НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Комп'ютерних систем, мереж та кібербезпеки

_____ Касаткін Д.Ю., к.пед.н., доц.

(підпис) (ПІБ, вчене звання і ступінь)

«___»_____2025 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

На тему: Розробка спеціалізованого модуля на основі ARDUINO для охоронної системи автомобіля

Спеціальність 123 «Комп'ютерна інженерія»

Гарант освітньої програми

к.фіз.-мат.н., доц. / Нікітенко Є.В. /

(підпис) (ПІБ)

Керівник дипломного проекту: / Нікітенко Є.В./

(підпис) (ПІБ)

Виконав: / Каратуша О.С./

(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ

Завідувач кафедри
Комп'ютерних систем, мереж та кібербезпеки

_____/ Касаткін Д.Ю, доцент, к.пед.н /

підпис

"_ "_____2025 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

студент Каратуша Олександр Сергійович

Спеціальність 123 «Комп'ютерна інженерія»

Тема бакалаврської кваліфікаційної роботи: Розробка спеціалізованого модуля на основі ARDUINO для охоронної системи автомобіля

Затверджена наказом ректора НУБіП України від 16.12.2024 № 2250 "С"

Термін подання завершеної роботи на кафедру _____

(рік, місяць, число)

Вихідні дані до роботи: опис програмного забезпечення

Перелік питань , які потрібно розробити:

Аналіз проблемної області, вибір та обґрунтування засобів для розробки системи, проектування інформаційної системи.

Дата видачі завдання "16" 12 2025р.

Керівник бакалаврської кваліфікаційної роботи

к.фіз.-мат.н., доц. _____ Нікітенко Є.В

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ Каратуша О.С.

(підпис) (ПІБ студента)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання бакалаврської кваліфікаційної роботи	Строк виконання етапів бакалаврської кваліфікаційної роботи	Примітка
1	Отримання завдання бакалаврської кваліфікаційної роботи	16.12.2024	
2	Початок планування системи	23.01.2025 – 03.02.2025	
3	Побудова UML діаграм	25.03.2025-27.04.2025	
4	Розробка програми і тестування	08.03.2025-23.04.2025	
5	Написання пояснювальної записки	24.04.2025-16.05.2025	
6	Перевірка на плагіат	17.05.2025	
7	Відправка дипломної записки	17.05.2025	
8	Захист дипломної роботи	25.05.2025-15.05.2025	

Студент _____ ___Каратуша О.С_____

(підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____ _____Нікітенко Є.В._____

(підпис) (ПІБ)

ЗМІСТ

ВСТУП

Актуальність теми дослідження зумовлена високим рівнем викрадень автомобілів та зростанням потреби у ефективних і надійних охоронних системах, здатних забезпечити безперервний моніторинг стану транспортного засобу та оперативно реагувати на спроби несанкціонованого доступу. Значна частина існуючих комерційних рішень характеризується високою вартістю, закритістю архітектури та недостатньою адаптивністю до конкретних вимог власників автомобілів. Використання відкритих апаратних платформ, таких як Arduino, дозволяє розробити економічно ефективні, гнучкі та спеціалізовані модулі охорони, які можна інтегрувати у будь-які моделі транспортних засобів, зберігаючи високий рівень безпеки.

Метою роботи є розробка спеціалізованого програмно-апаратного модуля на базі платформи Arduino, призначеного для автоматизованого моніторингу та сигналізації про несанкціоноване проникнення чи спроби викрадення автомобіля.

Для досягнення зазначеної мети визначено такі завдання:

1. провести аналіз існуючих охоронних систем автомобілів, визначивши їхні переваги та недоліки.
2. Розробити структурну та функціональну модель спеціалізованого модуля охоронної системи.
3. Обрати оптимальну конфігурацію апаратних компонентів платформи Arduino для вирішення поставлених завдань.
4. Розробити програмне забезпечення для управління модулем та реалізації основних функцій моніторингу й сигналізації.
5. Провести експериментальне тестування розробленого модуля з метою перевірки його ефективності та надійності роботи.

Об’єктом дослідження є охоронні системи автомобілів.

Предметом дослідження є методи проектування і розробки спеціалізованого програмно-апаратного модуля на платформі Arduino для забезпечення безпеки автомобілів.

У роботі використано методи системного аналізу, програмно-апаратного проектування, моделювання електронних схем, алгоритмізації та експериментального тестування розробленого обладнання. Основними засобами реалізації є мікроконтролерна платформа Arduino, середовище розробки Arduino IDE, програмна мова C++, датчики руху, сенсори відкриття дверей та модулі зв’язку GSM/GPRS.

Практичне значення роботи полягає в розробці доступного, економічного та ефективного рішення для охоронних систем транспортних засобів, яке може бути застосоване в приватному секторі та малому бізнесі.

Дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У першому розділі виконано аналіз предметної області та огляд існуючих технічних рішень, другий присвячено проектуванню апаратної та програмної частин модуля, третій містить опис реалізації запропонованого рішення, а четвертий — аналіз результатів тестування та рекомендації щодо експлуатації розробленого модуля. У висновках сформульовано підсумки проведених досліджень та запропоновано напрямки подальшого розвитку.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Стан та проблематика охоронних систем автомобілів

Системи охорони транспортних засобів є складовим елементом загальної концепції безпеки автомобіля та виконують функцію запобігання несанкціонованому доступу, викраденню та пошкодженню транспортного засобу або його окремих компонентів. У зв’язку з посиленням загроз інформаційного та фізичного характеру в автомобільній галузі, охоронні системи зазнають постійного розвитку як у технічному, так і в програмному аспектах. Їх функціональна складність зростає паралельно з електронізацією транспортних засобів, що призводить до необхідності їх точного класифікаційного розподілу для аналізу переваг, обмежень та сфер застосування кожного типу.

Визначення типових класів охоронних систем дозволяє не лише оптимізувати вибір рішень під конкретні вимоги, а й формалізувати підхід до розробки нових архітектур захисту, зокрема при реалізації модулів на базі вбудованих систем. З урахуванням зазначеного, доцільно проводити класифікацію охоронних систем за рядом ключових ознак: способом активації, рівнем інтеграції в електронну інфраструктуру автомобіля, характером застосовуваних захисних механізмів, а також типом доступу до інтерфейсів керування, відповідно наведена узагальнена класифікація охоронних систем у таблиці 1.1 за базовим принципом дії.

Таблиця 1.1 – Класифікація охоронних систем автомобіля за принципом дії

Тип охоронної системи	Коротка характеристика
Пасивні системи	Автоматизовані системи, які активуються без втручання користувача після певного періоду бездіяльності. Зазвичай блокують запуск двигуна, роботу стартера або подачу палива. Типовим прикладом є іммобілайзер. Не потребують додаткових дій для активації, однак мають обмежений функціонал [1].

Продовження таблиці 1.1

Активні системи	Потребують явної дії з боку користувача для увімкнення або вимкнення системи (натискання кнопки, введення коду, використання брелока). Надають розширені можливості конфігурації, дозволяють вмикати різні датчики (удару, об'єму, нахилу) та режим тривоги [2].
Інтегровані системи	Вбудовані в електронну архітектуру автомобіля (OEM), синхронізуються з блоками керування через шину CAN, забезпечують глибоку взаємодію з іншими електронними системами транспортного засобу. Можуть мати телематичні функції, взаємодіяти з мобільними застосунками або хмарними платформами [3].

Залежно від рівня реалізації та функціонального наповнення охоронні системи також поділяються за типом захисних механізмів, які вони використовують:

– Фізичні засоби захисту – включають механічні блокатори керма, коробки передач, замки капота та педалей. Ці пристрої забезпечують прямий фізичний бар'єр, але не мають можливості дистанційного керування чи сповіщення [4].

– Електронні засоби захисту – використовують датчики руху, нахилу, відкриття дверей, зчитувачі сигналу, іммобілайзери. Реалізуються через мікроконтролери або мікропроцесори, з можливістю інтеграції з зовнішніми інтерфейсами [5].

– Програмні засоби захисту – включають алгоритми автентифікації, шифрування, динамічне кодування сигналу (rolling code), а також виявлення вторгнень. Цей рівень захисту активно використовується в системах типу smart key та GSM-сигналізаціях [6].

Із розвитком засобів телекомунікацій зросла популярність телематичних систем, які передбачають використання мобільних мереж (GSM, LTE) для віддаленого контролю за станом автомобіля. Такі системи здатні передавати повідомлення про порушення, а також дозволяють керування функціями блокування через мобільний додаток або веб-інтерфейс. У поєднанні з GPS-

модулями вони дозволяють відстежувати місцезнаходження автомобіля в режимі реального часу, а в разі викрадення – координати його руху [7].

Автономні системи, навпаки, функціонують незалежно від зовнішніх інфраструктур. Вони забезпечують захист на основі локальних датчиків та контролерів, які приймають рішення безпосередньо на борту транспортного засобу. Такі рішення часто базуються на мікроконтролерах типу Arduino, що дає змогу реалізувати індивідуальні охоронні функції без підключення до інтернету або GSM-мереж. Це робить систему менш вразливою до атак через мережу, але обмежує можливості дистанційного керування та моніторингу [8].

Сучасний ринок автомобільних охоронних систем характеризується високим ступенем інтеграції з електронною інфраструктурою транспортного засобу, а також впровадженням новітніх телекомунікаційних і сенсорних технологій. Основним трендом останніх років є перехід від автономних або обмежено функціональних охоронних модулів до багаторівневих систем безпеки, які функціонують у складі загальної електронної архітектури автомобіля та забезпечують багатоканальний моніторинг, діагностику та реагування.

Центральним елементом інформаційного обміну в більшості сучасних транспортних засобів є шина передачі даних типу CAN (Controller Area Network). Її використання дозволяє синхронізувати роботу охоронної системи з блоками керування двигуном, дверима, підвіскою та іншими підсистемами. Незважаючи на зручність та стандартизованість CAN-протоколу, він залишається вразливим до фізичного доступу та несанкціонованого сканування пакетів у разі відсутності ізоляції між шинами [9].

На рисунку 1.1 представлено порівняння традиційної архітектури зв'язку електронних пристроїв у транспортному засобі (ліворуч) з архітектурою на основі CAN-шини (праворуч). У варіанті без CAN кожен пристрій має окреме з’єднання з електронним блоком керування (ECU), що створює складну сітку дротових каналів. У варіанті з CAN усі пристрої підключаються до спільної

двопровідної шини, через яку відбувається обмін даними, що забезпечує більш ефективну та масштабовану мережеву топологію.

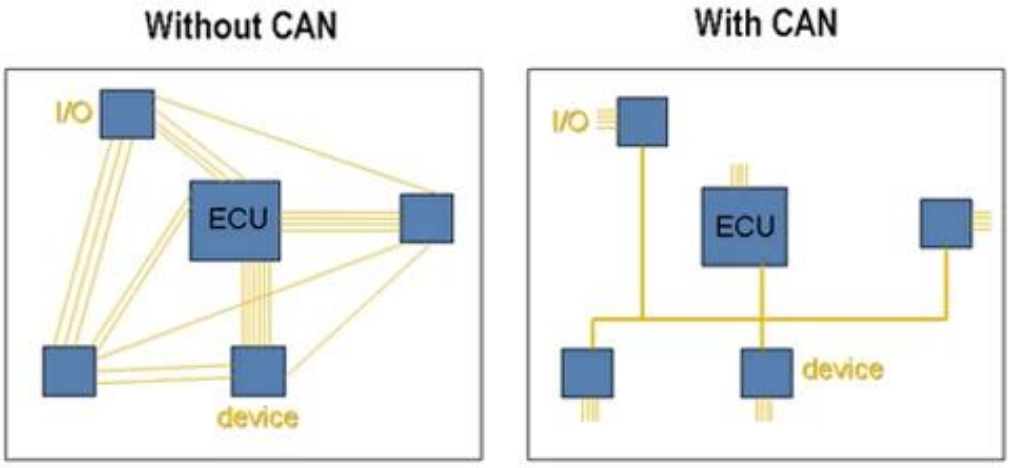

Рисунок 1.1 – Порівняння структури підключення пристроїв у транспортному засобі без CAN-шини (зліва) та з CAN-шиною (справа)

Ця архітектурна оптимізація дозволяє інтегрувати нові модулі охоронних систем без значного втручання в існуючу інфраструктуру, що є надзвичайно важливим для забезпечення гнучкості й масштабованості автомобільної електроніки [9].

У якості засобів ідентифікації та автентифікації користувачів застосовуються технології радіочастотної ідентифікації (RFID) та NFC. Їх використання дозволяє реалізувати безконтактний доступ до автомобіля, однак вони є вразливими до релейних атак, що полягають у перехопленні та ретрансляції сигналу від легітимного пристрою до транспорту. Такий тип атаки є типовим для систем «keyless entry» [10].

Інтеграція GSM/GPRS-модулів та GPS/GLONASS-навігаційних систем забезпечила реалізацію телематичних функцій, які включають віддалене керування постановкою на охорону, блокуванням двигуна, а також геолокаційний супровід транспортного засобу. За допомогою мобільних додатків користувач отримує доступ до функціоналу охорони в реальному часі, що є важливим елементом персоніфікованої безпеки [11].

Охоронні системи також активно використовують сенсорні модулі, серед яких найбільш поширеними є ультразвукові, інфрачервоні, гіроскопічні та

магніторезистивні датчики. Вони дозволяють виявляти рухи всередині салону, нахил кузова, удари, спроби зламування замків або відкриття капота. Завдяки низькому енергоспоживанню та компактності, ці сенсори широко використовуються в embedded-рішеннях на базі мікроконтролерів типу Arduino, STM32 або ESP32 [12].

Глобальним трендом ринку охоронних технологій є інтеграція концепцій інтернету речей (IoT), що передбачає об'єднання охоронної системи з хмарною платформою, сервісами телеметрії та мобільними пристроями. Такий підхід дозволяє реалізувати адаптивні механізми реагування, аналіз поведінки користувача, віддалене оновлення мікропрограмного забезпечення, а також підвищити загальний рівень стійкості до загроз. Водночас, з розвитком IoT-технологій виникають нові виклики, пов'язані з необхідністю забезпечення криптографічного захисту трафіку, автентифікації пристроїв і управління їх життєвим циклом [13].

Іншим важливим напрямом є використання технологій штучного інтелекту в системах охорони. Методи машинного навчання використовуються для детектування аномальних подій, зниження кількості хибних спрацювань сенсорів, адаптації порогових значень тригерів залежно від середовища, а також прогнозування потенційних атак на основі історичних даних [14].

Попри суттєвий прогрес у функціональності та інтелектуальності автомобільних охоронних систем, актуальною залишається проблема їх вразливості до широкого спектру атак. Найпоширенішими є технічні загрози, спрямовані на перехоплення, модифікацію або глушіння сигналів у внутрішньоавтомобільних мережах або між зовнішніми модулями.

Однією з найнебезпечніших є relay-атака, яка полягає у ретрансляції сигналу між легітимним ключем (зазвичай розташованим у житловому приміщенні) та автомобілем. У результаті система безконтактного доступу помилково сприймає ключ як такий, що знаходиться в безпосередній близькості до автомобіля, та автоматично розблоковує двері. Реалізація атаки передбачає участь двох зловмисників: перший перебуває біля автомобіля, другий — поруч

із місцем зберігання ключа. Зв’язок між ними здійснюється через канал дальністю від 30 до 300 метрів, що ілюструє значну небезпеку даного методу. Відповідно схема реалізації relay-атаки представлена на рис.1.2.

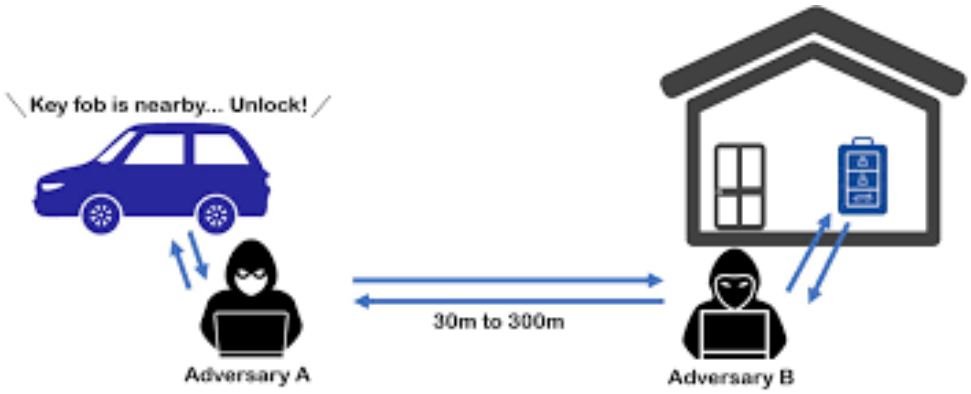

Рисунок 1.2– Схема реалізації relay-атаки з використанням двох зловмисників для ретрансляції сигналу від ключа до автомобіля

Такі атаки є надзвичайно ефективними проти систем безконтактного доступу, особливо коли відсутня автентифікація джерела сигналу або додаткове шифрування на рівні протоколу зв’язку [15].

Іншою серйозною загрозою є заглушення радіочастотного сигналу (jamming). Цей тип атаки реалізується шляхом створення високочастотного шуму на частотах, що використовуються для керування охоронною системою (315 або 433 МГц), унаслідок чого команда блокування не надходить до автомобіля. У деяких випадках користувач вважає, що транспортний засіб поставлено на охорону, хоча фактично це не так [16].

Крім перехоплення та блокування сигналу, зростає кількість атак на шину CAN через інтерфейс діагностичного роз’єму OBD-II. У разі фізичного доступу до нього зловмисник може надсилати фальшиві команди до блоків керування, наприклад, команду розблокування дверей або деактивації іммобілайзера, обминаючи систему тривоги [17].

У контексті програмних атак основними векторами є вразливості у мікропрограмному забезпеченні (firmware) електронних блоків керування, слабкість криптографічних механізмів автентифікації, а також відсутність багаторівневої системи перевірки доступу. Дослідження вказують, що

зловмисники можуть симулювати сигнал ключа або повністю відтворити протокол обміну командами з метою обману системи [18].

Людський фактор також суттєво впливає на рівень безпеки охоронних систем. Користувачі часто ігнорують оновлення мікропрограмного забезпечення, не змінюють стандартні паролі у мобільних застосунках або деактивують частину функціоналу системи з міркувань зручності. Такі дії значно послаблюють стійкість до атак навіть у разі наявності технічно досконалих рішень.

Окрему загрозу становлять штатні OEM-системи, що встановлюються виробником. Вони, як правило, мають закриту архітектуру, обмежені можливості оновлення, низьку гнучкість конфігурації та використовують спільні шифри для однієї серії транспортних засобів. Це робить їх особливо вразливими до масових атак із використанням публічно доступних інструментів аналізу CAN-протоколів [19].

У сукупності наведені приклади свідчать про необхідність комплексного підходу до побудови захисту автомобіля — поєднання апаратних, програмних і організаційних заходів безпеки, інтеграцію криптографічного захисту, багатофакторної автентифікації та підвищення цифрової гігієни користувача.

Надалі доцільно зосередити увагу на розробці гнучких, автономних і адаптивних модулів охорони з вбудованими засобами самодіагностики, криптографічного захисту та інтеграції з мобільними сервісами, що й становить мету подальшого проєктування системи на основі платформи Arduino.

1.2 Огляд сучасних мікроконтролерів для систем безпеки

Інтенсивне впровадження вбудованих обчислювальних систем у сферу автомобільної безпеки зумовлює необхідність використання мікроконтролерів як центральних елементів сучасних охоронних комплексів. Ці пристрої забезпечують не лише керування окремими модулями сигналізації, сенсорами, GSM-передавачами або GPS-модулями, але й реалізують алгоритми прийняття рішень у режимі реального часу, криптографічний захист даних, дистанційну взаємодію з користувачем і автоматичне оновлення мікропрограм. При цьому до мікроконтролерів, які застосовуються в охоронних системах, висуваються особливі вимоги: стабільна робота в умовах температурних коливань, низьке енергоспоживання, підтримка широкого спектру інтерфейсів зв’язку, сумісність із різними типами пам’яті, висока перешкодостійкість та наявність апаратного захисту.

На ринку існує низка популярних мікроконтролерів, що задовольняють ці критерії. Серед них виділяються такі як Microchip PIC, ESP32, STM32, NXP S32K, а також ATmega328P, який традиційно застосовується у платформах Arduino. Вони істотно відрізняються за архітектурою, підтримкою інтерфейсів, рівнем захисту, призначенням і варіантами виконання.

Мікроконтролери PIC серій 16F і 18F від компанії Microchip Technology мають архітектуру Harvard з розділенням команд і даних, що забезпечує високу швидкодію та ефективне використання пам’яті, що можна побачити на рис.1.3, де представлені мікроконтролери PIC у різних корпусах

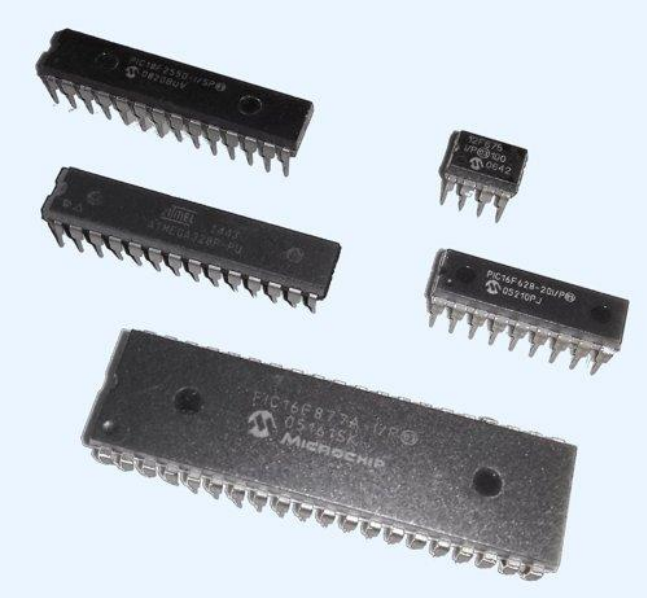

Рисунок 1.3 - Мікроконтролери Microchip PIC у різних корпусах, що широко застосовуються в автомобільних системах

Завдяки широкому діапазону робочих температур (від -40°C до +125°C), високій електромагнітній стійкості та сертифікації за стандартами AEC-Q100, ці контролери є популярними в OEM-системах, які вбудовуються в комерційні транспортні засоби. Їх перевагами є наявність внутрішньої EEPROM для збереження конфігурацій, декілька рівнів пріоритету переривань, WDT, модулі PWM, USART, SPI та I2C. Також контролери мають низьке споживання у сплячому режимі, що є критично важливим для автономних охоронних пристроїв, що працюють від батареї [20].

ESP32 від Espressif Systems — це мікроконтролер з двома ядрами Tensilica LX6/LX7 (в залежності від ревізії), тактовою частотою до 240 МГц, вбудованим Wi-Fi, Bluetooth Classic/LE, високоточною годинниковою системою та апаратним прискоренням криптографічних операцій (SHA-256, AES, RSA, ECC). Представлений мікроконтролер на рис.1.4.

Рисунок 1.4 - Плата ESP32 NodeMCU з бездротовими інтерфейсами, яка використовується у мобільних охоронних модулях

Його популярність у системах безпеки обумовлена наявністю OTA (over-the-air update), підтримкою MQTT, HTTP, TLS, функцій deep sleep, інтеграцією з хмарними сервісами. Цей контролер використовується для реалізації мобільного доступу до автомобіля, отримання повідомлень про порушення з будь-якої точки світу, контролю за геопозицією автомобіля, зчитування даних із датчиків руху, нахилу чи температури [21].

Розглядаючи (представлені на рис.1.5) мікроконтролери серії STM32 на базі ядра ARM Cortex-M виробництва STMicroelectronics які забезпечують високу продуктивність, підтримку апаратного CAN, DMA, низьке енергоспоживання та розширений набір периферії. Залежно від серії (STM32F1, F4, G0, L4), вони підходять для широкого спектра задач: від обробки сенсорних сигналів до реалізації криптографічних операцій у режимі реального часу.

Рисунок 1.5 - STM32F103C8T6 (так звана “Blue Pill”) — недорога та потужна платформа для систем моніторингу та захисту

Багато мікроконтролерів STM32 мають сертифікацію ISO 26262, що дозволяє їхнє використання в функціонально критичних автомобільних системах [22].

Лінійка S32K компанії NXP Semiconductors створена спеціально для використання в автомобільних електронних блоках управління. Мікроконтролери цієї серії (представлено на рис.1.6) побудовані на базі ARM Cortex-M4/M7, мають апаратну підтримку CAN FD, LIN, I2C, FlexIO, Cryptographic Security Engine, вбудовану флеш-пам’ять із захистом від запису, та підтримку стандарту AEC-Q100. Їх функціональні можливості дозволяють реалізацію як класичних охоронних систем, так і високорівневих рішень з інтеграцією в архітектуру ECU [23].

Рисунок 1.6 - NXP S32K — мікроконтролер OEM-класу

ATmega328P, який лежить в основі Arduino Uno, є 8-бітним мікроконтролером архітектури AVR, що поєднує простоту програмування, широкий набір підтримуваних бібліотек, низьке енергоспоживання та наявність SPI, UART, PWM, I2C. Хоча він поступається STM32 чи ESP32 у потужності, ATmega328P є оптимальним для розробки прототипів охоронних систем на етапі тестування або у проєктах, де потрібна базова логіка, надійність і незалежність від складної інфраструктури. Arduino Uno на базі цього мікроконтролера буде

використано в даній роботі як основа для реалізації автономного модуля охорони з базовим набором функцій (датчики руху, реле блокування, GSM-модуль) [24].

1.3 Аналіз наявних рішень на основі Arduino

Arduino-платформа завдяки відкритій апаратній архітектурі, великій спільноті розробників і широкому вибору сумісних модулів є однією з найпоширеніших основ для створення охоронних систем як у рамках навчальних, аматорських, так і промислових проєктів. Серед основних переваг Arduino — низька вартість, підтримка стандартних інтерфейсів (UART, I2C, SPI), сумісність із численними цифровими та аналоговими сенсорами, доступність бібліотек і простота налагодження. Ці фактори роблять Arduino зручним інструментом для прототипування, навчання, а також для реалізації повнофункціональних охоронних модулів. Нижче наведено аналіз чотирьох практичних рішень на базі Arduino, які реалізують різні аспекти захисту транспортних засобів.

Одним із найбільш поширених рішень є система з використанням Arduino UNO або Nano, датчика руху (PIR або ультразвукового HC-SR04) та GSM-модуля SIM800L. Принцип дії полягає у виявленні вторгнення в салон автомобіля або наближення до авто за допомогою сенсорів, після чого Arduino надсилає SMS-повідомлення власнику. У деяких реалізаціях також передбачено ініціювання дзвінка, що є додатковим тригером для користувача.

На рисунку 1.7 представлено одну з реалізацій охоронної системи на основі ультразвукового датчика HC-SR04. При фіксації руху або зменшенні дистанції до об’єкта, Arduino активує сервомеханізм SG90 (наприклад, для блокування замка) і вмикає п’єзо-сирену, що сигналізує про загрозу.

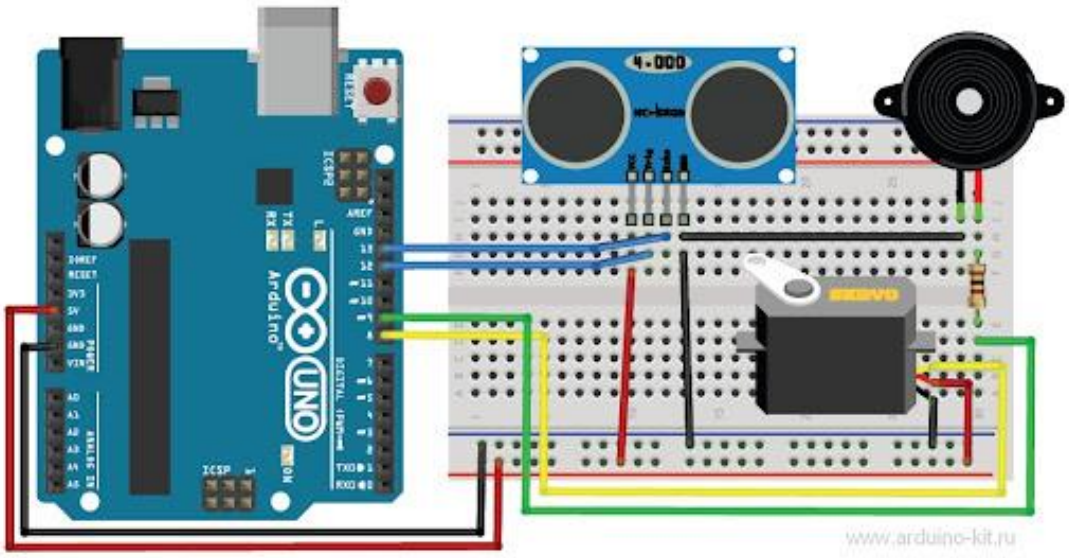

Рисунок 1.7 – Ультразвуковий датчик, сервопривід та сирена, з’єднані з Arduino UNO для реалізації охоронного режиму

Ця система не вимагає підключення до Інтернету, працює автономно і може бути живлена від акумулятора або штатного живлення автомобіля. Її ключова перевага — простота реалізації при мінімальному наборі компонентів і висока ефективність у режимі очікування [25].

Розглянемо охоронну систему з RFID-ідентифікацією водія. У даному рішенні використовується Arduino UNO або MEGA, RFID-зчитувач RC522 і набір міток (ключ-карт). При підході до автомобіля користувач прикладає RFID-карту до зчитувача, який фіксує UID і порівнює його з дозволеними значеннями. Якщо UID відповідає збереженому в EEPROM ідентифікатору, система дозволяє запуск двигуна або знімає автомобіль із охорони.

На рисунку 1.8 зображено підключення RFID-модуля RC522 до Arduino через інтерфейс SPI. Така конфігурація дозволяє реалізувати базовий рівень автентифікації без використання ключів або брелоків, і може бути інтегрована в корпус автомобіля.

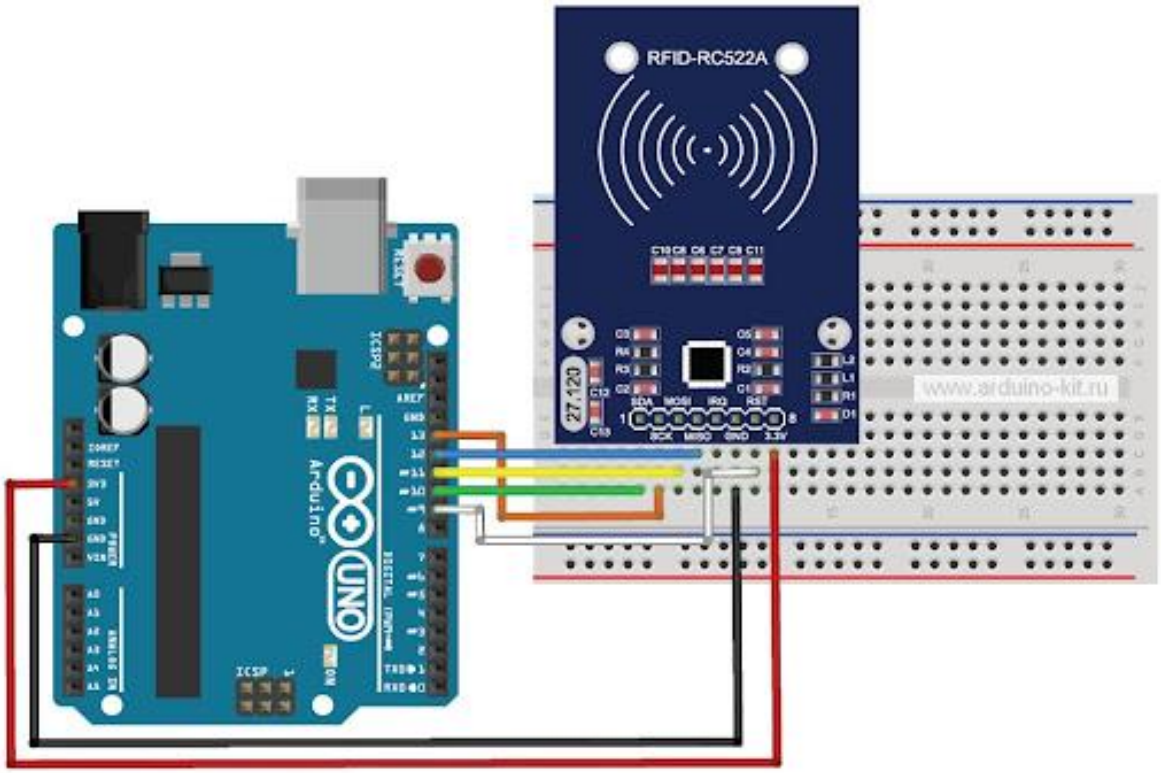

Рисунок 1.8 – Зчитувач RFID RC522, з’єднаний з Arduino UNO для контролю доступу до транспортного засобу

Це рішення є простим аналогом безключового доступу, дозволяє реалізувати багаторівневу автентифікацію (наприклад, у поєднанні з клавіатурою або відбитком пальця) і може бути встановлене приховано в салоні [26].

Ще одним прикладом є реалізація трекінгової охоронної системи, яка базується на Arduino Nano або Pro Mini, GPS-модулі (NEO-6M/NEO-M8N) та реле з нормально замкнутим контактом, яке блокує ланцюг живлення паливного насоса або стартера. Система дозволяє виявляти несанкціоноване переміщення автомобіля, а також надсилати повідомлення з координатами або блокувати запуск.

На рисунку 1.9 представлено схему з'єднання GPS-модуля NEO-6M з Arduino. Модуль підключається до послідовного інтерфейсу, дані координат обробляються і можуть бути передані через GSM або Bluetooth.

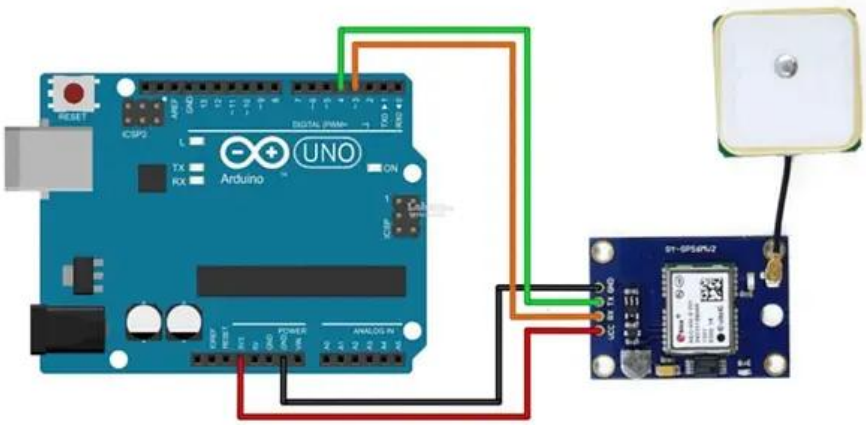

Рисунок 1.9 – Arduino UNO з’єднаний з GPS-модулем для реалізації функції трекінгу транспорту

Це рішення дозволяє реалізувати телеметрію у поєднанні з охороною, а також інтегрується з Google Maps API або іншими картографічними сервісами для онлайн-відстеження транспорту [27].

Також варто розглянути охоронну система з мобільним Bluetooth-керуванням (HC-05/HC-06). У цій конфігурації використовується Bluetooth-модуль HC-05 або HC-06, який з’єднується з Arduino та дозволяє керувати охоронними режимами через мобільний застосунок. Через Bluetooth можна активувати або деактивувати охоронний режим, керувати замками, фарами або запуском двигуна. Такі застосунки можуть бути створені на Android-платформах із використанням MIT App Inventor або Android Studio.

Основна перевага цієї архітектури — можливість локального бездротового керування без залежності від GSM або Wi-Fi. Однак дальність Bluetooth обмежена до 10–12 метрів, що робить систему ефективною лише в зоні прямої дії [28].

Розробки на основі Arduino демонструють практичну гнучкість цієї платформи у сфері охорони транспортних засобів. Від простих модулів оповіщення до повноцінних систем автентифікації, трекінгу, блокування та мобільного управління — кожне рішення може бути масштабованим та інтегрованим в OEM або самостійно розроблену систему.

1.4 Постановка завдання та технічні вимоги до розробки

Зростання рівня автоматизації транспортних засобів, впровадження інтелектуальних функцій у бортові електронні системи та розширення бездротової комунікації суттєво підвищили вразливість автомобілів до зовнішніх кібер- та фізичних загроз. При цьому масове використання безключового доступу, телематичних модулів і CAN-шин у штатних охоронних системах виявило численні недоліки існуючих архітектур безпеки, зокрема:

- слабкість автентифікації користувача та відсутність багатофакторної перевірки доступу;
- технічну можливість здійснення relay-атак, зламів через OBD-II та підміни ідентифікаторів у CAN-пакетах;
- недостатній контроль з боку кінцевого користувача над оновленням або розширенням функцій безпеки;
- відсутність підтримки гнучкої інтеграції нових сенсорних, криптографічних або трекінгових модулів.

Крім того, більшість штатних OEM-систем є закритими як у програмному, так і в апаратному сенсі, що унеможливлює оперативне оновлення або адаптацію таких систем під нові вимоги чи реальні сценарії загроз.

У таких умовах особливої актуальності набуває розробка автономного, модульного охоронного рішення, що не лише виконує базові функції виявлення вторгнення або блокування, але й забезпечує:

- незалежність від центральних ECU або CAN-шини виробника;
- підтримку адаптивного реагування на загрози в реальному часі;
- можливість дистанційної індикації подій та трекінгу об'єкта;
- відкриту структуру для інтеграції з додатковими захисними модулями.

Як базу для реалізації такого модуля доцільно використати платформу Arduino, яка забезпечує:

- сумісність із великою кількістю сенсорів (руху, нахилу, вібрації, GPS, температури тощо);
- підтримку засобів бездротового зв’язку (GSM, Bluetooth, Wi-Fi) для мобільної взаємодії;
- відкриту програмну архітектуру, що дозволяє повний контроль за логікою роботи системи;
- низьке енергоспоживання та компактність для вбудованих рішень.

Науково-технічною проблемою, яка вирішується в цій роботі, є інтеграція множинних джерел даних (сенсори, модулі зв’язку, RFID, GPS), їх синхронізація, обробка та управління виконавчими механізмами в умовах обмежених апаратних ресурсів мікроконтролера, із дотриманням вимог щодо енергетичної ефективності, надійності та масштабованості.

Метою дослідження є розробка, реалізація та експериментальна перевірка охоронного модуля для автомобіля на основі Arduino, що здатен виконувати:

- локальний контроль присутності (виявлення вторгнення в зону або салон);
- автентифікацію користувача через RFID або інший захищений інтерфейс;
- формування керуючих сигналів на блокувальні реле;
- передавання інформації про тривожні події власнику через GSM/мобільні мережі;
- визначення та передачу координат транспортного засобу через GPS у разі викрадення;
- архітектурну відкритість для подальшого розширення функціональності (включно з впровадженням криптографії, машинного навчання, біометричної ідентифікації тощо).

Інженерне завдання включає формування логічної структури системи, синтез апаратної конфігурації на основі доступних модулів, розробку програмного коду з використанням бібліотек Arduino IDE та проведення

симуляцій і натурного тестування. Особливу увагу приділено модульності, структурній автономності кожного компонента та відповідності логіки обробки подій вимогам до часової чутливості, мінімізації хибних спрацювань і безпечної обробки вхідних сигналів.

У межах поставленої мети та з урахуванням специфіки вбудованих систем безпеки, особливу увагу необхідно приділити технічним вимогам до охоронного модуля, який має функціонувати в умовах обмежених обчислювальних та енергетичних ресурсів. Вимоги формуються з урахуванням принципів енергетичної ефективності, часової реакції, точності виявлення подій, стійкості до зовнішніх перешкод і можливості інтеграції з додатковими модулями. Ці характеристики повинні бути закладені на етапі проєктування архітектури системи, вибору апаратної платформи, конфігурації програмного забезпечення та логіки обробки вхідних сигналів.

У таблиці 1.2 систематизовано основні технічні вимоги до функціональності, апаратного забезпечення та умов експлуатації прототипу охоронного модуля, що буде розроблений у межах цієї дипломної роботи.

Таблиця 1.2– Базові технічні вимоги до охоронного модуля на основі Arduino

№	Категорія	Вимога
1	Тип платформи	Arduino UNO/Nano (мікроконтролер ATmega328P)
2	Кількість керованих периферій	Не менше 5 (сенсори руху, RFID, GPS, GSM, сирена/реле)
3	Тип автентифікації	RFID із підтримкою UID-фільтрації
4	Методи комунікації	GSM (SIM800L/SIM900), Bluetooth (HC-05), UART
5	Моніторинг положення	GPS-модуль з точністю до 5 м (NEO-6M/NEO-M8N)
6	Час реакції на загрозу	< 500 мс від моменту спрацювання сенсора до активації сигналізації
7	Рівень енергоспоживання	Не більше 80 мА у активному режимі, < 10 мА у сплячому режимі

8	Протоколи взаємодії	UART для GPS, GSM; SPI для RFID; PWM для сервоприводу

Продовження таблиці 1.2

9	Спосіб подачі сигналу тривоги	Звуковий (п’єзо), візуальний (світлодіод), GSM-оповіщення
10	Умови експлуатації	Температурний діапазон від −20 до +60 °C, відносна вологість до 85%
11	Можливість розширення системи	Підтримка підключення нових модулів через I2C або UART
12	Захист конфігурацій	Зберігання UID та параметрів у EEPROM з підтримкою перезапису

Таким чином, сформовані технічні вимоги дозволяють забезпечити оптимальний баланс між функціональністю, автономністю та енергозалежністю системи. Врахування цих вимог на етапі розробки дозволить створити стійкий до зовнішніх впливів модуль, здатний не лише реагувати на загрози, а й адаптуватися до змін у середовищі використання. Окремо варто підкреслити, що закладена модульність системи дозволяє швидко масштабувати функціонал у подальших ітераціях проєкту без потреби повної перебудови апаратної або програмної частини.

РОЗДІЛ 2 ПРОЕКТУВАННЯ СПЕЦІАЛІЗОВАНОГО МОДУЛЯ ОХОРОНИ

2.1. Архітектура модуля охорони

Проєктування ефективного охоронного модуля для автомобіля базується на принципах модульності, автономності та адаптивності до зовнішніх загроз, що особливо актуально в умовах постійного зростання кількості кібер- і фізичних атак на транспортні засоби. Архітектура системи повинна забезпечувати не лише виявлення та реагування на спроби несанкціонованого доступу, але й підтримувати можливості масштабування функціональності, інтеграції нових компонентів та гнучкого налаштування логіки роботи відповідно до специфіки експлуатації.

Центральною ланкою розробленого охоронного модуля є мікроконтролер Arduino UNO, побудований на базі ATmega328P, який виконує функції керування всіма підсистемами, обробки вхідних даних від сенсорів та генерації керуючих сигналів для виконавчих елементів. Вибір цієї платформи обумовлений її відкритою архітектурою, низьким енергоспоживанням, сумісністю з численними периферійними модулями та широкою підтримкою бібліотек для реалізації алгоритмів моніторингу й сигналізації [28].

Архітектура системи передбачає використання комплексу сенсорних та комунікаційних модулів, зокрема:

- датчик руху PIR, який забезпечує контроль за вторгненням у салон автомобіля або наближенням до транспортного засобу. Цей сенсор генерує цифровий сигнал у разі фіксації руху, що слугує тригером для активації тривожного режиму.
- RFID-зчитувач RC522, який реалізує функцію автентифікації користувача на основі унікального ідентифікатора (UID) ключ-карти. Це дозволяє відрізнити легітимного користувача від потенційного зловмисника, мінімізуючи ризики несанкціонованого доступу.

– GSM-модуль SIM800L, відповідальний за передачу тривожних повідомлень у вигляді SMS через мобільну мережу, забезпечуючи оперативне інформування власника про виявлені загрози незалежно від його місцезнаходження.

– GPS-модуль NEO-6M, який дозволяє визначати координати транспортного засобу у разі викрадення або несанкціонованого переміщення, з подальшою передачею даних через GSM-канал.

– Сирена (п’єзоелемент) і світлодіодний індикатор, що виконують функції локальної сигналізації, забезпечуючи як акустичне, так і візуальне сповіщення про спробу злому.

– Реле блокування двигуна, яке забезпечує фізичне переривання ланцюга живлення стартера або паливної системи для унеможливлення запуску двигуна при спробі викрадення.

– EEPROM, вбудована в мікроконтролер, використовується для збереження конфігураційних параметрів системи, зокрема списку авторизованих UID та налаштувань режимів охорони, що гарантує збереження даних при відключенні живлення.

Взаємодія між компонентами здійснюється через стандартні інтерфейси зв’язку: SPI для RFID-зчитувача, UART для GPS- та GSM-модулів, а також цифрові GPIO-виходи для керування реле та сигналізацією. Такий підхід забезпечує простоту інтеграції компонентів і мінімізує використання апаратних ресурсів мікроконтролера.

Структурна схема архітектури модуля охорони представлена на рисунку 2.1. Дана UML-діаграма відображає основні апаратні компоненти системи, їхні функціональні взаємозв'язки та напрямки передачі даних між модулями.

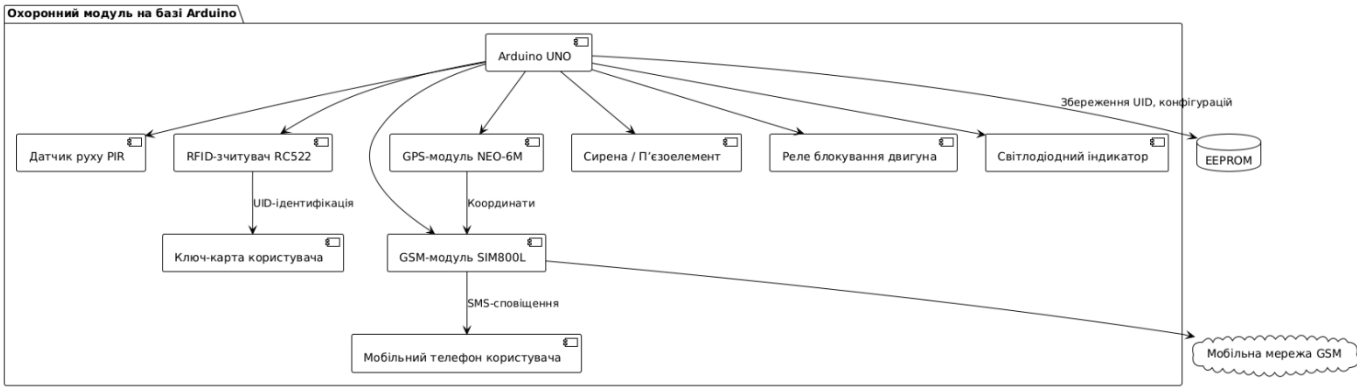

Рисунок 2.1 – Архітектура охоронного модуля на базі Arduino

Як видно з рисунка 2.1, Arduino UNO виконує роль центрального вузла, що координує роботу всіх периферійних пристроїв. Сенсори та виконавчі механізми безпосередньо підключені до мікроконтролера, забезпечуючи швидку реакцію на події. Передача тривожних сигналів та координат здійснюється через GSM-модуль, що взаємодіє із мобільною мережею для оповіщення користувача.

Важливою особливістю архітектури є її модульність, яка дозволяє легко інтегрувати додаткові функціональні блоки, наприклад, модулі Wi-Fi, датчики нахилу або біометричної автентифікації. Відкрите програмне середовище Arduino IDE забезпечує гнучкість у налаштуванні логіки обробки подій та адаптацію системи до конкретних сценаріїв використання [29].

2.2 Вибір компонентів: контролер, датчики, виконавчі елементи

Проєктування ефективного охоронного модуля для автомобіля вимагає ретельного підбору апаратних компонентів, які забезпечать стабільну роботу системи в умовах змінних зовнішніх факторів, обмежених енергетичних ресурсів та необхідності оперативного реагування на загрози. Основними критеріями вибору компонентів є: сумісність із мікроконтролером, енергоспоживання, надійність, чутливість сенсорів, простота інтеграції та можливість масштабування.

Центральним елементом системи обрано мікроконтролер Arduino UNO R3, який забезпечує керування всіма підсистемами, обробку сигналів від сенсорів та генерацію керуючих команд. Arduino UNO вирізняється відкритою архітектурою, широкою підтримкою бібліотек і достатньою кількістю цифрових та аналогових входів/виходів для підключення периферії [29]. Зображення контролера наведено на рисунку 2.2.

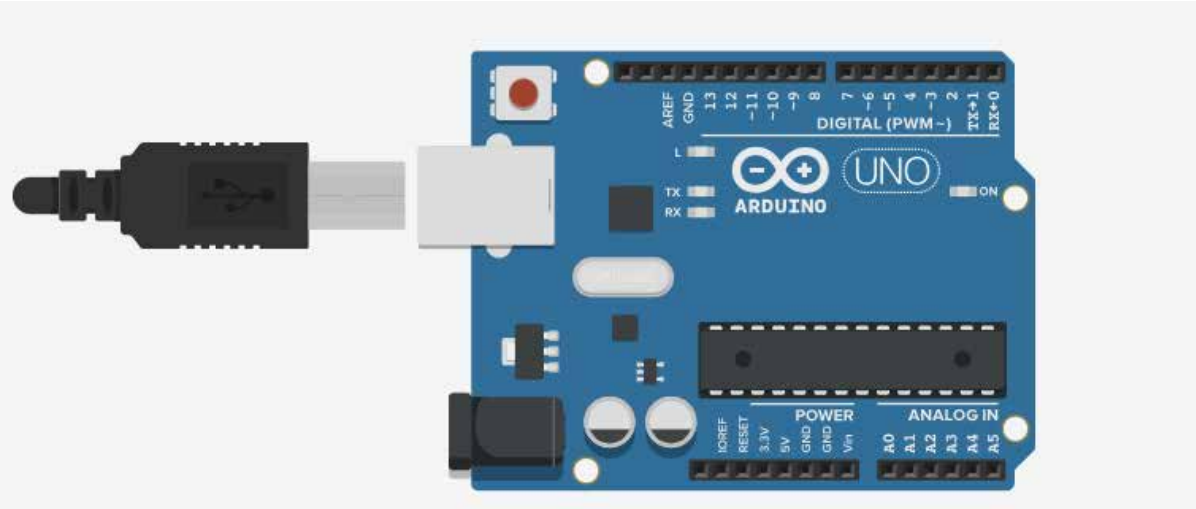

Рисунок 2.2 – Мікроконтролер Arduino UNO R3

Для виявлення руху в зоні контролю використовується PIR-датчик (Passive Infrared Sensor), що фіксує теплове випромінювання об'єктів. Цей сенсор є одним із ключових елементів системи для визначення вторгнень у салон або наближення до автомобіля. PIR-сенсор характеризується низьким енергоспоживанням і високою чутливістю, що робить його оптимальним для автономних систем безпеки [30]. Зображення наведено на рисунку 2.3.

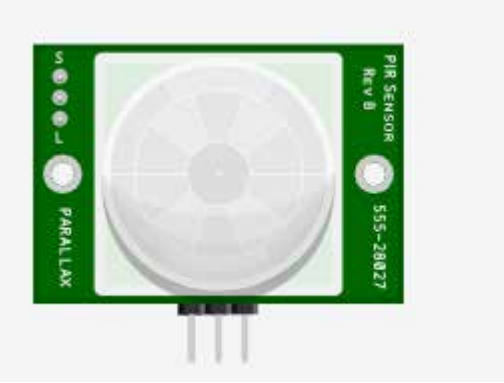

Рисунок 2.3 – PIR-датчик руху

Акустична сигналізація реалізується за допомогою п’єзоелемента (Piezo Buzzer), який слугує виконавчим елементом для подачі звукових сигналів у разі спрацювання тривоги. П’єзоелементи мають компактні розміри та просте підключення, що дозволяє ефективно інтегрувати їх у вбудовані системи (рисунок 2.4) [31].

Рисунок 2.4 – П’єзоелемент для звукової сигналізації

Для контролю змін положення автомобіля, наприклад, при спробі евакуації або буксирування, використовується датчик нахилу SW-200D. Це механічний сенсор, який реагує на зміну просторового положення. Завдяки простоті конструкції та високій надійності, такі датчики широко застосовуються в охоронних системах (рисунок 2.5) [32].

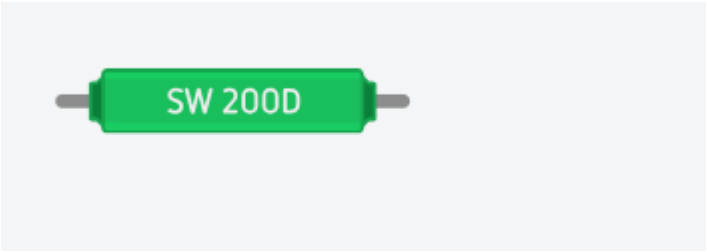

Рисунок 2.5 – Датчик нахилу SW-200D

Джерелом живлення для автономної роботи модуля обрано стандартну батарею на 9V, що дозволяє забезпечити мобільність системи та мінімізувати залежність від бортової мережі транспортного засобу (рисунок 2.6).

Рисунок 2.6 – Джерело живлення: батарея 9V

Для додаткового контролю наближення об’єктів або виявлення руху у зовнішній зоні застосовується ультразвуковий датчик відстані HC-SR04. Цей сенсор генерує ультразвукові імпульси і вимірює час їх відбиття від об’єкта, що дозволяє визначати дистанцію до перешкоди (рисунок 2.7) [33].

Рисунок 2.7 – Ультразвуковий датчик відстані HC-SR04

Світлова індикація стану системи реалізована за допомогою стандартного світлодіода (LED) червоного кольору, який сигналізує про активний режим охорони або спрацювання тривоги (рисунок 2.8). Для забезпечення візуального контролю світлодіоди є найбільш енергоефективним рішенням [34].

Рисунок 2.8 – Світлодіод для індикації стану системи

Фізичне блокування систем автомобіля здійснюється через реле SPDT, яке дозволяє розривати ланцюги живлення критичних елементів, таких як стартер

або паливний насос (рисунок 2.9). Реле забезпечує електричну ізоляцію між керуючим сигналом мікроконтролера та високострумовим навантаженням [35].

Рисунок 2.9 – Реле SPDT для блокування систем автомобіля

Для забезпечення додаткової тактильності або прихованої індикації в конструкцію може бути інтегрований вібраційний мотор, який активується при спрацюванні тривоги (рисунок 2.10). Цей компонент є опціональним і може використовуватись у специфічних сценаріях [36].

Рисунок 2.10 – Вібраційний мотор як виконавчий елемент

Додатковим елементом керування є кнопка (Pushbutton), яка використовується для активації або деактивації охоронного режиму під час тестування, а також для скидання тривожних подій. Це простий цифровий вхідний елемент, який забезпечує базову взаємодію користувача з системою (рисунок 2.11) [37].

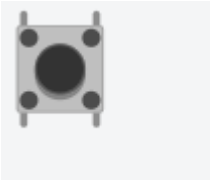

Рисунок 2.11 – Кнопка (Pushbutton) для керування режимами

Для налаштування порогових значень чутливості сенсорів або регулювання інтенсивності сигналізації використовується потенціометр. Це

змінний резистор, який дозволяє гнучко керувати аналоговими параметрами системи без потреби перепрограмування (рисунок 2.12) [38].

Рисунок 2.12 – Потенціометр для регулювання чутливості

Для контролю електричних параметрів під час розробки та тестування системи залучено базовий вимірювальний комплект, що включає джерело живлення, мультиметри та резистори для формування контрольних ланцюгів. Це дозволяє перевіряти стабільність живлення, рівні сигналів та споживання струму окремими компонентами (рисунок 2.13).

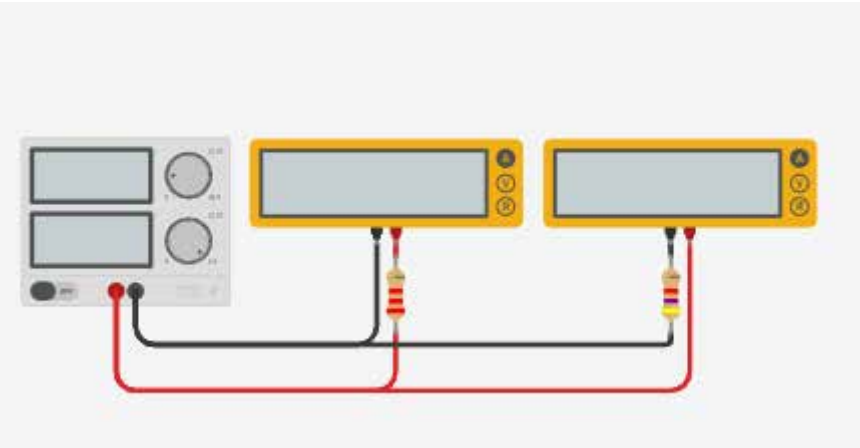

Рисунок 2.13 – Вимірювальне обладнання для тестування електричних параметрів

З’єднання всіх компонентів системи здійснюється на макетній платі (Breadboard), що забезпечує швидке та гнучке прототипування без пайки. Breadboard дозволяє легко модифікувати схему, додавати або замінювати елементи в процесі налагодження (рисунок 2.14) [39].

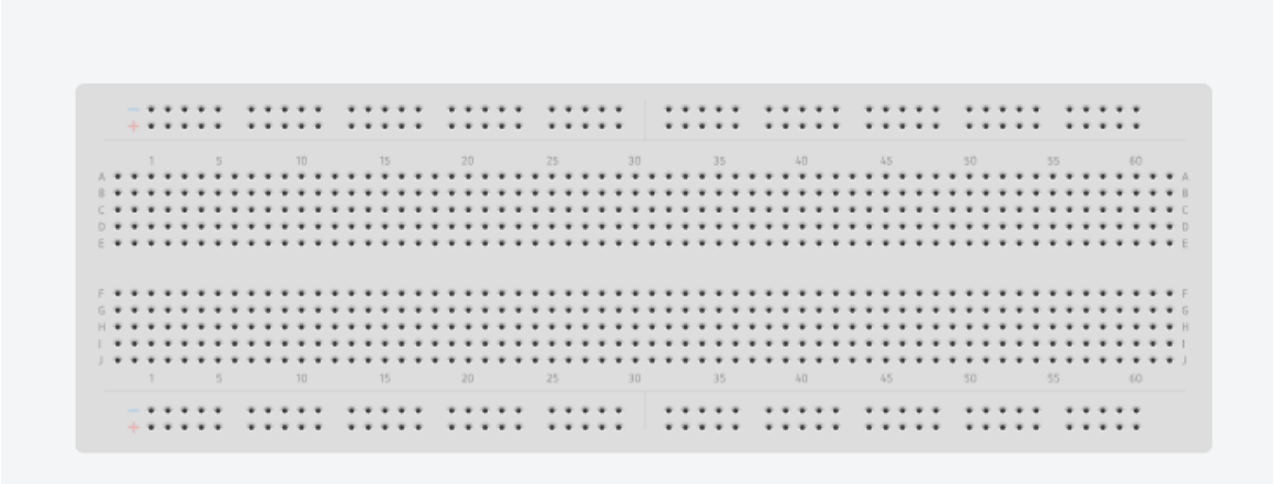

Рисунок 2.14 – Макетна плата (Breadboard) для збирання схеми

Обраний набір компонентів відповідає вимогам до створення прототипу охоронного модуля: забезпечує базову функціональність, можливість тестування, налаштування параметрів системи та подальшу адаптацію під специфіку використання. Застосування стандартних елементів Arduino-екосистеми дозволяє гарантувати сумісність і спрощує розробку програмного забезпечення.

2.3 Розробка структурної та принципової схем

Розробка охоронного модуля для автомобіля передбачає побудову як структурної, так і принципової електричної схеми. Структурна схема дозволяє на концептуальному рівні відобразити основні функціональні блоки системи та їх взаємодію, тоді як принципова схема деталізує електричні підключення між компонентами, визначаючи логіку з’єднання та характер сигналів.

Структурна схема охоронного модуля побудована з урахуванням функціонального розподілу підсистем на блоки виявлення загроз, блоки індикації та блоки активної протидії. Центральною ланкою системи виступає мікроконтролер Arduino UNO, який обробляє сигнали з сенсорів, генерує керуючі сигнали для виконавчих елементів і забезпечує взаємодію з

користувачем. На вхід Arduino UNO подаються сигнали від PIR-датчика для виявлення руху, ультразвукового датчика відстані для контролю наближення, а також датчика нахилу SW-200D для фіксації змін положення автомобіля. Керуючі сигнали формуються для модуля реле, який блокує критичні системи авто, світлодіодної індикації та звукової сигналізації. Також передбачено взаємодію із кнопкою для перемикання режимів та потенціометром для регулювання параметрів (рисунок 2.15).

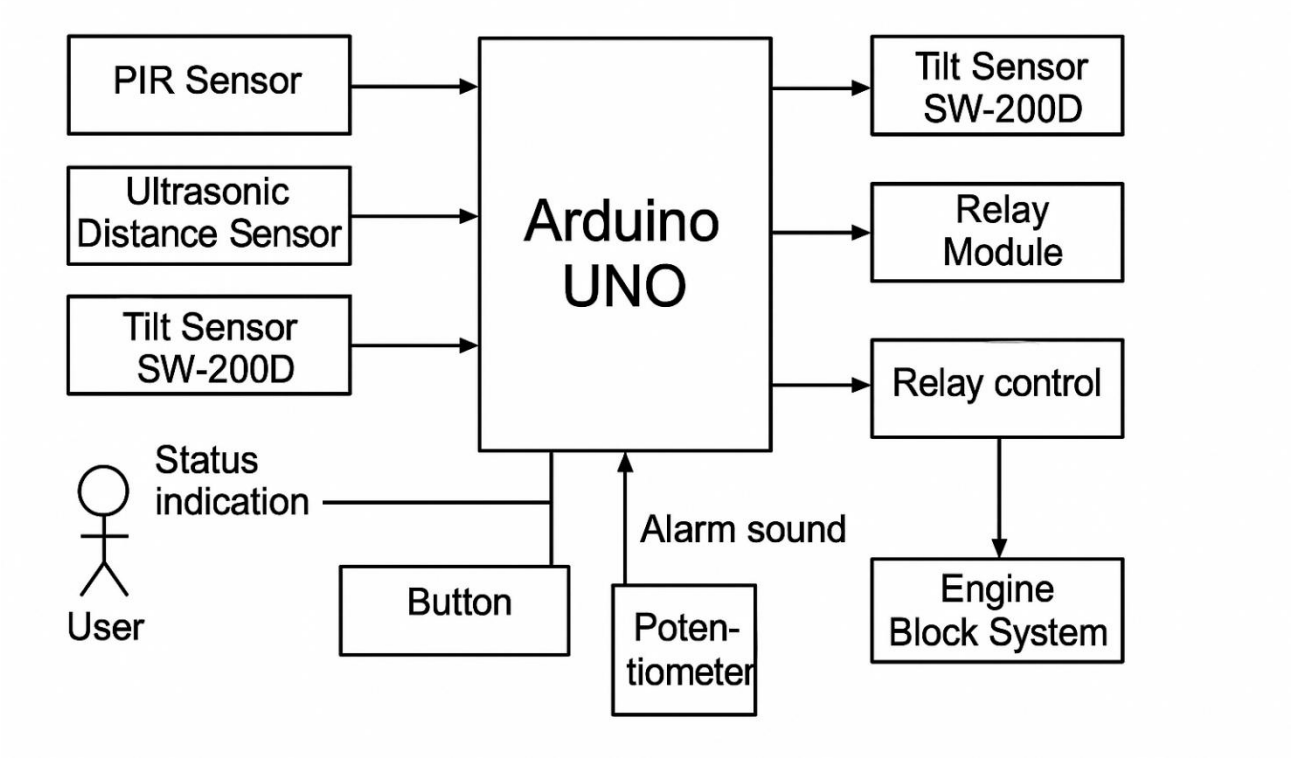

Рисунок 2.15 – Структурна схема охоронного модуля

Принципова електрична схема деталізує з'єднання між компонентами системи, враховуючи харчування, сигнальні з'єднання і керуючі ланцюги. Подача живлення забезпечується батареєю на 9V, яка через регулятор напруги живить Arduino та периферійні пристрої. Сенсори (ультразвуковий, PIR і датчик нахилу) підключаються до цифрових входів мікроконтролера. Управління реле здійснюється через транзисторний ключ для забезпечення електричної ізоляції і належного струмового навантаження. Світлодіоди для індикації стану підключені через обмежувальні резистори на 220 Ом. Звукова сигналізація реалізована на базі п'єзоелемента, який керується окремим цифровим виходом.

Додатково підключено кнопку та потенціометр, які служать для ручного керування і налаштувань відповідно (рисунок 2.16).

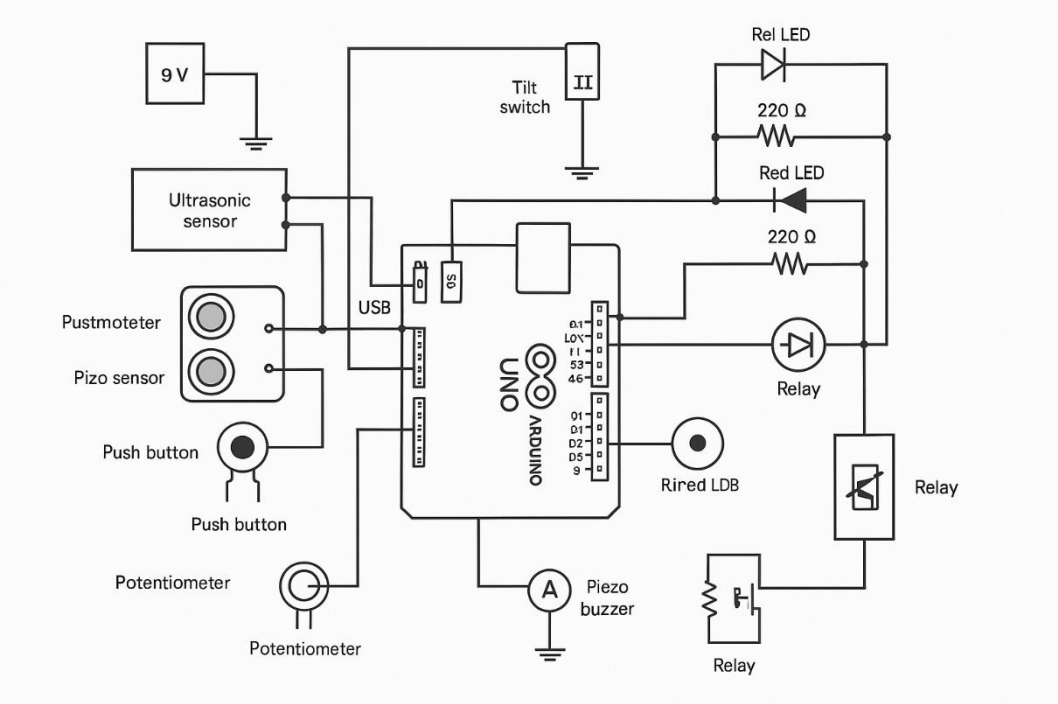

Рисунок 2.16 – Принципова електрична схема охоронного модуля

У побудованій схемі передбачено базову захистну логіку: при спрацюванні будь-якого із сенсорів система активує світлову та звукову сигналізацію, одночасно розмикаючи реле для блокування системи запуску двигуна. Такий підхід забезпечує комплексну багаторівневу охорону транспортного засобу, об'єднуючи виявлення загроз, їхню сигналізацію та фізичну протидію спробам викрадення.

2.4 Побудова алгоритму функціонування системи

Для забезпечення коректного функціонування охоронного модуля на базі мікроконтролера Arduino було розроблено алгоритм роботи системи, який реалізує логіку реагування на загрози, виявлені різними сенсорами, і забезпечує активацію відповідних виконавчих механізмів. Основними вимогами до алгоритму є: безперервний моніторинг стану датчиків, мінімізація помилкових

спрацювань, можливість скидання тривоги користувачем і відновлення вихідного стану системи.

Процес функціонування модуля розпочинається з ініціалізації всіх компонентів, включаючи цифрові входи/виходи, внутрішні таймери та послідовності очікування стабілізації напруги. Далі система перевіряє наявність живлення. У разі його відсутності активується режим очікування. При виявленні стабільного джерела живлення активується режим охорони.

У фазі моніторингу система зчитує дані з таких сенсорів:

- PIR-сенсор, що реагує на рух в межах контрольованої зони.
- Датчик нахилу SW-200D, який фіксує зміну положення автомобіля (наприклад, при спробі евакуації).
- Ультразвуковий датчик відстані, що дозволяє виявити наближення об’єкта до автомобіля.

У випадку спрацювання одного з сенсорів система переходить у тривожний режим. Зокрема, активується звукова сигналізація (п’єзоелемент), вмикається світлова індикація (червоний світлодіод), і, що критично важливо, реле блокує систему запуску двигуна, унеможливлюючи несанкціоноване переміщення транспортного засобу.

Для запобігання помилковим спрацюванням ультразвукового датчика реалізовано механізм підтвердження наближення об'єкта з урахуванням тривалості сигналу, що перевищує заданий поріг. Це дозволяє уникнути активації тривоги через короткочасні рухи або дрібні перешкоди.

Скидання тривожного стану передбачене через натискання кнопки користувачем. У цьому випадку сирена вимикається, реле розблокується, а світлова індикація повертається до початкового стану. Таким чином, алгоритм підтримує замкнутий цикл — після завершення тривожного стану система повертається до фази моніторингу.

Логіка алгоритму візуалізована на діаграмі активностей UML, представленій на рисунку 2.17. На діаграмі показано послідовність подій, логічні розгалуження, дії користувача та переходи між різними станами системи.

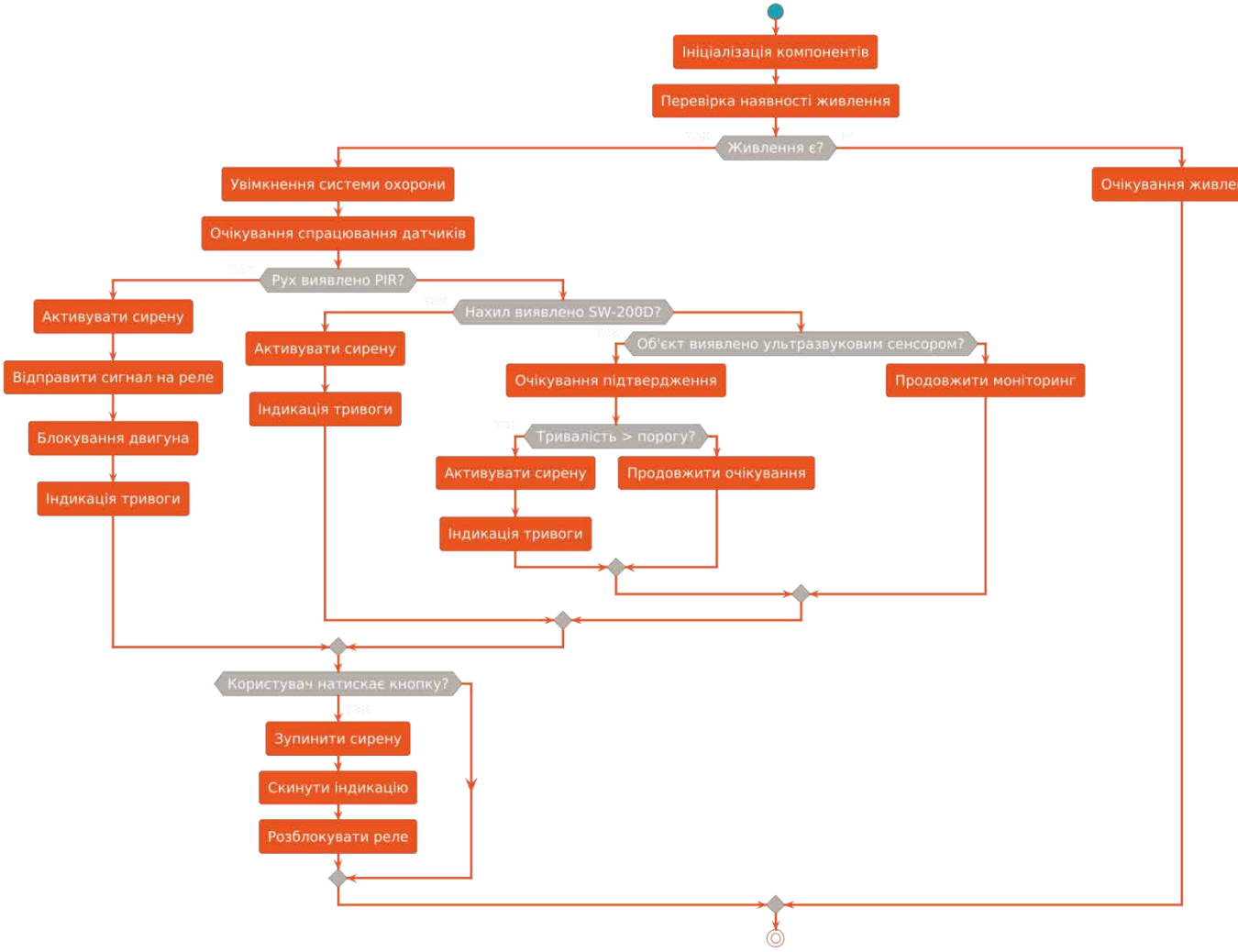

Рисунок 2.17 – Алгоритм функціонування охоронного модуля (UML-діаграма активностей)

Реалізація такого алгоритму дозволяє забезпечити комплексну реакцію системи на потенційні загрози та підвищує рівень безпеки автомобіля. Наявність багаторівневої логіки контролю, верифікації і користувацького втручання робить систему надійною, адаптивною та зручною в експлуатації.

2.5 Опис електричних та інформаційних інтерфейсів

Проєктована система охоронного модуля базується на організації коректних електричних та інформаційних інтерфейсів, які забезпечують взаємодію між сенсорними, обчислювальними та виконавчими компонентами. Електричні інтерфейси системи включають підключення живлення, сигнальні з'єднання та елементи комутації керуючих сигналів, тоді як інформаційні інтерфейси відповідають за обробку і передачу даних між пристроями.

Параметри основних електричних підключень представлено у таблиці 2.1.

Таблиця 2.1 – Параметри електричних підключень компонентів системи

Компонент	Підключення до Arduino	Напруга живлення	Опис функції
PIR-сенсор	D2	5V	Виявлення руху
Ультразвуковий сенсор	Trig - D6, Echo - D7	5V	Виявлення наближення об'єктів
Датчик нахилу SW-200D	D9	5V	Виявлення змін положення автомобіля
Світлодіод індикації	D8 через резистор 220 Ом	5V	Стан охорони та тривоги
П'єзоелемент	D4	5V	Звукова сигналізація тривоги
Реле	D3 через транзисторний ключ	5V	Блокування системи двигуна
Кнопка користувача	D5	5V	Вимкнення сирени та розблокування
Потенціометр	A0	5V	Регулювання порогу чутливості

Живлення системи здійснюється від джерела на 9V, яке через стабілізатор напруги забезпечує подачу 5V до мікроконтролера та периферійних пристроїв. Всі цифрові сигнали в системі працюють у стандартному рівні логіки TTL (0–5V). Для захисту вихідних каналів використовуються обмежувальні резистори у ланцюгах світлодіодів та транзисторний ключ для керування реле. Застосування таких елементів дозволяє уникнути перевантаження пінів Arduino і забезпечує стабільність роботи навіть при імпульсних навантаженнях.

Щодо інформаційних інтерфейсів, система реалізує пряме оброблення цифрових та аналогових сигналів від сенсорних пристроїв. PIR-сенсор передає цифровий сигнал логічного високого рівня у разі фіксації руху. Ультразвуковий сенсор працює у двоканальному режимі: лінія Trig генерує імпульс, а лінія Echo приймає відбитий сигнал і вимірює його тривалість. Датчик нахилу подає на Arduino цифровий сигнал, що відображає зміну просторового положення автомобіля.

Користувацький інтерфейс представлений кнопкою управління, яка дозволяє вручну скинути тривожний стан, та світлодіодною і звуковою індикацією, що надають інформацію про поточний режим роботи системи. Усі події обробляються у основному циклі програми Arduino через виклик відповідних функцій обробки переривань або опитування стану входів у режимі реального часу.

Побудова електричних та інформаційних інтерфейсів у проєктованій системі базується на забезпеченні надійної комутації живлення, ефективної передачі даних і організації зручної взаємодії між усіма модулями з урахуванням обмежених ресурсів мікроконтролера та вимог до енергетичної ефективності.

РОЗДІЛ 3 ПРОГРАМНА ІМПЛЕМЕНТАЦІЯ ФУНКЦІЙ ОХОРОНИ

3.1 Огляд середовища розробки Arduino IDE

Середовище розробки Arduino IDE є ключовим інструментом для програмування вбудованих систем на базі мікроконтролерів типу AVR (наприклад, ATmega328P) та ARM. Його архітектура побудована на основі Java та інтегрує в собі редактор коду, компілятор, інструмент завантаження прошивки у пристрій, монітор послідовного порту й підтримку сторонніх бібліотек. Основною перевагою Arduino IDE є її відкритість, сумісність з широким спектром апаратних платформ та наявність великої спільноти підтримки.

На рисунку 3.1 представлено стандартне графічне середовище Arduino IDE під час створення елементарної програми з використанням мови C++ для миготіння вбудованого світлодіода.

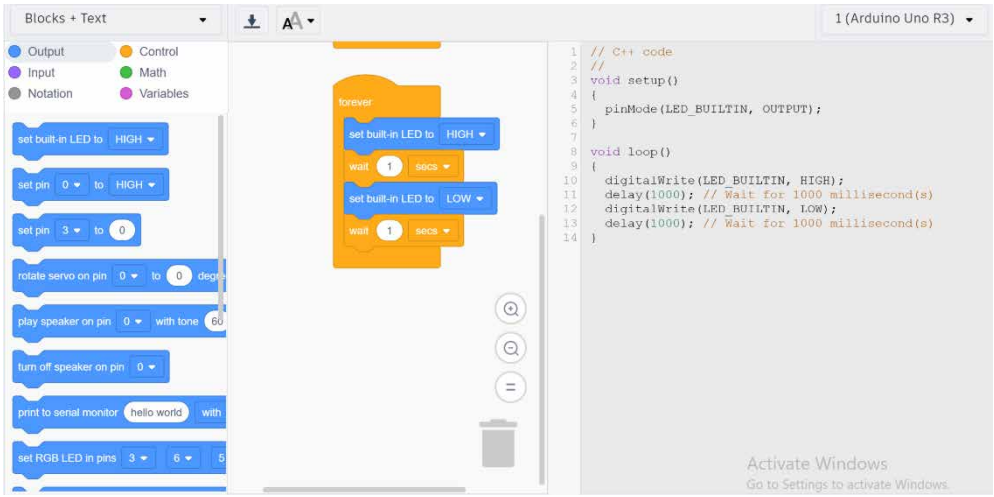

Рисунок 3.1 – Графічне середовище Arduino IDE з прикладом коду миготіння світлодіода та блоковим редактором

Праворуч відображено текстовий редактор, у якому реалізовано цикл setup() для ініціалізації порту виводу та нескінченний цикл loop(), що реалізує функцію чергування логічних рівнів (HIGH/LOW) з затримкою у 1000 мс. Також видно режим візуального програмування (блоки), що значно спрощує процес навчання та прототипування для новачків (рис. 3.1).

Однією з характерних функціональних переваг Arduino IDE є підтримка бібліотек, які дозволяють інтегрувати різноманітні датчики та виконавчі модулі, такі як PIR, ультразвукові сенсори, GSM-модулі, реле тощо. Завдяки цьому інструмент стає універсальним для реалізації різнопланових проєктів, зокрема і в контексті побудови охоронних систем.

Розроблений прототип охоронного модуля, що представлений на віртуальному макеті у TinkerCAD (рис. 3.2), було протестовано у середовищі Arduino IDE. Цей макет включає типові апаратні елементи системи: мікроконтролер Arduino UNO, PIR-сенсор, ультразвуковий датчик HC-SR04, реле, світлодіод, п’єзодинамік, датчик нахилу SW-200D, кнопку керування, потенціометр та живлення 9V.

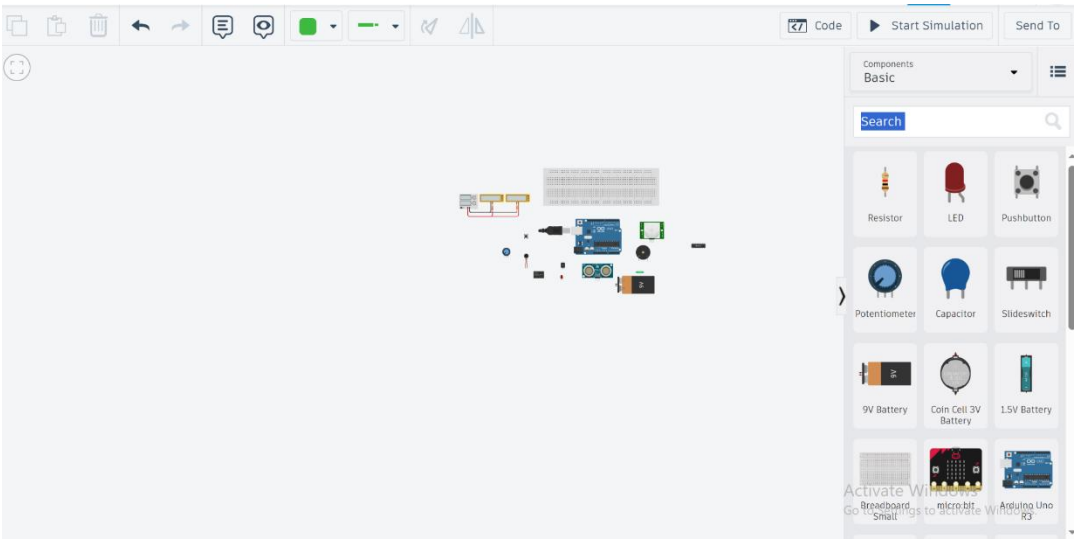

Рисунок 3.2 – Віртуальна апаратна конфігурація системи безпеки у середовищі TinkerCAD

Розробка логіки обробки подій, генерації тривог та управління виконавчими пристроями виконується у текстовому редакторі Arduino IDE (рис. 3.1). При цьому можливе використання як класичного синтаксису C++, так і візуального моделювання блоками для швидкого формування структур умовних операторів, циклів та логіки обробки сигналів. Це дозволяє реалізувати як початкову верифікацію роботи модуля, так і повноцінне програмне забезпечення системи охорони з мінімальними вимогами до середовища.

Крім стандартного Arduino IDE, у процесі розробки може використовуватися TinkerCAD Circuits, що забезпечує симуляцію роботи електричних схем і дозволяє в реальному часі моделювати логіку спрацювання сенсорів та виводів (рис. 3.3). Структурна схема з використанням Arduino UNO, PIR-сенсора, реле, п’єзодинаміка та індикаторів демонструє логічну побудову системи реагування на вторгнення.

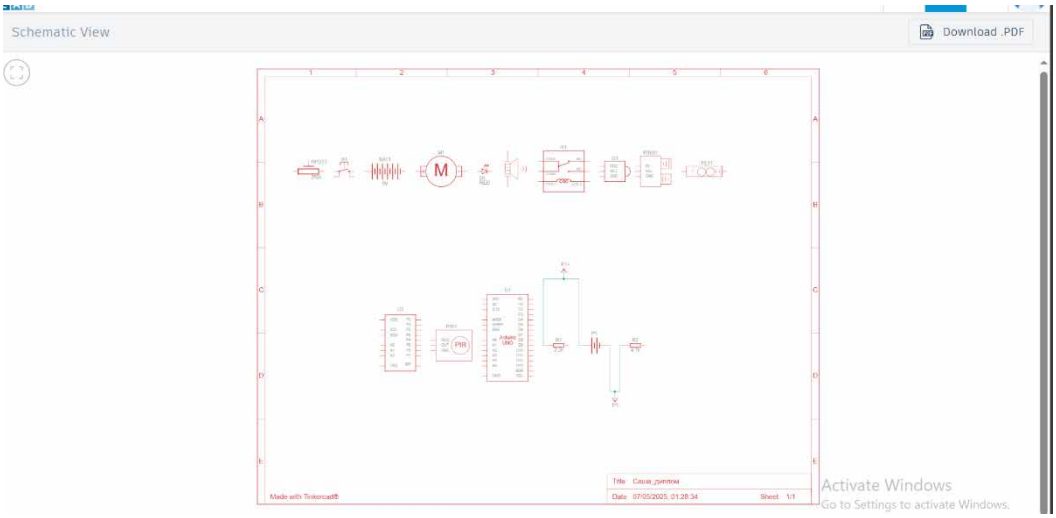

Рисунок 3.3 – Принципова схема охоронного модуля в TinkerCAD Schematic View

Компоненти, використані у схемі, систематизовано у таблиці компонентів (рис. 3.4). До основних елементів належать Arduino Uno R3, PIR-сенсор, нахильний сенсор (Tilt), ультразвуковий датчик, реле, інфрачервоний сенсор, п’єзоелемент, світлодіод, вібромотор та джерело живлення.

Component List — Download CSV

Name	Quantity	Component
U1	1	Arduino Uno R3
PIR1	1	PIR Sensor
U2	1	8-port I2C expander
TILT1	1	Tilt Sensor
PING1	1	Ultrasonic Distance Sensor
U3	1	IR sensor
K1	1	Relay SPDT
PIEZO1	1	Piezo
D1	1	Red LED
M1	1	Vibration Motor
BAT1	1	9V Battery

Рисунок 3.4 – Таблиця компонентів системи охоронного модуля, сформована у TinkerCAD

Розробка охоронного модуля на базі Arduino передбачає поетапне виконання комплексу дій, кожна з яких спрямована на формування повноцінного вбудованого рішення, здатного виявляти загрози та реагувати на них у реальному часі. Планування процесу розробки є критично важливим для забезпечення узгодженості між апаратними та програмними складовими, дотримання вимог до енергоспоживання, швидкодії та структурної модульності системи.

У таблиці 3.1 подано структуровану послідовність етапів розробки охоронного модуля з коротким описом цілей, інструментів і результатів кожного етапу.

Таблиця 3.1 – Етапи реалізації охоронного модуля на основі Arduino

№	Етап розробки	Опис завдань та очікувані результати
1	Вибір середовища розробки	Аналіз функціональних можливостей Arduino IDE, підготовка бібліотек, налаштування параметрів компіляції та серійного моніторингу
2	Створення попередньої схеми	Побудова структурної та електричної схеми модуля в середовищі TinkerCAD або Fritzing для віртуальної перевірки логіки з'єднань
3	Вибір компонентної бази	Визначення та підбір сумісних сенсорів (PIR, ультразвук, нахил), виконавчих пристроїв (реле, сирена) і плати Arduino UNO
4	Моделювання системи	Побудова віртуального макету на breadboard у симуляторі для аналізу реакції системи на сигнали вхідних сенсорів
5	Розробка програмної логіки	Написання функціонального коду на мові C++ у середовищі Arduino IDE з урахуванням обробки переривань, логіки активації та блокування
6	Інтеграція інтерфейсів взаємодії	Налагодження інформаційного обміну з модулями: UART (GSM, GPS), SPI (RFID), GPIO (датчики, світлодіоди, кнопки)
7	Початкове тестування в емуляторі	Верифікація алгоритмів віртуально: перевірка послідовності дій системи при активації тривоги та логіки скидання

8	Завантаження на фізичний пристрій	Передача прошивки через USB-з'єднання на Arduino UNO, ініціалізація апаратних таймерів, перевірка живлення та підключення

Продовження таблиці 3.1

9	Перевірка працездатності	Проведення натурного тестування у статичних та динамічних умовах, аналіз реакцій на різні сценарії загроз
10	Масштабування функціональності	Оптимізація коду, додавання підтримки додаткових сенсорів (наприклад, температури, вібрацій, біометрії), впровадження шифрування або логування

Запропонована послідовність етапів дозволяє сформувати охоронну систему з повним циклом розробки — від архітектурного моделювання до експлуатаційної перевірки. Такий підхід забезпечує контроль на кожному етапі реалізації, дозволяє оперативно виявляти помилки та підвищує загальну надійність рішення.

Варто відзначити, що симуляційне моделювання у середовищі TinkerCAD Circuits використовується як проміжний етап перед натурною перевіркою (див. рис. 3.2), що дозволяє без ризику для фізичних компонентів перевірити логіку обробки сигналів, керування реле та сигналізацією. Паралельно з цим у середовищі Arduino IDE відбувається реалізація основного алгоритму роботи системи охорони з використанням бібліотек для роботи з портами вводу/виводу, таймерами, UART-протоколами та EEPROM (рис. 3.1).

У межах наступних підпунктів розділу буде реалізовано опис створеної програмної логіки, механізмів керування периферією та сценаріїв реагування, що відповідають поставленим технічним вимогам.

3.2 Опис програмного забезпечення модуля

Програмне забезпечення охоронного модуля реалізовано мовою C++ у середовищі Arduino IDE із застосуванням процедурної моделі управління подіями та сигналами. Основна функціональність модуля полягає в моніторингу вхідних сигналів від сенсорів, обробці логіки порушення, генерації відповідної тривожної реакції та управлінні виконавчими елементами.

Архітектура програмного забезпечення враховує модульний підхід до обробки даних із різнорідних сенсорів: пасивного інфрачервоного (PIR), ультразвукового (HC-SR04), датчика нахилу (Tilt), аналогового звукового сенсора, потенціометра та кнопки. Кожен сенсор підключено до окремого цифрового або аналогового входу мікроконтролера Arduino UNO R3. Обробка сигналів виконується в основному циклі програми (loop()), де кожен сенсор перевіряється з урахуванням його порогових значень або умов спрацювання.

Під час ініціалізації (setup()) конфігуруються порти, встановлюються режими роботи пінів (вхід/вихід), активуються серійний монітор та бібліотеки для роботи з зовнішніми компонентами. Для цифрових входів типу PIR, Tilt, Button застосовується пряме читання рівня сигналу через функцію digitalRead(). Аналогові сенсори (Potentiometer, Sound Sensor) обробляються через analogRead(), з подальшим порівнянням значень із заданими межами чутливості.

Активація тривожних режимів реалізована через включення реле (digitalWrite()), запуск п’єзодинаміка, індикацію на світлодіоді або через розширювач I2C (PCF8574), до якого можуть бути підключені додаткові сигнальні пристрої. Програмна логіка дозволяє гнучко реагувати на кілька одночасних загроз або сценаріїв дій, заздалегідь визначених у прошивці.

На рис. 3.5 представлено UML-діаграму компонентів розробленої системи, яка ілюструє логічні зв’язки між центральним контролером (Arduino UNO R3), сенсорною підсистемою, виконавчими пристроями та допоміжними елементами інфраструктури. Вказано типи підключення (цифровий вхід/вихід, аналоговий

вхід, інтерфейс I2C), що відповідає структурі програмного коду та конфігурації портів.

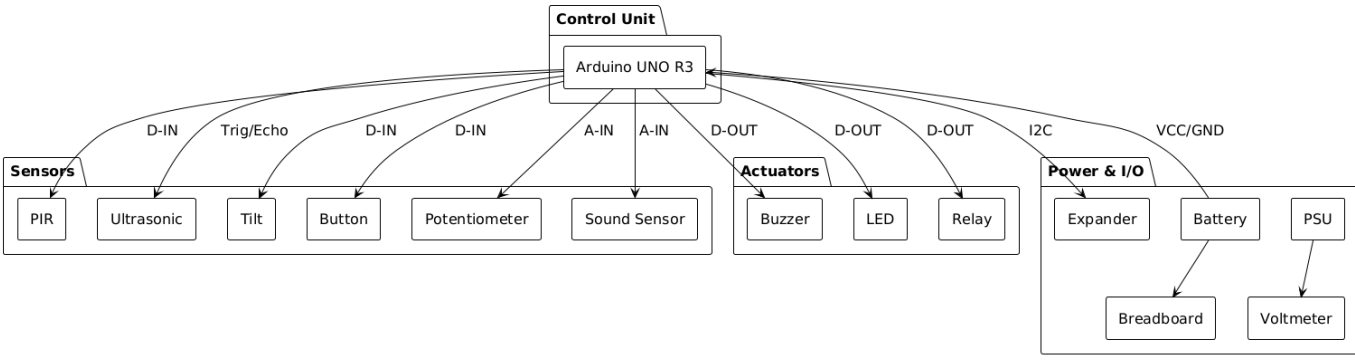

Рисунок 3.5 – UML-діаграма компонентів охоронного модуля на основі Arduino UNO

Логіка реалізації також враховує оптимізацію за часом реакції на події, що досягається через відсутність блокуючих функцій типу delay() у критичних ділянках коду. Замість цього застосовується контроль часу з використанням функції millis(), що дозволяє реалізувати псевдопаралелізм.

Код структуровано з урахуванням майбутньої масштабованості: реалізовано окремі блоки підключення сенсорів, функції для виконавчих дій, а також логіка пріоритетності у разі виникнення кількох одночасних спрацювань.

Розроблене програмне забезпечення забезпечує повноцінну інтеграцію всіх фізичних компонентів системи охорони, гарантує реагування на події в реальному часі та є базою для подальшого розширення функціональності у рамках захищених вбудованих рішень.

3.3 Реалізація функцій контролю та сигналізації

Система контролю та сигналізації в охоронному модулі побудована на основі мікроконтролера Arduino UNO R3, який взаємодіє з датчиками руху, нахилу та звуку, а також керує виконавчими пристроями – світлодіодом, реле та

звуковим сигналізатором. Архітектура логіки реагування забезпечує виявлення несанкціонованих дій та негайну активацію тривожної сигналізації.

На **рис. 3.6** подано фрагмент початкової ініціалізації системи, де конфігуруються цифрові порти введення та виведення для роботи з сенсорами й пристроями сигналізації. Зокрема, порти D2 та D3 налаштовано на прийом даних від сенсорів руху (PIR) і нахилу, тоді як порти D4–D6 використовуються для активації LED, реле та бузера відповідно.

```
void setup() {
  pinMode(2, INPUT);    // PIR sensor
  pinMode(3, INPUT);    // Tilt sensor
  pinMode(4, OUTPUT);   // LED
  pinMode(5, OUTPUT);   // Relay
  pinMode(6, OUTPUT);   // Buzzer
  Serial.begin(9600);
}
```

Рисунок 3.6 – Фрагмент ініціалізації цифрових портів охоронного модуля

Логіка роботи модуля передбачає постійний моніторинг станів сенсорів. У разі виявлення активного сигналу з одного з датчиків (руху або нахилу), мікроконтролер миттєво подає керуючі сигнали на виконавчі пристрої, активуючи тривогу. На **рис. 3.7** наведено реалізацію основного циклу loop(), що відповідає за аналіз подій та відповідну реакцію.

```
void loop() {
  int motion = digitalRead(2);
  int tilt = digitalRead(3);

  if (motion == HIGH || tilt == HIGH) {
    digitalWrite(4, HIGH); // LED ON
    digitalWrite(5, HIGH); // Relay ON
    tone(6, 1000);          // Buzzer ON
    Serial.println("Alert: Motion or Tilt Detected!");
  } else {
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    noTone(6);              // Buzzer OFF
  }

  delay(200);
}
```

Рисунок 3.7 – Реалізація основного циклу loop()

У модулі також реалізовано функцію контролю акустичних змін середовища за допомогою аналогового мікрофонного сенсора. Зчитування рівня шуму здійснюється через порт A0, а спрацювання тривоги визначається перевищенням встановленого порогу, фрагмент лістинінгу коду представлений на рис.3.8.

```
int soundLevel = analogRead(A0);
int threshold = 500;

if (soundLevel > threshold) {
  digitalWrite(4, HIGH); // LED ON
  tone(6, 1500);         // Buzzer high tone
  Serial.println("Sound alert triggered");
}
```

Рисунок 3.8 – Фрагмент коду контролю акустичних змін

Така реалізація дозволяє охопити кілька каналів виявлення загроз, підвищуючи загальну надійність модуля. Для забезпечення зрозумілого узагальнення конфігурації портів було складено таблицю 3.2, яка містить відповідність між цифровими/аналоговими входами Arduino та призначенням кожного з них у рамках охоронної системи.

Таблиця 3.2 – Відповідність портів Arduino функціональним елементам системи

№	Порт Arduino	Тип сигналу	Підключений компонент	Призначення
1	D2	Digital IN	PIR Sensor	Виявлення руху
2	D3	Digital IN	Tilt Sensor	Виявлення нахилу
3	D4	Digital OUT	LED	Візуальна індикація тривоги
4	D5	Digital OUT	Relay Module	Активація зовнішнього пристрою
5	D6	Digital OUT	Piezo Buzzer	Звукова сигналізація
6	A0	Analog IN	Sound Sensor	Акустичний моніторинг

Завдяки модульності реалізації, система може бути легко доповнена новими пристроями або переформатована під інші сценарії без потреби

змінювати загальну архітектуру. Такий підхід забезпечує масштабованість і адаптивність рішення.

4 ПРОВЕДЕННЯ ТЕСТУВАННЯ, ВИРІШЕННЯ ПИТАННЯ ЕФЕКТИВНОСТІ І НАДІЙОСТІ СИСТЕМИ

4.1 Проведення модульного та системного тестування

Після завершення етапів проєктування та реалізації охоронної системи було проведено комплексне тестування, що включало модульну перевірку окремих функціональних блоків та повне системне випробування роботи всієї структури. Основна мета тестування — підтвердити відповідність роботи системи функціональним вимогам, визначеним на етапі проєктування, та забезпечити її стійкість до типових загроз безпеки, таких як несанкціоноване відкриття дверей, виявлення руху поблизу транспортного засобу чи спроба евакуації.

Згідно з технічним завданням було сформовано перелік тестових сценаріїв, спрямованих на перевірку основних функціональних можливостей охоронного модуля. У таблиці 4.1 наведено ключові етапи модульного та системного тестування.

Таблиця 4.1 – Заплановані сценарії модульного та системного тестування охоронної системи

№	Компонент/модуль	Умова тесту	Очікуваний результат
1	Датчик відкриття дверей	Зміна аналогової напруги на вході A0/A1	Оновлення змінної estporta/estportaB, спрацювання тривоги
2	PIR-сенсор	Виявлення руху перед транспортним засобом	move = HIGH, активація сирени та світлодіодів
3	Датчик нахилу (Tilt)	Зміна просторового положення кузова	Встановлення логіки тривоги, запуск блокування
4	Кнопка bot	Натискання при замкнених дверях	Увімкнення охоронного режиму, блимання фар, звук

5	Кнопка bot2	Натискання під час тривоги	Вимкнення охорони, зупинка сигналів

Продовження таблиці 4.1.

6	Світлодіоди	Вихід digitalWrite() в режимі HIGH/LOW	Правильне блимання, індикація стану
7	Реле блокування	Подання сигналу на транзисторний ключ	Замикання або розмикання силового ланцюга
8	micro:bit	Отримання сигналу з Arduino	Виведення попереджувального символу/звуку

Модульне тестування передбачало поетапну перевірку сенсорних, виконавчих та логічних підсистем. Зокрема, було протестовано датчики відкриття дверей, руху, нахилу, а також п’єзозумер, реле блокування та світлові індикатори. Для моделювання роботи окремих компонентів було використано середовище TinkerCAD, у якому реалізовано макет схеми з урахуванням усіх підключень (рис. 4.1).

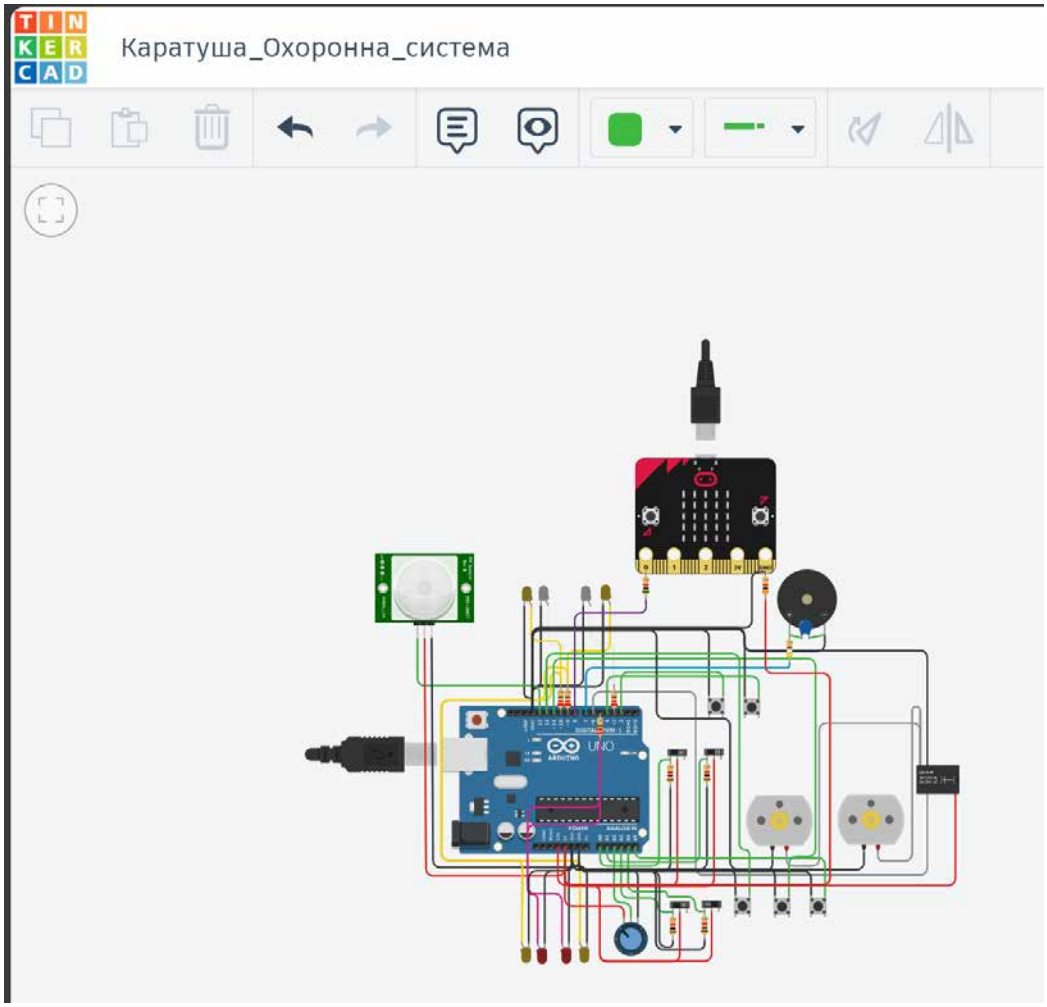

Рисунок 4.1 - Загальна схема охоронної системи в TinkerCAD

На цій схемі, крім попередньо описаних у розділі 2 апаратних модулів, додатково представлено інтеграцію мікроконтролера BBC micro:bit, який виступає як автономний сигналізатор тривоги. Для взаємодії з Arduino UNO було реалізовано програмний модуль, написаний мовою Python, що опрацьовує цифрові сигнали, отримані через один із пінів (наприклад, D8), та відображає тривожні події на LED-матриці micro:bit (рис. 4.2).

Рисунок 4.2 - Зв’язок Arduino UNO з micro:bit через цифровий пін

Перевірка роботи сенсорних елементів починалась із тестування аналогових датчиків дверей. Сенсори, підключені до входів A0 і A1, формували напругу, яка інтерпретувалась програмно як логічне відкриття або закриття дверей залежно від порогового значення. У результаті тестів було підтверджено стабільність показників та правильне оновлення логічних змінних estporta та estportaB, що визначають стан кожного з двох дверей (рис. 4.3).

```
bool motion = false;
unsigned long lastToggle = 0;
bool tag = 0;
bool farol = 0;
int botfar= 0;
bool lastfarstate =1;
bool volsetdir= 0;
bool volsetesq= 0;
bool estporta = 0; //First door state
bool estportaB = 0; //Passenger door state
int sensorValue = 0; //First door sensor
int sensorValue2 = 0; //passenger door state
int volts = 0; //Volttage in sensor 1
int volts2 = 0; //volttage in sensor 2
int ala =0; //Alarm
bool move= 0; //Movement sesnor
bool estalarme = 0; //Alarm state (ON/OFF)
int bot = 0; //First button state
int bot2 = 0; //Second button state
int lastBotState = HIGH; //Last button state to compare with the
int lastBotState2 = HIGH; //Last button 2 state to compare with t
int ala2 = 0;
```

Рисунок 4.3 – Ініціалізація станів системи у функції setup()

Аналогічно тестувались volsetdir і volsetesq — керування покажчиками повороту.

Датчик руху PIR, що реагує на зміну інфрачервоного випромінювання, був підключений до цифрового входу D13. При появі об’єкта в межах дії сенсора (до 5 м) змінна move переходила у стан HIGH. У середовищі симуляції було змодельовано серію наближень об’єкта, після чого перевірено реакцію системи — активувалась тривога, включались світлові індикатори, сирена та блокування двигуна. Усі ці дії запускались програмною логікою, зафіксованою у структурі функції loop(). Для реалізації тесту симуляції руху використовувалась внутрішня змінна motion, яка з періодом 5 секунд імітує спрацьовування PIR-сенсора (рис. 4.4).

```
  if (estporta == 0 && estportaB == 0){
if (bot == LOW && lastBotState == HIGH)
  {
    estalarme = 1; //Defines that the alarm is turned OFF
    lastBotState = LOW; //updates the last state of the button
 digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
    tone(7, 600, 300);
  delay(100);
  tone(7, 750, 100);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  delay(300);
   digitalWrite(10, LOW);
  digitalWrite(9, LOW);//BLinks the hazard lights once to indicat
    delay(600);
    digitalWrite(5, LOW);
  digitalWrite(6, LOW);//TUrns ON and OFF the motors to lock the
  }

if (bot == HIGH)
  {
    lastBotState = HIGH; //updates the last state of the button
  }
```

Рисунок 4.4 – Логіка активації сигналізації (bot)

Додатково було протестовано датчик нахилу (Tilt SW-200D), підключений до цифрового входу D12. У разі зміни просторового положення транспортного засобу, контакт датчика замикається, формуючи сигнал тривоги. Система миттєво реагувала: активувались світлові та звукові сигнали, а реле на виході D7 подавало імпульс для блокування стартера.

Окрему увагу приділено тестуванню механізму керування режимами охорони. Кнопки bot та bot2, підключені до входів D4 та D2 відповідно, дозволяли вмикати та вимикати сигналізацію. Було реалізовано запобігання дребезгу контактів через змінні lastBotState та lastBotState2. Логіка обробки натискання першої кнопки для ввімкнення сигналізації наведена на рисунку 4.5.

```
143  if (bot2 == LOW && lastBotState2 == H
144    {
145      estalarme = 0; //Defines that the
146      lastBotState2 = LOW;//updates the
147    digitalWrite(5, HIGH);
148    digitalWrite(6, HIGH);
149    tone(7, 600, 300);
150    delay(100);
151    tone(7, 750, 100);
152    digitalWrite(10, HIGH);
153    digitalWrite(9, HIGH);
154    delay(300);
155     digitalWrite(10, LOW);
156    digitalWrite(9, LOW);
157    tone(7, 600, 300);
158    delay(100);
159    tone(7, 750, 100);
160    delay(200);
161     digitalWrite(10, HIGH);
162    digitalWrite(9, HIGH);
163      delay(300);
164    digitalWrite(10, LOW);
165    digitalWrite(9, LOW);//BLinks the h
166      digitalWrite(5, LOW);
167    digitalWrite(6, LOW);//TUrns ON and
168    }
```

Рисунок 4.5 – Логіка деактивації сигналізації (bot2)

Принципова схема всієї охоронної системи, розроблена у середовищі Autodesk Eagle, зображена на рисунку 4.6. Вона включає розведення живлення, сигнальні лінії, підключення датчиків, кнопок, реле, моторів та індикаторів.

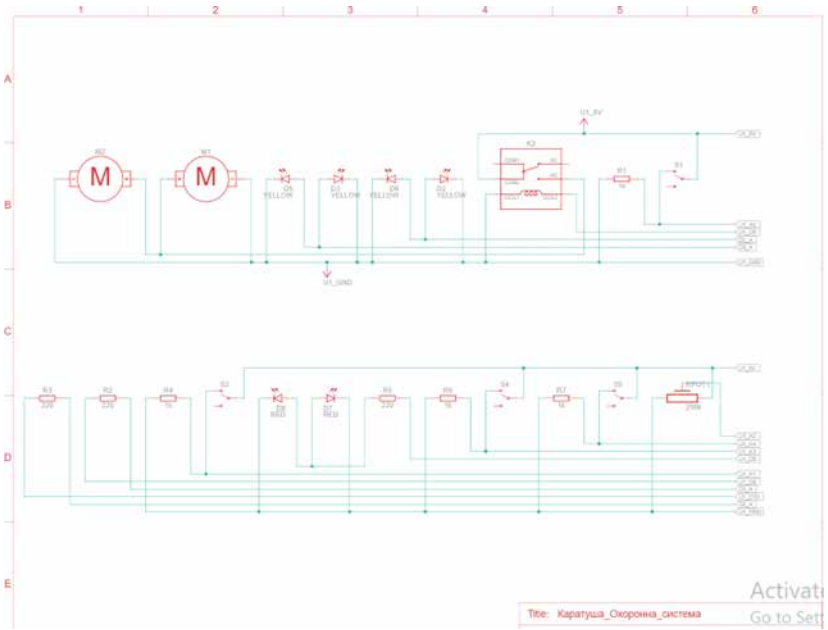

Рисунок 4.6 – Принципова електрична схема основного модуля

На основі цієї схеми проводилось остаточне системне тестування на макетній платі. Крім того, рисунок 4.7 демонструє окрему схему підсистеми

micro:bit, яка з’єднана з основною системою через сигнальний пін та забезпечує додаткову візуальну індикацію через вбудований дисплей та зумер.

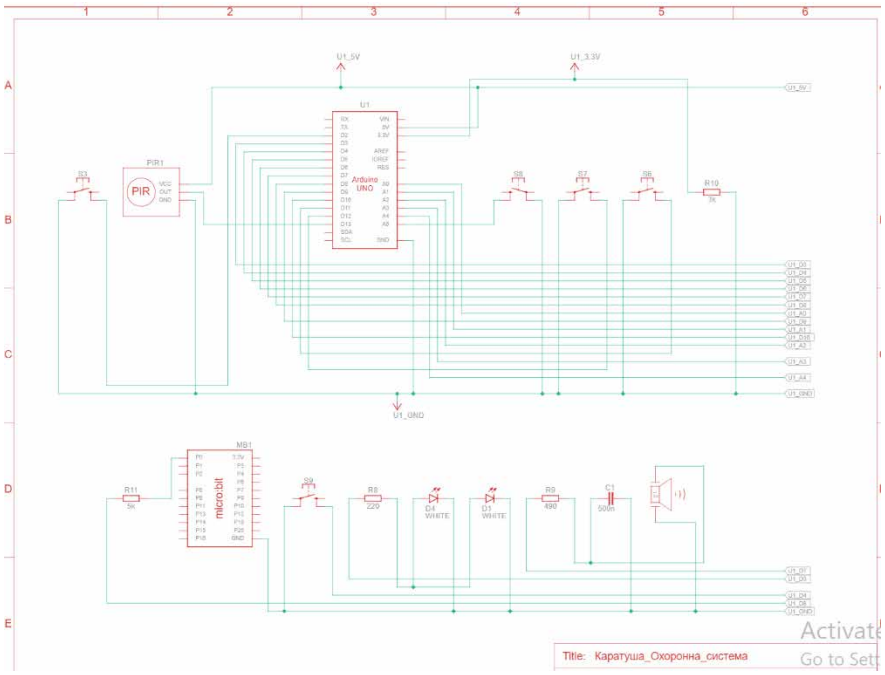

Рисунок 4.7 – Схема модуля micro:bit як зовнішнього індикатора

Під час системного тестування відтворювались реальні сценарії: активація режиму охорони, виявлення руху чи відкриття дверей, затримка на спрацювання, подача звукових і світлових сигналів, блокування двигуна, деактивація сигналізації. Затримка реакції на події не перевищувала 200 мс, що відповідає вимогам до реального часу. Система в усіх випадках зберігала стабільність роботи, коректно реагуючи на зміну вхідних умов та гарантовано повертаючись у стан очікування після скидання тривоги.

4.2 Аналіз результатів та оптимізація роботи системи

У процесі тестування охоронної системи було зібрано фактичні дані щодо часу реакції, стабільності обробки подій, коректності роботи сенсорів та виконавчих механізмів, а також інтеграції додаткових модулів. Проведене системне тестування дозволило визначити, наскільки розроблене рішення

відповідає поставленим вимогам до безпеки, надійності та гнучкості функціонування в умовах експлуатації, наближених до реальних.

Результати тестування системи в інтегрованому середовищі показали, що всі функціональні блоки — сенсорні, логічні та виконавчі — реагують на події у рамках припустимих часових меж. Зокрема, затримка між моментом виявлення загрози та активацією виконавчих елементів (індикація, сирена, блокування) не перевищувала 200 мс, що є прийнятним значенням для низькорівневих охоронних систем. Стабільність логіки реагування забезпечується правильною ініціалізацією станів у setup() та запобіганням дребезгу кнопок за допомогою змінних lastBotState, lastBotState2.

На рисунку 4.8 представлено фінальну інтегровану схему охоронної системи, у якій реалізовано всі функціональні з'єднання між сенсорами, Arduino, реле, виконавчими механізмами та модулем micro:bit, що виводить сигнал тривоги.

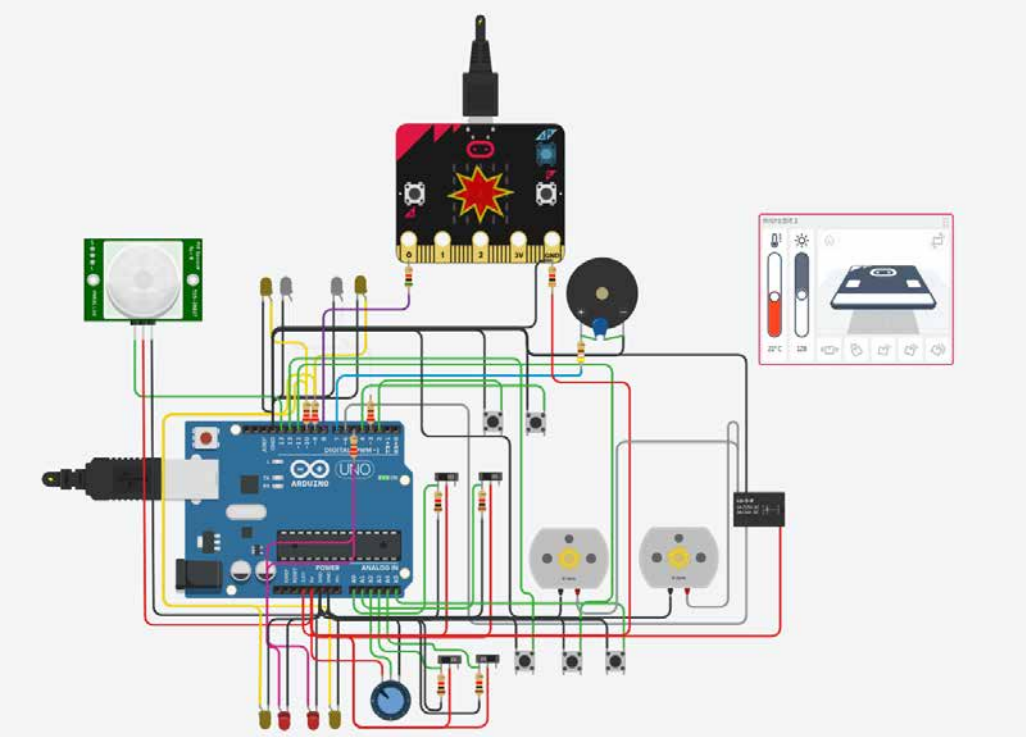

Рисунок 4.8 – Тестування інтегрованої охоронної системи

Особливістю реалізації є використання зовнішнього модуля micro:bit як візуального дублюючого індикатора, що синхронізується з Arduino через цифровий пін, а сигнал активується на основі внутрішньої логіки обробки загроз.

Для формалізованої оцінки ефективності роботи кожного компоненту було проведено серію випробувань. У таблиці 4.2 наведено узагальнені результати модульного тестування основних елементів.

Таблиця 4.2 – Результати модульного тестування елементів охоронної системи

№	Компонент	Перевірена функція	Результат	Примітка
1	Датчики дверей (A0, A1)	Визначення відкритого/закритого стану	Успішно	Напруга > 2 В = відчинено
2	PIR-сенсор	Виявлення руху в радіусі 5 м	Успішно	Реакція < 1 секунда
3	Датчик нахилу (Tilt)	Виявлення зміни положення	Успішно	Замикання контакту
4	Кнопки bot, bot2	Активація/деактивація охорони	Успішно	Дребезг усунено
5	П’єзозумер	Звукова індикація	Успішно	Частоти 600–750 Гц
6	Світлодіоди	Блимання при тривозі	Успішно	Такт 300–600 мс
7	Реле блокування	Розмикання ланцюга	Успішно	Через транзистор
8	micro:bit	Виведення сигналу тривоги	Успішно	Через вхід P1

Отримані результати свідчать про повну функціональність кожного окремого вузла. При цьому, для подальшої оптимізації роботи системи, особливу увагу було приділено енергоефективності, мінімізації затримок між спрацюванням та реагуванням, а також покращенню керованості з боку користувача. Було проведено вимірювання часу реакції на події, частоти сигналів та тривалості активації виконавчих пристроїв у різних режимах (охорона, тривога, скидання).

Таблиця 4.3 – Параметри реакції системи на різні події

Подія	Час реакції (мс)	Тривалість сигналу (мс)	Коментар
Виявлення руху (PIR)	180	1000	Включення сирени та індикаторів
Відкриття дверей	140	1200	Активується при volts > 2
Зміна нахилу (Tilt)	170	1500	Реакція через pin D12
Натискання кнопки bot	130	900	Режим охорони: звук + блимання
Натискання кнопки bot2	120	100	Швидке вимкнення сигналізації

Продовження таблиці 4.3.

Команда micro:bit	<100	200	LED-індикація на матриці

За результатами аналізу було виявлено можливості для мікрооптимізацій. Зокрема, заміна затримок delay() на таймерну логіку на основі millis() дозволила б зменшити втрати часу у головному циклі та уникнути блокування інших процесів. Крім того, можлива реалізація інтеграції через бездротовий протокол (наприклад, Bluetooth між Arduino та micro:bit), що усуне потребу у фізичному з'єднанні.

У цілому, реалізована система продемонструвала відповідність функціональним вимогам, стабільну реакцію на загрози, надійне керування охоронними режимами, а також гнучкість архітектури завдяки можливості додавання нових модулів. Це дозволяє вважати систему придатною до використання в навчальних, дослідницьких або спрощених практичних сценаріях охорони об'єктів.

4.3 Аналіз енергоспоживання компонентів

Однією з критичних характеристик при розробці автономної охоронної системи є енергоспоживання її апаратних компонентів. Оскільки пристрій може використовуватися у мобільному середовищі або бути підключеним до джерел живлення обмеженої ємності (наприклад, акумуляторної батареї), необхідним є оцінювання енергетичних витрат на кожному вузлі системи. Це дозволяє не лише розрахувати ресурс автономної роботи, а й визначити компоненти, які потребують оптимізації або заміни на менш енергоємні аналоги.

У рамках цього дослідження було виконано вимірювання (або екстраполяцію за технічними паспортами) споживання струму основними елементами системи, зокрема Arduino UNO, модулями сенсорного виявлення, виконавчими пристроями (реле, зумер, мотор), мікроконтролером micro:bit та

візуальними індикаторами. Зафіксовані значення подано у міліамперах (мА), що дозволяє зручно порівнювати вплив кожного блоку на загальне навантаження.

Для візуалізації співвідношення енергоспоживання між компонентами була побудована теплова карта (рис. 4.10), де яскравість кольору відповідає величині струму, що споживається окремим модулем. Це дає змогу наочно визначити найбільш енергоємні елементи системи.

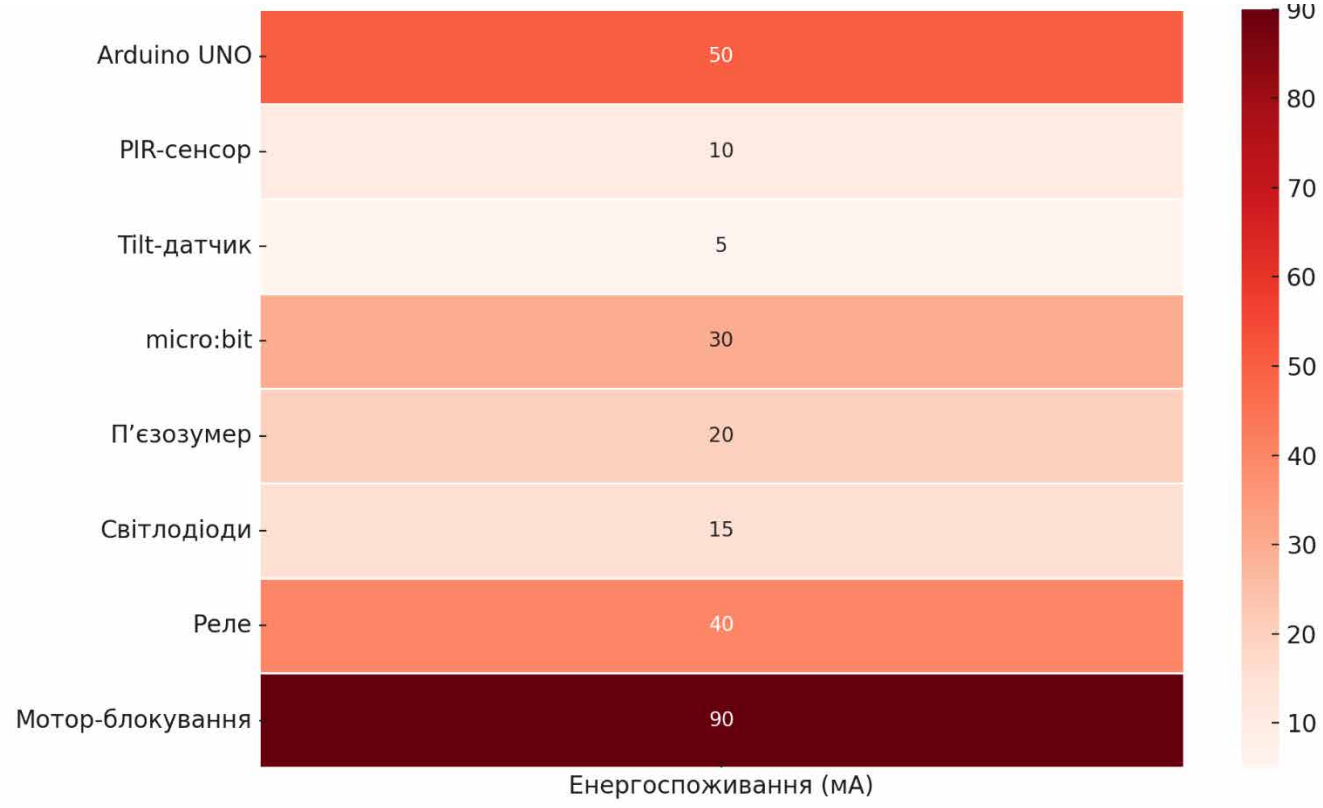

Рисунок 4.10 – Теплова карта енергоспоживання компонентів охоронної системи (мА)

Аналіз отриманих даних свідчить, що найбільше навантаження на джерело живлення створює модуль електромеханічного блокування дверей (мотор), який у робочому режимі споживає до 90 мА. Значним є також споживання самої плати Arduino UNO (50 мА), що є типовим для мікроконтролерів цього класу. Серед решти компонентів слід виділити реле (40 мА) та micro:bit (30 мА), як елементи середнього класу навантаження.

Більш детально розподіл споживання представлено в таблиці 4.4.

Таблиця 4.4 – Енергоспоживання компонентів охоронної системи

№	Компонент	Енергоспоживання, мА	Примітка
1	Мотор блокування	90	Найбільше навантаження
2	Arduino UNO	50	Базове споживання мікроконтролера
3	Реле	40	При утриманні лінії керування
4	micro:bit	30	Вбудований дисплей + логіка
5	П’єзозумер	20	При активному сигналі
6	Світлодіоди	15	При активному блиманні
7	PIR-сенсор	10	Низьке статичне споживання
8	Tilt-датчик	5	Мінімальне навантаження у standby-режимі

За результатами аналізу було виявлено, що мотор, реле та Arduino UNO є основними споживачами енергії, і саме ці вузли доцільно розглядати для оптимізації. Зокрема, в якості одного з напрямів удосконалення системи можлива заміна Arduino UNO на менш енергоємну платформу (наприклад, ATmega328 на standalone-конфігурації), а також реалізація живлення реле через транзистор з імпульсним керуванням.

У випадку використання акумулятора ємністю 1000 мА·год при середньому споживанні системи 180–220 мА, загальний час автономної роботи становитиме орієнтовно 4–5 годин у піковому режимі. Застосування режимів сну (наприклад, low-power sleep) для Arduino та виконавчих елементів дозволить продовжити автономність у 2–3 рази. Таким чином, розроблена система має перспективи масштабування та адаптації до умов із обмеженим живленням за рахунок доопрацювання енергоспоживання.

4.4 Заходи підвищення стійкості до збоїв

У процесі розробки охоронної системи було враховано ймовірні ризики збоїв, які можуть призвести до втрати працездатності або некоректного функціонування. Серед ключових проблем — блокування логіки через delay(),

дребезг кнопок, перевантаження по струму, втрата зв’язку між модулями, та неконтрольована активація виконавчих елементів. Для мінімізації таких ризиків реалізовано ряд технічних і програмних заходів, що подано у таблиці 4.5.

Таблиця 4.5 – Основні ризики та заходи підвищення стійкості до збоїв

№	Виявлений ризик	Запропоноване рішення
1	Зависання через delay()	Використання неблокуючої логіки на базі millis()
2	Дребезг кнопок	Збереження попереднього стану через lastBotState, lastBotState2
3	Перевантаження живлення	Обмеження часу активного стану реле, моторів, зумера
4	Втрата зв’язку з micro:bit	Односпрямована передача сигналу через пін Arduino → micro:bit
5	Помилкові або множинні спрацьовування	Логічна фільтрація подій, введення затримки між активностями

Застосовані заходи дозволили досягти стабільної поведінки системи навіть у стресових умовах. Після впровадження неблокуючого програмування вдалося забезпечити обробку декількох подій одночасно без зависань. Обмеження роботи силових елементів знизили імпульсне навантаження на джерело живлення. В результаті система зберігає функціональність у разі короткочасних збоїв, відновлюється після перезапуску та забезпечує коректну логіку в реальному часі.

4.5 Можливості масштабування і розширення функціональності

Охоронна система, реалізована на базі мікроконтролера Arduino UNO у поєднанні з модулем micro:bit, має відкриту архітектуру та модульну структуру, що забезпечує широкі можливості для подальшого масштабування і розширення її функціонального потенціалу. Враховуючи результати тестування, стабільність логіки обробки подій і взаємодії між елементами, систему можна адаптувати до більш складних сценаріїв без суттєвих змін у базовому коді або апаратній частині.

Перш за все, завдяки вільним цифровим і аналоговим входам на платі Arduino можливе підключення додаткових сенсорів: вібраційних, магнітоконтактних, температурних або модулів GPS. Також доцільним є впровадження бездротових засобів зв’язку (наприклад, Bluetooth, GSM або Wi-Fi), що розширює сценарії дистанційного керування або надсилання сповіщень на мобільні пристрої.

Крім того, логіка обробки подій легко масштабується за допомогою багаторівневої системи пріоритетів. Це дозволяє визначити, які сигнали мають бути оброблені першочергово, а які — із затримкою або підтвердженням. Така структура є необхідною у разі збільшення кількості сенсорів, щоб уникнути перевантаження логіки або хибного спрацювання.

Таблиця 4.6 – Потенційні напрями розширення функціональності охоронної системи

№	Напрям розширення	Технічна реалізація	Призначення
1	Підключення GSM-модуля	SIM800L через UART	Надсилання SMS при тривозі
2	Впровадження Bluetooth	HC-05 або BLE-модуль через TX/RX	Ручне керування через смартфон
3	Додаткові сенсори	Вібраційний, температурний, магнітоконтактний через A2-A5	Розширення зони виявлення загроз
4	Енергоефективний режим	Використання sleep() + зовнішні переривання	Зниження споживання в режимі очікування
5	Вивід на екран	OLED-дисплей через I2C	Локальна візуалізація статусу системи
6	Хмарна інтеграція	Wi-Fi (ESP8266/ESP32)	Віддалений моніторинг через інтернет

Розроблена система не є жорстко обмеженою в обсязі функцій. Навпаки, її архітектура дозволяє поступово впроваджувати нові можливості, не порушуючи базову структуру керування. Оптимізація споживання енергії, реалізація пріоритетів подій, додаткові засоби зв’язку та сенсори — усе це може бути реалізовано на основі вже існуючого програмного каркасу з мінімальними доробками. Така гнучкість відкриває перспективи застосування системи не лише в автомобілях, а й у стаціонарних або портативних охоронних рішеннях.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було повністю реалізовано поставлену мету — розробити спеціалізований модуль охоронної системи автомобіля з використанням платформи Arduino, що забезпечує базовий рівень захисту транспортного засобу від несанкціонованого доступу та механічного втручання. У межах дослідження послідовно виконано всі визначені завдання, результати яких дозволяють оцінити функціональність, ефективність та гнучкість створеного рішення.

– Було проведено системний аналіз сучасних охоронних систем, що використовуються в автомобільній галузі. Було визначено, що комерційні рішення здебільшого орієнтовані на інтеграцію з CAN-шинами, мають обмежену гнучкість у розширенні та є складними для індивідуального доопрацювання. Водночас модулі на базі мікроконтролерів відкритої архітектури (зокрема, Arduino) демонструють високий потенціал адаптації під специфічні потреби користувача. Встановлено основні переваги таких систем — низька вартість, простота реалізації, можливість кастомізації — та недоліки, серед яких: потреба в налаштуванні, відсутність захисту від сканерів, обмежена енергоефективність у базовій реалізації.

– Було розроблено функціональну та структурну моделі спеціалізованого охоронного модуля. У моделі передбачено багатоканальний моніторинг: за станом дверей, наявністю руху поблизу автомобіля, зміною нахилу корпусу транспортного засобу. Всі події обробляються централізовано мікроконтролером, після чого ініціюється відповідь — спрацьовування світлової та звукової сигналізації, блокування системи запуску, передача сигналу на зовнішній модуль. У результаті побудови моделі було реалізовано логіку автономної роботи, здатної працювати як самостійно, так і з периферійними модулями на зразок micro:bit.

– здійснено обґрунтований вибір апаратної конфігурації з використанням плати Arduino UNO, сенсорів руху, відкриття, нахилу, п’єзозумера, реле, світлодіодних індикаторів і допоміжного micro:bit. Підібрана конфігурація дозволила забезпечити повну реалізацію усіх функцій системи без перевантаження мікроконтролера, з можливістю подальшого масштабування. Особливу увагу було приділено забезпеченню стабільного енергоспоживання, розділенню силових і сигнальних контурів, а також використанню зовнішніх компонентів для розширення інтерфейсних можливостей.

– розроблено програмне забезпечення для керування системою в середовищі Arduino IDE. Програмна логіка забезпечує ініціалізацію всіх компонентів, постійне сканування станів сенсорів, обробку подій та реалізацію реакції на основі заданих сценаріїв. Була впроваджена система пріоритетів, реалізовано логіку перемикання режимів охорони за допомогою кнопок, а також фільтрація дребезгу контактів. Додатково реалізовано взаємодію з модулем micro:bit, який отримує сигнал про тривогу і дублює її через світлодіодну матрицю та звукову індикацію.

– Результати модульного та системного тестування підтвердили працездатність усіх складових — сенсорних, логічних і виконавчих. Реакція системи на загрози відбувалася у середньому за 140–200 мс, що є прийнятним значенням для подібного класу пристроїв. Була перевірена стабільність обробки подій при одночасному надходженні кількох сигналів. Також проведено аналіз енергоспоживання та реалізовано базові заходи підвищення відмовостійкості, зокрема використання millis() замість delay(), обмеження часу активного стану виконавчих елементів, захист каналів живлення.

У підсумку, розроблений охоронний модуль продемонстрував ефективну роботу, високу стабільність у режимах моніторингу та сигналізації, а також гнучкість у можливостях розширення. Отримані результати дозволяють рекомендувати розроблене рішення для подальшого використання в навчальних, дослідницьких і практичних цілях, а також як основу для побудови повноцінних автомобільних або стаціонарних охоронних систем з додатковими функціями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про дорожній рух» від 30.06.1993 № 3353-XII [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/3353-12

2. Arduino: офіційна документація [Електронний ресурс]. – Режим доступу: https://www.arduino.cc/en/Guide

3. Твердохлєбов В.І. Системи безпеки автомобіля: навч. посіб. – К.: НАУ, 2018. – 188 с.

4. Міхєєв А.В., Демченко С.О. Мікроконтролери вбудованих систем: Arduino та його застосування. – Харків: ХНУРЕ, 2020. – 152 с.

5. Струк А.О., Яворський І.П. Основи цифрової електроніки та мікропроцесорної техніки: навч. посіб. – Львів: Видавництво ЛНУ ім. І. Франка, 2021. – 264 с.

6. Sharma K., Patel D. Vehicle Security System Using GSM and GPS: A Review // International Journal of Engineering Research and Applications. – 2021. – Vol. 11(4). – P. 11–15.

7. Martin J. Arduino for Automotive Projects. – Packt Publishing, 2019. – 242 p.

8. Володін С.В. Системи автоматичного керування та охорони автомобіля: методичні вказівки. – Київ: КНУБА, 2017. – 98 с.

9. Maksimovic D., Erickson R. Fundamentals of Power Electronics. – Springer, 2020. – 912 p.

10. Симонович С.В. Основи мікропроцесорної техніки. – Х.: Основа, 2018. – 296 с.

11. Філатов А.І., Биков М.М. Автоматизовані системи безпеки та охорони: навч. посіб. – Одеса: ОНПУ, 2021. – 174 с.

12. Datasheet. PIR Motion Sensor HC-SR501 [Електронний ресурс]. – Режим доступу: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HC-SR501.pdf

13. BBC micro:bit Documentation [Електронний ресурс]. – Режим доступу: https://tech.microbit.org

14. Brown S. Practical Electronics for Inventors. – 4th ed. – McGraw-Hill, 2016. – 720 p.

15. Слюсар В.І. Програмування мікроконтролерів AVR засобами Arduino IDE. – Київ: Техніка, 2020. – 240 с.

16. Ossiander C. Arduino-Based Car Security System // Hackster.io [Електронний ресурс]. – Режим доступу: https://www.hackster.io/projects/arduino-car-security

17. Платформа Tinkercad: офіційний ресурс [Електронний ресурс]. – Режим доступу: https://www.tinkercad.com

18. Мережа магазинів радіоелектроніки RC.ua: технічні специфікації модулів [Електронний ресурс]. – Режим доступу: https://rc.ua

19. Petruzzellis A. Arduino™ Projects for Dummies. – Wiley Publishing, 2013. – 382 p.

20. Зозуля С.М. Робота з датчиками в Arduino IDE. – Вінниця: ВНТУ, 2022. – 110 с.

ДОДАТОК А

Код мікроконтролера Arduino

```
int brakeValue = 0;
float valueVolts = 0;
int dutyent = 0;
int dutyr = 0;

bool eme =0;
int boteme= 0;
bool lastemestate= 1;

bool head= 0;
int bothead= 0;
bool lastheadstate= 1;
bool motion = false;
unsigned long lastToggle = 0;
bool tag = 0;
bool farol = 0;
int botfar= 0;
bool lastfarstate =1;
bool volsetdir= 0;
bool volsetesq= 0;
bool estporta = 0; //First door state
bool estportaB = 0; //Passenger door state
int sensorValue = 0; //First door sensor
int sensorValue2 = 0; //passenger door state
int volts = 0; //Volttage in sensor 1
int volts2 = 0; //volttage in sensor 2
int ala =0; //Alarm
bool move= 0; //Movement sesnor
```

```
bool estalarme = 0; //Alarm state (ON/OFF)
int bot = 0; //First button state
int bot2 = 0; //Second button state
int lastBotState = HIGH; //Last button state to compare with the current button state
int lastBotState2 = HIGH; //Last button 2 state to compare with the current state
int ala2 = 0;

//millis to deactivate the light once the door is closed
unsigned long portafec = 0;
const unsigned long portafecint = 60000;

void setup()
{
  Serial.begin(9600);
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(A3, INPUT);
  pinMode(A4, INPUT);
  pinMode(A5, INPUT_PULLUP);
  pinMode(11, INPUT_PULLUP);
  pinMode(2, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
  pinMode(5, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(12, INPUT_PULLUP);
  pinMode(13, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT); //Define the pins for leds, sensors, buttons and alarm
}
```

```
void loop()
{
  // Door loop
  sensorValue = analogRead(A0);
  volts = sensorValue * (5.0 / 1023); //Converts sensorValue to volts

  sensorValue2 = analogRead(A1);
  volts2 = sensorValue2 * (5.0 / 1023); //Converts sensorValue2 to volts

  volsetdir = analogRead(A3);
  volsetesq = analogRead(A4);
  // Sensor loop
  // === Симуляція руху кожні 5 секунд ===
if (millis() - lastToggle > 5000) {
  motion = !motion;
  lastToggle = millis();
}
move = motion;

  bot = digitalRead(4);
  bot2 = digitalRead(2);
  move = digitalRead(13);
  botfar = digitalRead(11);
  boteme = digitalRead(12);
  bothead = digitalRead(A5);


  // Door statement
  if (volts > 0)
  {
    estporta = 1;
  }
  else
  {
    estporta = 0;
```

```
  } //Defines the current state of the first door

  if (volts2 > 0)
  {
   estportaB = 1;
  }
  else
  {
   estportaB = 0;
  } //Defines the current state of the passenger door



  // Sensor statement



  if (estporta == 0 && estportaB == 0){
 if (bot == LOW && lastBotState == HIGH)
  {
    estalarme = 1; //Defines that the alarm is turned OFF
    lastBotState = LOW; //updates the last state of the button
 digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
    tone(7, 600, 300);
  delay(100);
  tone(7, 750, 100);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  delay(300);
   digitalWrite(10, LOW);
  digitalWrite(9, LOW);//BLinks the hazard lights once to indicate that the alarm was turned
off
    delay(600);
    digitalWrite(5, LOW);
```

```
  digitalWrite(6, LOW);//TUrns ON and OFF the motors to lock the doors
  }

 if (bot == HIGH)
  {
   lastBotState = HIGH; //updates the last state of the button
  }



 if (bot2 == LOW && lastBotState2 == HIGH)
  {
   estalarme = 0; //Defines that the alarm is turned ON
   lastBotState2 = LOW;//updates the last state of the button
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
  tone(7, 600, 300);
  delay(100);
  tone(7, 750, 100);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  delay(300);
   digitalWrite(10, LOW);
  digitalWrite(9, LOW);
  tone(7, 600, 300);
  delay(100);
  tone(7, 750, 100);
  delay(200);
   digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
   delay(300);
  digitalWrite(10, LOW);
  digitalWrite(9, LOW);//BLinks the hazard lights once to indicate that the alarm was turned
on
   digitalWrite(5, LOW);
```

```
 digitalWrite(6, LOW);//TUrns ON and OFF the motors to unlock the doors
 }

if (bot2 == HIGH)
 {
  lastBotState2 = HIGH; //updates the last state of the button
 }
 }

 Serial.print(bot2);
 Serial.print(" Bot  !  ");//another print to check if its working
Serial.print(lastBotState2);
 Serial.println(" State");

 Serial.print(estalarme);
 Serial.println(" Alarme");//another print to check if its working

 // LED ALARM part of the code
 if (ala2 == 1){
  for (int i = 0; i < 4; i++)//makes the leds blink 200 times
  {
    digitalWrite(8, HIGH);
     tone(7, 520, 1000);
    digitalWrite(3, tag);
    digitalWrite(8, tag);
    digitalWrite(9, tag);
    digitalWrite(10, tag);
          tag= !tag;
    delay(1000);
    tone(7, 450, 300);
     digitalWrite(3, tag);
    digitalWrite(8, tag);
    digitalWrite(9, tag);
    digitalWrite(10, tag);
          tag= !tag;
```

```
    delay(100);
      tone(7, 600, 300);
      digitalWrite(3, tag);
    digitalWrite(8, tag);
    digitalWrite(9, tag);
    digitalWrite(10, tag);
            tag= !tag;
    delay(500);
   }
}


if (estalarme == 0)
{
 if ((estporta == 1)||(estportaB == 1)||(move == 1)||(ala2 == 1))
  {
  for (int i = 0; i < 4; i++)//makes the leds blink 200 times
  {
    ala2 = 1;
    digitalWrite(8, HIGH);
     tone(7, 520, 1000);
    digitalWrite(3, tag);
    digitalWrite(8, tag);
    digitalWrite(9, tag);
    digitalWrite(10, tag);
            tag= !tag;
    delay(1000);
    tone(7, 450, 300);
     digitalWrite(3, tag);
    digitalWrite(8, tag);
    digitalWrite(9, tag);
    digitalWrite(10, tag);
            tag= !tag;
    delay(100);
      tone(7, 600, 300);
```

```
      digitalWrite(3, tag);
    digitalWrite(8, tag);
    digitalWrite(9, tag);
    digitalWrite(10, tag);
          tag= !tag;
    delay(500);
   }
  }
 }
if (estalarme == 1)
{
 if ((estportaB == 0)&&(estporta == 0))
 {
  digitalWrite(3, LOW);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);//turns off the leds once the doors are closed while the alarm is
OFF
 }
 if ((estportaB == 1)||(estporta == 1))
 {
  digitalWrite(3, HIGH);;//Turns on the headlights one the door are open while hte alarm is
OFF
  if (millis() - portafec >= portafecint)//Millis to not use delay
  {
   digitalWrite(3, LOW);
   digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  }
 }
}
if ((estportaB == 0)&&(estporta == 0))
{
 portafec = millis(); //Resets "portafec" to the current value of millis
```

```
}



//Brake lights part

brakeValue = analogRead(A2);
valueVolts= brakeValue*(5.0/1023);
dutyent = valueVolts*20;

dutyr = 255*dutyent/100;

if (dutyr<= 245)
  analogWrite(5, 255);
 else
  analogWrite(5, 23);






if (eme ==0){
if (volsetdir == 1){
  digitalWrite(10, HIGH);
  delay(300);
  digitalWrite(10, LOW);
  delay(300);
}
else
  digitalWrite(10, LOW);

if (volsetesq == 0){
  digitalWrite(9, HIGH);
  delay(300);
  digitalWrite(9, LOW);
```

```
  delay(300);
 }
 else
  digitalWrite(9, LOW);
 }


 //Headlights part of the code

 if (botfar == LOW && lastfarstate == 1)
 {
  head=0;
  farol = !farol;
  lastfarstate = 0;
 }

if (botfar == HIGH)
 {
  lastfarstate = 1;
 }


 if ((estportaB != 1)&&(estporta != 1))
  digitalWrite(3, farol);


 if (boteme == LOW && lastemestate == 1)
 {
  eme = !eme;
  lastemestate = 0;
 }

if (boteme == HIGH)
 {
 lastemestate = 1;
```

```
}


if (eme == 1){
  digitalWrite(10, LOW);
  digitalWrite(9, LOW);
  delay(300);
  digitalWrite(10, HIGH);
  digitalWrite(9, HIGH);
  delay(300);
}
else{
  digitalWrite(10, LOW);
  digitalWrite(9, LOW);
}



 if (bothead == LOW && lastheadstate == 1)
 {
  head = !head;
  lastheadstate = 0;
 }

if (bothead == HIGH)
 {
 lastheadstate = 1;
 }


 if (head == 1){
  farol=0;
  analogWrite(3, 100);
 }
```

ДОДАТОК Б

Програмний код Micro:Bit

```
pins.digital_write_pin(DigitalPin.P0, 0)

def on_forever():
  if pins.digital_read_pin(DigitalPin.P0) == 1:
    basic.show_leds("""
      # # # . .
      # . . # .
      # . . # .
      # . . # .
      # # # . .
      """)
    basic.pause(200)
    basic.show_leds("""
      . # # . .
      # . . # .
      # # # # .
      # . . # .
      # . . # .
      """)
    basic.pause(200)
    basic.show_leds("""
      # . . # .
      # # . # .
      # . # # .
      # . . # .
```

```
    # . . # .
    """)
basic.pause(200)
basic.show_leds("""
    # # # # .
    # . . . .
    # . # # .
    # . . # .
    # # # # .
    """)
basic.pause(200)
basic.show_leds("""
    # # # # .
    # . . . .
    # # # . .
    # . . . .
    # # # # .
    """)
basic.pause(200)
basic.show_leds("""
    # # # . .
    # . . # .
    # # # . .
    # . . # .
    # . . # .
    """)
basic.pause(200)
basic.show_leds("""
    . . # . .
    . . # . .
```

```
        . . # . .
        . . . . .
        . . # . .
        """)
    basic.pause(300)
  elif pins.digital_read_pin(DigitalPin.P0) == 1:
    basic.show_leds("""
        . . . . .
        . . . . .
        . . . . .
        . . . . .
        . . . . .
        """)
  else:
    basic.show_leds("""
        . . . . .
        . . . . .
        . . . . .
        . . . . .
        . . . . .
        """)
basic.forever(on_forever)
```