Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки автоматичного пошуку прихованих камер”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Кобізь А. Ю.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О. М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *бакалавр*
Галузь знань ._12 *“Інформаційні технології”*
Спеціальність *125 “Кібербезпека”*
Освітньо-професійна (освітньо-наукова) програма *“ Кібербезпека ”*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
“____” _____2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Кобізь Андрій Юрійович
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для автоматичного пошуку прихованих камер*
2. Керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “ 1 7 ” січня 2025 року №57-02
3. Строк подання роботи до захисту *22.05.2025 р.*
4. Мета та завдання випускної кваліфікаційної роботи *Метою роботи є розробка програмного забезпечення системи кібербезпеки автоматичного пошуку прихованих камер.*
5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. *Призначення та область використання.*
   2. *Перегляд аналогічних існуючих систем.*
   3. *Опис і обґрунтування проектних рішень.*
   4. *Етапи програмування системи.*
   5. *Впровадження системи в промислову експлуатацію*
   6. *Висновки.*
6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Принципова схема	*1 аркуш*
Блок-схема алгоритму роботи додатку	*3 аркуша*

7. Дата видачі завдання « 17 » січня 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем керування	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.05.2025 р.	
8.	Попередній захист роботи	22.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Кобізь А.Ю. Програмне забезпечення системи кібербезпеки автоматичного пошуку прихованих камер. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки автоматичного пошуку прихованих камер.

Метою розробки є програмне забезпечення системи кібербезпеки з автоматичним скануванням приміщення для виявленням прихованих камер за допомогою мікроконтролера ESP32-CAM.

Результат роботи – програмна реалізація системи кібербезпеки автоматичного пошуку прихованих камер.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 7/8/10/11, MacOS, Linux а також мобільних пристроях з ОС Android/iOs.

Програму розроблено в середовищі Arduino IDE.

Ключові слова: кібербезпека, приховані камери, ESP32-CAM, відблиск, автоматизоване сканування.

ABSTRACT

Kobiz A.Y. Software of the cybersecurity system for automatic search of hidden cameras. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is designed for a cybersecurity system for automatic search for hidden cameras.

The purpose of the development is to develop software for a cybersecurity system with automatic room scanning to detect hidden cameras using the ESP32-CAM microcontroller.

The result of the work is a software implementation of a cybersecurity system for automatic search for hidden cameras.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with the software are given.

The program can be used on PCs with Windows 7/8/10/11, MacOS, Linux and mobile devices with Android/iOs.

The program is developed in the Arduino IDE environment.

Keywords: cybersecurity, hidden cameras, ESP32-CAM, glare, automated scanning.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП..... 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи..... 5
1.2 Область застосування..... 6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти..... 9
2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування..... 16
2.3 Розгорнута постановка завдання 21
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 23
3.1 Опис функціонування системи 23
3.2 Розробка структурної схеми..... 27
3.3 Розробка функціональної схеми 30
3.4 Розробка діаграми процесів..... 33
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 36
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 36
4.2 Захист розробленого програмного забезпечення..... 44
5 ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 47
6 ОСНОВНІ ВИСНОВКИ..... 50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 52

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІЧ – інфрачервоне випромінювання

ПЗ – програмне забезпечення

AP – точка доступу

IP – мережева адреса

SPIFFS – файлова система флеш-пам’яті

RGB – червоний, зелений, синій

ВО – виділена зона

RF – радіочастота

JPEG – формат компресованих фотографій

ESP32 – Esspresif System’s on a Chip 32-bit – модель плати

Wi-Fi – безпровідна мережа

HTTP – протокол передачі гіпертексту

IDE – інтегроване середовище розробки

IoT – інтернет речей

ВСТУП

Актуальність теми. Сучасний світ все більше залежить від технологій, які гарантують безпеку та конфіденційність матеріальних ресурсів, приватного життя та інформації. З появою компактних пристроїв для запису аудіо та відео виникає зростаючий великий ризик несанкціонованого стеження за допомогою прихованих камер[1]. Такі камери можуть працювати автономно тривалий час, маскуючись під звичайні предмети та активізуватись тільки при виявленні руху, що робить їх досить економними в споживанні енергії[15]. Ці пристрої можуть бути застосовані для промислового шпигунства, втручання в приватне життя людей без їхнього знання, збору конфіденційних даних та шантажу, а, також, можуть виступати як частина диверсійної діяльності[2].

Таким чином, створення дешевих, портативних і автономних пристроїв для виявлення прихованих камер є важливим завданням[3][4]. IoT-технології можуть ефективно слугувати для моніторингу критичної інфраструктури, та інших сфер, забезпечуючи контроль як кібербезпеки, так і фізичних параметрів[20]. Це підвищує рівень захищенності від витоків технічними каналами, як в бізнесі, об'єктах критичної інфраструктури, державних закладах, так і в побуті.[19]

Мета та завдання дослідження. Метою роботи є розробка програмного забезпечення системи кібербезпеки, яка буде виявляти приховані відеокамери. Система базується на платі ESP32-CAM з двома кроковими двигунами для повного аналізу приміщення.

Для досягнення поставленої мети визначено наступні завдання:

− Огляд існуючих технічних рішень для виявлення прихованих камер.

− Розробка алгоритму, що забезпечує сканування приміщення за допомогою рухомої камери з обробкою зображення, щоб виявляти світлові аномалії, які властиві для оптики камер[5].

− Програмна реалізація інтерфейсу користувача системи кібербезпеки, що

надасть змогу керувати системою та переглядати результати аналізу.

Практичне значення отриманих результатів. Висновки даного дослідження можуть бути використані для розробки портативних пристроїв виявлення прихованих камер у офісах, конференц-залах, об’єктах з підвищеними вимогами до рівня безпеки або побутових умовах.

Найголовнішою особливістю є те, що дана система характеризується економічною доступністю та простотою користування, що підійде для користувача без спеціальних знань.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки, яка виявлятиме приховані камери є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

З постійним розвитком сучасні системи моніторингу з використанням IoT все активніше запроваджуються у сферах безпеки [17]. Таким чином, розроблюваний пристрій та всю систему слід розглядати як приклад IoT-сенсорного вузла, адаптованого під завдання безпекового моніторингу, а саме – виявлення несанкціонованих засобів відеоспостереження.

Система виявлення прихованих камер на основі плати ESP32-CAM є проактивною системою для сканування приміщень. Функція приладу - виявлення оптичних елементів камер, які можуть бути прихованими в різних предметах інтер'єру в полях зору пристрою.

Технологія виявлення працює шляхом розсіювання фізичних світлових відблисків від поверхні оптичного елементу прихованої камери під дією щільно сконцентрованого світлового променя.[6][7]

Складові компоненти системи

Повноцінна система складається з наступних складових: плати ESP32-CAM, що включає в собі камеру і вбудований спалах та два покрокових двигуни з автоматичним позиціонуванням. Для забезпечення повного покриття оберту камери в кімнатному просторі один двигун відповідає за горизонтальний оберт, а інший – за вертикальне переміщення. Це надає змогу проводити повноцінний аналіз приміщення без ручного керування.

Червоне освітлення є основним компонентом системи, яке дозволяє визначати звичайні відображення на поверхнях з зеркальним покриттям, що вказує на положення оптичного компоненту. [9][11][12]

Як працює пристрій виявлення прихованих камер за допомогою відблисків від оптичних елементів?

Після кожного повороту двигунів камера робить знімок приміщення, який зберігається в буфері. Далі алгоритм обробляє зображення, виявляючи нестандартні пікселі в червоному каналі. У випадку, якщо такі пікселі будуть виявленими, система буде позначати Bouding Box (BO) – виділена область підозрілих пікселів на фото для подальшого збереження фотографії у Serial Peripheral Interface Flash File System (SPIFFS) – файлова система флеш-пам'яті.

Ця система працює автономно, відрізняючись від пасивних методів, які вимагають ручного наведення детектора або постійного перегляду відеопотоку та надає можливість прочесувати час від часу певний регіон у визначених межах, сповіщаючи лише про потенційно загрозливі місця. Це прямо впливає на зниження навантаження на користувача і неефективності виявлення прихованих камер.

1.2 Область застосування

Сфери застосування автоматизованого пристрою для виявлення прихованих камер

Коли існує високий ризик нелегального відеоспостереження або коли важлива конфіденційність дана автоматична система кібербезпеки для виявлення прихованих камер на базі ESP32-CAM має першорядне значення. Чому даний комплекс може використовуватись в абсолютній більшості сфер бізнесу, як приватного, так і корпоративного? Тому, що має такі переваги, як:

– Компактність;

– Автономність;

– Простота використання;

– Можливість керувати з будь якого пристрою(ПК, смартфони з iOS/Android).

Готельно-ресторанний бізнес. Одним із основних варіантів використання є перевірка наявності прихованих камер у приміщеннях готелів, хостелів і

орендованих квартих. Рішення про використання пристрою для сканування кімнат, будь то орендованого чи власного, дозволяє зменшити особистий ризик і уникнути компрометуючих ситуацій, оскільки гості даних закладів можуть стикатись з порушенням приватності.

Кімнати для ділових зустрічей. У бізнес-середовищі система може використовуватися для перевірки офісів, конференц-залів і переговорних кімнат перед важливими зустрічами. Регулярна перевірка приміщень є цілком законною практикою безпеки в сучасному світі, коли витік чутливих так конфіденційних даних може призвести до значних фінансових або репутаційних збитків.

Державні установи та військові об'єкти. У спеціалізоних установах, таких як, наприклад, архіви, наукові лабораторії, військомати, судові або поліцейські відділи та інші об'єкти з високим рівнем безпеки, зазвичай встановлюють правила щодо боротьби з відсутністю шпигунських пристроїв та шпигунського програмного забезпечення. Дана система на базі ESP32-CAM цілком може стати невід'ємною частиною практик безпеки, адже вона дозволяє співробітникам швидко та безперешкодно оглянути приміщення.

Освітні та демонстраційні програми. Одним із варіантів використання пристрою є використання в якості навчального стенду для студентів, які навчаються на спеціальностях кібербезпеки, електроніки, робототехніки або автоматизації. Система повністю відкрита для всебічного розширення та адаптування під різні потреби, а також експерементів з алгоритмами обробки зображень, керування кроковими двигунами та зв'язку з вбудованими веб-серверами завдяки відкритій архітектурі Arduino.

Особисте використання. Крім того, пристрій може бути корисним і для тих, хто стурбований своєю безпекою вдома, наприклад, у кімнатах для роботи, в спальні у дітей, гаражах та інших приміщеннях, де є підозра на можливе встановлення прихованих камер сторонніми особами. Система доступна для широкого кола людей, які не мають технічних знань та навичок, оскільки вона проста в експлуатації та має спрощений користувацький інтерфейс.

Таким чином, виходячи з вищеперерахованого та того, що пристрій та програмне забезпечення системи кібербезпеки з виявлення прихованих камер є універсальним та гнучким, що дозволяє її використовувати для різних цілей, робить цю задачу актуальною, що потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Hidden Camera Detector – мобільний додаток (див. рис. 1.1), який використовує сканування магнітної активності та інфрачервного (ІЧ) випромінювання, щоб визначати потенційні шпигунські камери. Даний застосунок дозволяє проскановувати електроніку та отримувати покази, схожі на показання камери, підносячи телефон рукою до підозрілих об’єктів.

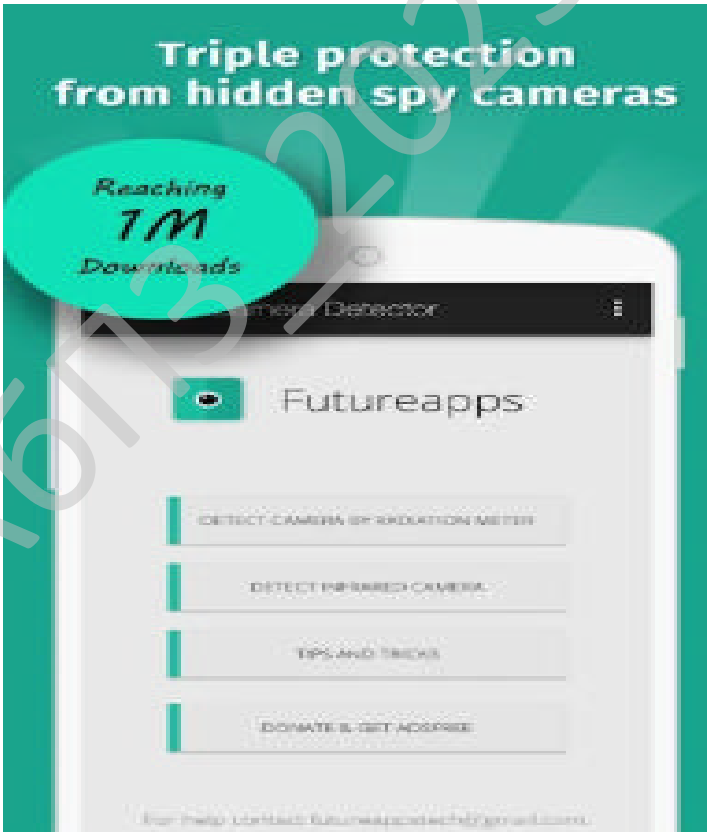

Рисунок 1.1 – Інтерфейс головного меню додатку Hidden Camera Detector [35]

Основний функціонал, який надає даний застосунок:

– Детектор магнітного поля: дозволяє аналізувати активність навколо смартфону та видає звуковий сигнал при виявленні підозрілої активності;

– Інфрачервоний детектор: виявляє інфрачервоне випромінювання, невидиме для людського ока;

– Поради та допомога при ручному виявленні прихованого об’єктива;

– Можливість ділитись місцезнаходженням виявленого потенційної камери з іншими користувачами застосунку.

Перевагами використання даного застосунку є простота його використання. Інтерфейс – інтуїтивно зрозумілий та надається покрокова інструкція користувача з додатковими порадами та відповідями на поширені запитання, що мають на меті допомогти уникнути хибних спрацювань. Також до плюсів відноситься мобільність даного рішення, так як це мобільний застосунок у смартфоні. Варто відмітити також подвійний режим сканування – магнітний аналіз та ІЧ-детекція (див рис. 1.2).

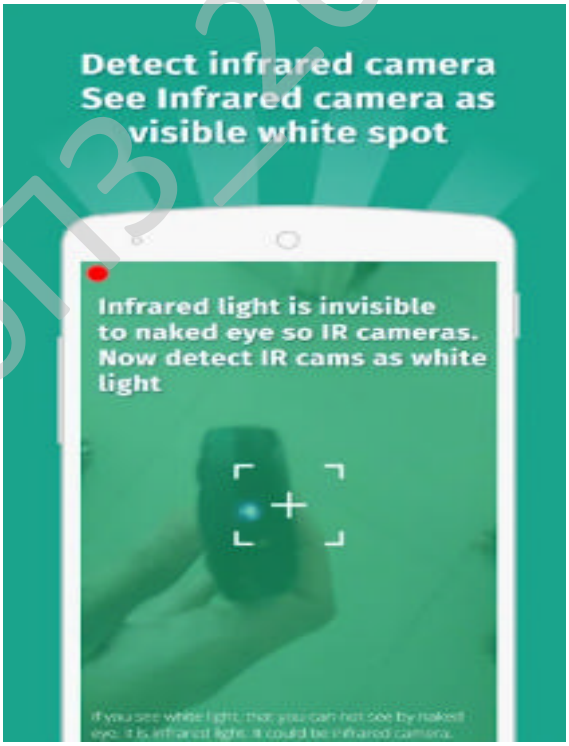

Рисунок 1.2 – Момент виявлення прихованої камери [35]

Недоліки та чому даний застосунок в абсолютній більшості випадків буде видавати похибку або взагалі не виявлятиме прихованих камер. Першою та

найголовнішою причиною є залежність від апаратного забезпечення, а саме відсутність магнітного датчику на смартфоні. Це вже унеможливлює проведення сканування у магнітному аналізі. Для коректної роботи ІЧ-детектора треба закрити всі інші застосунки що використовують камери, інакше додаток виявлення прихованих камер не працюватиме. Причиною похибок також можуть бути металеві предмети та електроніка, що викликають помилкові сигнали та недосконалі алгоритми обробки зображень. Користувачеві необхідно у ручному режимі проводити сканування та перевіряти наявність об’єктиву, тому що додаток ніяк не повідомляє про його наявність в разі сканування за допомогою ІЧ-детектора. І не варто забувати про технічну нестабільність даного застосунку. При використанні додаток може аварійно завершити роботи.

Ефективність даного програмного рішення знаходиться під великими сумнівами у зв’язку з наведеними недоліками та дуже залежить від апаратної частини смартфону та певною мірою від користувацького досвіду.

ІЧ-детектор камер Baseus Heyo Series II – невеликий портативний пристрій, який вможе виявляти приховані камери в різних місцях(див. рис. 1.3). Має два режими виявлення – візуальний та інфрачервоний. Пристрій може виявляти потенційні шпигунські пристрої незалежно від того, приховані чи використовуються

Рисунок 1.3 – Зовнішний вигляд ІЧ-детектора [36]

Основні функції пристрою включають:

– Виявлення інфрачервного випромінювання: дозволяє виявляти інфрачервоне світло від об’єктивів камер, яке зазвичай невидиме для людського ока. Для даного виявлення використовується ІЧ-режим зі світлодіодним підсвічуванням;

– Візуальне сканування: характерні відблиски від об’єктивів можна виявляти завдяки спеціальному окуляру в поєднанні з світлодіодним підсвічуванням;

– Ручне сканування: надає користувачеві можливість оглядати приміщення самостійно, через потенційно підозрілі точки через фільтр пристрою;

– Портативність: пристрій легко можна брати з собою будь куди;

– Режим ліхтарику: можливість використати світлодіодний ліхтар як звичайний ліхтар.

Перевагами даного пристрою є компактність та незалежність від

смартфонів, сенсорів та інших додатків, швидкість та ефективність завжди однакова. Його дуже просто використовувати, для цього треба натиснути всього одну кнопку ввімкнення та включити підсвічування. Для проведення сканування необхідно дивитись у спеціальний фільтр.

Недоліками компактного пристрою є не повністю автоматизоване виявлення, тому користувач повинен постійно фізично оглядати територію за допомоогою пристрою, а це є питанням безпеки та певних знань. Також для більшої ефективності виявлення необхідна повна темрява або слабке освітлення, особливо в інфрачервоному режимі. Варто зазначити, що виявляються лише об'єктиви, які здатні відбивати світло або ІЧ-випромінювання. Камери, що є добре замаскованими або пасивно зняті можуть бути невидимими для пристрою. Також слабкою стороною є невеликий радіус дії. Щоб отримати якісне та ефективне сканування необхідно бути якомога ближче до прихованої камери, що може бути дійсно проблемою.

Оскільки пристрій базується на спеціалізованій оптиці замість параметрів смартфона, його ефективність є значно вищою за мобільний додаток. Однак, ця ефективність напряму залежить від погодних умов, світлового освітлення та якості ручного сканування.

K-18 RF Detector – професійний багатофункціональний пристрій для виявлення прихованих камер, жучків, GPS-трекерів, бездротових передавачів і інших пристроїв моніторингу (див. рис. 1.4).

Рисунок 1.4 – Зовнішній вигляд K-18 RF Detector [37]

Пристрій використовує різні технології, включаючи радіочастотне тестування, виявлення лінз, підсвічування та магнітне виявлення. Він може допомогти знайти як і пасивне, так і активне моніторингове обладнення.

Основні функції пристрою:

– RF-детектор (радіочастотний детектор): виявляє активні пристрої, що передають сигнали в діапазонах від 1 МГц до 8000 МГц. Прикладом є Wi-Fi камери, приховані жучки;

– Оптичний детектор об’єктива: за допомогою червоного LED-підсвічування можливо виявляти відблиск від оптичних елементів прихованої камери;

– Магнітний датчик: виявляє магнітні поля прихованих пристроїв;

– Регулювання чутливості: надає змогу налаштовувати рівень сигналу, щоб зменшити можливість помилкової тривоги;

– Декілька варіантів сповіщення про отриманий сигнал: можливість користувачу вибрати, яким чином пристрій повідомлятиме про виявлені підозрілі сигнали (вібрація або звуковий режим).

Перевагами використання даного пристрою є професійна точність. Він дозволяє виявляти активні та пасивні пристрої на достатньо великих відстанях та за перешкодами (стіни, меблі та інші об'єкти). Багатофункціональність дозволяє використовувати три різні методи виявлення прихованих пристроїв: радіочастотний, оптичний і магнітний. Пристрій є компактним та мобільним, працює від батарейок. Підходить як для повсякденного використання, так і для бізнесменів, журналістів, співробітників служб безпеки та фахівців з антишпигунства.

Недоліками та обмеженнями гаджета є те, що для використання потрібні певні знання та навички налаштування та користування. Адже неправильно відкалібрована чутливість може викликати хибні тривоги з побутовою технікою. Також він не може виявляти повністю пасивні пристрої, які не мають лінзи або взагалі не випромінюють сигналів, тому необхідно проводити ручну перевірку світлодіодного фільтра. Хоча дане рішення є багатофункціональним, в цей час воно є дорогим для звичайного користувача.

Завдяки багатофункціональності та стабільній апаратній реалізації цей пристрій працює набагато ефективніше, ніж його споживчий еквівалент.

На основі вище проаналізованих програмних рішень та пристроїв варто створити порівняльну таблицю на основі автоматизація процесів виявлення прихованих пристроїв (табл. 1.1)

Таблиця 1.1 – порівняння пристроїв на основі автоматизації процесів виявлення прихованих пристроїв

Назва пристрою	Категорія пристрою	Автоматизація

Продовження таблиці 1.1

Hidden Camera Detector(моб. застосунок)	Програмне рішення	Відсутня
Baseus Heyo Series II	Побутовий пристрій	Відсутня
K-18 RF Detector	Професійний пристрій	Відсутня

Огляд доступних систем виявлення прихованих камер показав, що, хоча існує багато різних методів і пристроїв, жоден з проаналізованих не надає можливість автоматизації в пошуку шпигунських камер. Це підкреслює необхідність розробки нової системи, яка враховує переваги та недоліки існуючих технологій, щоб забезпечити повну автоматизацію процесу виявлення прихованих камер.

Загалом з існуючих типів детекторів виявлення прихованих камер можна виділити наступні:

- Радіочастотні детектори;
- Інфрачервоні детектори;
- Комбіновані детектори;
- Детектори професійного рівня. [13]

Існуюча більшість пристрої використовує ідею фізичного впливу на середовище для виявлення цифрових ознак наявності прихованої камери. Такий підхід є досить ефективним за різних умов використання. Він дозволяє виявляти шпигунські камери не дивлячись на те, чи вони провідні чи безпровідні, чи мають шифрування чи ні.[10][14]

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Для розробки системи кіберзахисту для виявлення прихованих камер було вирішено використати апаратну платформу ESP32-CAM і середовища розробки Arduino IDE. Технічні характеристики пристрою, економічна доцільність, широка підтримка спільноти, можливості та простота інтеграції з використанням клієнт-серверної архітектури були факторами, які визначали цей вибір.[24][25][26][29]

Arduino IDE 2.3.6

Arduino IDE 2.0 – нове крос-платформне інтегроване середовище розробки, яке дозволяє створювати, компілювати, завантажувати та налагоджувати проєкти для плат Arduino, ESP32 та інших сумісних плат. Заснована з нуля, вона використовує вбудований редактор Monaco, який використовується у Visual Studio Code. Даний редактор має досить простий та зручний інтерфейс, високу продуктивність, підсвічування синтаксису, швидку навігацію між частинами блоків коду та автозавершення частин коду.

Основні можливості Arduino IDE 2.3.6:

– Підтримка різноманітних вкладок і відкритих проєктів.

– Повністю оновлений менеджер плат, бібліотек і розширень, створений сторонніми розробниками.

– Підсвічування помилок у коді та система компіляції та завантаження журналів.

– Покращено серійний монітор, а також додано новий серійний графічний редактор, що надає змогу створювати графіки у реальному часі.

– Покрокове виконання, точки зупинки, та попередній перегляд змінних для плати які підтримують Arduino Zero, MKR, ESP32.

– Arduino Core API надає змогу підтримувати понад тисячу різних моделей плат включаючи класичні(UNO, Nano, Mega), MKR, Portenta, ESP32/ESP8266, STM32.

– Підтримка крос-платформ(Windows, macOS, Linux x64/ARM64).

– Легка інтеграція з Arduino Cloud/IoT.

– Підтримка завантаження прошивки через інтернет через повітря.

– ПЗ з відкритим вихідним кодом, яке має велику спільноту та постійні регулярні оновлення, що доступні через офіційні репозитарії GitHub.

– Більше стабільності та продуктивності.

– Масштабованість, що надає велику підтримку як для початківців так і для досвідчених розробників, надаючи інтуїтивно зрозумілий інтерфейс для простого програмування та професійні функції для досвідчених користувачів.

Плата ESP32-CAM випущена у 2019 році. Недорога та потужна, з інтегрованими модулями WiFi та Bluetooth та вбудованою камерою OV2640 в поєднанні з можливостями мікроконтролера ESP32. Мікроконтролер є досить таки популярним рішенням та використовується у різних цілях.

В дослідженні [23] вона використовується як автоматизована система поливу. Сам контролер же забезпечував безпровідний моніторинг і керування через Wi-Fi з можливістю віддаленого доступу через мобільний додаток Blynk. Порівнянно з комерційними аналогами пристрій має низьку собівартість, енергоефективність та розширюваність.

З дослідження [31] стає ясно, що контролер та розроблені на ній системи мають надійну роботу сенсорів та швидкий час відповіді системи, підтверджуючи ефективність рішення для проведення скануючих, моніторингових рішень.

Найбільший аргумент для використання у розробці системи для вибору даного контролера є те, що він може бути використаний як модуль реального візуального моніторингу [28]. Розроблені системи цілком демонструють стабільну роботу, і можуть працювати повноцінно без потреби складного налаштування з боку кінцевого користувача [34]. Популярність даної моделі пов’язана у зв’язку з компактними розмірами, низькій ціні та широкій функціональності[30]. Її використовують у різних сферах, починаючи з реалізації проєктів із комп’ютерним зором, розпізнаванням облич, віддаленим відеоспостереженням та закінчуючи повноцінними автоматизованими IoT-системами.[16][17][28][38][39]

Основні можливості моделі ESP32-CAM:

– Двоядерний мікроконтролер ESP32-S (Xtensa LX6 до 240 МГц), із

вбудованими модулями Wi-Fi 802.11 b/g/n, Bluetooth 4.2 (BLE).

– Інтегрована камера OV2640 з 2 мп та роздільною здатністю до 1600x1200, підтримка форматів JPEG, RGB565.

– Підтримка карти пам’яті microSD.

– Має багатофункціональних 14 пінів GPIO(UART, SPI, I2C, PWM, ADC, DAC, LEDC).

– Вбудований слот для камери з підтримкою інтерфейсу SCCB.

– Програмується через UART (TX/RX) за допомогою зовнішнього USB-UART адаптера.

– Низьке енергоспоживання в режимі сну.

– Широкий діапазон живлення: від 3.3 В до 5 В.

– Підтримка Arduino IDE, ESP-IDF, PlatformIO, MicroPython.

Підсумовуючи особливості плати ESP32-CAM можна сказати що даний модуль ідеально підійде для системи з виявлення прихованих камер у зв’язку з компактністю, наявністю камери та гнучкістю програмування. Цей модуль дозволяє створити автономну систему, яка може виконувати багато завдань без підключення до зовнішнього комп’ютера або хмарних обчислень. Пристрій може автоматично фотографувати навколишнє середовище, а вбудований модуль Wi-Fi дозволяє користувачеві передавати фотографії через веб-інтерфейс. Веб-браузер користувача виступає як клієнт, а веб-сервер ініціалізується безпосередньо на самій платі ESP32-CAM і виконує функцію сервера. Таким чином, повний дизайн клієнт-сервера реалізується, включаючи обмін даними, управління пристроями та розподілом результатів обробки в режимі реального часу.

Мікроконтролер достатньо ефективний, щоб виконувати локальну обробку зображення, наприклад перевіряти пікселі на наявність яскравих відблисків, які можуть вказувати на наявність прихованої камери.

Окрім збору та передачі даних, даний модуль дозволяє керувати зовнішніми пристроями. Два крокових двигуни, інтегрованих в проєкт, зможуть приводитись в рух по команді користувача використовуючи веб-інтерфейс.

Мова C/C++ була обрана в якості основної мови програмування через високу продуктивність і здатності тонкого налаштування ресурсів мікроконтролера. З обмеженнями оперативної пам’яті та обчислювальних ресурсів плати ESP32 це особливо важливо.

Для реалізації веб-інтерфейсу використовується HTML з CSS та JavaScript.

Для плат ESP32 Espressif існує спеціальний Arduino Core, який дозволяє використовувати всі стандартні функції Arduino разом із розширеними можливостями ESP32.

Основні особливості Arduino Core:

– Підтримка плат ESP32 для роботи з Wi-Fi, Bluetooth, датчиками, пінами, камерами, SPIFFS тощо.

– Код пишеться на C/C++, і є спрощеним завдяки готовим бібліотекам та функціям setup() та loop().

– Підтримує прості скетчі та повноцінні проєкти.

– Підтрмує серійний дебагінг, моніторинг порту та просту інтеграцію із зовнішніми модулями.

Основні програмні компоненти системи

ESP32-CAM є основною частиною системи, яка виконує функції зйомки, аналізу, зберігання даних, контролювання крокових двигунів та обробки веб-сервера.

Крокові двигуни забезпечують точне переміщення камери для просторового сканування вздовж вертикальних та горизонтальних осей.

Вбудований світлодіод працює як підсвічування для виявлення відблисків.

Камера OV2640 надає змогу створювати зображення розміром 320x240 пікселів, чого має бути достатньо для аналізу фотографій.

Зображення та проміжні результати зберігатимуться без використання зовнішньої пам’яті за допомогою систем SPIFFS.

Локальний веб-сервер дозволятиме клієнтам отримувати доступ до результатів та керувати пристроєм за допомогою браузера.

Клієнт-серверна архітектура дозволить розділити відповідальність між клієнтом і пристроєм. Це зробить його простим у керуванні, матиме можливість масштабування.

Засоби та ресурси, обрані для створення системи дозволять створити ефективну, невелику та не дорогу систему з високою автономністю. Вони пропонують гнучкість налаштування, чудову можливість для модернізації та застосування в широкому діапазоні контекстів.

Таким чином, за допомогою використання ESP32-CAM в Arduino IDE та мові програмування C/C++ в клієнт-серверній архітектурі можна створити досить ефективну, доступну та просту сучасну систему автоматичного виявлення прихованих камер.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмно-апаратний комплекс, який призначено для системи кібербезпеки виявлення прихованих камер за допомогою активного сканування приміщення та аналізу отриманих зображень у червоному спектрі з використанням ESP32-CAM, крокових двигунів та алгоритмів обробки зображень.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів як комерційних, так і відкритих проєктів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальшій побудові програмного забезпечення;

б) обґрунтувати вибір методики побудови системи кібербезпеки виявлення прихованих камер. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, алгоритми для сканування та обробки фотографій, що дозволить реалізувати задачу поставлену технічним

завданням. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) реалізувати веб-інтерфейс користувача, з метою керування пристроєм та отримуванням інформації про координати можливих підозрілих ділянок;

д) розробити рекомендації впровадження та подальшої експлуатації системи в побутових або промислових умовах, враховуючи конструктивні обмеження щодо руху механізмів та особливостей монтажу;

е) провести розрахунки по визначенню доцільності впровадження та ефективності розробленої системи;

ж) розробити заходи з охорони праці та цивільного захисту, що стосуються використання пристрою та монтажу в побутовому середовищі;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Ціль системи полягає в тому, щоб автоматично просканувати ціле приміщення кімнати без втручання людини в цей процес, щоб знайти потенційні приховані камери. Вони фіксуються у вигляді світлових відблисків у певному діапазоні освітлення.

Для досягнення цієї мети в системі використовується поєднання плати ESP32-CAM, крокових двигунів, камери, вбудованого спалаху та програмного забезпечення, яке виконує алгоритм сканування, зберігання, аналіз та подальший вивід обробленого зображення в веб-інтерфейс.

Система працює в автоматичному режимі, яка включає в собі початкову ініціалізацію пристрою та всіх компонентів, їх конфігурацію, покрокове позиціонування двигунів, фотографування у форматі RGB565, подальшу обробку фотографії з ідентифікацією підозрілих об’єктів, якщо такі були виявлені.

Доступ до результатів виконання програмного забезпечення пристрою системи кібербезпеки можна отримати через веб-інтерфейс, реалізований у вигляді веб-сторінки, яка зберігатиметься на веб-сервері.

Ініціалізація системи

Налаштування основних об’єктів відбувається після підключення живлення до плати ESP32-CAM та під час її завантаження.

На початковому етапі ініціалізації визначаються:

– Піни, які будуть прив’язані до двигунів.

– Ініціалізація веб-серверу.

– Конфігурація IP-адреси мережі та пристрою.

– Оголошення констант у вигляді кроків для ходу двигунів та інших змінних, які призначені для синхронізації процесів руху двигунів з процесами

фотографування та обробкою фото.

Після початкової ініціалізації системи відбувається етап повноцінної ініцалізації всіх використаних в системі компонентів з подальшою конфігурацією в системній функції setup – що викликається один раз, під час кожного запуску пристрою. Для зручного дебагінгу коду ініціалізується серійний монітор, який дозволяє в режимі реального часу моніторити діяльність всіх алгоритмів та процесів.

Ініціалізація з конфігурацією, що відбувається під час запуску системи, включає в собі:

– Запуск серійного монітору.
– Налаштування швидкостей двигунів.
– Ініціалізація SPIFFS.
– Ініціалізація та конфігурація AP (точки доступу).
– Очікування на приєднання клієнта до AP.
– Конфігурація шляхів для веб-серверу.
– Ініціалізацію камери.

Пристрій не використовується в загальній мережі, але все ж таки важливо врахувати можливі мережеві загрози, характерні для IoT-платформ. Згідно із дослідженнями IoT небезпек [8], навіть прості Telnet-доступи з типовими паролями можуть стати критичною вразливістю для мережевих атак. Враховуючи це, у системі передбачено базовий захист – доступ до веб-інтерфейсу здійснюється через Wi-Fi мережу, що унеможливлює підключення сторонніх користувачів без автентифікації. Це надає можливість завжди підтримувати актуальність паролю.

Алгоритм сканування простору

Після успішної ініціалізації пристрою та підключення користувача по мережі Wi-Fi, він переходить в режим очікування. Через веб-сторінку користувач має змогу запустити процес сканування. Внаслідок чого, двигуни починають працювати щоби виставити камеру в початкове положення для початку

сканування. У зв’язку з обмеженням куту оберту горизонтального та вертикального двигунів, камері доводиться розгортатись в певному діапазоні, щоб уникнути перегину дротів або механічного натягу. Це і є причиною, чому двигунам треба виставити камеру в початкове положення.

Основна логіка сканування виконується за допомогою вкладених ітерацій горизонтального та вертикального переміщення.

Досягнувши свого початкового положення, камера починає рухатись вниз по вертикалі з кроком в 15 градусів і зупиняється на кожному рівні щоб зробити фотографію. Даний алгоритм дозволяє повністю обійти кімнату у вертикальній осі – від підлоги до стелі.

Після кожної зупинки відбувається процес захвату зображення камери у форматі RGB565, що дозволяє запускати алгоритм обробки фото та виявлення підозрілих пікселів (якщо такі виявлені).

В моменті, коли повторення вертикального двигуна досягає нижнього пору, то камера повертається за допомогою цього ж двигуна у своє початкове вертикальне положення. З цього моменту приводиться в дію горизонтальний двигун, який переміщує камеру по горизонтальній осі на фіксований крок. Цей процес виконуються поки не отримаємо фіксований крок у весь горизонтальний сектор, який можна позначити як від +90 градусів до -90 градусів. По завершенню першого огляду сектору, камера повертається в початкове положення, та змінює свій напрям вертикального сканування для повторної перевірки з протилежного боку.

Технологія фотографування та збереження зображень

Зображення зберігаються у оперативній пам’яті у форматі RGB565 тому, що зображення має низьку роздільну здатність (320x240) для економії простору файлової системи, але достатню для обробки алгоритмами пошуку відблисків. Також, у зв’язку з економією простору у файловій системі, пристрій зберігатиме лише останню зроблену та проаналізовану фотографію.

Для зберігання фотографії початкова фотографія, зроблена у форматі

RGB565, має перекодуватись у JPEG формат. В цьому форматі проводиться зберігання зображення у SPIFFS, яке стає доступним до перегляду користувачу.

Технологія виявлення відблисків

Для оброблення та аналізу фотографії використовується окремий червоний канал. Це означає, що пікселі змінюються таким чином, що для побудови карти відбиття використовується лише R (червоний) компонент. Таким чином, зображення стає більш чітким через білі плями, які можуть бути характерними ознаками присутності об’єктива камери.

Аналіз відбувається в процесі побудиви масиву пікселів, який перевірятиметься значенням R-каналу. Внаслідок чого, після аналізу інтенсивності області можна визначити чи підозріле воно.

В даній технології виявлення відблисків обов’язково мають бути фільтри, які відсіюватимуть широкі області.

Технологія доступу до пристрою через веб-інтерфейс

Веб-інтерфейс системи доступний через веб-сервер, який обробляє HTTP-запити. Користувач отримує доступ за допомогою клієнтського браузера, який підтримує скриптову мову програмування JavaScript.

В розпорядженні користувача стає доступним повне керування(старт/стоп сканування), перегляд стану виконання сканування та графічне відображення результатів.

Функціонал веб-інтерфейсу написаний за допомогою HTML, CSS, JavaScript. Що надає можливість динамічно оновлювати інформацію без перезавантаження сторінки.

Повний цикл функціонування системи

Таким чином, повний цикл системи включає в собі:

– Ініціалізацію апаратної та програмної частин;

– Початкове позиціонування;

– Покрокове сканування простору у двох плоищнах;

– Фотофіксацію кожної позиції;

– Аналіз кожного фото для виявлення потенційних відблисків;

– Збереження результатів і подання їх через веб-інтерфейс.

3.2 Розробка структурної схеми

Розробка структурної схеми передбачає створення структурної схеми, яка відображає логічну побудову системи кібербезпеки виявлення прихованих камер. Це дозволить уявно розділити складну систему на окремі логічні блоки, які взаємодіють між собою через чітко визначені інтерфейси. Це спростить налагодження, тестування та подальше масштабування проєкту та є невід'ємним етапом розробки цілої системи.

Структурну схему програмної частини можна розділити на окремі модулі. З точки зору архітектури система є модульною, що дозволяє використовувати кожен компонент незалежно від інших з мінімальною кількістю точок зв'язку.

Система поділятиметь на 3 основні рівні:

– Рівень ініціалізації та комунікації;

– Рівень інтерфейсу та візуалізації;

– Рівень управління та обробки;

Ініціалізація та комунікаційний рівень

На цьому етапі апаратні та програмні частини вже готові до використання. У першу чергу запускається модуль ініціалізації системних компонентів, який запускає апаратні частини та оголошує необхідні зміни для сихронізації та конфігурації.

До модуля ініціалізації компонентів системи відносяться:

– Камера;

– Крокові двигуни з драйверами A4988;

– Створення та налаштування AP;

Після завершення ініціалізації системи починають працювати наступні два модулі:

– Модуль підключення клієнта – система перебуває в стані очікування та не продовжує подальшу ініціалізацію поки не буде підключено клієнта до АР.

– Модуль запуску веб-сервера – керує вбудованим HTTP-сервером, приймає запити на сторінці керування, відправляє команди та виводить фотографії.

Ці два компоненти працюють разом, щоб створити основу мережевого з'єднання між пристроєм та клієнтом без простою системи та створює веб-інтерфейс для доступу користувача.

Рівень керування та обробки

Центром цього рівня є модуль відображення інтерфейсу, який відповідає за показ веб-сторінки клієнта, її оновлення та отримання команд від користувача. Він не тільки відповідає за надсилання команд запуску сканування, а й за надання вказівок щодо представлення результатів.

Основний функціонал модулю відображення інтерфейсу:

– Запуск сканування (активує модуль запуску сканування);

– Відображення прогресу сканування (оновлює модуль оновлення прогрес-бару);

– Відображення результатів з виявленими підозрілими спалахми;

– Продовження сканування(активує модуль продовження сканування);

Основні функціональні модулі пов'язані з ним:

– Модуль запуску сканування – відповідає за початкову позицію руху камери, відповідно до запрограмованого профілю руху.

– Модуль аналізу фотографій – відповідає за алгоритм виявлення відблисків, перевіряючи кожен кадр, знятий системою. Використовує порогове значення для перевірки інтенсивності пікселів в R-каналі.

– Модуль зберігання оброблених фотографій – відповідає за збереження останньої обробленої фотографії в SPIFFS. Що дозволяє потім модулю відображення інтерфейсу виводити останнє зроблене та проаналізоване зображення.

– Модуль продовження сканування – відповідає за продовження сканування у разі виявлення підозрілої ділянки. Сповіщує користувача про виявлену зону, та очікує додаткової перевірки.

Інтерфейсно-візуалізаційний рівень

На даному рівні відбуваєть візуалізація прогресу проробленої роботи системи, що дозволяє зрозуміти скільки залишилось приблизно до закінчення сканування.

Даний рівень складається всього лише з одного модулю, а саме: модулю оновлення прогресу – це модуль, який регулярно оновлює статус сканування у режимі реального часу. Цей модуль дозволяє динамічно оновлювати прогрес-бар та статус роботи системи без потреби в оновленні цілої сторінки.

Запропонована архітектура (рис. 3.1) системи забезпечує чіткий розподіл функцій між модулями та значну гнучкість у зміні окремих компонентів без порушення загальної логіки системи. Модульна конструкція полегшує тестування та налагодження, дозволяє включати додаткові функції для майбутнього розширення, наприклад як хмарне зберігання результатів, надсилання сповіщень, підключення штучного інтелекту для виявлення та класифікації сприйнятих відблисків.

Система є надійної та масштабованою завдяки структурованому підходу та чіткому розподілу логіки в модулях. Це робить її легко розширюваною в модулях, що дозволяє додати все нові модулі без впливу на роботу основної системи.

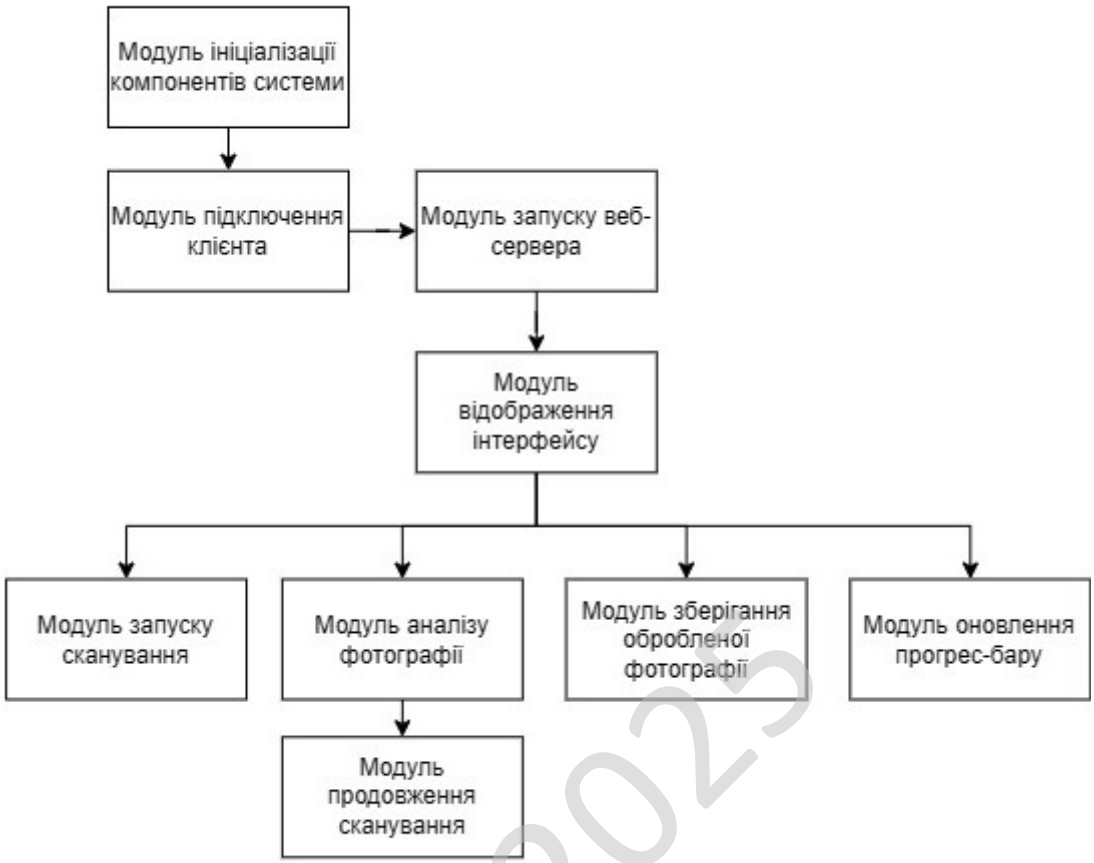

Рисунок 3.1 – Структурна схема проєкту

3.3 Розробка функціональної схеми

Функціональна схема проєкту показує логічну послідовність всіх основних етапів роботи системи кібербезпеки виявлення прихованих камер на базі модуля ESP32-CAM. Її конструкція базується на принципі покрокової обробки даних, тобто від ініціалізації модулів до завершення одного циклу сканування з обробкою зображення та відправленням результату користувачеві. На схемі (рис. 3.2) кожен блок є окремим функціональним блоком, який виконує певну функцію в загальній структурі

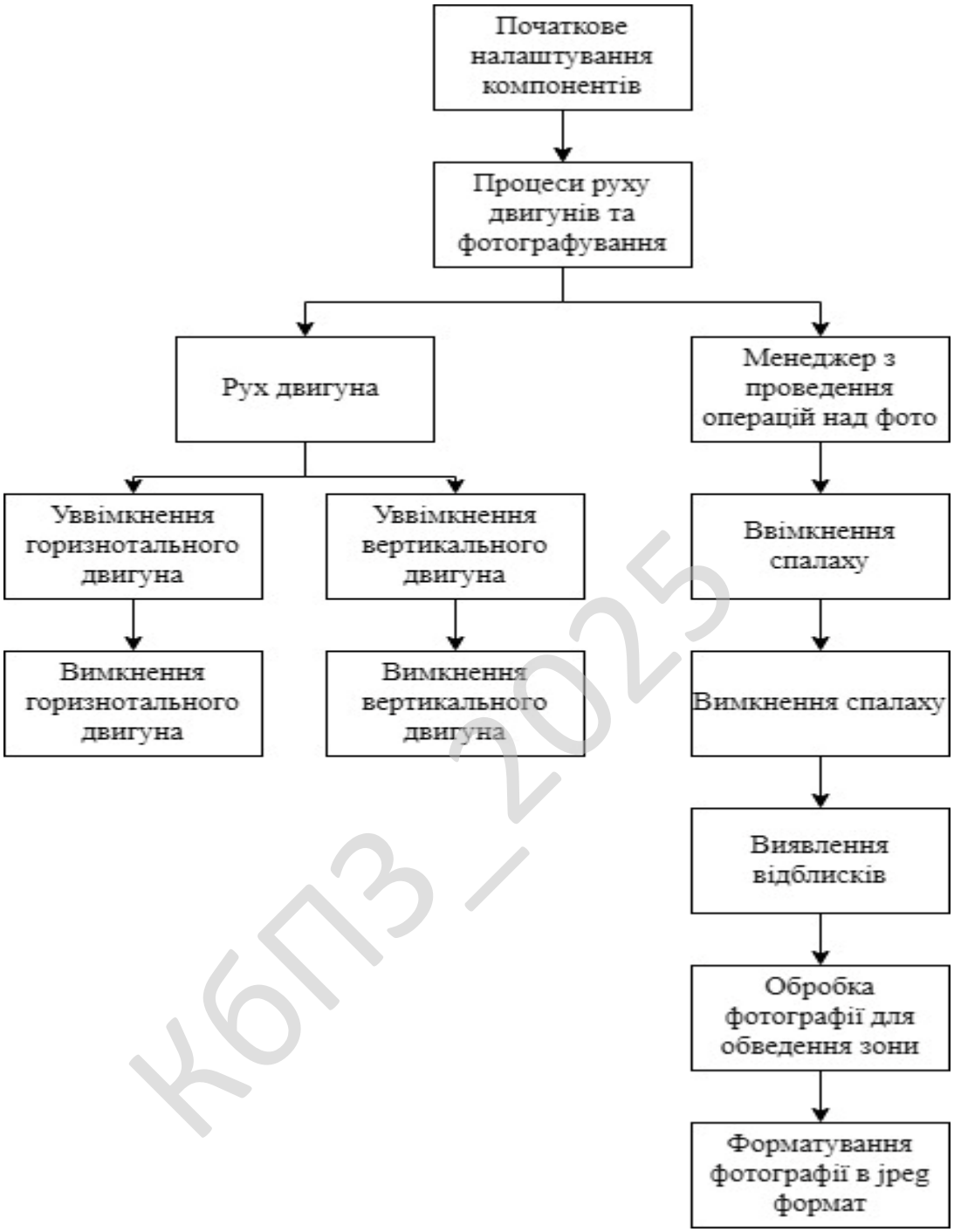

Рисунок 3.2 – Функціональна схема проєкту

Живлення пристрою – це перше завдання в ланцюжку функцій. Воно пробуджує контролер, крокові двигуни, перефирійні модулі, такі як камера, підсвічуванням спалахом. Також на даному етапі відбувається налаштування параметрів камери, встановлення файлової системи SPIFFS, конфігурція Wi-Fi

мережі. Сам пристрій тут ще не виконує жодних активних завдань. Замість цього він переходить у режим очікування, щоб прийняти наступні команди від користувача.

Наступним завданням даного пристрою є процеси руху двигунів та фотографування. Він починається після установки, конфігурації всіх апартних частин та команди користувача.

Цикл сканування складається з переміщення камери по координатах приміщення і створення фотографії на кожному кроці цього процесу. Кожна точка сканування є окремою точку огляду, з якої буде зроблено знімок із увімкненим підсвічуванням.

Керування положенням камери (двигунами) активує покрокові двигуни (горизонтальний або вертикальний) і переміщує положення камери. Після досягнення нової координати двигун деактивується, щоб уникнути перенапруги, вібрацій та наслідків, що можуть спричинити похибку та неточності під час фотозйомки.

Підсвічування спалахом включається перед кожним процесом фотографування. Світло надає змогу виявляти відблиски від зеркальних поверхонь, які властиві об’єктивам камер. Поки фотографія не буде зроблена, підсвічування залишається увімкненим. Час загорання спалаху перед фотозйомкою – 200 мс, а час захитання спалаху після фотозйомки – 150 мс.

Зображення захоплюється напряму в менеджері з проведення операцій над фото для зручності передачі самої фотографії між різними функціями. Камера робить фото в режимі з підсвічуванням. Зображення зчитується у форматі RGB565. Фотографія робить тільки після завершення всіх рухів, щоб уникнути розмиття або спотворення картинки.

Обробка фотографій на виявлення відблисків є одним з основних функціональних етапів роботи системи. Даний етап поділяється на декілька підетапів. Виділення червоного каналу це перша частина роботи даного пристрію. Вона дозволяє обробляти зйомок так, щоб залишити тільки

інтенстивність червоного світла – саме це дозволяє віднайти більш точно підозріли відблиски від оптичних елементів. Наступним кроком проводиться пошук яскравих плям. Алгоритм сканує групу пікселів з високою яскравістю червоного кольору, вловлюючи потенційні світлові аномалії. Як тільки знаходяться такі аномалії, починає працювати зональний аналіз, де аналізується розмір, форма об’єкту. Якщо такі аномалії були виявленими, викликається функція яка обводить дану зону, вказуючи на можливе місцерозташування прихованої камери.

Після процесу обробки, зображення необхідно перетворити з формату RGB565 у формат JPEG. В цьому процесі фотографія перетворюється на остаточне зображення, яке вже можна використовувати для виведення користувачу через веб-інтерфейс. На цьому ж етапі воно зберігається замінюючи останнє зроблене, що зроблено для економії ресурсів пам’яті. Буфер ж в цей момент теж очищається.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результати виконання бакалаврського проєктування, наведена на рисунку 3.3.

Рисунок 3.3 – Діаграма взаємодії процесів

Після початку роботи пристрою, відбуваються спочатку процеси ініціалізації системи та конфігурації АР. Далі пристрій переводиться в режим очікування підключення клієнта, заморожуючи всі інші процесі. Коли клієнт підключається завершується остаточна конфігурація веб-серверу та пристрій переходить знову в режим очікування на команду від клієнта. На даному процесі всі підготовчі процеси виконані і пристрій готовий до роботи. Коли системі надходить команда на запуск сканування – відбувається процес поворотів двигунів у початкове положення. Далі проводиться повний аналіз фотографії. У випадку якщо було виявлено відблиск проводиться обробка та виділення підозрілої зони. Після чого система зберігає знімок та переходить в стан очікування, чекаючи на команду продовження або зупинки сканування від користувача. Це надає змогу та час користувачу перевірити чи дійсно там щось є, чи ні. Надалі якщо цикл сканування підходить до свого завершення то

відбувається процес перевірки завершення сканування, який у разі успішності повертає двигуни у початкове положення та переходить до процесу очікування команди від клієнта. На даному етапі клієнт може як знову запустити новий цикл сканування, так і завершити роботу пристрою. Якщо ж перевірка завершення сканування неуспішна, цикл сканування продовжується.

Принципова схема пристрою

В даній схемі (рис. 3.4) показуються всі використовувані елементи в пристрої та схема підключення контактів.

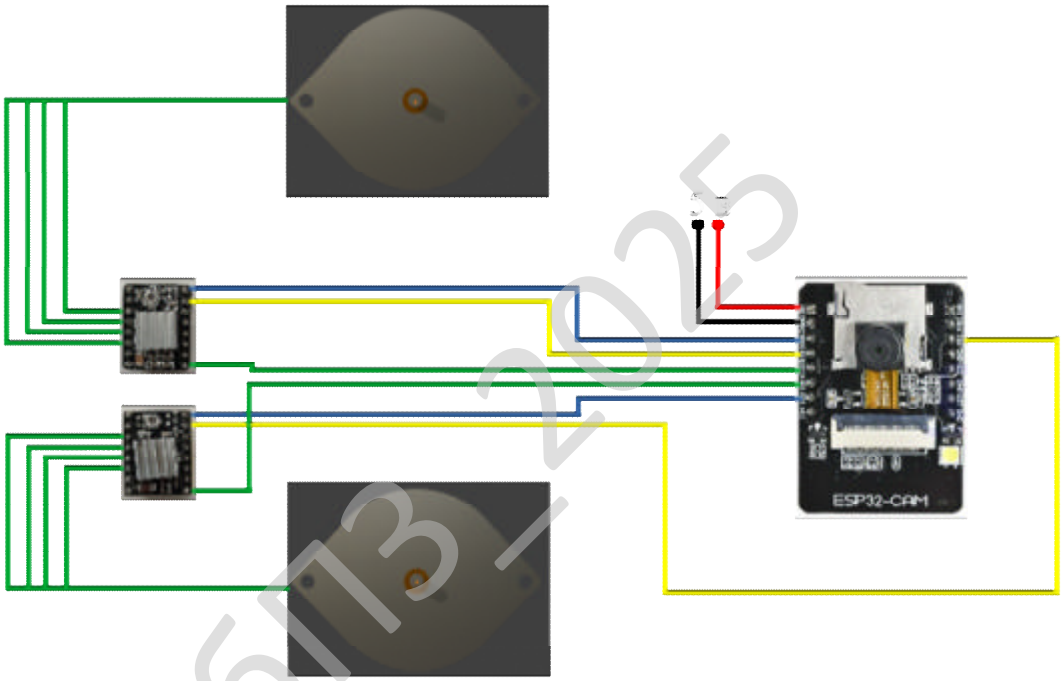

Рисунок 3.4 – Принципова схема пристрою

Таким чином, розглянувши опис системи, структурну, функціональну схеми та діаграму взаємодії процесів можна переходити до опису блок-схем основної програми, та підпрограм, що використовуються для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми.

Кроки, які виконуються під час її роботи наступні: спочатку відбувається ініціалізація констант, пінів підключення двигунів, камери та спалаху. Далі відбувається етап конфігурації компонентів. Після якого, очікується підключення клієнта для продовження. Коли клієнт підключається відбувається запуск циклу сканування. Даний цикл викликає низку підпрограм, які відповідають за свої функціональні частини, а саме: за рух двигунів, керування зображеннями, уввімкнення та вимкнення спалаху, аналізом та виявленням спалаху, обведенням зони відблиску якщо вона є, та останньою частиною це форматування зображення в зжатий формат зрозумілий браузеру.

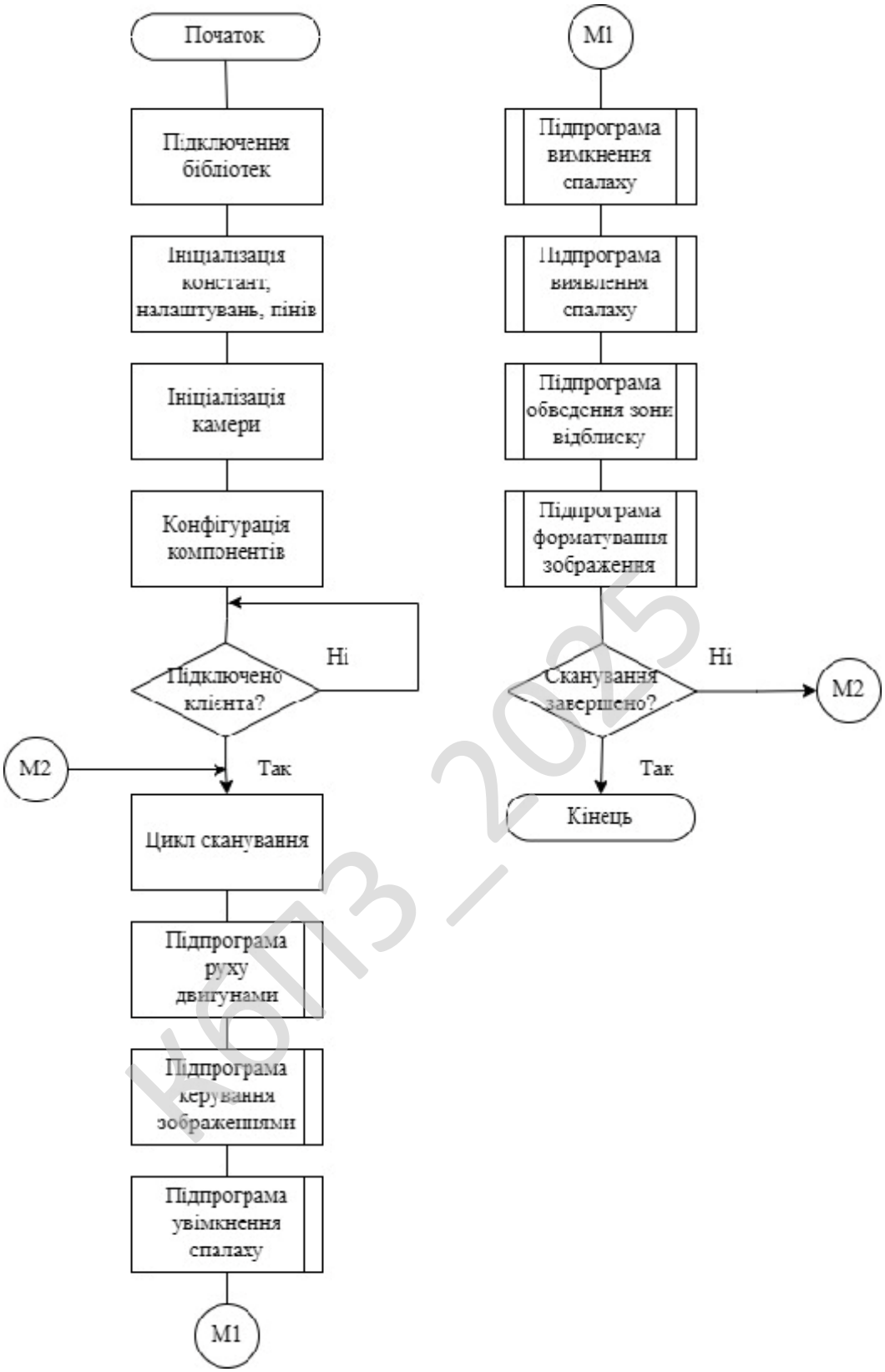

Рисунок 4.1 – Блок-схема основної програми

Детальніше варто розглянути підпрограми основного циклу, конфігурації елементів, виявлення спалаху.

На рисунку 4.2 розташовується блок-схема підпрограми setup(), що викликається лише один раз при кожній подачі живлення на пристрій.

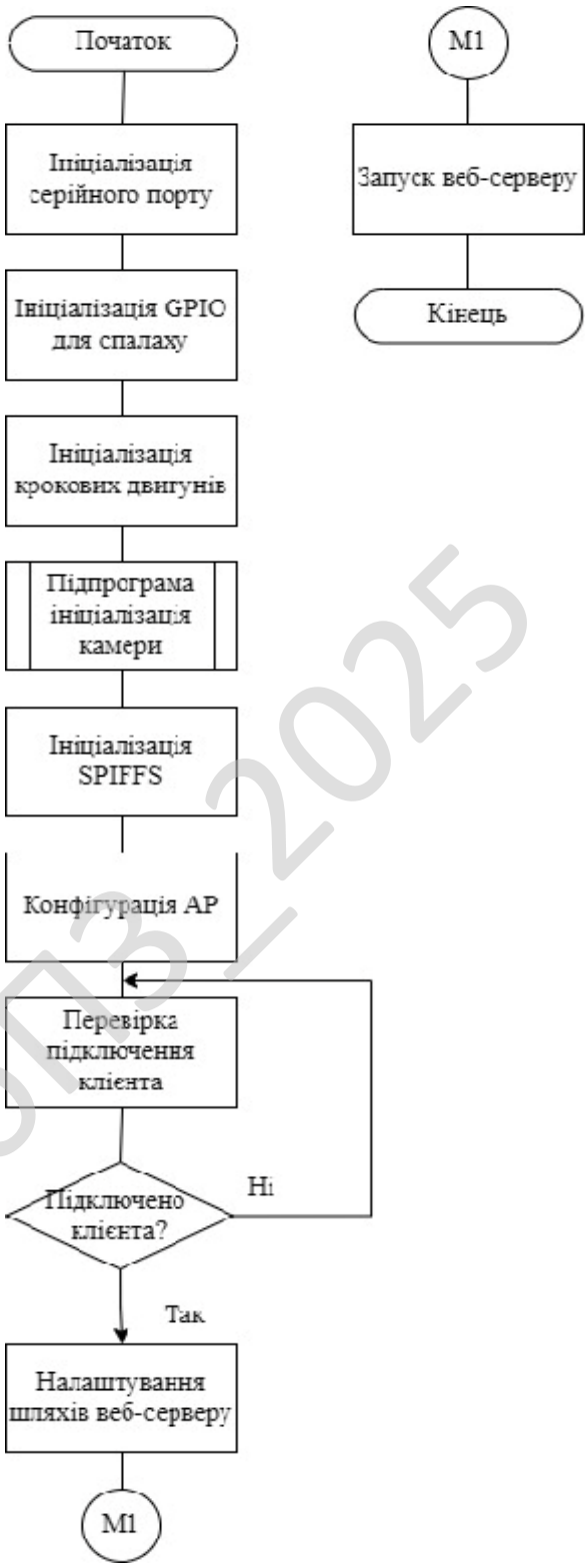

Рисунок 4.2 – Блок-схема підпрограми setup()

Для ініціалізації серійного монітору використовується наступна команда, що дозволяє налагодити моніторинг роботи ESP32-CAM через послідовний порт на швидкості 115200 біт/с.

Він дозволяє досить зручно моніторити всі процеси, які виконує пристрій. Після успішної ініціалізації серійного монітору, варто розглянути яким саме чином ініціалізуються виводи (GPIO) до двигунів та спалаху, та самі налаштування руху двигунів, як для вертикального, так і для горизонтального. Для прив'язки GPIO двигунів використовується функція pinMode() з вхідними даними про під'єднаний на схемі пін керування двигуном для зчитування сигналу, та параметром OUTPUT – пін що працює як вихід, використовується для подачі сигналу. Дана функція є стандартною для Arduino, вона задає режим роботи певного піну мікроконтролера.

Піни визначені глобально на початку файлу, відповідно до приєднання на мікроконтролері, після підключення бібліотек.

Тобто, даною частиною коду задається напрямок керування живленням для драйверів двигунів, що надає змогу в подальшому використувати їх у циклі сканування. Після налаштування вони одразу ж вимикаються, щоби уникнути навантаження до моменту використання у циклі сканування.

Ініціалізація файлової системи SPIFFS здійснюється за допомогою умовного оператора if().

Розглянемо, яким чином відбувається очікування підключення клієнта. Для перевірки цього, використовується певна функція класу WiFi.h, що перевіряє кількість підключених пристроїв до точки доступу. В разі, якщо ця кількість більше 1, то це означає що клієнт успішно підключений до мережі.

Для візуального зворотного зв'язку в консоль серійного монітору постійно виводиться крапка, поки не буде встановлено з'єднання. Це дозволяє зрозуміти, що пристрій знаходиться в режимі очікування і він не завис.

Після налаштування базових компонентів пристрою, ключовою складовою для обробки, керування та взаємодії з користувачем є організація маршрутів

запитів до веб-сервера, що дозволяє отримувати максимальні інформативні дані про процес сканування або його результати та виведення їх у веб-інтерфейсі. Для цього використовується функція webserver.on(), що налаштовує HTTP-запити на задані URL-адреси.

Перший і основний маршрут – “/” використовується для обробки головної сторінки системи. Приймає та обробляє запити типу GET. Він відповідає за надсилання HTML-сторінки користувачеві. Сама веб-сторінка знаходиться у змінній index_page, заздалегідь збереженій у пам’яті PROGMEM при розробці ПЗ. Воно то й дозволяє виводити в браузер інтерфейс з керуванням сканування

Другий маршрут – “/start-scan” використовується для запуску процесу сканування. Він відповідає за старт процесу, якщо його ще не було розпочато. Для контролювання стану процесу використовується змінна scan_in_progress типу bool. Для ініціалізації налаштувань двигунів перед початком процесу сканування використовується змінна initialized типу bool.

Наступний маршрут – “/photo” , що відповідає за надсилання останнього зробленого знімка. Він використовує систему SPIFFS, щоб відкритий останній зроблений знімок в кореневій папці з назвою /last.jpg. Якщо файл існує, він успішно передається клієнту як JPEG-зображення. Такий підхід дозволяє переглядати результати сканування безпосередньо в браузері.

Маршрут “/status” дозволяє клієнтові дізнатись, чи в даний момент відбувається сканування. Це є корисною інформацією, що дозволяє інтерфейсу у режимі реальног очасу відображати стан пристрою. За допомогою цього маршруту повертається всього лише два можливі значення: true або false. Це дає зручну можливість для обробки на клієнтській стороні за допомогою JavaScript.

Маршрут “/progress” повертає поточний прогрес сканування у вигляді відсотка. Змінна, що зберігає відсоткове значення виконаного сканування називається scanning_progress. Вона надсилається як текст, що дозволяє візуалізувати його на веб-сторінці у вигляді прогрес-бару та числового значення.

Маршрут “/reset-flash-detection” використовується для продовження

процесу сканування після зупинки, викликаної виявленням потенційного відблиску. Цей запит дозволяє користувачу вручну відновлювати процес сканування. Визначається виявлення потенційної прихованої камери за допомогою змінної isFlashDetected типу bool. В серійний монітор відповідно виводиться відповідне повідомлення.

Маршрут “/continue-scan” працює схоже на попередній, але реалізує логіку підтвердження виявлення. Це дозволяє продовжити сканування лише після ручного підтвердження користувачем. Таким чином, програма уникає помилкових зупинок та продовжити роботу після проведення аналізу виявленого сигналу.

Останній маршрут “/is-flash-detected” дозволяє дізнатись чи було виявлено системою підозрілий відблиск. Цей запит викоиртсовується інтерфейсом для зручного оновлення статусу, підсвічування кнопки або виведення попередження про потенційну приховану камеру.

За допомогою даної логіки організовується повноцінна архітектура клієнт-серверної взаємодіїї з ESP32-CAM. Ці маршрути створюють повноцінну взаємодію між пристроєм та користувачем через браузер.

Після реалізації всіх маршрутів веб-сервер запускається і готовий до обробки GET-запитів за допомогою команди begin().

На рисунку 4.3 зображено блок-схему основного циклу сканування.

Дана підпрограма відповідає напряму за керуванням положенням двигунів, виклики підпрограм для створення та аналізу зображень, та синхронізації їх між собою.

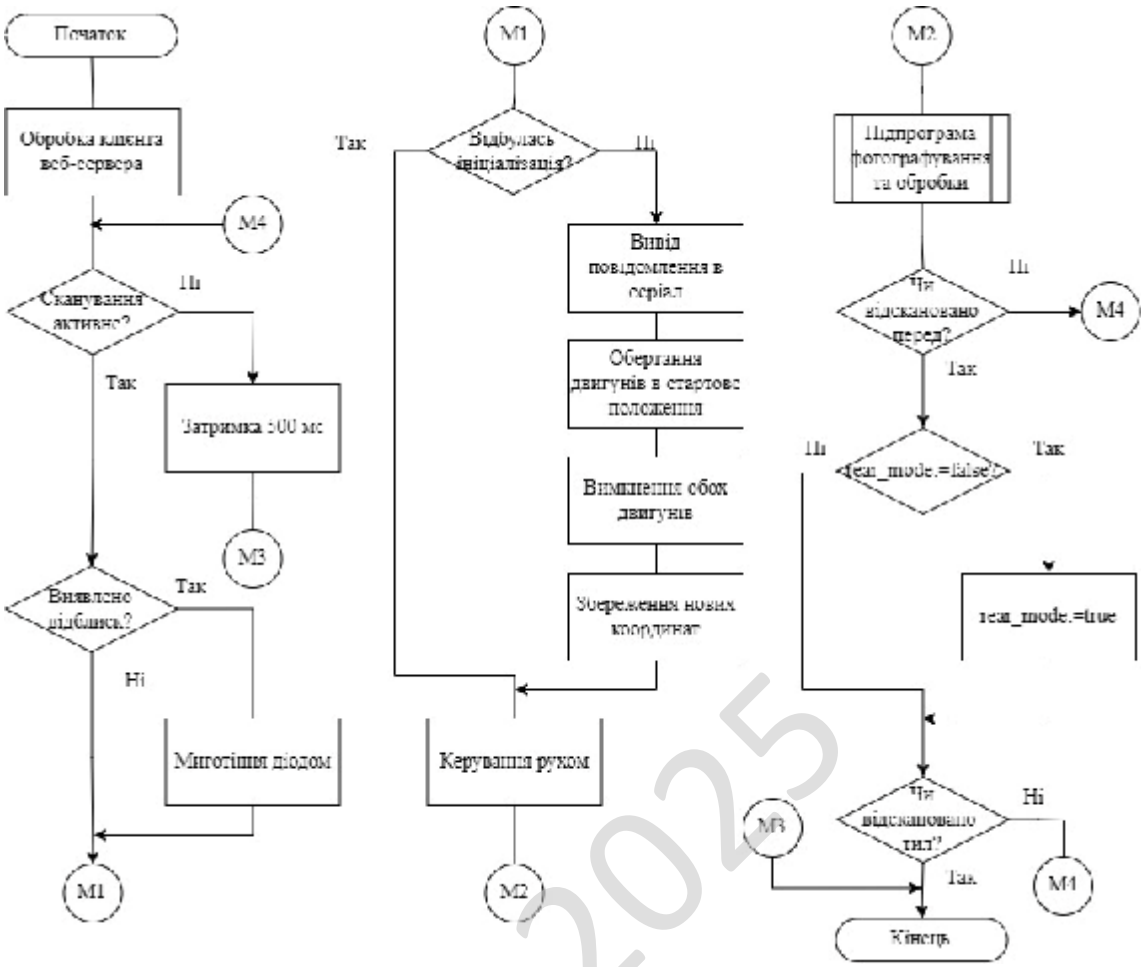

Рисунок 4.3 – Блок-схема підпрограми циклу сканування

За допомогою однієї змінною scan_in_progress відбувається запуск процесу сканування. Система знаходиться в очікуванні, поки користувач через веб-інтерфейс не активує сканування.

Для старту реалізується процедура ініціалізації пристрою перед початком сканування, що виконується лише один раз за одне сканування – при першому запуску сканування. За допомогою умовного оператора if перевіряється, чи була виконана ініціалізація. Це дозволяє уникнути повторної ініціалізації під част наступних ітерацій сканування. Через серійний монітор завжди виводяться повідомлення про початок ініціалізації, та руху двигунів у початкові позиції.

Після, викликаються дві функції moveStepper() для горизонтального (stepper_h) та вертикального (stepper_d) крокових двигунів. Дана функція приймає в себе наступні аргументи: цільову позицію, до якої слід перемістити двигун та

пін, через який здійснюється живлення обмотки двигуна. Це дозволяє обидвом двигунам переміщуватись у вихідне положення, яке відповідає початку сканування.

В ході ітерації циклу обов'язково існує очікування завершення руху обох двигунів за допомого циклу while() та функції distanceToGo(), що повертає кількість кроків, які залишилось зробити до досягнення цільової позиції. Це означає, що поки кількість кроківі для руху буде більшою за 0, двигуни будуть продовжувати свій рух, за допомогою команди run(), поки задача не буде досягнута.

В даному коді, для двигунів виставляються максимальний поріг кроку для сканування фронтової частини приміщення перед пристроєм. Поки дистанція двигунів не буде дорівнювати 0, двигуни будуть рухатись, інакше вони вимикаються. Також додатково в даній частині коду для моніторингу використання ресурсів реалізований вивід вільної пам'яті в серійний монітор за допомогою функції ESP.getFreeHeap().

Якщо обидва двигуни досягають своїх кінцевих позицій в даній ітерації циклу сканування, тоді відбувається їх повна зупинка, фіксується фото з затримкою, щоб фото вийшло не розмазаним та сфокусованим. Коли живлення двигунів вимкнене, відбувається одночасний вивід інформації про позиції обох двигунів серійний монітор та вспливають сповіщення про процес зйомки знімку у поточній точці. За фото частину відповідає функція photoHandler(). Вона робить фотографію та проводить всі пов'язані дії із обробкою зображення. Також додається за допомогою постінкремента змінна scanning_progress з постійним інкрементом – 1. Вона використовується для відображення загального прогресу виконання циклу сканування приміщення.

Далі відбувається рух двигунів. Спочатку вертикальний проходить свій повний сектор руху. Якщо він закінчив свій сектор змінюється значення змінної scanning_down, що дозволяє повернути вертикальний двигун у своє стартове положення та в дію приводиться горизонтальний двигун. У зв'язку з технічними

обмеженнями та неможливістю провести за один раз сканування в 360 градусів, якщо горизонтальний двигун досягне свого мінімума, тобто пройде повністю свій сектор, тоді відбувається процес зміни кроків вертикальної камери, що дозволяє розвернути камеру в вертикальній осі в тилову частину пристрою. Проводиться повторне сканування з врахуванням цього. Це дозволяє охопити повністю все приміщення.

Також слід роз’яснити алгоритм для виявлення відблисків. Даний алгоритм дозволяє виявляти до 3 відблисків на зображенні. Він працює таким чином, що на початку створює масиви для зберігання координат відблисків, виділяє для пікселів які будуть оброблятись пам’ять динамічним методом за допомогою malloc(). Виявлення зони з підозрілим відблиском відбувається у процесі пошуку найяскравіших пікселів та поєднання їх у групи між собою за відстаню. Аналіз проводить у червоному спектрі для більшої точності. Оцінювання яскравості пікселя відбувається за допомогою формули: brightness = (r * 0.299 + g * 0.587 + b * 0.114). Якщо піксель перевищує поріг яскравості на нього ставиться мітка, яка в подальшому буде використана для перевірки або додавання пікселя до такого ж яскравого пікселя з ціллю об’єднати їх. Після цього оброблені пікселі звільняються для вивільнення пам’яті, та, за допомогою фільтрів, відбувається процес перевірки валідності. Фільтри мають на меті відкинути занадто великі або витягнуті об’єкти. Наступним кроком виділяється зона, яка буде відведена іншому алгоритму, що малюватиме прямокутник навколо даної зони.

Алгоритм виділення зони (ВО) окремо малює лінії спочатку для горизонтальних прямих, потім для вертикальних прямих, таким чином утворюючи прямокутники жовтого кольору.

4.2 Захист розробленого програмного забезпечення

Однією з функцій захисту при розробці ПЗ для пошуку прихованих камер є власний алгоритм обробки зображень, який проводить пошук та виявлення

оптичних відблисків. Зазначений алгоритм, а також метод і програма на його основі, служить як для виконання функціонального завдання пристрою, так і є своєрідним захистом логіки обробки даних через складність його реалізації для зловмисників, які не мають достатніх знань про контекст та параметри.

Основна мета алгоритму полягає в тому, щоб зчитати зображення з червоним підсвічуванням та в червоному спектрі, щоб знайти яскраві пікселі, які можуть вказувати на те, що оптичний елемент відбиває світло.

Спочатку для масиву оброблювавних пікселів використовуєтьс динамічне виділення пам’яті. Наступним кроком є розподіл RGB-каналів зображення у форматі RGB-565. Яскравість можна обчислити за допомогою формули перерахунку моделі сприйняття світла людиною:

Brightness = 0.299 * R + 0,587 * G + 0.114 *B

Якщо піксель має достатньо високу яскравість у червоному каналі і всі задані пороги проходять, алгоритм вважає його підозрілим. Він дозволяє об’єднати пікселі в виділення за критерієм просторової близькості, якщо вони будуть достатньо близько друг до друга. Дане об’єднання називається кластером пікселів.

Кожен отриманий кластер яскравих пікселів проходить фільтрацію за площею та співвідношенням сторін, щоб запобігти хибним спрацьовуванням. Фільтри допомагають усунути відблиски від інших джерел світла або відбите світло від плоских блискучих поверхонь, таких як зеркала, вікна та інші.

Одночасно можливе оброблення до трьох точок відблисків за допомогою системи окремих координатних масивів. Це запобігає перевантаженню або неправильній роботі в умовах обмеженої пам’яті пристрою.

У процесі обробки зображення окреслюється жовтим(-ими) прямоктниками та зберігається у файловій системи пристрою у форматі JPEG.

Таким чином, розроблене ПЗ захищається на рівні алгоритмічної слкадності, обмеженості ресурсів, внутрішнього контролю достовірності результатів. Це унеможливлює стороннє втручання, підміну або зловживання

алгоритмами системи зі сторони осіб, які зовсім не мають уявлення як працює системи “під капотом”.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Необхідно розглянути роботу програмного забезпечення системи кібербезпеки виявлення прихованих камер шляхом пошуку відблисків від оптичних елементів з точки зору користувача.

Для початку необхідно під’єднати живлення до пристрію за допомогою блоку живлення.

Після того як пристрій заморгає вбудованим світлодіодом необхідно підключитись до щойно створеної Wi-Fi мережі HiddenCameraDetector_Access (рис. 5.1) за допомогою будь якого пристрою, що може відкривати веб-сторінки та підтримує JavaScript.

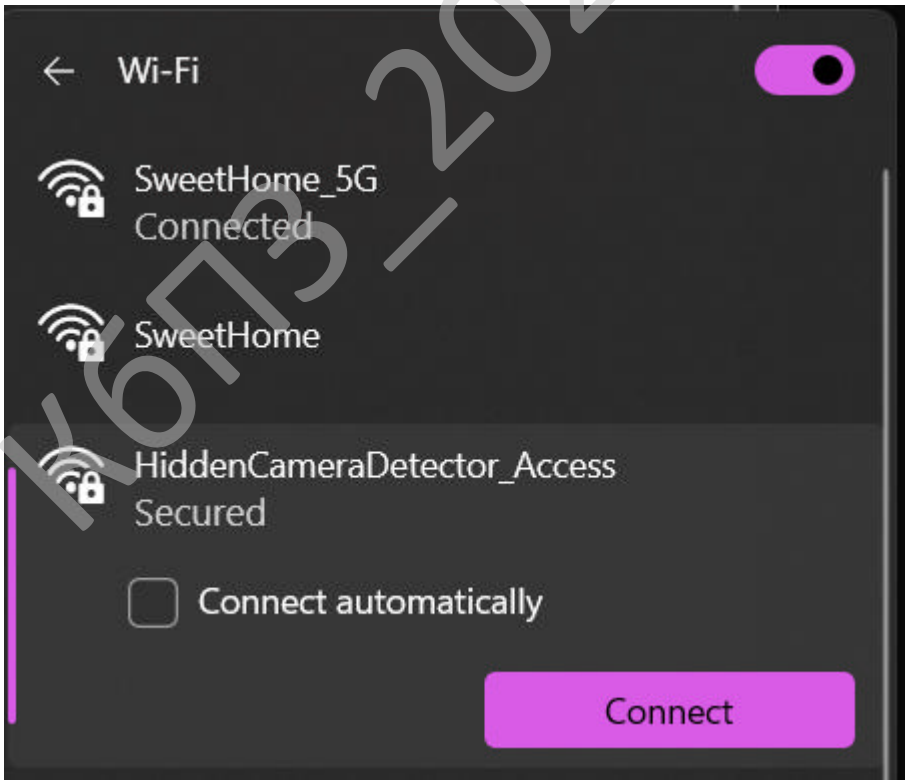

Рисунок 5.1 – Новостворена Wi-Fi мережа пристрою

Дана мережа захищена паролем. При першому під’єднанні з користувацького пристрою мережа запросить пароль. В такому випадку треба

ввести наступне код: camera123.

Після під’єднання до АР необхідно відкрити браузер на вибір користувача(напр. Mozilla Firefox, Microsoft Edge, Google Chrome та інші). В полі адреси необхідно ввести наступну IP-адресу: 192.168.4.1. Якщо апаратний комплекс успішно заживлений, всі компоненти ініціалізовані та сконфігуровані, користувацький пристрій успішно під’єднаний до мережі пристрою, то має відкритись веб-сторінка (рис. 5.2) керування пристроєм.

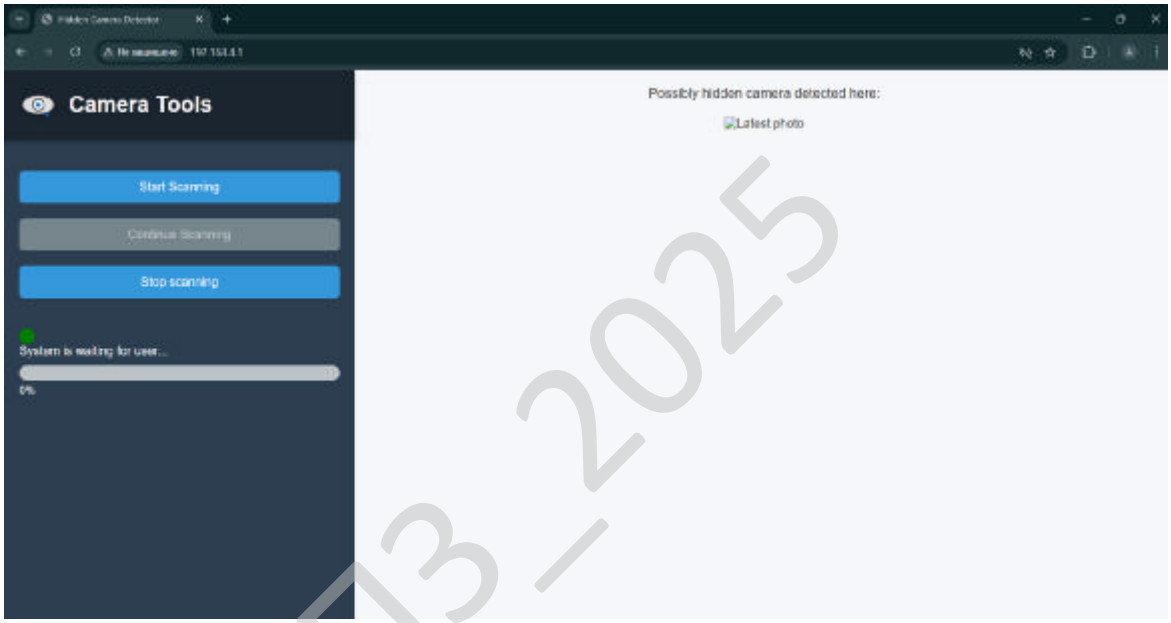

Рисунок 5.2 – Веб-інтерфейс пристрою

В даному меню керування пристроєм маємо 3 кнопки, що відповідають за: початок сканування, продовження сканування – за замовчуванням дана кнопка неактивна, вона стає активною тільки у випадку якщо алгоритм виявлятиме підозрілий відблиск, кнопка термінового завершення сканування. Також є інформативні статус-бар, який поєднує в собі флажок зайнятості системи (зелений колір – якщо система знаходиться в режимі простою, червоний – запущено процес сканування) та прогрес-бар, що показує графічний прогрес процесу сканування та прогрес в цифрах для зручності.

Для запуску сканування необхідно натиснути на кнопку Start Scanning.

Якщо алгоритм виявлення відблисків підозрює на приховану камеру то

виводиться відповідно сповіщення та стає активною кнопка продовження сканування. Підозріла зона обводиться на зображенні (рис. 5.3) жовтим прямокутником.

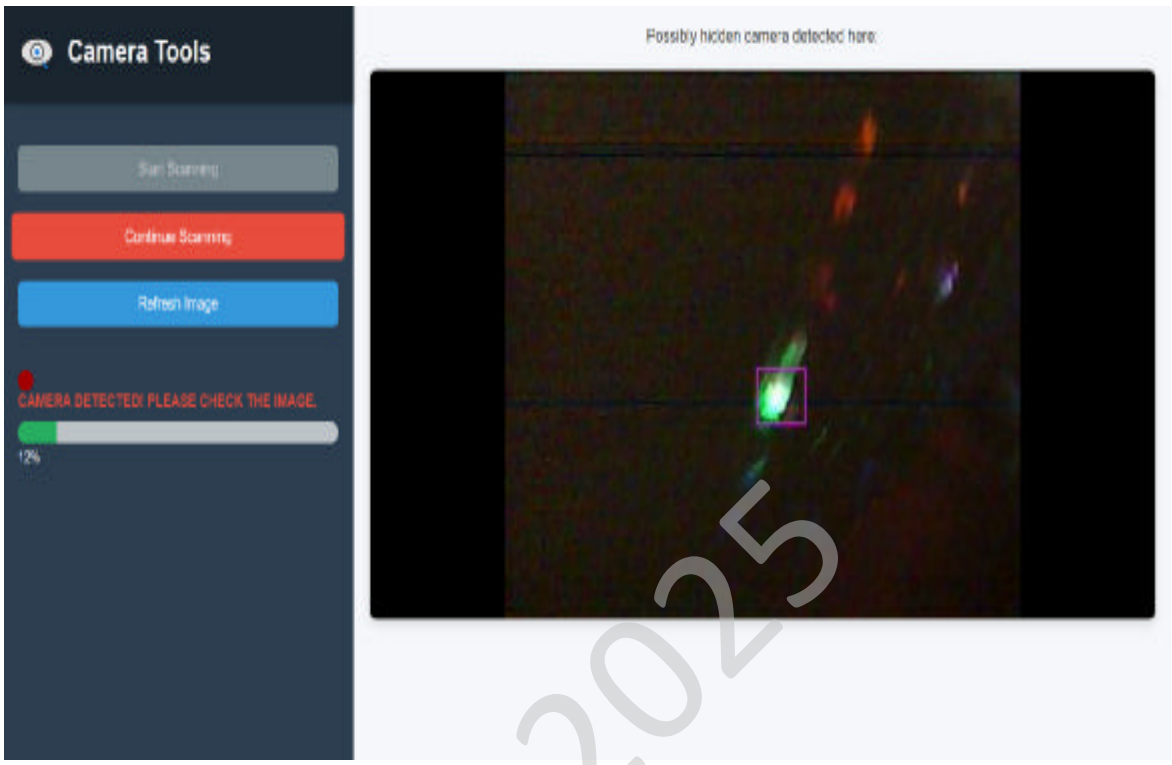

Рисунок 5.3 – Вигляд ПЗ під час сканування при виявленні відблиску лінзи

6 ОСНОВНІ ВИСНОВКИ

Апаратне та програмне забезпечення, розроблене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти , призначене для виявлення прихованих камер шляхом сканування простору за допомогою плати ESP32-CAM та двох крокових двигунів. Система поєднує методи комп'ютерного зору, автоматичного управлінняя та основ веб-програмування з цифровою обробкою зображень.

У результаті дослідження було досягнуто наступних результатів:

– Проведено аналіз існуючих рішень виявлення прихованих камер, що дозволило визначити основні недоліки ручних методів та переваги автоматизованих методів.

– Розроблено алгоритм сканування простору, який за допомогою двох крокових двигунів рухає камеру по визначеній області в горизонтальних та вертикальних осях, з фіксацією позиції в кожному положенні.

– Розроблено систему аналізу зображень, яка розпізнає яскраві плями в червоному спектрі RGB. Вона дозволяє більш точно виявляти потенційні відблиски, характерні оптичним елементам прихованих камер.

– Розроблено веб-інтерфейс, який дозволяє отримувати інформацію про сканування, виведення результатів сканування. Це забезпечує зручність і можливість відстеження з будь якого пристрою.

Розроблена система вирішує актуальну проблему виявлення прихованих камер відеоспостереження в невеликих приміщеннях. Система використовує недорогі компоненти, що дозволяє широко використовувати її.

З технічної точки зору система відповідає принципам модульності, масштабованості та відкритого коду. Програмне забезпечення розроблено на мові C++ за допомогою Arduino IDE. Веб-інтерфейс створюється за допомогою

HTML, а також елементів CSS та JavaScript, які вбудовані у прошивку пристрою у вигляді заголовного файлу.

При проєктуванні системи були враховані такі фактори, як енергоспоживання, обмеження на максимальні кути повороту крокових двигунів, обмеження доступу до внутрішніх сервісів ESP32 та чутливість до коливання освітленності. Архітектура проєкту виконана в клієнт-серверному підході.

В кінцевому підсумку запропоноване рішення демонструє практичну застосовність, обґрунтовує ефективність використаних методів і технологій і демонструє відповідність технічному завданню. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення, в тому числі:

– Удосконалення алгоритмів виявлення відблисків;

– Застосування інших модулів;

– Інтеграція у хмарні сервісі з метою збереження результатів аналізу;

– Адаптація алгоритмів та визначення порогів в різних умовах;

– Інтеграції захисту пристрою різного рівня для захисту від атак на IoT-пристрої.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ziwei Liu, Feng Lin', Chao Wang, Yijie Shen, Zhongjie Ba, Li Lu, Wenyao Xu, Kui Ren. «CamRadar: Hidden Camera Detection Leveraging Amplitude-modulated Sensor Images Embedded in Electromagnetic Emanations» Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 6, 4, Article 173 (December 2022), 25 p.

2. Samuel Herodotou, Feng Hao. «Spying on the Spy: Security Analysis of Hidden Cameras» Warwick University, 2023. 19 p.

3. Zhiyuan Yu, Zhuohang Li, Yuanhaur Chang, Skylar Fong, Jian Liu, Ning Zhang. 2022. «HeatDeCam: Detecting Hidden Spy Cameras via Thermal Emissions». In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022. 14 p.

4. Susmi Elsa Joy, Nikhil Plement, Shinu CM, Sudhish Sankar, Alpha Mathew. «A review of application to detect spy camera implementation». International Research Journal of Modernization in Engineering Technology and Science, 2023. 4 p.

5. Vaishali Koul , Rakshita Macheri , RibhuVats , Liya Baby , Poonam Bari. «Hidden Camera Detection» International Journal of Advance Research in Science and Engineering. 2017, 5 p.

6. Sriram Sami, Sean Rui Xiang Tan, Bangjie Sun, Jun Han. «LAPD: Hidden Spy Camera Detection using Smartphone Time-of-Flight Sensors». The 19th ACM Conference on Embedded Networked Sensor Systems (SenSys '21), 2021. 14 p.

7. Kevin Wu, Brent Lagesse. «Do You See What I See? Detecting Hidden Streaming Cameras Through Similarity of Simultaneous Observation» Computing and Software Systems University of Washington Bothell. 10 p.

8. Y. M. Pa Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, «IoTPOT: Analysing the Rise of IoT Compromises». Usenix Workshop on Offensive Technology (WOOT), 2015. 9 p.

9. Helen Coffey. «How to spot a hidden camera in your Airbnb». URL: https://www.independent.co.uk/travel/news-and-advice/airbnb-how-to-spot-hidden-

camera-b2544883.html (дата доступу 11.02.2025).

10. B. Lagesse, K. Wu, J. Shorb, and Z. Zhu, «Detecting Spies in IoT Systems using Cyber-Physical Correlation» IEEE Workshop on Mobile and Pervasive IoT, 2018. 6 p.

11. Коротка Г. М, Петренко Т. А. «Особливості процесу виявлення прихованих відеокамер» Національнний університет “Чернігівська політехніка”, 2021. 3 ст.

12. «Пошук прихованих камер». URL: https://iac.com.ua/uk/poshuk-prihovanih-kamer/ (дата доступу 16.02.2025).

13. https://ua.big-secu.com/news/motion-detection-and-long-standby-of-hidden-se-71961362.html https://ua.big-secu.com/news/how-hidden-camera-detectors-work-a-comprehens-69785533.html (дата доступу 17.02.2025).

14. Григорук П. Р., Катаєв В. С. «Аналіз методів виявлення приховани відеокамер» Вінницький національний технічний університет, 2025. 2 ст.

15. «Виявлення руху та тривалий режим очікування прихованих камер безпеки: повний посібник із підвищення ефективності відеоспостереження». URL: https://ua.big-secu.com/news/motion-detection-and-long-standby-of-hidden-se-71961362.html (дата доступу: 26.02.2025).

16. «Посібник з мікроконтролерів ESP32». URL: https://ua.ariat-tech.com/blog/ESP32-Microcontroller-Guide.html (дата доступу 08.03.2025).

17. Darko Hercog, Tone Lerher, Mitja Truntic, Oto Tezak. «Design and implementation of ESP32-Based IoT Devices» The Special Issue IoT Sensors and Technologies for Education, 2023. 20 p.

18. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

19. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Книшук А.В. «Вступ до кібербезпеки»:

20. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

21. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

22. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

23. Ahmed D. Hassebo, Kevin B. Montes, Erick Cabrera, «Arduino-ESP32 based Smart Irrigation System» CUNY Academic Works, 2025. 10 p.

24. «Освоєння ESP32: Вичерпний посібник із найкращого навчального набору для майбутніх новаторів». URL: https://ua.sz-kuongshun.com/info/mastering-the-esp32-a-comprehensive-guide-88862689.html (дата доступу 29.03.2025).

25. «ESP32-CAM Video streaming and face recognition with Arduino IDE». URL: https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/ (дата доступу 30.03.2025).

26. Ben Everard, «ESP32-CAM Review». URL: https://hackspace.raspberrypi.com/articles/esp32-cam-review (дата доступу 01.04.2025).

27. Ibrahim Bin Mansur, «Building a Real-time OCR System with ESP32-CAM: A Complete Guide». URL: https://medium.com/engineering-iot/building-a-real-time-ocr-system-with-esp32-cam-a-complete-guide-86f40a5312dc (дата доступу 01.04.2025).

28. «Adventures with an ESP32-CAM». URL: https://hackaday.io/page/21807-adventures-with-an-esp32-cam (дата доступу

02.04.2025).

29. Ninad Mehendale, «Object Detection using ESP 32 CAM». URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4152378 (дата доступу 02.04.2025)

30. Ilham Saputra, Khoirul Ashabi, «Application of ESP32-CAM for Cloud-Based Surveillance System in 3D Printing» Journal Inov Tek Seri Mesin, 2022. 6 p.

31. Wahyu Sukestyastama Putra, Andy Setyawan, «Room security system design using ESP32-CAM with fuzzy algorithm» Mobile and Forensics (MF), 2021. 7 p.

32. Henry Dietz, Dillon Abney, Paul Eberhart, Nick Santini, William Davis, Elisabeth Wilson, Michael McKenzie, «ESP32-CAM as a programmable camera research platform» in *Proc. IS&T Int'l. Symp. on Electronic Imaging: Imaging Sensors and Systems*, 2022, pp 232-1 - 232-6

33. «Getting started with the ESP32-CAM». URL: https://dronebotworkshop.com/esp32-cam-intro/ (дата доступу 06.04.2025).

34. Theron Wierenga, «Build a video camera using the ESP32-CAM board». URL: https://www.nutsvolts.com/magazine/article/build-a-video-camera-using-the-esp32-cam-board (дата доступу 06.04.2025).

35. FutureApps Google play URL: https://play.google.com/store/apps/details?id=hiddencamdetector.futureapps.com.hiddencamdetector&hl=uk (дата доступу 16.04.2025)

36. Baseus Shop Online. URL: https://baseusonline.com/product/1487/baseus-heyo-series-ii-infrared-camera-detector-mini-travel-hotel-hidden-camera-finder-anti-theft-dual-detection-modes-portable-led-flashlight-black (дата доступу 16.04.2025)

37. E-gadget Shop Online. URL: https://e-gadget.ua/antizhuchok_protect_k18/ (дата доступу 16.04.2025)

38. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special

Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

39. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

40. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

41. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

42. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

43. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

44. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС,

важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

45. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

46. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

47. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

48. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

49. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Книшук А.В. «Вступ до кібербезпеки»: навчальний посібник – Кропивницький: ЦНТУ – 2022. – 968 с.

50. Теорія та практика сучасного інформаційно-психологічного протиборства: навчальний посібник / [В.М. Петрик, С.О. Гнатюк, М.М. Присяжнюк та ін.]; за заг. ред. С.О. Гнатюка, В.М. Петрика та О.А Смірнова. – Полтава, 2022. – 334 с.

51. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

52. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

53. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

54. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

55. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook)

56. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

57. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

58. Смірнов О.А., Стасєв Ю.В., Бараннік В.В. Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Інформаційна безпека держави. Підручник – Кіровоград: РВЛ КНТУ, 2016. – 263 с

59. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

Додаток А

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....3
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....6
7 Перелік документів, що розробляються.....6
8 Етапи розробки.....6
9 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки автоматичного пошуку прихованих камер.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 57-02 від 17. 01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки автоматичного пошуку прихованих камер.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– аналіз існуючих систем пошуку прихованих камер;

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи кібербезпеки з веб-інтерфейсом та з користувачем;

– розробка програми, що реалізує спроєктовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки автоматичного пошуку прихованих камер;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів, операційної системи та використовуваного пристрою для доступу до інтерфейсу пристрою.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати архітектуру клієнт/сервер, системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 7/8/10/11, macOs, Linux, мобільними пристроями з ОС Android/iOs і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з веб-сторінкою, що працюють під управлінням ОС Windows 7/8/10/11, macOs, Linux, мобільними пристроями з ОС Android/iOs.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Arduino IDE.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Принципова схема пристрою – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 3 аркуша.
- Пояснювальна записка – 59 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, принципової схеми, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 25.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 02.06.2025 р.

Додаток Б

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за першим (бакалаврським) рівнем вищої освіти

_____ Дрєєв О. М.

Програмне забезпечення системи кібербезпеки автоматичного пошуку прихованих камер

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Sketch_maya4a.ino

```
#include <Arduino.h>
#include <AccelStepper.h>
#include <esp_camera.h>
#include <WiFi.h>
#include <WebServer.h>
#include <html.h>
#include <SPIFFS.h>
#include <JPEGDecoder.h>
#include <JPEGENC.h>

// Піни
#define LED_FLASH 4
#define LED_BUILDIN 33
const int stepper_h_enabled_pin = 15;
const int stepper_d_enabled_pin = 14;

// WiFi дані для доступу
#define WiFi_SSID "HiddenCameraDetector_Access" // SSID for the access point
#define WiFi_PASSWORD "camera123" // Password for the access point

#define IMAGE_WIDTH 320
#define IMAGE_HEIGHT 240
#define BRIGHTNESS_THRESHOLD 220      // Поріг загальної яскравості для
відблиску
#define COLOR_CHANNEL_THRESHOLD 180   // Мінімальна яскравість кожного каналу
#define MAX_GLARE_AREA 20000          // Максимальна площа для відблиску
#define MIN_GLARE_AREA 50             // Мінімальна площа для відблиску
#define MAX_ASPECT_RATIO 2.5          // Максимальне співвідношення сторін для
відблиску

// Ініціалізуємо змінну веб-серверу
WebServer webserver(80);

// Змінна для контролю зупинки при виявленні відблиску
bool isFlashDetected = false;

// Wifi конфігурація
IPAddress local_IP(192, 168, 4, 1); // статична айпі адреса
IPAddress gateway(192, 168, 4, 1);
IPAddress subnet(255, 255, 255, 0);

// Створення двигунів
AccelStepper stepper_h(AccelStepper::DRIVER, 13, 12);
AccelStepper stepper_d(AccelStepper::DRIVER, 0, 2);

// Обмеження руху
const int H_MIN = 0;
const int H_MAX = 100;       // ~180°
const int H_CENTER = 50;     // ~90°
const int H_STEP = 25;       // ~45°

const int D_FRONT_MIN = -8;  // ~-15°
const int D_FRONT_MAX = 25;  // ~45°
const int D_REAR_MIN = 75;   // ~135°
const int D_REAR_MAX = 108;  // ~195°
const int D_STEP = 8;        // ~15°

bool initialized = false;
bool scanning_down = true;
bool rear_scanning = false;
uint scanning_progress = 0;

int current_h = H_MAX;
int current_d = D_FRONT_MAX;

bool scan_in_progress = false;
```

```
camera_config_t config;

// Ввімкнення/вимкнення драйверів
void enable_stepper_h()
{
  digitalWrite(stepper_h_enabled_pin, LOW);
}
void disable_stepper_h()
{
 digitalWrite(stepper_h_enabled_pin, HIGH);
}
void enable_stepper_d()
{
  digitalWrite(stepper_d_enabled_pin, LOW);
}
void disable_stepper_d()
{
  digitalWrite(stepper_d_enabled_pin, HIGH);
}

// Спалах
void turnOnFlashing()
{
  digitalWrite(LED_FLASH, HIGH);
}

void turnOffFlashing()
{
  digitalWrite(LED_FLASH, LOW);
}

// Рух двигуна з автоматичним увімкненням
void moveStepper(AccelStepper &stepper, int position, int enable_pin)
{
  digitalWrite(enable_pin, LOW);
  stepper.moveTo(position);
}

void initCamera()
{
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = 5;
  config.pin_d1 = 18;
  config.pin_d2 = 19;
  config.pin_d3 = 21;
  config.pin_d4 = 36;
  config.pin_d5 = 39;
  config.pin_d6 = 34;
  config.pin_d7 = 35;
  config.pin_xclk = 0;
  config.pin_pclk = 22;
  config.pin_vsync = 25;
  config.pin_href = 23;
  config.pin_sscb_sda = 26;
  config.pin_sscb_scl = 27;
  config.pin_pwdn = 32;
  config.pin_reset = -1;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_RGB565; //PIXFORMAT_RGB565 | PIXFORMAT_JPEG

  // QVGA = 320x240
  config.frame_size = FRAMESIZE_QVGA;
  config.jpeg_quality = 10;
  config.fb_count = 1;

  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK)
```

```
  {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  Serial.println("Camera init success");
}

bool capturePhotoSaveSpiffs()
{
  turnOnFlashing();
  delay(200);
  camera_fb_t * fb = esp_camera_fb_get();
  delay(100);
  turnOffFlashing();
  if (!fb)
  {
    Serial.println("Camera capture failed");
    return false;
  }

  File file = SPIFFS.open("/last.jpg", FILE_WRITE);
  if (!file)
  {
    Serial.println("Failed to open file in writing mode");
    esp_camera_fb_return(fb);
    return false;
  }

  file.write(fb->buf, fb->len);
  file.close();
  esp_camera_fb_return(fb);

  Serial.println("Photo saved to /last.jpg");
  return true;
}

// Функція конвертації RGB565 формату в JPEG
bool rgb565_to_jpeg(uint8_t *rgb565_data, size_t rgb565_len, int width, int
height, const char* filename)
{
    uint8_t *jpg_buf = NULL;
    size_t jpg_len = 0;

    // Перевіряємо вхідні параметри
    if (!rgb565_data || rgb565_len == 0 || width == 0 || height == 0 ||
!filename)
    {
        Serial.println("Помилка конвертації: некоректні параметри");
        return false;
    }

    // fmt2jpg — функція з esp_camera.h, яка кодує RGB565 у JPEG
    bool ok = fmt2jpg(rgb565_data, rgb565_len, width, height, PIXFORMAT_RGB565,
80, &jpg_buf, &jpg_len);

    if (!ok || jpg_buf == NULL || jpg_len == 0)
    {
        Serial.println("Помилка конвертації RGB565 в JPEG");
        if (jpg_buf) free(jpg_buf);
        return false;
    }

    // Перевіряємо, чи є файлова система
    if (!SPIFFS.begin(true))
    {
        Serial.println("Помилка монтування SPIFFS");
        free(jpg_buf);
        return false;
```

```
    }

    File file = SPIFFS.open(filename, FILE_WRITE);
    if (!file)
    {
        Serial.println("Не вдалося відкрити файл для запису");
        free(jpg_buf);
        return false;
    }

    // Записуємо з перевіркою помилок
    size_t written = file.write(jpg_buf, jpg_len);
    file.close();
    free(jpg_buf);

    if (written != jpg_len)
    {
        Serial.println("Помилка запису: записано не всі дані");
        return false;
    }

    Serial.print("Зображення успішно збережено як JPEG. Розмір: ");
    Serial.println(jpg_len);
    return true;
}

// Функція для обведення участків з виявленим спалахом
void draw_rectangle_rgb565(uint8_t* buf, int width, int height, int x0, int y0,
int x1, int y1)
{
  // yellow color - maximum brigthness
  uint16_t color = 0xFFE0; // yell (0xFFE0 = red + green)

  // Перевірка кордонів зони
  x0 = max(0, min(width-1, x0));
  y0 = max(0, min(height-1, y0));
  x1 = max(0, min(width-1, x1));
  y1 = max(0, min(height-1, y1));

  // Малювання горизонтальних ліній
  for (int x = x0; x <= x1; x++)
  {
    // Верхня горизонталь
    uint16_t* pixel = (uint16_t*)(buf + ((y0 * width + x) * 2));
    *pixel = color;

    // Нижня горизонталь
    pixel = (uint16_t*)(buf + ((y1 * width + x) * 2));
    *pixel = color;
  }

  // Малювання вертикальних ліній
  for (int y = y0; y <= y1; y++)
  {
    // Ліва вертикаль
    uint16_t* pixel = (uint16_t*)(buf + ((y * width + x0) * 2));
    *pixel = color;

    // Права вертикаль
    pixel = (uint16_t*)(buf + ((y * width + x1) * 2));
    *pixel = color;
  }
}

bool detectFlash(camera_fb_t* fb)
{
  int width = fb->width;
  int height = fb->height;
  uint16_t* buf = (uint16_t*)fb->buf;
```

```
  // Максимальна кількість відблисків для виявлення
  const int MAX_FLASHES = 3; // 3 для економії

  // Масиви для зберігання координат відблисків
  int min_x[MAX_FLASHES], min_y[MAX_FLASHES], max_x[MAX_FLASHES],
max_y[MAX_FLASHES];
  bool flash_active[MAX_FLASHES];
  int flash_count = 0;

  // Ініціалізація масивів
  for (int i = 0; i < MAX_FLASHES; i++)
  {
    min_x[i] = width;
    min_y[i] = height;
    max_x[i] = 0;
    max_y[i] = 0;
    flash_active[i] = false;
  }

  // Виділення пам'яті динамічним методом
  uint8_t* processed = (uint8_t*)malloc(width * height);
  if (!processed)
  {
    Serial.println("Помилка виділення пам'яті для processed");
    return false; // Не вдалося виділити пам'ять
  }
  memset(processed, 0, width * height);

  // Пошук яскравих пікселів та поєднання їх за відстаню
  for (int y = 0; y < height; y++)
  {
    for (int x = 0; x < width; x++)
    {
      // Піксель перевірений?
      if (processed[y * width + x]) continue;

      uint16_t pixel = buf[y * width + x];

      // Отримуємо компоненти RGB
      uint8_t r = (pixel >> 11) & 0x1F; // 5 біт для червоного
      uint8_t g = (pixel >> 5) & 0x3F;  // 6 біт для зеленого
      uint8_t b = pixel & 0x1F;         // 5 біт для синього

      // Нормалізація компонентів RGB від 0 до 255
      r = (r * 255) / 31;
      g = (g * 255) / 63;
      b = (b * 255) / 31;

      // Оцінювання яскравості пікселя
      int brightness = (r * 0.299 + g * 0.587 + b * 0.114);

      // Перевірка чи піксель достатньо яскравий
      if (brightness > BRIGHTNESS_THRESHOLD &&
          r > COLOR_CHANNEL_THRESHOLD &&
          g > COLOR_CHANNEL_THRESHOLD &&
          b > COLOR_CHANNEL_THRESHOLD)
      {

        // Мітка на піксель
        processed[y * width + x] = 1;

        // Перевірка або додавання пікселя
        bool added_to_existing = false;
        for (int i = 0; i < flash_count; i++)
        {
          if (!flash_active[i]) continue;

          // Якщо піксель близько до іншого такого ж, об єднати їх
```

```
          if (x >= min_x[i] - 10 && x <= max_x[i] + 10 &&
              y >= min_y[i] - 10 && y <= max_y[i] + 10)
          {
            min_x[i] = min(min_x[i], x);
            min_y[i] = min(min_y[i], y);
            max_x[i] = max(max_x[i], x);
            max_y[i] = max(max_y[i], y);
            added_to_existing = true;
            break;
          }
        }

        // Якщо ні - створити нову групу
        if (!added_to_existing && flash_count < MAX_FLASHES)
        {
          min_x[flash_count] = x;
          min_y[flash_count] = y;
          max_x[flash_count] = x;
          max_y[flash_count] = y;
          flash_active[flash_count] = true;
          flash_count++;
        }
      }
    }
  }

  // Вивільнення пам яті
  free(processed);

  // Перевірка всіх відблисків
  bool any_valid_flash = false;

  for (int i = 0; i < flash_count; i++)
  {
    if (flash_active[i])
    {
      int area = (max_x[i] - min_x[i] + 1) * (max_y[i] - min_y[i] + 1);
      float aspect_ratio = float(max(max_x[i] - min_x[i] + 1, max_y[i] -
min_y[i] + 1)) /
                           float(min(max_x[i] - min_x[i] + 1, max_y[i] - min_y[i]
+ 1));

      // Перевірка чи існує відблиск
      if (area > MAX_GLARE_AREA || area < MIN_GLARE_AREA || aspect_ratio >
MAX_ASPECT_RATIO)
      {
        flash_active[i] = false;
        Serial.print("Відблиск #"); Serial.print(i+1);
        Serial.println(" відкинуто через розмір/форму");
        Serial.print("Площа: "); Serial.print(area);
        Serial.print(", Співвідношення сторін: "); Serial.println(aspect_ratio);
      }
      else
      {
        any_valid_flash = true;
        Serial.print("Відблиск #"); Serial.print(i+1);
        Serial.println(" підтверджено!");
        Serial.print("Координати: (");
        Serial.print(min_x[i]); Serial.print(","); Serial.print(min_y[i]);
Serial.print(") - (");
        Serial.print(max_x[i]); Serial.print(","); Serial.print(max_y[i]);
Serial.println(")");
        Serial.print("Площа: "); Serial.print(area);
        Serial.print(", Співвідношення сторін: "); Serial.println(aspect_ratio);

        // Виділяємо зону, яку слід обвести
        int draw_min_x = max(0, min_x[i] - 5);
        int draw_min_y = max(0, min_y[i] - 5);
        int draw_max_x = min(width - 1, max_x[i] + 5);
```

```
        int draw_max_y = min(height - 1, max_y[i] + 5);

        // Обведенням жовтим прямокутником зони з підозрілим відблиском
        draw_rectangle_rgb565(fb->buf, width, height, draw_min_x, draw_min_y,
draw_max_x, draw_max_y);
      }
    }
  }

  // Зберігаємо фото в JPEG форматі
  bool saved = rgb565_to_jpeg(fb->buf, fb->len, width, height, "/last.jpg");
  if (!saved)
  {
    Serial.println("Помилка збереження зображення");
  }

  if (!any_valid_flash)
  {
    Serial.println("Відблисків не знайдено або всі потенційні відблиски
відкинуто");
    return false;
  }

  return true;
}

void photoHandler()
{
  Serial.println("Початок фотографування...");

  // Перевіряємо доступну пам'ять перед обробкою
  Serial.print("Вільна пам'ять перед обробкою: ");
  Serial.println(ESP.getFreeHeap());

  // Перезапускаємо камеру для надійності
  /*esp_camera_deinit();
  delay(100);
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK)
  {
    Serial.printf("Помилка ініціалізації камери: 0x%x\n", err);
    return;
  }*/

  turnOnFlashing();
  delay(250);
  camera_fb_t* fb = esp_camera_fb_get();
  delay(200);
  turnOffFlashing();

  if (!fb)
  {
    Serial.println("Помилка захоплення зображення");
    return;
  }

  // Перевіряємо розмір отриманого буфера
  if (fb->len == 0 || fb->width == 0 || fb->height == 0)
  {
    Serial.println("Отримано некоректне зображення");
    esp_camera_fb_return(fb);
    return;
  }

  Serial.println("Фото зроблено, аналізую на наявність відблисків...");
  Serial.print("Розмір буфера: "); Serial.println(fb->len);
  Serial.print("Розміри зображення: "); Serial.print(fb->width);
Serial.print("x"); Serial.println(fb->height);
```

```
  // Спроба аналізу зображення на наявність відблиску
  bool flash_found = false;

  // Використовуємо try-catch-подібний підхід з обробкою помилок
  // ESP32 не підтримує справжній try-catch, тому робимо це через повернення
помилок
  flash_found = detectFlash(fb);

  // Виводимо результат
  if (flash_found)
  {
    Serial.println("УВАГА! Виявлено потенційну приховану камеру!");
    Serial.println("Перевірте фото для детальної інформації");
    digitalWrite(LED_BUILDIN, LOW); // Блимаємо світлодіодом як сигнал тривоги

    // Встановлюємо глобальну змінну для зупинки сканування
    isFlashDetected = true;
    Serial.println("Сканування призупинено. Очікуємо підтвердження від
користувача.");

    // Блимаємо світлодіодом декілька разів (але коротше, щоб не затримувати
обробку)
    for (int i = 0; i < 2; i++)
    {
      digitalWrite(LED_BUILDIN, HIGH);
      delay(50);
      digitalWrite(LED_BUILDIN, LOW);
      delay(50);
    }
  } else {
    Serial.println("Камеру не виявлено");
    digitalWrite(LED_BUILDIN, HIGH); // Вимкнути світлодіод
  }

  // Звільняємо буфер фрейму
  esp_camera_fb_return(fb);

  // Перевіряємо пам'ять після обробки
  Serial.print("Вільна пам'ять після обробки: ");
  Serial.println(ESP.getFreeHeap());

  Serial.println("Аналіз фото завершено");
}

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

  stepper_h.setMaxSpeed(100.0);
  stepper_h.setAcceleration(50.0);

  stepper_d.setMaxSpeed(50.0);
  stepper_d.setAcceleration(50.0);

  pinMode(LED_FLASH, OUTPUT);
  digitalWrite(LED_FLASH, LOW);
  pinMode(LED_BUILDIN, OUTPUT);
  digitalWrite(LED_BUILDIN, HIGH);

  pinMode(stepper_h_enabled_pin, OUTPUT);
  disable_stepper_h();
  pinMode(stepper_d_enabled_pin, OUTPUT);
  disable_stepper_d();

  Serial.println("Stepper drivers initialized.");
  //Flashing(10);

  // Initialize SPIFFS
  if (!SPIFFS.begin(true))
  {
```

```
    Serial.println("An error occurred while mounting SPIFFS");
    return;
  }

// Initialize camera
Serial.println("Camera initialization started...");
initCamera();

// Initialize WiFi access point
WiFi.softAPConfig(local_IP, gateway, subnet);
WiFi.softAP(WiFi_SSID, WiFi_PASSWORD);

Serial.println("AP is configured...");

Serial.print("Waiting for client connection to AP...");

while (WiFi.softAPgetStationNum() == 0)
{
  delay(500);
  Serial.print(".");
}

Serial.println("\nClient connected to AP!");

// Route webserver pages
webserver.on("/", HTTP_GET, []()
{
  webserver.send_P(200, "text/html", index_page);
});

webserver.on("/start-scan", HTTP_GET, []()
{
  if (!scan_in_progress) {
    scan_in_progress = true;
    initialized = false; // щоб переініціалізувалося
    webserver.send(200, "text/plain", "Scanning started");
  } else {
    webserver.send(200, "text/plain", "Already scanning");
  }
});

webserver.on("/photo", HTTP_GET, []()
{
  File file = SPIFFS.open("/last.jpg", "r");
  if (!file || file.isDirectory()) {
    webserver.send(404, "text/plain", "Photo not found");
    return;
  }

  webserver.streamFile(file, "image/jpeg");
  file.close();
});

webserver.on("/status", HTTP_GET, []()
{
  if (scan_in_progress) {
    webserver.send(200, "text/plain", "true");
  } else {
    webserver.send(200, "text/plain", "false");
  }
});

webserver.on("/progress", HTTP_GET, []()
{
  webserver.send_P(200, "text/plain", String(scanning_progress).c_str());
});

webserver.on("/reset-flash-detection", HTTP_GET, []()
{
```

```
    isFlashDetected = false;
    webserver.send(200, "text/plain", "Сканування відновлено");
    Serial.println("Відновлено сканування після виявлення відблиску");
  });

  // Маршрут для продовження після підтвердження
  webserver.on("/continue-scan", HTTP_GET, []()
  {
    if (isFlashDetected)
    {
      isFlashDetected = false;
      webserver.send(200, "text/plain", "Сканування продовжено");
      Serial.println("Продовжуємо сканування після підтвердження виявлення");
    } else {
      webserver.send(200, "text/plain", "Сканування не було зупинено");
    }
  });

  // Маршрут для перевірки, чи виявлено відблиск
  webserver.on("/is-flash-detected", HTTP_GET, []()
  {
    webserver.send(200, "text/plain", isFlashDetected ? "true" : "false");
  });

  // Webserver configuration
  webserver.begin();

  Serial.println("WebServer is ON...");

  // End of the setup
  Serial.println("Setup complete.");
}

void loop()
{
  webserver.handleClient();

  if (!scan_in_progress)
  {
    Serial.println("Очікуємо на користувачу для старту...");
    delay(500);
    return;
  }

  // Перевіряємо, чи було виявлено відблиск
  if (isFlashDetected)
  {
    static unsigned long lastBlink = 0;
    static unsigned long lastBeep = 0;
    unsigned long currentMillis = millis();

    if (currentMillis - lastBlink > 300) { // Швидше блимаємо для привертання
уваги
      lastBlink = currentMillis;
      digitalWrite(LED_BUILDIN, !digitalRead(LED_BUILDIN));
    }

    delay(10); // Короткий delay для кращої відповіді веб-сервера
    return;
  }

  if (!initialized)
  {
    Serial.println("Ініціалізація: рухаюсь до початкових позицій...");
    moveStepper(stepper_h, H_MAX, stepper_h_enabled_pin);
    moveStepper(stepper_d, D_FRONT_MAX, stepper_d_enabled_pin);

    while (stepper_h.distanceToGo() != 0 || stepper_d.distanceToGo() != 0)
    {
```

```
      stepper_h.run();
      stepper_d.run();
    }

    disable_stepper_h();
    disable_stepper_d();

    current_h = H_MAX;
    current_d = D_FRONT_MAX;
    initialized = true;
    Serial.println("Ініціалізацію завершено.");

    // Додаємо очищення пам'яті і перезавантаження системи на початку сканування
    Serial.print("Очищення пам'яті. Доступно: ");
    Serial.println(ESP.getFreeHeap());
    return;
  }

  stepper_h.run();
  stepper_d.run();

  if (stepper_h.distanceToGo() == 0 && stepper_d.distanceToGo() == 0)
  {
    disable_stepper_h();
    disable_stepper_d();

    Serial.print("Позиція досягла: H=");
    Serial.print(current_h);
    Serial.print(" D=");
    Serial.println(current_d);

    Serial.println("Фотографування фото...");
    photoHandler();
    scanning_progress += 1;
    delay(400); // Зменшуємо затримку для пришвидшення сканування

    // Додаємо очищення пам'яті після кожної фотографії
    if (scanning_progress % 5 == 0)
    { // Кожні 5 фото
      Serial.print("Проміжне очищення пам'яті. Доступно: ");
      Serial.println(ESP.getFreeHeap());
    }

    if (scanning_down)
    {
      current_d -= D_STEP;
      if ((rear_scanning && current_d < D_REAR_MIN) || (!rear_scanning &&
current_d < D_FRONT_MIN))
      {
        // Коли досягли нижньої межі, повертаємося на початкову позицію
        current_d = rear_scanning ? D_REAR_MAX : D_FRONT_MAX;
        moveStepper(stepper_d, current_d, stepper_d_enabled_pin);

        // Змінюємо горизонтальну позицію
        current_h -= H_STEP;
        if (current_h < H_MIN)
        {
          if (!rear_scanning)
          {
            Serial.println("Перемикання на заднє сканування...");
            rear_scanning = true;
            current_h = H_MAX;
            current_d = D_REAR_MAX;
          }
          else
          {
            Serial.println("СКАНУВАННЯ ЗАВЕРШЕНО. Повертання на стартову
позицію...");
            moveStepper(stepper_h, H_CENTER, stepper_h_enabled_pin);
```

```
            moveStepper(stepper_d, D_FRONT_MAX, stepper_d_enabled_pin);
            while (stepper_h.isRunning() || stepper_d.isRunning())
            {
              stepper_h.run();
              stepper_d.run();
            }
            disable_stepper_h();
            disable_stepper_d();

            // Скидаємо все
            scan_in_progress = false;
            rear_scanning = false;
            initialized = false;
            scanning_progress = 0;
            Serial.println("ЗАВЕРШЕНО! Очікування на дії користувача...");
            return;
          }
        }
        moveStepper(stepper_h, current_h, stepper_h_enabled_pin);
      }
    }

    moveStepper(stepper_d, current_d, stepper_d_enabled_pin);
  }
}
```

html.h

```
const char index_page[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hidden Camera Detector</title>
    <style>
      * {
        box-sizing: border-box;
        margin: 0;
        padding: 0;
      }

      body {
        font-family: Arial, sans-serif;
        background-color: #f5f7fa;
        color: #333;
        height: 100vh;
        display: flex;
        flex-direction: row;
      }

      /* Sidebar */
      .sidebar {
        width: 30%;
        background-color: #2c3e50;
        color: white;
        padding: 20px;
        display: flex;
        flex-direction: column;
        justify-content: flex-start;
        gap: 20px;
      }

      .sidebar-header {
        display: flex;
        align-items: center;
        gap: 15px;
        padding: 20px;
        border-bottom: 2px solid #1a252f;
        margin: -20px -20px 20px -20px;
        background-color: #1a252f;
        box-shadow: 0 2px 5px rgba(0,0,0,0.3);
      }

      .sidebar-header svg {
        width: 50px;
        height: 50px;
        filter: drop-shadow(0 2px 3px rgba(0,0,0,0.3));
      }

      .sidebar h1 {
        font-size: 1.8em;
        margin: 0;
        color: #ffffff;
        text-shadow: 1px 1px 2px rgba(0,0,0,0.3);
        font-weight: 600;
      }

      .button {
        padding: 12px 16px;
        background-color: #3498db;
        color: white;
        border: none;
        border-radius: 6px;
        font-size: 1em;
        cursor: pointer;
```

```
  transition: background-color 0.3s ease;
}

.button:hover {
  background-color: #2980b9;
}

.button:disabled {
  background-color: #95a5a6;
  cursor: not-allowed;
  opacity: 0.7;
}

.status-container {
  margin-top: 20px;
}

.status-label {
  font-size: 1.1em;
  margin-bottom: 8px;
}

.status-indicator {
  width: 20px;
  height: 20px;
  border-radius: 50%;
  display: inline-block;
  vertical-align: middle;
  margin-left: 10px;
  background-color: green; /* Default: free */
}

/* Main content */
.main {
  width: 70%;
  padding: 20px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: flex-start;
  height: 100vh;
  overflow: hidden;
}

.main p {
  font-size: 1.1em;
  margin-bottom: 20px;
  text-align: center;
}

#photo {
  width: 100%;
  height: 70vh;
  object-fit: contain;
  border: 2px solid #ccc;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0,0,0,0.1);
  background-color: #000;
}

#statusIndicator {
    width: 20px;
    height: 20px;
    border-radius: 50%;
    background-color: gray;
    transition: background-color 0.3s ease;
  }

  /* Анімація миготіння */
```

```
@keyframes pulse {
  0%   { background-color: red; }
  50%  { background-color: darkred; }
  100% { background-color: red; }
}

.blinking {
  animation: pulse 1s infinite;
}

.progress-bar {
  width: 100%;
  height: 20px;
  background-color: #bdc3c7;
  border-radius: 10px;
  overflow: hidden;
  margin-top: 10px;
}

#progressFill {
  height: 100%;
  width: 0%;
  background-color: #27ae60;
  transition: width 0.5s ease;
}

#progressText {
  font-size: 0.9em;
  margin-top: 5px;
  display: inline-block;
}

.camera-detected {
  color: #e74c3c !important;
  font-weight: bold !important;
  text-transform: uppercase;
  animation: blink-text 1s infinite;
}

@keyframes blink-text {
  0% { opacity: 1; }
  50% { opacity: 0.5; }
  100% { opacity: 1; }
}

.highlight-button {
  background-color: #e74c3c !important;
  animation: pulse-button 1.5s infinite;
}

@keyframes pulse-button {
  0% { transform: scale(1); }
  50% { transform: scale(1.05); }
  100% { transform: scale(1); }
}

/* Адаптивні стилі */
@media screen and (max-width: 1024px) {
  body {
    flex-direction: column;
  }

  .sidebar {
    width: 100%;
    height: auto;
  }

  .main {
    width: 100%;
```

```
            height: auto;
            padding: 10px;
          }

          #photo {
            height: 50vh;
          }
        }

        @media screen and (max-width: 768px) {
          .sidebar-header {
            padding: 15px;
          }

          .sidebar-header svg {
            width: 40px;
            height: 40px;
          }

          .sidebar h1 {
            font-size: 1.5em;
          }

          #photo {
            height: 40vh;
          }
        }

        @media screen and (max-width: 480px) {
          .sidebar {
            padding: 10px;
          }

          .button {
            padding: 10px 14px;
            font-size: 0.9em;
          }

          #photo {
            height: 30vh;
          }
        }

    </style>
  </head>
  <body>
    <div class="sidebar">
      <div class="sidebar-header">
        <svg viewBox="0 0 120 120" fill="none"
xmlns="http://www.w3.org/2000/svg">
          <ellipse cx="60" cy="60" rx="50" ry="30" fill="#e0e0e0" stroke="#333"
stroke-width="4"/>
          <circle cx="60" cy="60" r="18" fill="#fff" stroke="#333" stroke-
width="4"/>
          <circle cx="60" cy="60" r="8" fill="#1976d2"/>
          <circle cx="65" cy="55" r="3" fill="#fff" opacity="0.7"/>
          <rect x="80" y="80" width="20" height="6" rx="3" fill="#1976d2"
transform="rotate(45 80 80)"/>
        </svg>
        <h1>Camera Tools</h1>
      </div>
      <button class="button" id="startScanButton"
onclick="startScan()">Розпочати сканування</button>
      <button class="button" id="continueButton" onclick="continueScan()"
disabled>Продовжити сканування</button>
      <button class="button" id="refreshButton"
onclick="refreshImage()">Зупинити сканування</button>

      <div class="status-container">
```

```
      <div id="statusIndicator"></div>
      <span id="statusText">Перевіряю статус...</span>
      <div class="progress-bar">
        <div id="progressFill"></div>
      </div>
    <span id="progressText">0%</span>

  </div>

</div>

<div class="main">
  <p>Можлива прихована камера розміщена: :</p>
  <img id="photo" src="/photo" alt="Latest photo">
</div>

<script>
  // Авто-оновлення зображення кожну 1 секунду
  setInterval(() => {
      const img = document.getElementById("photo");
      img.src = "/photo?" + new Date().getTime(); // кешобхідник
    }, 1000); // кожну 1 секунд

  // Змінні для стану застосунку
  let isScanning = false;
  let isFlashDetected = false;

  // Функція для оновлення стану кнопок
  function updateButtonsState() {
    const startButton = document.getElementById("startScanButton");
    const continueButton = document.getElementById("continueButton");

    if (isFlashDetected) {
      // Якщо виявлено відблиск - можна продовжити сканування
      startButton.disabled = true;
      continueButton.disabled = false;
    } else if (isScanning) {
      // Якщо сканування в процесі - обидві кнопки неактивні
      startButton.disabled = true;
      continueButton.disabled = true;
    } else {
      // Якщо сканування не активне - можна почати сканування
      startButton.disabled = false;
      continueButton.disabled = true;
    }
  }

  function startScan() {
    fetch('/start-scan')
      .then(res => res.text())
      .then(msg => {
        alert(msg);
        isScanning = true;
        updateButtonsState();
      });
  }

  function continueScan() {
    fetch('/continue-scan')
      .then(res => res.text())
      .then(msg => {
        alert(msg);
        isFlashDetected = false;
        updateButtonsState();
      });
  }

  function refreshImage() {
    const img = document.getElementById("photo");
```

```
        img.src = "/photo?" + new Date().getTime();
      }

      function updateStatus() {
        // Перевіряємо статус сканування
        fetch("/status")
          .then(response => response.text())
          .then(status => {
            const indicator = document.getElementById("statusIndicator");
            const statusText = document.getElementById("statusText");

            isScanning = status.trim() === "true";

            if (isScanning) {
              indicator.classList.add("blinking");
              statusText.innerText = "Система в процесі сканування простору...";
            } else {
              indicator.classList.remove("blinking");
              indicator.style.backgroundColor = "green";
              statusText.innerText = "Система очікує команди користувача...";
            }

            // Після оновлення статусу сканування, перевіряємо статус виявлення
            checkFlashDetected();
          })
          .catch(err => {
            console.error("Status fetch error:", err);
          });
      }

      function checkFlashDetected() {
        fetch("/is-flash-detected")
          .then(response => response.text())
          .then(status => {
            isFlashDetected = status.trim() === "true";
            const statusText = document.getElementById("statusText");
            const continueButton = document.getElementById("continueButton");

            // Оновлюємо текст статусу, якщо виявлено відблиск
            if (isFlashDetected) {
              statusText.innerText = "МОЖЛИВО ВИЯВЛЕНО КАМЕРУ! Перевірте
зображення.";
              statusText.classList.add("camera-detected");
              continueButton.classList.add("highlight-button");
            } else {
              statusText.classList.remove("camera-detected");
              continueButton.classList.remove("highlight-button");
            }

            updateButtonsState();
          })
          .catch(err => {
            console.error("Flash detection status fetch error:", err);
          });
      }

      function updateProgress() {
        fetch("/progress")
          .then(response => response.text())
          .then(value => {
            const percent = parseInt(value);
            const fill = document.getElementById("progressFill");
            const text = document.getElementById("progressText");

            if (!isNaN(percent)) {
              fill.style.width = percent + "%";
              text.innerText = percent + "%";
            }
          })
```

```
            .catch(err => console.error("Progress fetch error:", err));
        }

    setInterval(updateStatus, 2000);
    setInterval(updateProgress, 5000);
    window.onload = function() {
      updateStatus();
      updateProgress();
    };
      </script>
    </body>
</html>
)rawliteral";
```