ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА
(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерних систем та технологій
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Комплексна система безпеки з лазерними сенсорами та Telegram-інтеграцією»
(тема кваліфікаційної роботи українською мовою)
«Comprehensive security system with laser sensors and Telegram integration»
(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання
спеціальності 123 – Комп'ютерна інженерія
(код, назва спеціальності)
Освітня програма «Комп'ютерна інженерія»
(назва)
Соценко Максим Миколайович
(прізвище, ім'я, по-батькові здобувача)
Керівник к.т.н., доцент Камєнєва А.В. _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)
Рецензент проф. Гунченко Ю.О.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
комп'ютерних систем та технологій
№ ___ від ____._____. 2025 р.
Завідувач кафедри
_____ Юрій ГУНЧЕНКО
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № _____
протокол № __від __._____.2025 р.
Оцінка_____/_____/_____
(за національною шкалою/шкалою ECTS/ бали)
Голова ЕК
_____ Світлана АНТОЩУК
(підпис) (ім'я, прізвище)

Одеса 2025

АНОТАЦІЯ

У кваліфікаційній роботі розробляється тема «Комплексна система безпеки з лазерними сенсорами та Telegram-інтеграцією».

Мета роботи – створення сучасної системи безпеки, яка поєднує лазерні сенсори, мікроконтролери Arduino та інтеграцію з месенджером Telegram для забезпечення оперативного моніторингу та керування. Особлива увага приділяється високій точності виявлення порушень, автоматизації реагування та зручності взаємодії користувача з системою через інтерактивний інтерфейс Telegram. Система дозволяє отримувати миттєві сповіщення про події, включаючи фото чи відео, а також дистанційно керувати функціями безпеки, такими як активація камери чи перевірка статусу сенсорів.

Результатом роботи є гнучка система безпеки, яка забезпечує високоточне виявлення загроз за допомогою лазерних сенсорів, інтеграцію з Telegram для оперативного інформування та керування, а також можливість масштабування для використання в різних умовах – від приватних будинків до промислових об’єктів. Система вирізняється простотою використання, високою надійністю та адаптивністю до потреб користувача.

ANNOTATION

The qualification work is dedicated to the development of the topic «Comprehensive Security System with Laser Sensors and Telegram Integration».

The objective of the work is to create a modern security system that combines laser sensors, Arduino microcontrollers, and integration with the Telegram messenger to ensure real-time monitoring and control. Particular emphasis is placed on the high accuracy of detecting intrusions, automation of response, and user-friendly interaction with the system through Telegram's interactive interface. The system enables instant notifications about events, including photos or videos, and allows remote management of security functions, such as activating cameras or checking sensor statuses.

The result of the work is a flexible security system that provides precise threat detection using laser sensors, seamless integration with Telegram for prompt notifications and control, and scalability for use in various environments – from private homes to industrial facilities. The system is characterized by ease of use, high reliability, and adaptability to user needs.

ЗМІСТ

Стор.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ 5
ВСТУП 6
1 ТЕОРЕТИЧНІ ВІДОМОСТІ 8
1.1 Огляд сучасних систем безпеки та їх особливостей 8
1.2 Принципи роботи лазерних сенсорів у системах безпеки 11
1.3 Мікроконтролери Arduino як основа систем безпеки.....12
1.4 Можливості Telegram API для інтеграції в системи безпеки.....15
1.5 Порівняльний аналіз існуючих систем безпеки та обґрунтування вибору підходу.....17
1.6 Постановка задачі дослідження 19
2 ПРОЕКТУВАННЯ СИСТЕМИ 21
2.1 Розробка апаратної частини системи безпеки.....21
2.2 Реалізація програмної частини 28
3 ІНСТРУКЦІЯ КОРИСТУВАЧА..... 35
ВИСНОВКИ..... 40
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 42
ДОДАТОК А Програмний код ESP32-CAM..... 45
ДОДАТОК Б Програмний код ESP32..... 52

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

API – прикладний програмний інтерфейс (Application Programming Interface).

GSM – глобальна система мобільного зв’язку (Global System for Mobile Communications).

ESP32 – 32-бітний мікроконтролер із підтримкою Wi-Fi та Bluetooth, розроблений компанією Espressif Systems.

ESP-NOW – бездротовий протокол зв’язку, що дозволяє обмінюватися даними між пристроями ESP32 без використання Wi-Fi мережі.

ESP32-CAM – модуль ESP32 з вбудованою камерою, що підтримує Wi-Fi та зберігання даних на SD-карті.

SSID – ідентифікатор назви Wi-Fi мережі (Service Set Identifier).

IDE – інтегроване середовище розробки (Integrated Development Environment).

SD – карта пам’яті стандарту Secure Digital для зберігання даних.

WiFi – бездротова технологія передачі даних у локальній мережі (Wireless Fidelity).

ID – ідентифікатор, унікальний номер або позначення об'єкта (Identifier).

GPIO – універсальний вхід/вихід (General Purpose Input/Output) на мікроконтролерах.

OV2640 – модель камери з CMOS-сенсором, яку використовують у модулі ESP32-CAM.

PIR – пасивний інфрачервоний датчик руху (Passive Infrared Sensor).

AJAX – сучасна комерційна система безпеки українського виробництва з бездротовими датчиками та мобільним керуванням.

iOS – операційна система мобільних пристроїв Apple (iPhone Operating System).

ВСТУП

Сучасний світ стрімко розвивається, а разом з ним зростають вимоги до безпеки в різних сферах життя – від приватних будинків до промислових об’єктів. Традиційні системи безпеки, такі як звукові сигналізації чи камери відеоспостереження, часто не відповідають сучасним потребам через обмежену функціональність, залежність від людського фактора та повільну реакцію на загрози. У цьому контексті виникає потреба у створенні інноваційних, автоматизованих і гнучких систем безпеки, які здатні забезпечити високий рівень захисту, оперативне реагування та зручність управління для користувача [1].

Комплексна система безпеки з лазерними сенсорами та інтеграцією з месенджером Telegram є сучасним рішенням, яке поєднує передові технології для моніторингу та захисту. Лазерні сенсори забезпечують високу точність виявлення руху чи перетину контрольованих зон, що робить їх ефективними для охорони дверей, вікон чи інших стратегічно важливих ділянок. Інтеграція з Telegram дозволяє користувачам отримувати миттєві сповіщення у реальному часі про події, включаючи фото чи відео, а також дистанційно керувати системою через зручний інтерфейс месенджера. Використання мікроконтролерів Arduino як центрального елемента системи забезпечує її доступність, простоту реалізації та можливість масштабування [2].

Історія розвитку систем безпеки тісно пов’язана з еволюцією технологій. Ще в 1970-х роках почали з’являтися перші автоматизовані системи охорони, які базувалися на простих датчиках руху [3]. Проте вони мали обмежену функціональність і не забезпечували оперативного зв’язку з власником. З появою Інтернету та мобільних технологій у 2000-х роках системи безпеки почали інтегруватися з мережевими інтерфейсами, дозволяючи віддалений доступ до даних. Сьогодні, завдяки розвитку API месенджерів, таких як Telegram, та доступності апаратних платформ, таких як Arduino, створюються

системи, які не лише виявляють загрози, але й забезпечують інтерактивну взаємодію з користувачем у реальному часі [4].

Актуальність теми підтверджується зростаючим попитом на рішення для «розумних будинків» та інтегрованих систем безпеки. За даними глобального дослідження особливостей і способів використання технологій, які опубліковані у звіті Digital 2024 Global Overview Report, кількість користувачів Telegram становить 800 мільйонів [5], що робить цей месенджер ідеальною платформою для створення доступних і зручних систем сповіщення. Крім того, зростання інтересу до автоматизації та інтелектуальних систем безпеки вказує на необхідність розробки рішень, які поєднують високу ефективність, простоту використання та низьку вартість.

Таким чином, мета даної кваліфікаційної роботи – розробка комплексної системи безпеки з використанням лазерних сенсорів та інтеграцією з Telegram для забезпечення оперативного моніторингу та керування.

Для досягнення цієї мети необхідно вирішити такі задачі:

- дослідити предметну область, включаючи принципи роботи лазерних сенсорів та особливості Telegram API;
- реалізувати на апаратному рівні систему на базі Arduino, яка включає лазерні сенсори, камери та модулі зв’язку (Wi-Fi або GSM);
- розробити програмне забезпечення для інтеграції системи з Telegram, що забезпечує надсилання сповіщень, обробку команд користувача та захист даних;
- провести тестування системи для забезпечення її надійності, точності та швидкості реагування в реальних умовах.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Огляд сучасних систем безпеки та їх особливостей

Системи безпеки є критично важливим елементом сучасного суспільства, забезпечуючи захист від різноманітних загроз, таких як несанкціоноване проникнення, крадіжки, пожежі чи аварійні ситуації, у житлових, комерційних і промислових об’єктах. Зі зростанням урбанізації, технологічного прогресу та рівня злочинності вимоги до систем безпеки значно зросли. Сучасні рішення повинні не лише ефективно виявляти загрози, але й забезпечувати швидке реагування, інформативність, зручність керування та адаптивність до різних умов експлуатації. Крім того, зростає попит на інтеграцію з мобільними платформами, що дозволяє користувачам отримувати сповіщення та керувати системою віддалено.

Еволюція систем безпеки почалася з простих механічних засобів, таких як замки, решітки та фізична охорона, які домінували до середини XX століття. Перші автоматизовані системи, що використовували електричні датчики, наприклад, магнітні контакти на дверях і вікнах або датчики розбиття скла активували звукові сигнали в разі порушення, але мали суттєві недоліки: обмежену функціональність, відсутність віддаленого доступу та залежність від людського фактора, оскільки реагування вимагало присутності охоронця або власника [3]. Далі із розвитком мікропроцесорних технологій системи безпеки почали використовувати більш досконалі датчики, такі як пасивні інфрачервоні (PIR) та ультразвукові, а також камери відеоспостереження, що значно підвищило їхню ефективність і дозволило фіксувати події з більшою точністю [6].

На початку 2000-х років поширення Інтернету та мобільних технологій змінило парадигму систем безпеки. З’явилися мережеві системи, які забезпечували віддалений моніторинг через вебінтерфейси або спеціалізовані мобільні додатки. Сучасні комерційні рішення, такі як AJAX Systems, Ring,

Arlo та Hikvision, інтегрують різноманітні датчики, камери високої роздільної здатності та хмарні сервіси, що дозволяють користувачам отримувати push-сповіщення, переглядати відео в реальному часі та керувати системою через смартфон. Ці системи також підтримують інтеграцію з екосистемами розумного будинку, наприклад, з Amazon Alexa або Google Home, що розширює їхні можливості [7].

Сучасні системи безпеки можна класифікувати за кількома основними типами:

– традиційні системи безпеки: включають звукові сигналізації, інфрачервоні датчики руху (PIR), магнітні контакти та прості камери. Вони відносно дешеві (вартість близько від $50 до $200) і прості в установці, але мають обмежену інтерактивність, низьку інформативність і часто потребують підключення до охоронних служб, що підвищує експлуатаційні витрати.

– розумні системи безпеки: поєднують датчики, камери, хмарні сервіси та мобільні додатки. Приклади включають AJAX Systems (з підтримкою датчиків руху, диму, затоплення) та Ring (з акцентом на відеодомофони та камери). Такі системи забезпечують віддалений доступ і інтеграцію з розумними пристроями, але їхня висока початкова вартість (близько від $200 до $1000) і залежність від платних хмарних підписок роблять їх менш доступними для масового користувача.

– системи на основі відкритих платформ: використовують мікроконтролери, такі як Arduino або Raspberry Pi, для створення кастомізованих рішень. Вони вирізняються низькою вартістю (близько від $10 до $100) і гнучкістю, дозволяючи інтегрувати різні сенсори та модулі зв’язку (Wi-Fi, GSM) [8]. Однак такі системи часто потребують технічних знань для налаштування та не завжди забезпечують зручний інтерфейс для кінцевого користувача.

Незважаючи на значний прогрес, сучасні системи безпеки мають низку обмежень. Традиційні системи безпеки є недостатньо інформативними, оскільки передають інформацію лише через звуковий сигнал або до

охоронного пульта, що ускладнює оперативне реагування. Вони також вразливі до помилкових спрацьовувань, викликаних, наприклад, рухом тварин чи змінами освітлення. Розумні системи, хоча й пропонують зручний інтерфейс і віддалений доступ, мають високу вартість і залійний доступ, мають високу вартість і залежать від стабільного інтернет-з'єднання, що може бути проблематичним у віддалених районах. Крім того, хмарні сервіси, які використовуються в таких системах, підвищують ризик компрометації даних у разі кібератак. Системи на основі відкритих платформ, попри свою доступність, часто мають обмежений функціонал у порівнянні з комерційними рішеннями та потребують додаткових зусиль для створення зручного інтерфейсу.

Актуальність розробки нових систем безпеки зумовлена кількома ключовими факторами. По-перше, зростання популярності концепції «розумного будинку» створює попит на інтегровані рішення, які поєднують безпеку, автоматизацію та зручність керування. По-друге, сучасні користувачі прагнуть доступних за вартістю систем, які не поступаються комерційним аналогам за ефективністю. По-третє, швидкий розвиток месенджерів, таких як Telegram, відкриває нові можливості для створення інтерактивних систем сповіщення без необхідності розробки власних додатків. Використання лазерних сенсорів, які вирізняються високою точністю та стійкістю до зовнішніх факторів, у поєднанні з мікроконтролерами Arduino та інтеграцією з Telegram дозволяє створити економічно вигідне рішення, що забезпечує оперативне сповіщення, зручне керування та можливість масштабування. Такий підхід є особливо актуальним для приватних користувачів, малого бізнесу та невеликих промислових об'єктів, які потребують надійних, але доступних систем безпеки [9].

1.2 Принципи роботи лазерних сенсорів у системах безпеки

Лазерні сенсори відіграють важливу роль у сучасних системах безпеки завдяки високій точності та швидкості виявлення порушень. Вони створюють невидимі бар'єри, які фіксують несанкціонований доступ або рух у зонах, таких як двері, вікна чи периметри об'єктів. Їхня здатність працювати з мінімальними помилковими спрацьовуваннями робить їх ефективним рішенням для систем, де потрібна надійність і оперативність.

Принцип роботи лазерних сенсорів заснований на взаємодії між емітером і приймачем. Емітер генерує вузьконаправлений лазерний промінь, який спрямовується на приймач, створюючи безперервний сигнал. Переривання променя, наприклад, через рух об'єкта, фіксується приймачем, і система розпізнає це як порушення. Деякі моделі використовують технологію вимірювання часу відбиття променя, що дозволяє визначати відстань до об'єкта або його характеристики. Такі сенсори працюють на відстанях від кількох сантиметрів до десятків метрів і ефективні навіть у темряві чи за слабкого освітлення.

Лазерні сенсори вирізняються високою точністю завдяки вузькому променю, який дозволяє виявляти об'єкти з точністю до міліметра, що значно перевищує можливості інфрачервоних чи ультразвукових датчиків. Вони швидко реагують на переривання променя, забезпечуючи миттєве виявлення загроз. Крім того, сенсори гнучкі в налаштуванні: можна регулювати довжину променя, кут нахилу чи чутливість, що робить їх придатними для охорони вузьких проходів, вікон або великих відкритих територій.

Порівняно з іншими датчиками, лазерні сенсори менш чутливі до змін освітлення, температури чи вологості, що забезпечує стабільну роботу в складних умовах. Їхній вузький промінь мінімізує помилкові спрацьовування, викликані, наприклад, рухом тварин чи повітряними потоками. Завдяки відсутності рухомих частин вони мають тривалий термін служби та потребують мінімального обслуговування. Лазерні сенсори застосовуються

для охорони периметрів, захисту цінних об’єктів або контролю доступу до приміщень, що робить їх універсальними для різних сценаріїв безпеки.

Однак лазерні сенсори мають певні обмеження. Для їхньої роботи потрібне точне вирівнювання між емітером і приймачем, що може ускладнити встановлення на нерівних поверхнях або великих відстанях. Пил, бруд чи волога на лінзах можуть послабити сигнал, викликаючи помилкові спрацьовування [10]. Для вирішення цього застосовуються захисні кожухи або моделі з автоматичним очищенням. Також дзеркальні поверхні поблизу сенсора можуть викликати відбиття променя, що потребує ретельного вибору місця встановлення. Енергоспоживання зазвичай низьке, але моделі з додатковими функціями, такими як Time-of-Flight, можуть вимагати стабільного живлення.

Для ефективного використання лазерних сенсорів необхідно забезпечити точне вирівнювання, уникати перешкод у зоні променя та періодично очищати лінзи від забруднень. У системах безпеки вони часто комбінуються з камерами чи мікроконтролерами, що дозволяє не лише виявляти порушення, але й передавати детальну інформацію, наприклад, фото з місця події. У контексті системи з Telegram-інтеграцією лазерні сенсори є оптимальним вибором, оскільки їхня швидкість і точність забезпечують оперативне сповіщення користувача через месенджер, підвищуючи ефективність реагування на загрози.

1.3 Мікроконтролери Arduino як основа систем безпеки

Мікроконтролери Arduino є популярною платформою для створення систем безпеки завдяки своїй доступності, гнучкості та простоті використання. Вони дозволяють розробляти економічно вигідні рішення, які поєднують обробку сигналів від сенсорів, керування виконавчими пристроями та інтеграцію з мережевими інтерфейсами. У систем безпеки Arduino виконує роль центрального контролера, що забезпечує обробку даних від лазерних

сенсорів, активацію камер чи сигналізацій і передачу сповіщень через Telegram, що робить його оптимальним вибором для реалізації пропонованої системи.

Arduino – це відкрита апаратно-програмна платформа, яка складається з мікроконтролерів (таких як Arduino Uno, Nano, Mega) і середовища розробки Arduino IDE для написання програм на мові C/C++. Платформа підтримує широкий спектр модулів, включаючи сенсори руху, камери, модулі Wi-Fi (наприклад, ESP8266 або ESP32) і GSM (SIM800/900), що дозволяє створювати системи з різноманітним функціоналом. У системах безпеки Arduino обробляє сигнали від лазерних сенсорів, аналізує їх і формує відповідні дії, такі як надсилання сповіщень через месенджер або активація виконавчих пристроїв [11].

Основною перевагою Arduino є його доступність. Вартість базових моделей, таких як ESP32, становить приблизно до 10 доларів, що робить платформу привабливою для створення бюджетних систем безпеки. Простота програмування в Arduino IDE дозволяє навіть користувачам із базовими знаннями створювати функціональні проєкти, а велика спільнота розробників забезпечує доступ до численних бібліотек і навчальних матеріалів. Гнучкість платформи полягає в можливості підключення різноманітних модулів, що дозволяє адаптувати систему до конкретних потреб, наприклад, додавати нові сенсори чи змінювати способи зв’язку (Wi-Fi, GSM, Bluetooth).

У порівнянні з іншими платформами, такими як Raspberry Pi чи STM32, Arduino вирізняється простотою використання та нижчою вартістю. Raspberry Pi, хоча й потужніший за обчислювальною здатністю, але є дорожчим і складнішим у налаштуванні для простих систем безпеки, де потрібна лише базова обробка сигналів. STM32, крім високої продуктивності, також вимагає глибших знань у програмуванні та апаратному забезпеченні, що робить його менш доступним для широкого кола розробників. Arduino, навпаки, поєднує простоту, доступність і достатню функціональність для реалізації систем безпеки [12].

Однак Arduino має певні обмеження в обчислювальної потужністю (наприклад, Arduino Uno має 8-бітний мікроконтролер із частотою 16 МГц і 2 КБ оперативної пам’яті) може створювати проблеми при обробці великих обсягів даних або складних алгоритмів. Залежність від стабільного живлення також є критичною, оскільки перебої в електропостачанні можуть призвести до збоїв у роботі системи. Ці недоліки можна компенсувати оптимізацією програмного коду, використанням моделей із вищою продуктивністю (наприклад, Arduino Mega) або додаванням резервного живлення, такого як акумулятори.

У пропонованій системі безпеки Arduino відіграє ключову роль, обробляючи сигнали від лазерних сенсорів і забезпечуючи зв’язок із Telegram через модуль Wi-Fi або GSM. Наприклад, при виявленні порушення лазерним сенсором мікроконтролер може активувати камеру для зйомки фото чи відео, обробити дані та передати їх через Telegram-бота користувачу. Гнучкість платформи дозволяє легко масштабувати систему, додаючи нові сенсори (наприклад, датчики диму чи температури) або інтегруючи додаткові функції, такі як керування освітленням чи сиренами [13].

Для забезпечення ефективної роботи Arduino в системах безпеки необхідно враховувати кілька аспектів. По-перше, важливо оптимізувати код для мінімізації затримок у реагуванні, що особливо критично для оперативного виявлення загроз. По-друге, вибір відповідного модуля зв’язку (Wi-Fi чи GSM) залежить від умов експлуатації: Wi-Fi підходить для об’єктів із стабільним інтернетом, тоді як GSM забезпечує зв’язок у віддалених районах. По-третє, необхідно передбачити захист від перебоїв живлення, щоб гарантувати безперервну роботу системи. Завдяки своїй універсальності, доступності та широкій підтримці Arduino є оптимальним вибором для створення гнучкої та економічно вигідної системи безпеки з лазерними сенсорами та Telegram-інтеграцією.

1.4 Можливості Telegram API для інтеграції в системи безпеки

Telegram API є потужним інструментом для створення інтерактивних систем безпеки, дозволяючи забезпечити оперативне сповіщення користувачів і зручне керування системою через месенджер. Telegram, як платформа, вирізняється високим рівнем безпеки, широкою популярністю та підтримкою Bot API, що робить його оптимальним вибором для інтеграції в системи безпеки. У контексті пропонованої системи з лазерними сенсорами, Telegram API забезпечує передачу сповіщень про порушення, надсилання фото чи відео з місця події та обробку команд користувача, що підвищує ефективність і зручність моніторингу.

Telegram Bot API дозволяє створювати ботів, які взаємодіють із користувачами через текстові повідомлення, кнопки, зображення чи інші мультимедійні дані. У системах безпеки боти можуть виконувати кілька ключових функцій: надсилати миттєві сповіщення про спрацювання лазерних сенсорів, передавати зображення з камер, отримувати команди для активації чи деактивації системи, а також надавати статус сенсорів чи інших компонентів. Наприклад, у разі виявлення порушення бот може надіслати повідомлення з текстом «Виявлено рух у зоні 1» разом із фотографією, а користувач може відповісти командою для увімкнення сирени чи перевірки стану системи.

Однією з головних переваг Telegram є його доступність і простота використання. Месенджер працює на всіх основних платформах (iOS, Android, Windows, macOS), що дозволяє користувачам отримувати сповіщення та керувати системою з будь-якого пристрою. Завдяки підтримці Bot API розробка інтерактивного інтерфейсу не вимагає створення окремого мобільного додатка, що значно знижує витрати та час на реалізацію. Простота створення бота робить Telegram привабливим для інтеграції навіть у бюджетні системи безпеки.

Ще однією перевагою є безпека передачі даних. Telegram використовує протокол MTProto для шифрування повідомлень, а для секретних чатів підтримує end-to-end шифрування, що забезпечує захист інформації від перехоплення. Це особливо важливо для систем безпеки, де сповіщення можуть містити конфіденційні дані, такі як зображення з камер. Крім того, Telegram дозволяє створювати закриті чати або обмежувати доступ до бота, що мінімізує ризик несанкціонованого використання.

Порівняно з іншими месенджерами, такими як WhatsApp чи Viber, Telegram вирізняється більш відкритим і гнучким API. WhatsApp Business API, хоча й підтримує створення ботів, має складніший процес інтеграції, потребує офіційної реєстрації бізнес-акаунта та має обмеження на масові розсилки. Viber API менш розвинений і орієнтований переважно на комерційні чат-боти, а не на технічні інтеграції. Telegram, навпаки, пропонує широкі можливості для кастомізації: від простих текстових повідомлень до інтерактивних меню з кнопками чи навіть Mini Apps, які можуть виконувати складніші функції [14].

У пропонованій системі безпеки Telegram API відіграє центральну роль у забезпеченні інтерактивності. Бот може надсилати сповіщення про спрацювання лазерних сенсорів у реальному часі, передавати фото чи відео з камери, а також обробляти команди, такі як увімкнення сигналізації, перевірка стану сенсорів чи вимкнення системи. Наприклад, користувач може отримати повідомлення про порушення і відповісти командою для активації камери, після чого бот надішле зображення з місця події. Такий підхід значно спрощує взаємодію з системою порівняно з традиційними методами, такими як дзвінки від охоронної служби чи використання спеціалізованих додатків.

Для ефективної інтеграції Telegram API необхідно оптимізувати роботу бота, щоб мінімізувати затримки в надсиланні сповіщень і забезпечити надійну обробку команд. Важливо також передбачити захист від зловживань, наприклад, обмеживши доступ до бота лише для авторизованих користувачів. Завдяки своїй простоті, безпеці та гнучкості Telegram API є оптимальним

вибором для створення інтерактивної системи безпеки, яка поєднує оперативне сповіщення, зручне керування та економічну ефективність [15].

1.5 Порівняльний аналіз існуючих систем безпеки та обґрунтування вибору підходу

Для оцінки унікальності пропонованої системи безпеки з лазерними сенсорами та Telegram-інтеграцією проведено порівняльний аналіз із існуючими аналогами, які поділяються на три основні типи: традиційні системи безпеки, розумні системи безпеки та системи на основі відкритих платформ, таких як Arduino. Аналіз враховує ключові критерії, зокрема точність виявлення, швидкість реагування, зручність керування, масштабованість і відносну вартість, щоб визначити переваги та недоліки кожного підходу та обґрунтувати доцільність запропонованого рішення.

Традиційні системи безпеки, наприклад сигналізації від Paradox або DSC, використовують звукові сирени, інфрачервоні датчики руху (PIR) і магнітні контакти. Вони відносно дешевші та прості в установці, що робить їх популярними для базового захисту будинків чи офісів. Однак їхня функціональність обмежена: інформація передається лише через звуковий сигнал або до охоронної служби, що знижує інформативність і уповільнює реагування [16]. Інфрачервоні датчики схильні до помилкових спрацьовувань через зміни температури чи рух тварин, а відсутність віддаленого доступу без додаткових модулів робить ці системи менш зручними для сучасних користувачів.

Розумні системи безпеки, такі як AJAX Systems, Ring або Arlo, пропонують значно ширший функціонал. Вони інтегрують датчики руху, диму, затоплення, камери високої роздільної здатності та хмарні сервіси, дозволяючи отримувати сповіщення через мобільні додатки та переглядати відео в реальному часі. Ці системи зручні, підтримують інтеграцію з екосистемами розумного будинку та мають високу точність завдяки сучасним

сенсорам. Проте вони значно дорожчі, а необхідність платних хмарних підписок додатково підвищує витрати. Залежність від стабільного інтернет-з'єднання обмежує їхнє використання у віддалених районах, а хмарні сервіси підвищують ризик компрометації даних.

Системи на основі відкритих платформ, таких як Arduino чи Raspberry Pi, є економічно вигідною альтернативою. Вони використовують доступні мікроконтролери та сенсори (PIR, ультразвукові) для створення кастомізованих рішень і є найдешевшими серед аналогів. Такі системи гнучкі, дозволяють додавати нові модулі та адаптувати функціонал під конкретні потреби. Однак їхня точність часто нижча через менш прецизійні сенсори, а налаштування вимагає технічних знань, що обмежує аудиторію до ентузіастів і розробників. Більшість таких систем не інтегруються з інтерактивними інтерфейсами, як-от месенджери, що знижує зручність керування.

Пропонована система безпеки з лазерними сенсорами, Arduino та Telegram-інтеграцією поєднує переваги аналогів, усуваючи їхні ключові недоліки. Лазерні сенсори забезпечують високу точність виявлення і стійкість до зовнішніх факторів, таких як зміни освітлення чи температури, що перевершує інфрачервоні чи ультразвукові датчики, використовувані в традиційних системах і Arduino-проєктах. Швидкість реагування є миттєвою: при спрацюванні сенсора Telegram-бот надсилає сповіщення в реальному часі, що швидше, ніж у традиційних систем, і порівнянно з розумними системами. Зручність керування досягається завдяки Telegram-боту, який дозволяє отримувати сповіщення, переглядати фото чи відео та надсилати команди з будь-якого пристрою, що простіше, ніж спеціалізовані додатки розумних систем.

Пропонована система є дешевшою за розумні системи та порівнянною за вартістю з традиційними рішеннями й іншими Arduino-проєктами. Її масштабованість дозволяє додавати нові сенсори, наприклад диму чи температури, або інтегрувати додаткові функції, як-от керування освітленням, що робить її конкурентною з Arduino-проєктами та розумними системами.

Telegram-інтеграція усуває потребу в розробці власного додатка чи хмарних сервісів, знижуючи витрати та ризики безпеки даних порівняно з розумними системами.

Вибір підходу обґрунтовується поєднанням високої точності лазерних сенсорів, доступності та гнучкості Arduino, а також простоти й безпеки Telegram-інтеграції. Це дозволяє створити економічно вигідну систему, яка забезпечує оперативне сповіщення, зручне керування та адаптивність до різних умов експлуатації. Унікальність рішення полягає в інтеграції сучасних технологій із бюджетною платформою, що робить систему доступною для приватних користувачів, малого бізнесу та невеликих промислових об’єктів.

1.6 Постановка задачі дослідження

Метою дослідження є розробка комплексної системи безпеки, яка поєднує лазерні сенсори, мікроконтролер Arduino та інтеграцію з Telegram для забезпечення високоточної, оперативної та зручної системи моніторингу й керування. Система спрямована на виявлення несанкціонованого доступу або руху в контрольованих зонах із подальшим сповіщенням користувача через месенджер, що дозволяє швидко реагувати на загрози та адаптувати рішення до різних умов експлуатації, таких як охорона приватних будинків, офісів чи невеликих промислових об’єктів.

Для досягнення поставленої мети визначено такі задачі дослідження:

– проаналізувати принципи роботи лазерних сенсорів, їхні технічні характеристики та особливості застосування в системах безпеки для забезпечення високої точності й надійності виявлення;

– дослідити можливості Telegram Bot API для створення інтерактивного інтерфейсу, який забезпечує оперативне надсилання сповіщень, передачу мультимедійних даних і обробку команд користувача [17];

– розробити апаратну частину системи на базі мікроконтролера Arduino, включаючи підключення лазерних сенсорів, камери для фіксації подій і модуля зв’язку Wi-Fi для інтеграції з Telegram;

– реалізувати програмне забезпечення для обробки сигналів від сенсорів, формування сповіщень і взаємодії з Telegram-ботом, забезпечуючи стабільну та швидку роботу системи;

– провести тестування системи в різних умовах для оцінки її точності, швидкості реагування, надійності та зручності керування, а також визначити можливі обмеження та шляхи їх усунення.

Очікуваним результатом є створення гнучкої та економічно вигідної системи безпеки, яка поєднує високу точність лазерних сенсорів, простоту й доступність платформи Arduino та інтерактивність Telegram-інтерфейсу. Система має забезпечувати миттєве сповіщення користувача про порушення, передачу візуальної інформації з місця події та можливість віддаленого керування, що робить її придатною для широкого спектра застосувань, від побутового до комерційного використання.

2 ПРОЕКТУВАННЯ СИСТЕМИ

Для побудови мікропроцесорної системи безпеки з лазерними сенсорами та Telegram-інтеграцією необхідно розробити дві системи – апаратну (відповідає за виявлення вторгнень за допомогою лазерного сенсора, зйомку фото та звукову сигналізацію) і програмну (відповідає за обробку даних від апаратної частини, віддалене керування через Telegram та відправку сповіщень).

2.1 Розробка апаратної частини системи безпеки

Апаратна частина є фундаментом усієї системи, і її розробка вимагала не лише ретельного планування, а й практичного тестування для забезпечення стабільності й ефективності роботи. Система побудована таким чином, щоб бути одночасно функціональною, енергоефективною та доступною для реалізації в реальних умовах, що робить її перспективною для використання в побутових або невеликих комерційних сценаріях.

Основою апаратного забезпечення слугують два мікроконтролери: ESP32 (див. рис. 2.1) та ESP32-CAM (див. рис. 2.2), які взаємодіють між собою через протокол ESP-NOW. Вибір цих пристроїв не був випадковим — він зумовлений їхніми унікальними характеристиками, що ідеально відповідають потребам системи безпеки.

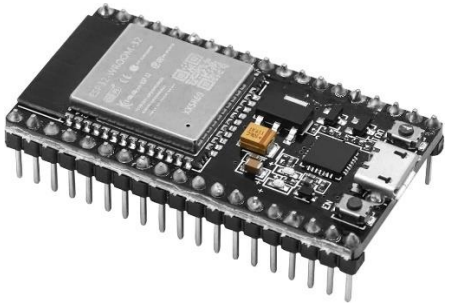

Рисунок 2.1 – Зовнішній вигляд плати ESP32

Рисунок 2.2 – Зовнішній вигляд плати ESP32-CAM

ESP32 — це потужний мікроконтролер із вбудованими модулями Wi-Fi і Bluetooth, а також підтримкою ESP-NOW, що дозволяє організувати швидкий і надійний бездротовий зв’язок із мінімальним енергоспоживанням. Цей мікроконтролер відповідає за обробку сигналів від лазерного сенсора та керування звуковим сповіщенням через базер, що робить його центральним вузлом системи для аналізу подій у реальному часі [18].

Характеристики ESP32 наведені в табл. 2.1.

Таблиця 2.1 – Характеристики плати ESP32

Параметр	Значення
Вартість	150 – 200 грн
Мікроконтролер	Xtensa Dual-Core 32-bit LX6
Напруга живлення(рекомендована)	3.3 В
Напруга живлення(максимальна)	3.6 В
Цифрові входи/виходи	34 (залежить від конфігурації)
Аналогові входи	18 (12-біт ADC)
Максимальний ток одного виводу	40 мА
Flash-пам’ять	4 МБ
SRAM	520 КБ
Протоколи зв’язку	ESP-NOW, HTTP, MQTT
Тактова частота	До 240 МГц

Продовження таблиці 2.1

Параметр	Значення
Підтримувані інтерфейси	Wi-Fi (802.11 b/g/n), Bluetooth 4.2/BLE, SPI, I2C, UART, ESP-NOW

ESP32-CAM, своєю чергою, доповнює систему завдяки вбудованій камері OV2640 із роздільною здатністю до 1600x1200 пікселів і слоту для SD-карти, що дає змогу не лише знімати фото, а й зберігати їх локально за потреби. Цей пристрій також забезпечує зв’язок із Telegram через Wi-Fi, надсилаючи зображення користувачу в разі виявлення порушення. Використання двох окремих мікроконтролерів дозволяє розподілити задачі між ними: ESP32 зосереджується на сенсорній логіці та локальних сповіщеннях, а ESP32-CAM — на фіксації подій і віддаленій комунікації. Така архітектура підвищує продуктивність системи, зменшує ймовірність перевантаження одного пристрою та спрощує процес налаштування й масштабування.

Характеристики ESP32-CAM наведені в таблиці 2.2.

Таблиця 2.2 – Характеристики плати ESP32-CAM

Параметр	Значення
Вартість	185 – 300 грн
Мікроконтролер	ESP32-S Dual-Core 32-bit LX6
Напруга живлення(рекомендована)	5 В
Напруга живлення(максимальна)	5.5 В
Цифрові входи/виходи	10 (доступних GPIO)
Аналогові входи	6 (12-біт ADC)
Максимальний ток одного виводу	40 мА
Flash-пам’ять	4 МБ
SRAM	520 КБ
Протоколи зв’язку	ESP-NOW, SPI
Тактова частота	До 240 МГц

Продовження таблиці 2.2

Параметр	Значення
Підтримувані інтерфейси	Wi-Fi (802.11 b/g/n), Bluetooth 4.2/BLE, SPI, I2C, UART, ESP-NOW, SDIO

Ключовими компонентами системи, крім мікроконтролерів, є лазерний сенсор (див. рис. 2.3), камера, базер і спалах, кожен із яких відіграє важливу роль у загальній функціональності.

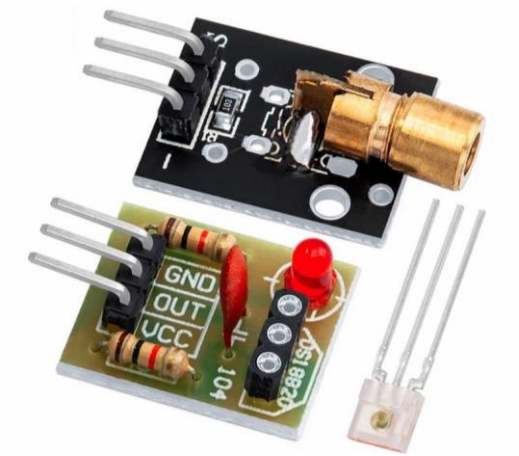

Рисунок 2.3 – Зовнішній вигляд лазерного сенсора

Лазерний сенсор складається з лазерного діода потужністю 5 мВт із довжиною хвилі 650 нм у червоному спектрі та фототранзистора, які разом формують невидимий бар’єр для моніторингу контрольованої зони. Лазерний діод підключений до одного з вихідних пінів ESP32, а саме до GPIO 25, через що мікроконтролер може вмикати або вимикати його за потреби. Фототранзистор, розташований навпроти діода, підключений до GPIO 33 у режимі INPUT_PULLUP, що забезпечує стабільне зчитування сигналу завдяки внутрішньому підтягуючому резистору. Принцип роботи цього сенсора доволі простий, але ефективний: лазерний промінь постійно спрямований на

фототранзистор, і коли хтось чи щось перетинає цей промінь, сигнал на фототранзисторі змінюється, що інтерпретується системою як подія порушення. У такому разі ESP32 активує базер — невеликий звуковий пристрій, підключений до GPIO 13, який видає гучний сигнал протягом 5 секунд, якщо система перебуває в режимі підвищеного захисту. Камера OV2640, інтегрована в ESP32-CAM, фіксує зображення зони порушення, а спалах, підключений до GPIO 4, автоматично вмикається в умовах недостатнього освітлення, щоб забезпечити чіткість знімків навіть у темряві.

Характеристики лазерного сенсора наведені в таблиці 2.3.

Таблиця 2.3 – Характеристики лазерного сенсора

Параметр	Значення
Вартість	50 – 100 грн
Тип компонента	Лазерний діод (650 нм, червоний), фототранзистор
Потужність	5 мВт
Довжина хвилі	650 нм (червоний)
Напруга живлення	3.3 В – 5 В
Струм споживання	~20-40 мА
Сигнал	Цифровий (HIGH/LOW)

Процес підключення компонентів до мікроконтролерів був ретельно продуманий, щоб гарантувати їхню коректну взаємодію. На ESP32 лазерний діод живиться через GPIO 25, який налаштований як вихідний пін, і керується простими командами високого або низького рівня сигналу. Фототранзистор, підключений до GPIO 33, постійно зчитує стан променя, а внутрішній підтягуючий резистор запобігає хаотичним стрибкам сигналу в разі шумів. Базер, також керований через вихідний пін GPIO 13, активується лише за командою від ESP32, коли система реєструє переривання променя. На ESP32-CAM камера OV2640 використовує стандартний набір пінів, визначених у

прошивці, і не потребує додаткового підключення, тоді як спалах на GPIO 4 вмикається програмно перед зйомкою, якщо датчик світла (опціонально підключений до системи) фіксує низький рівень освітленості. Зв’язок між ESP32 і ESP32-CAM здійснюється через ESP-NOW — протокол однорангового зв'язку, який забезпечує обмін даними між платами розробки на базі ESP без використання спеціального маршрутизатора WiFi [19]. Це значно зменшує затримки в порівнянні з традиційними мережевими протоколами та економить енергію, що особливо важливо для автономних систем. Для зв’язку з Telegram ESP32-CAM підключається до локальної Wi-Fi-мережі, використовуючи заздалегідь задану назву мережі і пароль, після чого відправляє зображення через спеціально створеного бота.

Схема підключення охоронної системи наведена на рисунку 2.4.

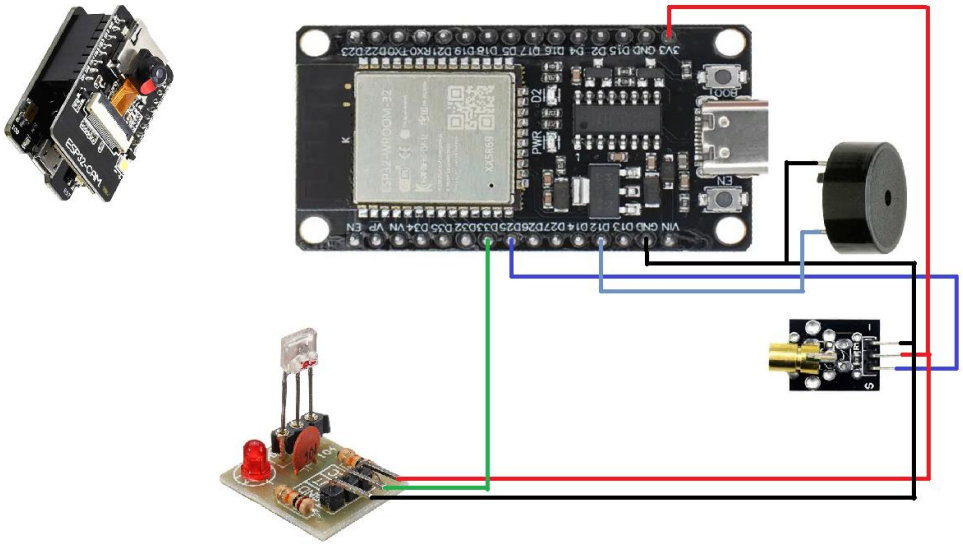

Рисунок 2.4 – Схема підключення охоронної системи

Для реалізації апаратної частини комплексної системи відповідної до вищевказаної схеми підключено (див. рис. 2.5).

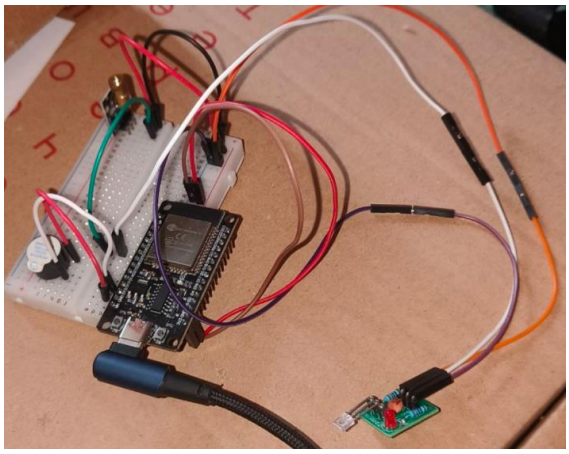

Рисунок 2.5 – Підключення модулів між собою відповідно до схеми

Живлення системи — ще один важливий аспект, який потребував уваги під час розробки. Обидва мікроконтролери можуть працювати від стандартного джерела живлення на 5 В, яке подається через USB-порт або зовнішній адаптер. Однак, щоб зробити систему більш універсальною та придатною для використання в умовах без доступу до розетки, було передбачено можливість підключення акумулятора, наприклад, літій-іонного на 3.7 В або звичайний павербанк. Такий підхід дозволяє системі працювати автономно протягом кількох годин, залежно від ємності акумулятора, а режим сну для ESP32 у періоди неактивності додатково подовжує час роботи. Усі дроти та з’єднання між компонентами були ретельно ізольовані та закріплені, щоб уникнути коротких замикань чи обривів під час експлуатації.

Щоб скласти апаратну частину, потрібно було лазерний діод і фототранзистор розмістити на протилежних сторонах умовної зони контролю, наприклад, дверного отвору чи проходу. Камеру на ESP32-CAM встановили так, щоб її поле зору охоплювало всю зону потенційного порушення — зазвичай на висоті близько метра від підлоги з кутом нахилу, що забезпечує максимальну видимість. Базер розмістили в зручному місці, де його звук буде добре чутний, наприклад, поруч із входом або в центральній частині приміщення. Обидва мікроконтролери розмістили в межах дії ESP-NOW, яка

в приміщенні може сягати до 100 метрів і вище за відсутності значних перешкод, що робить систему гнучкою для різних конфігурацій простору.

Після завершення складання я протестував імітуючи реальні сценарії використання. Лазерний сенсор бездоганно фіксував перетин променя, передаючи сигнал на ESP32, який активував базер і надсилав команду через ESP-NOW до ESP32-CAM. Камера, у свою чергу, миттєво робила знімок і відправляла його в Telegram, навіть у темряві завдяки спалаху. Результати тестування показали, що система працює стабільно, реагуючи на порушення протягом 1-2 секунд, а якість зображень залишається достатньою для ідентифікації об’єктів чи осіб у зоні дії.

2.2 Реалізація програмної частини

Розробка програмного забезпечення для системи безпеки, яка використовує лазерні сенсори та інтегрується з Telegram, стала ключовим етапом створення проєкту. Програмна частина забезпечує злагоджену роботу апаратних компонентів, таких як лазерний сенсор, камера і buzzer, а також надає користувачу зручний інтерфейс для віддаленого керування через Telegram-бот. Це програмне забезпечення об’єднує два мікроконтролери — ESP32 і ESP32-CAM, кожен із яких виконує власний набір завдань: перший обробляє сигнали сенсора та активує звуковий сигнал, а другий відповідає за зйомку зображень і комунікацію з користувачем.

Програмне забезпечення було створено в середовищі Arduino IDE, яке обрано через його простоту, широку підтримку бібліотек і сумісність із мікроконтролерами ESP32. Arduino IDE дозволило швидко писати, тестувати та завантажувати прошивки, що було особливо важливо на етапі ітеративної розробки. Використання мови програмування C++ забезпечило гнучкість у реалізації алгоритмів і точний контроль над апаратними ресурсами.

Для забезпечення функціональності системи було використано кілька ключових бібліотек. Бібліотека esp_now.h реалізує бездротовий зв’язок між

ESP32 і ESP32-CAM через протокол ESP-NOW, який вирізняється швидкістю передачі даних і низьким енергоспоживанням, дозволяючи системі працювати навіть без активного WiFi-з'єднання[19]. Бібліотека WiFi.h підключає ESP32-CAM до локальної мережі, що дуже необхідно для роботи Telegram-бота. Бібліотека UniversalTelegramBot.h забезпечує інтеграцію з Telegram API, дозволяючи надсилати зображення та отримувати команди від користувача[17]. Також, esp_camera.h керує камерою, відповідаючи за зйомку та обробку зображень. Ці бібліотеки є добре документованими, стабільними та широко застосовуваними, що зробило їх гарним вибором для проєкту.

Архітектура програмного забезпечення побудована за модульним принципом, що дозволяє чітко розмежувати функціональність і полегшує подальше масштабування. Основна ідея полягала в розподілі задач між двома мікроконтролерами для підвищення продуктивності. ESP32 обробляє сигнали від фототранзистора, виявляючи переривання лазерного променя, і надсилає команди до ESP32-CAM через ESP-NOW. ESP32-CAM, у свою чергу, відповідає за зйомку фото, їхню передачу через Telegram і обробку команд користувача. Такий розподіл обов'язків зменшує навантаження на кожен пристрій і підвищує швидкість реакції системи. Модульна структура також спрощує модифікацію окремих компонентів, наприклад, додавання нових функцій чи сенсорів, без необхідності переробляти всю програму.

Одним із центральних елементів програмного забезпечення є алгоритм моніторингу лазерного сенсора, який працює на ESP32. Цей алгоритм постійно зчитує стан фототранзистора через цифровий пін GPIO 33. Якщо лазерний промінь перервано, система перевіряє активний режим захисту: у базовому режимі (рівень 1) надсилається команда на зйомку фото, а в підвищеному (рівень 2) додатково активується buzzer на 5 секунд. Для уникнення хибних спрацьовувань передбачено мінімальний інтервал у 5 секунд між послідовними командами. Затримка в 100 мілісекунд між циклами забезпечує стабільність роботи, не перевантажуючи мікроконтролер. Цей

алгоритм дозволяє системі миттєво реагувати на порушення, забезпечуючи надійність і точність.

На ESP32-CAM реалізовано алгоритм обробки команд через Telegram-бот, який є основним інтерфейсом для користувача. Бот підключається до WiFi-мережі, ініціалізується з токеном, отриманим від BotFather, і кожну секунду перевіряє нові повідомлення. Підтримуються чотири команди: photo для зйомки знімка, level1 для активації базового режиму, level2 для підвищеного режиму та alarm_off для вимкнення сигналізації. Кожна команда проходить перевірку авторизації за chat ID, що гарантує безпеку доступу. Якщо команда правильна, то бот виконує дію, наприклад, надсилає команду через ESP-NOW до ESP32 або робить фото та відправляє його в чат. Цей алгоритм забезпечує зручне віддалене керування системою з будь-якого пристрою, підключеного до Telegram.

Моніторинг лазерного сенсора (лістинг 2.1) реалізований у ESP32.

```
#define SENSOR_PIN 33
#define BUZZER_PIN 13
int protectionLevel = 0;
unsigned long lastPhotoTime = 0;
unsigned long lastBuzzerTime = 0;
const long photoInterval = 5000;
const long buzzerDuration = 5000;
void loop() {
  if (protectionLevel > 0) {
    bool sensorState = digitalRead(SENSOR_PIN);
    unsigned long currentTime = millis();
       if (sensorState == HIGH) { // Промінь перервано
      if (currentTime - lastPhotoTime >= photoInterval) {
        Serial.println("Лазер перервано, надсилаю команду на
фото");
        strcpy(msgToSend.command, "photo");
        esp_now_send(espCamAddress, (uint8_t*)&msgToSend,
sizeof(msgToSend));
        lastPhotoTime = currentTime;
        if (protectionLevel == 2) {
          digitalWrite(BUZZER_PIN, HIGH);
          lastBuzzerTime = currentTime;
    }
       }
```

Лістинг 2.1 – Моніторинг лазерного сенсора

```
        } else if (protectionLevel == 2 &&
digitalRead(BUZZER_PIN) == HIGH &&
                   currentTime - lastBuzzerTime >=
buzzerDuration) {
          digitalWrite(BUZZER_PIN, LOW); // Вимкнути buzzer
        }
      }
      delay(100); // Затримка для стабільності
    }
```

Лістинг 2.1, аркуш 2

Цей код демонструє, як ESP32 відстежує стан фототранзистора, надсилає команду на зйомку фото через ESP-NOW і активує buzzer у режимі підвищеного захисту. Затримка в 100 мс забезпечує стабільність роботи системи.

Для ESP32-CAM було реалізовано обробку команд Telegram-бота (лістинг 2.2).

```
#include <WiFi.h>
#include <UniversalTelegramBot.h>
#define CHAT_ID "123456789"
#define BOT_TOKEN "YOUR_BOT_TOKEN"
WiFiClientSecure client;
UniversalTelegramBot bot(BOT_TOKEN, client);
void handleNewMessages(int numNewMessages) {
  for (int i = 0; i < numNewMessages; i++) {
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != CHAT_ID) {
      bot.sendMessage(chat_id, "Доступ заборонено", "");
      continue;
    }
    String text = bot.messages[i].text;
    Serial.println("Отримано команду: " + text);
     if (text == "/photo") {
      sendPhoto = true; // Прапорець для зйомки фото
      bot.sendMessage(CHAT_ID, "Фото відправлено", "");
    } else if (text == "/level1") {
      strcpy(msgToSend.command, "level1");
      esp_now_send(espLaserAddress, (uint8_t*)&msgToSend,
sizeof(msgToSend));
     bot.sendMessage(CHAT_ID, "Режим 1: лазер+камера", "");
    } else if (text == "/level2") {
```

Лістинг 2.2 – Перевірка та обробка команд бота

```
            strcpy(msgToSend.command, "level2");
            esp_now_send(espLaserAddress, (uint8_t*)&msgToSend,
sizeof(msgToSend));

            bot.sendMessage(CHAT_ID, "Режим 2:
лазер+камера+базер", "");
          } else if (text == "/alarm_off") {
            strcpy(msgToSend.command, "alarm_off");
            esp_now_send(espLaserAddress, (uint8_t*)&msgToSend,
sizeof(msgToSend));
            bot.sendMessage(CHAT_ID, "Сигналізація вимкнена", "");
          }
        }
      }
      void loop() {
        int numNewMessages =
bot.getUpdates(bot.last_message_received + 1);
        if (numNewMessages) {
          handleNewMessages(numNewMessages);
        }
        if (sendPhoto) {
          captureAndSendPhoto();
          sendPhoto = false;
        }
        delay(1000);
      }
```

Лістинг 2.2, аркуш 2

Цей код обробляє команди від Telegram-бота, перевіряє авторизацію та надсилає відповідні повідомлення або команди через ESP-NOW. Функція captureAndSendPhoto (лістинг 2.3) відповідає за зйомку та передачу зображень.

```
      void captureAndSendPhoto() {
        camera_fb_t *fb = esp_camera_fb_get();
        if (!fb) {
          Serial.println("Помилка зйомки");
          bot.sendMessage(CHAT_ID, "Не вдалося зробити фото", "");
          return;
        }
        bot.sendPhoto(CHAT_ID, fb->buf, fb->len);
        esp_camera_fb_return(fb);
        Serial.println("Фото відправлено");
      }
```

Лістинг 2.3 – Функція зйомки та передачі фото

Ця функція використовує бібліотеку esp_camera.h для отримання кадру з камери та надсилає його через Telegram API (див. рис. 2.6).

Рисунок 2.6 – Приклад зображення, отриманого через Telegram-бот

Розроблене програмне забезпечення забезпечує стабільну роботу системи безпеки. Лазерний сенсор надійно виявляє переривання променя, що активує зйомку фото та, за потреби, звуковий сигнал. Telegram-бот дозволяє користувачу отримувати зображення, змінювати режими захисту або вимикати сигналізацію з будь-якого місця, де є доступ до інтернету. Використання ESP-NOW забезпечило швидкість реакції системи на рівні біля 50 мілісекунд, що є значною перевагою порівняно з системами, які залежать від WiFi. У тестах команда photo надсилала зображення протягом 1-2 секунд, а зміна режимів відбувалася миттєво.

Програмне забезпечення є гнучким і готовим до масштабування. Наприклад, можна додати збереження зображень на SD-карту, щоб система працювала без WiFi, або інтегрувати датчики руху чи температури для розширення функціоналу. Бібліотека UniversalTelegramBot дозволяє легко додавати нові команди, наприклад, для активації нового сенсора чи

надсилання звітів. Використання Arduino IDE спростило розробку, а модульна структура коду забезпечує можливість швидкої адаптації до нових вимог.

На додачу, є кілька напрямів для вдосконалення. Наприклад, покращення якості зображень через використання камери з вищою роздільною здатністю або декілька лазерів для збільшення точності. Інтеграція алгоритмів машинного навчання для аналізу зображень могла б додати інтелектуальні функції, такі як розпізнавання осіб чи об’єктів.

Реалізоване програмне забезпечення забезпечує ефективну роботу системи безпеки, поєднуючи швидкість реакції, надійність і зручність керування. Воно є міцною основою для подальших досліджень і вдосконалень, відкриваючи можливості для створення більш універсальних і потужних систем захисту. Повний код програмного забезпечення наведено в додатках А та Б.

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

Для початку роботи необхідно підготувати апаратні компоненти. Спершу підключіть модуль ESP32-CAM до USB-порту комп’ютера, щоб завантажити прошивку та перевірити функціонування камери. Далі під’єднайте модуль ESP32 до джерела живлення, наприклад, через USB або зовнішній акумулятор. Лазерний діод і фототранзистор слід розмістити на протилежних сторонах контрольованої зони, наприклад, дверного отвору, і вирівняти так, щоб лазерний промінь точно потрапляв на фототранзистор. Якщо вирівнювання виконано неправильно, система може некоректно виявляти порушення, про що буде повідомлено в серійному моніторі Arduino IDE під час тестування. Звуковий пристрій має бути підключений до відповідного піна ESP32, а камера OV2640 на ESP32-CAM — готова до роботи. Після завершення апаратного налаштування відкрийте додаток Telegram на смартфоні чи комп’ютері, знайдіть бот системи за його унікальним токеном, отриманим через BotFather, і додайте його до своїх чатів (див. рисунок 3.1).

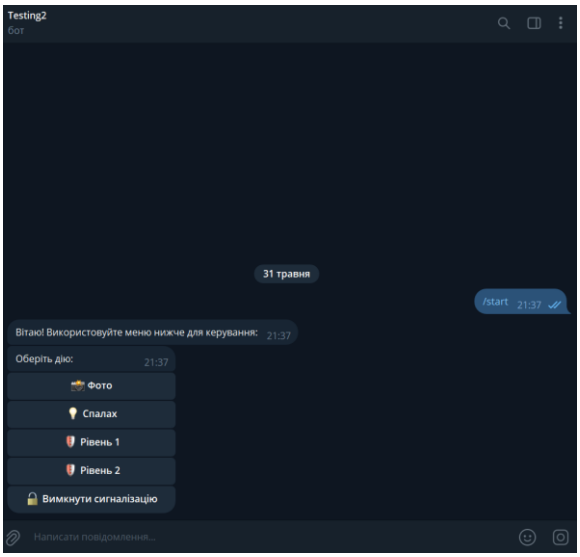

Рисунок 3.1 – Початковий вигляд чату з Telegram-ботом системи безпеки

У чаті з ботом надішліть команду start, щоб розпочати взаємодію. Бот відповість привітальним повідомленням і надасть список доступних команд: photo для зйомки знімка, level1 для активації базового режиму захисту, level2 для підвищеного режиму з використанням buzzer'а та alarm_off для вимкнення системи. Якщо бот не реагує, перевірте, чи модуль ESP32-CAM підключений до WiFi-мережі. Для цього переконайтеся, що в прошивці ESP32-CAM правильно вказані SSID і пароль вашої WiFi-мережі. У разі невдалого підключення бот надішле повідомлення про помилку, наприклад: «Не вдалося підключитися до WiFi». Після успішного підключення до WiFi система готова до використання. Для активації захисту надішліть команду level1, щоб увімкнути базовий режим, у якому лазерний сенсор реагує на переривання променя, а ESP32-CAM робить знімок. Бот підтвердить активацію повідомленням (див. рисунок 3.2).

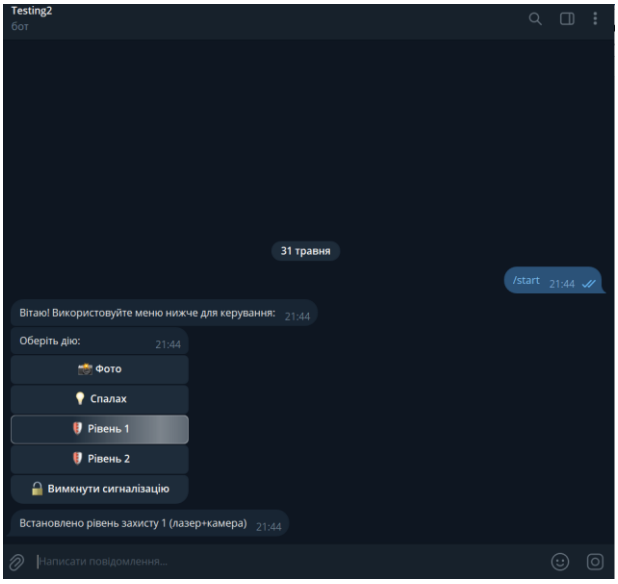

Рисунок 3.2 – Підтвердження активації режиму захисту 1

Якщо потрібен посилений захист, надішліть команду level2, яка додатково активує buzzer при виявленні порушення. У цьому випадку бот відповість: «Режим 2: лазер+камера+базер». Якщо команда введена

неправильно або чат не авторизований (наприклад, chat ID не збігається з указаним у прошивці), бот повідомить: «Доступ заборонено» (див. рисунок 3.3).

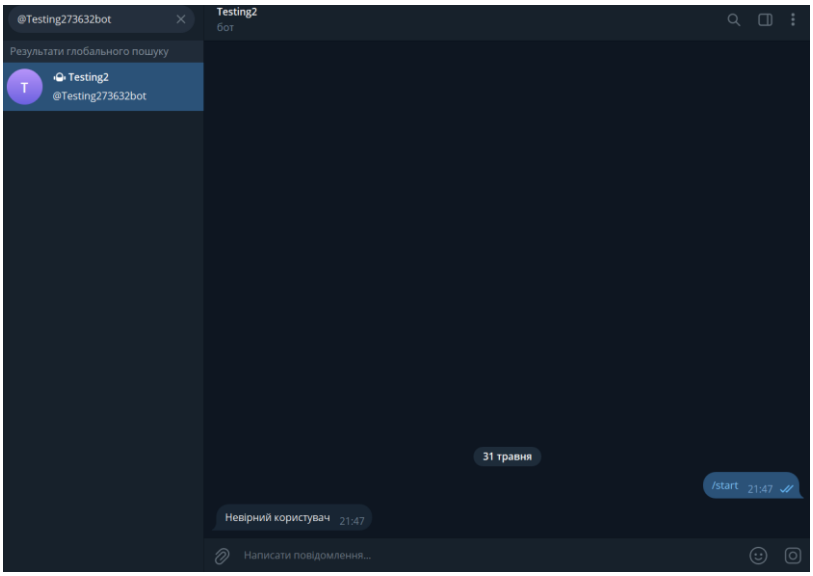

Рисунок 3.3 – Повідомлення про заборону доступу

Коли система перебуває в активному стані, вона автоматично відстежує переривання лазерного променя. При виявленні порушення ESP32 надсилає команду до ESP32-CAM через протокол ESP-NOW, який забезпечує швидку передачу даних без залежності від WiFi. Камера робить знімок, і зображення надсилається в чат Telegram протягом 1-2 секунд за умови стабільного інтернет-з’єднання. У режимі підвищеного захисту додатково вмикається buzzer на 5 секунд, створюючи звукове сповіщення. Користувач отримує зображення з позначкою часу події, що дозволяє точно ідентифікувати момент порушення (див. рисунок 3.4).

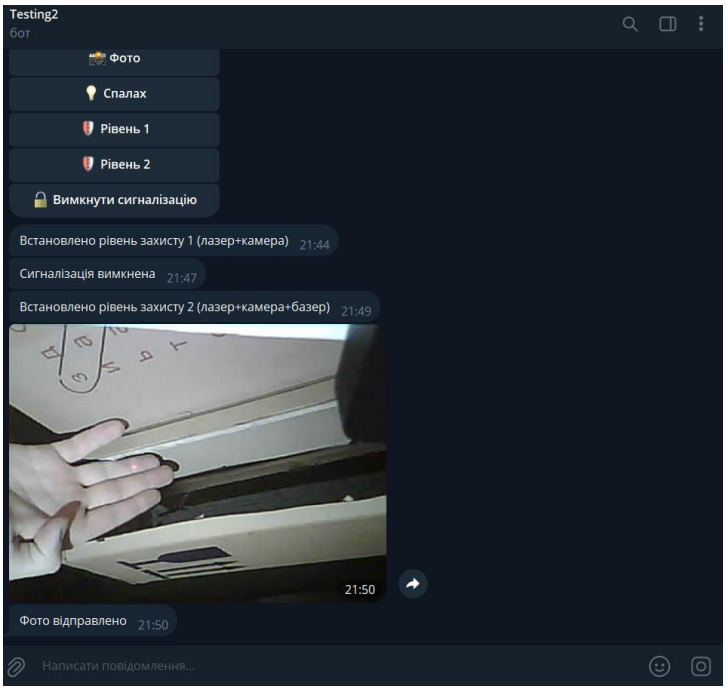

Рисунок 3.4 – Зображення, отримане через Telegram після переривання променя

Для перевірки роботи системи без реального порушення можна надіслати команду photo, яка змусить ESP32-CAM зробити знімок і надіслати його в чат. Це корисно для тестування камери або перевірки освітлення в контрольованій зоні. Якщо зображення не надходить, перевірте, чи камера правильно ініціалізована, і переконайтеся, що WiFi-з’єднання стабільне. У разі помилки зйомки бот надішле повідомлення: «Не вдалося зробити фото».

Щоб вимкнути систему, надішліть команду alarm_off. Бот підтвердить вимкнення повідомленням: «Сигналізація вимкнена», а ESP32 припинить моніторинг сенсора та активацію buzzer’а. Ця команда корисна, коли потрібно тимчасово деактивувати захист, наприклад, під час перебування в контрольованій зоні. Після вимкнення система не реагуватиме на переривання променя, але залишатиметься підключеною до Telegram, дозволяючи надсилати команду photo для перевірки (див. рисунок 3.5).

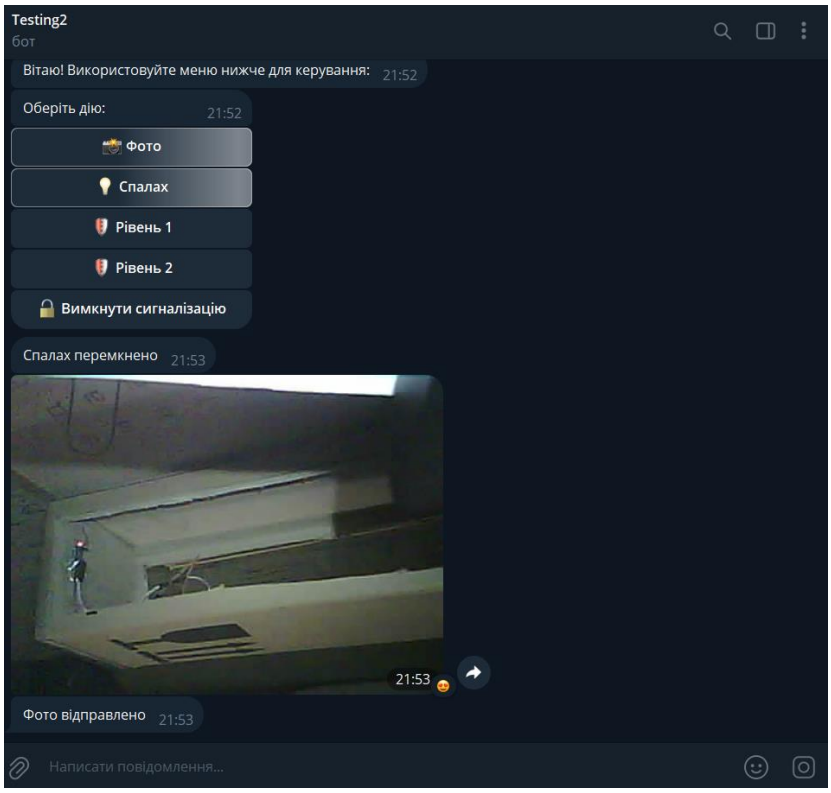

Рисунок 3.5 – Виконання команди photo

Система безпеки є зручним і ефективним рішенням для моніторингу приміщень. Вона дозволяє швидко налаштувати захист, отримувати сповіщення в реальному часі та керувати системою з будь-якої точки світу через соціальну мережу. Завдяки простоті використання та гнучкості налаштувань система підходить як для домашнього, так і для комерційного застосування.

ВИСНОВКИ

Розробка системи безпеки з лазерними сенсорами та інтеграцією з Telegram, представлена в цій дипломній роботі, стала відповіддю на потребу створення доступного, ефективного та зручного рішення для захисту приміщень. Метою роботи було створення апаратно-програмного комплексу, який здатен виявляти несанкціоноване вторгнення в контрольовану зону, сповіщати користувача через Telegram та забезпечувати віддалене керування системою. У процесі виконання роботи було вирішено низку завдань, включаючи проектування апаратної частини, розробку програмного забезпечення, інтеграцію з Telegram та тестування системи в лабораторних умовах. Отримані результати підтверджують успішне досягнення поставлених цілей і відкривають перспективи для практичного застосування та подальшого розвитку.

Одним із ключових результатів роботи стало створення апаратного комплексу на базі двох мікроконтролерів: ESP32, який відповідає за моніторинг лазерного сенсора та активацію звукового сигналу, та ESP32-CAM, оснащеного камерою для зйомки та комунікації через Telegram. Лазерний сенсор, що складається з діода та фототранзистора, виявився надійним інструментом для виявлення порушень, забезпечуючи високу точність у тестових умовах. Використання протоколу ESP-NOW для зв’язку між модулями дозволило досягти швидкості реакції системи на рівні близько 50 мілісекунд, що є значною перевагою порівняно з аналогічними системами, залежними від WiFi. Це забезпечило стабільну роботу навіть у разі тимчасової втрати мережевого з’єднання, що є важливим для реальних сценаріїв використання.

Результати тестування підтвердили надійність і практичність системи. У лабораторних умовах система стабільно виявляла переривання лазерного променя, коректно активувала звуковий сигнал у режимі підвищеного захисту та надсилала чіткі зображення через Telegram.

Система безпеки, розроблена в рамках цієї роботи, має широкий спектр потенційних застосувань. Вона може бути використана для захисту приватних будинків, офісів, складських приміщень або інших об’єктів, де потрібен оперативний моніторинг. Завдяки низькій вартості компонентів і простоті налаштування система є доступною альтернативою комерційним рішенням. Її інтеграція з Telegram робить керування зручним навіть для користувачів без технічної підготовки, оскільки не вимагає встановлення додаткового програмного забезпечення. Для комерційного використання систему можна адаптувати, додавши підтримку кількох зон моніторингу або інтеграцію з хмарними сервісами для зберігання даних.

Напрямки подальшого розвитку системи є перспективними. Інтеграція додаткових сенсорів, наприклад, руху чи температури, розширить функціонал, зробивши систему більш універсальною. Застосування алгоритмів машинного навчання для аналізу зображень могло б додати інтелектуальні можливості, такі як розпізнавання осіб або автоматичне визначення загроз. Ще одним перспективним напрямком є створення мобільного додатка, який би доповнив Telegram-бот, надаючи розширений інтерфейс і аналітику. Ці вдосконалення зроблять систему конкурентоспроможною на ринку безпеки.

Таким чином, дипломна робота досягла поставлених цілей, створивши функціональну систему безпеки, яка поєднує доступність, надійність і сучасний інтерфейс керування. Отримані результати підтверджують її практичну цінність і відкривають широкі можливості для подальшого вдосконалення та застосування в реальних умовах. Система є прикладом ефективного використання сучасних технологій для вирішення актуальних завдань безпеки, демонструючи потенціал мікроконтролерів і бездротових комунікацій у створенні інноваційних рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Intelligent smart home automation and security system using Arduino and Wi-Fi / J. Chandramohan, R. Nagarajan, K. Satheeshkumar, N.Ajithkumar, P.A. Gopinath, S. Ranjithkumar // International Journal of Engineering and Computer Science. – 2017. – Vol. 6, № 3. – PP. 20694–20698. [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/315648361_Intelligent_Smart_Home_Automation_and_Security_System_Using_Arduino_and_Wi-fi (дата звернення: 02.04.2025).
2. Соценко М.М., Шаріпова І.В. Система безпеки з лазерними сенсорами під час використання та формування зон селекції коефіцієнтів перетворення хаара, їх властивості для відновлення зображень // Комп'ютерні інтелектуальні системи та мережі. Матеріали XVIII Всеукраїнської науково практичної WEB конференції аспірантів, студентів та молодих вчених (25-27 березня 2025 р.). – Кривий Ріг: Криворізький національний університет, 2025. – с.283-286 [Електронний ресурс] – Режим доступу: https://sites.google.com/view/kicm/ (дата звернення: 02.04.2025).
3. Tech Trends: The Next Evolution of Motion Detection [Електронний ресурс] – Режим доступу: https://www.securityinfowatch.com/alarms-monitoring/alarm-systems-intrusion-detection/article/21219314/tech-trends-the-next-evolution-of-motion-detection?utm_source=chatgpt.com (дата звернення: 03.04.2025).
4. Telegram: ESP32 Motion Detection with Notifications (Arduino IDE) [Електронний ресурс] – Режим доступу: https://randomnerdtutorials.com/telegram-esp32-motion-detection-arduino/ (дата звернення: 03.04.2025).
5. Баловсяк Н. Популярність соцмереж і можливості для брендів. Як людство взаємодіє з цифровими технологіями — звіт Digital 2024. Медіамейкер. 26.03.2024. [Електронний ресурс] – Режим доступу:

https://mediamaker.me/yak-lyudstvo-vzayemodiye-z-czyfrovymy-tehnologiyamy-zvit-digital-2024-8566/ (дата звернення: 05.04.2025).

6. Датчики руху [Електронний ресурс] – Режим доступу: https://ua.stamping-welding.com/news/motion-sensors-70565502.html (дата звернення: 07.04.2025).
7. The best doorbell cameras [Електронний ресурс] – Режим доступу: https://www.theverge.com/22954554/best-video-doorbell-camera?utm_source=chatgpt.com (дата звернення: 10.04.2025).
8. Eslamdoost M. Arduino and Raspberry Pi in an Amazing Smart Home. Cademix Institute of Technology. 21.06.2021. [Електронний ресурс] – Режим доступу: https://www.cademix.org/arduino-and-raspberry-smart-home/ (дата звернення: 15.04.2025).
9. Соценко М.В., Камєнєва А.В., Комплексна система безпеки з лазерними сенсорами та telegram-інтеграцією // Інформатика, інформаційні системи та технології: тези доповідей XXII Всеукраїнської конференції студентів і молодих науковців. – Одеса, 25 квітня 2025 р. – Одеса, 2025. – С. 221–222 (дата звернення: 30.04.2025).
10. Laser Security Alarm Explained and Why You Need One [Електронний ресурс] – Режим доступу: https://www.iemrobotics.com/blogs/science-kits-blogs/laser-security-alarm-explained-and-why-you-need-one?utm_source=chatgpt.com (дата звернення: 02.05.2025).
11. Hatem H., Shehab J., Abdul-Rahman I. ARDUINO Microcontroller Based Building Security System // Engineering and Technology Journal. 2017. Т. 35 : Part A, No. 5. С. 532–536. URL: https://www.researchgate.net/publication/365422120_ARDUINO_Microcontroller_Based_Building_Security_System (дата звернення: 05.05.2025).
12. Arduino vs Raspberry Pi vs STM32: Choosing the Right Development Board for Your Projects [Електронний ресурс] – Режим доступу: https://www.richardelectronics.com/blog/projects/raspberry/arduino-vs-

raspberry-pi-vs-stm32-choosing-the-right-development-board-for-your-projects?utm_source=chatgpt.com (дата звернення: 09.05.2025).

13. Arduino vs Raspberry Pi: Choosing the Right Platform for Your Next Project [Електронний ресурс] – Режим доступу: https://www.wevolver.com/article/arduino-vs-raspberry-pi-choosing-the-right-platform-for-your-next-project?utm_source=chatgpt.com (дата звернення: 12.05.2025).
14. Telegram vs WhatsApp: Best App for Business in 2025 [Електронний ресурс] – Режим доступу: https://sendpulse.com/blog/telegram-whatsapp-comparison (дата звернення: 13.05.2025).
15. Home Security System using ESP32-CAM and Telegram Application / Adarsh S., Bhuvaneshwar A., Monisha M.S., Nisarga M., Mallamma C.G. // International Journal of Engineering and Computer Science. – 2023. – Vol. 10, № 05. – PP. 631–635. [Електронний ресурс] – Режим доступу: https://www.irjet.net/archives/V10/i5/IRJET-V10I599.pdf?utm_source=chatgpt.com (дата звернення: 13.05.2025).
16. What is PIR Sensor? 2024 Ultimate Guide on How it Works and Key Considerations for Optimal Performance / Roombanker [Електронний ресурс] – Режим доступу: https://www.roombanker.com/blog/what-is-pir-sensor/ (дата звернення: 15.05.2025).
17. Telegram Bot API [Електронний ресурс] – Режим доступу: https://core.telegram.org/bots/api (дата звернення: 16.05.2025).
18. ESP32 Technical Reference Manual [Електронний ресурс] – Режим доступу: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf (дата звернення: 20.05.2025).
19. How To Get Started With ESP-NOW [Електронний ресурс] – Режим доступу: https://www.digikey.com/en/maker/tutorials/2024/how-to-get-started-with-esp-now (дата звернення: 24.05.2025).

ДОДАТОК А

Програмний код ESP32-CAM

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <esp_now.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

const char* ssid = "Qwerty";
const char* password = "b2hn4z8e";
String BOTtoken = "7954160160:AAH-
4stXTUGsrj5ktYFrwfFv2PKjyQF4gFg";
String CHAT_ID = "1027658801";

WiFiClientSecure clientTCP;
UniversalTelegramBot bot(BOTtoken, clientTCP);

#define FLASH_LED_PIN 4
bool flashState = LOW;
bool sendPhoto = false;

int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

uint8_t espLaserAddress[] = {0x48, 0xE7, 0x29, 0x9F, 0xDC,
0x3C};
```

Лістинг А.1 – Програмний код ESP32-CAM

```
typedef struct {
  char command[20];
} command_t;

command_t msgToSend;
command_t receivedCmd;

unsigned long lastPhotoTime = 0;
const unsigned long photoInterval = 1000;

void ensureWiFiConnected() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi відключено, підключаємо...");
    WiFi.begin(ssid, password);
    int attempts = 0;
    while (WiFi.status() != WL_CONNECTED && attempts < 10) {
      delay(500);
      Serial.print(".");
      attempts++;
    }
    if (WiFi.status() == WL_CONNECTED) {
      Serial.println("\nWiFi підключено, IP: " +
WiFi.localIP().toString());
    } else {
      Serial.println("\nНе вдалося підключитися до WiFi");
    }
  }
}

void OnDataRecv(const esp_now_recv_info_t *recv_info, const
uint8_t *data, int len) {
  if (len != sizeof(command_t)) {
    Serial.println("Невірний розмір даних ESP-NOW");
    return;
  }
  memcpy(&receivedCmd, data, sizeof(receivedCmd));
  Serial.print("Отримано команду: ");
  Serial.println(receivedCmd.command);
  if (strcmp(receivedCmd.command, "photo") == 0) {
    unsigned long currentTime = millis();
    if (currentTime - lastPhotoTime >= photoInterval) {
      sendPhoto = true;
      lastPhotoTime = currentTime;
    } else {
      Serial.println("Занадто часта відправка фото,
пропускаємо");
    }
  }
}
```

Лістинг А.1, аркуш 2

```
     void OnDataSent(const uint8_t *mac_addr,
esp_now_send_status_t status) {
       Serial.print("Статус відправки: ");
       Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Успіх" :
"Помилка");
     }
     void configInitCamera() {
       camera_config_t config;
       config.ledc_channel = LEDC_CHANNEL_0;
       config.ledc_timer = LEDC_TIMER_0;
       config.pin_d0 = Y2_GPIO_NUM;
       config.pin_d1 = Y3_GPIO_NUM;
       config.pin_d2 = Y4_GPIO_NUM;
       config.pin_d3 = Y5_GPIO_NUM;
       config.pin_d4 = Y6_GPIO_NUM;
       config.pin_d5 = Y7_GPIO_NUM;
       config.pin_d6 = Y8_GPIO_NUM;
       config.pin_d7 = Y9_GPIO_NUM;
       config.pin_xclk = XCLK_GPIO_NUM;
       config.pin_pclk = PCLK_GPIO_NUM;
       config.pin_vsync = VSYNC_GPIO_NUM;
       config.pin_href = HREF_GPIO_NUM;
       config.pin_sscb_sda = SIOD_GPIO_NUM;
       config.pin_sscb_scl = SIOC_GPIO_NUM;
       config.pin_pwdn = PWDN_GPIO_NUM;
       config.pin_reset = RESET_GPIO_NUM;
       config.xclk_freq_hz = 20000000;
       config.pixel_format = PIXFORMAT_JPEG;
       config.grab_mode = CAMERA_GRAB_LATEST;

       if (psramFound()) {
         config.frame_size = FRAMESIZE_UXGA;
         config.jpeg_quality = 10;
         config.fb_count = 2;
       } else {
         config.frame_size = FRAMESIZE_SVGA;
         config.jpeg_quality = 12;
         config.fb_count = 1;
       }
       esp_err_t err = esp_camera_init(&config);
       if (err != ESP_OK) {
         Serial.printf("Помилка ініціалізації камери: 0x%x",
err);
         ESP.restart();
       }

       sensor_t *s = esp_camera_sensor_get();
       s->set_framesize(s, FRAMESIZE_CIF);
       Serial.println("Камера ініціалізована");
     }
```

Лістинг А.1, аркуш 3

```
     void sendMenu() {
       String keyboardJson = "[[{\"text\":\"
Фото\",\"callback_data\":\"/photo\"}],"
                             "[{\"text\":\"
Спалах\",\"callback_data\":\"/flash\"}],"
                             "[{\"text\":\" Рівень
1\",\"callback_data\":\"/level1\"}],"
                             "[{\"text\":\" Рівень
2\",\"callback_data\":\"/level2\"}],"
                             "[{\"text\":\" Вимкнути
сигналізацію\",\"callback_data\":\"/alarm_off\"}]]";
       bot.sendMessageWithInlineKeyboard(CHAT_ID, "Оберіть дію:",
"", keyboardJson);
     }
     void handleNewMessages(int numNewMessages) {
       for (int i = 0; i < numNewMessages; i++) {
         String chat_id = String(bot.messages[i].chat_id);
         if (chat_id != CHAT_ID) {
           bot.sendMessage(chat_id, "Невірний користувач", "");
           continue;
         }
         String text = bot.messages[i].text;
         if (bot.messages[i].type == "callback_query") {
           text = bot.messages[i].text;
         }
         Serial.print("Отримано команду Telegram: ");
         Serial.println(text);

         if (text == "/start") {
           bot.sendMessage(CHAT_ID, "Вітаю! Використовуйте меню
нижче для керування:", "");
           sendMenu();
         } else if (text == "/photo") {
           sendPhoto = true;
         } else if (text == "/flash") {
           flashState = !flashState;
           digitalWrite(FLASH_LED_PIN, flashState);
           bot.sendMessage(CHAT_ID, "Спалах перемкнено", "");
         } else if (text == "/level1") {
           strcpy(msgToSend.command, "level1");
           esp_now_send(espLaserAddress, (uint8_t*)&msgToSend,
sizeof(msgToSend));
           bot.sendMessage(CHAT_ID, "Встановлено рівень захисту 1
(лазер+камера)", "");
         } else if (text == "/level2") {
           strcpy(msgToSend.command, "level2");
           esp_now_send(espLaserAddress, (uint8_t*)&msgToSend,
sizeof(msgToSend));
           bot.sendMessage(CHAT_ID, "Встановлено рівень захисту 2
(лазер+камера+базер)", "");
```

Лістинг А.1, аркуш 4

```
    } else if (text == "/alarm_off") {
          strcpy(msgToSend.command, "alarm_off");
          esp_now_send(espLaserAddress, (uint8_t*)&msgToSend,
sizeof(msgToSend));
          bot.sendMessage(CHAT_ID, "Сигналізація вимкнена", "");
        } else {
          bot.sendMessage(CHAT_ID, "Невідома команда", "");
        }
      }
    }
    String sendPhotoTelegram() {
      Serial.println("Захоплення фото...");
      camera_fb_t *fb = esp_camera_fb_get();
      if (!fb) {
        Serial.println("Помилка захоплення зображення");
        bot.sendMessage(CHAT_ID, "Помилка захоплення
зображення", "");
        ESP.restart();
        return "Помилка";
      }
      Serial.println("Зображення захоплено, розмір: " +
String(fb->len));

      const char* myDomain = "api.telegram.org";
      if (!clientTCP.connect(myDomain, 443)) {
        Serial.println("Не вдалося підключитися до
api.telegram.org");
        bot.sendMessage(CHAT_ID, "Помилка підключення до
Telegram", "");
        esp_camera_fb_return(fb);
        return "Помилка підключення";
      }
      String head = "--RandomNerdTutorials\r\nContent-
Disposition: form-data; name=\"chat_id\"; \r\n\r\n" + CHAT_ID +
"\r\n--RandomNerdTutorials\r\nContent-Disposition: form-data;
name=\"photo\"; filename=\"esp32-cam.jpg\"\r\nContent-Type:
image/jpeg\r\n\r\n";
      String tail = "\r\n--RandomNerdTutorials--\r\n";
      uint32_t imageLen = fb->len;
      uint32_t extraLen = head.length() + tail.length();
      uint32_t totalLen = imageLen + extraLen;
      clientTCP.println("POST /bot" + BOTtoken + "/sendPhoto
HTTP/1.1");
      clientTCP.println("Host: " + String(myDomain));
      clientTCP.println("Content-Length: " + String(totalLen));
      clientTCP.println("Content-Type: multipart/form-data;
boundary=RandomNerdTutorials");
      clientTCP.println();
      clientTCP.print(head);
      uint8_t *fbBuf = fb->buf;
```

Лістинг А.1, аркуш 5

```
  size_t fbLen = fb->len;
  for (size_t n = 0; n < fbLen; n += 1024) {
    if (n + 1024 < fbLen) {
      clientTCP.write(fbBuf, 1024);
      fbBuf += 1024;
    } else {
      size_t remainder = fbLen % 1024;
      clientTCP.write(fbBuf, remainder);
    }
  }
  clientTCP.print(tail);
  esp_camera_fb_return(fb);

  String getBody = "";
  int waitTime = 10000;
  long startTimer = millis();
  while ((startTimer + waitTime) > millis()) {
    while (clientTCP.available()) {
      char c = clientTCP.read();
      getBody += String(c);
      startTimer = millis();
    }
    if (getBody.length() > 0) break;
  }
  clientTCP.stop();
  Serial.println("Відповідь від Telegram: " + getBody);
  if (getBody.indexOf("\"ok\":true") >= 0) {
    Serial.println("Фото успішно відправлено");
    bot.sendMessage(CHAT_ID, "Фото відправлено", "");
  } else {
    Serial.println("Помилка відправки фото");
    bot.sendMessage(CHAT_ID, "Помилка відправки фото", "");
  }
  return getBody;
}
void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  Serial.begin(115200);
  pinMode(FLASH_LED_PIN, OUTPUT);
  digitalWrite(FLASH_LED_PIN, flashState);
  configInitCamera();
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT);
  ensureWiFiConnected();
  if (esp_now_init() != ESP_OK) {
    Serial.println("Помилка ініціалізації ESP-NOW");
    return;
  }
```

Лістинг А.1, аркуш 6

```
    esp_now_register_recv_cb(OnDataRecv);
      esp_now_register_send_cb(OnDataSent);
      esp_now_peer_info_t peerInfo = {};
      memcpy(peerInfo.peer_addr, espLaserAddress, 6);
      peerInfo.channel = 0;
      peerInfo.encrypt = false;
      if (esp_now_add_peer(&peerInfo) != ESP_OK) {
        Serial.println("Не вдалося додати peer");
        return;
      }
      Serial.println("ESP32-CAM готовий!");
    }
    void loop() {
      ensureWiFiConnected();
      if (sendPhoto) {
        Serial.println("Відправляю фото...");
        sendPhotoTelegram();
        sendPhoto = false;
      }
      if (millis() > lastTimeBotRan + botRequestDelay) {
        int numNewMessages =
bot.getUpdates(bot.last_message_received + 1);
        while (numNewMessages) {
          Serial.println("Отримано відповідь від Telegram");
          handleNewMessages(numNewMessages);
          numNewMessages =
bot.getUpdates(bot.last_message_received + 1);
        }
        lastTimeBotRan = millis();
      }
    }
```

Лістинг А.1, аркуш 7

ДОДАТОК Б

Програмний код ESP32

```
#include <WiFi.h>
#include <esp_now.h>
const char* ssid = "Qwerty";
const char* password = "b2hn4z8e";
#define LASER_PIN 25
#define SENSOR_PIN 33
#define BUZZER_PIN 13
uint8_t espCamAddress[] = {0x34, 0x98, 0x7A, 0xB7, 0x1E,
0x70};
typedef struct {
  char command[20];
} command_t;
command_t msgToSend;
int protectionLevel = 0;
unsigned long lastPhotoTime = 0;
unsigned long lastBuzzerTime = 0;
const unsigned long photoInterval = 1000;
const unsigned long buzzerDuration = 5000;
void OnDataRecv(const esp_now_recv_info_t *recv_info, const
uint8_t *data, int len) {
  command_t receivedCmd;
  if (len != sizeof(command_t)) return;
  memcpy(&receivedCmd, data, sizeof(receivedCmd));
  Serial.print("Отримано команду: ");
  Serial.println(receivedCmd.command);
  if (strcmp(receivedCmd.command, "level1") == 0) {
    protectionLevel = 1;
    digitalWrite(LASER_PIN, HIGH);
    digitalWrite(BUZZER_PIN, LOW);
    Serial.println("Встановлено рівень захисту 1
(лазер+камера)");
  } else if (strcmp(receivedCmd.command, "level2") == 0) {
    protectionLevel = 2;
    digitalWrite(LASER_PIN, HIGH);
    digitalWrite(BUZZER_PIN, LOW);
    Serial.println("Встановлено рівень захисту 2
(лазер+камера+базер)");
  } else if (strcmp(receivedCmd.command, "alarm_off") == 0)
{
    protectionLevel = 0;
    digitalWrite(LASER_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
    Serial.println("Сигналізація вимкнена");
  }
}
```

Лістинг Б.1 – Програмний код ESP32

```
     void OnDataSent(const uint8_t *mac_addr,
esp_now_send_status_t status) {
       Serial.print("Статус відправки: ");
       Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Успіх" :
"Помилка");
     }
     void setup() {
       Serial.begin(115200);
       pinMode(LASER_PIN, OUTPUT);
       pinMode(SENSOR_PIN, INPUT_PULLUP);
       pinMode(BUZZER_PIN, OUTPUT);
       digitalWrite(LASER_PIN, LOW);
       digitalWrite(BUZZER_PIN, LOW);
       WiFi.mode(WIFI_STA);
       WiFi.begin(ssid, password);
       while (WiFi.status() != WL_CONNECTED) {
         delay(500);
         Serial.print(".");
       }
       Serial.println("\nWiFi підключено");
       if (esp_now_init() != ESP_OK) {
         Serial.println("Помилка ініціалізації ESP-NOW");
         return;
       }
       esp_now_register_recv_cb(OnDataRecv);
       esp_now_register_send_cb(OnDataSent);

       esp_now_peer_info_t peerInfo = {};
       memcpy(peerInfo.peer_addr, espCamAddress, 6);
       peerInfo.channel = 0;
       peerInfo.encrypt = false;
       if (esp_now_add_peer(&peerInfo) != ESP_OK) {
         Serial.println("Не вдалося додати peer");
         return;
       }

       Serial.println("ESP32 готовий!");
     }
     void loop() {
       if (protectionLevel > 0) {
         bool sensorState = digitalRead(SENSOR_PIN);
         unsigned long currentTime = millis();
             if (sensorState == HIGH) {
           if (currentTime - lastPhotoTime >= photoInterval) {
             Serial.println("Лазер не потрапляє на датчик,
відправляю команду на фото");
             strcpy(msgToSend.command, "photo");
             esp_now_send(espCamAddress, (uint8_t*)&msgToSend,
sizeof(msgToSend));
             lastPhotoTime = currentTime;
```

Лістинг Б.1, аркуш 2

```
      if (protectionLevel == 2) {
                digitalWrite(BUZZER_PIN, HIGH);
                lastBuzzerTime = currentTime;
              }
            }
          } else if (protectionLevel == 2 &&
digitalRead(BUZZER_PIN) == HIGH && currentTime - lastBuzzerTime
>= buzzerDuration) {
            digitalWrite(BUZZER_PIN, LOW);
          }
        }
        delay(100);
      }
```

Лістинг Б.1, аркуш 3