3 – Інтерфейс PlatformIO IDE.

Перевагами PlatformIO є підтримка багатьох платформ, присутність IntelliSense, що підсвічує символи, дає можливість автодоповнення, перехід по символах та рефакторинг, широкий асортимент інструментів для налагодження, можливість керувати бібліотеками та залежностями, кросплатформність та можливість інтегрувати зовнішні інструменти. Також проєкти PlatformIO мають повноцінну структуру. Недоліками PlatformIO є його більша складність в порівнянні з іншими програмними засобами, може повільно працювати на старій техніці, можуть виникати помилки, повідомлення про які можуть бути неочевидними, та відсутність GUI для деяких налаштувань.

3.1.4 Обгрунтування вибору засобу розробки програмного забезпечення

Для розробки програмного забезпечення для системи охорони приміщення було вирішено використовувати PlatformIO.

По-перше, це середовище розробки сумісне з текстовим редактором Visual Studio Code, який є текстовим редактором, що має велику кількість розширень, які дозволяють з легкістю налаштувати програмне забезпечення під себе.

По-друге, PlatformIO має підтримку систем контролю версій Git, що є незамінним інструментом при розробці будь-якого програмного забезпечення.

По-третє, PlatformIO має більше інструментів для налагодження в порівнянні з іншими, що дозволяє з більшою простотою виявляти причини виникнення проблем у програмному забезпеченні, що розробляється.

3.2 Налаштування проєкту PlatformIO

Для створення проєкту PlatformIO треба вибрати налаштування проєкту. У налаштуваннях була обрана плата AI Thinker ESP32-CAM та фреймворк Arduino.

Також є налаштування проєкту, які зберігаються у файлі platformio.ini. В цьому файлі було вибрано наступні налаштування:

- швидкість UART-монітора: 115200;
- зовнішня PSRAM: ввімкнено;
- підключені бібліотеки: TinyGSM.

Лістинг 4.1 – Налаштування файлу platformio.ini

```
[env:esp32cam]
platform = espressif32
board = esp32cam
framework = arduino
monitor_speed = 115200
board_build.psram = enabled
lib_deps =
    TinyGSM
```

3.3 Розробка програмного коду

Код для мікроконтролера ESP32-CAM написаний мовою C++ з використанням фреймворку Arduino. Код складається з модулів, які відповідають за певний функціонал прошивки мікроконтролера.

3.3.1 Розробка коду для інтеграції датчиків диму та руху.

Перший модуль що відповідає за датчик газу та датчик руху. Оскільки ці датчики працюють таким чином, що вони формують сигнал при спрацюванні, то було вирішено об’єднати їх під одним модулем. Є декілька варіантів, як реалізувати обробку сигналів від цих датчиків: опитування портів мікроконтролера та виконання певних дій, якщо було зафіксовано певний стан, або прив’язування портів до переривань. Оскільки пріоритетом в системі охорони є швидке реагування на загрози, було прийнято рішення прив’язати порти до переривань.

Основною функцією модуля є attach_interrupts(). Вона визначає порт MOTION_SENSOR_PIN як вхідний порт з резистором, що підтягує до землі, оскільки в спокійному стані на виходах датчика руху низький логічний рівень, який встановлюється у високий при виявленні руху. Порт SMOKE_SENSOR_PIN визначається як вхідний порт з резистором, що підтягує догори, оскільки в спокійному стані датчик диму має на виходах високий логічний рівень та встановлюється у низький при виявленні диму. Далі функції attachInterrupt() прив’язують порти MOTION_SENSOR_PIN та SMOKE_SENSOR_PIN до обробників переривань motionDetectedInterrupt та smokeDetectedInterrupt відповідно. Оскільки датчик руху при спрацюванні збільшує рівень напруги на виходах, обробник переривань спрацьовує при переході зі стану логічного нуля в стан логічної одиниці, а обробник переривань датчику диму спрацьовує при переході зі стану логічної одиниці в стан логічного нуля, бо датчик диму при спрацюванні зменшує рівень напруги на виходах.

Лістинг 4.2 – Код функції attach_interrupts()

```
void attach_interrupts()
{
  pinMode(MOTION_SENSOR_PIN, INPUT_PULLDOWN);
  pinMode(SMOKE_SENSOR_PIN, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(MOTION_SENSOR_PIN),
    motionDetectedInterrupt, RISING);

  attachInterrupt(digitalPinToInterrupt(SMOKE_SENSOR_PIN),
    smokeDetectedInterrupt, FALLING);
}
```

Тепер, при активації датчиків, програма буде зупинятись, та будуть викликатись обробники переривань. Нижче наведено лістинг коду обробників переривань. Обробники переривань мають атрибут IRAM_ATTR. Це потрібно для того, щоб розмістити функції переривань у більш швидку IRAM пам’ять, оскільки переривання мають виконуватись швидко. Flash пам’ять є більш повільним типом пам’яті та може бути зайнятою. Також були оголошені змінні volatile bool motionDetected та volatile bool smokeDetected. Це прапорці, які вказують, чи було виявлено рух або дим відповідно. В їх оголошенні є ключове слово volatile, що впливає на обробку компілятором цієї змінної. Це вказує на те, що ця змінна може змінюватись поза межами коду, і тому її не треба оптимізовувати. В самих функціях обробника переривання змінюються змінні motionDetected та smokeDetected.

Лістинг 4.3 – Код обробників переривань

```
volatile bool motionDetected = false;
volatile bool smokeDetected = false;

static void IRAM_ATTR motionDetectedInterrupt()
{
    motionDetected = true;
}

static void IRAM_ATTR smokeDetectedInterrupt()
```

```
{
    smokeDetected = true;
}
```

3.3.2 Розробка модуля трансляції відео з камери

Другий модуль відповідає за роботу веб-сервера на ESP32-CAM та трансляцію відео на цей веб-сервер. Ці задачі було вирішено об’єднати в один клас, оскільки вони взаємопов’язані. Даний клас називається surveillance_handle. Він використовує конструктор за замовчуванням, та має публічні методи begin(), send_frame(), handle_client() та приватний start_stream().

Лістинг 4.4 – Клас surveillance_handle

```
class surveillance_handle
{
public:
    surveillance_handle() = default;
    surveillance_status_t begin();
    surveillance_status_t send_frame();
    void handle_client();

private:

    void start_stream();
    WebServer server{80};
    WiFiClient client;
    bool is_streaming = false;
    uint32_t last_frame_time_ms = 0;
    uint16_t frame_interval_ms = 1000 / FRAMES_PER_SECOND;
};
```

Метод begin() відповідає за ініціалізацію камери та ввімкнення сервера та повертає результат виконання методу. Дані конфігурації камери зберігаються в змінній config. Далі викликається функція esp_camera_init(&config), яка ініціалізує камеру з конфігурацією, зазначеною в змінній config. Далі перевіряється статус виконання цієї функції, і в випадку якщо функція не повернула ESP_OK статус, то метод повертає

SURVEILLANCE_CAM_ERROR, що сигналізує про проблему з камерою. В іншому випадку, в UART-монітор записується, що камера ініціалізована успішно. Після цього вмикається сервер з обробником HTTP GET-запиту до кореневої сторінки веб-сервера. При відкритті користувачем кореневої сторінки ESP32CAM відбувається виконання анонімної функції, в якій відбувається запис в UART-монітор, що було викликано цю анонімну функцію, та виконується метод цього ж класу start_stream(). Після цього ESP32CAM запускає веб-сервер за допомогою server.begin(), записує в UART-монітор, що сервер запустився, та повертає статус SURVILLANCE_OK.

Лістинг 4.5 – Код методу begin()

```
surveillance_status_t surveillance_handle::begin()
{

    camera_config_t config;

    config.ledc_channel = LEDC_CHANNEL_0;
    config.ledc_timer   = LEDC_TIMER_0;

    config.pin_pwdn     = 32;
    config.pin_reset    = -1;
    config.pin_xclk     = 0;
    config.pin_sccb_sda = 26;
    config.pin_sccb_scl = 27;

    config.pin_d7       = 35;
    config.pin_d6       = 34;
    config.pin_d5       = 39;
    config.pin_d4       = 36;
    config.pin_d3       = 21;
    config.pin_d2       = 19;
    config.pin_d1       = 18;
    config.pin_d0       = 5;

    config.pin_vsync    = 25;
    config.pin_href     = 23;
    config.pin_pclk     = 22;

    config.xclk_freq_hz = 20000000;
    config.pixel_format = PIXFORMAT_JPEG;
    config.frame_size   = FRAMESIZE_VGA;
    config.jpeg_quality = 12;
```

```
        config.fb_count     = 1;
        config.fb_location  = CAMERA_FB_IN_PSRAM;
        config.grab_mode    = CAMERA_GRAB_WHEN_EMPTY;

        esp_err_t err = esp_camera_init(&config);
        if (err != ESP_OK)
        {
            Serial.printf("Camera init failed: 0x%x (%s)\n", err,
esp_err_to_name(err));
            return SURVEILLANCE_CAM_ERROR;
        }

        Serial.println("Camera initialized successfully");

        server.on("/", HTTP_GET, [this]() {
            Serial.println("HTTP GET / - start_stream called");
            this->start_stream();
        });

        server.begin();
        Serial.println("HTTP server started on port 80");
        return SURVEILLANCE_OK;
    }
```

Далі йде метод start_stream(). Він відповідає за підготовку HTTP-сторінки для трансляції відеопотоку. Спочатку отримується поточне з’єднання користувача, який відкрив сторінку ESP32-CAM в браузері. Потім йде перевірка, чи клієнт підключений. Якщо клієнт не підключений, відбувається запис повідомлення в UART-моніторі про те, що клієнт не підключений, та вихід з методу. Якщо клієнт підключений, то у полі класу client зберігається поточний клієнт, встановлюється значення поля is_streaming, яку визначає, чи відбувається трансляція, встановлюється значення last_frame_time_ms в вигляді поточного часу в мілісекундах та записується повідомлення про підключення клієнта в UART-моніторі. Після цього формується HTTP-відповідь для браузера з зазначенням типу змісту, який буде надсилатись, і відправляється сформована відповідь, що дозволить браузеру послідовно приймати JPEG-картинки в якості відеопотоку.

Лістинг 4.6 – Код методу start_stream()

```
void surveillance_handle::start_stream()
{
```

```
          WiFiClient c = server.client();
          if (!c || !c.connected()) {
              Serial.println("No HTTP client connected.");
              return;
          }

          client = c;
          is_streaming = true;
          last_frame_time_ms = millis();

          Serial.println("Client connected - streaming started");

          String response = "HTTP/1.1 200 OK\r\n"
                            "Content-Type:          multipart/x-mixed-replace;
boundary=frame\r\n"
                            "\r\n";
          client.print(response);
      }
```

Передостаннім методом є метод send_frame(). Цей метод відповідає за формування зображення з камери ESP32-CAM та відправлення цього зображення на веб-сервер ESP32-CAM. На початку перевіряється, чи зараз відбувається трансляція відеопотоку та чи під’єднаний клієнт. У випадку, якщо одна з умов не задоволена, тоді відбувається закриття з клієнтом, скидається прапорець is_streaming та метод повертає SURVEILLANCE_WEB_ERROR, що свідчить про помилку, пов’язану з веб-сервером. Якщо перевірка пройшла успішно, то фіксується поточний час в мілісекундах у змінній current_time_ms. Далі відбувається перевірка, чи пройшло достатньо часу з фіксації минулого кадру, щоб не перевантажувати ESP32-CAM, оптимізувати мережевий трафік та встановити стабільну кадрову частоту. Якщо перевірка не була пройдена успішно, то повертається статус SURVEILLANCE_OK. У випадку, якщо перевірку було пройдено, створюється буфер fb, який ініціалізується функцією esp_camera_fb_get(), що повертає структуру camera_fb_t* з JPEG-кадром. Якщо кадр не було отримано, то в UART-монітор записується повідомлення про помилку при отриманні кадра, прапорець is_streaming скидається, клієнт закривається, і метод повертає SURVEILLANCE_CAM_ERROR. В протилежному випадку поточний час знімку кадру зберігається в полі last_frame_time_ms, після чого

відбувається відправка HTTP-відповіді з кадром. Потім відбувається звільнення пам’яті буфера fb, і метод повертає статус SURVEILLANCE_OK.

Лістинг 4.7 - Код методу send_frame()

```
surveillance_status_t surveillance_handle::send_frame()
{
    if (!is_streaming || !client.connected()) {
        client.stop();
        is_streaming = false;
        return SURVEILLANCE_WEB_ERROR;
    }

    uint32_t current_time_ms = millis();
    if (current_time_ms - last_frame_time_ms >= frame_interval_ms) {
        camera_fb_t *fb = esp_camera_fb_get();
        if (!fb) {
            Serial.println("Camera capture failed");
            is_streaming = false;
            client.stop();
            return SURVEILLANCE_CAM_ERROR;
        }

        last_frame_time_ms = current_time_ms;

        client.printf("--frame\r\nContent-Type:   image/jpeg\r\nContent-
Length: %u\r\n\r\n", fb->len);
        client.write(fb->buf, fb->len);
        client.print("\r\n");

        esp_camera_fb_return(fb);

    }

    return SURVEILLANCE_OK;
}
```

Останній метод цього класу - handle_client(). Цей метод є обгорткою для server.handleClient(), який використовується для обробки HTTP-запитів.

Лістинг 4.8 – Код методу handle_client()

```
void surveillance_handle::handle_client()
{
    server.handleClient();
}
```

3.3.3. Розробка основної бізнес-логіки

Останнім модулем є файл main.c, в якому написана вся бізнес-логіка проєкту та в якому об’єднуються інші модулі. Також там є код для керування GSM-модулем SIM800L. Він не потребує окремого модуля, оскільки для нього існують сторонні бібліотеки, які дозволяють працювати з ним.

В цьому проєкті було використано бібліотеку TinyGSM розроблену Володимиром Шиманським. Ця бібліотека розповсюджується під ліцензією GNU Lesser General Public License (LGPL-3.0), що дозволяє використання, зміну та розповсюдження програм, які використовують цю бібліотеку. Вона використовується для керування GSM-модулями в проєктах ESP32 та інших платформах. Ця бібліотека може взаємодіяти з різними модулями GSM, включаючи SIM800L. Бібліотека має функції для надсилання SMS, здійснення голосових дзвінків, якщо GSM-модуль підтримує цю функцію, підключення до мережі Інтернет через GPRS, 3G, 4G, може працювати з MQTT, HTTP, TCP/UDP, та отримувати інформацію про GSM-модуль.

Файл main.c складається з двох основних функцій: setup() та loop().

Функція setup() відповідає за ініціалізацію компонентів системи охорони приміщення. Вона викликається один раз перед основним циклом роботи програми.

На самому початку ініціалізується апаратне UART з’єднання з UART-монітором з частотою 115200 за допомогою Serial.begin(115200). Це необхідно для відстеження повідомлень, які надсилає ESP32-CAM у процесі виконання програми. Це дозволяє виявляти помилки та спостерігати за процесом виконання програми. Після цього ініціалізується UART з’єднання з модулем SIM800L при виконанні SerialAT.begin(9600, SERIAL_8N1, MODEM_RX, MODEM_TX), де 9600 – частота UART, SERIAL_8N1 – конфігурація UART з’єднання, яка означає, що використовується 8 бітів даних та 1 стоп-біт, MODEM_RX і MODEM_TX – GPIO-порти, на які будуть

отримуватись і з яких будуть надсилатись дані. Далі відбувається спроба ввімкнення модему за допомогою modem.restart(). Після виконання в UART-монітор за допомогою Serial.println() надсилається повідомлення, чи було успішно ввімкнено модем. Після цього перевіряється, чи можливо вести з модемом комунікацію, і в UART-монітор виводиться результат цієї перевірки. Після цього відбувається спроба підключення до WiFi за допомогою функції WiFi.begin(ssid, password). SSID та пароль до WiFi-мережі знаходяться в окремому файлі secrets.h, який є доданим до файлу .gitignore, щоб дані про WiFi-мережу не були оголошені. Далі в UART-монітор виводиться повідомлення про локальну IP-адресу в мережі, щоб користувач міг зайти на неї і почати дивитися відеопотік. Далі викликається surveillance.begin() для ініціалізації веб-сервера та початку відеопотоку. Після цього викликається функція attach_interrupts(), яка підключає переривання на виводах мікроконтролера, до яких підключені датчик диму та датчик руху.

Лістинг 4.9 – Код функції setup()

```
void setup()
{
  Serial.begin(115200);

  SerialAT.begin(9600, SERIAL_8N1, MODEM_RX, MODEM_TX);
  delay(300);

  Serial.println("Initializing modem...");
  if (!modem.restart()) {
    Serial.println("Modem restart failed!");
  } else {
    Serial.println("Modem restarted");
  }

  if (!modem.testAT()) {
    Serial.println("Failed to communicate with modem");
  } else {
    Serial.println("Modem connected!");
    Serial.println("Modem Info: " + modem.getModemInfo());
  }

  WiFi.begin(ssid, password);
  delay(1000);
  Serial.println("WiFi IP: " + WiFi.localIP().toString());
```

```
  surveillance.begin();
  attach_interrupts();
}
```

Далі функція loop(). Це функція, яка буде виконуватись постійно в нескінченному циклі. Ця функція забезпечує постійне виконання бізнес-логіки пристрою.

З самого початку викликається метод surveillance.handle_client(). Цей метод відповідає за обробку HTTP-запитів від клієнта. Після цього викликається метод surveillance.send_frame(), який відповідає за формування та передачу кадру клієнту. І далі перевіряється, чи було викликане переривання на виході, до якого підключений датчик руху. У випадку, якщо воно було викликано, то в UART-монітор виводиться повідомлення, що було виявлено рух, за допомогою SIM800L відправляється повідомлення на номер телефону, що було виявлено рух, та скидається прапорець motionDetected. Потім йде перевірка переривання на виході, до якого підключений датчик диму. В випадку, якщо дим було виявлено, в UART-монітор виводиться повідомлення про те, що було виявлено дим, надсилається повідомлення на номер телефону, що було виявлено загрозу диму, та скидається прапорець smokeDetected. Після цього функція буде виконуватись, поки мікроконтролер має живлення і не буде перезавантажений.

Лістинг 4.10 – Код функції loop()

```
void loop()
{
  surveillance.handle_client();
  surveillance.send_frame();

  if (motionDetected)
  {
    Serial.println("Motion detected!");
    modem.sendSMS(phone_number, "Motion detected!");
    motionDetected = false;
  }
```

```
  if (smokeDetected)
  {
    Serial.println("Smoke detected!");
    modem.sendSMS(phone_number, "Smoke detected!");
    smokeDetected = false;
  }
}
```

ВИСНОВКИ

У даній кваліфікаційній роботі були детально розглянуті аспекти створення системи охорони приміщення на базі мікроконтролера ESP32, зокрема використання інфрачервоного датчика руху та GSM модуля для забезпечення охорони приміщення.

Важливість системи охорони полягає у забезпеченні захисту приміщення від несанкціонованого доступу. Вчасне виявлення руху у контрольованій зоні та оперативне інформування про порушення є критичними для забезпечення надійної охорони.

Розглянуто стандартну класичну схему системи охорони з використанням інфрачервоного датчика руху для виявлення руху у контрольованій зоні. Схему модифіковано відповідно до сучасних вимог користувача, додано можливість відправки повідомлень через GSM модуль. У результаті отримано модернізовану схему системи охорони, яка забезпечує високу ефективність та надійність.

На основі технічного завдання у вигляді функціоналу розроблено алгоритм роботи даної системи. Алгоритм передбачає виявлення руху, обробку сигналу мікроконтролером, відправку повідомлення про порушення на мобільний телефон користувача та спостереження за приміщенням у вигляді відеотрансляції. Це дозволяє оперативно реагувати на потенційні загрози та забезпечувати максимальний рівень безпеки.

Таким чином, виконання кваліфікаційної роботи дозволило успішно реалізувати систему охорони на базі мікроконтролера ESP32, з використанням інфрачервоного датчика руху та GSM модуля. Це забезпечило отримання цінного досвіду у розробці та впровадженні сучасних технологій безпеки, а також підтвердило ефективність та надійність створеної системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ЩО ТАКЕ ARDUINO IDE. REPORTER | Information portal. URL: https://reporter.zp.ua/shho-take-arduino-ide.html?utm_source=chatgpt.com (дата звернення: 09.06.2025).

2. From Wired to Wireless: Discovering the Types of Home Security Systems / Master Home Security s. r. o. – Режим доступу: www / URL: https://www.zoomon.camera/types-of-home-security-system/ 30.05.2023 – Загол. з екрану

3. Home Security: The History, The Evolution, The Future / Origin Wireless Inc. - Режим доступу: www / URL: https://www.originwirelessai.com/insights/home-security-the-history-the-evolution-the-future/ 08.11.2021 – Загол. з екрану.

4. Staff L. E. In-Depth: How MQ2 Gas/Smoke Sensor Works? & Interface it with Arduino. Last Minute Engineers. URL: https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/ (date of access: 07.06.2025).

5. UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter / Analog Devices Inc. – Режим доступу: www / URL : https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html/ Грудень 2020

6. Understanding the Key Components of a Home Security System / Vivint Inc. – Режим доступу: www / URL: https://www.vivint.com/resources/article/components-of-a-home-security-system/ 20.06.2023 – Загол. з екрану