Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

(тема)
Розроблення мобільної системи відеонагляду на базі ESP32 CAM

Виконав:
здобувач 2 року навчання,
групи КІТПВм-23-2
Лихо Т.А.
(прізвище, ініціали)

Спеціальність
КІТПВ
освітньої програми 174 Автоматизація,
комп'ютерно-інтеровані технології та робототехніка
(код і повна назва напряму)

Тип програми освітньо-професійна
(повна назва освітньої програми)

Керівник доц. Максимова С.С.
(посада, прізвище, ініціали)

Допускається до захисту
зав. кафедри _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет	Автоматики і комп’ютеризованих технологій
Кафедра	Комп’ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти	другий (магістерський)
Спеціальність	Комп’ютерно-інтегровані технологічні процеси та виробництва
Тип програми	освітньо-професійна
Освітня програма	174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві *Лиху Тимуру Андрійовичу*
(прізвище, ім’я, по батькові)

1. Тема роботи *Розроблення мобільної системи відеонагляду на базі ESP32 CAM*

затверджена наказом по університету від 22.11. 2024 р. № 1231 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13.01. 2025 р.

3. Вихідні дані до роботи *Тип робота - наземний, триколісний. Система керування - дистанційна. Набір виконуваних дій - пересування в усі сторони, зупинка, передача відеопотоку. Мережа - Wi-Fi. Програмний метод - MQTT-протокол.*

4. Перелік питань, що потрібно опрацювати в роботі *4.1 Вступ, 4.2 Огляд та аналіз сучасних мобільних систем відеонагляду, 4.3 Розроблення мобільної системи відеоспостереження, 4.4 Експериментальна частина, 4.5 Охорона праці, 4.6 Загальні висновки.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 6 с. формату А4*

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Керівник (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання завдання на кваліфікаційну роботу*	*25.11.2024*	*Виконано*
2	*Аналіз літератури та супутніх джерел*	*27.11.2024*	*Виконано*
3	*Огляд та аналіз сучасних мобільних систем відеонагляду*	*30.11.2024*	*Виконано*
4	*Розроблення мобільної системи відеоспостереження*	*05.12.2024*	*Виконано*
5	*Планування та проведення експериментів*	*15.12.2024*	*Виконано*
6	*Аналіз результатів проведеної роботи*	*17.12.2024*	*Виконано*
7	*Оформлення пояснювальної записки*	*25.12.2024*	*Виконано*
8	*Оформлення графічної частини*	*26.12.2024*	*Виконано*
9	*Підготовка презентації*	*27.12.2024*	*Виконано*
10	*Подання роботи на рецензування*	*30.12.2024*	*Виконано*
11	*Попередній захист*	*07.01.2025*	*Виконано*
12	*Подання роботи до екзаменаційної комісії*	*13.01.2025*	*Виконано*

Дата видачі завдання *25.11.2024*

Здобувач _____ *Лихо Т.А.*
(підпис) (прізвище, ініціали)

Керівник роботи _____ *доц. Максимова С.С.*
(підпис) (посада, прізвище, ініціали)

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

"13" січня 2025 р.  Лихо Т.А.

РЕФЕРАТ

Пояснювальна записка: 105 с., 8 табл., 31 рис., 8 додатків, 41 джерел інформації.

МОБІЛЬНИЙ РОБОТ, РОБОТОТЕХНІКА, СИСТЕМА ВІДЕОНАГЛЯДУ, ESP32-CAM, ВЕБ-ЗАСТОСУНОК, ДИСТАНЦІЙНЕ КЕРУВАННЯ, PYTHON, ARDUINO.

Об’єкт дослідження – процес відеонагляду за виробничими процесами на приладобудівних підприємствах.

Предмет дослідження – система відеонагляду за перебігом виробничих процесів на приладобудівних підприємствах.

Мета дослідження – підвищення ефективності відеонагляду за перебігом виробничих процесів на приладобудівних підприємствах шляхом розроблення мобільної системи відеонагляду на базі ESP32-CAM.

Методи досліджень – методи об’єктно-орієнтованого програмування мовою Arduino (C/C++) та Python, системний підхід, експериментальне тестування системи у реальних умовах.

У роботі було проведено аналіз існуючих конструкторських та програмних рішень побудови роботизованих систем відеонагляду.

На базі проведеного дослідження розроблено прототип мобільного робота з камерою та відповідне програмне забезпечення для передачі відеопотоку. А також, розроблено користувацький інтерфейс для керування мобільним роботом.

Отримані результати можуть бути використані для впровадження роботизованих охоронних систем, або промислових та побутових роботизованих систем відеонагляду.

Також, отримані результати відносяться до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», а саме пункту 9.4 «Сприяти

прискореному розвитку високо- та середньовисокотехнологічних секторів переробної промисловості, які формуються на основі використання ланцюгів «освіта – наука – виробництво» та кластерного підходу за напрямами: розвиток інноваційної екосистеми; розвиток інформаційно-телекомунікаційних технологій (ІКТ); застосування ІКТ в АПК, енергетиці, транспорті та промисловості; високотехнологічне машинобудування; створення нових матеріалів; розвиток фармацевтичної та біоінженерної галузей».

ABSTRACT

Explanatory note: 105 pp., 8 tables, 31 figures, 8 appendices, 41 sources of information.

MOBILE ROBOT, ROBOTICS, SURVEILLANCE SYSTEM, ESP32-CAM, WEB APPLICATION, REMOTE CONTROL, PYTHON, ARDUINO.

The object of the study is the process of video surveillance of production processes at instrument-making enterprises.

The subject of the study is the video surveillance system monitoring the course of production processes at instrument-making enterprises.

The aim of the study is to improve the efficiency of video surveillance of production processes at instrument-making enterprises by developing a mobile video surveillance system based on ESP32-CAM.The research methods included object-oriented programming using Arduino (C/C++) and Python, a systems approach, and experimental testing of the system in real-world conditions.

The study analyzed existing hardware and software solutions for building robotic video surveillance systems. Based on the research, a prototype of a mobile robot with a camera was developed, along with software for video stream transmission. A user interface was also created to control the mobile robot.

The obtained results can be applied to the implementation of robotic security systems, as well as industrial and household robotic video surveillance systems.

Furthermore, these results align with Sustainable Development Goal 9: «Industry, Innovation, and Infrastructure», specifically target 9.4 «Promote the accelerated development of high- and medium-high-tech sectors of the manufacturing industry, which are formed based on the use of the 'education–science–production' value chains and the cluster approach in the following areas: development of an innovation ecosystem; advancement of information and

communication technologies (ICT); application of ICT in agriculture, energy, transportation, and industry; high-tech mechanical engineering; creation of new materials; development of the pharmaceutical and bioengineering sectors».

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АКБ – акумуляторна батарея;

АМР – автономний мобільний робот;

АСК – автоматизована система керування;

ДС – дистанційне керування;

МР – мобільний робот;

МРВ – мобільний робот відеонагляду;

МСВ – мобільна система відеонагляду;

ПЗ – програмне забезпечення;

ШНП – штатит нахилу та панорамування;

IoT – Internet of Things;

MQTT – message queuing telemetry transport.

ВСТУП

У сучасному світі однією з найважливіших потреб суспільства є безпека. Розвиток технологій провокує зростання необхідності у створенні ефективних охоронних систем, у тому числі систем відеонагляду, які можуть забезпечити високий рівень безпеки у приватних будинках, складах, виробництвах та громадських місцях. Традиційні системи відеоспостереження часто є дорогими та складними у встановленні та обслуговування, саме тому виникає потреба у розробці більш ефективних, доступних та простих у використанні рішень. Одним з таких рішень є мобільна система відеонагляду на базі мікроконтролеру ESP32-CAM.

Розроблення мобільної системи відеонагляду на базі ESP32-CAM є актуальним завданням, оскільки цей модуль поєднує в собі високу функціональність та доступність. ESP32-CAM має вбудовану камеру та підтримку Wi-Fi, що дозволяє створювати компактні та мобільні системи відеоспостереження. Така система може бути використана для моніторингу будинків, офісів, громадських місць та інших об’єктів, де необхідний постійний контроль.

Об’єктом дослідження є процес відеонагляду за виробничими процесами на приладобудівних підприємствах.

Предметом дослідження є система відеонагляду за перебігом виробничих процесів на приладобудівних підприємствах.

Метою роботи є підвищення ефективності відеонагляду за перебігом виробничих процесів на приладобудівних підприємствах шляхом розроблення мобільної системи відеонагляду на базі ESP32-CAM.

Для виконання поставленої мети необхідно:

– провести аналіз вже існуючих рішень мобільних систем відеонагляду;

– провести підбір необхідних компонентів для побудови мобільного робота;

– провести аналіз мов програмування та технологій, що можуть бути використані для реалізації поставленої мети;

– налагодити програмне забезпечення для передачі відеопотоку з камери мобільної платформи у користувацький інтерфейс;

– налагодити програмне забезпечення для дистанційного керування мобільною платформою, використовуючи мережу Інтернет.

Методи досліджень – методи об’єктно-орієнтованого програмування мовою Arduino (C/C++) та Python, системний підхід, експериментальне тестування системи у реальних умовах.

Оформлення пояснювальної записки виконано відповідно рекомендацій [1] та вимог ДСТУ 3008:2015 [2].

Результати дослідження апробовано та опубліковано у [18, 27].

1 ОГЛЯД ТА АНАЛІЗ СУЧАСНИХ МОБІЛЬНИХ СИСТЕМ ВІДЕОНАГЛЯДУ

У контексті даної роботи мобільна система відеоспостереження складається з двох основних частин, а саме роботизованої платформи з камерою та клієнтського програмного забезпечення.

Роботизована платформа представляє собою мобільного робота, загальні відомості про якого наведено нижче.

1.1. Загальні відомості про мобільних роботів

Мобільний робот (МР) – це автоматична машина, що має можливість пересуватися у навколишньому середовищі й не прикріплені до одного фізичного місця [3].

Мобільні роботи можуть мати дистанційне керування, або бути автономними (АМР), тобто здатними самостійно орієнтуватися у навколишньому середовищі, а також приймати певні рішення [4, 5].

Мобільних роботів класифікують:

– за середовищем, у якому вони пересуваються на наземні, повітряні та морські;

– за пристроєм, що використовується для пересування на крокуючі, колісні та гусеничні.

Найбільш популярними, практичними та простими у виконанні є колісні та гусеничні мобільні роботи, тому будемо розглядати саме ці класи роботів більш детально.

Колісні мобільні роботи найчастіше використовуються через їх простоту, високу швидкість та маневреність у середовищах з відносно рівною поверхнею землі. Цей клас роботів використовується для переміщення та/або сортування невеликих вантажів на складах та виробництвах (рис. 1.1, а),

моніторингу забруднення навколишнього середовища (рис. 1.1, б), патрулювання місцевості (рис. 1.1, в), або доставки товарів вулицями міст (рис. 1.1, г).

а б

в г

Рисунок 1.1 – Вигляд колісних мобільних роботів: а - робот для переміщення, комплектування та сортування «Dynamo»[6], б - робот моніторингу забруднення «Rasmus» [7], в - охоронний патрульний робот «S5.2» [8], г - робот-кур’єр «Camello» [9]

У той час, гусеничні роботи підходять для роботи в більш складних умовах, де потрібно долати перешкоди або працювати на нерівних поверхнях. Наприклад, гусеничний МР від компанії RoboTech Vision (рис. 1.2), що

виконує роботу з транспортування вантажів до 80 кг та косить траву між рядами у виноградниках.

Рисунок 1.2 – Вигляд гусеничного мобільного робота від компанії RoboTech Vision [10]

Гусеничні мобільні роботи мають великий недолік, а саме низьку швидкість та погану маневреність на рівних поверхнях, що робить клас колісних мобільних роботів більш оптимальним для побудови мобільної системи відеонагляду.

Колісний мобільний робот на базі ESP32-CAM є найпростішим варіантом колісного мобільного робота для відеоспостереження (рис. 1.3).

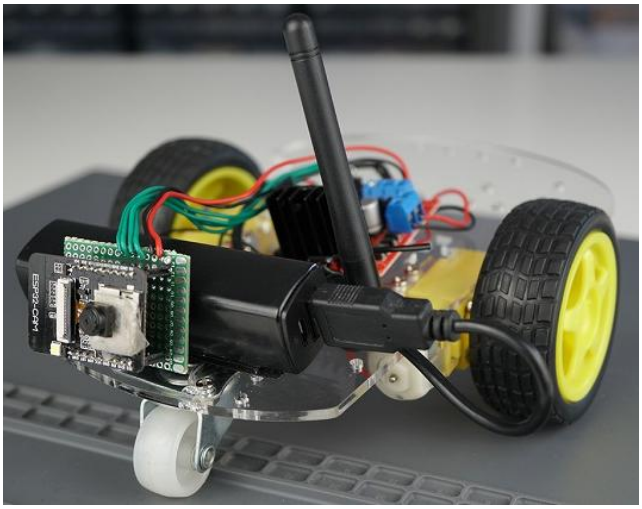


Рисунок 1.3 – Вигляд мобільного робота на базі ESP32-CAM [11]

Цей робот пересувається за допомогою двох керуючих коліс та одного допоміжного колеса, що утримує робота у рівновазі. Управління таким роботом відбувається за допомогою контролю обертами кожного з керуючих коліс, таким чином лише за допомогою двох керуючих коліс робот може пересуватися у всі сторони.

1.2. Основні вимоги та структурні елементи мобільних роботів

Для створення колісного мобільного робота відеонагляду (МРВ) необхідно визначити перелік вимог.

Вимоги до МРВ:

– трансляція зображення;
– дистанційне керування;
– вільне пересування у просторі;
– автономне живлення.

Спочатку оглянемо та порівняємо можливі варіанти плат з мікроконтролером, що забезпечить з’єднання через Wi-Fi, передачу зображення у реальному часі з оптимальними показниками якості та компактність конструкції. У табл. 1.1 наведено основні технічні характеристики плат керування.

Таблиця 1.1 – Основні технічні характеристики плат керування

Назва	Freenove ESP32-WROVER CAM [12]	ESP32-CAM [13]	Seeed Studio XIAO ESP32S3 [14]
Мікроконтролер	ESP32-Wrover-E	ESP-32S	ESP32-S3R8
Кількість ядер, шт. / тактова частота, Мгц	2 / 240	2 / 240	2 / 240

Продовження таблиці 1.1.

ОЗУ / PSRAM, МБайт	4 / 4	4 / 4	8 / 8
Кількість доступних входів / виходів, шт.	6	10	11+2
Габаритні розміри, мм	126 x 83 x 26	40,5 x 27 x 4,5	21 x 17,8 x 15
Ціна, грн	445,00	248,00	1315,00

Усі наведені у таблиці плати мають можливість підключення до Wi-Fi та Bluetooth, а також камеру OV2640, що підключається до плати лише одним шлейфом. Таке підключення покращує ергономічність готового робота та полегшує підключення камери до мікроконтролера, а також забезпечує надійність та якість передачі зображення [15].

Отже тепер, необхідно обрати драйвер, що дозволяє контролювати роботу електродвигунів. Характеристики плат драйверів двигунів постійного струму наведено у табл. 1.2.

Таблиця 1.2 – Характеристики плат драйверів двигунів постійного струму

Назва	$U_{\text{р.лог.}}$, В	$U_{max.\text{мот.}}$, В	I_{max}, А	Керування кількістю обертів	Керування напрямком руху	Ціна, грн
DRV8833 [16]	3 – 5,5	10,8	2	Є	Є	52,00
L298N [17,18]	5	35	2	Є	Є	65,50
TB6612FNG [19]	2,7-5,5	15	2	Є	Є	94,00

1.3. Реалізація автоматизованого керування мобільним роботом

Автоматизована система керування (АСК) – це сукупність керованого об’єкта, автоматичних вимірювань та керуючих пристроїв, у якій частину функцій виконує людина [20, 21]. З цього твердження можна зробити висновок, що до АСК відноситься і дистанційне керування, що часто використовується для управління мобільними роботами.

Дистанційне керування (ДС) мобільними роботами може здійснюватись за допомогою радіохвиль певної частоти. Одним з методів ДС є концепція Інтернету речей (з англ. Internet of Things, IoT). IoT – це система фізичних об’єктів («речей»), взаємопов’язаних між собою за допомогою вбудованих датчиків, програмного забезпечення та/або інших технологій. Цей зв’язок потрібний для того, щоб передавати дані на інші пристрої в системі або в інші системи через Інтернет [18, 22]. Застосування цієї концепції для керування мобільними роботами передбачає використання таких засобів передачі даних в мережі, як ZigBee, Wi-Fi та Bluetooth [23]. Розглянемо кожен з цих засобів детальніше.

ZigBee – це стандарт бездротової мережі з малим енергоспоживанням та коротким радіусом дії, що був розроблений для підтримки бездротових мереж, що мають велику кількість пристроїв, працюючих з низькою потужністю, наприклад у розумному будинку, або промисловій автоматизації. Працює він у діапазоні 2,4 ГГц, що забезпечує завадостійкість та дозволяє забезпечувати зв’язок між різними пристроями відділеними на відстань до 100 метрів у прямій лінії зору, а також дозволяє збільшувати радіус дії за рахунок підключення в мережу ретрансляторів [24].

Bluetooth – це бездротова технологія, що використовується для передачі даних в персональних мережах. Він передає дані по смузі частот від 2,4 до 2,485 ГГц і працює на коротших відстанях, ніж Wi-Fi. За допомогою Bluetooth можна синхронізувати пару пристроїв, таких як телефони, навушники, колонки, комп’ютери і багато іншого. З розвитком Bluetooth v4.0 з’явилася

можливість реалізувати функцію низького енергоспоживання і збільшеного радіусу дії до декількох десятків метрів [22].

Wi-Fi – це локальна бездротова технологія передачі даних, що використовує частоти 2,4 ГГц або 5 ГГц. Така технологія добре підходить для передачі великих обсягів даних між пристроями, але вимагає багато енергії для роботи та має невеликий рівень пропускної здатності. Великою перевагою використання цієї технології для керування мобільними роботами є те, що з її допомогою МР має можливість підключення до мережі Інтернет, що дозволяє керувати роботом з будь-якої точки земної кулі, за умови наявності мережі Інтернет.

1.4. Використання протоколу MQTT для керування мобільним роботом

MQTT – це стандартизований протокол або набір правил для обміну повідомленнями, який використовується для взаємодії між комп'ютерами. Інтелектуальні датчики, носимі пристрої та інші пристрої IoT зазвичай передають та отримують дані через мережі з обмеженими ресурсами та пропускною здатністю. До основних переваг протоколу MQTT відносять його простоту та ефективність, масштабованість, надійність та безпечність.

Протокол MQTT працює на принципах моделі «видавець/підписник». У традиційній мережевій комунікації клієнти та сервери спілкуються один з одним безпосередньо. Клієнти запитують ресурси або дані від сервера, після чого сервер обробляє та відправляє відповідь. Проте MQTT використовує модель видавець/підписник для роз’єднання відправника повідомлення (видавця) від отримувача повідомлення (підписника). Замість цього третій компонент, який називається мережевим брокером, відповідає за зв’язок між видавцями та підписниками. Завдання брокера полягає в фільтрації всіх вхідних повідомлень від видавців та їх правильному розподілі підписникам. Брокер роз’єднує видавців та підписників за наступними принципами:

– просторове роз’єднання, тобто видавець та підписник не мають інформації про мережеве розташування один одного і не обмінюються такими даними, як IP-адреси або номери портів;

– часове роз’єднання, за цим принципом видавець на підписник не використовують, або не мають одночасного мережевого з’єднання;

– роз’єднання синхронізації, тобто видавці та підписники можуть відправляти та отримувати повідомлення без перешкод один одному, що забезпечує асинхронність [25, 26].

Таким чином, схема організації керування мобільним роботом буде виглядати, як наведено на рис. 1.4.

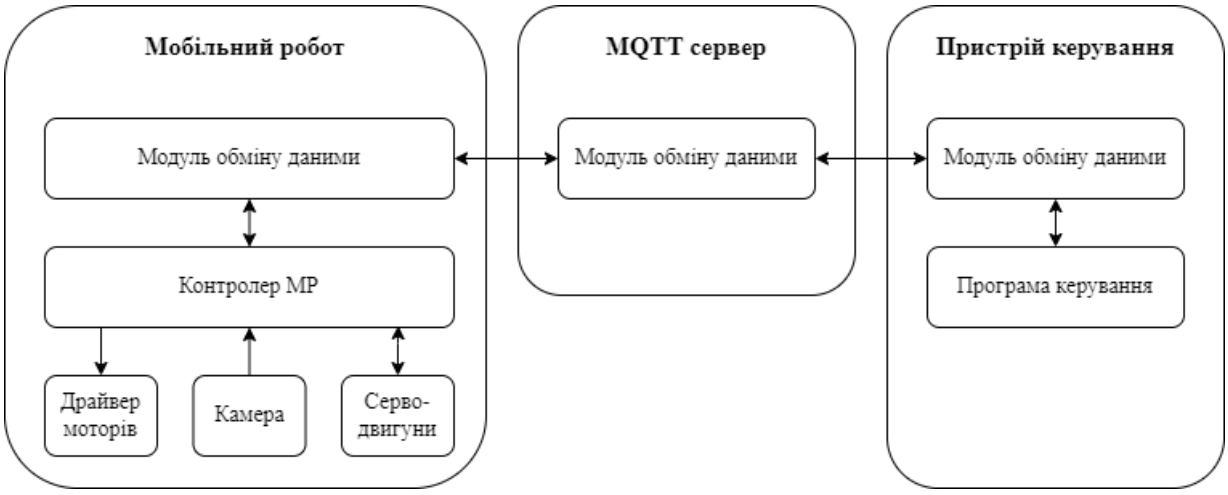


Рисунок 1.4 – Схема організації керування мобільним роботом за допомогою протоколу MQTT

Таким чином, можна передавати не тільки керуючі сигнали для пересування мобільного робота та керування штативом нахилу та повороту, а також можна забезпечити передачу зображення, перетворивши зображення у текст, наприклад за допомогою стандарту кодування BASE64.

1.5. Постановка задач та мети дослідження

Огляд літературних джерел, що присвячені тим чи іншим аспектам розроблення та керування МР, показав, наявність накопиченого досвіду у даній тематиці. Основні результати зосереджені на вирішенні окремих завдань, таких як дальність зв’язку, точність, час автономної роботи та вартість.

Наближені методи широко використовуються через складність реалізації та неможливість застосування точних методів для великих практичних задач.

Завданнями вдосконалення та оптимізації мобільної системи відеонагляду (МСВ) за процесами виробництва є підвищення енергоефективності, збільшення радіусу зв’язку та часу автономної роботи. Чисельні публікації з даної тематики не покривають усіх проблем оптимізації та вдосконалення таких систем, що вимагає подальших досліджень. Тому актуальним залишається завдання покращення та оптимізації МСВ.

Метою даної роботи є розроблення МСВ за виробничими процесами та програмного забезпечення системи дистанційного керування, що забезпечить високу якість відеопотоку та надійність роботи, що включає апаратну та програмну складові.

Для досягнення поставленої мети необхідно:

– провести огляд відомостей про мобільні роботи;

– розглянути основні вимоги та структурні елементи мобільних роботів;

– провести аналіз можливих реалізацій автоматизованого керування мобільними роботами;

– дослідити можливість використання протоколу MQTT для керування мобільною платформою та передачі відеопотоку;

– сформулювати постановку мети та задач дослідження;

– розробити або вдосконалити існуючу систему дистанційного керування мобільною платформо.;

– обрати, розробити або вдосконалити алгоритм передачі зображення;

– розробити програмну та апаратну частини мобільної системи відеонагляду ;

– провести дослідження ефективності системи та порівняти з подібним рішенням.

1.6. Висновки до розділу 1

У першому розділі даної роботи було проведено аналіз існуючих МР та МСВ, їх класифікацію, структурні елементи. Було встановлено характеристики структурних елементів МР.

Також було проведено дослідження можливих засобів бездротових передачі даних, а саме ZigBee, Bluetooth та Wi-Fi.

Проведено аналіз можливості використання протоколу MQTT для керування мобільною системою та передачі зображення.

Сформовано постановку мети та задач дослідження.

2 РОЗРОБЛЕННЯ МОБІЛЬНОЇ СИСТЕМИ ВІДЕОСПОСТЕРЕЖЕННЯ

2.1. Обґрунтування вибору основних компонентів мобільної системи відеоспостереження

Враховуючи вище перераховані компоненти та технології, що необхідні для розробки МСВ, оберемо та обґрунтуємо найбільш оптимальні.

Для початку необхідно визначитися з мікроконтролером, що буде виступати у ролі головного компоненту МР та виконувати функції отримання та перетворення зображення, а також керування шасі та штативом нахилу/панорамування (ШНП).

Порівняння плат мікроконтролерів з камерою, характеристики яких наведено у табл. 1.1, показало, що оптимальною платою керування є ESP32-CAM (рис. 2.1).

Плата ESP32-CAM має шлейф для підключення камер роздільною здатністю понад 800x600 пікселів, має вбудований Wi-Fi та Bluetooth. Цей модуль поєднує у собі мікроконтролер ESP-32S з додатковою оперативною пам’яттю 4 МБайти та відеокамеру OV2640 [15], а габаритні розміри 40,5 x 27 x 4,5 мм забезпечать компактність мобільної платформи.

Рисунок 2.1 – Модуль ESP32-CAM [27]

Модуль ESP32-CAM керується за допомогою прошивки, написаною мовами Arduino, MicroPython та ін., що завантажується у мікроконтролер за допомогою USB-to-TTL перетворювачу. Потужності модуля достатньо для отримання та обробки зображення, обробки керуючих сигналів та інших задач.

Наступним кроком необхідно обрати драйвер двигунів постійного струму. Тут необхідно щоб драйвер міг керувати не тільки швидкістю руху, але і напрямком, при чому необхідно щоб на двигуни можна було подавати якомога вищу напругу, а ціна повинна бути відносно низькою. З трьох драйверів, що наведено у табл. 1.2 оптимальним варіантом є модуль драйвера на базі мікросхеми L298, що зображено на рис. 2.2.

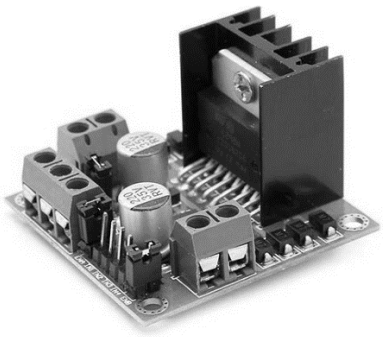


Рисунок 2.2 – Модуль L298N на базі мікросхеми L298 [28]

Даний модуль є найбільш поширеним серед багатьох робототехнічних проектів, має відносно невисоку ціну, а максимальна напруга живлення моторів складає 35В. Також цей модуль підходить для проекту через можливість керування як швидкістю, так і напрямком руху за допомогою лише чотирьох контактів, що важливо в умовах обмеженої кількості вільних контактів на платі ESP32-CAM.

Також вдалим доповненням буде реалізувати можливість вертикального та горизонтального повороту камери, що дозволить швидко та зручно регулювати положення камери. Це можливо реалізувати встановивши камеру на штатив, в який інтегровано два серводвигуни, наприклад SG90, рис. 2.3.

Рисунок 2.3 – Штати нахилу та панорамування з серводвигунами SG90 [29]

2.2. Обґрунтування вибору технологій системи

Згідно з поставленою задачею, необхідно розробити МСВ, що буде використовувати технології Інтернету речей. Тобто необхідно створити зв’язок між мобільною платформою з камерою (мобільним роботом) та програмним забезпеченням для керування МР. Обравши всі необхідні апаратні компоненти системи, необхідно обрати та обґрунтувати метод організації віддаленого керування роботизованою платформою.

Для керування МР буде використано технологію передачі даних Wi-Fi та протокол MQTT. Таким чином, основний код програми керування та перетворення зображення буде знаходитись у прошивці МР, потужності мікроконтролера ESP-32S буде достатньо для отримання та передачі відеопотоку, а також обробки команд керування. Тому МСВ буде складатись з МР, що буде передавати зображення на MQTT сервер, який у свою чергу буде перенаправляти повідомлення у програмне забезпечення для відображення зображення користувачу, а команди керування мобільною платформою будуть вводитися користувачем, передаватись на MQTT сервер і перенаправлятись до МР, який буде реагувати на команди відповідним чином.

2.3. Розрахунок параметрів джерела живлення

Забезпечення необхідного джерела живлення є важливою задачею при створенні мобільних платформ, бо кожен елемент має своє енергоспоживання, робоча напруга та струм кожного з компонентів МР наведено у табл. 2.1.

Таблиця 2.1 – Робоча напруга компонентів мобільного робота

Назва компоненту	Робоча напруга, В	Робочий струм, А
ESP32-CAM	5	0,18 - 0,31
L298N	5 - 35	2
SG90	3,5 - 5	0,50 - 0,80

Необхідно зауважити, що модуль драйвера L298N містить у собі DC-DC перетворювач напруги з 5-35В до 5В, що дає можливість не використовувати додатковий DC-DC перетворювач у конструкції МР.

Для оптимальної швидкості роботи електродвигунів необхідна напруга 12В, тому джерело живлення повинно бути 12В.

Для цього пропонується зібрати акумуляторний блок, що складається з трьох акумуляторів типу 18650 підключених послідовно. Обраний акумулятор Samsung INR18650-35E мають номінальну напругу 3,6 В, напругу повного заряду – 4,2 В, та максимальний постійний струм розряду 8А, а ємність складає 3500 мАг [30].

Вихідна напруга акумуляторного блоку розраховується за формулою:

$$U_{\text{вих}} = \quad U_{ak} \cdot n, \tag{2.1}$$

де U_{ak} – напруга акумулятора;

n – кількість акумуляторів у блоці.

Проведемо розрахунок вихідної напруги для трьох послідовно з’єднаних акумулятори, напруга кожного становить 4,2 В:

$$U_{\text{вих}} = \quad 4{,}2 \cdot 3 = 12{,}6 \text{ В}.$$

Також розрахуємо максимальне сумарне споживання струму МР, для цього скористаємося наступною формулою:

$$I_{max} = \sum_{n=1} I_{max_n}, \tag{2.2}$$

де I_{max} – максимальний струм споживання МР,

I_{max_n} – максимальний струм споживання *n*-ого з компоненту МР,

n – компонент МР.

Таким чином струм споживання МР, з урахуванням даних з табл. 2.1 становить:

$$I_{max} = 0{,}31 + 2 + 0{,}8 \cdot 2 = 3{,}91 \text{ А}.$$

Також необхідно зауважити, що обрані акумулятори не обладнані схемою захисту, тому важливо використовувати окрему плату захисту, наприклад плату HX-3S-JHA10, що наведено на рис. 2.4.

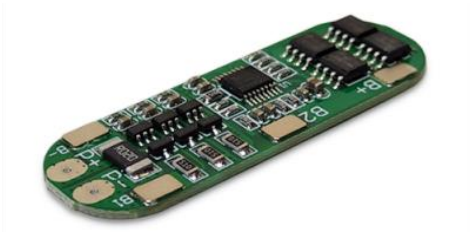


Рисунок 2.4 – Модуль захисту для акумуляторів HX-3S-JHA10 [31]

Цей модуль керування акумуляторною батареєю (АКБ) для трьох літій-іонних акумуляторів, що забезпечує захист під перезаряду, перерозряду, короткого замикання та перенавантаження для акумуляторів. Також модуль оснащений балансиром, що дозволяє вирівнювати заряд та розряд між

окремими акумуляторами, що допомагає продовжити строк служби АКБ. Даний модуль має максимальний струм зарядки та розрядки до 10 А.

Не менш важливо забезпечити правильний заряд АКБ, для цього можна використати зарядний модуль для трьох акумуляторів типу 18650, який зображено на рис. 2.5.

Рисунок 2.5 – Модуль заряду трьох акумуляторів типу 18650 з USB Type-C [32]

Цей модуль може забезпечити правильний заряд АКБ, зарядним струмом 0,75 А, при чому вхідна напруга до 6 В та 2 А, таку потужність має більшість зарядних пристроїв для смартфонів, а наявність роз’єму USB Type-C дозволяє використовувати звичайний зарядний пристрій для телефону, як блок живлення для зарядки МР.

2.4 Дослідження стійкості руху мобільного робота

Важливим етапом перед розробкою будь-якого мобільного робота є дослідження стійкості руху такого робота. Мобільний робот, що підходить для виконання поставленої мети має два ведучих колеса, що приводяться в обертання електродвигунами та одне псевдо колесо. Для того щоб такий МР міг пересуватись прямолінійно потрібно враховувати фактори, що можуть вплинути на нестійкість руху МР, для цього виконується моделювання динаміки руху мобільного робота, наприклад за допомогою програмного середовища MatLab.

Динаміка руху МР описується нелінійною системою диференціальних рівнянь. Модуль V швидкості центра мас та кутова швидкість Ω робота задовільняють динамічним рівнянням [33]:

$$\dot{V} = -\frac{2c_2}{mr^2}V + a\Omega^2 + \frac{c_1}{mr}\left(U_L + U_R\right), \tag{2.3}$$

$$\dot{\Omega} = -\frac{2c_2 l^2}{I_A r^2}\Omega - \frac{ma}{I_A}V\Omega + \frac{c_1 l}{I_A r}\left(U_R - U_L\right), \tag{2.4}$$

де m – маса корпуса,

I_A – момент інерції корпуса,

V – лінійна швидкість центру мас,

Ω – кутова швидкість МР,

L – напіввідстань між центрами ведучих коліс МР.

Для вирішення питання прямолінійного стійкого руху МР необхідно врахувати, що у такому випадку на двигуни ведучих коліс подається однакова напруга, тому можемо ввести наступні позначення [33]:

$$\mu = -\frac{2c_2}{mr^2} \text{ та} \tag{2.5}$$

$$p = \frac{c_1}{mr}\left(U_L + U_R\right), \tag{2.6}$$

де μ – параметр, пропорційний коефіцієнту сил в’язкого тертя, p – позитивний параметр, пропорційний сумі напруг, що подаються на провідні колеса робота, a – абсциса центра мас робота в рухомій системі. Таким чином, коли a, тобто центр мас та третє псевдо колесо знаходяться попереду ведучих коліс. Величина $\frac{1}{\rho^2} = \frac{m}{I_A}$, де ρ – радіус інерції робота щодо осі, яка проходить через

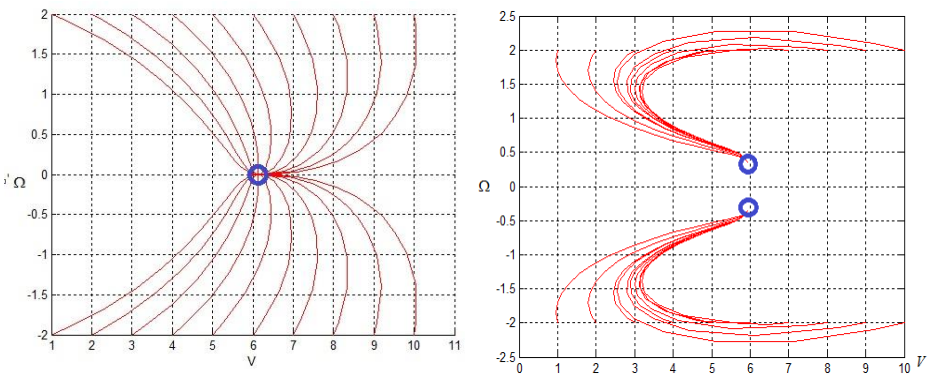
центр мас. Таким чином, диференціальні рівняння руху МР набувають такого вигляду:

$$\frac{dV}{dt} = -\mu V + a\Omega^2 + p, \tag{2.7}$$

$$\frac{d\Omega}{dt} = -\mu \frac{l^2}{r^2}\Omega - \frac{a}{p^2}V\Omega. \tag{2.8}$$

Розглянувши дану модель динаміки МР та провівши моделювання за допомогою солвера «ode45» у MatLab, було отримано фазові портрети прямолінійного руху МР у випадках, коли центр мас і псевдоколесо знаходиться попереду ведучих коліс, тобто $a > 0$ та коли центр мас і псевдоколесо знаходиться позаду від ведучих коліс, тобто $a < 0$, розглянемо детальніше кожен з випадків.

На рис. 2.6, а наведено фазовий портрет при $a > 0$. Бачимо, що у такому випадку система є стійкою та має особливу точку типу «вузол». На рис. 0.0 видно, що лінійна швидкість V з часом прямує до сталого значення 6, при цьому кутова швидкість поступово наближається до нуля, тому можемо зробити висновок, що у такому випадку розміщення коліс, не буде відбуватись відгалуження від запланованої траєкторії руху, а також ефект закручення або заносів МР при поворотах.



а б

Рисунок 2.6 – Фазовий портрет: а - при $a > 0$, б - при $a < 0$[33]

Розглянемо зворотній випадок, фазовий портрет при $a < 0$ наведено на рис. 2.6, б.

У такому випадку центр мас і псевдо колесо знаходяться позаду ведучих коліс МР, бачимо наступне, при збільшенні поданої напруги на електродвигуни МР, і, відповідно, збільшенні лінійної швидкості утворюються 2 особливі точки системи. При цьому бачимо, що, як на рис.2.6, а так і на рис. 2.6, б лінійна швидкість прямує до значення 6, але у даному випадку кутова швидкість набуває сталого позитивного значення, що у свою чергу може призводити до появи обертального руху з певною постійної швидкістю, що може призвести до неконтрольованих заносів під час поворотів або ефекту «закручення».

Можемо зробити наступні висновки, коли псевдо колесо знаходиться попереду ведучих коліс МР буде рухатись по прямій траєкторії з будь-якою швидкістю, на відміну від МР що має псевдо колесо позаду від ведучих коліс, тому у таких випадках є доцільним використовувати алгоритми для стабілізації руху МР з отриманням зворотного зв’язку про положення робота у певний момент часу.

2.5 Схеми підключення компонентів мобільного робота

Після проведення розрахунків блоку живлення та підбору усіх необхідних компонентів МР необхідно розробити схеми підключення компонентів. Для початку розглянемо принципову схему живлення МР, що наведено на рис. 2.7.

За даною схемою напруга 5В для живлення ESP32-CAM та сервоприводів отримується за допомогою DC-DC перетворювача, що встановлений на модулі L298N.

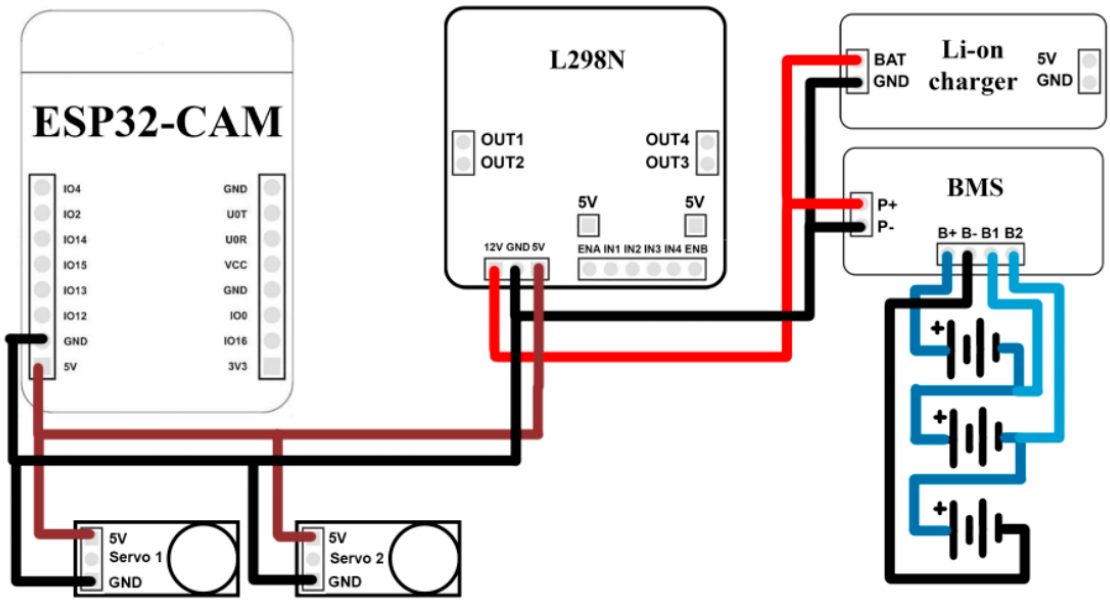


Рисунок 2.7 – Принципова схема живлення компонентів мобільного робота

Розробивши схему підключення живлення можна переходити до розроблення схеми підключення логічної частини, принципову схему наведено на рис. 2.8.

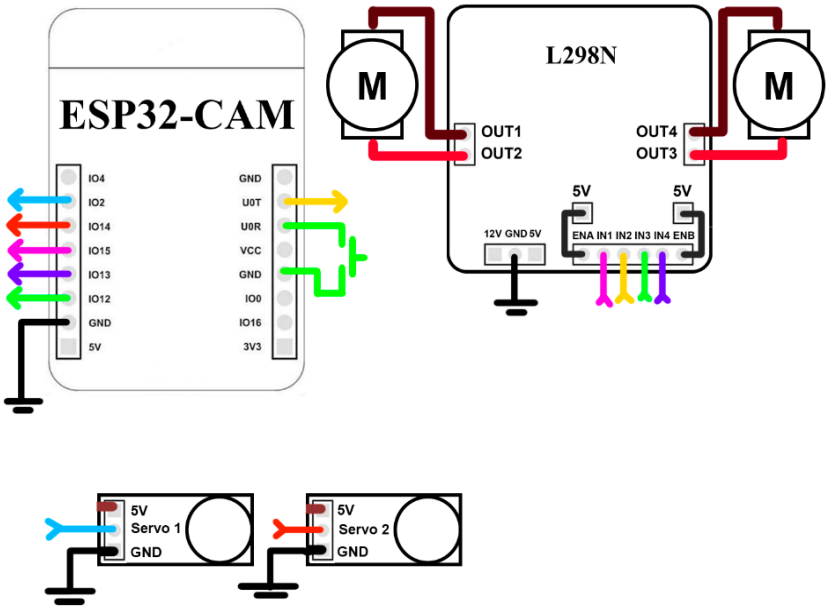


Рисунок 2.8 – Принципова схема логічної частини МР

На схемі видно, що використовуються такі контакти ESP32-CAM, як U0T та U0R, що за замовчуванням використовуються для зв’язку модуля з

іншими пристроями через UART, але у даному проекті це необхідно тільки для відлагодження роботи МР, тому було прийнято рішення перепризначити ці виходи для корисного використання, а для забезпечення відладки можна запрограмувати перехід у цей режим.

Також є особливість підключення ESP32-CAM до драйвера L298N. Підключення, що показано на рис. 2.8 забезпечує керування швидкістю обертів та напрямком руху моторів, використовуючи програмний ШІМ-сигнал, що може формулювати ESP-32S.

Підключення моторів до драйверу також потребує уваги, через відсутність певної полярності у двигунів постійного струму важко визначити напрямок його руху, тому може виникнути ситуація, що один або два мотори будуть рухатись у неправильному напрямку. Для вирішення цієї проблеми необхідно змінити полярність підключення електродвигунів до драйверу.

У разі, якщо керування серводвигунами не коректне, необхідно змінити їх підключення на зворотне.

Таким чином, отримали загальну принципову схему МР, що наведено на рис. 2.9.

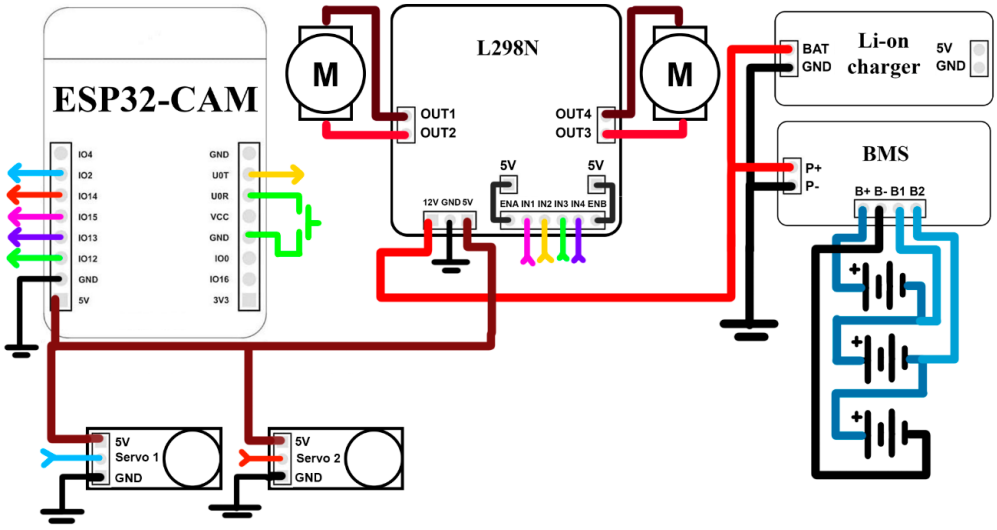


Рисунок 2.9 – Загальна принципова схема підключення компонентів МР

Тепер, коли всі компоненти та схема їх підключення відома, можна зібрати прототип мобільного робота відеонагляду за виробничими процесами, рис. 2.10.

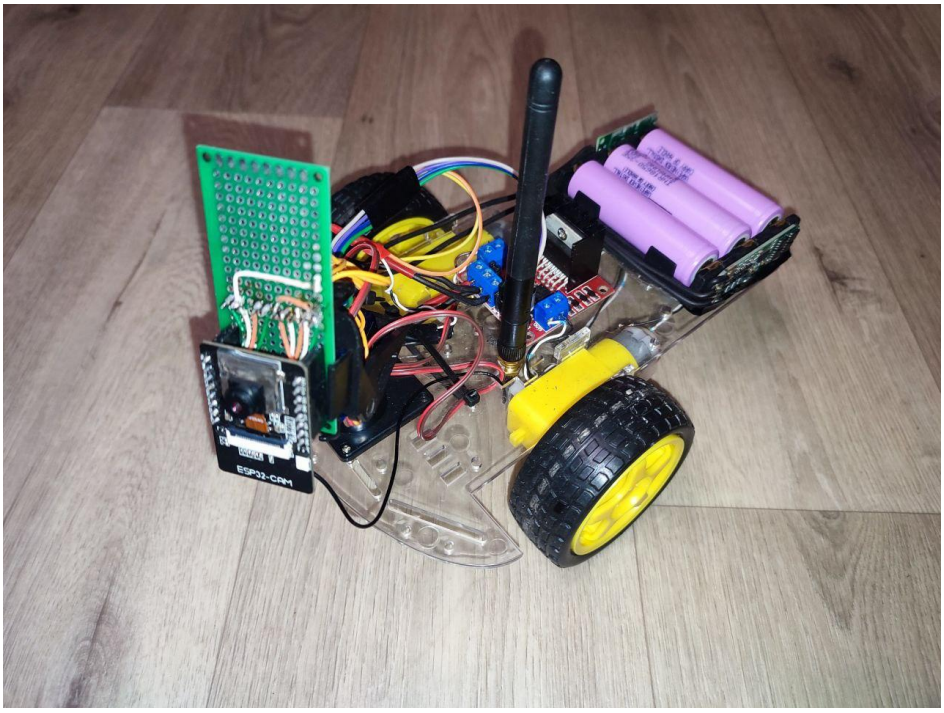


Рисунок 2.10 – Вигляд прототипу мобільного робота відеонагляду

2.6 Алгоритми роботи програмного забезпечення МР

Перед написання ПЗ для мобільної платформи необхідно визначитися з основними задачами, які повинен виконувати МР, як було визначено раніше, робот має виконувати:

- рух моторів вперед, назад, ліворуч та праворуч;
- зупинку електродвигунів;
- поворот штативу;
- нахил штативу;
- прийом, обробку та передачу зображення.

Для спрощення роботи під час розробки програмного забезпечення для МР необхідно розробити алгоритми програми МР. Почати потрібно з

підключення бібліотек та запуску основних процесів МР. На рис. 2.11 наведено схему алгоритму завантаження бібліотек та основних процесів для роботи МР. Цей алгоритм показує, процес ініціалізації МР в залежності від режиму роботи робота, стану кнопки при запуску та наявності даних у пам’яті EEPROM. Нескінченний цикл тут необхідний для запиту даних з MQTT серверу та клієнтської частини програми.

Далі необхідно розробити детальні алгоритми роботи окремих блоків програми. Почнемо з алгоритму перевірки стану кнопки при старті (рис. 2.12). Цей алгоритм повинен зчитувати початкове положення кнопки та у випадку, коли кнопка була затиснута при запуску програми, необхідно змінити значення змінної SETTINGS_CHANGE на True, що увімкне режим зміни налаштувань та перейти до виконання інших блоків алгоритму.

Тепер розглянемо алгоритм створення точки доступу Wi-Fi для налаштування МР, у випадку, коли при включенні була затиснута кнопка, або у EEPROM не було знайдено даних для підключення до Wi-Fi, рис. 2.13. Бачимо, що даний алгоритм створює нову точку доступу Wi-Fi та ініціалізує веб-сервер для налаштувань МР.

На цьому можна закінчити на алгоритмах, що необхідні для ініціанізації компонентів програми, тому перейдемо до алгоритмів основного циклу програми. Розглянемо головний цикл програми більш детально, схему алгоритму головного циклу програми наведено на рис. 2.14. Бачимо, що у цьому алгоритмі, що виконується нескінченно перевіряється стан кнопки на початку та кінці циклу, також виконується перевірка стану підключення до Wi-Fi та MQTT серверу у заданому проміжку часу. Також основний цикл програми містить дві гілки, одна для відображення сторінки налаштувань, у режимі налаштувань, інша – для передачі зображення на MQTT сервер та підтримки зв’язку з сервером MQTT.

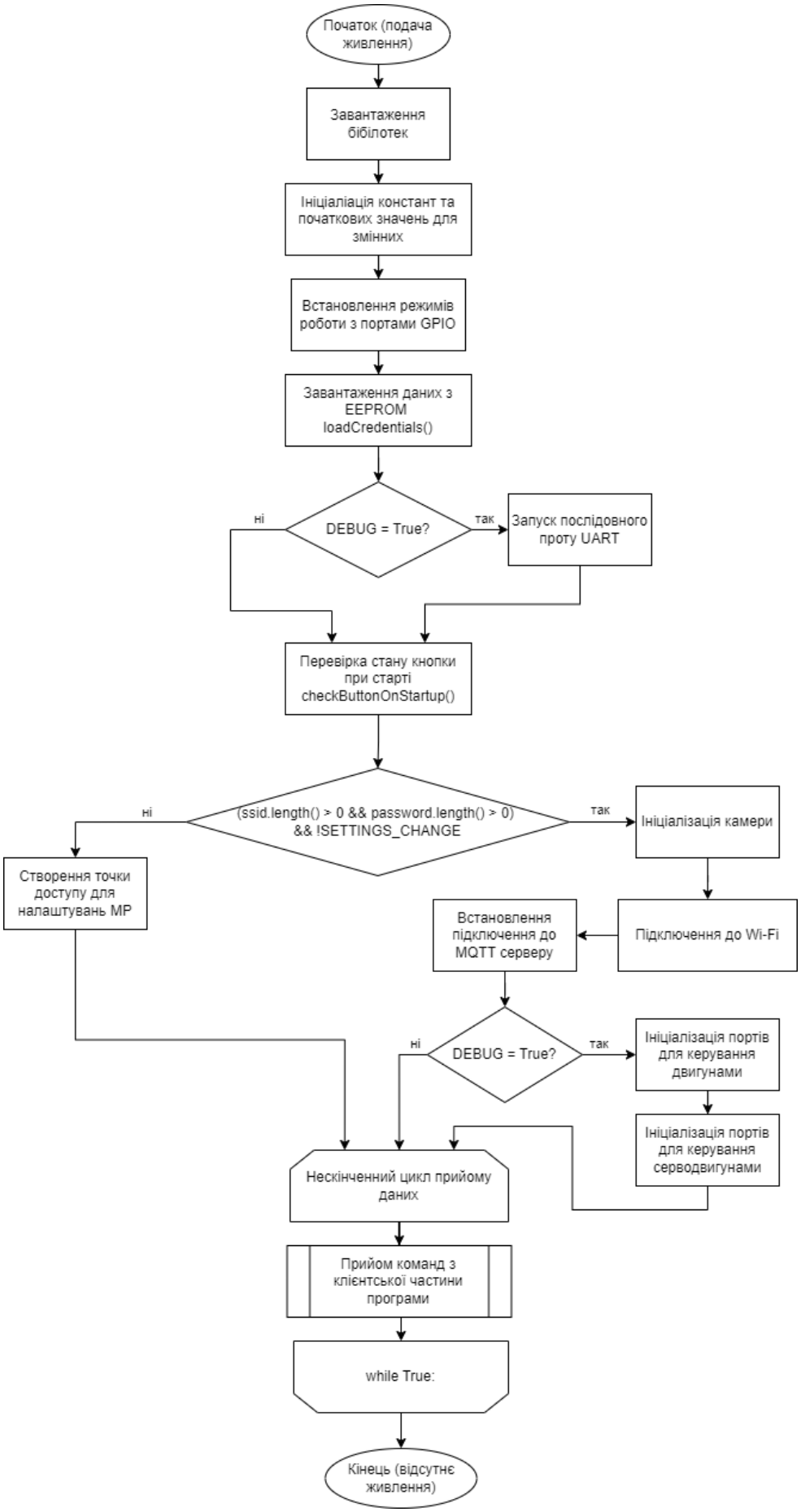


Рисунок 2.11 – Схема алгоритму роботи основних процесів МР

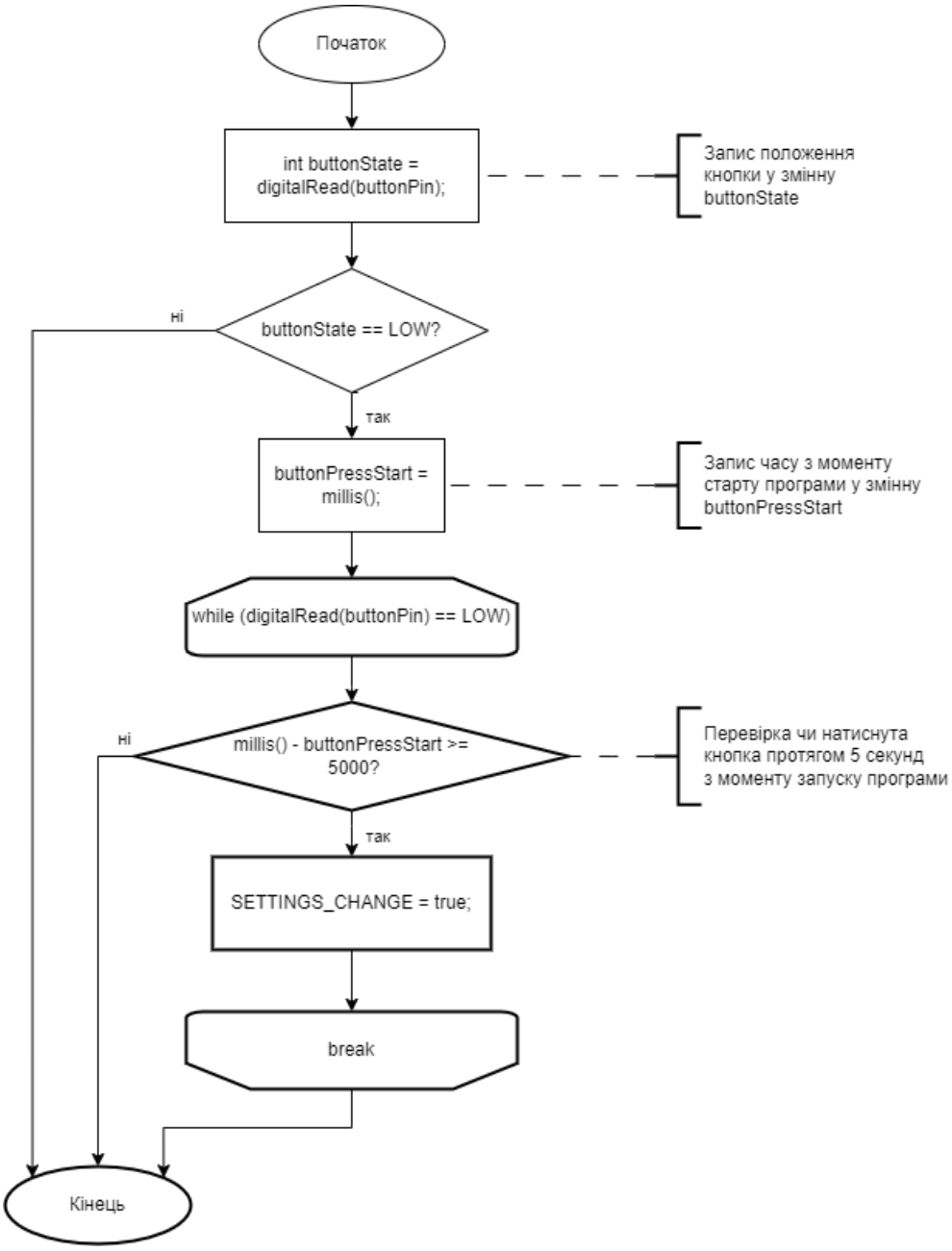


Рисунок 2.12 – Схема алгоритму роботи функції «checkButtonOnStartup»

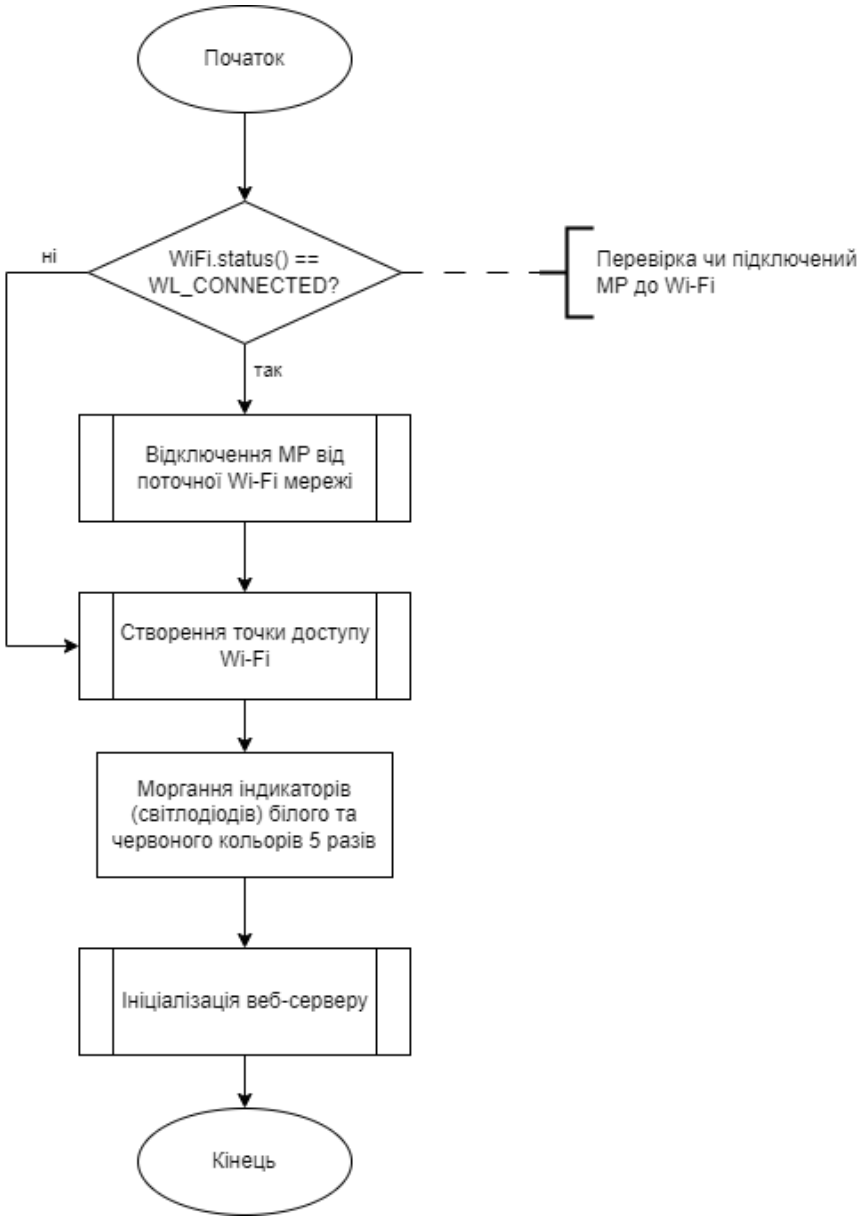


Рисунок 2.13 – Схема алгоритму роботи функції «createWiFiPoint»

Тепер розглянемо кожен з алгоритмів, що виконуються у головному циклі програми.

Спершу розглянемо схему алгоритму функції перевірки стану кнопки, рис. 2.15. Цей алгоритм дозволяє відслідковувати натискання кнопки, завдяки чому користувач може переходити у різні режими роботи МР.

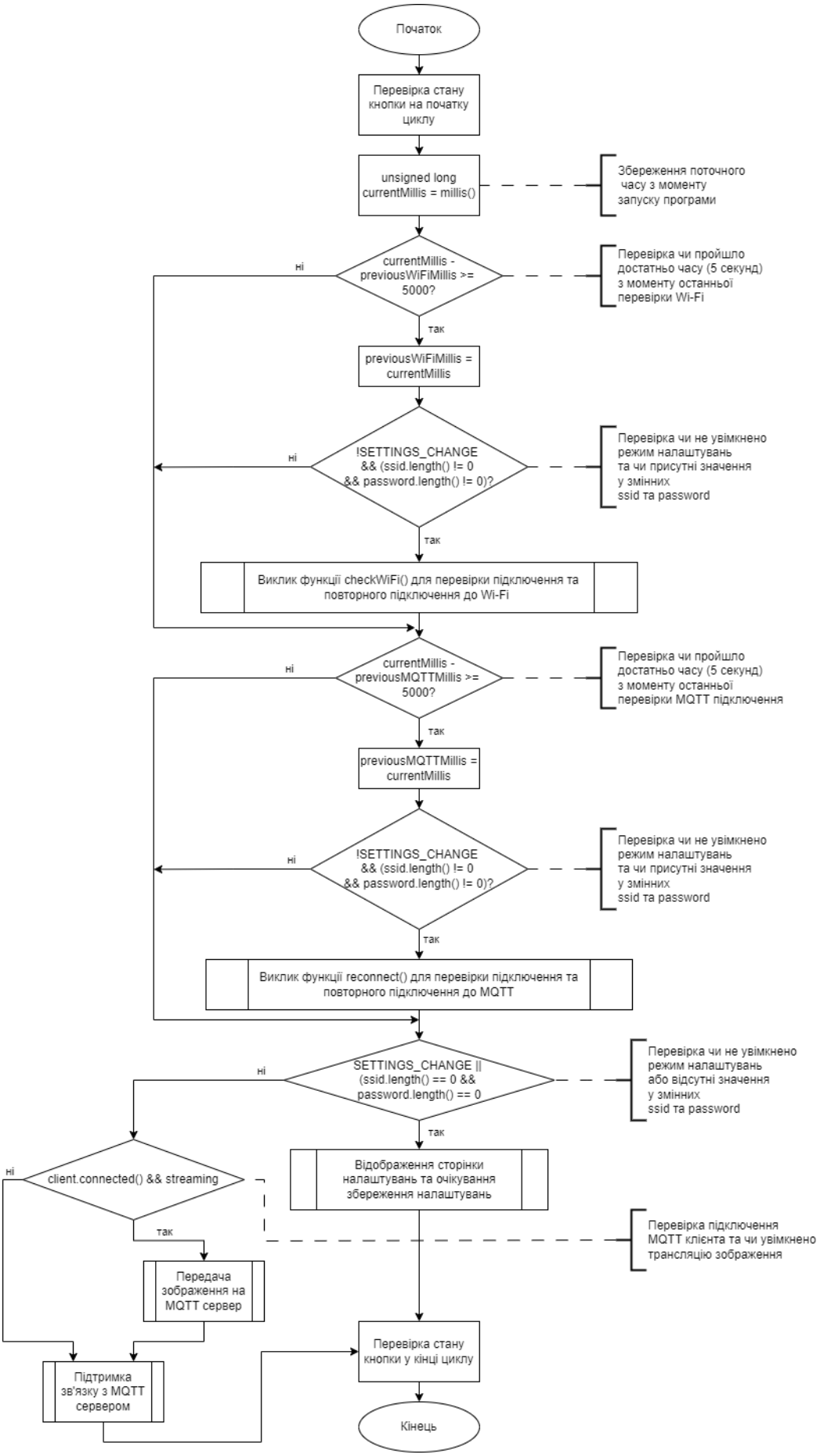


Рисунок 2.14 – Схема алгоритму основного циклу програми

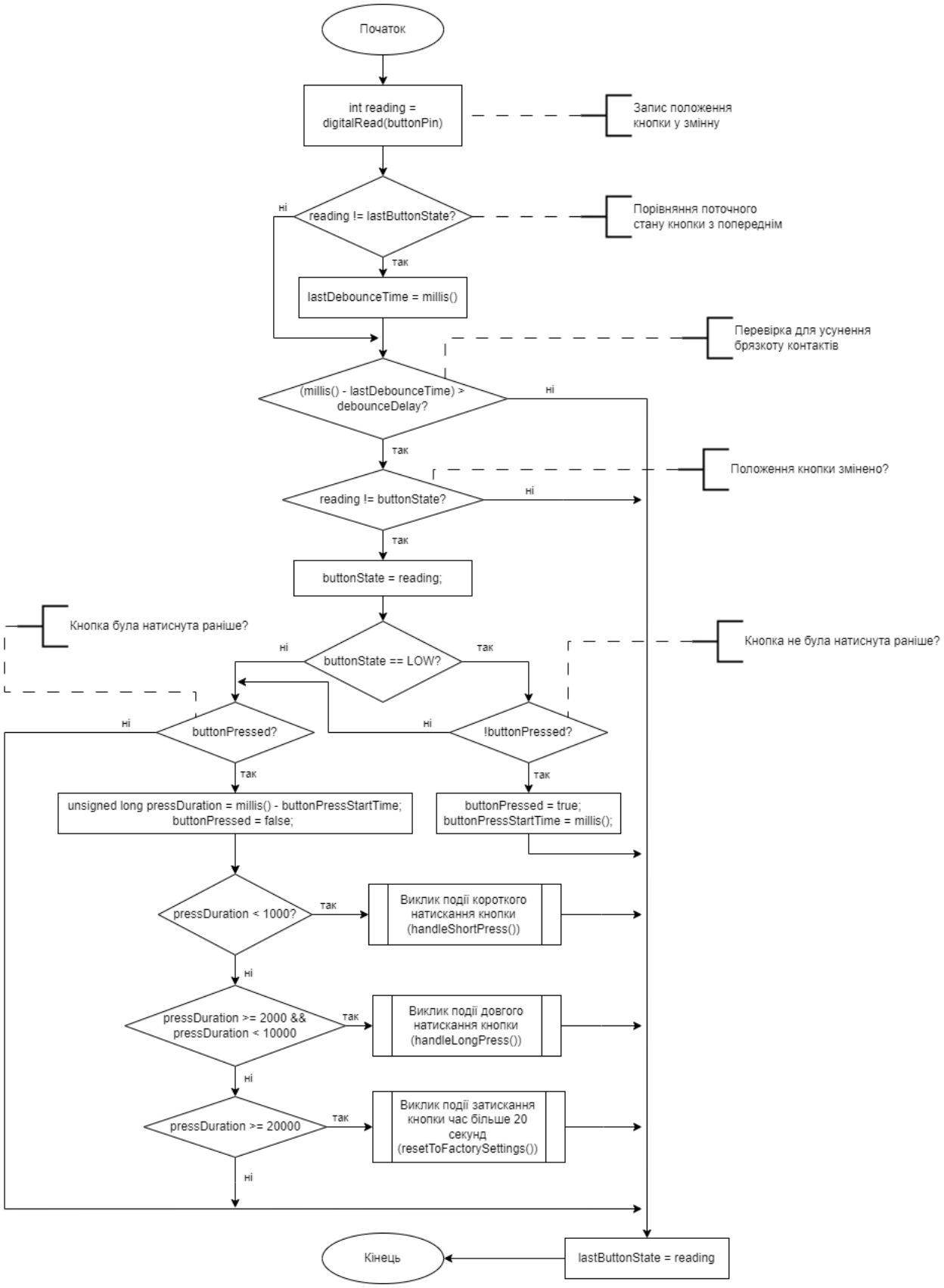


Рисунок 2.15 – Схема алгоритму обробки натискання кнопки

Алгоритм зображений на рис. 2.15 має можливість підрахування кількості та тривалості натискань кнопки, таким чином користувач натискає певну комбінацію за допомогою кнопки, а МР переходить у відповідний режим роботи.

Розглянемо алгоритм перевірки та перепідключення МР до мережі Wi-Fi, схема алгоритму наведена на рис. 2.16. Цей алгоритм необхідний для підтримки підключення до мережі та інформування користувача у разі неможливості підключення до мережі.

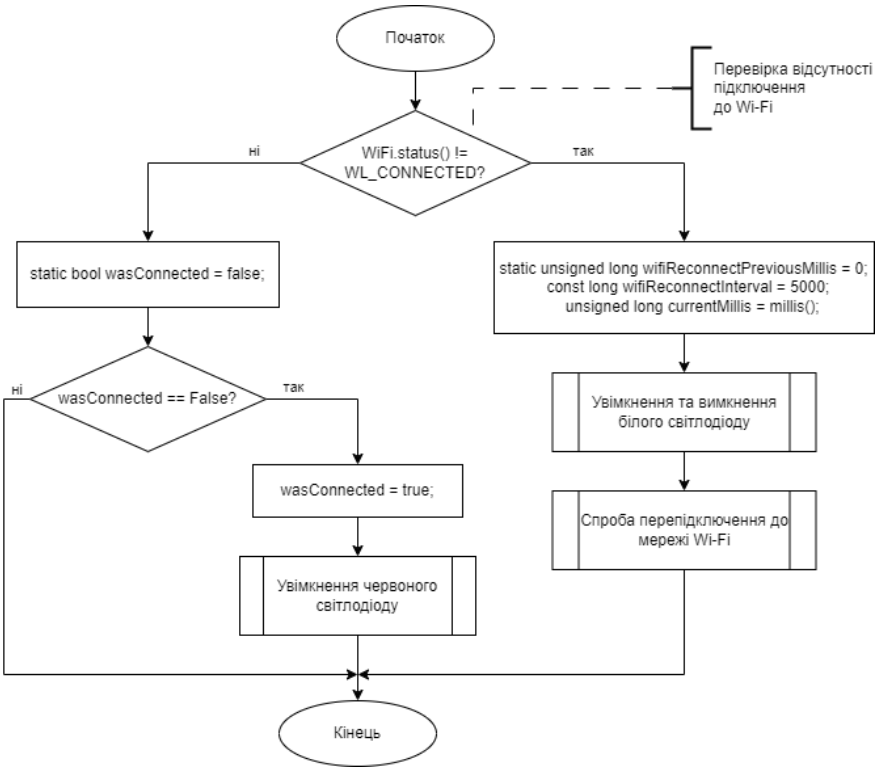


Рисунок 2.16 – Схема алгоритму перевірки та перепідключення МР до мережі Wi-Fi

Алгоритм роботи функції для перевірки та перепідключення до MQTT серверу має подібну структуру, тому його розглядати не будемо.

Наступним кроком розглянемо алгоритм роботи обробника команд, що надходять з MQTT сервера до МР, схему алгоритму наведено на рис. 2.17. Бачимо, що даний алгоритм викликає відповідну функцію, в залежності від команди, що надходить до МР.

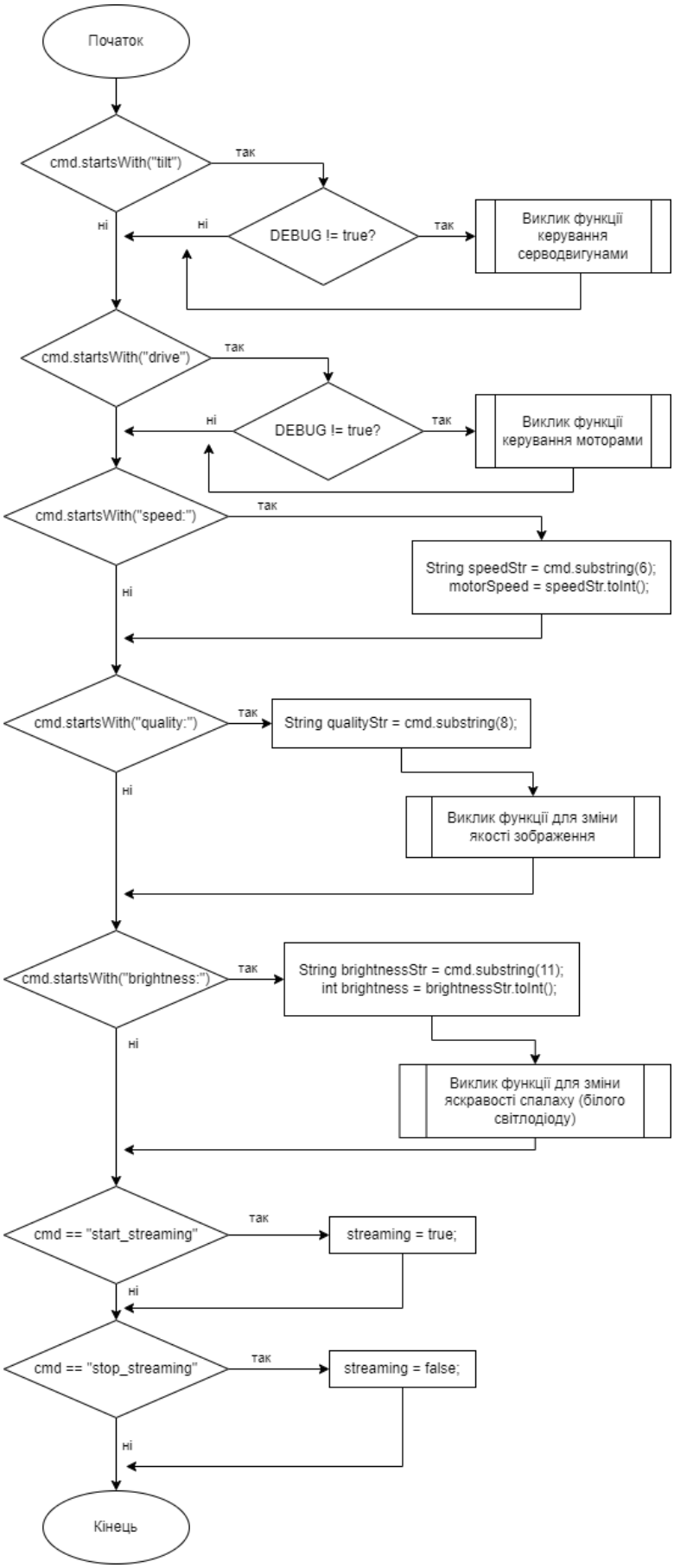


Рисунок 2.17 – Схема алгоритму обробника команд з MQTT серверу

Також видно, що для функцій керування сервоприводами та моторами додано перевірку чи знаходиться МР у режимі відладки, це зроблено для того щоб уникнути помилок у роботі МР, коли один вихід використовується у двох місцях програми, з двома різними налаштуваннями та призначенням.

Перейдемо до алгоритмів керування елементами МР, тобто обробка команд для керування шасі та штативу нахилу та панорамування. Почнемо з алгоритму керування сервоприводами для штативу нахилу та панорамування, схему алгоритму наведено на рис. 2.18.

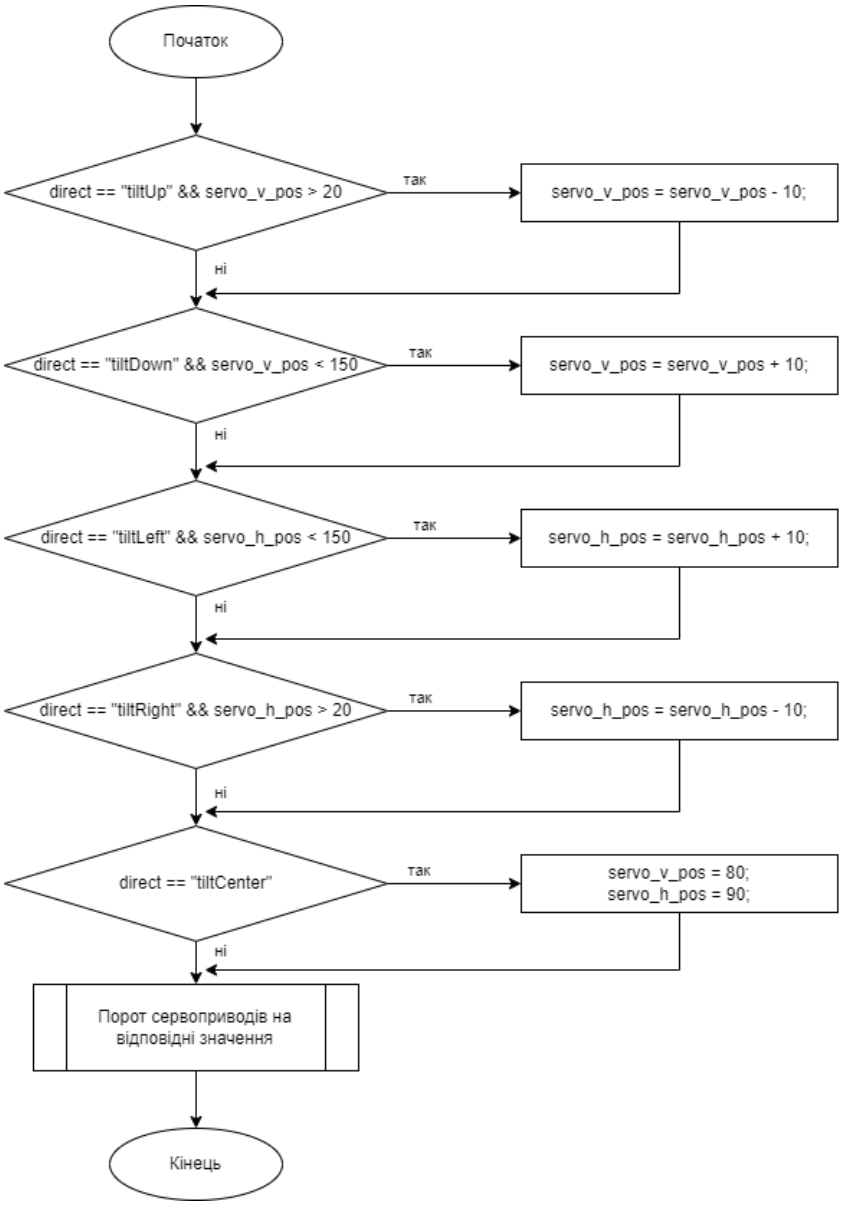


Рисунок 2.18 – Схема алгоритму роботи функції керування двома сервоприводами

Як бачимо, алгоритм перевіряє напрямок руху серводвигуна, що надходять з MQTT серверу та перевіряє щоб значення не перевищували очікувані, тобто не менше 20 і не більше 150. У кінці роботи алгоритму виконується поворот певного серводвигуна, в залежності від отриманого значення кута повороту.

Тепер розглянемо алгоритм керування шасі МР, схему алгоритму зображено на рис. 2.19.

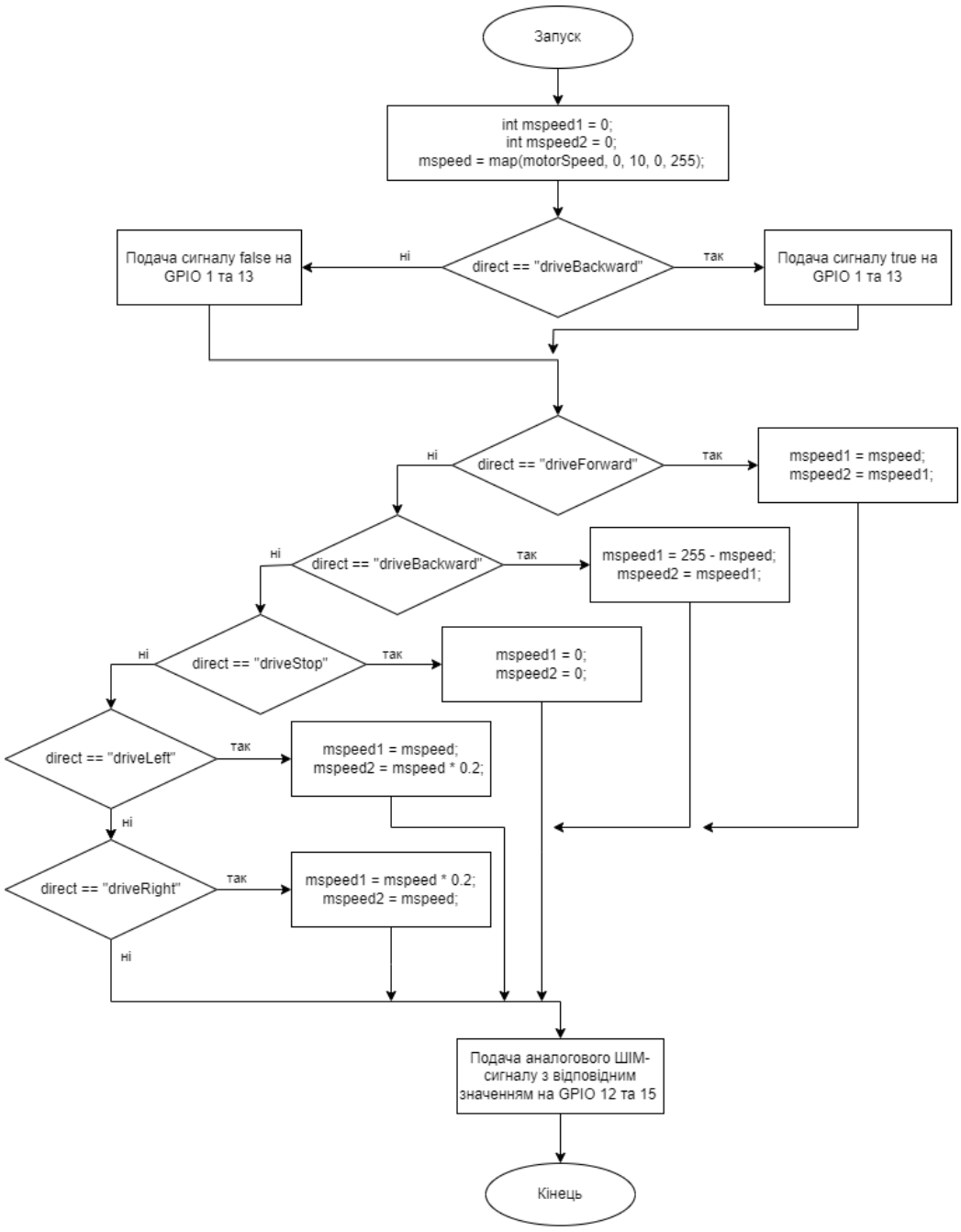


Рисунок 2.19 – Схема алгоритму роботи функції керування шасі МР

Тепер можемо розглянути алгоритм відправки зображень на MQTT-сервер, спрощену схему алгоритму наведено на рис. 2.20. Даний алгоритм отримує буфер зображення з камери ESP32-CAM, після успішного отримання зображення необхідно перекодувати у формат BASE64 для зручного передавання зображення через MQTT, далі, за умови успішного створення клієнта та підключення до серверу, починається процес публікації кожного отриманого зображення на сервер, коли зображення перевищує розмір у 2048 байт, зображення розбивається на частини. Потім публікація зображення завершується, буфер даних очищується і повертається порожній рядок, що свідчить про успішне виконання алгоритму.

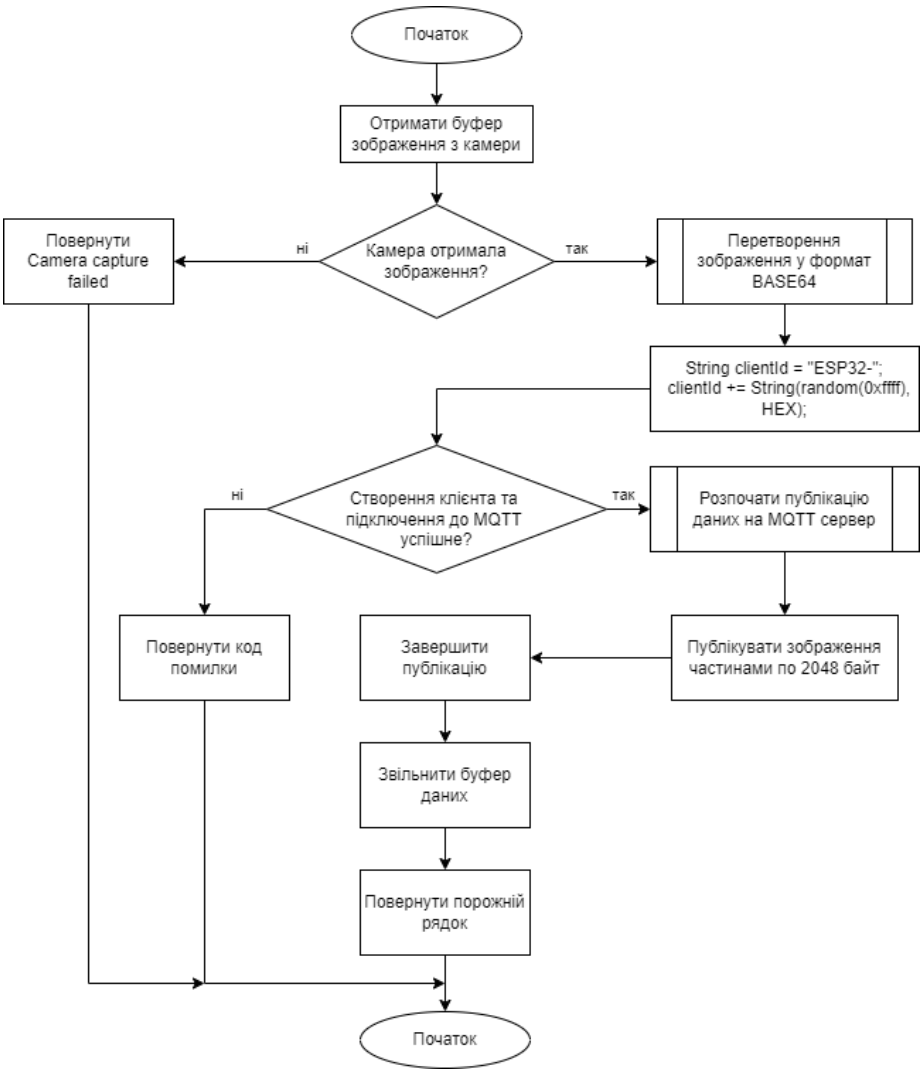


Рисунок 2.20 – Схема алгоритму перетворення та відправки зображень на MQTT сервер

2.7 Алгоритм роботи клієнтської частини програмного забезпечення

На рис. 2.21 наведено спрощену схему алгоритму роботи клієнтської частини програми, тобто веб-застосунку для керування МР через протокол MQTT.

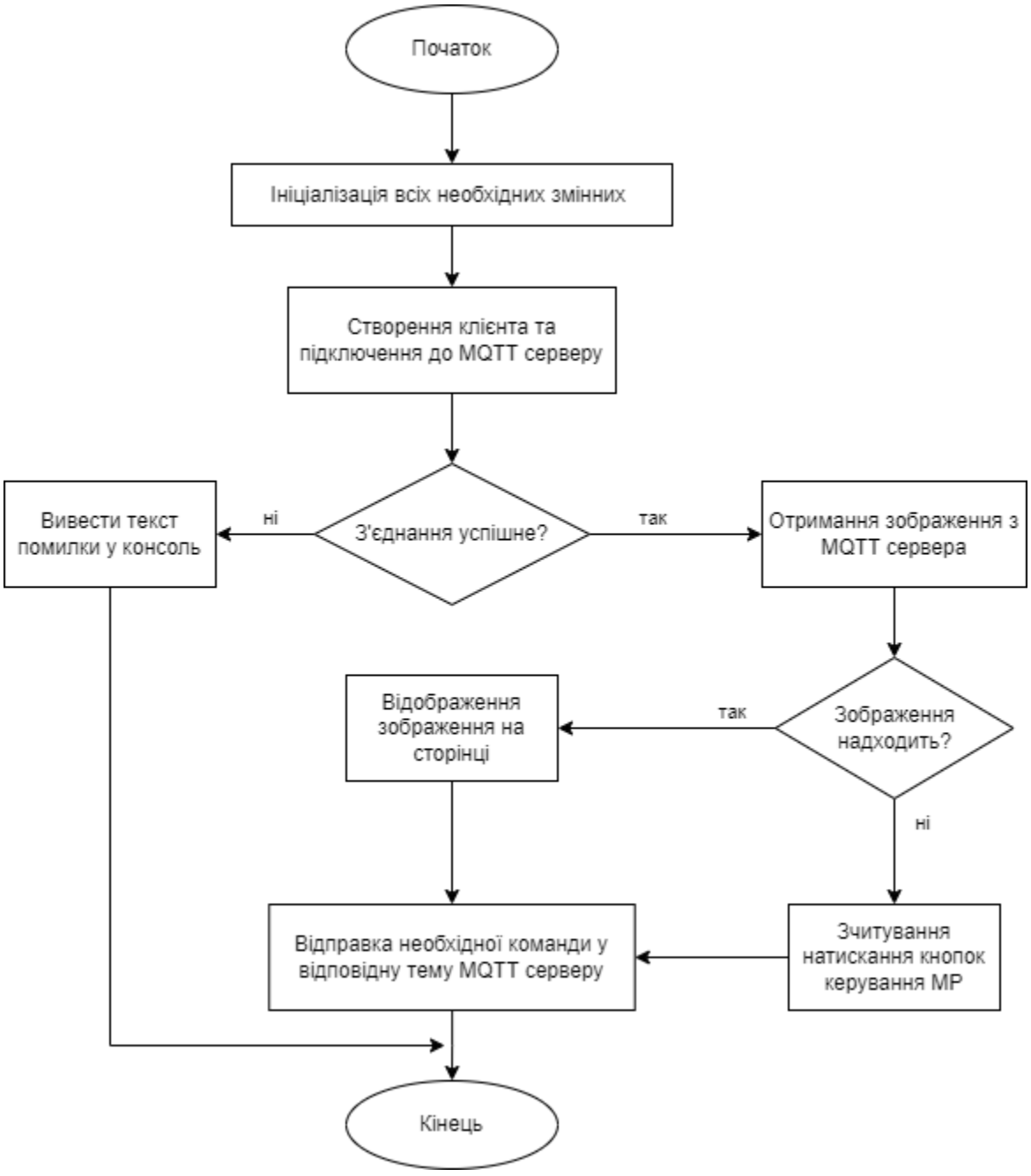


Рисунок 2.21 – Схема алгоритму роботи клієнтської частини програми для керування МР

2.8 Розроблення програмного забезпечення для мобільного робота

Розробка програмного забезпечення для МР є важливою складовою для мобільної системи відеонагляду, бо МР має приймати та обробляти команди, що надходять з MQTT серверу та транслювати зображення з камери на сервер.

У якості мови програмування було обрано мову Arduino, що базується C/C++, але розроблена таким чином щоб бути простішою та легшою для вивчення. Ця мова програмування є дуже поширеною для інших подібних проектів з робототехніки та IoT.

2.8.1 Початкові налаштування, головний цикл програми

Спочатку для правильної роботи всіх компонентів програми необіхдно підключити певний набір бібліотек:

```
#include <ESP32Servo.h>
#include <WebServer.h>
#include <EEPROM.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <Arduino.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include "Base64.h"
```

Бібліотека ESP32Servo використовується для підтримки мікроконтролерами ESP32 керування серводвигунами.

WebServer призначена для забезпечення функціоналу для створення вбудованого веб-сервера за допомогою плати ESP32.

EEPROM – це бібліотека, що забезпечує збереження та зчитування даних з пам’яті EEPROM.

Бібліотека WiFi необхідна для роботи з Wi-Fi на ESP32.

PubSubClient – це бібліотека для реалізації протоколу MQTT, забезпечує можливість публікації та підписки на MQTT-теми для передачі даних між пристроями та серверами через MQTT-брокери.

Arduino.h – це основна бібліотека для написання скетчів для плат Arduino та ESP32.

Бібліотеки soc/soc.h і soc/rtc_cntl_reg.h є системними, що призначені для роботи з низькорівневими компонентами системи на ESP32.

Бібліотека esp_camera.h необхідна для роботи з камерою, підключеною до ESP32 (наприклад, ESP32-CAM) та озволяє отримувати зображення з камери, а також налаштовувати параметри камери (якість, роздільна здатність тощо).

Бібліотека Base64.h ппризначена для кодування та декодування даних у форматі Base64 та забезпечує кодування зображень або інших бінарних даних у текстовий формат для легшої передачі через протоколи, які не підтримують бінарні дані (наприклад, HTTP або MQTT).

Наступним кроком необхідно зробити налаштування МР, тобто налаштувати режими роботи портів, камери, Wi-Fi. Налаштування відбувається в блоці «setup» головної програми. Код для налаштувань МР наведено нижче:

```
pinMode(buttonPin, INPUT_PULLUP);
pinMode(LED_PIN, OUTPUT);
ledcSetup(LEDC_CHANNEL, 5000, 8);
ledcAttachPin(FLASH_PIN, LEDC_CHANNEL);
loadCredentials();
if (DEBUG) {
   Serial.begin(115200);
```

```
    }
    loadCredentials();
    checkButtonOnStartup();
    if ((ssid.length() > 0 && password.length() > 0) && !SETTINGS_CHANGE)
{
      loger("Using saved credentials: %s | %s", ssid.c_str(), password.c_str());
      WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
      randomSeed(micros());
      initCamera();
      initWiFi();
      client.setServer(mqtt_server.c_str(), mqtt_port);
      client.setCallback(callback);
      client.setBufferSize(2048);
      if (!DEBUG) {
        initMotor();
        initServo();
      }
    } else if (SETTINGS_CHANGE || (ssid.length() == 0 && password.length()
== 0)){
      createWiFiPoint();
    }
```

Бачимо, що для скорочення коду використовуються функції initCamera(), initWiFi(), initMotor(), initServo() та createWiFiPoint(). Розглянемо код кожної з цих функцій детальніше.

Код функції для ініціалізації камери наведено нижче:

```
    void initCamera() {
      camera_config_t config;
      config.ledc_channel = LEDC_CHANNEL_0;
```

```
config.ledc_timer = LEDC_TIMER_2;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
if (psramFound()) {
  config.frame_size = FRAMESIZE_UXGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
```

```
    loger("Camera init failed with error 0x%x", err);
    ESP.restart();
  }
  sensor_t * s = esp_camera_sensor_get();
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);
    s->set_brightness(s, 10);
    s->set_saturation(s, 0);
  }
  s->set_framesize(s, FRAMESIZE_CIF);
}
```

Дана функція налаштовує конфігурацію камери, встановлюючи відповідні GPIO, кожен з яких вказується як зарезервована змінна, для передачі даних, тактового сигналу, синхронізації та інших параметрів. Залежно від наявності PSRAM (додаткової пам’яті), функція вибирає розмір кадру (роздільну здатність) та якість зображення JPEG. Якщо камера ініціалізується успішно, вона перевіряє модель сенсора (OV3660) і налаштовує деякі параметри, такі як вертикальне відображення, яскравість та насиченість.

Функція initWiFi() використовується для підключення ESP32 до Wi-Fi мережі в режимі станції (WIFI_STA). Вона виконує до 5 спроб підключення до заданої Wi-Fi мережі, якщо підключення успішне, виводить IP-адресу та вмикає червоний світлодіод (постійно горить), у випадку, коли підключення не вдалося після 5 спроб, світлодіод блимне 5 разів, що сигналізує про помилку. Код даної функції наведено нижче:

```
void initWiFi() {
  WiFi.mode(WIFI_STA);
  for (int i = 0; i < 5; i++) {
    WiFi.begin(ssid, password);
```

```
    delay(1000);
    loger("Connecting to %s", ssid.c_str());
    long int StartTime = millis();
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      if ((StartTime + 5000) < millis()) break;
    }
    if (WiFi.status() == WL_CONNECTED) {
      loger("STAIP address: %s", WiFi.localIP().toString().c_str());
      digitalWrite(LED_PIN, LOW);
    }
    break;
  }
  if (WiFi.status() != WL_CONNECTED) {
    for (int k = 0; k < 5; k++) {
      digitalWrite(LED_PIN, LOW);
      delay(100);
      digitalWrite(LED_PIN, HIGH);
      delay(100);
    }
  }
}
```

Розглянемо код для ініціалізації моторів, задаючи режими роботи для пінів керування моторами як виходи. Після цього вона викликає функцію motorControl("driveStop"), щоб зупинити мотори. Код даної функції наведено нижче:

```
void initMotor() {
  pinMode(motor1_IN, OUTPUT);
```

```
  pinMode(motor1_PWM, OUTPUT);
  pinMode(motor2_IN, OUTPUT);
  pinMode(motor2_PWM, OUTPUT);
  motorControl("driveStop");
}
```

Нижче наведено код функції initServo():

```
void initServo() {
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);
  servo_v.setPeriodHertz(50);
  servo_h.setPeriodHertz(50);
  servo_v.attach(SERVO_V_PIN, 500, 2400);
  servo_h.attach(SERVO_H_PIN, 500, 2400);

  servoControl("tiltCenter");
}
```

Функція initServo() ініціалізує серводвигуни на ESP32. Дана функція виділяє таймери для керування сервомоторами (2-й та 3-й таймери), встановлює частоту для серводвигунів на 50 Гц (стандартна частота для сервоприводів), підключає серводвигуни до відповідних пінів (SERVO_V_PIN і SERVO_H_PIN), задаючи мінімальні та максимальні значення ширини імпульсу (500–2400 мікросекунд), а також викликає функцію servoControl("tiltCenter"), яка встановлює серводвигуни в центральну позицію.

Останньої функцією, що необхідна у випадку коли МР запускається вперше, або у режимі налаштування – це createWiFiPoint(), код якої наведено нижче:

```
void createWiFiPoint(){
  if (WiFi.status() == WL_CONNECTED){
    WiFi.disconnect();
  }
  WiFi.softAP(apSSID, apPassword);
  loger("Access Point created");
  for (int k = 0; k < 6; k++) {
    digitalWrite(LED_PIN, LOW);
    ledcWrite(LEDC_CHANNEL, 10);
    delay(500);
    digitalWrite(LED_PIN, HIGH);
    ledcWrite(LEDC_CHANNEL, 0);
    delay(500);
  }
  server.on("/", HTTP_GET, handleRoot);
  server.on("/save", HTTP_POST, handleSave);
  server.begin();
}
```

Функція createWiFiPoint() створює точку доступу Wi-Fi на ESP32. Якщо пристрій вже підключений до Wi-Fi, функція спочатку відключає його, після чого створює нову точку доступу з SSID і паролем, визначеними в змінних apSSID і apPassword, виводить повідомлення про створення точки доступу, після чого блимає світлодіодом 5 разів для індикації. Потім відбувається налаштування веб-сервеу для обробки HTTP-запитів, тобто обробляє запит на кореневу сторінку ("/") за допомогою функції handleRoot та POST-запити на "/save" за допомогою функції handleSave, потім запускає веб-сервер.

Переходимо до основного циклу програми, код якого наведено нижче:

```
void loop() {
  btnControl();
  unsigned long currentMillis = millis();
  if (currentMillis - previousWiFiMillis >= wifiInterval) {
    previousWiFiMillis = currentMillis;
    if (!SETTINGS_CHANGE && (ssid.length() != 0 && password.length() != 0)){
      checkWiFi();
    }
  }
  if (currentMillis - previousMQTTMillis >= mqttInterval) {
    previousMQTTMillis = currentMillis;
    if (!SETTINGS_CHANGE && (ssid.length() != 0 && password.length() != 0)){
      reconnect();
    }
  }
  if (SETTINGS_CHANGE || (ssid.length() == 0 && password.length() == 0)) {
    server.handleClient();
  } else {
    if (client.connected() && streaming) {
      unsigned long currentMillis = millis();
      feedback = sendImage();
      sendText(feedback);
    }
    client.loop();
  }
  btnControl();
}
```

Головний цикл програми loop() виконується постійно та виконує такі дії:

– перевіряє стан кнопок за допомогою функції btnControl();

– перевіряє та відновлює підключення до Wi-Fi та MQTT серверу з певною частотою;

– коли МР у режимі налаштування або відсутні Wi-Fi налаштування, обробляє HTTP-запити через веб-сервер;

– коли підключення до MQTT є активним і ввімкнено передачу даних (streaming), відправляє зображення через MQTT, викликаючи функції sendImage() і sendText();

– викликає client.loop(), щоб підтримувати зв’язок з MQTT сервером.

Основні функції такі, як функції для отримання перетворення та передавання зображення, пересування МР, керування штативу панорамування та нахилу та деякі додаткові функції буде розглянуто детальніше нижче.

2.8.2 Функції для отримання, перетворення та передавання зображення

Найважливішими функціями для будь-якої системи відеонагляду є функції отримання, обробки та передачі відеосигналу.

Код функції для отримання, обробки та передачі зображення наведено нижче:

```
String sendImage() {
  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  if (!fb) {
    loger("Camera capture failed");
    return "Camera capture failed";
  }
```

```
        char *input = (char *)fb->buf;
        char output[base64_enc_len(3)];
        String imageFile = "";
        for (int i = 0; i < fb->len; i++) {
          base64_encode(output, (input++), 3);
          if (i % 3 == 0) imageFile += String(output);
        }
        int fbLen = imageFile.length();
        String clientId = "ESP32-";
        clientId += String(random(0xffff), HEX);
        if (client.connect(clientId.c_str(), mqtt_login.c_str(),
mqtt_password.c_str())) {
          client.beginPublish(MQTT_PUBLISH_TOPIC, fbLen, true);
          String str = "";
          for (size_t n = 0; n < fbLen; n = n + 2048) {
            if (n + 2048 < fbLen) {
              str = imageFile.substring(n, n + 2048);
              client.write((uint8_t*)str.c_str(), 2048);
            }
            else if (fbLen % 2048 > 0) {
              size_t remainder = fbLen % 2048;
              str = imageFile.substring(n, n + remainder);
              client.write((uint8_t*)str.c_str(), remainder);
            }
          }
          client.endPublish();
          esp_camera_fb_return(fb);
          return "";
        }
        esp_camera_fb_return(fb);
```

```
  return "failed, rc=" + client.state();
 }
```

Функція sendImage() захоплює зображення з камери ESP32, перетворює його в base64-формат і відправляє через MQTT. Спочатку захоплюється зображення за допомогою esp_camera_fb_get(). Якщо захоплення не вдається, повертається повідомлення про помилку. Якщо успішно, зображення кодується в base64 і зберігається в рядок imageFile. Потім ESP32 підключається до MQTT-сервера з унікальним ідентифікатором клієнта і відправляє зображення по частинах (по 2048 байтів). Після завершення публікації зображення звільняється пам'ять кадру. Якщо з'єднання не вдалося, повертається повідомлення про помилку

Також було додано функцію для зміни якості зображення, код якої наведено нижче:

```
void qualityChange(String quality) {
  sensor_t *s = esp_camera_sensor_get();
  if (s != NULL) {
    if (quality == "CIF") {
      s->set_framesize(s, FRAMESIZE_CIF);
    } else if (quality == "VGA") {
      s->set_framesize(s, FRAMESIZE_VGA);
    } else if (quality == "SVGA") {
      s->set_framesize(s, FRAMESIZE_SVGA);
    } else if (quality == "QVGA") {
      s->set_framesize(s, FRAMESIZE_QVGA);
    } else if (quality == "HQVGA") {
      s->set_framesize(s, FRAMESIZE_HQVGA);
    } else if (quality == "QQVGA") {
      s->set_framesize(s, FRAMESIZE_QQVGA);
```

```
    } else {
      loger("Failed to get sensor pointer");
    }
  }
}
```

2.8.3 Функції для пересування мобільного робота

Для забезпечення мобільності системи відеонагляду необхідно створити функції для керування шасі МР.

Функція motorControl() керує напрямком і швидкістю двох моторів на основі вказаного напряму руху (direct). Залежно від команди (наприклад, driveForward, driveBackward, driveStop, driveLeft, або driveRight), функція задає напрямок обертання моторів через цифрові виходи (motor1_IN і motor2_IN) і розраховує швидкість для кожного мотора (mspeed1, mspeed2). Швидкість визначається шляхом перетворення значення швидкості motorSpeed з діапазону 0–10 у діапазон 0–255 і застосовується через ШІМ-сигнал на виходи motor1_PWM та motor2_PWM. Код цієї функції наведено нижче:

```
void motorControl(String direct) {
  int mspeed1 = 0;
  int mspeed2 = 0;
  mspeed = map(motorSpeed, 0, 10, 0, 255);
  if (direct == "driveBackward") {
    digitalWrite(motor1_IN, 1);
    digitalWrite(motor2_IN, 1);
  } else {
    digitalWrite(motor1_IN, 0);
    digitalWrite(motor2_IN, 0);
```

```
  }
  if (direct == "driveForward") {
    mspeed1 = mspeed;
    mspeed2 = mspeed1;
  } else if (direct == "driveBackward") {
    mspeed1 = 255 - mspeed;
    mspeed2 = mspeed1;
  } else if (direct == "driveStop") {
    mspeed1 = 0;
    mspeed2 = 0;
  } else if (direct == "driveLeft") {
    mspeed1 = mspeed;
    mspeed2 = mspeed * 0.2;
  } else if (direct == "driveRight") {
    mspeed1 = mspeed * 0.2;
    mspeed2 = mspeed;
  }
  analogWrite(motor1_PWM, mspeed1);
  analogWrite(motor2_PWM, mspeed2);
}
```

2.8.4 Функції керування штативом панорамування та нахилу

Для можливості зручного та швидкого повороту камери використовується FPV-штатив (рис. 2.3). Функція servoControl() керує положенням двох серводвигунів на основі команди (direct). Вона змінює вертикальне положення серводвигуна (servo_v_pos) при командах "tiltUp" або "tiltDown" та горизонтальне положення серводвигуна (servo_h_pos) при командах "tiltLeft" або "tiltRight", забезпечуючи плавне переміщення в межах заданих діапазонів (від 20 до 150). Команда "tiltCenter" встановлює

серводвигуни в центральну позицію. Після обробки кожної команди відповідні серводвигуни оновлюються через методи write(). Код даної функції наведено нижче:

```
void servoControl(String direct) {
  if (direct == "tiltUp" && servo_v_pos > 20) {
    servo_v_pos = servo_v_pos - 10;
  } else if (direct == "tiltDown" && servo_v_pos < 150) {
    servo_v_pos = servo_v_pos + 10;
  } else if (direct == "tiltLeft" && servo_h_pos < 150) {
    servo_h_pos = servo_h_pos + 10;
  } else if (direct == "tiltRight" && servo_h_pos > 20) {
    servo_h_pos = servo_h_pos - 10;
  } else if (direct == "tiltCenter") {
    servo_v_pos = 80;
    servo_h_pos = 90;
  }
  servo_v.write(servo_v_pos);
  servo_h.write(servo_h_pos);
  delay(50);
}
```

2.8.5 Додаткові функції мобільного робота

Розроблений МР повинен мати декілька режимів роботи:

– режим відладки (debug mode) – цей режим необхідний при сервісному обслуговуванні. У даному режимі вимикаються функції для керування шасі та ШНП, а також вмикаються функції для відображення логів;

– режим налаштувань (settings mode) – цей режим потрібен у випадку, коли користувачу необхідно змінити налаштування МР, такі як Wi-Fi або MQTT;

– стандартний режим – у цьому режимі працюють функції керування шасі та ШНП, але вимкнено функції відображення логів.

Для переходу між режимами використовується певна комбінація натискання кнопки, ця комбінація відслідковується за допомогою функції btnControl, код якої наведено нижче:

```
void btnControl(){
  int reading = digitalRead(buttonPin);
  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;
      if (buttonState == LOW) {
        if (!buttonPressed) {
          buttonPressed = true;
          buttonPressStartTime = millis();
        }
      } else {
        if (buttonPressed) {
          unsigned long pressDuration = millis() - buttonPressStartTime;
          buttonPressed = false;
          if (pressDuration < 1000) {
            handleShortPress();
          } else if (pressDuration >= 2000 && pressDuration < 10000) {
            handleLongPress();
```

```
        } else if (pressDuration >= 20000) {
          resetToFactorySettings();
          resetPressState();
        }
      }
    }
  }
  lastButtonState = reading;
}
```

Ця функція викликає подію handleShortPress, якщо кнопка була натиснута менше 1-єї секунди, handleLongPress – якщо більше 2-х секунд, та менше 10-ти секунд, або resetToFactorySettings, коли кнопка натиснута більше 20-ти секунд.

Події handleShortPress та handleLongPress у свою чергу перевіряють поточний індекс вказівника у масиві, якщо він менше 11, тоді виконується запис 0 або 1, відповідно, у наступний елемент масиву pressSequence, після чого викликається функція checkCombination, що у свою чергу перевіряє яку комбінацію було введено користувачем та викликає відповідну функцію для зміни режиму роботи. Лістинг коду наведено нижче:

```
void checkCombination() {
  // Комбінація для режиму відкладки: ". . . . . _ . . . . ."
  int debugModePattern[11] = {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0};
  // Комбінація для режиму налаштувань: ". . . _"
  int settingsModePattern[4] = {0, 0, 0, 1};
  if (pressIndex == 11 && comparePatterns(pressSequence, debugModePattern, 11)) {
    enterDebugMode();
```

```
      resetPressState();
      return;
    }
    if (pressIndex == 4 && comparePatterns(pressSequence, settingsModePattern, 4)) {
      enterSettingsMode();
      resetPressState();
      return;
    }
    // Якщо введено більше 11 натискань, скидаємо послідовність
    if (pressIndex >= 11) {
      resetPressState();
    }
  }
  // Функція порівняння послідовності з еталонною
  bool comparePatterns(int *inputPattern, int *targetPattern, int length) {
    for (int i = 0; i < length; i++) {
      if (inputPattern[i] != targetPattern[i]) {
        return false;
      }
    }
    return true;
  }
```

Розглянемо функцію для входу у режим відладки (debug mode), лістинг коду наведено нижче:

```
  void enterDebugMode() {
    // Візуальна індикація входу у режим відладки
    for (int i = 0; i < 2; i++){
```

```
    for (int i = 0; i < 100; i += 5) {
      ledcWrite(LEDC_CHANNEL, i);
      delay(100);
    }
    for (int i = 100; i > 0; i -= 5) {
      ledcWrite(LEDC_CHANNEL, i);
      delay(100);
    }
  }
  debugControl();
}
```

Ця функція після виконання візуальної індикації переходу у режим відладки викликає функцію debugControl, код якоъ наведено нижче:

```
void debugControl(){
  DEBUG = !DEBUG;
  loger(DEBUG ? "Debug mode activated." : "Debug mode deactivated.");
  EEPROM.begin(512);
  EEPROM.write(196, DEBUG);
  EEPROM.commit();
  EEPROM.end();
  delay(1000);
  ESP.restart();
}
```

Функція debugControl інвертує значення глобальної змінної DEBUG, виводить відповідний лог та записує значення змінної DEBUG у EEPROM пам’ять, після чого виконує перезавантаження мікроконтролера MP.

Подібним чином працює подія enterSettingsMode, лістинг коду якої наведено нижче:

```
void enterSettingsMode() {
  // Візуальна індикація
  for (int i = 0; i < 100; i += 5) {
    ledcWrite(LEDC_CHANNEL, i);
    delay(100);
  }
  for (int i = 100; i < 0; i -= 5) {
    ledcWrite(LEDC_CHANNEL, i);
    delay(100);
  }
  settingsControl();
}
```

Ця подія також викликає відповідну функцію settingsControl, що виконує подібні до debugControl кроки, нижче наведено код функції:

```
void settingsControl(){
  SETTINGS_CHANGE = !SETTINGS_CHANGE;
  loger(SETTINGS_CHANGE ? "Change settings mode activated." : "Change settings mode deactivated.");
  EEPROM.begin(512);
  EEPROM.write(197, SETTINGS_CHANGE);
  EEPROM.commit();
  EEPROM.end();
  delay(1000);
  ESP.restart();
}
```

Також у лістингу коду функції btnControl можна побачити подію resetToFactorySettings, що необхідна для скидання налаштувань МР до заводських налаштувань, код даної події наведено нижче:

```
void resetToFactorySettings() {
  loger("Resetting to factory settings...");
 // Індикація скидання до налаштувань за замовчуванням
 for (int i = 0; i < 10; i++) {
  ledcWrite(LEDC_CHANNEL, 55);
  delay(500);
  ledcWrite(LEDC_CHANNEL, 0);
  delay(500);
 }
 factoryBoot();
}
```

Ця подія викликає функцію factoryBoot, що у свою чергу видаляє всі налаштування МР з EEPROM пам’яті, код функції наведено нижче:

```
void factoryBoot(){
 ssid = "";
 password = "";
 mqtt_server = "";
 mqtt_port = 0;
 mqtt_login = "";
 mqtt_password = "";
 saveCredentials();
 loger("FACTORY BOOT. Data saved");
 delay(1000);
 ESP.restart();
```

```
}
```

У цій функцію бачимо, що використовується функція saveCredentials, що необхідна для зручного збереження великої кількості даних у пам’яті EEPROM, лістинг коду наведено нижче:

```
void saveCredentials() {
  EEPROM.begin(512);
  EEPROM.writeString(0, ssid);
  EEPROM.writeString(32, password);
  EEPROM.writeString(64, mqtt_server);
  EEPROM.writeString(96, String(mqtt_port));
  EEPROM.writeString(128, mqtt_login);
  EEPROM.writeString(160, mqtt_password);
  EEPROM.commit();
  EEPROM.end();
}
```

Також у програмі використовується функція для завантаження даних з енергонезалежної пам’яті MP loadCredentials, код якої наведено нижче:

```
void loadCredentials() {
  EEPROM.begin(512);
  ssid = EEPROM.readString(0);
  password = EEPROM.readString(32);
  mqtt_server = EEPROM.readString(64);
  mqtt_port = EEPROM.readString(96).toInt();
  mqtt_login = EEPROM.readString(128);
  mqtt_password = EEPROM.readString(160);
  DEBUG = EEPROM.read(196);
```

```
  SETTINGS_CHANGE = EEPROM.read(197);
  EEPROM.end();
}
```

Ще однією з не менш важливих функцій є функція loger, що виконує ті самі функції, що і стандартний метод Serial.println(), тобто виводить текст у послідовний порт, але ця функція працює лише за умови, що МР знаходиться у режимі відладки. Код даної функції наведено нижче:

```
void loger(const char* format, ...) {
  if (DEBUG == true) {
    char buffer[256]; // Буфер для форматованого рядка
    // Використовуємо stdarg.h для работи зі змінною кількістю аргументів
    va_list args;
    va_start(args, format);
    vsnprintf(buffer, sizeof(buffer), format, args);
    va_end(args);
    Serial.println(buffer);
  }
}
```

2.9 Розроблення користувацької та серверної частини

Після етапу розроблення програмного забезпечення МР, що виконує команди, надіслані користувачем, можемо переходити до створення програми, що забезпечить зручне віддалене керування мобільним роботом [34]. Доцільним є створення веб-застосунку з адаптивним дизайном, це дозволить користувачам з будь-якої платформи керувати МР без встановлення відповідного застосунку.

У якості мови програмування для створення веб-застосунку було обрано Python [35] та фреймворк Django [36], додатковою мовою програмування для реалізації функцій керування МР було обрано JavaScript [37].

Мову програмування Python було обрано через те, що це найдинамічніша мова програмування, згідно з опитуванням від Stack Overflow, станом на 2020 рік близько 13,5 % користувачів задавали питання саме щодо цієї мови програмування. Також ця мова забезпечує високу швидкість розробки, завдяки лаконічній мові і відмінним бібліотекам.

Обраний фреймворк Django є набором перевірених інструментів для створення всіх компонентів веб-додатків, а за даними SimilarTech, 63% сайтів на Python побудовані на цьому фреймворку [38].

JavaScript було обрано через його можливість реалізувати інтерактивність для веб-сторінки, він дозволяє змінювати, додавати та видаляти елементи на сторінці без перезавантаження. JavaScript також забезпечує можливість реагувати на події, такі як кліки, наведення курсору або введення даних користувачем [39].

2.9.1 Опис серверної частини програми

Розпочнемо розробку зі створення серверної частини веб-застосунку.

Фреймворк Django використовує шаблон проектування «Модель-Вид-Контролер»(MVC, Model-View-Controller), тому розробку потрібно почати саме зі створення моделей.

Модель представляє собою дані та реагує на команди контролера, змінюючи свій стан, іншими словами моделі це таблиці бази даних (БД) веб-застосунку. Для правильної роботи веб-застосунку та створення окремих «кабінетів» для кожного з користувачів було створено ER-діаграму необхідної бази даних, рис. 2.22.

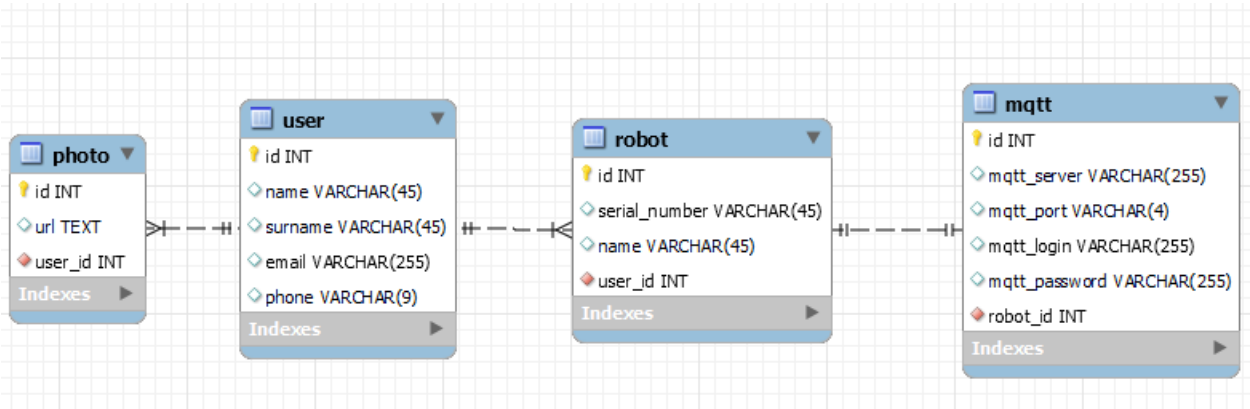


Рисунок 2.22 – ER-діаграма бази даних для веб-застосунку мобільної системи відеонагляду

На рисунку бачимо, що повинно бути мінімум 4 моделі. Модель User повинна описувати данні користувачів системи, Robot – данні про серійний номер МР, назву МР та його налаштування MQTT, Mqtt – данні необхідні для підключення до MQTT-серверу, а модель Photo – для зберігання посилань на фото, зроблених певним користувачем. Також видно, що один користувач може мати декілька роботів та декілька фото, а кожен робот має свої налаштування. Таким чином, можна описати моделі мовою Python так, як наведено додатку Е.

Наступним кроком необхідно створити контролери для відображення даних з БД на сторінці. Почнемо з найважливішого сторінок реєстрації та авторизації.

Лістинг коду контролера для реєстрації користувача на ведено нижче:

```
def register(request):
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('home')
    else:
        form = CustomUserCreationForm()
    return render(request, 'remoteapp/registration_page.html', {'form':
form})
```

Цей контролер відображає на сторінці форму CustomUserCreationForm для реєстрації користувача, коли користувач введе коректні данні для реєстрації, його буде переведено на сторінку керування МР.

Контролер user_login відповідає за відображення форми UserLoginForm для авторизації у системі, код контролера наведено нижче:

```
def user_login(request):
    if request.method == 'POST':
        form = UserLoginForm(data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('select_default_robot')
    else:
        form = UserLoginForm()
    return render(request, 'remoteapp/autorization_page.html', {'form':
form})
```

Якщо користувач введе правильні дані для авторизації, його буде переадресовано на сторінку керування МР.

Також важливим контролером є user_logout, що дозволяє користувачу вийти з поточного сеансу, лістинг коду наведено нижче:

```
@login_required
def user_logout(request):
    logout(request)
    return redirect('login')
```

У даній функції використовується декоратор @login_required, що не дозволяє відображення поточної сторінки для неавторизованого користувача.

Наступним розглянемо контролер select_default_robot, який використовується при авторизації у системі та відповідає за вибір робота за замовчуванням, а у разі відсутності роботів у користувача перенаправляє на сторінку додавання робота. Код даного контролера наведено нижче:

```
@login_required
def select_default_robot(request):
```

```
    try:
        select_robot = Robot.objects.get(user=request.user,
default_selected=True)
    except ObjectDoesNotExist:
        return redirect('add_new_robot')
    else:
        return redirect('home', select_robot.pk)
```

На сторінці керування МР повинна бути можливість переключення між різними роботами користувача, тому додамо відповідний контролер select_robot_in_root, код якого наведено нижче:

```
@login_required
def select_robot_in_root(request, robot_id):
    user_agent = request.user_agent
    if user_agent.is_mobile:
        return redirect('index_mobile', robot_id)
    else:
        return redirect('home', robot_id)
```

Окрім того, що користувач може обрати необхідного робота, також функція виконує перевірку чи користувач користується застосунком з телефону та перенаправляє користувача на відповідну сторінку в залежності від пристрою.

Для відображення головної сторінки веб-застосунку використовуються контролери index та index_mobile, що перевіряють та перенаправляють користувача на відповідну сторінку в залежності від пристрою, перевіряють чи існує підключення до MQTT, чи має користувач роботів, а також чи авторизовано користувача. Лістинги коду контролерів наведено нижче:

```
@login_required
def index(request, robot_id):
    user_agent = request.user_agent
    if user_agent.is_mobile:
        return redirect('index_mobile', robot_id)
    else:
        mqtt_settings = Mqtt.objects.get(robot__user=request.user,
robot=robot_id)
        robots = Robot.objects.filter(user=request.user)
        select_robot = Robot.objects.get(pk=robot_id)
        if mqtt_settings:
```

```
            mqtt_data = mqtt_settings.as_dict()
        else:
            mqtt_data = {}

        if not request.user.is_authenticated:
            return redirect('login')
        else:
            return render(request, 'remoteapp/index_desktop.html',
                          {'mqtt_data_json': json.dumps(mqtt_data),
                           'robots': robots, 'select_robot': select_robot})

@login_required
def index_mobile(request, robot_id):
    user_agent = request.user_agent
    if not user_agent.is_mobile:
        return redirect('home', robot_id)
    else:
        try:
            mqtt_settings = Mqtt.objects.get(robot__user=request.user,
robot=robot_id)
        except ObjectDoesNotExist:
            return redirect('add_mqtt_connection', robot_id)
        robots = Robot.objects.filter(user=request.user)
        select_robot = Robot.objects.get(pk=robot_id)
        if not select_robot:
            redirect('add_new_robot')
        if mqtt_settings:
            mqtt_data = mqtt_settings.as_dict()
        else:
            mqtt_data = {}
        if not request.user.is_authenticated:
            return redirect('login')
        else:
            return render(request, 'remoteapp/index_mobile.html',
                          {'mqtt_data_json': json.dumps(mqtt_data),
                           'robots': robots, 'select_robot': select_robot})
```

Також ці контролери передають на сторінку данні про MQTT, що будуть використані для підключення до MQTT за допомогою програми мовою JavaScript.

Решта контролерів необхідні для реалізації додаткових функцій, таких як відображення сторінки загальних налаштувань, сторінки перегляду та зміни особистих даних користувача, сторінок для перегляду та зміни налаштувань MQTT для роботів користувача, сторінок для перегляду та змін налаштувань мобільних роботів, сторінки для перегляду галереї користувача, тощо. Лістинг коду цих контролерів можна переглянути у додатку Е.

Найважливішою частиною веб-застосунку є сторінка керування МР. Для розроблення функціоналу для керування було написано код програми мовою

JavaScript, додаток Е. Для початку потрібно отримати всі необхідні змінни, код наведено нижче:

```
var keyboardTopic = "robot/keyboard";
var flashTopic = "camera/flash";
var qualityTopic = "camera/quality";
var mqttDataElement = document.getElementById('mqtt-data');
var mqttData = JSON.parse(mqttDataElement.textContent);
var mqttServer = mqttData.mqtt_server;
var mqttPort = mqttData.mqtt_port;
var mqttLogin = mqttData.mqtt_login;
var mqttPassword = mqttData.mqtt_password;
```

Дані для підключення до MQTT отримаються з прихованого HTML елементу з ідентифікатором «mqtt-data» та розпаковуються з формату JSON.

Наступним кроком створюється підключення до MQTT, фрагмент коду наведено нижче:

```
var client = new Paho.MQTT.Client(mqttServer, mqttPort, "clientId" + new
Date().getTime());

client.connect({
    onSuccess: function () {
       console.log(mqttData);
       console.log("Connected to MQTT server");
       client.subscribe("camera/test");
    }
});
```

Цей код створює MQTT-клієнта за допомогою бібліотеки Paho MQTT, підключається до MQTT-сервера і підписується на тему «camera/test». Після успішного підключення в консоль виводяться дані змінної mqttData і повідомлення про те, що клієнт підключився до сервера.

Для обробки події втрати з’єднання з MQTT-сервером було додано функцію, код якої наведено нижче:

```
client.onConnectionLost = function (responseObject) {
    if (responseObject.errorCode !== 0) {
       console.log("Connection lost:", responseObject.errorMessage);
```

```
    }
};
```

Якщо з’єднання втрачається і є помилка (коли responseObject.errorCode не дорівнює 0), в консоль виводиться повідомлення про втрату з’єднання разом з текстом помилки (responseObject.errorMessage).

Для відображення отриманого зображення з MQTT-серверу використовується функція, що наведена нижче:

```
client.onMessageArrived = function (message) {
    var imageData = message.payloadString;
    if (imageData != undefined && imageData.length > 0) {
       $("#streamedImage").attr("src", imageData);
       var isChecked = $(".stream-ctrl
input[type='checkbox']").prop("checked");
       if (!isChecked) {
          $("#streamedImage").attr("src", "https://img.freepik.com/premium-
vector/no-signal-old-tv-screen_268834-925.jpg");
       }
    }
};
```

Цей код обробляє події отримання повідомлень через MQTT. Коли надходить повідомлення, дані з нього (message.payloadString) зберігаються в змінну imageData. Якщо ці дані не порожні, вони використовуються для оновлення зображення на веб-сторінці через елемент з ідентифікатором «#streamedImage». Якщо прапорець у блоці «.stream-ctrl» не відмічений (checkbox вимкнений), зображення змінюється на стандартну картинку «No signal» за заданим URL.

Для відправки керуючих команд було створено функцію, код якої наведено нижче:

```
function sendCommand(topic, command) {
    var message = new Paho.MQTT.Message(command);
    message.destinationName = topic;
    client.send(message);
}
```

Ця функція надсилає команду через MQTT. Вона приймає два параметри: topic (тема, до якої буде надіслано повідомлення) і command (текст

повідомлення). Створюється об’єкт Paho.MQTT.Message з вмістом команди, а потім визначається його цільова тема (message.destinationName = topic). Після цього повідомлення відправляється на MQTT-сервер через клієнта client.send(message).

Для реалізації керування кнопками у інтерфейсі веб-застосунку було створено обробники подій для відправки команд через MQTT при натисканні кнопок. При натисканні будь-якої кнопки керування шасі або ШНП викликається функція sendCommand(), яка надсилає відповідну команду на MQTT-сервер. Кнопки для керування положенням камери надсилають повідомлення: tiltUp, tiltLeft, center, tiltRight, tiltDown. Кнопки для керування шасі робота надсилають повідомлення: driveForward, driveLeft, driveStop, driveRight, driveBackward.

Кожен обробник події mousedown викликає функцію sendCommand() з відповідною темою (keyboardTopic) і командою, що вказує на бажану дію (рух, нахил або поворот).

Код обробників подій для керування шасі або ШНП МР наведено нижче:

```
$("#upButton").mousedown(function () { sendCommand(keyboardTopic, "tiltUp");
});
$("#leftButton").mousedown(function () { sendCommand(keyboardTopic,
"tiltLeft"); });
$("#centerButton").mousedown(function () { sendCommand(keyboardTopic,
"center"); });
$("#rightButton").mousedown(function () { sendCommand(keyboardTopic,
"tiltRight"); });
$("#downButton").mousedown(function () { sendCommand(keyboardTopic,
"tiltDown"); });

$("#forwardDriveButton").mousedown(function () { sendCommand(keyboardTopic,
"driveForward"); });
$("#leftDriveButton").mousedown(function () { sendCommand(keyboardTopic,
"driveLeft"); });
$("#stopDriveButton").mousedown(function () { sendCommand(keyboardTopic,
"driveStop"); });
$("#rightDriveButton").mousedown(function () { sendCommand(keyboardTopic,
"driveRight"); });
$("#backwardDriveButton").mousedown(function () { sendCommand(keyboardTopic,
"driveBackward"); });
```

Для того щоб після відпускання кнопки робот зупинився було додано обробник:

```
$(".remote-ctrls .btn").mouseup(function () { sendCommand(keyboardTopic,
"driveStop"); });
```

Також користувач повинен мати можливість змінювати якість відеопотоку, швидкість руху МР, яскравість спалаху та увімкнення/вимкнення трансляції, для цього було додано такі обробники:

```
$("#qualityDropdown a").click(function () {
    var quality = $(this).text();
    sendCommand(qualityTopic, "quality:" + quality);
});

$("#motor-speed").change(function () {
    var speed = $(this).val();
    sendCommand(keyboardTopic, "speed:" + speed);
});

$("#flash-brightness").change(function () {
    var brightness = $(this).val();
    sendCommand(flashTopic, "brightness:" + brightness);
});

$(".stream-ctrl input[type='checkbox']").change(function () {
    var isChecked = $(this).prop("checked");
    var command = isChecked ? "start_streaming" : "stop_streaming";
    sendCommand("camera/test/get", command);
});
```

Для користувача, що користується системою відеонагляду з персонального комп’ютера може бути зручно керувати МР за допомогою клавіатури, для цього було додано такий обробник:

```
var keyPressed = {};

$(document).keydown(function (e) {
    if (!keyPressed[e.which]) {
        keyPressed[e.which] = true;
        switch (e.which) {
            case 87: // W
                sendCommand(keyboardTopic, "driveForward");
                break;
            case 38: // Up arrow
                sendCommand(keyboardTopic, "tiltUp");
                break;
            case 65: // A
                sendCommand(keyboardTopic, "driveLeft");
                break;
            case 37: // Left arrow
                sendCommand(keyboardTopic, "tiltLeft");
                break;
            case 83: // S
                sendCommand(keyboardTopic, "driveBackward");
```

```
            break;
         case 40: // Down arrow
            sendCommand(keyboardTopic, "tiltDown");
            break;
         case 68: // D
            sendCommand(keyboardTopic, "driveRight");
            break;
         case 39: // Right arrow
            sendCommand(keyboardTopic, "tiltRight");
            break;
      }
   }
}).keyup(function (e) {
   keyPressed[e.which] = false;
   switch (e.which) {
      case 87: // W
         sendCommand(keyboardTopic, "driveStop");
         break;
      case 65: // A
         sendCommand(keyboardTopic, "driveStop");
         break;
      case 83: // S
         sendCommand(keyboardTopic, "driveStop");
         break;
      case 68: // D
         sendCommand(keyboardTopic, "driveStop");
         break;
   }
});
```

За допомогою, цього обробника користувач має можливість керувати шасі мобільного робота натискаючи клавіші W, A, S, D та керувати нахилом або поворотом камери натискаючи клавіші зі стрілками.

2.9.2 Створення графічного інтерфейсу

Одним з важливих етапів є створення зручного та ергономічного інтерфейсу для керування системою мобільного відеонагляду. Інтерфейс повинен мати адаптивний дизайн для мобільних пристроїв та комп’ютерів.

Для розроблення користувацького інтерфейсу пропонується використовувати мову розмітки HTML та таблиці стилів CSS, такий підхід часто використовується для створення веб-сторінок для веб-застосунків.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об’єкти, такі як інтерактивні форми, можуть бути вбудовані у візуалізовану сторінку. HTML

надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи [40].

CSS - це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних, як HTML [41]. Правило «@media» дозволяє вказати тип носія, для якого буде застосовано вказаний стиль, часто це називають медіа-запитами.

Таким чином можна розробити окремі HTML-документи для керування МР з різними CSS-стилями, а для сторінок реєстрації, авторизації та налаштувань, де верстка не сильно відрізняється, можна адаптовувати дизайн лише за допомогою медіа-запитів CSS.

Розроблення інтерфейсу почнемо зі сторінки керування МР, що є головною сторінкою користувацької частини мобільної системи відеонагляду, в результаті було отримано вигляд сторінки, як показано на рис. 2.23.

Рисунок 2.23 – Вигляд головної сторінки користувацького інтерфейсу для персонального комп’ютера

Інтерфейс містить велике вікно для відображення відеопотоку, кнопки для керування шасі МР праворуч та кнопки для керування поворотом та нахилом камери ліворуч. Посередині розміщено два повзунки для регулювання яскравістю спалаху та швидкістю руху МР. У лівій верхній частині бачимо інформацію про назву системи, ім’я користувача та статус підключення до MQTT серверу. У правій верхній частині розміщено кнопку для переходу у налаштування, перемикач для увімкнення та вимкнення трансляції та дві кнопки, натиснувши на які користувач може обрати якість відеопотоку та обрати певного робота, якщо їх декілька у користувача. Також у правій нижній частині вікна для трансляції можна помітити логотип об’єктиву, натиснувши на який користувач може зробити фото.

Якщо відкрити цю сторінку на мобільному пристрої, отримаємо такий вигляд, рис. 2.24.

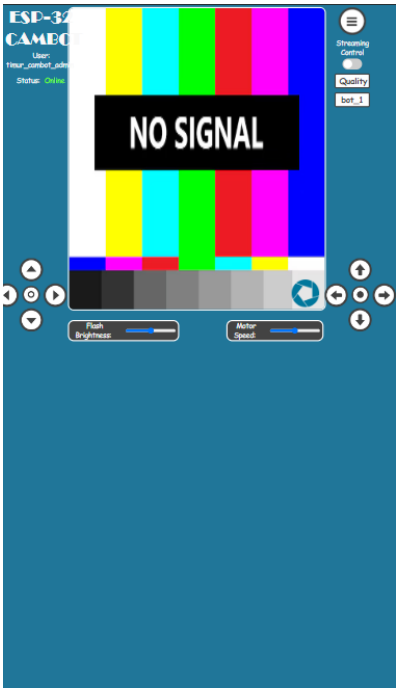


Рисунок 2.24 – Вигляд сторінки керування МР на мобільному пристрої до адаптації

Такий інтерфейс є незручним для користувача, тому було створено окрему сторінку для мобільних пристроїв, що наведено на рис. 2.25.

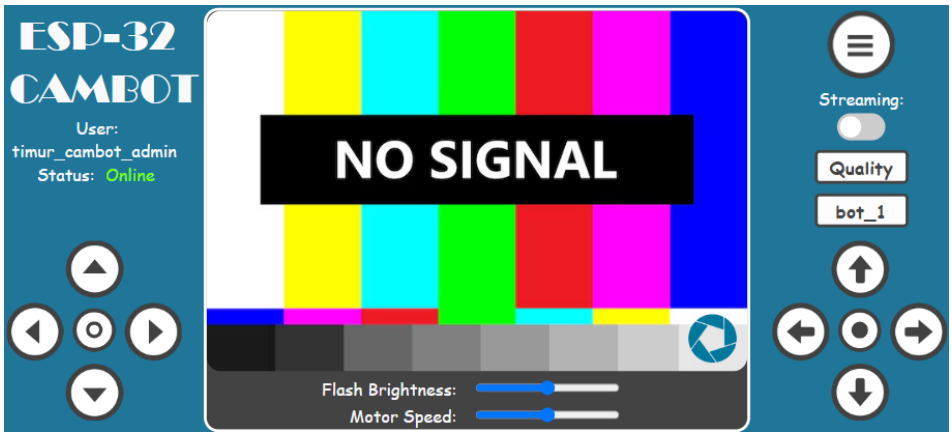


Рисунок 2.25 – Вигляд адаптованої сторінки керування МР на мобільному пристрої

Бачимо, що елементи керування залишились такими ж, як було на рис. 2.23, але компактне та ергономічне розміщення дозволить користувачу зручно керувати МР через телефон. Дизайн було адаптовано саме під горизонтальне розміщення смартфону для зручності. Якщо користувач змінить положення телефону на вертикальне, то побачить сторінку з повідомленням про необхідність горизонтального положення телефону, рис. 2.26.

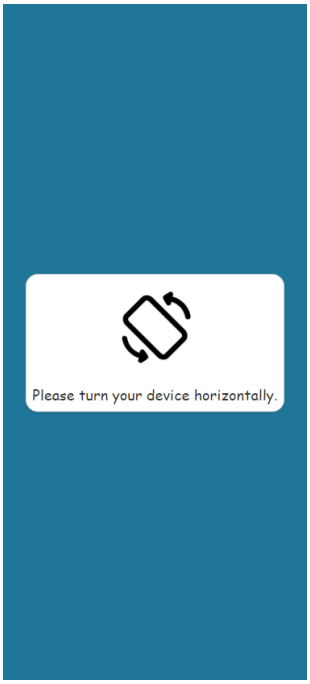


Рисунок 2.26 – Повідомлення для користувача про необхідність горизонтального розміщення смартфону

Це було досягнуто за допомогою JavaScript коду, який наведено нижче:

```
window.onload = function () {
    handleOrientationChange();
};

function handleOrientationChange() {
    var isPortrait = window.matchMedia("(orientation: portrait)").matches;
    var isLandscape = window.matchMedia("(orientation: landscape)").matches;
    var robotIdPattern = /\/remoteapp\/(index_mobile|index)\/\d+\/$/;

    if (isPortrait && robotIdPattern.test(window.location.pathname)) {
        window.location.href = "/remoteapp/rotate_device/";
    } else if (isLandscape && !robotIdPattern.test(window.location.pathname))
{
        window.location.href = "/remoteapp/select_default_robot/";
    }
}

screen.orientation.addEventListener("change", handleOrientationChange);
```

Цей код виконує перенаправлення сторінки залежно від орієнтації екрана (портрет або ландшафт) і відповідного URL-шляху. При завантаженні сторінки викликається функція handleOrientationChange(), яка перевіряє орієнтацію екрану. Якщо пристрій у портретній орієнтації і URL відповідає певному шаблону (шлях /remoteapp/index_mobile/ або /remoteapp/index/ з ідентифікатором робота), користувача перенаправляє на сторінку /remoteapp/rotate_device/. Якщо пристрій у ландшафтній орієнтації, але URL не відповідає цьому шаблону, користувач перенаправляється на сторінку /remoteapp/select_default_robot/. Додатково подія зміни орієнтації екрана відслідковується через screen.orientation.addEventListener, щоб перенаправлення виконувалося автоматично при зміні орієнтації.

Решту виглядів інтерфейсу можна побачити у керівництві користувача, додаток Д.

2.10 Висновки до розділу 2.

У другому розділі роботи було проведено обґрунтування вибору основних компонентів МСВ, зокрема плат керування та драйверів двигунів,

тощо. Було розроблено схеми підключення компонентів МР та алгоритми роботи програмного забезпечення для керування роботом і передачі відеопотоку. Результати досліджень показали, що обрані компоненти та технології забезпечують достатню якість відеопотоку та надійність роботи системи. Таким чином, було створено прототип МСВ за процесами на виробництві на базі ESP32-CAM, який відповідає поставленим вимогам та може бути використаний для ефективного відеоспостереження.

3 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

3.1 План-програма експерименту

Перед проведенням експериментів необхідно визначити тему, методику, цілі експериментів та інші характеристики експериментів, для цього розробляється план-програма експериментів.

Темою дослідження є порівняння розробленої системи мобільного відеонагляду на базі ESP32-CAM з аналогами.

Робоча гіпотеза: Ефективне використання мікроконтролеру для передачі потокового відео та керування мобільною платформою можливе за допомогою програмного забезпечення та оптимальних налаштувань.

Методика експерименту наступна:

– пошук аналогів до розробленої системи відеонагляду;

– підготовка мобільної системи відеонагляду на базі ESP32-CAM;

– запуск веб-серверу та перевірка доступу до МСВ через користувацький інтерфейс;

– тестування часу автономної роботи мобільного робота;

– тестування на відстані для оцінки стабільності передачі відеопотоку;

– тестування зручності керування МСВ;

– Порівняння коштовності системи.

Необхідні матеріали, прилади, установки:

– документація або технічні данні про аналогічну систему;

– мобільна система відеонагляду на базі ESP32-CAM;

– персональний комп’ютер з підключенням до мережі Інтернет.

Виконавцем експерименту є тестувальник, що виконує тести та аналізує результати.

Календарний план робіт:

- день 1 – пошук аналогів та їх технічних даних для порівняння;
- день 2-4 – тестування автономності МСВ;
- день 5 – тестування на відстані та оптимізація параметрів;
- день 6 – тестування зручності керування МСВ;
- день 7 – порівняння коштовності МСВ;
- день 8 – аналіз результатів та підготовка звіту.

3.2 Проведення експериментів

Для порівняння було обрано мобільну систему з камерою на базі Raspberry Pi 3 Model B. Така система має найбільш подібний функціонал, що забезпечить більш об’єктивні результати тестування.

3.2.1 Тестування автономності МСВ.

Для початку необхідно провести розрахунок теоретичного часу автономної роботи систем. Для цього необхідно визначити потужність споживання окремих компонентів МР (табл.3.1) за формулою:

$$P = U \cdot I, \tag{3.1}$$

де P – потужність споживання окремого компонента МР, Вт,

U – напруга живлення компонента МР, В,

I – робочий струм компонента МР, А.

Таблиця 3.1 – Необхідні характеристики компонентів систем для розрахунку потужності споживання окремих МР

Компонент	Струм споживання, А	Напруга живлення, В	Потужність споживання, Вт
ESP32-CAM	0,31	5	1,55
Raspberry Pi 3 Model B	2,5	5	12,5
L298N	2	14,8	29,6
	2	12,6	25,2
SG90	0,8	5	4

Таким чином, можемо розрахувати загальну потужність споживання для розробленого МР за формулою:

$$P = \sum P_i, \tag{3.2}$$

де P – загальна потужність споживання МР, Вт,

P_i – потужність споживання окремого компоненту МР, Вт.

Також відомо, що МР на базі ESP32-CAM оснащено АКБ з ємністю 3500 мАг та напругою 12,6 В, а МР на базі Raspberry Pi 3 Model B оснащено АКБ ємністю 3500 мАг та напругою 14,8 В.

Також необхідно розрахувати загальний струм споживання МР за формулою 3.3:

$$I = \frac{P}{U}, \tag{3.3}$$

де I – загальний струм споживання МР, А,

P – загальна потужність споживання МР, Вт,

U – напруга живлення МР, В.

Таким чином отримаємо, що загальний струм споживання МР на базі ESP32-CAM становить:

$$I_{ESP} = \frac{1{,}5 + 25{,}2 + 4 \cdot 2}{12{,}6} = 2{,}76 \text{ А},$$

а загальний струм споживання МР на базі Raspberry Pi 3 Model B становить:

$$I_{Rasp} = \frac{12{,}5 + 29{,}6}{14{,}8} = 2{,}84 \text{ А}.$$

Отримавши усі необхідні данні, можемо розрахувати приблизний час автономної роботи МР за формулою 3.4:

$$t = \frac{C}{I}, \tag{3.4}$$

де *t* – час автономної роботи, г,

C – ємність АКБ, Аг,

I – загальний струм споживання МР, А.

Підставивши всі змінні у формулу і отримаємо час автономної роботи МР на базі ESP32-CAM:

$$t_{ESP} = \frac{3{,}5}{2{,}76} = 1{,}27 \text{ годин}.$$

При чому, час автономної роботи МР на базі Raspberry Pi 3 Model B становить:

$$t_{Rasp} = \frac{3{,}5}{2{,}84} = 1{,}23 \text{ годин}.$$

Якщо перевести ці значення у хвилини, виходить, що час автономної роботи розробленого МР становить 76 хвилин, а час роботи МР на базі Raspberry Pi становить 72 хвилини, що на дві хвилини менше.

Перейдемо до тестування реальної тривалості автономної роботи кожної з МСВ. Експеримент було проведено 10 разів для отримання середніх результатів, табл. 3.2.

Таблиця 3.2 – Результати експериментів для визначення часу автономної роботи систем

№	Час роботи МР на базі ESP32-CAM, хв	Час роботи МР на базі Raspberry Pi, хв
1	78	81
2	80	75
3	85	77
4	103	108
5	80	82
6	82	76
7	79	85
8	75	77
9	84	76
10	89	79
Середнє значення, хв		
83,5		81,6

Бачимо, що різниця складає, приблизно 2 хвилини, але МР на базі ESP32-CAM має більший час автономної роботи.

3.2.2 Тестування дальності стабільної передачі відеопотоку.

Варто зауважити, що дальність стабільної передачі ввдеопотоку буде вимірюватись від Wi-Fi роутера до МР з камерою. Експеримент буде проведено 10 разів для отримання найбільш об’єктивних результатів. Результати експериментів наведено у табл. 3.3. Бачимо, що дальність стабільної передачі даних у МР на базі ESP32-CAM більша на 3,5 метри, така дальність обумовлена наявністю у даного робота зовнішньої антени.

Таблиця 3.3 – Результати експериментів для визначення дальності стабільної передачі відеопотоку

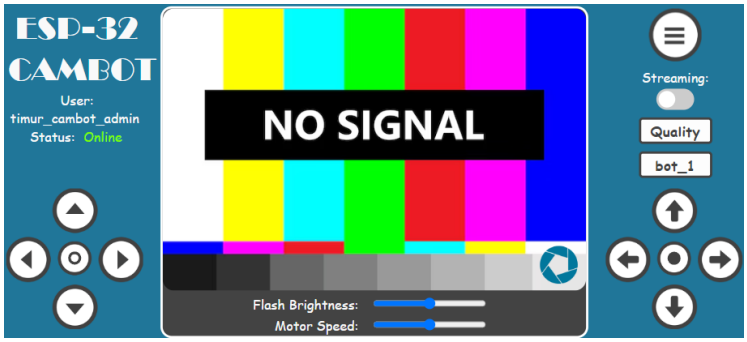
№	Час роботи МР на базі ESP32-CAM, м	Час роботи МР на базі Raspberry Pi, м
1	48	48
2	52	46
3	55	49
4	48	47
5	50	50
6	49	48
7	50	43
8	55	47
9	46	45
10	51	46
Середнє значення, м		
50,4		46,9

Важливо, що розроблена система передає відеопотік та керується через MQTT сервер, що дозволяє керувати такою системою на великих відстанях, за

умови підключення до мережі Інтернет, на відміну від аналогічної системи, що керується тільки через внутрішню мережу, за допомогою HTTP протоколу.

3.2.3 Тестування зручності інтерфейсу керування.

Наступним параметром для порівняння систем є зручність інтерфейсу керування. На рис. 3.1 наведено інтерфейс керування розробленою системою на базі ESP32-CAM, бачимо, що інтерфейс керування має вікно перегляду трансляції, кнопки для пересування МР у чотири сторони, але і має кнопки для керування штативом нахилу та панорамування, керування яскравістю спалаху, керування швидкості двигунів, налаштування якості відеопотоку, кнопку «Зробити фото», перемикач для керування трансляцією, а також кнопку меню, для налаштувань МСВ.



а

б

Рисунок 3.1 – Інтерфейс керування МСВ на базі ESP32-CAM: а – для мобільних пристроїв, б – для десктопних пристроїв.

На рис. 3.2 наведено інтерфейс керування МР на базі Raspberry Pi 3 Model B.

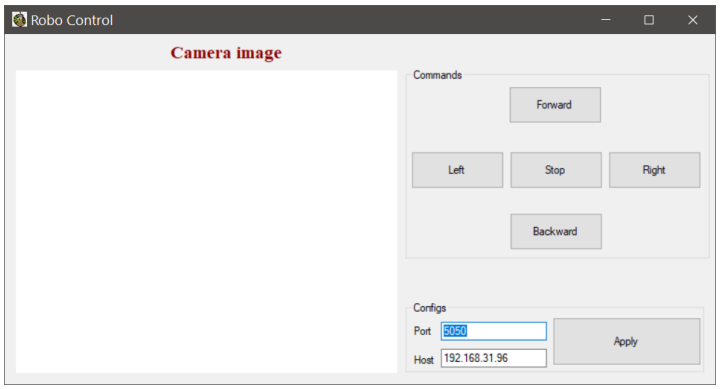


Рисунок 3.2 – Інтерфейс керування МР на базі Raspberry Pi 3 Model B

Даний інтерфейс працює тільки на десктопних пристроях, має вікно для перегляду відеопотоку, кнопки для пересування МР у чотири сторони, а також вікно для налаштування підключення. Такий інтерфейс задовольняє мінімальним потребам, але розроблений інтерфейс для МСВ (рис. 3.1) має більшу кількість налаштувань та може бути відкрито на будь-якому пристрої з браузером, за умови підключення до мережі Інтернет. Також розроблений інтерфейс керування (рис. 3.1) має зручне керування МР за допомогою кнопок на клавіатурі, що є звичним для користувачів персональними комп’ютерами.

3.2.4 Порівняння коштовності систем.

Розрахуємо коштовність систем на базі ESP32-CAM (табл. 3.4) та на базі Raspberry Pi 3 Model B (табл 3.5).

Таблиця 3.4 – Коштовність компонентів системи на базі ESP32-CAM

Назва компоненту	Вартість, грн
ESP32-CAM	341
Драйвер L298N	87

Продовження таблиці 3.4

2 серводвигуни SG90	158
Комплект шасі з платформою	230
Зовнішня антена	115
Штатив нахилу та панорамування	80
3 АКБ Samsung INR18650-35E	630
Модуль заряду трьох акумуляторів типу 18650 з USB Type-C	14
Модуль захисту для акумуляторів HX-3S-JHA10	140
Загальна сума, грн	1795

Таблиця 3.5 – Коштовність компонентів системи на базі Raspberry Pi 3 Model B

Назва компоненту	Вартість, грн
Raspberry Pi 3 Model B	3395
Модуль камери	234
Драйвер L298N	87
Комплект шасі з платформою	230
4 АКБ Samsung INR18650-35E	840
Модуль заряду чотирьох акумуляторів типу 18650 з USB Type-C	130
Модуль захисту для акумуляторів HX-4S-A10	110
Загальна сума, грн	5026

Бачимо, що система на базі Raspberry Pi 3 Model B дорожча на 3231 гривню, найбільша різниця у ціні мікроконтроллерів, вибір мікроконтролера Raspberry Pi 3 Model B є не доцільним для використання у таких задачах через високу вартість, але такий мікроконтролер має більшу кількість вільних портів

та більшу обчислювальну потужність, що відкриває можливості для вдосконалення системи, на відміну від системи на базі ESP32-CAM.

3.3 Результати експериментів

У результаті проведення експериментів з порівняння аналогічних систем на базі ESP32-CAM та Raspberry Pi 3 Model B було визначено, що час автономної роботи систем майже не відрізняється. Дальність стабільної передачі відеопотоку більша у розробленої МСВ у середньому на 3,5 метри за рахунок використання зовнішньої Wi-Fi антени. Зручність та універсальність інтерфейсу керування розробленої МСВ краща, ніж у аналогічної системи. А також, вартість розробленої системи менша на 3231 гривню, ніж вартість аналогічної системи. Згідно з цими результатами, можемо зробити висновок, що розроблена МСВ краща за всіма показниками, але варто зауважити, що покращення та додавання нового функціоналу розробленій системі буде проблематичним через відсутність додаткової обчислювальної потужності, а також відсутність вільних портів, на відміну від аналогічної системи.

3.4 Висновки до розділу 3.

У третьому розділі даної роботи було розроблено план-програму для проведення експериментів для порівняння аналогічних систем. Було проведено експерименти за показниками: час автономної роботи, дальність стабільної передачі відеопотоку, зручність інтерфейсу керування та коштовність системи. Визначено, що розроблена система краща за всіма показниками, але на відміну від аналогу має менше можливостей для вдосконалення.

4 ОХОРОНА ПРАЦІ

4.1 Вимоги до приміщення

Роглянемо питання охорони праці для приміщення, яке має робоче місце з комп'ютером та використовується для розроблення та програмування мобільних роботів.

Такого роду роботи відносяться до категорії «Іа» за величиною загальних енерговитрат організму, тобто до робіт, що виконуються сидячи та не потребують фізичного напруження. Відповідно, категорія робіт «Іа» повинні виконуватись у приміщеннях, що мають наступні значення метрологічних параметрів, для забезпечення комфортних умов праці:

а) у холодну пору року:

1) температура повітря – 22-24°С;
2) відносна вологість – 40-60%;
3) швидкість руху повітря – не більше 0,1 м/с;

б) у теплу пору року:

1) температура повітря – 23-25°С;
2) відносна вологість – 40-60%;
3) швидкість руху повітря – не більше 0,1 м/с.

До роботи з персональним комп’ютером можуть бути допущені тільки працівники, які не мають медичних протипоказань і пройшли інструктаж з охорони праці. Особи, що працюють з комп’ютером понад половину робочого часу, зобов’язані проходити медичні огляди перед початком роботи та періодично згідно із встановленими нормами.

Робоче місце для роботи з комп’ютером має бути організоване відповідно до вимог безпеки та ергономіки. Стіл повинен бути достатньо просторим, щоб на ньому зручно розміщувалися монітор, клавіатура,

необхідні пристрої та документи. Поверхня столу має мати мінімальні відбиваючі властивості.

Клавіатуру необхідно розмістити так, щоб перед нею залишалося місце для зручного розташування рук працівника, принаймні 30 см від краю столу. Монітор має бути розташований так, щоб забезпечити комфортне спостереження працівника за екраном. Екран слід встановлювати нижче рівня очей, перпендикулярно до лінії погляду, що опускається під кутом приблизно 15° від горизонталі.

Для зменшення впливу електромагнітного випромінювання, мінімальна відстань між екраном монітора і працівником повинна складати 50 см, а оптимальна – це 60-70 см.

Крісло чи стілець для роботи має бути стійким і мати можливість регулювання висоти сидіння, нахилу спинки, а також відстані між спинкою та переднім краєм сидіння. Регулювання всіх параметрів має виконуватися окремо, легко і надійно фіксуватися.

Приміщення у якому виконується розробка має наступні характеристики: площа – 22,5 м^2, висота – 2,5 м, одне робоче місце, обладнання – стіл з ПК, монітор та периферійні пристрої. Відповідно до ДНАОП 0.00-1.31-99, одне окреме робоче місце повинно мати площу не менше 6 м^2 та 20 м^3 об’єму, тому можна вважати, що приміщення задовольняє вимогам.

Відповідно до ДБН В.25-28-2006 приміщення з ПК повинно мати як природне так і штучне освітлення, при чому коефіцієнт природного освітлення не повинен бути нижчим за 1,5%. Проведемо розрахунок питомої потужності освітлення приміщення за формулою (4.1):

$$W = \frac{W_\Sigma}{S}, \tag{4.1}$$

де W – питома потужність, Вт/м^2,

S – площа приміщення, м^2,

$W_\sum$ - загальна потужність освітлювальної установки, Вт.

Загальна потужність освітлювальної установки розраховується за формулою (4.2):

$$W_\Sigma = W_{св} \cdot n_{св}, \quad (4.2)$$

де $W_{св}$ – потужність одного освітлювального приладу, Вт,

$n_{св}$ – кількість освітлювальних приладів у приміщенні, шт.

Таким чином виходить, що $W_\sum$ = 150 Вт, тоді $W = 150/22{,}5 = 6{,}66$ Вт/м^2, щоб отримати величину у люксах необхідно поділити це значення на 0,0079, в результаті отримаємо 843 лк, що перевищує мінімально допустимий рівень освітленості 300 лк більше ніж у два рази.

4.2 Висновки до розділу 4.

У даному розділі було розглянуто вимоги охорони праці для приміщення, призначеного для програмування мобільних роботів. Проведені розрахунки підтвердили відповідність приміщення нормам площі, об’єму, параметрів повітря та освітленості. Освітлення перевищує мінімальний рівень у 300 лк, що забезпечує комфортні умови праці. Робоче місце відповідає вимогам безпеки та ергономіки, створюючи сприятливі умови для виконання завдань.

ВИСНОВКИ

У даній кваліфікаційній роботі було розглянуто процес розроблення мобільної роботизованої системи відеонагляду за перебігом виробничих процесів на приладобудівних підприємствах.

Перший розділ роботи було присвячено аналізу сучасного стану та проблем мобільних систем відеонагляду, було визначено основні вимоги та компоненти для створення власної системи. Розглянуто різні типи мобільних роботів, їхні переваги та недоліки, а також можливості їх використання в різних умовах. Для розв’язання задачі створення такої системи необхідно розробити апаратну та програмну частини системи, при чому програмна частина може бути поділена на програму для мобільного робота та клієнтську програму для керування мобільним роботом.

За результатами проведеного огляду сучасних рішень при розробці мобільних роботів визначено основні компоненти робота для системи відеонагляду, таких як плати керування, драйвери двигунів постійного струму та джерело живлення, що забезпечують надійну роботу мобільної роботизованої платформи. Виконано обґрунтований відбір апаратних компонентів серед усіх можливих, враховуючи їх технічні характеристики, вартість та сумісність з іншими елементами системи.

Конструювання мобільного робота передбачає собою розрахунок джерела живлення, розроблення схем підключення апаратних компонентів, а також розрахунок стійкості руху колісного робота, завдяки чому було визначено, що випадок положення псевдо колеса перед двома ведучими колесами триколісного робота є оптимальним, тобто навіть на великих швидкостях мобільний робот буде тримати прямолінійну траєкторію.

Також було розроблено схеми алгоритмів роботи програмного забезпечення мобільного робота для відеонагляду, які охоплюють всі основні функції роботизованої платформи, такі як трансляція відеопотоку в реальному

часі, дистанційне керування пересуванням робота за допомогою бездротової мережі Wi-Fi та використання протоколу MQTT, що широко застосовується у IoT проектах.

Розроблено алгоритми роботи серверної та клієнтської чистин програми для керування системою, що має достатній та зручний функціонал для керування мобільним роботом відеонагляду. Програма керування може бути опублікована на хостингу, що дозволить керувати системою незалежно від місцезнаходження користувача.

Було розроблено план-програму та проведено експерименти для порівняння розробленої системи з аналогічною. За результатами проведених експериментів зроблено висновок, що розроблена система краща за аналог для порівняння за всіма показниками, а саме часом автономної роботи, дільністю стабільної передачі відеопотоку, зручністю інтерфейсу керування, а також коштовністю системи. Проте варто зауважити, що аналогічна система, що розглядалася має більше можливостей до вдосконалення, на відміну від розробленої системи.

Результати роботи можуть бути використані у промислових системах відеонагляду за виробничими процесами.

Подальше удосконалення розробки може бути у напрямку додавання комп’ютреного зору, але це можливо за умови додавання ще одного мікроконтролера.

Робота має відношення до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», а саме пункту 9.4 «Сприяти прискореному розвитку високо- та середньовисокотехнологічних секторів переробної промисловості, які формуються на основі використання ланцюгів «освіта – наука – виробництво» та кластерного підходу за напрямами: розвиток інноваційної екосистеми; розвиток ІКТ; застосування ІКТ в АПК, енергетиці, транспорті та промисловості; високотехнологічне машинобудування; створення нових матеріалів; розвиток фармацевтичної та біоінженерної галузей».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з «Підготовки та захисту кваліфікаційної роботи» здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка освітньо-професійних програм: «Комп'ютерно-інтегровані технологічні процеси і виробництва»; «Комп'ютеризовані та робототехнічні системи» / Невлюдов І.Ш. та ін., Р. В. Харків: ХНУРЕ, 2023, 55 с.

2. ДСТУ 3008: 2015 Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. ДП «УкрНДНЦ», 2016. – 31 с.

3. Contributors to Wikimedia projects. Mobile robot - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Mobile_robot (date of access: 17.09.2024).

4. A Decentralized Cluster Formation Containment Framework for Multirobot Systems / J. Hu et al. IEEE Transactions on Robotics. 2021. P. 1–20. URL: https://doi.org/10.1109/tro.2021.3071615 (date of access: 18.09.2024).

5. Невлюдов І.Ш., Євсєєв В.В., Максимова С.С. - BEAM робототехніка: Навчальний посібник. – Oktan Print – Prague.: 2024.- 276 с. (дата звернення: 18.09.2024).

6. Addverb | Lagerautomatisierung | Globales Robotik-Unternehmen. URL: https://addverb.com/wp-content/smush-webp/2023/07/5.jpg.webp (дата звернення: 19.09.2024).

7. Autonomus delivery robot "Camello". OTSAW | Accelerating Robotics Globally. URL: https://otsaw.com/wp-content/uploads/2023/04/camello-cutout-min.png (date of access: 28.09.2024).

8. International conference on intelligent robots and systems. 2008 IEEE/RSJ international conference on intelligent robots and systems (IROS), Nice, 22–26 September 2008. 2022. P. 2210–2211.

9. Security & Inspections robots for Industry leaders Turnkey robotic solutions RaaS. URL: https://smprobotics.com/wp-

content/uploads/2015/10/security_patrol_robot_s52.png (дата звернення: 27.09.2024).

10. The Crawler mobile robot. RoboTech Vision | Development of autonomous robots with AI. URL: https://robotechvision.com/wp-content/uploads/2021/09/farming-mobile-robot-vineyard-crawler-robotechvision-blog.jpg (date of access: 28.09.2024).

11. ESP32-CAM remote controlled robot assembled. URL: https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2021/01/ESP32-CAM-Remote-Controlled-Robot-Assembled.jpg?w=750&quality=100&strip=all&ssl=1 (date of access: 20.09.2024).

12. ESP32WROVERE & ESP32WROVERIE Datasheet. 2023. Espressif Systems. URL: https://www.espressif.com/sites/default/files/documentation/esp32-wrover-e_esp32-wrover-ie_datasheet_en.pdf (date of access: 25.09.2024).

13. ESP32-CAM development board. 5 p. URL: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf (date of access: 13.11.2023).

14. Getting started with seeed studio XIAO ESP32S3 (sense) | seeed studio wiki. Seeed Studio Wiki. URL: https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/ (date of access: 27.09.2024).

15. Лихо Т. А. Вибір обладнання для розробки мобільного робота для відеонагляду. Збірник студентських наукових статей «Автоматизація та приладобудування» «Automation and Development of Electronic Devices» ADED-2023. 2023. № 2. С. 197–202. URL: https://tapr.nure.ua/wp-content/uploads/2023/11/zbirnik-aded2023_2.pdf (дата звернення: 01.10.2024).

16. Драйвер двигунів двоканальний DRV8833 купити у Києві та Україні. Arduino в Україні. URL: https://arduino.ua/ru/prod3697-draiver-dvigatelei-dvyhkanalnii-drv8833 (дата звернення: 28.09.2024).

17. Драйвер двух двигунів на L298N купити у Києві та Україні. Arduino в Україні. URL: https://arduino.ua/ru/prod406-draiver-dvyh-dvigatelei-na-l298n (дата звернення: 28.09.2024).

18. Невлюдов І.Ш., Андрусевич А.О., Євсєєв В.В. Проектування мобільних роботів на базі одноплатних комп'ютерів (Raspberry Pi и мови Python 3.6): Підручник. – Харків: 2020.-257 с. (дата звернення: 28.09.2024).

19. Драйвер двигунів двоканальний на TB6612FNG купити у Києві та Україні. Arduino в Україні. URL: https://arduino.ua/ru/prod2377-draiver-dvigatelei-dvyhkanalnii-na-tb6612fng (дата звернення: 28.09.2024).

20. Учасники проектів Вікімедіа. Автоматизована система керування – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Автоматизована_система_керування (дата звернення: 27.09.2024).

21. ДСТУ 2941-94 Системи оброблення інформації. Розроблення систем. Терміни та визначення. Інститут програмних систем НАН України, 1994. – 22 с.

22. Що таке IoT (інтернет речей) простими словами?. Atiko. URL: https://www.atiko.com.ua/articles-ua/chto-takoe-iot-prostymi-slovami/ (дата звернення: 01.10.2024).

23. Учасники проектів Вікімедіа. Інтернет речей – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Інтернет_речей (дата звернення: 01.10.2024).

24. Що це таке Zigbee - iven.com.ua. Інтернет-магазин опалення та водопостачання, Київ, Україна - iVen.com.ua. URL: https://iven.com.ua/content/14-ssho-ce-take-zigbee (дата звернення: 02.10.2024).

25. Лихо Т. А. Розроблення Веб-сторінки керування мобільним роботом через протокол MQTT / Т. А. Лихо // Computer-integrated technologies, automation and robotics 2024 : proceedings of the I st All-Ukrainian Conference, Kharkiv, May 16-17, 2024. – Kharkiv, 2024. – P. 66-71. URL:

https://openarchive.nure.ua/server/api/core/bitstreams/95c1b0df-eab1-4f5c-be69-beb263aa5e34/content (date of access: 5.10.2024).

26. What is MQTT? - MQTT Protocol Explained - AWS. Amazon Web Services, Inc. URL: https://aws.amazon.com/what-is/mqtt/?nc1=h_ls (date of access: 5.10.2024).

27. Модуль Wi-Fi ESP32-CAM з камерою 2MP. Arduino Придбати в Києві, Україна. URL: https://arduino.ua/products_pictures/large_aoc673_1.jpg (дата звернення: 5.10.2024).

28. Модуль на базі мікросхеми L298. Arduino Придбати в Києві, Україна. URL: https://arduino.ua/products_pictures/large_L298moduleRed-2.jpg (дата звернення: 6.10.2024).

29. RoboFactory - Making Home Automation Simple and Affordable. URL: https://www.robofactory.co.za/2918-large_default/pan-and-tilt-kit-for-sg90-servo-servos-excluded.jpg (дата звернення: 06.10.2024).

30. 18650 Samsung INR 35E 3500mah 8A 3.7V, оригінал Корея. BATTEREX.com.ua | Акумулятори, зарядні пристрої та аксесуари. URL: https://batterex.com.ua/ru/rechargeable-batteries/18650_batteries/18650-samsung/18650_samsung_35e_3500mah (дата звернення: 09.10.2024).

31. Плата BMS 3s 18650. NAILTRONIC.LAB. URL: https://shop.nailtronic.com/image/cache/catalog/image/catalog/hx-3s-jha10front451-500x400.png (дата звернення: 13.10.2024).

32. Зарядний пристрій для АКБ з Type-C. РКС Компоненти - РАДІОМАГ. URL: https://www.rcscomponents.kiev.ua/img/18650-type-c-3s-2a-m-2.jpg (дата звернення: 13.10.2024).

33. Аналіз стійкості руху мобільних роботів: моделювання та демонстрація у середовищі Матлаб. eVNUIR. URL: https://evnuir.vnu.edu.ua/handle/123456789/15728 (дата звернення: 15.11.2024).

34. Шостак Б.О., Невлюдов І.Ш., Филипенко О.І. Людино-машинний интерфейс в технічних засобах автоматизації. – Харків: «НТМТ», 2019. – 244 с. (дата звернення: 15.11.2024).

35. Python (programming language) - Wikipedia. Wikipedia. URL: https://en.wikipedia.org/wiki/Python_(programming_language) (date of access: 11.11.2024).

36. Django. Django Project. URL: https://www.djangoproject.com/ (date of access: 12.11.2024).

37. JavaScript | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (date of access: 12.11.2024).

38. Розробка веб-додатків на Python | WebCase. Webcase. URL: https://webcase.com.ua/uk/blog/razrabotka-veb-prilozhenij-s-ispolzovaniem-python-i-django/ (дата звернення: 14.11.2024).

39. Роль JavaScript у розробці веб-додатків. REDSTONE. URL: https://redstone.agency/blog/rol-javascript-u-rozrobtsi-veb-dodatkiv/ (дата звернення: 15.11.2024).

40. HTML – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/HTML (дата звернення: 03.12.2024).

41. CSS – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/CSS (дата звернення: 05.12.2024).