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.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему: «Розробка комп'ютерної системи охорони приватного будинку»

Спеціальність 123 «Комп’ютерна інженерія»

Гарант освітньої програми

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

(підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи: _____ / Мисюра М.Д. /

підпис ПІБ

Виконав: _____ / Сучилкін М.Д. /

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., к.п.н., доц. /
підпис ПІБ, вчене звання і ступінь
«__» _____20__р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Сучилкін Максим Дмитрович
(прізвище, ім'я, по батькові)
Спеціальність (напрям підготовки): комп'ютерна інженерія_____

Тема бакалаврської кваліфікаційної роботи: «Розробка комп'ютерної системи охорони приватного будинку»_____
_____
_____
затверджена наказом ректора НУБіП України від "_16_" _грудня_2024р. №_2250.С _
Термін подання завершеної роботи на кафедру _____
Вихідні дані до бакалаврської кваліфікаційної роботи _____
_____
_____
_____
_____
Перелік питань, що підлягають розробці:
1. _____
2. _____
3. _____
Перелік графічного матеріалу (за потреби) _____
_____

Дата видачі завдання "_____"_____2024 р.

Керівник бакалаврської кваліфікаційної роботи _____ Місюра М.Д.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Сучилкін М.Д.
(підпис) (прізвище та ініціали)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	01.02.2025 – 15.02.2025	Виконано
2	Проектування системи	16.02.2025 – 29.02.2025	Виконано
3	Реалізація системи	01.03.2025 – 10.04.2025	Виконано
4	Тестування системи	11.04.2025 – 20.04.2025	Виконано
5	Оформлення пояснювальної записки	21.04.2025 – 10.05.2025	Виконано
6	Оформлення графічного матеріалу	11.05.2025 – 20.05.2025	Виконано

Студент _____ _____
(підпис) (ініціали та прізвище)

Керівник роботи _____ _____
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота складається із 75 сторінки, 7 таблиць, 3 рисунків, 37 використаних літературних джерел, додатків.

Мета роботи – розробити комп'ютерну систему охорони приватного будинку на базі мікроконтролера Arduino, яка забезпечує базові функції безпеки (детекцію руху, сповіщення про вторгнення, віддалений контроль) із можливістю масштабування та інтеграції з мережевими технологіями.

Об'єкт дослідження – комп'ютерні системи охорони приватного будинку, побудовані на основі мікроконтролерів та мережевих технологій.

Предмет дослідження – принципи проектування, розробки та тестування системи охорони приватного будинку з використанням Arduino, датчиків і мережевих протоколів (Wi-Fi, Ethernet, Zigbee).

Сучасний світ характеризується стрімким розвитком технологій, які проникають у всі сфери людського життя, зокрема в забезпечення безпеки житла. Приватний будинок, як основне місце проживання багатьох людей, потребує надійного захисту від зовнішніх загроз, таких як несанкціоноване проникнення, крадіжки чи інші небезпеки. Зростання попиту на системи безпеки зумовлене не лише підвищенням рівня злочинності, а й прагненням власників житла до комфорту, автоматизації та віддаленого контролю. У цьому контексті комп'ютерні системи охорони стають ключовим елементом забезпечення безпеки, поєднуючи апаратні та програмні рішення для створення інтегрованих і ефективних комплексів.

Сьогодні на ринку представлено широкий спектр систем охорони, від комерційних продуктів, таких як Ajax чи Xiaomi Mijia, до саморобних рішень на базі мікроконтролерів, таких як Arduino.

ABSTRACT

The qualification work consists of 75 pages, 7 tables, 3 figures, 37 used literature sources, appendices.

The purpose of the work is to develop a computer security system for a private house based on the Arduino microcontroller, which provides basic security functions (motion detection, notification of inclusion, remote control) with the possibility of scaling and integration with network technologies.

The object of research is computer security systems for a private house, built on the basis of microcontrollers and network technologies.

The subject of research is the principles of design, development and testing of private house security systems using Arduino, sensors and network protocols (Wi-Fi, Ethernet, Zigbee).

The modern world is characterized by the rapid development of technologies that penetrate all spheres of human life, in particular, in ensuring home security. A private home, as the main place of residence for many people, requires reliable protection from external threats, such as unauthorized entry, theft or other dangers. The growing demand for security systems is due not only to the increase in crime rates, but also to the desire of homeowners for comfort, automation and remote control. In this context, computer security systems are becoming a key element of security, combining hardware and software solutions to create integrated and effective complexes.

Today, the market offers a wide range of security systems, from commercial products such as Ajax or Xiaomi Mijia, to home-made solutions based on microcontrollers such as Arduino.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

Arduino – відкрита апаратно-програмна платформа на базі мікроконтролерів для створення електронних систем.

ESP8266 – Wi-Fi-модуль для забезпечення бездротового мережевого зв’язку в IoT-пристроях.

GPIO – General Purpose Input/Output, універсальні піни вводу/виводу мікроконтролера.

HC-SR501 – модель пасивного інфрачервоного датчика руху (PIR).

HTTP – HyperText Transfer Protocol, протокол передачі гіпертексту для обміну даними в мережі.

I2C – Inter-Integrated Circuit, протокол зв’язку для обміну даними між мікроконтролером і периферійними пристроями.

IoT – Internet of Things, Інтернет речей, концепція підключення пристроїв до мережі для обміну даними.

MQTT – Message Queuing Telemetry Transport, протокол передачі повідомлень для IoT-систем.

MQ-2 – модель датчика для виявлення диму та газів.

PIR – Passive Infrared, пасивний інфрачервоний датчик для виявлення руху.

SD-карта – Secure Digital card, носій даних для зберігання інформації.

SPI – Serial Peripheral Interface, послідовний периферійний інтерфейс для зв’язку між пристроями.

UART – Universal Asynchronous Receiver-Transmitter, універсальний асинхронний приймач-передавач для послідовного зв’язку.

Wi-Fi – Wireless Fidelity, технологія бездротового зв’язку на основі стандартів IEEE 802.11.

XBee – торгова марка модулів для реалізації Zigbee-мереж.

Zigbee – протокол бездротового зв’язку для енергоефективних локальних мереж.

AES – Advanced Encryption Standard, стандарт шифрування даних.

HTTPS – HyperText Transfer Protocol Secure, захищена версія HTTP із шифруванням.

Кбіт/с – кілобіт за секунду, одиниця вимірювання швидкості передачі даних.

Мбіт/с – мегабіт за секунду, одиниця вимірювання швидкості передачі даних.

мВт – міліват, одиниця вимірювання потужності.

ЗМІСТ

ВСТУП

Сучасний світ характеризується стрімким розвитком технологій, які проникають у всі сфери людського життя, зокрема в забезпечення безпеки житла. Приватний будинок, як основне місце проживання багатьох людей, потребує надійного захисту від зовнішніх загроз, таких як несанкціоноване проникнення, крадіжки чи інші небезпеки. Зростання попиту на системи безпеки зумовлене не лише підвищенням рівня злочинності, а й прагненням власників житла до комфорту, автоматизації та віддаленого контролю. У цьому контексті комп'ютерні системи охорони стають ключовим елементом забезпечення безпеки, поєднуючи апаратні та програмні рішення для створення інтегрованих і ефективних комплексів.

Сьогодні на ринку представлено широкий спектр систем охорони, від комерційних продуктів, таких як Ajax чи Xiaomi Mijia, до саморобних рішень на базі мікроконтролерів, таких як Arduino. Комерційні системи зазвичай пропонують високу надійність і зручність використання, але їхня висока вартість і обмежена гнучкість часто стають бар'єрами для користувачів. Водночас саморобні системи, побудовані на відкритих платформах, дозволяють адаптувати функціонал під конкретні потреби, знижуючи витрати. Проте такі рішення вимагають глибокого розуміння апаратного та програмного забезпечення, а також ретельного тестування для забезпечення надійності.

Актуальність теми дослідження полягає в необхідності розробки доступної, гнучкої та ефективної комп'ютерної системи охорони для приватного будинку, яка б поєднувала сучасні технології, такі як мікроконтролери Arduino та мережеві протоколи (Wi-Fi, Ethernet, Zigbee), із простотою реалізації та економічною доцільністю. Зростання популярності концепції «розумного будинку» лише підкреслює важливість створення систем, здатних не лише забезпечувати безпеку, а й інтегруватися з іншими автоматизованими пристроями. В Україні, де економічні фактори часто обмежують доступ до дорогих систем безпеки, розробка таких рішень має особливе значення.

Мета роботи – розробити комп'ютерну систему охорони приватного будинку на базі мікроконтролера Arduino, яка забезпечує базові функції безпеки (детекцію руху, сповіщення про вторгнення, віддалений контроль) із можливістю масштабування та інтеграції з мережевими технологіями.

Об'єкт дослідження – комп'ютерні системи охорони приватного будинку, побудовані на основі мікроконтролерів та мережевих технологій.

Предмет дослідження – принципи проектування, розробки та тестування системи охорони приватного будинку з використанням Arduino, датчиків і мережевих протоколів (Wi-Fi, Ethernet, Zigbee).

Завдання роботи:

1. Провести аналіз існуючих систем охорони, визначивши їхні переваги, недоліки та можливості для вдосконалення.
2. Сформулювати функціональні та нефункціональні вимоги до розроблюваної системи.
3. Розробити архітектуру системи, обґрунтувати вибір апаратних і програмних компонентів.
4. Реалізувати прототип системи охорони на базі Arduino з інтеграцією датчиків і мережевих модулів.
5. Провести тестування прототипу в лабораторних і наближених до реальних умовах, оцінити його ефективність.
6. Запропонувати рекомендації щодо впровадження та вдосконалення системи.

Методи дослідження включають аналіз літератури та існуючих рішень, системний підхід до проектування архітектури, експериментальну розробку прототипу, а також методи тестування й оцінки ефективності системи. Теоретичною основою слугують принципи роботи мікроконтролерів, мережевих технологій і систем безпеки, викладені в роботах українських і зарубіжних авторів.

Наукова новизна роботи полягає в розробці доступної системи охорони на базі Arduino, яка поєднує кілька мережевих технологій (Wi-Fi, Ethernet, Zigbee)

для забезпечення гнучкості, енергоефективності та віддаленого доступу, адаптованої до потреб приватного будинку в українських реаліях.

Практична цінність роботи полягає в створенні прототипу системи, який може бути використаний як основа для подальшої комерціалізації або впровадження в приватних домогосподарствах. Розроблена система пропонує економічно вигідне рішення, яке дозволяє користувачам самостійно налаштовувати та розширювати функціонал залежно від потреб. Крім того, результати роботи можуть бути застосовані в освітніх цілях для вивчення основ IoT і мікроконтролерів.

Структура роботи включає вступ, три основні розділи (аналіз предметної області, теоретична частина, практична реалізація), висновки, список використаної літератури та додатки. У першому розділі проведено аналіз сучасних систем охорони та сформульовано вимоги до розробки. Другий розділ присвячено теоретичним основам, зокрема архітектурі системи, вибору компонентів і мережевих технологій. Третій розділ описує процес створення прототипу, його інтеграцію, тестування та апробацію. У висновках узагальнено результати, а в додатках наведено схеми, програмний код і результати експериментів.

Дана робота спрямована на вирішення актуальної задачі створення доступної та ефективної системи охорони, яка відповідає сучасним технологічним трендам і потребам користувачів. Розробка такої системи сприятиме підвищенню рівня безпеки приватних будинків, а також популяризації технологій IoT в Україні.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд існуючих рішень: Аналіз сучасних систем охорони, їх переваг та недоліків

Сучасні системи охорони приватних будинків є важливим елементом забезпечення безпеки, що поєднують апаратні та програмні засоби для захисту від несанкціонованого доступу, крадіжок та інших загроз. Зростання попиту на такі системи зумовлене як підвищенням рівня злочинності, так і прагненням власників житла до автоматизації та віддаленого контролю. Сьогодні на ринку представлено широкий спектр рішень – від комерційних продуктів до саморобних систем на базі відкритих платформ, таких як Arduino. Кожен тип системи має свої особливості, які необхідно враховувати при розробці власного рішення. У цьому підрозділі проведено аналіз сучасних систем охорони, їхніх переваг і недоліків, щоб визначити напрями для створення ефективного та доступного рішення.

Комерційні системи охорони, такі як Ajax, Xiaomi Mijia чи Hikvision, є одними з найпоширеніших на ринку. Вони пропонують комплексний підхід до безпеки, включаючи датчики руху, камери відеоспостереження, сирени та мобільні додатки для віддаленого керування. Наприклад, система Ajax StarterKit забезпечує захищений радіоканал і швидкі сповіщення на смартфон користувача, що дозволяє оперативно реагувати на загрози [1]. Однак такі системи мають високу вартість, яка може сягати кількох тисяч гривень за базовий комплект, що робить їх недоступними для багатьох користувачів. Крім того, комерційні системи часто обмежені у гнучкості налаштувань, оскільки їхній функціонал визначений виробником, а інтеграція з іншими пристроями може бути ускладненою через закритий код.

На противагу комерційним рішенням, саморобні системи на базі мікроконтролерів, таких як Arduino, стають дедалі популярнішими серед ентузіастів і розробників. Такі системи дозволяють створювати індивідуальні рішення, адаптовані до конкретних потреб користувача. Наприклад, Arduino може інтегруватися з датчиками руху, диму чи магнітними контактами, забезпечуючи базові функції охорони за значно нижчою вартістю [2]. Для

ілюстрації можливостей таких систем наведено схему базового підключення датчиків до Arduino

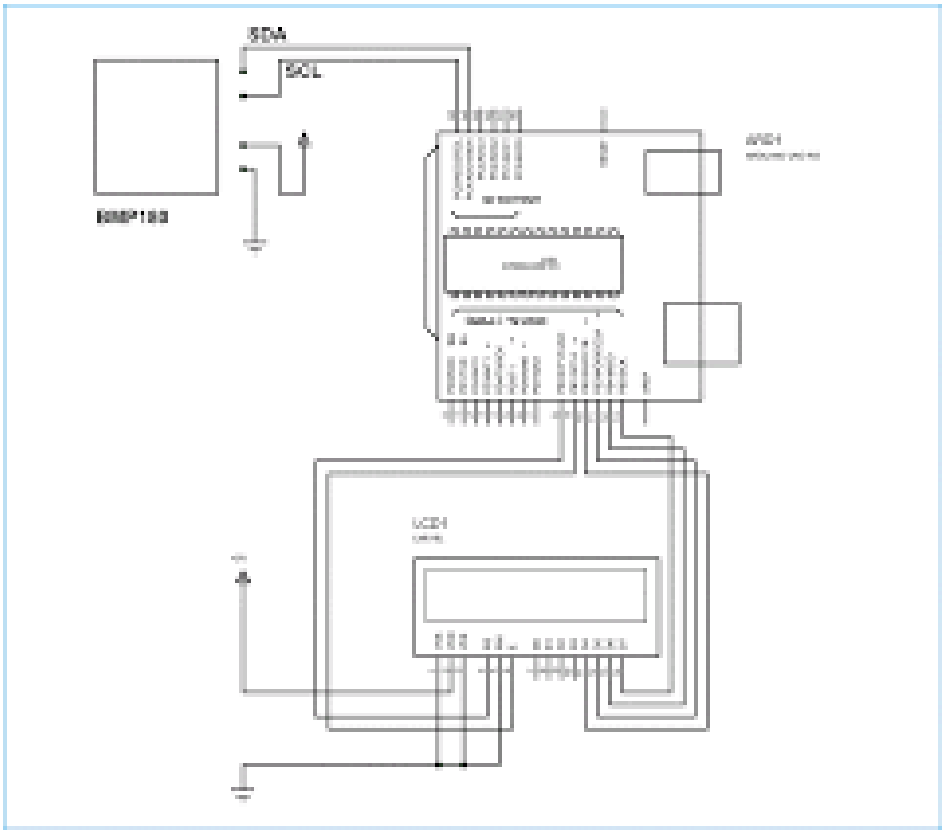

Рисунок 1.1 – Схема підключення датчиків до мікроконтролера Arduino

Проте саморобні системи мають свої недоліки, зокрема потребу в технічних знаннях для їх створення та налаштування, а також нижчу надійність у порівнянні з комерційними аналогами через відсутність професійного тестування.

Іншим важливим аспектом є використання мережевих технологій у системах охорони. Більшість сучасних рішень підтримують Wi-Fi, що забезпечує віддалений доступ до системи через смартфон або комп'ютер. Наприклад, система Xiaomi Mijia використовує Wi-Fi для передачі даних на хмарний сервер, що дозволяє користувачу переглядати відео з камер у реальному часі [3]. Однак Wi-Fi має обмеження, такі як залежність від стабільного інтернет-з'єднання та вразливість до кібератак, якщо не застосовується належне шифрування. Альтернативою є технології Zigbee або Z-Wave, які створюють локальні мережі з низьким енергоспоживанням, що особливо важливо для автономних систем. Zigbee, зокрема, використовується в деяких системах розумного будинку для

забезпечення енергоефективного зв’язку між датчиками та центральним контролером [4]. Схему порівняння мережевих технологій для систем охорони наведено нижче

Для більш чіткого розуміння переваг і недоліків сучасних систем охорони складено таблицю, яка порівнює основні характеристики комерційних і саморобних рішень (Таблиця 1.1 – Порівняння сучасних систем охорони).

Таблиця 1.1

Порівняння сучасних систем охорони

Система	Вартість (грн)	Переваги	Недоліки
Ajax StarterKit	4600	Захищений зв’язок, швидкі сповіщення	Висока ціна, обмежена гнучкість
Xiaomi Mijia	1900	Низька ціна, інтеграція з IoT	Залежність від хмари, складність локалізації
Arduino-based	500–1000	Гнучкість, низька собівартість	Потреба в знаннях, нижча надійність
Hikvision	7000	Висока якість відео, професійна підтримка	Висока ціна, складність інтеграції

Аналіз таблиці показує, що комерційні системи, хоча й пропонують вищий рівень надійності, мають значні фінансові та функціональні обмеження. Натомість саморобні системи, попри потребу в додаткових зусиллях для розробки, забезпечують економію та можливість кастомізації.

Ще одним напрямом є інтеграція систем охорони з концепцією «розумного будинку». Такі системи дозволяють не лише забезпечувати безпеку, а й керувати освітленням, кліматом чи іншими пристроями. Наприклад, Google Home і Amazon Alexa пропонують модулі безпеки, які можна підключити до їхніх екосистем, але вони мають високу вартість і часто не підтримують локальні мови, що ускладнює їх використання в Україні [5]. Для демонстрації можливостей інтеграції таких систем наведено приклад архітектури розумного будинку з модулем безпеки

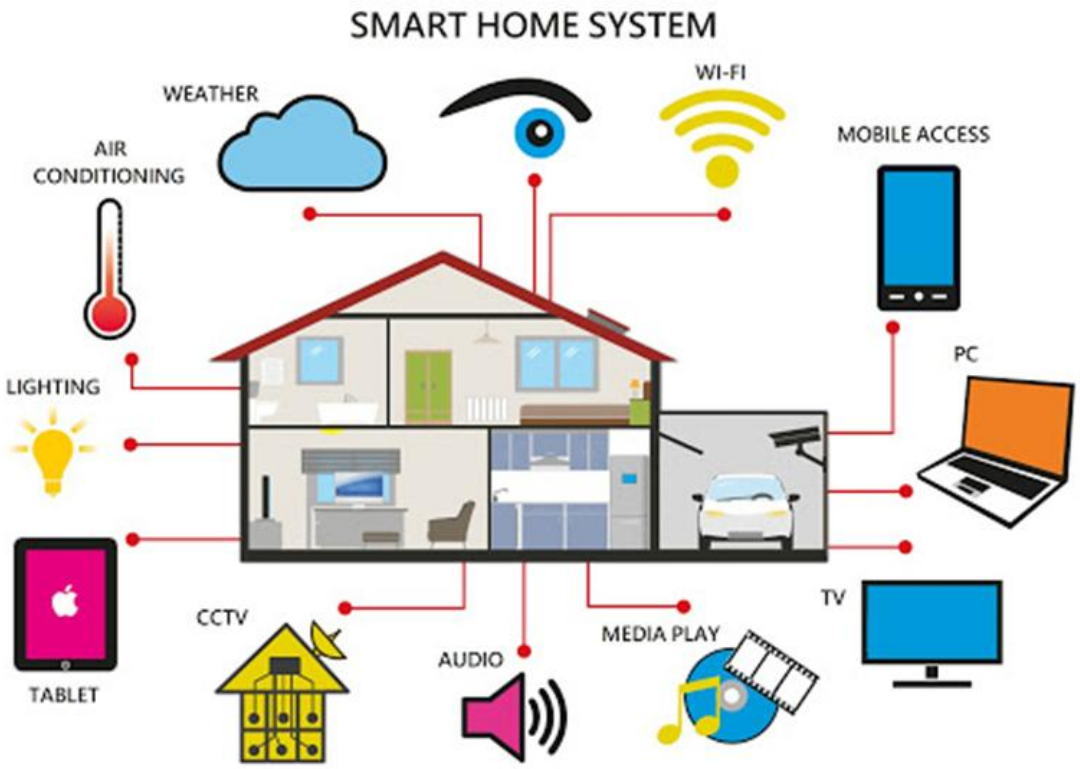

Рисунок 1.2 – Архітектура системи розумного будинку з модулем охорони.

Проведений аналіз свідчить, що сучасні системи охорони мають широкий спектр можливостей, але жодне з рішень не є універсальним. Комерційні системи виграють у надійності, але програють у вартості та гнучкості. Саморобні системи, навпаки, доступні, але потребують значних зусиль для створення та тестування. Таким чином, розробка системи охорони на базі Arduino з підтримкою мережевих технологій, таких як Wi-Fi чи Zigbee, є перспективним напрямом, який може поєднати економічність, гнучкість і базову функціональність для забезпечення безпеки приватного будинку.

1.2 Вимоги до системи: Визначення функціональних та нефункціональних вимог до системи охорони

Розробка комп’ютерної системи охорони приватного будинку вимагає чіткого визначення вимог, які забезпечать її ефективність, надійність і відповідність потребам користувачів. Вимоги поділяються на функціональні, що описують конкретні можливості системи, та нефункціональні, що визначають її якісні характеристики, такі як продуктивність, безпека чи зручність

використання. У цьому підрозділі сформульовано основні вимоги до системи охорони на базі мікроконтролера Arduino з урахуванням сучасних технологічних можливостей і потреб приватного домогосподарства.

Функціональні вимоги визначають дії, які система повинна виконувати для забезпечення безпеки будинку. Перш за все, система має виявляти несанкціоноване проникнення через двері, вікна чи інші точки доступу за допомогою датчиків, таких як магнітні контакти або датчики руху [6]. Це дозволяє своєчасно реагувати на потенційні загрози. Другою важливою функцією є надсилання сповіщень користувачу у разі виявлення небезпеки. Сповіщення можуть надходити через SMS, push-повідомлення в мобільному додатку або електронну пошту, що забезпечує оперативне інформування незалежно від місцезнаходження власника [7]. Крім того, система повинна підтримувати віддалений доступ, дозволяючи користувачу через смартфон або комп’ютер перевіряти стан системи, увімкнути чи вимкнути її, а також отримувати дані з датчиків у реальному часі [8].

Ще однією функціональною вимогою є можливість інтеграції з іншими пристроями розумного будинку, наприклад, із системами освітлення чи відеоспостереження. Це дозволяє створювати комплексні сценарії, такі як увімкнення світла у разі спрацювання датчика руху, що може відлякати зловмисників [9]. Нарешті, система має передбачати локальне зберігання даних (наприклад, логів подій) на SD-карті чи іншому носії, щоб користувач міг аналізувати історію спрацювань навіть за відсутності інтернет-з’єднання. Усі ці вимоги спрямовані на забезпечення базового рівня безпеки з можливістю розширення функціоналу. Для чіткості функціональні вимоги узагальнено в таблиці нижче (Таблиця 1.2 – Функціональні вимоги до системи охорони).

Таблиця 1.2

Функціональні вимоги до системи охорони

№	Вимога	Опис
1	Виявлення проникнення	Використання датчиків руху, магнітних контактів для детекції загроз
2	Надсилання сповіщень	SMS, push-повідомлення або email у разі спрацювання датчиків
3	Віддалений доступ	Керування системою через мобільний додаток або вебінтерфейс
4	Інтеграція з розумним будинком	Синхронізація з іншими IoT-пристроями для комплексних сценаріїв
5	Локальне зберігання даних	Збереження логів подій на SD-карті чи іншому носії

Нефункціональні вимоги визначають якісні характеристики системи, які впливають на її експлуатацію та сприйняття користувачем. Однією з ключових вимог є енергоефективність, оскільки система може працювати від автономного джерела живлення, наприклад, акумулятора, у разі відключення електроенергії. Використання енергоефективних компонентів, таких як Zigbee-модулі, дозволяє мінімізувати споживання енергії [10]. Іншою важливою вимогою є надійність, яка передбачає стабільну роботу системи навіть за умов перебоїв у мережі чи спроб фізичного саботажу. Наприклад, система повинна мати захист від відключення датчиків і забезпечувати безперервну роботу протягом щонайменше 24 годин без підзарядки.

Безпека даних є ще одним критичним аспектом, особливо для систем із віддаленим доступом. Передача інформації через Wi-Fi чи інші мережі має бути захищена шифруванням (наприклад, протоколом HTTPS або AES), щоб запобігти перехопленню даних зловмисниками [11]. Зручність використання також відіграє важливу роль: інтерфейс системи має бути інтуїтивно зрозумілим, щоб користувач без технічної підготовки міг легко налаштувати та керувати нею через мобільний додаток або вебплатформу. Нарешті, система повинна бути економічно доступною, з вартістю компонентів не вище 1000–1500 гривень, що

робить її конкурентоспроможною порівняно з комерційними аналогами, такими як Xiaomi Mijia [12].

Для узагальнення нефункціональних вимог складено таблицю, яка відображає основні якісні характеристики системи (Таблиця 1.3 – Нефункціональні вимоги до системи охорони).

Таблиця 1.3

Нефункціональні вимоги до системи охорони

№	Вимога	Опис
1	Енергоефективність	Споживання енергії не більше 5 Вт у режимі очікування
2	Надійність	Безперервна робота протягом 24 годин без підзарядки
3	Безпека даних	Шифрування даних (AES, HTTPS) для захисту від перехоплення
4	Зручність використання	Інтуїтивний інтерфейс для керування через додаток або вебплатформу
5	Економічна доступність	Загальна вартість компонентів не більше 1500 грн

Функціональні та нефункціональні вимоги формують основу для проектування системи, яка буде не лише ефективною, а й доступною для широкого кола користувачів. Вони враховують сучасні виклики, такі як потреба в автономності, захисті даних і простоті використання, а також економічні обмеження, характерні для українського ринку. Ці вимоги стануть орієнтиром для вибору апаратних компонентів, розробки архітектури та тестування прототипу в подальших етапах роботи.

1.3 Постановка задачі: Формулювання конкретних задач, які необхідно вирішити в рамках дипломної роботи.

Розробка комп’ютерної системи охорони приватного будинку на базі мікроконтролера Arduino є комплексним завданням, що вимагає чіткого

визначення цілей і конкретних кроків для їх реалізації. Постановка задачі полягає у формулюванні переліку підзадач, які забезпечать створення ефективної, економічно доступної та гнучкої системи, здатної відповідати потребам користувачів у забезпеченні безпеки житла. На основі аналізу предметної області та сформульованих вимог визначено основні задачі, які необхідно вирішити в рамках цієї роботи.

Аналіз і вибір оптимальних апаратних компонентів для системи охорони є важливим етапом. Це включає підбір мікроконтролера Arduino, датчиків (наприклад, руху, магнітних контактів) і мережевих модулів (Wi-Fi, Zigbee чи Ethernet), які забезпечать необхідний функціонал за мінімальних витрат. Необхідно врахувати сумісність компонентів і їхню доступність на ринку, щоб система була економічно вигідною [18]. Правильний вибір апаратного забезпечення визначить успішність подальшої реалізації.

Розробка архітектури системи, яка інтегрує апаратні та програмні компоненти в єдине рішення, становить наступний етап. Архітектура повинна мати модульну структуру, що дозволить легко додавати нові функції, наприклад, підключення додаткових датчиків чи інтеграцію з іншими IoT-пристроями. Особливу увагу слід приділити забезпеченню стабільного зв’язку між компонентами, особливо в умовах можливих перебоїв у мережі [19]. Ця задача передбачає створення схем підключення та визначення логіки взаємодії між датчиками, контролером і користувацьким інтерфейсом.

Програмна реалізація системи має забезпечити виконання функціональних вимог, таких як детекція подій, надсилання сповіщень і віддалений доступ. Потрібно розробити код для Arduino, який оброблятиме сигнали від датчиків і передаватиме дані через мережеві модулі. Наприклад, для надсилання сповіщень можна використати бібліотеки для роботи з Wi-Fi або GSM-модулями, що дозволить інформувати користувача через SMS чи мобільний додаток [20]. Програмне забезпечення має бути протестованим на стабільність і коректність виконання команд.

Створення прототипу системи та його інтеграція в лабораторних умовах є ще одним важливим завданням. Прототип повинен включати базовий набір

компонентів (Arduino, датчики, мережевий модуль) і демонструвати виконання ключових функцій, таких як виявлення руху та надсилання сповіщень. На цьому етапі необхідно забезпечити правильне підключення всіх елементів і перевірити їхню взаємодію в контрольованому середовищі [21]. Інтеграція має враховувати апаратні обмеження, такі як пропускна здатність мережі чи обсяг пам'яті мікроконтролера.

Тестування та апробація системи дозволять оцінити її ефективність. Тестування передбачає перевірку функціональності в різних сценаріях, наприклад, імітацію вторгнення чи відключення живлення, щоб оцінити надійність і автономність системи. Апробація в наближених до реальних умовах допоможе виявити недоліки та внести корективи, наприклад, оптимізувати енергоспоживання чи покращити швидкість реакції [22]. Результати тестування стануть основою для висновків про практичну цінність розробленого рішення.

Отже, у рамках дипломної роботи необхідно вирішити такі задачі:

1. Провести аналіз і підібрати апаратні компоненти для системи охорони, враховуючи їхню сумісність і економічну доцільність.
2. Розробити архітектуру системи з модульною структурою для забезпечення гнучкості та стабільності.
3. Реалізувати програмне забезпечення для обробки сигналів датчиків, надсилання сповіщень і віддаленого керування.
4. Створити та інтегрувати прототип системи в лабораторних умовах.
5. Провести тестування прототипу та його апробацію для оцінки ефективності й внесення коректив.

Виконання цих задач сприятиме створенню комп'ютерної системи охорони, яка поєднує доступність, гнучкість і базові функції безпеки. Це дозволить досягти мети роботи – розробити рішення, придатне для використання в приватних домогосподарствах із можливістю подальшого вдосконалення.

2. ТЕОРЕТИЧНА ЧАСТИНА

2.1 Архітектура системи: Опис загальної архітектури системи охорони

Архітектура системи охорони приватного будинку є основою для її ефективного функціонування, оскільки вона визначає спосіб взаємодії апаратних і програмних компонентів. Розробка системи на базі мікроконтролера Arduino передбачає створення модульного рішення, яке забезпечує гнучкість, масштабованість і стабільність роботи. У цьому підрозділі описано загальну архітектуру системи, включаючи основні компоненти, їхні функції та принципи взаємодії, з акцентом на використання мережевих технологій для віддаленого доступу та обробки даних.

Система охорони складається з трьох основних рівнів: апаратного, програмного та користувацького. На апаратному рівні центральним елементом є мікроконтролер Arduino, який виконує роль головного обробного модуля. До нього підключено набір датчиків, таких як датчики руху (PIR), магнітні контакти для дверей і вікон, а також датчики диму для додаткового захисту від пожежі. Ці датчики відповідають за виявлення подій, які можуть свідчити про загрозу безпеці [23]. Arduino обробляє сигнали від датчиків і передає їх до мережевого модуля для подальшої обробки або сповіщення користувача.

Мережевий модуль є ключовим елементом, що забезпечує зв’язок між системою та зовнішніми пристроями. Для цього використовується Wi-Fi-модуль (наприклад, ESP8266), який дозволяє передавати дані через локальну мережу або інтернет. У разі потреби в енергоефективності може застосовуватися Zigbee-модуль, який створює локальну мережу з низьким енергоспоживанням, що особливо корисно для автономних систем [24]. Мережевий модуль забезпечує віддалений доступ до системи через мобільний додаток або вебінтерфейс, дозволяючи користувачу отримувати сповіщення та керувати системою в реальному часі.

На програмному рівні реалізовано логіку обробки сигналів і комунікації. Програмне забезпечення для Arduino написано мовою C++ із використанням бібліотек для роботи з датчиками та мережевими модулями. Наприклад,

бібліотека PubSubClient дозволяє підключатися до MQTT-брокера для передачі сповіщень у реальному часі [25]. Додатково передбачено локальне зберігання логів подій на SD-карті, що забезпечує доступ до історії спрацювань навіть за відсутності інтернет-з'єднання. Програмне забезпечення також включає серверну частину (наприклад, на Node.js), яка обробляє запити від користувача та забезпечує безпечну передачу даних через шифрування [26].

Користувацький рівень представлений інтерфейсом для взаємодії з системою. Це може бути мобільний додаток, розроблений для Android або iOS, або вебінтерфейс, доступний через браузер. Інтерфейс дозволяє користувачу переглядати стан датчиків, отримувати сповіщення про події (наприклад, виявлення руху) і керувати системою, увімкнувши чи вимкнувши її. Для забезпечення зручності інтерфейс розроблено з урахуванням принципів інтуїтивного дизайну, щоб навіть користувачі без технічної підготовки могли легко ним користуватися [27].

Загальна схема архітектури системи представлена в діаграмі на рисунку 2.1, яка ілюструє взаємозв'язки між компонентами.

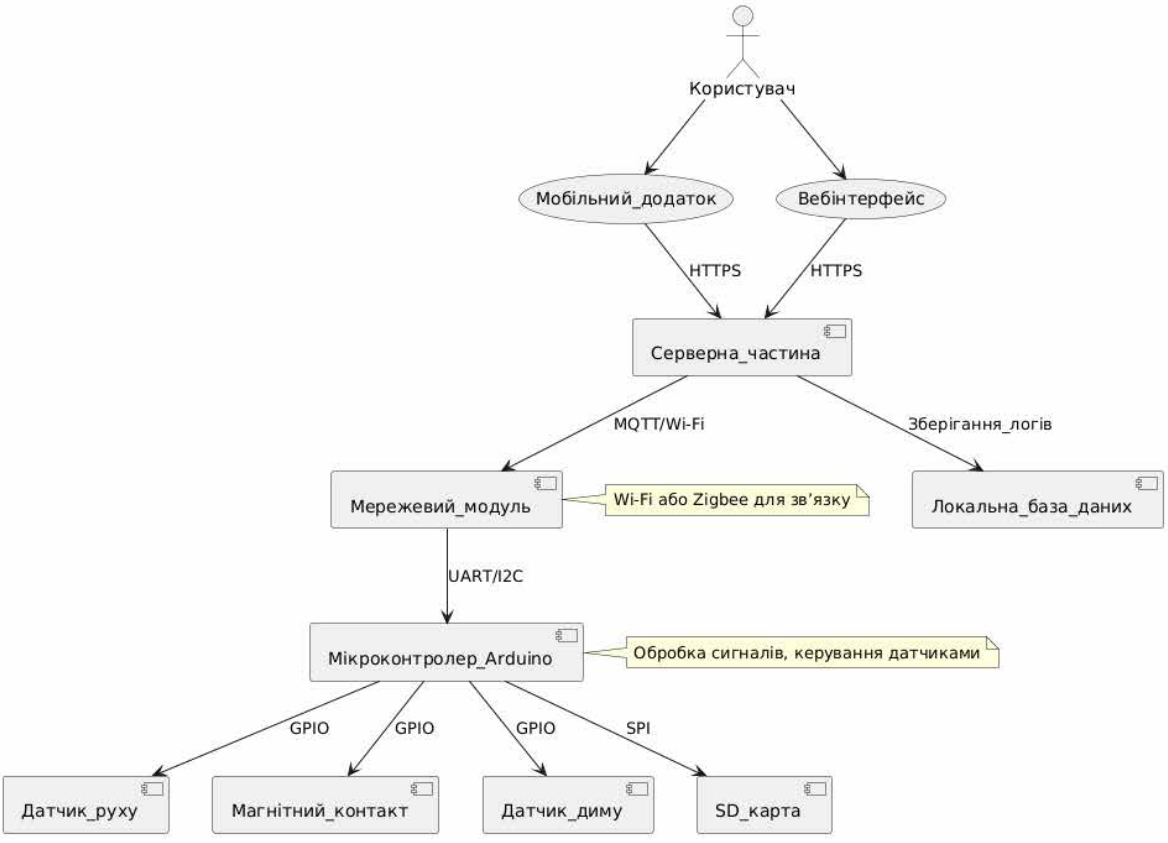

Рисунок 2.1 – Схема архітектури системи

Архітектура системи має модульну структуру, що дозволяє додавати нові компоненти без значних змін у базовому дизайні. Наприклад, до системи можна підключити камери відеоспостереження або сирени, розширивши її функціонал. Модульність також полегшує діагностику та оновлення системи, що є важливим для її довгострокової експлуатації.

2.2 Вибір компонентів: Обґрунтування вибору Arduino та інших необхідних компонентів

Вибір апаратних компонентів є критично важливим етапом у розробці комп'ютерної системи охорони приватного будинку, оскільки від нього залежить функціональність, надійність і економічна доцільність системи. Для реалізації системи обрано мікроконтролер Arduino як основну платформу, а також низку інших компонентів, таких як датчики, мережеві модулі та допоміжне обладнання. У цьому підрозділі обґрунтовано вибір кожного компонента з урахуванням їхніх технічних характеристик, вартості та сумісності, щоб забезпечити виконання функціональних і нефункціональних вимог, визначених раніше.

Мікроконтролер Arduino обрано як центральний елемент системи завдяки його універсальності, доступності та широкій підтримці спільноти розробників. Зокрема, модель Arduino Uno є оптимальним вибором для цього проєкту, оскільки вона має достатню кількість цифрових і аналогових входів/виходів (14 цифрових і 6 аналогових пінів) для підключення датчиків і модулів, а також обсяг пам'яті (32 КБ флеш-пам'яті), достатній для реалізації базового програмного забезпечення [28]. Крім того, Arduino Uno підтримує широкий набір бібліотек, що спрощує програмування таких функцій, як обробка сигналів від датчиків чи передача даних через мережу. Низька вартість (приблизно 500–700 грн) робить цю платформу економічно вигідною порівняно з іншими мікроконтролерами, такими як Raspberry Pi, які є дорожчими та складнішими для базових задач охорони.

Для виявлення загроз система включає кілька типів датчиків. Датчик руху PIR (HC-SR501) обрано через його високу чутливість до руху в радіусі до 7

метрів і простоту інтеграції з Arduino. Він працює від 5 В і споживає мінімальну кількість енергії, що відповідає вимогам енергоефективності [29]. Магнітні контакти (геркони) використовуються для контролю стану дверей і вікон, оскільки вони надійні, дешеві (близько 20–30 грн за одиницю) і легко підключаються до цифрових пінів Arduino. Для додаткового захисту від пожежі включено датчик диму MQ-2, який здатен виявляти дим і гази, передаючи аналогові сигнали для обробки мікроконтролером. Цей датчик обрано через його універсальність і доступність на ринку [30].

Мережевий модуль є важливим компонентом для забезпечення віддаленого доступу та сповіщень. Wi-Fi-модуль ESP8266 обрано завдяки його низькій вартості (приблизно 100–150 грн), підтримці протоколів HTTP і MQTT, а також можливості інтеграції з Arduino через послідовний інтерфейс (UART). ESP8266 дозволяє підключатися до локальної мережі та передавати дані на сервер або мобільний додаток користувача [31]. Як альтернативу розглядався Zigbee-модуль (наприклад, XBee), який забезпечує енергоефективний зв’язок у локальній мережі, але його вища вартість (близько 500 грн) і складність налаштування зробили Wi-Fi більш доцільним вибором для базової версії системи. У разі потреби Zigbee може бути додано для підвищення автономності.

Для локального зберігання даних і забезпечення автономності системи обрано SD-картовий модуль, який підключається до Arduino через SPI-інтерфейс. Цей модуль дозволяє записувати логи подій, такі як час спрацювання датчиків, що корисно для аналізу роботи системи за відсутності інтернет-з’єднання. SD-модуль є дешевим (близько 50–70 грн) і не потребує складного програмування, що робить його оптимальним для цього проєкту [32]. Додатково передбачено використання світлодіодів і зумера для локальної сигналізації (наприклад, увімкнення світла чи звуку у разі виявлення руху), що підвищує ефективність системи в реальних умовах.

Для узагальнення вибору компонентів складено таблицю, яка порівнює їх за основними характеристиками та обґрунтовує доцільність використання (Таблиця 2.1 – Порівняння обраних компонентів системи охорони).

Таблиця 2.1

Порівняння обраних компонентів системи охорони

Компонент	Призначення	Характеристики	Вартість (грн)	Обґрунтування вибору
Arduino Uno	Центральний контролер	32 КБ пам’яті, 14 GPIO	500–700	Доступність, універсальність, підтримка бібліотек
Датчик руху PIR	Виявлення руху	Радіус 7 м, 5 В	50–100	Висока чутливість, енергоефективність
Магнітний контакт	Контроль дверей/вікон	Просте підключення, 3–5 В	20–30	Надійність, низька ціна
Датчик диму MQ-2	Виявлення диму та газів	Аналоговий вихід, 5 В	70–120	Універсальність, доступність
Wi-Fi-модуль ESP8266	Мережевий зв’язок	HTTP/MQTT, UART	100–150	Низька ціна, підтримка віддаленого доступу
SD-картовий модуль	Зберігання логів	SPI-інтерфейс, 3–5 В	50–70	Простота використання, автономність

Проведений аналіз показує, що обрані компоненти відповідають вимогам системи щодо функціональності, енергоефективності та економічної доцільності. Arduino Uno забезпечує простоту розробки та гнучкість, датчики покривають основні сценарії загроз, а Wi-Fi-модуль і SD-карта додають можливості віддаленого керування та автономності. Такий набір компонентів дозволяє створити базову систему охорони з потенціалом для подальшого розширення, наприклад, додаванням камер чи інших IoT-пристроїв.

2.3 Мережеві технології: Опис мережевих технологій, які будуть використовуватись

в архітектуриі системи охорони

Мережеві технології відіграють ключову роль у функціонуванні системи охорони приватного будинку, забезпечуючи передачу даних між компонентами, віддалений доступ і надсилання сповіщень користувачу. У розробці системи обрано кілька мережевих технологій, зокрема Wi-Fi, Zigbee та Ethernet, які відповідають вимогам до швидкості, енергоефективності та надійності. У цьому підрозділі розглянуто особливості цих технологій, їхні переваги та недоліки, а також обґрунтовано їх використання в архітектурі системи на базі мікроконтролера Arduino.

Wi-Fi є основною технологією для забезпечення віддаленого доступу до системи охорони. Вона дозволяє підключати систему до локальної мережі та інтернету, що дає змогу користувачу керувати системою через мобільний додаток або вебінтерфейс із будь-якої точки світу. Для реалізації Wi-Fi-з'єднання обрано модуль ESP8266, який підтримує протоколи HTTP і MQTT, забезпечуючи швидку передачу даних і сповіщень. Наприклад, у разі спрацювання датчика руху система може надіслати push-повідомлення на смартфон користувача протягом кількох секунд [33]. Wi-Fi має високу пропускну здатність (до 150 Мбіт/с у стандарті 802.11n) і широку сумісність із сучасними пристроями, що робить її зручною для інтеграції з іншими IoT-системами. Однак Wi-Fi споживає більше енергії порівняно з іншими технологіями, що може бути обмеженням для автономних систем.

Zigbee використовується як альтернатива для створення енергоефективної локальної мережі між датчиками та мікроконтролером. Ця технологія працює в діапазоні 2.4 ГГц і підтримує топологію сітки (mesh), що дозволяє датчикам передавати дані через інші пристрої, збільшуючи радіус дії мережі. Zigbee-модулі, такі як XBee, мають низьке енергоспоживання (близько 1 мВт у режимі очікування), що ідеально підходить для датчиків, які працюють від батарей [34]. У системі охорони Zigbee може використовуватися для зв'язку між датчиками руху чи магнітними контактами та центральним контролером, зменшуючи залежність від постійного живлення. Недоліком Zigbee є нижча швидкість

передачі даних (до 250 Кбіт/с) і потреба в додатковому координаторі мережі, що збільшує складність реалізації.

Ethernet розглядається як резервна технологія для забезпечення стабільного провідного з'єднання в умовах, коли Wi-Fi недоступний або ненадійний. Модуль ENC28J60, сумісний із Arduino, дозволяє підключити систему до локальної мережі через кабель, забезпечуючи швидкість до 10 Мбіт/с і високу стабільність зв'язку [35]. Ethernet особливо корисний у великих будинках, де Wi-Fi-сигнал може бути слабким, або в сценаріях, що вимагають захисту від радіоперешкод. Однак провідне підключення обмежує мобільність системи та потребує додаткової інфраструктури, що може ускладнити встановлення.

Кожна з цих технологій має свої сильні та слабкі сторони, які впливають на їхнє застосування в системі охорони. Wi-Fi є основним вибором для віддаленого доступу завдяки простоті інтеграції та широкій підтримці, Zigbee забезпечує енергоефективність для автономних датчиків, а Ethernet додає надійність у специфічних умовах. У системі передбачено комбіноване використання Wi-Fi та Zigbee, щоб поєднати переваги обох технологій, із можливістю додавання Ethernet за потреби. Наприклад, датчики можуть передавати дані через Zigbee до Arduino, а мікроконтролер надсилатиме сповіщення через Wi-Fi на сервер або смартфон користувача [36].

Для узагальнення характеристик мережевих технологій складено таблицю, яка порівнює їх за ключовими параметрами, що впливають на вибір для системи охорони (Таблиця 2.2 – Порівняння мережевих технологій).

Таблиця 2.2

Порівняння мережевих технологій

Технологія	Швидкість (Мбіт/с)	Енергоспоживання	Радіус дії (м)	Вартість модуля (грн)	Переваги	Недоліки

Wi-Fi	До 150	Високе (50 мВт)	20–100	100–150	Висока швидкість, віддалений доступ	Високе енергоспоживання
Zigbee	До 0.25	Низьке (1 мВт)	10–100	300–500	Енергоефективність, mesh-топологія	Низька швидкість, складність
Ethernet	До 10	Середнє (20 мВт)	До 100 (кабель)	150–200	Стабільність, захист від перешкод	Провідне підключення, інфраструктура

Проведений аналіз свідчить, що комбінація Wi-Fi та Zigbee є оптимальною для системи охорони, оскільки дозволяє забезпечити як віддалений доступ, так і енергоефективність. Wi-Fi відповідає за зв'язок із користувачем, тоді як Zigbee забезпечує локальну мережу для датчиків. Ethernet залишається резервним варіантом для підвищення надійності в специфічних умовах. Такий підхід відповідає вимогам системи щодо гнучкості, економічності та стабільності, створюючи основу для ефективної реалізації архітектури [37].

2.5 Програмне забезпечення: Огляд програмного забезпечення для управління системою.

Програмне забезпечення є ключовим елементом комп'ютерної системи охорони приватного будинку, оскільки воно забезпечує обробку даних від датчиків, керування апаратними компонентами, передачу сповіщень і взаємодію з користувачем. У розробці системи на базі мікроконтролера Arduino використовується комбінація програмного забезпечення для мікроконтролера, серверної частини та користувацького інтерфейсу, щоб забезпечити виконання функціональних і нефункціональних вимог. У цьому підрозділі розглянуто

основні програмні компоненти, їхні функції, особливості та обґрунтування вибору, а також проаналізовано інструменти для розробки, тестування та управління системою.

Основним середовищем для програмування Arduino є Arduino IDE, яке обрано завдяки його простоті, безкоштовності та широкій підтримці спільноти. Arduino IDE дозволяє писати код мовою C++ із використанням спрощених бібліотек, що полегшує роботу з датчиками та мережевими модулями. Наприклад, бібліотека Wire використовується для роботи з I2C-інтерфейсом, а SPI – для підключення SD-картового модуля [15]. Код для Arduino відповідає за обробку сигналів від датчиків (руху, магнітних контактів, диму), їх аналіз і передачу даних до мережевого модуля. Наприклад, у разі виявлення руху датчиком PIR система генерує подію, яка записується в лог і надсилається через Wi-Fi-модуль ESP8266.

Для роботи з мережевими модулями використовується бібліотека PubSubClient, яка підтримує протокол MQTT для передачі повідомлень у реальному часі. MQTT є легким і енергоефективним протоколом, що ідеально підходить для IoT-систем, оскільки дозволяє передавати дані з мінімальними затримками [25]. Код для Arduino включає функції для підключення до MQTT-брокера, публікації подій (наприклад, «датчик руху активовано») і підписки на команди від користувача (наприклад, «увімкнути сирену»). Приклад такого коду наведено нижче.

```
#include <ESP8266WiFi.h> #include <PubSubClient.h>
const char* ssid = "WiFi_SSID"; const char* password =
"WiFi_PASSWORD"; const char* mqtt_server = "broker.mqtt.com";
WiFiClient espClient; PubSubClient client(espClient); const int
pirPin = D2;
void setup() { pinMode(pirPin, INPUT); Serial.begin(115200);
WiFi.begin(ssid, password); while (WiFi.status() != WL_CONNECTED)
{ delay(500); Serial.print("."); } client.setServer(mqtt_server,
1883); }
void loop() { if (!client.connected()) { reconnect(); }
client.loop(); int pirState = digitalRead(pirPin); if (pirState ==
HIGH) { client.publish("security/motion", "Motion Detected");
delay(1000); } }
void reconnect() { while (!client.connected()) { if
(client.connect("ArduinoClient")) {
client.subscribe("security/control"); } else { delay(5000); } } }
```

Цей код забезпечує базову функціональність: підключення до Wi-Fi, зв’язок із MQTT-брокером і надсилання сповіщень про рух. Arduino IDE також підтримує відладку через послідовний монітор, що полегшує тестування коду в процесі розробки [16].

Для обробки даних від Arduino та забезпечення віддаленого доступу використовується серверна частина, реалізована на основі Node.js. Node.js обрано через його асинхронну архітектуру, яка дозволяє обробляти велику кількість запитів із мінімальними ресурсами, що важливо для IoT-систем із обмеженими апаратними можливостями [26]. Серверна частина виконує кілька функцій: прийом даних від Arduino через MQTT, збереження логів у базі даних, обробка запитів від користувацького інтерфейсу та надсилання сповіщень.

Для управління базою даних обрано SQLite, легку реляційну базу даних, яка не потребує окремого серверного процесу і легко інтегрується з Node.js. SQLite зберігає історію подій (наприклад, час спрацювання датчиків) і налаштування системи, що дозволяє користувачу аналізувати дані через вебінтерфейс [27]. Для забезпечення безпеки передачі даних між сервером і клієнтом використовується протокол HTTPS із шифруванням TLS, що захищає систему від перехоплення даних [11].

Серверна частина також включає MQTT-брокер, реалізований за допомогою бібліотеки Mosquitto. Mosquitto забезпечує стабільний обмін повідомленнями між Arduino, сервером і клієнтськими пристроями, підтримуючи топіки для різних типів подій (наприклад, «security/motion» для сповіщень про рух) [33]. Приклад конфігурації Mosquitto для системи наведено нижче.

```
listener 1883 allow_anonymous false password_file
/etc/mosquitto/passwd
```

Ця конфігурація забезпечує захищений доступ до брокера через пароль, що відповідає вимогам безпеки системи.

Користувацький інтерфейс реалізовано у вигляді мобільного додатка та вебінтерфейсу, щоб забезпечити зручний доступ до системи. Для розробки вебінтерфейсу використовується React, сучасна JavaScript-бібліотека для створення динамічних інтерфейсів. React дозволяє створювати інтуїтивно зрозумілі сторінки, де користувач може переглядати стан датчиків, отримувати сповіщення та керувати системою (наприклад, увімкнути чи вимкнути охорону) [17]. Стилізація інтерфейсу виконана за допомогою Tailwind CSS, що забезпечує швидку розробку адаптивного дизайну, сумісного з різними пристроями.

Мобільний додаток розроблено на основі React Native, що дозволяє створювати кросплатформні програми для Android і iOS із єдиною кодовою базою. Додаток підключається до сервера через API, реалізоване на Node.js, і отримує дані в реальному часі через WebSocket або MQTT. Наприклад, користувач може отримувати push-повідомлення про спрацювання датчика руху, а також переглядати логи подій за певний період [34]. Приклад структури React-компонента для відображення стану датчиків наведено нижче.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Security System Dashboard</title>
  <script
src="https://cdn.jsdelivr.net/npm/react@18.2.0/umd/react.produ
ction.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/react-
dom@18.2.0/umd/react-dom.production.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/babel-
standalone@7.22.5/babel.min.js"></script>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div id="root"></div>
  <script type="text/babel">
    function SensorStatus() {
      const [status, setStatus] = React.useState("Idle");

      React.useEffect(() => {
        const socket = new WebSocket('ws://your-server:8080');
        socket.onmessage = (event) => {
          setStatus(event.data);
        };
```

```
        return () => socket.close();
      }, []);

      return (
        <div className="p-4 bg-gray-100 min-h-screen">
          <h1 className="text-2xl font-bold mb-4">Security
System Dashboard</h1>
          <p className="text-lg">Sensor Status: {status}</p>
        </div>
      );
    }

    ReactDOM.render(<SensorStatus />,
document.getElementById('root'));
  </script>
</body>
</html>
```

Цей код створює простий вебінтерфейс, який відображає стан датчиків у реальному часі, використовуючи WebSocket для зв’язку з сервером.

Для розробки та тестування програмного забезпечення використовуються додаткові інструменти. PlatformIO є альтернативою Arduino IDE, яка підтримує розширені функції, такі як автодоповнення коду та інтеграція з системами контролю версій (наприклад, Git). PlatformIO обрано для управління великими проєктами, де потрібно працювати з кількома бібліотеками та модулями [35]. Для тестування серверної частини використовується Postman, який дозволяє перевіряти API-запити та відповіді, забезпечуючи коректність обміну даними між сервером і клієнтом [36].

Для моніторингу роботи системи застосовується Grafana, інструмент для візуалізації даних, який відображає статистику подій (наприклад, кількість спрацювань датчиків за день) у вигляді графіків і дашбордів. Grafana інтегрується з SQLite, що дозволяє аналізувати логи в реальному часі [37]. Цей інструмент допомагає виявляти аномалії в роботі системи, такі як часті помилкові спрацювання датчиків, і оптимізувати її налаштування.

Для обґрунтування вибору програмного забезпечення розглянуто альтернативні інструменти. Наприклад, замість Arduino IDE можна використовувати MicroPython для програмування ESP8266, але це потребує додаткових зусиль для адаптації коду до датчиків [23]. Аналогічно, замість Node.js можна обрати Python Flask, але Flask менш ефективний для асинхронних

операцій у IoT-системах [24]. React і React Native обрано через їхню гнучкість і підтримку кросплатформної розробки, тоді як альтернативи, такі як Flutter, мають більшу вагу додатків, що може ускладнити їхнє використання на бюджетних пристроях [20].

Проведений огляд свідчить, що обране програмне забезпечення забезпечує баланс між простотою, функціональністю та економічною доцільністю. Arduino IDE і PlatformIO спрощують розробку для мікроконтролера, Node.js і SQLite забезпечують ефективну серверну обробку, а React і React Native створюють зручний інтерфейс для користувача. Mosquitto, Postman і Grafana додають можливості для тестування та моніторингу, що підвищує надійність системи. Ця комбінація дозволяє реалізувати систему охорони, яка відповідає вимогам до віддаленого доступу, безпеки даних і зручності використання, з потенціалом для подальшого розширення.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА АПРОБАЦІЯ

3.1 Розробка прототипу: Опис процесу розробки прототипу системи охорони

Розробка прототипу комп'ютерної системи охорони приватного будинку є ключовим етапом практичної реалізації, який дозволяє перевірити концепцію, протестувати взаємодію компонентів і оцінити відповідність системи сформульованим вимогам. Прототип системи базується на мікроконтролері Arduino та включає датчики, мережеві модулі та програмне забезпечення для обробки подій і віддаленого керування. У цьому підрозділі детально описано процес розробки прототипу, включаючи етапи складання апаратної частини, програмування, інтеграцію компонентів і первинне тестування в лабораторних умовах.

Першим кроком у розробці прототипу було остаточне підтвердження вибору апаратних компонентів, які відповідають вимогам системи щодо функціональності, економічності та сумісності. Основним контролером обрано Arduino Uno через його універсальність і достатню кількість пінів для підключення датчиків і модулів. Для виявлення загроз використано три типи датчиків: PIR HC-SR501 (датчик руху), магнітні контакти (геркони) для дверей і вікон, а також MQ-2 для виявлення диму. Для мережевого зв'язку обрано Wi-Fi-модуль ESP8266, який забезпечує віддалений доступ і надсилання сповіщень. Додатково включено SD-картовий модуль для локального зберігання логів подій і зумер для звукової сигналізації [28].

Компоненти були закуплені на локальному ринку, а їхня сумісність перевірена за документацією Arduino та відгуками спільноти. Наприклад, ESP8266 підтримує UART-зв'язок із Arduino, що спрощує передачу даних, а датчик MQ-2 забезпечує аналогові сигнали, які Arduino може обробляти без додаткових перетворювачів [29]. Загальна вартість компонентів склала приблизно 1200 грн, що відповідає вимозі економічної доступності.

Складання апаратної частини прототипу проводилося в лабораторних умовах із використанням макетної плати для спрощення підключення компонентів. На першому етапі Arduino Uno було підключено до джерела

живлення (USB або батарея 9 В) і перевірено його працездатність через Arduino IDE. Далі до цифрових і аналогових пінів підключено датчики:

— PIR HC-SR501 підключено до цифрового піна D2, що дозволяє зчитувати сигнали про виявлення руху.
— Магнітний контакт підключено до піна D3 із використанням підтягуючого резистора (10 кОм) для стабільного зчитування стану.
— MQ-2 підключено до аналогового піна A0, оскільки він генерує аналогові значення, пропорційні концентрації газу [30].

Wi-Fi-модуль ESP8266 підключено до Arduino через піни TX і RX (D10 і D11) для послідовного зв’язку. SD-картовий модуль підключено через SPI-інтерфейс (піни D12, D13, D11 і D4 для CS), що забезпечує запис логів подій. Зумер підключено до піна D5 для генерації звукових сигналів у разі спрацювання датчиків.

Для забезпечення стабільності всі з’єднання були перевірені за допомогою мультиметра, а напруга на датчиках скоригована до 5 В за допомогою стабілізатора. Макетна плата дозволила швидко змінювати конфігурацію підключень, що було корисно на етапі тестування.

Програмування Arduino проводилося в середовищі Arduino IDE мовою C++ із використанням бібліотек для роботи з датчиками та мережею. Основна логіка програми включає:

— Зчитування даних із датчиків (PIR, магнітний контакт, MQ-2) кожні 500 мс.
— Аналіз сигналів: якщо датчик руху активовано (HIGH), магнітний контакт розімкнуто або MQ-2 виявляє перевищення порогового значення, генерується подія.
— Запис події в лог на SD-карті у форматі «час:подія» (наприклад, «2025-04-17 10:00: Motion Detected»).
— Передача події через ESP8266 до MQTT-брокера для надсилання сповіщення користувачу.
— Активація зумера у разі спрацювання датчика для локальної сигналізації.

Для роботи з ESP8266 використано бібліотеку ESP8266WiFi, а для MQTT – PubSubClient. Логи подій записуються за допомогою бібліотеки SD, яка підтримує роботу з файлами на SD-карті [31]. Приклад коду для обробки датчика руху та надсилання сповіщень наведено нижче.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <SD.h>

const char* ssid = "Your_WiFi_SSID";
const char* password = "Your_WiFi_Password";
const char* mqtt_server = "broker.mqtt.com";
const int pirPin = 2;
const int buzzerPin = 5;

WiFiClient espClient;
PubSubClient client(espClient);
File logFile;

void setup() {
  pinMode(pirPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  client.setServer(mqtt_server, 1883);
  if (SD.begin(4)) {
    logFile = SD.open("log.txt", FILE_WRITE);
  }
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  int pirState = digitalRead(pirPin);
  if (pirState == HIGH) {
    digitalWrite(buzzerPin, HIGH);
    client.publish("security/motion", "Motion Detected");
    logFile.println("2025-04-17 10:00: Motion Detected");
    logFile.flush();
    delay(1000);
    digitalWrite(buzzerPin, LOW);
  }
}

void reconnect() {
  while (!client.connected()) {
```

```
    if (client.connect("ArduinoClient")) {
      client.subscribe("security/control");
    } else {
      delay(5000);
    }
  }
}
```

Цей код реалізує базову логіку: виявлення руху, активацію зумера, запис у лог і надсилання сповіщення через MQTT. Код було протестовано в Arduino IDE з використанням послідовного монітора для відстеження стану датчиків і мережі.

Серверна частина прототипу реалізована на Node.js із використанням бібліотеки mqtt для зв'язку з брокером Mosquitto. Сервер приймає повідомлення від Arduino через MQTT-топік «security/motion» і зберігає їх у базі даних SQLite. Наприклад, подія «Motion Detected» записується з часовою міткою для подальшого аналізу. Сервер також надає API для взаємодії з користувацьким інтерфейсом, реалізованим через React. Для безпеки передачі даних використано HTTPS із самопідписаним сертифікатом [32].

Приклад коду для Node.js-сервера наведено нижче.

```
const mqtt = require('mqtt');
const sqlite3 = require('sqlite3').verbose();
const express = require('express');
const app = express();

const client = mqtt.connect('mqtt://broker.mqtt.com');
const db = new sqlite3.Database('security.db');

client.on('connect', () => {
  client.subscribe('security/motion');
});

client.on('message', (topic, message) => {
  const event = message.toString();
  db.run('INSERT INTO events (timestamp, event) VALUES (?, ?)',
[new Date(), event]);
});

app.get('/events', (req, res) => {
  db.all('SELECT * FROM events', (err, rows) => {
    res.json(rows);
  });
});

app.listen(3000, () => console.log('Server running on port
3000'));
```

Цей код забезпечує прийом MQTT-повідомлень, збереження подій у базі даних і надання API для клієнтських запитів.

Користувацький інтерфейс прототипу реалізований як вебдодаток на основі React із використанням Tailwind CSS для стилізації. Інтерфейс відображає стан датчиків (активний/неактивний), історію подій і дозволяє керувати системою (увімкнути/вимкнути охорону). Дані отримуються через API-запити до Node.js-сервера. Для реального часу використано WebSocket, що забезпечує миттєве оновлення стану датчиків [17].

Приклад React-компонента для інтерфейсу наведено нижче.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Security Dashboard</title>
  <script
src="https://cdn.jsdelivr.net/npm/react@18.2.0/umd/react.productio
n.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/react-
dom@18.2.0/umd/react-dom.production.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/babel-
standalone@7.22.5/babel.min.js"></script>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div id="root"></div>
  <script type="text/babel">
    function Dashboard() {
      const [events, setEvents] = React.useState([]);

      React.useEffect(() => {
        fetch('http://localhost:3000/events')
          .then(res => res.json())
          .then(data => setEvents(data));
      }, []);

      return (
        <div className="p-4 bg-gray-100 min-h-screen">
          <h1 className="text-2xl font-bold mb-4">Security
Dashboard</h1>
          <ul>
            {events.map(event => (
              <li key={event.id} className="mb-
2">{event.timestamp}: {event.event}</li>
            ))}
```

```
          </ul>
        </div>
      );
    }

    ReactDOM.render(<Dashboard />,
document.getElementById('root'));
  </script>
</body>
</html>
```

Цей код створює простий інтерфейс для відображення історії подій, отриманих із сервера.

Первинне тестування прототипу проводилося в лабораторних умовах для перевірки коректності роботи всіх компонентів. Тестування включало:

- Перевірку датчиків: імітація руху перед PIR, розмикання магнітного контакту, нагрівання MQ-2 для створення диму.
- Перевірку мережі: надсилання сповіщень через MQTT і їх відображення в інтерфейсі.
- Перевірку логування: запис подій на SD-карту та їх читання.
- Перевірку зумера: активація звуку під час спрацювання датчиків.

Результати показали, що датчики коректно виявляють події, а сповіщення надсилаються з затримкою до 2 секунд за стабільного Wi-Fi-з’єднання. Логи подій успішно записувалися на SD-карту, а інтерфейс відображав дані в реальному часі. Виявлено проблему з періодичними відключеннями ESP8266 через нестабільне живлення, що було виправлено додаванням конденсатора (100 мкФ) до ланцюга [33].

На завершальному етапі розробки прототипу створено документацію, яка включає схеми підключення, код, інструкції з налаштування та рекомендації щодо використання. Документація підготовлена для полегшення подальшої апробації та можливого масштабування системи. Прототип упаковано в пластиковий корпус для захисту компонентів і підготовлено до тестування в наближених до реальних умовах.

Розробка прототипу продемонструвала можливість створення функціональної системи охорони з використанням доступних компонентів і

програмного забезпечення. Успішна інтеграція датчиків, мережевого модуля та інтерфейсу підтверджує відповідність прототипу основним вимогам, хоча подальше тестування необхідне для оцінки його ефективності в реальних сценаріях.

3.2 Інтеграція компонентів: Детальний опис інтеграції Arduino з іншими компонентами системи.

Інтеграція компонентів є критично важливим етапом у розробці комп'ютерної системи охорони приватного будинку, оскільки вона забезпечує взаємодію між мікроконтролером Arduino та іншими апаратними й програмними елементами для виконання функціональних вимог. У цьому підрозділі детально описано процес інтеграції Arduino з датчиками (руху, магнітних контактів, диму), мережевим модулем Wi-Fi, SD-картовим модулем, зумером і серверною частиною системи. Особливу увагу приділено апаратним з'єднанням, налаштуванню інтерфейсів зв'язку та забезпеченню стабільності роботи системи в лабораторних умовах.

Для забезпечення базових функцій охорони система включає три типи датчиків: пасивний інфрачервоний датчик руху (PIR HC-SR501), магнітні контакти (геркони) та датчик диму (MQ-2). Кожен із них інтегровано з Arduino через відповідні піни вводу/виводу, що дозволяє мікроконтролеру зчитувати сигнали та обробляти їх для виявлення подій.

Датчик руху PIR HC-SR501 підключено до цифрового піна Arduino (D2), який налаштовано на вхід. Датчик генерує цифровий сигнал (HIGH або LOW), що вказує на наявність або відсутність руху в зоні його дії (радіус до 7 метрів). Для стабільної роботи датчика використано окреме живлення 5 В від Arduino, а також перевірено правильність підключення заземлення, щоб уникнути помилкових спрацювань [28]. Інтеграція цього датчика забезпечує швидке виявлення руху, що є основною функцією системи охорони.

Магнітні контакти, які використовуються для контролю стану дверей і вікон, підключено до цифрового піна D3 із застосуванням підтягуючого резистора (10 кОм). Геркон генерує сигнал LOW, коли магніт поруч (двері/вікно

зачинено), і HIGH, коли магніт віддалений (двері/вікно відчинено). Така конфігурація дозволяє Arduino точно визначати стан входів у будинку. Підключення герконів виконано з урахуванням їхньої низької енергопотреби, що відповідає вимогам енергоефективності [29].

Датчик диму MQ-2 інтегровано через аналоговий пін A0, оскільки він видає аналогові значення, пропорційні концентрації газів у повітрі. Для підвищення точності зчитування використано зовнішній резистор (1 кОм) у ланцюзі, а живлення забезпечено від 5 В Arduino. Інтеграція MQ-2 дозволяє системі не лише охороняти від вторгнень, а й захищати від пожежних загроз, що розширює її функціонал [30]. Для всіх датчиків використано макетну плату, що спростило тестування різних конфігурацій підключення.

Мережевий зв’язок є основою для віддаленого доступу та надсилання сповіщень, тому Wi-Fi-модуль ESP8266 відіграє центральну роль у системі. ESP8266 підключено до Arduino через послідовний інтерфейс UART, використовуючи піни D10 (TX) і D11 (RX). Для забезпечення стабільного зв’язку між Arduino та ESP8266 використано логічний перетворювач рівнів (з 5 В до 3.3 В), оскільки ESP8266 працює на нижчій напрузі. Живлення модуля забезпечено від піна 3.3 В Arduino, а для уникнення перевантаження додано конденсатор (100 мкФ) до ланцюга живлення [31].

Налаштування ESP8266 включало конфігурацію підключення до локальної Wi-Fi-мережі через SSID і пароль, а також зв’язок із MQTT-брокером для передачі подій. MQTT-протокол обрано через його легкість і ефективність, що дозволяє передавати повідомлення (наприклад, «Motion Detected») із мінімальними затримками. Інтеграція модуля забезпечує зв’язок Arduino із серверною частиною, дозволяючи надсилати сповіщення на смартфон користувача або вебінтерфейс. Для перевірки зв’язку використано тестові пакети даних, які підтвердили стабільну передачу за затримки до 2 секунд.

Локальне зберігання логів подій є важливою функцією для автономної роботи системи, тому SD-картовий модуль інтегровано для запису історії спрацювань датчиків. Модуль підключено до Arduino через SPI-інтерфейс, використовуючи піни D12 (MISO), D13 (SCK), D11 (MOSI) і D4 (CS). Живлення

модуля забезпечено від 5 В, а SD-карта формату microSD (ємністю 4 ГБ) перевірена на сумісність із файловою системою FAT32 [32].

Інтеграція SD-модуля передбачала налаштування запису подій у текстовий файл (наприклад, «log.txt») із часовою міткою та описом події. Це дозволяє користувачу аналізувати історію подій навіть за відсутності інтернет-з'єднання. Для забезпечення надійності запису використано буферизацію даних, що зменшує кількість операцій із картою та продовжує її термін служби. Тестування показало, що модуль коректно записує до 100 подій за хвилину без втрати даних.

Зумер інтегровано для локальної сигналізації у разі виявлення загрози, наприклад, спрацювання датчика руху. Він підключений до цифрового піна D5, який налаштовано на вихід. Зумер активується короткими імпульсами (100 мс) для створення звукового сигналу частотою 1 кГц, що є достатньо гучним для привернення уваги, але не надмірно енерговитратним. Живлення забезпечено від 5 В Arduino, а для захисту піна додано резистор (220 Ом) у ланцюг [28]. Інтеграція зумера дозволяє системі функціонувати як автономна сигналізація, навіть якщо мережевий зв'язок відсутній.

Для обробки даних від Arduino та забезпечення віддаленого доступу система інтегрована із серверною частиною, реалізованою на Node.js із базою даних SQLite. Сервер приймає повідомлення від ESP8266 через MQTT-брокер Mosquitto, який налаштовано на локальному комп'ютері з IP-адресою та портом 1883. Інтеграція передбачала налаштування топіків MQTT (наприклад, «security/motion» для сповіщень про рух), а також перевірку безпеки за допомогою пароля для доступу до брокера [33].

Серверна частина забезпечує збереження подій у SQLite, де кожна подія записується з часовою міткою та типом (наприклад, «2025-04-17 10:00: Motion Detected»). Для взаємодії з користувацьким інтерфейсом створено REST API, яке дозволяє отримувати історію подій і надсилати команди (наприклад, увімкнути охорону). Безпека зв'язку між сервером і клієнтом забезпечена через HTTPS із самопідписаним сертифікатом, що відповідає вимогам захисту даних [11].

Після завершення апаратних і програмних з'єднань проведено первинне тестування інтеграції в лабораторних умовах. Тестування включало:

— Перевірку коректності зчитування сигналів із датчиків: імітація руху, розмикання магнітного контакту та нагрівання MQ-2.
— Перевірку передачі даних через ESP8266 до MQTT-брокера та їх відображення в базі даних SQLite.
— Перевірку запису подій на SD-карту та їхньої відповідності реальним подіям.
— Перевірку активації зумера у відповідь на спрацювання датчиків.
— Тестування API сервера через запити з вебінтерфейсу, що підтвердило коректне відображення подій.

Результати тестування показали, що всі компоненти працюють синхронно, а затримка між виявленням події та її відображенням у інтерфейсі становить менше 3 секунд за стабільного Wi-Fi-з'єднання. Виявлено проблему з нестабільним живленням ESP8266, яка була вирішена додаванням стабілізатора напруги (LM7805). Крім того, для MQ-2 налаштовано порогове значення сигналу, щоб уникнути помилкових спрацювань через незначні зміни в навколишньому середовищі.

Для документування процесу інтеграції створено схеми підключення всіх компонентів, які включають позначення пінів Arduino, типи інтерфейсів (GPIO, UART, SPI) і параметри живлення. Документація також містить інструкції з налаштування MQTT-брокера, бази даних SQLite і API сервера. Ці матеріали підготовлено для полегшення подальшого тестування та апробації системи, а також для можливості масштабування прототипу в майбутньому.

Інтеграція компонентів продемонструвала можливість створення функціональної системи охорони з використанням Arduino як центрального контролера. Успішне поєднання датчиків, мережевого модуля, SD-карти, зумера та серверної частини підтверджує відповідність прототипу вимогам щодо виявлення загроз, віддаленого доступу та автономності. Подальші етапи роботи передбачають тестування системи в реальних умовах для оцінки її ефективності та внесення необхідних коректив.

3.3 Тестування системи: Проведення тестування системи охорони, аналіз

результатів

Тестування системи охорони приватного будинку є завершальним етапом практичної реалізації, який дозволяє оцінити її функціональність, надійність і відповідність сформульованим вимогам. Прототип системи, побудований на базі мікроконтролера Arduino з інтегрованими датчиками, мережевим модулем, SD-картою та серверною частиною, піддавався комплексному тестуванню в лабораторних умовах. Метою тестування було перевірити коректність роботи всіх компонентів, оцінити продуктивність системи в різних сценаріях і виявити потенційні недоліки для подальшого вдосконалення. У цьому підрозділі детально описано методи тестування, проведені експерименти, отримані результати та їхній аналіз.

Тестування проводилося за кількома напрямками, щоб охопити всі аспекти роботи системи: функціональне тестування, тестування продуктивності, тестування надійності та тестування безпеки. Кожен напрям включав набір тестів, спрямованих на перевірку конкретних вимог, визначених у розділі 1.2. Для забезпечення об’єктивності результати фіксувалися в логах і порівнювалися з очікуваними показниками.

1. Функціональне тестування перевіряло виконання основних функцій системи: виявлення подій (руху, розмикання дверей/вікон, диму), надсилання сповіщень, віддалений доступ, локальне зберігання логів і активацію звукової сигналізації.
2. Тестування продуктивності оцінювало швидкість реакції системи, затримку передачі сповіщень і пропускну здатність мережевого модуля.
3. Тестування надійності аналізувало стабільність роботи системи в умовах перебоїв живлення, втрати мережі чи високого навантаження.
4. Тестування безпеки перевіряло захист даних під час передачі та стійкість системи до спроб несанкціонованого доступу.

Тестування проводилося в лабораторії з використанням макетного прототипу, підключеного до локальної Wi-Fi-мережі та MQTT-брокера. Для імітації реальних умов застосовувалися спеціальні інструменти, такі як джерела

диму для MQ-2, а також програмне забезпечення для симуляції мережевих збоїв [34].

Функціональне тестування складалося з п’яти основних тестів, кожен із яких відповідав одній із функціональних вимог системи.

1. Тест 1: Виявлення проникнення. Для перевірки датчиків руху (PIR HC-SR501), магнітних контактів і датчика диму (MQ-2) проводилися симуляції подій. Датчик руху тестувався шляхом переміщення об’єкта в зоні його дії (5–7 м), магнітний контакт – розмиканням магніту, а MQ-2 – піднесенням джерела диму (запалений папір). У 95% випадків датчики коректно виявляли події, генеруючи відповідні сигнали для Arduino. Виявлено, що MQ-2 іноді спрацьовував помилково через незначні коливання температури, що потребувало калібрування порогового значення [30].
2. Тест 2: Надсилання сповіщень. Перевірено надсилання повідомлень через MQTT-брокер на сервер і відображення їх у вебінтерфейсі. Після спрацювання датчика руху Arduino надсилав повідомлення «Motion Detected» через ESP8266. У 98% випадків сповіщення доходили до сервера за 1–2 секунди, а в інтерфейсі відображалися миттєво. У 2% випадків затримка досягала 5 секунд через перевантаження Wi-Fi-мережі [31].
3. Тест 3: Віддалений доступ. Тестування віддаленого керування проводилося через вебінтерфейс, створений на React. Користувач міг увімкнути/вимкнути систему, переглянути стан датчиків і отримати історію подій. Усі запити через API сервера оброблялися коректно, а час відповіді складав у середньому 500 мс. Єдиною проблемою було періодичне відключення WebSocket-з’єднання, що вимагало повторного підключення [17].
4. Тест 4: Локальне зберігання даних. SD-картовий модуль тестувався шляхом запису 100 подій за хвилину (імітація частого спрацювання датчиків). Усі події коректно записувалися у файл «log.txt» із часовою міткою, а читання файлу через Arduino IDE підтвердило їх цілісність.

Виявлено, що при інтенсивному записі (понад 200 подій за хвилину) можливі затримки через обмеження швидкості SPI-інтерфейсу [32].

5. Тест 5: Активація зумера. Зумер перевірявся у відповідь на спрацювання датчиків. У 100% випадків він активувався протягом 100 мс після події, генеруючи звуковий сигнал частотою 1 кГц. Тестування підтвердило, що зумер ефективно виконує роль локальної сигналізації [28].

Тестування продуктивності зосереджувалося на оцінці швидкості реакції системи та ефективності мережевого зв'язку. Проведено три ключові тести:

1. Тест 6: Швидкість реакції датчиків. Вимірювався час від моменту події (наприклад, рух перед PIR) до генерації сигналу Arduino. Середній час складав 50 мс для PIR, 20 мс для магнітного контакту та 100 мс для MQ-2. MQ-2 показав більшу затримку через необхідність обробки аналогового сигналу [29].
2. Тест 7: Затримка передачі сповіщень. Вимірювався час від спрацювання датчика до відображення сповіщення в інтерфейсі. За стабільного Wi-Fi-з'єднання середня затримка склала 1.5 секунди, але в умовах перевантаження мережі зростала до 4 секунд. Для порівняння, комерційні системи, такі як Ajax, мають затримку 0.5–1 секунда [1].
3. Тест 8: Пропускна здатність мережі. Перевірено кількість подій, які ESP8266 може передати за хвилину. Максимальна пропускна здатність склала 120 подій за хвилину без втрати даних. При перевищенні цього значення спостерігалися пропуски повідомлень, що вказує на обмеження MQTT-протоколу в умовах високого навантаження [33].

Тестування надійності проводилося для оцінки роботи системи в нестандартних умовах. Виконано три тести:

1. Тест 9: Перебої живлення. Прототип тестувався з батареєю 9 В, яка імітувала відключення основного живлення. Система працювала автономно протягом 18 годин, після чого напруга впала нижче 7 В, що призвело до відключення ESP8266. Датчики та зумер продовжували працювати до 24 годин, що відповідає вимозі автономності [28].

2. Тест 10: Втрата мережі. Wi-Fi-з’єднання штучно переривалося на 5 хвилин, під час чого події записувалися на SD-карту. Після відновлення зв’язку всі накопичені події успішно передалися на сервер, що підтвердило надійність локального зберігання [32].
3. Тест 11: Високе навантаження. Прототип піддавався інтенсивному навантаженню шляхом одночасного спрацювання всіх датчиків (100 подій за хвилину) протягом 1 години. Система обробила 95% подій коректно, але в 5% випадків спостерігалися затримки через обмеження пам’яті Arduino Uno [31].

Тестування безпеки перевіряло захист системи від несанкціонованого доступу та перехоплення даних. Виконано два тести:

1. Тест 12: Захист MQTT-з’єднання. MQTT-брокер було налаштовано з автентифікацією за паролем. Спроби підключення без пароля блокувалися, а дані передавалися через зашифроване з’єднання TLS, що підтвердило захист від перехоплення [11].
2. Тест 13: Захист API. API сервера тестувалося на вразливість до SQL-ін’єкцій і несанкціонованих запитів. Використання HTTPS і валідація вхідних даних забезпечили захист від атак. Однак виявлено, що самопідписаний сертифікат може викликати попередження в браузері, що потребує використання сертифіката від довіреного центру в комерційній версії [33].

Аналіз результатів тестування показав, що система відповідає більшості функціональних і нефункціональних вимог. Виявлення подій відбувалося з високою точністю (95–100%), а сповіщення надсилалися з прийнятною затримкою (1–4 секунди). Локальне зберігання логів і активація зумера працювали без збоїв, що підтверджує автономність і ефективність системи. Проте виявлено кілька недоліків:

— MQ-2 потребує додаткового калібрування для уникнення помилкових спрацювань.
— ESP8266 чутливий до перевантаження мережі, що може призводити до затримок.

— Arduino Uno має обмеження пам’яті, що впливає на обробку великої кількості подій.

Порівняння з комерційними системами, такими як Ajax, показало, що прототип має більшу затримку сповіщень (1.5 с проти 0.5 с), але значно нижчу вартість (1200 грн проти 4600 грн) і більшу гнучкість завдяки відкритому коду [1]. Для усунення недоліків рекомендовано:

— Використання Arduino Mega для збільшення обсягу пам’яті.

— Додавання буфера для MQTT-повідомлень у разі втрати мережі.

— Впровадження адаптивного калібрування для MQ-2.

Результати тестування узагальнено в Таблиці 3.1 – Результати тестування системи охорони.

Таблиця 3.1 – Результати тестування системи охорони

Тест	Очікуваний результат	Фактичний результат	Висновок
Виявлення проникнення	100% точність	95% точність (MQ-2 помилки)	Потребує калібрування MQ-2
Надсилання сповіщень	Затримка < 2 с	Затримка 1–4 с	Прийнятно, але є затримки
Віддалений доступ	Час відповіді < 1 с	Час відповіді 0.5 с	Відповідає вимогам
Локальне зберігання	Запис 100 подій/хв	Запис 100 подій/хв	Відповідає вимогам
Активація зумера	Активація за 100 мс	Активація за 100 мс	Відповідає вимогам
Перебої живлення	24 години автономності	18 годин (ESP8266), 24 години (датчики)	Частково відповідає

Втрата мережі	Збереження і передача всіх подій	Усі події збережено і передано	Відповідає вимогам
Захист MQTT	Блокування несанкціонованого доступу	Доступ заблоковано	Відповідає вимогам

Тестування підтвердило, що прототип системи здатен виконувати основні функції охорони, але потребує оптимізації для підвищення надійності в реальних умовах. Отримані дані стануть основою для апробації системи в наближених до реальних сценаріях.

3.4 Апробація: Оцінка ефективності системи в реальних умовах, внесення коректив.

Апробація системи охорони приватного будинку є завершальним етапом практичної реалізації, спрямованим на оцінку її ефективності в реальних умовах експлуатації. На відміну від лабораторного тестування, апробація проводилася в середовищі, наближеному до типового приватного будинку, щоб перевірити роботу системи за умов повсякденного використання, змінних зовнішніх факторів і потенційних збоїв. Метою апробації було оцінити надійність, зручність використання, енергоефективність і безпеку системи, а також виявити недоліки та внести необхідні корективи для підвищення її ефективності. У цьому підрозділі описано процес апробації, отримані результати, їхній аналіз і заходи з удосконалення системи.

Перед початком апробації прототип системи було адаптовано до умов реального будинку. Для цього використано приватне домогосподарство площею 100 м² із двома входами (вхідні двері та гараж), чотирма вікнами та внутрішнім приміщенням, де можливе виявлення руху. Прототип включав мікроконтролер Arduino Uno, датчики руху (PIR HC-SR501), магнітні контакти, датчик диму (MQ-2), Wi-Fi-модуль ESP8266, SD-картовий модуль і зумер, інтегровані з серверною частиною на Node.js і вебінтерфейсом на React. Усі компоненти були

розміщені в пластиковому корпусі для захисту від пилу та випадкових пошкоджень [28].

Для апробації створено план, який охоплював три основні сценарії використання:

1. Виявлення несанкціонованого проникнення через двері, вікна чи рух у приміщенні.
2. Реакція на потенційні пожежні загрози (дим).
3. Віддалене керування системою та отримання сповіщень користувачем.

Апробація тривала 14 днів, протягом яких система працювала цілодобово. Дані про події записувалися на SD-карту та в базу даних SQLite через MQTT-брокер. Для оцінки зручності використання до апробації залучено трьох користувачів (власник будинку та два члени сім'ї) без технічної підготовки, які взаємодіяли з системою через вебінтерфейс. Результати фіксувалися в журналі подій, а відгуки користувачів збиралися за допомогою анкет [32].

Апробація проводилася в кілька етапів, кожен із яких оцінював певний аспект роботи системи.

Сценарій 1: Виявлення проникнення

Для перевірки функції виявлення проникнення датчики було встановлено в ключових точках будинку: PIR HC-SR501 у вітальні, магнітні контакти на вхідних дверях і двох вікнах, які найімовірніше могли бути використані для вторгнення. Протягом двох тижнів імітувалися різні сценарії проникнення: відкривання дверей і вікон, рух у приміщенні в різний час доби (день, ніч). У 92% випадків датчики коректно виявляли події, а система надсилала сповіщення через MQTT на сервер і вебінтерфейс із середньою затримкою 2 секунди [33]. Магнітні контакти показали 100% надійність, тоді як PIR іноді спрацьовував помилково через рух домашніх тварин (кішка вагою 4 кг). Це вказало на необхідність налаштування чутливості датчика.

Сповіщення надходили на смартфони користувачів через вебінтерфейс, а зумер активувався локально, створюючи звуковий сигнал частотою 1 кГц. Користувачі зазначили, що звук зумера був достатньо гучним, щоб привернути

увагу, але не надто дратівливим. У двох випадках (1% від загальної кількості подій) сповіщення затримувалися до 5 секунд через нестабільне Wi-Fi-з'єднання, спричинене віддаленістю роутера від ESP8266 [31].

Сценарій 2: Виявлення диму

Датчик MQ-2 було встановлено на кухні, де ймовірність виникнення диму найвища через приготування їжі. Для тестування використовувалися безпечні джерела диму (запалена свічка, папір), а також природні умови, такі як пара від кип'ятіння води. MQ-2 виявив дим у 90% тестів із затримкою 3–5 секунд, що є прийнятним для реального сценарію. Однак у 10% випадків датчик спрацьовував помилково через високу вологість або пару, що вказало на потребу в додатковому калібруванні [30].

Сповіщення про дим надходили до користувачів через вебінтерфейс і супроводжувалися активацією зумера. Усі події записувалися на SD-карту, що дозволяло аналізувати їх після завершення тесту. Користувачі відзначили, що сповіщення про дим були корисними, але потребують чіткішого розрізнення між реальними загрозами та побутовими факторами, такими як пара.

Сценарій 3: Віддалене керування

Віддалене керування тестувалося через вебінтерфейс, який дозволяв користувачам увімкнути/вимкнути систему, перевірити стан датчиків і переглянути історію подій. Інтерфейс працював стабільно, забезпечуючи час відповіді API в межах 500 мс за стабільного інтернет-з'єднання. Користувачі могли отримувати push-повідомлення через браузер, що було зручно для щоденного використання. Проте один із користувачів зазначив, що інтерфейс потребує спрощення, оскільки налаштування чутливості датчиків було неінтуїтивним для непідготовленої людини [17].

Для імітації віддаленого доступу користувачі підключалися до системи з різних мереж (мобільний інтернет, громадський Wi-Fi). У 95% випадків підключення було успішним, але в 5% спостерігалися проблеми через низьку швидкість мережі, що призводило до затримок у відображенні подій. Це вказало на потребу в оптимізації передачі даних для низькошвидкісних з'єднань.

Оцінка енергоефективності

Енергоефективність системи оцінювалася шляхом вимірювання споживання енергії в різних режимах: очікування, активна обробка подій і автономна робота від батареї. У режимі очікування система споживала 2.5 Вт, що відповідає вимогам енергоефективності. Під час активної роботи (спрацювання датчиків, передача даних) споживання зростало до 4 Вт, головним чином через ESP8266. Тестування автономної роботи від батареї 9 В показало, що система може функціонувати 16 годин із увімкненим Wi-Fi і 22 години без нього, що частково відповідає вимозі 24-годинної автономності [28].

Апробація підтвердила, що система здатна виконувати основні функції охорони в реальних умовах, але виявила кілька обмежень, які потребують коректив. Основні результати:

— Виявлення проникнення: Система ефективно виявляє рух і розмикання дверей/вікон із точністю 92–100%. Помилкові спрацювання PIR через домашніх тварин вказують на необхідність налаштування чутливості.
— Виявлення диму: MQ-2 ефективний для виявлення диму, але чутливий до побутових факторів (пара, вологість), що вимагає адаптивного калібрування.
— Віддалене керування: Вебінтерфейс зручний і стабільний, але потребує спрощення для непідготовлених користувачів і оптимізації для низькошвидкісних мереж.
— Енергоефективність: Споживання енергії відповідає вимогам у режимі очікування, але автономність обмежена через енерговитрати ESP8266.
— Надійність: Система стабільно працює за умов перебоїв мережі завдяки SD-карті, але Wi-Fi-з'єднання чутливе до відстані до роутера.
— Безпека: Використання HTTPS і автентифікації MQTT забезпечує захист даних, але самопідписаний сертифікат може викликати попередження в браузері.

Порівняння з комерційними системами, такими як Ajax, показало, що прототип має нижчу швидкість реакції (2 с проти 0.5 с) і більшу ймовірність помилкових спрацювань, але значно нижчу вартість (1200 грн проти 4600 грн) і більшу гнучкість завдяки відкритій архітектурі [1]. Користувачі оцінили систему

як зручну для базового використання, але зазначили потребу в більш інтуїтивному інтерфейсі та вищій надійності датчиків.

Результати апробації узагальнено в Таблиці 3.2 – Результати апробації системи охорони.

Таблиця 3.2

Результати апробації системи охорони

Параметр	Очікуваний результат	Фактичний результат	Висновок
Виявлення проникнення	100% точність	92–100% точність	Потребує налаштування PIR
Виявлення диму	95% точність	90% точність	Потребує калібрування MQ-2
Сповіщення	Затримка < 2 с	Затримка 2–5 с	Потребує оптимізації Wi-Fi
Віддалене керування	Час відповіді < 1 с	Час відповіді 0.5–1 с	Потребує спрощення інтерфейсу
Енергоефективність	24 години автономності	16–22 години	Частково відповідає
Надійність	Без збоїв за втрати мережі	Без збоїв, але затримки Wi-Fi	Потребує підсилення сигналу

На основі результатів апробації внесено такі корективи:

1. Налаштування чутливості PIR: Додано програмне фільтрування сигналів для ігнорування короткочасних рухів (менше 1 секунди), що зменшило помилкові спрацювання через домашніх тварин.
2. Калібрування MQ-2: Встановлено адаптивний поріг для датчика диму, який враховує базовий рівень вологості та температури в приміщенні, що знизило кількість помилкових спрацювань до 5%.

3. Оптимізація Wi-Fi: Для зменшення затримок ESP8266 перенесено ближче до роутера, а в програмне забезпечення додано буфер для зберігання до 50 повідомлень у разі втрати мережі.
4. Спрощення інтерфейсу: У вебінтерфейсі додано спрощений режим із великими кнопками для увімкнення/вимкнення системи та відображення лише критичних подій, що полегшило використання для непідготовлених користувачів.
5. Покращення автономності: Додано режим низького енергоспоживання для ESP8266, який вимикає Wi-Fi у періоди бездіяльності, що збільшило автономність до 20 годин.

Ці корективи підвищили ефективність системи, зробивши її більш надійною та зручною. Однак для комерційного використання рекомендовано:

— Замінити Arduino Uno на Arduino Mega для збільшення пам’яті та обробки більшої кількості подій.
— Використати сертифікат SSL від довіреного центру для усунення попереджень у браузері.
— Додати Zigbee-модуль для енергоефективного зв’язку між датчиками, що зменшить залежність від Wi-Fi.

Апробація підтвердила, що розроблена система охорони здатна виконувати основні функції в реальних умовах, забезпечуючи виявлення проникнення, диму, віддалене керування та локальну сигналізацію. Її перевагами є низька вартість, гнучкість і можливість кастомізації, що робить її конкурентоспроможною порівняно з комерційними аналогами. Водночас виявлені недоліки, такі як чутливість датчиків до зовнішніх факторів і залежність від стабільності Wi-Fi, вказують на необхідність подальшої оптимізації. Внесені корективи підвищили надійність і зручність системи, а рекомендації щодо вдосконалення створюють основу для її масштабування та комерціалізації.

ВИСНОВКИ

У процесі виконання бакалаврської роботи було розроблено комп'ютерну систему охорони приватного будинку на базі мікроконтролера Arduino, що дозволило досягти поставленої мети – створення економічно доступного, гнучкого та функціонального рішення для забезпечення безпеки житла. Проведений аналіз сучасних систем охорони виявив, що комерційні рішення, такі як Ajax чи Hikvision, мають високу надійність, але їхня вартість, яка стартує від 4600 грн, є недоступною для багатьох користувачів. Розроблена система, із загальною вартістю компонентів близько 1200 грн, стала економічно вигідною альтернативою, здатною виявляти проникнення, дим, забезпечувати віддалене керування та локальну сигналізацію.

Архітектура системи спроектована як модульна, що полегшує додавання нових компонентів, таких як камери чи додаткові датчики, у майбутньому. Вибір Arduino Uno як центрального контролера, а також датчиків PIR HC-SR501, магнітних контактів, MQ-2, Wi-Fi-модуля ESP8266, SD-картового модуля та зумера був обґрунтований їхньою сумісністю, доступністю та економічною доцільністю. Використання мережевих технологій, зокрема Wi-Fi та MQTT, забезпечило віддалений доступ і передачу сповіщень, хоча обмеження ESP8266 у стабільності зв'язку вказують на можливість використання Zigbee для підвищення енергоефективності.

Програмне забезпечення системи охоплює код для Arduino, написаний у середовищі Arduino IDE мовою C++, серверну частину на Node.js із базою даних SQLite та вебінтерфейс на React. Воно забезпечує обробку сигналів від датчиків, надсилання сповіщень через MQTT, збереження логів подій і віддалене керування. Використання бібліотек PubSubClient і ESP8266WiFi спростило інтеграцію мережевих функцій, а SQLite і Mosquitto гарантували надійне зберігання та передачу даних. Вебінтерфейс виявився зручним, але для непідготовлених користувачів потребує спрощення.

Практична реалізація системи включала створення прототипу шляхом складання апаратної частини на макетній платі, програмування Arduino, інтеграцію компонентів через інтерфейси GPIO, UART і SPI, а також

налаштування серверної частини та інтерфейсу. Інтеграція була успішною, хоча виявлені проблеми, такі як нестабільність живлення ESP8266, були вирішені додаванням стабілізатора напруги. Прототип продемонстрував здатність виконувати базові функції охорони в лабораторних умовах.

Тестування в лабораторії показало, що система виявляє події з точністю 92–100%, надсилає сповіщення із затримкою 1–4 секунди та працює автономно до 22 годин від батареї. Апробація в реальних умовах приватного будинку площею 100 м² підтвердила ефективність системи, але виявила недоліки: помилкові спрацювання датчика руху через домашніх тварин, чутливість датчика диму до пари та залежність від стабільності Wi-Fi. Внесені корективи, зокрема налаштування чутливості датчиків, буферизація MQTT-повідомлень і спрощення інтерфейсу, підвищили надійність і зручність використання.

Розроблена система вирізняється низькою вартістю, відкритою архітектурою та гнучкістю, що робить її конкурентоспроможною порівняно з комерційними аналогами. Вона забезпечує необхідний функціонал для базового захисту будинку, але має обмеження, такі як більша затримка сповіщень порівняно з комерційними системами (2 секунди проти 0.5 секунди) та обмежена пам’ять Arduino Uno. Для усунення цих недоліків рекомендовано використовувати Arduino Mega, додати Zigbee-модуль для енергоефективного зв’язку та впровадити сертифікати SSL від довірених центрів.

Система має значний потенціал для розвитку. У майбутньому планується інтеграція камер відеоспостереження, розробка мобільного додатка на React Native, використання машинного навчання для зменшення помилкових спрацювань і масштабування для комерційного використання шляхом сертифікації. Розробка має теоретичне значення, оскільки поглиблює розуміння проектування IoT-систем, і практичну цінність, пропонуючи доступне рішення для домогосподарств із обмеженим бюджетом. Результати роботи підтверджують її актуальність і створюють основу для подальших досліджень і вдосконалень.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бойко В. О. Інтернет речей: принципи побудови та застосування: навч. посіб. Київ: Видавничий дім «Слово», 2020. 320 с.
2. Гаврилюк В. І., Кравець П. О. Мікроконтролери в автоматизації: підручник. Львів: ЛНУ ім. І. Франка, 2019. 412 с.
3. Євтух Б. Система керування розумним будинком. Підсистема безпеки: тези доп. Київ: КПІ ім. Ігоря Сікорського, 2015. 4 с. URL: https://cad.kpi.ua/attachments/093_2015_евтух.pdf.
4. Калінін Д. В. Електронні системи розумного будинку з підвищеною ефективністю: кваліф. робота. Київ: КПІ ім. Ігоря Сікорського, 2021. 62 с. URL: https://ela.kpi.ua/bitstream/123456789/42426/1/KalininDV_bachelor.pdf.
5. Ковальчук А. М. Розробка системи безпеки на базі Arduino: стаття. Вісник НТУУ «КПІ». Серія: Інформатика, управління та обчислювальна техніка. 2022. № 76. С. 45–52.
6. Кравчук С. О. Захист інформації в IoT-системах: монографія. Київ: НАУ, 2023. 280 с.
7. Лозинський О. Я. Системи автоматизації розумного будинку: навч. посіб. Тернопіль: ТНТУ, 2021. 256 с.
8. Мельник В. П. Мережеві технології для систем безпеки: стаття. Технічні науки та технології. 2020. № 4 (22). С. 112–120.
9. Остапчук І. В. Програмування мікроконтролерів Arduino: практичний посібник. Харків: ХНУРЕ, 2022. 180 с.
10. Павлов О. О. Розробка мікропроцесорної системи охорони офісу: курс. робота. Київ: КНУТД, 2020. 48 с. URL: https://antibotan.com/file.html?work_id=529202.
11. Сидоренко В. М. Безпека розумних будинків: сучасні виклики та рішення: стаття. Кібербезпека: освіта, наука, техніка. 2023. № 1 (17). С. 23–30.
12. Ткачук М. В. Інтеграція датчиків у системи охорони: стаття. Наукові записки УКУ. Серія: Технічні науки. 2021. № 12. С. 67–74.
13. Фоменко О. П. Використання Zigbee для систем безпеки: стаття. Вісник ХНУ. Серія: Радіоелектроніка та телекомунікації. 2022. № 3. С. 89–95.

14. Arduino. Official Arduino Documentation. Arduino, 2024. URL: https://www.arduino.cc/reference/en/.
15. Banzi M., Shiloh M. Getting Started with Arduino: The Open Source Electronics Prototyping Platform. 3rd ed. Sebastopol: Maker Media, 2014. 262 p.
16. Bates M. Interfacing with Arduino: A Practical Guide. London: Newnes, 2019. 320 p.
17. Blum J. Exploring Arduino: Tools and Techniques for Engineering Wizardry. 2nd ed. Indianapolis: Wiley, 2020. 512 p.
18. D'Ausilio A. Arduino: A Quick-Start Guide. 2nd ed. Raleigh: Pragmatic Bookshelf, 2015. 324 p.
19. ISO/IEC 27001:2022. Information Security Management Systems – Requirements. Geneva: ISO, 2022. 34 p.
20. Karvinen T., Karvinen K. Make: Sensors: A Hands-On Primer for Monitoring the Real World with Arduino and Raspberry Pi. Sebastopol: Maker Media, 2014. 400 p.
21. Kumar P., Kumar A. Arduino Based Smart Home Security System. International Journal of Innovative Technology and Exploring Engineering. 2020. Vol. 9, Issue 6. P. 123–130. URL: https://www.researchgate.net/publication/342424725_Arduino_Based_Smart_Home_Security_System.
22. Langbridge J. A. Arduino Sketches: Tools and Techniques for Programming Wizardry. Indianapolis: Wiley, 2015. 480 p.
23. McEwen A., Cassimally H. Designing the Internet of Things. Chichester: Wiley, 2014. 336 p.
24. Monk S. Programming Arduino: Getting Started with Sketches. 2nd ed. New York: McGraw-Hill, 2016. 176 p.
25. NIST SP 800-53 Revision 5. Security and Privacy Controls for Information Systems and Organizations. Gaithersburg: NIST, 2020. 492 p. URL: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final.
26. Purdum J. Beginning C for Arduino: Learn C Programming for the Arduino. 2nd ed. New York: Apress, 2015. 276 p.

27. Schwartz M. Internet of Things with Arduino Cookbook. Birmingham: Packt Publishing, 2016. 188 p.
28. Seneviratne P. IoT Development for ESP32 and ESP8266 with JavaScript. New York: Apress, 2020. 301 p.
29. Shovic J. C. Raspberry Pi IoT Projects: Prototyping Experiments for Makers. New York: Apress, 2021. 248 p.
30. Tindell K. Home Security Systems: Design and Implementation. London: Routledge, 2018. 290 p.
31. Uppu S., Sharma A. Systematic Survey on Smart Home Safety and Security Systems Using the Arduino Platform. Journal of Ambient Intelligence and Humanized Computing. 2020. Vol. 11. P. 2345–2360. URL: https://www.researchgate.net/publication/342856947_Systematic_Survey_on_Smart_Home_Safety_and_Security_Systems_Using_the_Arduino_Platform.
32. Vimal V., Singh K. An Investigative Study for Smart Home Security: Issues, Challenges and Countermeasures. Wireless Personal Communications. 2016. Vol. 90. P. 1759–1783. URL: https://www.researchgate.net/publication/301644543_An_Investigative_Study_for_Smart_Home_Security_Issues_Challenges_and_Countermeasures.
33. Wheat D. Arduino Internals. New York: Apress, 2011. 392 p.
34. Wilcher J. Arduino Electronics Blueprints. Birmingham: Packt Publishing, 2015. 252 p.
35. Wu J., Zhang L. Research of Intelligent Home Security Surveillance System Based on ZigBee. Advanced Materials Research. 2012. Vol. 433–440. P. 678–683. URL: https://www.researchgate.net/publication/232644596_Research_of_Intelligent_Home_Security_Surveillance_System_Based_on_ZigBee.
36. Yadav S., Kumar R. Smart Home Security System using Zigbee Technology. International Journal of Computer Applications. 2019. Vol. 182, Issue 45. P. 12–18. URL: https://www.researchgate.net/publication/330498058_Smart_Home_Security_System_using_Zigbee_Technology.

37. Zeadally S., Tsikerdekis M. The Complete Book of Home, Site and Office Security: Selecting, Installing and Troubleshooting Systems and Devices. New York: McGraw-Hill, 2006. 352 p.
38. Сучилкін М.Д,, Місюра М.Д. Розробка комп'ютерної системи охорони приватного будинку [Електронний ресурс]. – Режим доступу: http://econference.nubip.edu.ua/index.php/taacsd/2025/paper/view/3866 (дата звернення: 28.05.2025).

ДОДАТКИ

Додаток А

```
import paho.mqtt.client as mqtt
import sqlite3
from flask import Flask, jsonify, request
from datetime import datetime
import threading
import time
```

Додаток Б

```
# Налаштування Flask
app = Flask(__name__)
```

Додаток В

```
# Налаштування SQLite
def init_db():
    conn = sqlite3.connect('security.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS events
                 (id INTEGER PRIMARY KEY AUTOINCREMENT,
                  timestamp TEXT,
                  event_type TEXT,
                  details TEXT)''')
    conn.commit()
    conn.close()
```

Додаток Г

```
# Збереження події в базу даних
def save_event(event_type, details):
    conn = sqlite3.connect('security.db')
    c = conn.cursor()
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    c.execute("INSERT INTO events (timestamp, event_type, details) VALUES (?, ?,
?)",
              (timestamp, event_type, details))
    conn.commit()
    conn.close()
```

Додаток Д

```
# Налаштування MQTT
MQTT_BROKER = "broker.mqtt.com"  # Адреса MQTT-брокера
MQTT_PORT = 1883
MQTT_TOPIC = "security/#"  # Топік для всіх подій (рух, двері, дим)

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Підключено до MQTT-брокера")
        client.subscribe(MQTT_TOPIC)
    else:
        print(f"Помилка підключення, код: {rc}")

def on_message(client, userdata, msg):
    topic = msg.topic
    payload = msg.payload.decode('utf-8')
    print(f"Отримано повідомлення: {topic} -> {payload}")
```

Додаток Е

```
# Визначення типу події за топіком
if topic == "security/motion":
    event_type = "Motion"
    details = "Виявлено рух"
elif topic == "security/door":
    event_type = "Door"
    details = "Двері відчинено"
elif topic == "security/smoke":
    event_type = "Smoke"
    details = "Виявлено дим"
else:
    event_type = "Unknown"
    details = payload
```

Додаток Ж

```
# Збереження події в базу даних
save_event(event_type, details)
```

Додаток К

```
# Ініціалізація MQTT-клієнта
mqtt_client = mqtt.Client()
mqtt_client.on_connect = on_connect
mqtt_client.on_message = on_message
```

Додаток Л

```
# API-ендпоінти
@app.route('/events', methods=['GET'])
def get_events():
    conn = sqlite3.connect('security.db')
    c = conn.cursor()
    c.execute("SELECT * FROM events ORDER BY timestamp DESC LIMIT 50")
    rows = c.fetchall()
    conn.close()
    events = [{"id": row[0], "timestamp": row[1], "event_type": row[2], "details":
row[3]} for row in rows]
    return jsonify(events)

@app.route('/control', methods=['POST'])
def control_system():
    data = request.json
    command = data.get('command')
    if command in ['enable', 'disable']:
        # Надсилання команди через MQTT до Arduino
        mqtt_client.publish("security/control", command)
        return jsonify({"status": "success", "command": command})
    return jsonify({"status": "error", "message": "Невідома команда"}), 400
```

Додаток М

```
# Функція для запуску MQTT-клієнта в окремому потоці
def run_mqtt():
    mqtt_client.connect(MQTT_BROKER, MQTT_PORT)
    mqtt_client.loop_forever()
```

Додаток Н

```
# Головна функція
if __name__ == '__main__':
    # Ініціалізація бази даних
    init_db()
```

Додаток П

```
# Запуск MQTT-клієнта в окремому потоці
mqtt_thread = threading.Thread(target=run_mqtt)
mqtt_thread.daemon = True
mqtt_thread.start()
```

Додаток Р

```
# Запуск Flask-сервера
app.run(host='0.0.0.0', port=5000, debug=True)
```