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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему:

Програмне забезпечення системи кібербезпеки навігації техніки на базі Arduino

Виконав здобувач вищої освіти
IV курсу, групи КБ20
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____Турик Б.Є.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
Босько В.В
«____» _____ 2024 р.
Рецензент _____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *бакалавр*
Галузь знань. 12 *"Інформаційні технології"*
Спеціальність *125 "Кібербезпека"*
Освітньо-професійна (освітньо-наукова) програма *"Кібербезпека"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
"____" _____2024 року

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

Турику Богдану Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки навігації техніки на базі Arduino*

2. Керівник роботи *Босько Віктор Васильвич, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від 01.04.2024 року № 135-02

3. Строк подання роботи до захисту *20.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи. *Розробка програмного забезпечення системи кібербезпеки навігації техніки на базі Arduino*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію
6. Висновки.

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « » 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем керування	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.05.2024 р.	
8.	Попередній захист роботи	20.05.2024 р.	

Дата видачі завдання
«__»_____2024р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання

«__»_____2024р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Турик Б.Є. Програмне забезпечення системи кібербезпеки навігації техніки на базі Arduino.

125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній кваліфікаційній бакалаврській роботі розроблено програмне забезпечення для автомобіля-робота який може навігувати в реальному часі і уникати перешкод.

Метою роботи. Проєкт має на меті створення управління для мобільного колісного робота з використанням платформи Arduino, включаючи розробку самого робота.

У проєкті було розроблено та реалізовано комплекс апаратного та програмного забезпечення для мобільного автомобіля-робота, який може навігувати в реальному часі і уникати перешкод за допомогою плати Arduino UNO, драйвера Adafruit Motor Shield і програмного забезпечення Arduino IDE, з використанням бібліотек AF Motor та Servo.

Практичним результатом даної роботи система управління для мобільного колісного робота з використанням платформи Arduino.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів.

В повній мірі описані всі компоненти розробленого програмного забезпечення.

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

Отримані результати мають значення для майбутніх досліджень та розробок у галузі систем навігації мобільних роботів, особливо в умовах невизначеного середовища з багатьма перешкодами.

Ключові слова: кібербезпека, навігація, самохідна техніка, мобільний робот, алгоритми уникнення перешкод, датчики, ультразвуковий датчик, Arduino.

ANNOTATION

Turik B.E. Arduino-based equipment navigation cyber security system software.

125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this qualifying bachelor's thesis, software is developed for a robot car that can navigate in real time and avoid obstacles.

The purpose of the work. The project aims to create control for a mobile wheeled robot using the Arduino platform, including the development of the robot itself.

The project developed and implemented a complex of hardware and software for a mobile robot car that can navigate in real time and avoid obstacles using an Arduino UNO board, an Adafruit Motor Shield driver, and Arduino IDE software, using the AF Motor and Servo libraries.

The practical result of this work is a control system for a mobile wheeled robot using the Arduino platform.

In the process of working on the software model, an analysis of existing hardware and software was performed.

All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The obtained results have implications for future research and development in the field of mobile robot navigation systems, especially in uncertain environments with many obstacles.

Keywords: cyber security, navigation, self-propelled equipment, mobile robot, obstacle avoidance algorithms, sensors, ultrasonic sensor, Arduino.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

МР – Мобільний робот

СК – система керування

ПУ – Пристрої управління

MVC - Model-View-Controller

ЛП – локальне позиціонування

КР – колісний робот

ОС - операційна система

XML - Extensible Markup Language

ГКС – глобальна карта середовища

ORM - Object-relational mapping

Sass - Syntactically Awesome Stylesheets

DOM - Document Object Model

EJS - Embedded JavaScript templating

ВСТУП

Актуальність теми. Технології автономного управління наземними транспортними засобами привертають увагу науковців і інженерів вже кілька десятиліть. Автомобільні виробники, виробники компонентів та IT-компанії зосереджені на покращенні механізмів автоматизації управління, стандартизації нових технологічних рішень і їх комерціалізації. Створення робокарів, яке може радикально змінити наше життя, подібно до впровадження паровозів і автомобілів, особливо важливе для осіб з обмеженими можливостями або старшого віку. Вивчення передових процесорів, датчиків, складних алгоритмів і картографування є ключовим для розвитку навігаційних систем і є пріоритетом у багатьох наукових дослідженнях.

В цьому контексті, актуальним є розв'язання задач алгоритмізації навігаційних систем та моделювання руху транспортних засобів. Використання Arduino для прототипування і тестування нових рішень може сприяти прискоренню розробки завдяки глибшому розумінню основ створення робокарів. Ось чому розробка системи навігації для самохідної техніки на базі Arduino є важливою і визначає тему, предмет і мету цього дослідження.

Об'єкт розробки – система навігації для самохідної техніки.

Предмет дослідження включає методи і алгоритми навігації такої техніки, а також апаратне і програмне забезпечення на базі Arduino.

Мета роботи полягає у створенні системи управління мобільного колісного робота з використанням Arduino і розробці такого робота.

Для досягнення визначеної мети необхідно виконати наступні завдання:

- розкласти на складові поняття "самохідна техніка" та "система навігації", та обгрунтувати користь використання платформи Arduino для створення прототипів такої техніки;

- виявити основні проблеми, які виникають при забезпеченні автономності

наземних транспортних засобів;

- визначити необхідне апаратне та програмне обладнання для створення прототипу самохідного транспортного засобу з використанням Arduino;

- розробити програмне забезпечення для реалізації обраного алгоритму, написати відповідні програми за допомогою Arduino IDE та мови програмування C/C++, створити схему з'єднань компонентів та зібрати прототип робота-автомобіля;

- провести тестування моделі, її здатності переміщуватися та уникати перешкод, а також оцінити результати тестування.

Практичне значення результатів роботи полягає в створенні алгоритму уникнення перешкод та прототипу самохідного колісного робота, який може бути застосований для розробки роботів, призначених для роботи у закритих просторах.

Особливий інтерес представляє використання такого робота для перевезення поранених, хворих, інвалідів або літніх осіб у медичних установах.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Традиційно термін "самохідна техніка" асоціюється переважно з автомобілями — самохідними колісними машинами, що приводяться в рух за допомогою вбудованих двигунів і призначені для перевезення людей, вантажів, транспортних засобів, виконання спеціальних робіт чи транспортування спеціального обладнання по безрейкових дорогах. Однак, з розвитком технологій, особливо у сфері радіотехніки, автомобілі стали більш автоматизованими. У 1920-х роках було представлено одну з перших моделей машини з дистанційним управлінням, яка за визначенням автора [1] може вважатися першим безпілотним автомобілем, оскільки керувалася бездротовим способом через радіо. Цей історичний приклад підкреслює значні зміни у визначенні і функціональності самохідних технологій, що продовжують еволюціонувати з часом (рисунок 1.1).

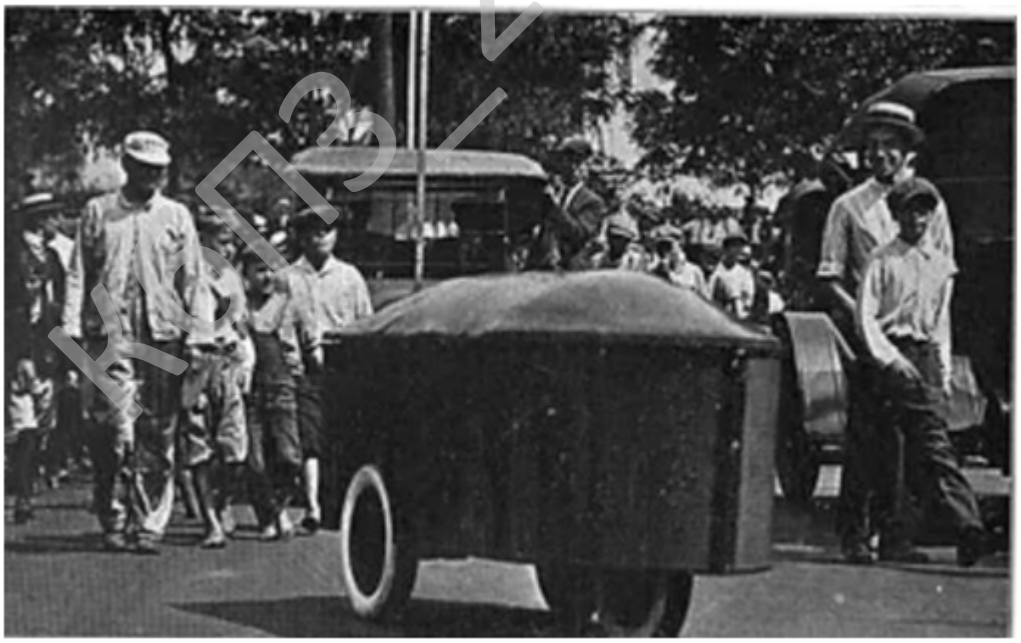

Рисунок 1.1 – Радіокерований автомобіль

На початку минулого століття журнал Scientific American опублікував концепцію самохідного транспортного засобу, який представляв собою "мрію

автомобіліста" — автомобіль, що керується за допомогою набору кнопок.

DARPA, агентство передових оборонних дослідницьких проєктів Міністерства оборони США, відіграло вагому роль у цій області, презентувавши наприкінці 1980-х років інноваційну розробку (див. рисунок 1.3), що отримала назву "автономний наземний транспортний засіб" (Autonomous Land Vehicle – ALV). Цей восьмиколісний робот, висотою 12 футів, обладнаний численними датчиками і може переміщатися з точки А до точки Б без втручання людини. [2].


SCIENTIFIC AMERICAN

THE WEEKLY JOURNAL OF PRACTICAL INFORMATION

NEW YORK, JANUARY 5, 1918

Рисунок 1.2 – Мрія та реальність: самохідна техніка

Рисунок 1.3 – ALV, розроблений DARPA

Пізніше в науковій спільноті з'явився ще один термін - "безпілотний наземний транспортний засіб" (unmanned ground vehicle - UGV), який працює при контакті із землею та без присутності людини на борту [3].

У сучасних документах Європейського Союзу використовуються наступні терміни [4]:

Автоматизований транспортний засіб (Automated vehicle): автотранспортний засіб (автомобіль, вантажівка або автобус), оснащений технологіями (з метою допомоги водієві), здатними передавати виконання елементів завдання водіння до комп'ютерної системи;

Автономний транспортний засіб (Autonomous vehicle): повністю автоматизований транспортний засіб, оснащений технологіями, здатними виконувати всі функції керування без втручання людини.

Національна Адміністрація Безпеки Дорожнього Руху США визначає автомобіль як автономний, якщо він працює, не вимагаючи від водія безпосередньої участі у керуванні, включаючи "кермування, прискорення та гальмування" [5].

Згідно з автором роботи [6, с. 150], автономний транспортний засіб - це неофіційний термін, який використовується для опису автомобіля, обладнаного

автоматизованою системою керування (automated driving system - ADS), який працює в автономному режимі, тобто автомобіль, що має високий рівень автоматизації керування.

Фахівці SAE International (раніше відомі як Товариство автомобільних інженерів – SAE) вказують, що деякі люди традиційно асоціюють автономію водіння виключно з повною автоматизацією рівня 5; тоді як інші застосовують це поняття до всіх рівнів автоматизації водіння; при цьому в законодавстві окремих штатів визначено, що автономність приблизно відповідає будь-якій ADS щонайменше рівня 3 [7].

Українські вчені Поліщук М.М. і Ткач М.М. вказують, що "роботизовані транспортні засоби, що перебувають в експлуатації, можна розділити на дві основні групи: транспортні роботи із твердим шляхопроводом і робокари - транспортні візки з безконтактним індуктивним або оптичним шляхопроводом.

Системи керування таких роботів можуть бути цикловими, позиційними з автоматичним адресуванням. Усі системи, як правило, побудовані на базі локального автомата з керуванням від центрального контролера ЕОМ нижнього й верхнього рівня. Вибір системи керування, як правило, визначається функціональним призначенням транспортного робота" [8, с.31]. Питання роботизованих транспортних засобів також розглянуто в роботах [9-11]. Оскільки компоненти таких автомобілів є як фізичними, так і віртуальними, вчені називають такі машини "кіберфізичні системи" (cyber-physical systems).

Згідно з опитуванням експертів SAE, яке було проведене серед 257 респондентів, щодо загальної назви зазначених вище машин, 44,5% вибрали "self-driving car" (самокерований автомобіль), оскільки саме цей термін використовувався Waymo LLC (раніше відому як Google Self-Driving Car Project) з того часу, як компанія представила свій проєкт кілька років тому. [7].

Зважаючи на ці висновки, у своїй роботі автор використовує термін "self-driving car" (SDC), припускаючи, що "самокерована" є більш точною характеристикою самохідної техніки, здатної рухатися без участі людини. Як

синонім використовується термін "мобільний робот" чи "робот-автомобіль".

Призначення системи

Архітектурна модель

Архітектурну модель самокерованих автомобілів (SDC) можна розглядати з різних площин, таких як фізичні компоненти, етапи розробки, логічні функції та технологічні блоки. тощо (рис.1.4).

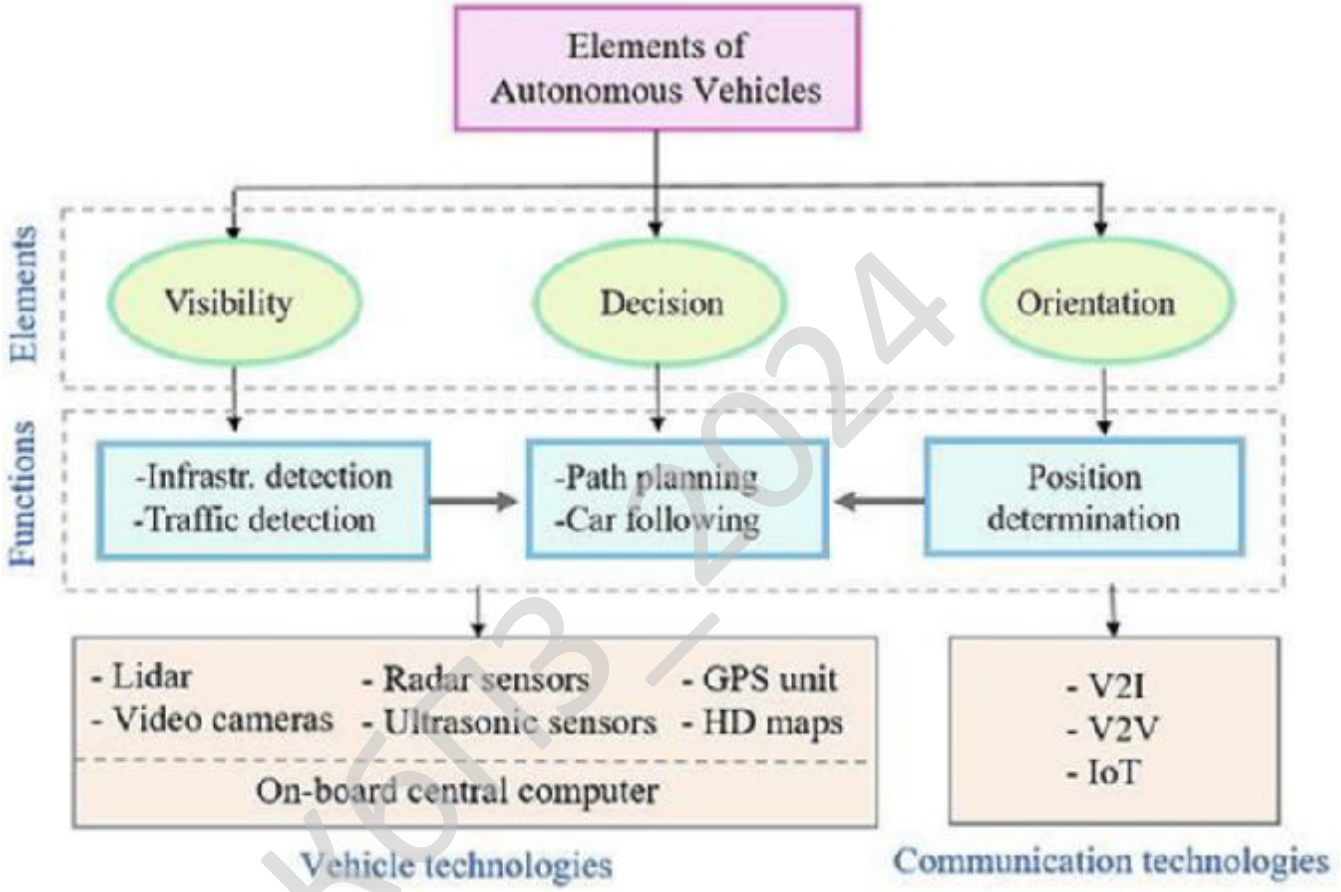

Рисунок 1.4 - Елементи SDC, їх функції та базові технології [12]

Для досягнення мети дослідження у цій роботі архітектуру самокерованих автомобілів (SDC) розглянуто з технічного та функціонального поглядів.

1.1 Область застосування

Апаратне та програмне забезпечення є двома основними аспектами

технічного вигляду архітектури SDC, і кожен шар включає компоненти, які представляють різні аспекти всієї системи [11-12] (рис. 1.5).

Для обробки інформації в самокерованих автомобілях (SDC) використовують обчислювальні платформи з декількома ядрами та графічні процесори (GPU).

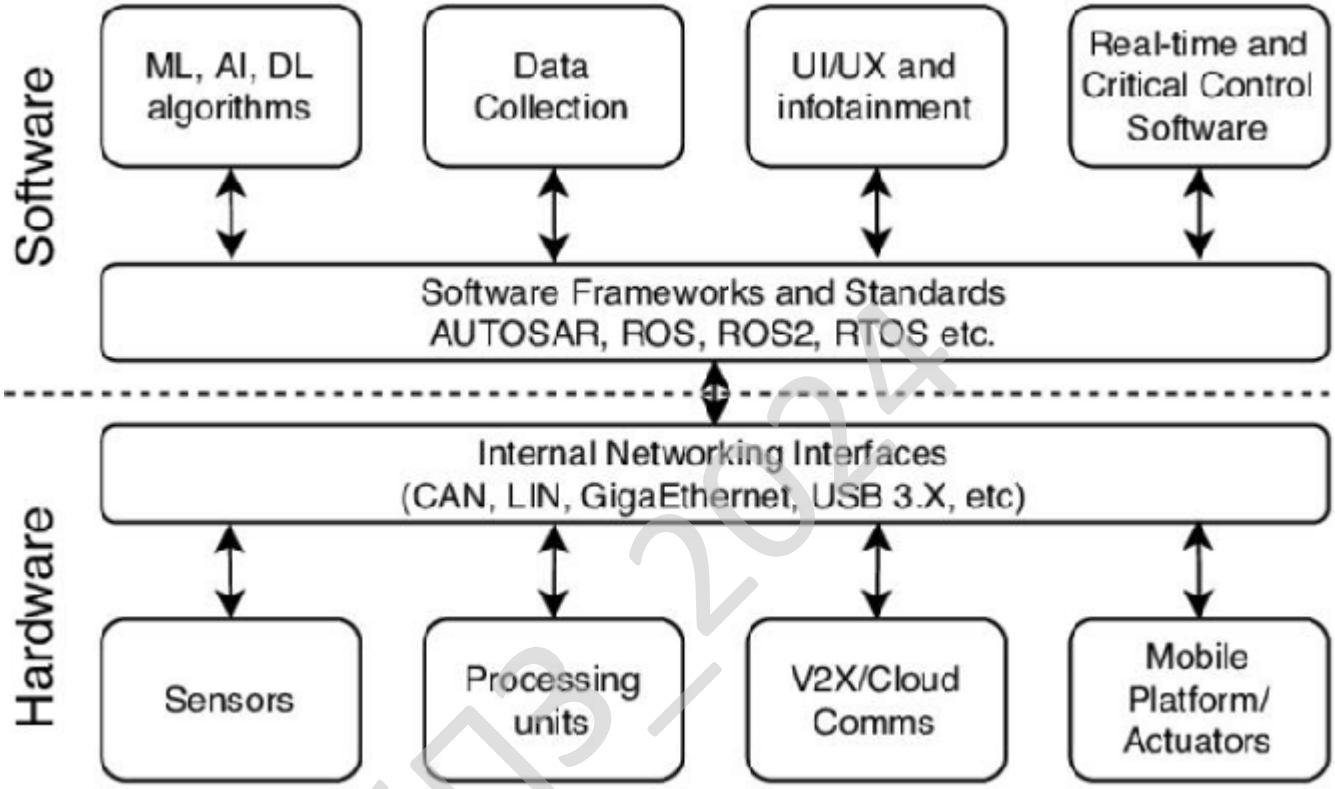

Рисунок 1.5 – Технічна архітектура системи SDC [11]

Разом із даними, що генеруються безпосередньо самокерованими автомобілями (SDC), обробляються також зовнішні дані, доступні з Інтернету, інших транспортних засобів або інфраструктури, що відомо як V2X (Vehicle-to-everything). Апаратна частина також включає сам транспортний засіб, це мобільна платформа та певні пристрої, які можуть бути різних видів залежно від застосування та місцевості, де буде працювати система. Можливості обробки поточних даних SDC такі, що їх іноді називають суперкомп'ютерами на колесах.

Це твердження не далеке від реальності, оскільки через складність в апаратній частині, програмна – також розвивається. Як приклад – AUTOSAR

(AUTomotive Open System Architecture) – це сучасна структура та стандарт для екосистеми автомобільної електроніки. Вона створена і управляється альянсом таких Гіганти автомобільної галузі, такі як BMW, Toyota, GM, Chrysler, Continental, Bosch, Daimler, Volkswagen та багато інших, активно працюють над AUTOSAR. AUTOSAR заснований на мові програмування C [13].

В основі реалізації функцій лежать чотири основні блоки (які зустрічаються у більшості запропонованих архітектур і рішень): сприйняття, планування і прийняття рішень, управління рухом і транспортним засобом, системний нагляд. Ці блоки представлені на рис. 1.6.

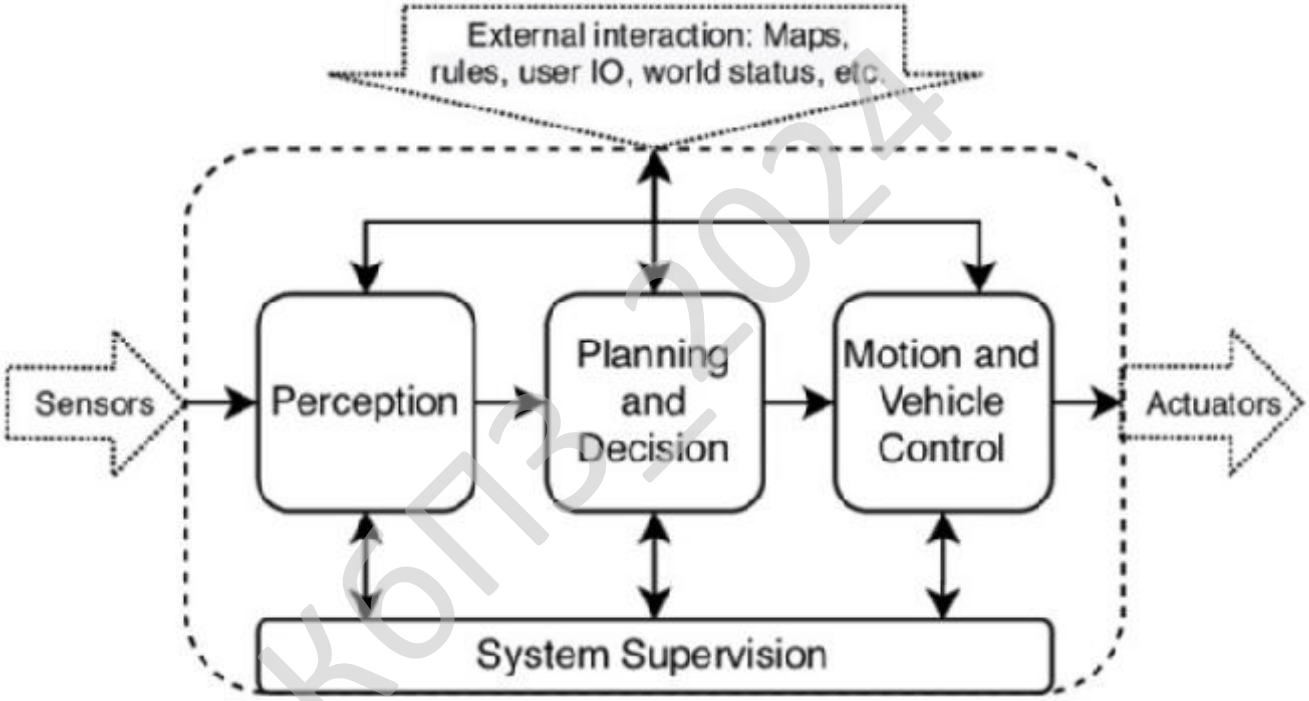

Рисунок 1.6 – Функціональна архітектура системи автономного водіння

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Розробники зосереджуються на конкретному наборі технологій для автономної навігації, зокрема, пов'язаних із уникненням перешкод. Це завдання зазвичай вирішується за допомогою різних типів датчиків і алгоритмів управління, що виконуються на мікроконтролерах або мікрокомп'ютерах.

Основою для переходу від автоматизації до автономності транспортного засобу є впровадження системи ADAS, яка є групою електронних технологій, що допомагають водіям керувати автомобілем і паркуватися.

Використовуючи автоматизовані технології та спираючись на датчики та камери (див. рисунок. 1.7), система виявляє перешкоди та реагує відповідним чином [14].

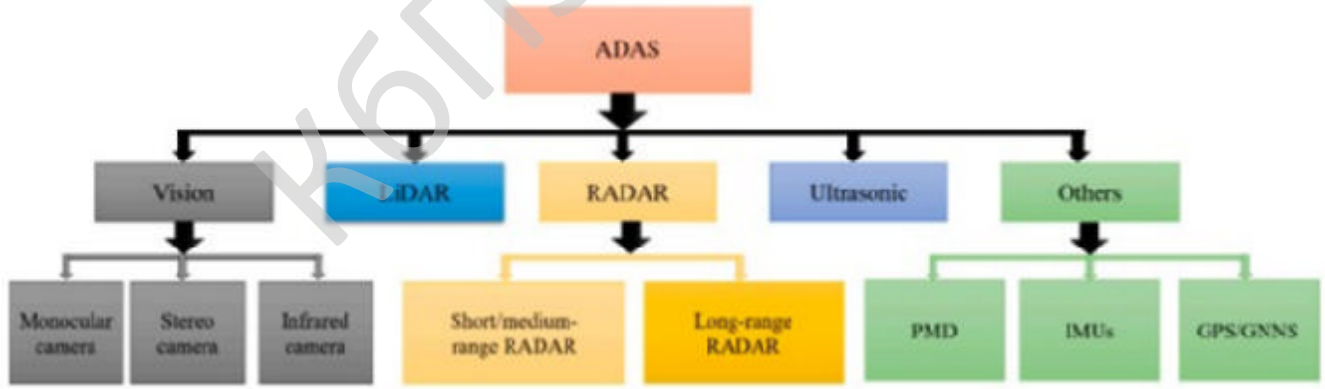

Рисунок 2.1 – Таксономія базових компонентів ADAS [14]

ADAS має різні типи систем, серед яких: адаптивний круїз-контроль (ACC), адаптивне переднє світло (AFL), автоматичне аварійне гальмування (AEB), виявлення сліпих зон (BSD), попередження про перехресний рух (CTA),

система моніторингу водія (DMS), попередження про лобове зіткнення (FCW), інтелектуальна система допомоги при паркуванні (IPA), попередження про вихід зі смуги руху (LDW), система нічного бачення, система виявлення пішоходів (PDS), розпізнавання дорожніх знаків (RSR), система контролю тиску в шинах (TPMS), асистент руху у «пробці» (TJA).

Для надійної та безпечної навігації самокерованих автомобілів (SDC) зазвичай використовують комбінацію різних датчиків, які мають різну фізичну природу та базуються на різних технологіях. Ці датчики використовуються для сприйняття безпосереднього оточення автомобіля. Їх розрізняють як пасивні, які виявляють наявну енергію (таку як світло або випромінювання, що відбивається від об'єктів у навколишньому середовищі), та активні, які посилають власний акустичний чи електромагнітний сигнал і сприймають його відбиття.

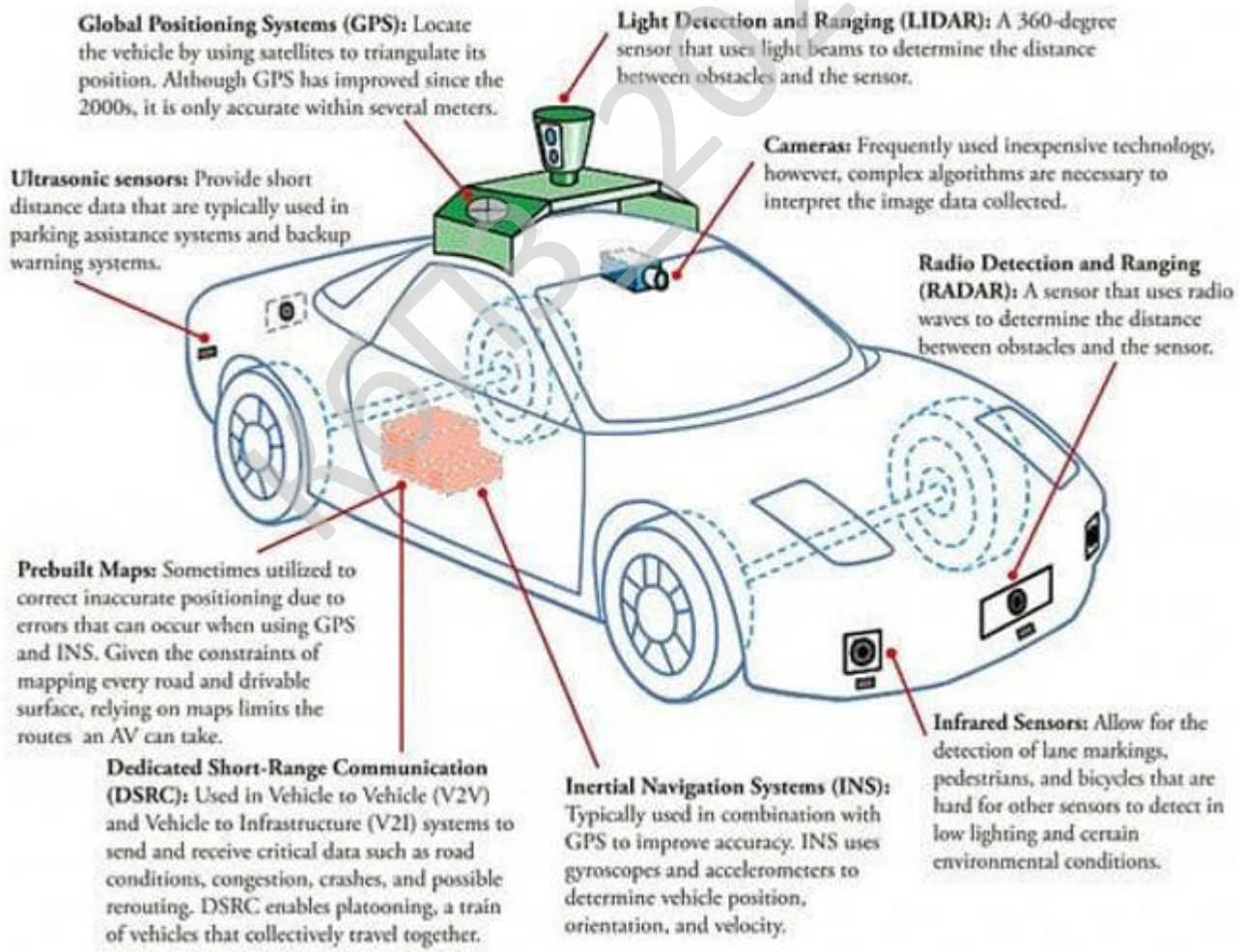

Рисунок 2.2 – Екосистема датчиків SDC

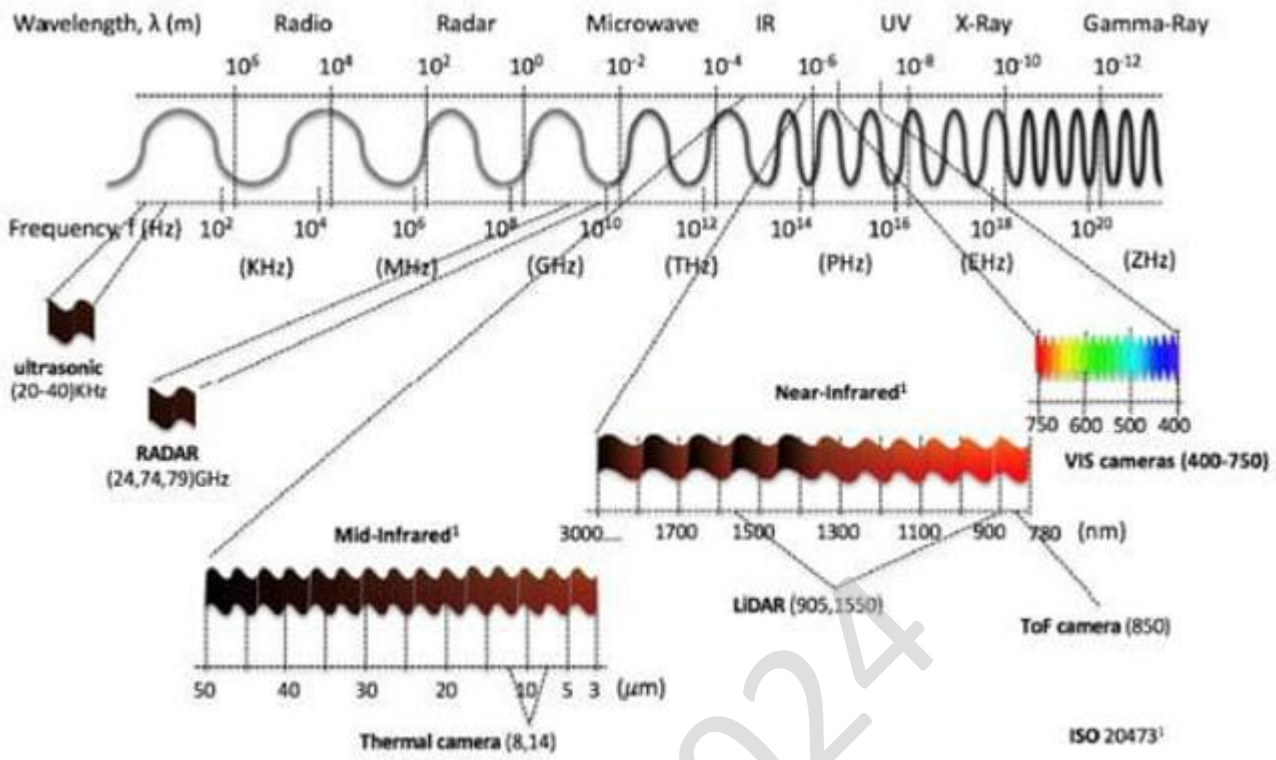

Рисунок 2.3 – Звукові та електромагнітні спектри, які використовуються датчиками SDC [15]

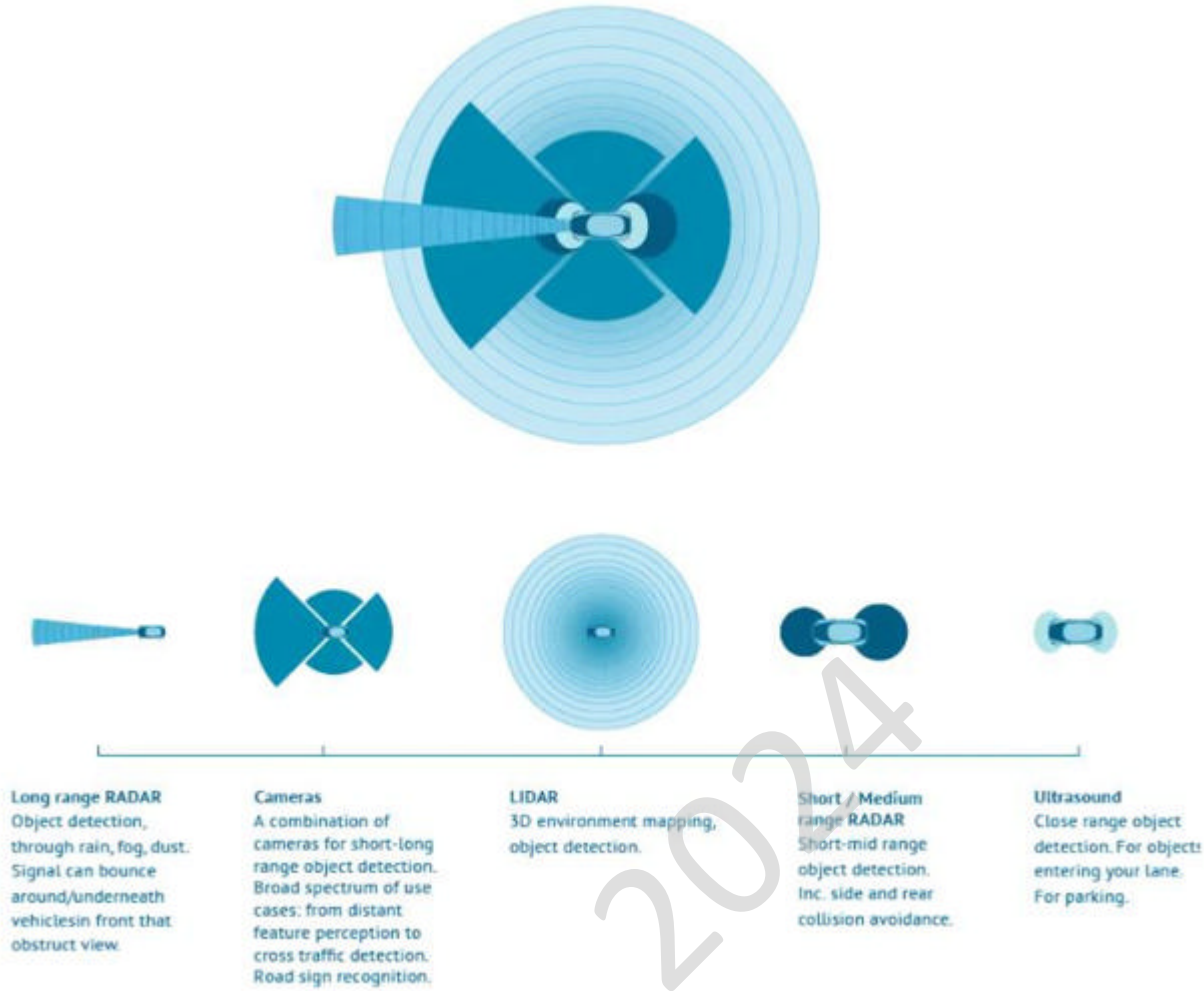

Рисунок 2.4 – Діапазон покриття датчиків [16].

Цифрові камери використовують сенсори зображення пристрою із зарядовим зв’язком (charge-coupled device – CCD) або комплементарний металоксид-напівпровідник (complementary metal-oxide semiconductor - CMOS) [18].

Камери використовуються для розпізнавання дорожніх знаків, читання ліній та іншої розмітки на дорозі, виявлення пішоходів, перешкод та багато іншого. Наприклад, BMW X5 використовує камеру TriFocal із MobilEye EyeQ4 (камера ZF S-Cam 3 має два основних компоненти – плату з процесором MobilEye EyeQ3 і плату з датчиком зображення CMOS від ON Semiconductor), що вважається прогресом у безпілотній навігації [19].

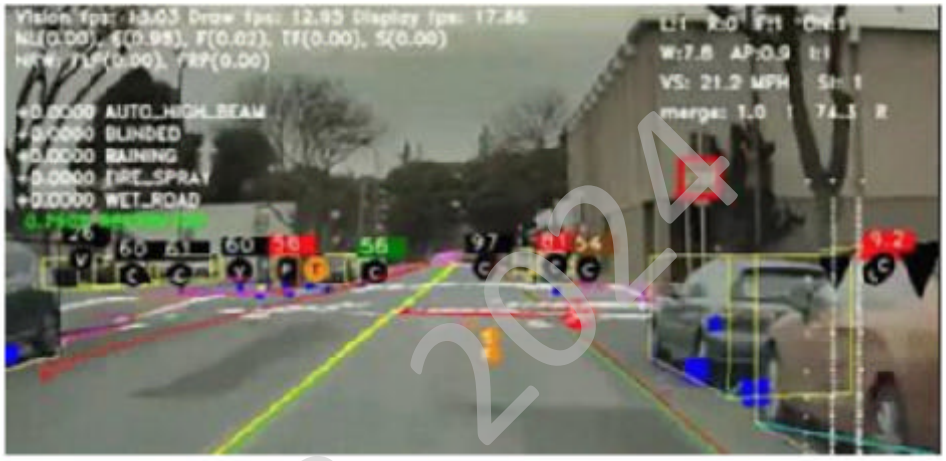

Рисунок 2.5 – Дані сприйняття камери, накладені на відеозображення

Основні переваги пасивних датчиків включають: висока роздільна здатність у пікселях і колір по всій ширині поля зору; постійна "частота кадрів" у полі зору; дві камери можуть створювати 3D-стереоскопічне зображення; відсутність джерела передачі знижує ймовірність перешкод від іншого автомобіля; низька вартість завдяки вдосконаленій технології; зображення, створені цими системами, легко розуміються користувачами і з ними легко взаємодіяти.

Основні недоліки пасивних датчиків включають:

- низька ефективність при слабкому освітленні або поганих погодних умовах, через те, що вони не мають власного джерела передачі і не можуть легко адаптуватися до цих умов;

- генерування великої кількості даних (від 0,5 до 3,5 Гбіт/с), що може бути проблематичним для внутрішньої обробки або передачі даних у хмару, оскільки це більше, ніж обсяг даних, який генерують активні датчики.

Активні датчики

Датчики RADAR (Radio Detection and Ranging) використовуються для виявлення великих об'єктів перед автомобілем та мають ряд переваг. Зазвичай вони працюють на частоті 76,5 ГГц і використовують два основних методи: пряме поширення та непряме поширення. Обидва методи використовують випромінювання радіочастот та вимірювання часу розповсюдження відбитих сигналів. Це дозволяє датчикам вимірювати розмір та відстань до об'єкта, а також його відносну швидкість.

Оскільки довжина хвиль RADAR становить лише кілька міліметрів, вони можуть виявляти об'єкти розміром у кілька сантиметрів. Крім того, ці датчики можуть "бачити" крізь погану погоду та інші перешкоди видимості.

Датчики RADAR є важливими для: попередження та пом'якшення наслідків зіткнень; виявлення сліпих зон; надання допомоги при зміні смуги руху та при паркуванні тощо [21].

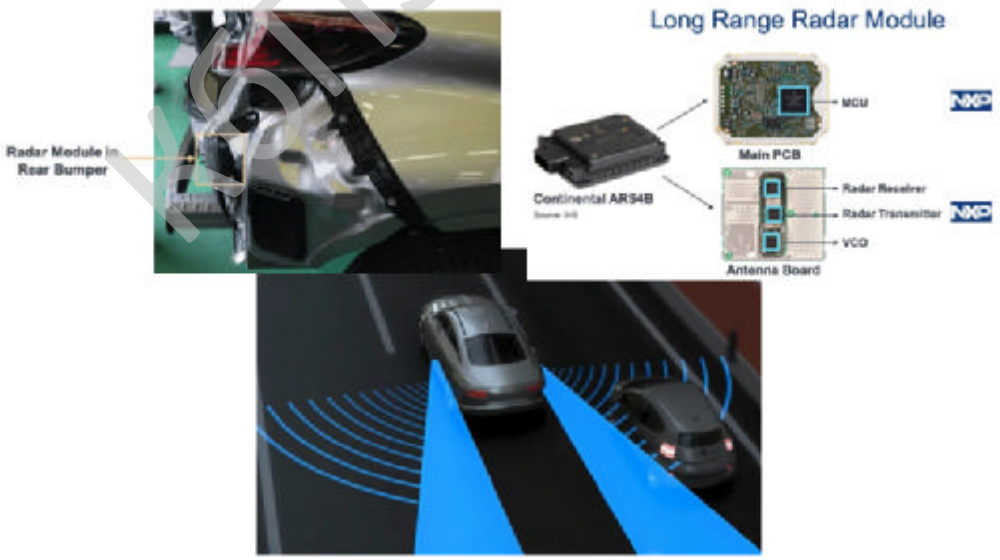

Рисунок 2.6 – Транспортний засіб, обладнаний радаром для виявлення сліпих зон

Ультразвуковий датчик - це пристрій, який використовує принцип ехолокації для визначення відстані до об'єктів. Він випромінює ультразвукові хвилі, які відбиваються від навколишніх об'єктів, а потім приймає їхні відбиті сигнали для визначення відстані до цих об'єктів.

На рисунку 2.7 зображена типова ультразвукова ехолокаційна схема датчика системи безпілотного керування. [23-25]

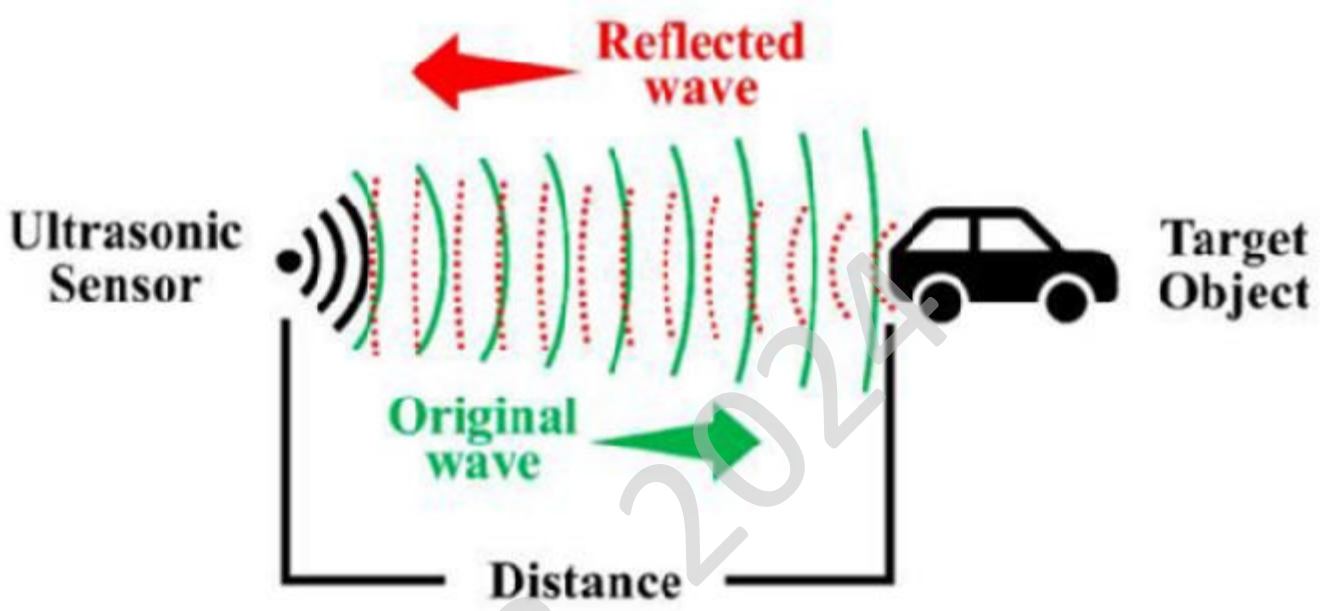

Рисунок 2.7 – Принцип дії ультразвукового датчика

Ці ультразвукові датчики використовуються для навігації та визначення відстані. Вони генерують високочастотний звук приблизно 48 кГц, що більше, ніж удвічі перевищує звичайний діапазон людського слуху, але мають обмежений діапазон порівняно з радаром. Проте, коли об'єкт (перешкода) знаходиться на близькій відстані, ультразвук стає менш витратним варіантом, ніж радар. Хоча ультразвукові датчики мають свої переваги, такі як відносно низька вартість та ефективність на близьких відстанях, вони також мають свої обмеження. Надмірний сторонній шум та температурні зміни можуть погіршити їх характеристики. Ультразвукові датчики мають сліпі зони в непрямому полі їх дії, що може призвести до неточних вимірів. Матеріали з високим коефіцієнтом поглинання звукових хвиль також можуть впливати на їхню ефективність.

Незважаючи на ці обмеження, багато автомобільних компаній використовують ультразвукові датчики у своїх інтелектуальних системах паркування, оскільки вони є досить надійним та вартісним рішенням для цієї конкретної задачі. (рис. 2.8).

LIDAR (Light Detection and Ranging) - це датчик, який використовує імпульсні лазери для створення хмари точок, що дозволяє отримати тривимірне зображення карти. Датчики LIDAR відправляють від 50 000 до 200 000 імпульсів на секунду. Шляхом порівняння різниці у хмарах точок, які отримані послідовно, можна виявити об'єкти та визначити їхній рух, що дозволяє створити тривимірну карту з діапазоном до 250 метрів.

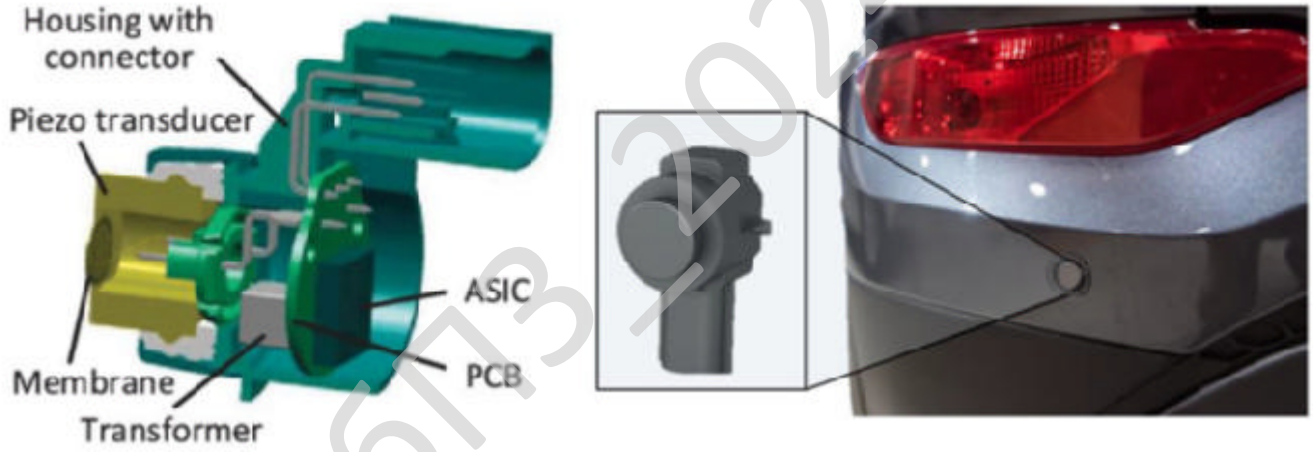

Рисунок 2.8 – Ультразвукові датчики розташовані у бамперах SDC [26]

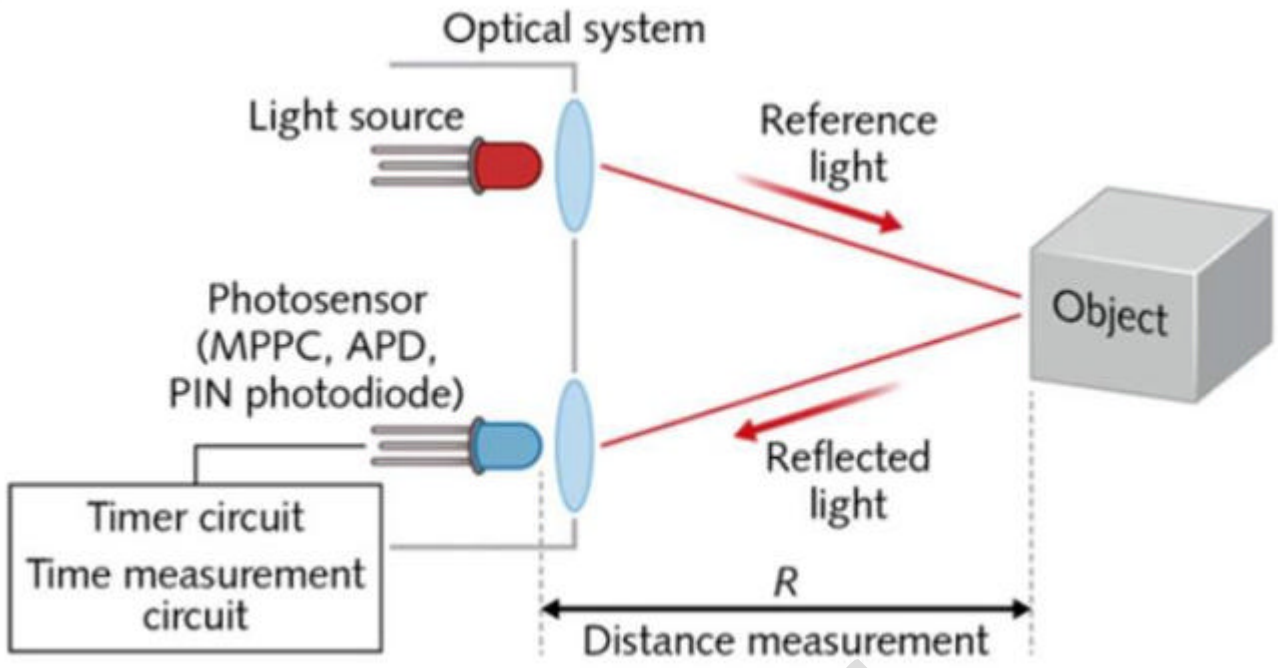

Рисунок 2.9 - Блок-схема LiDAR [15]

LIDAR функціонує на кшталт сонарних систем, використовуючи лазерні промені замість звукових хвиль. Бортовий комп'ютер зберігає інформацію про кожну відбиту точку лазера, що перетворюється на анімоване 3D-зображення навколишнього середовища. LIDAR має переваги у широкому полі зору з потенційним охопленням 360 градусів і великому радіусі дії, а також у більш точних вимірах відстані порівняно з пасивними (оптичними) датчиками. Однак датчики LIDAR зазвичай дорожчі через їхню механічну складність.

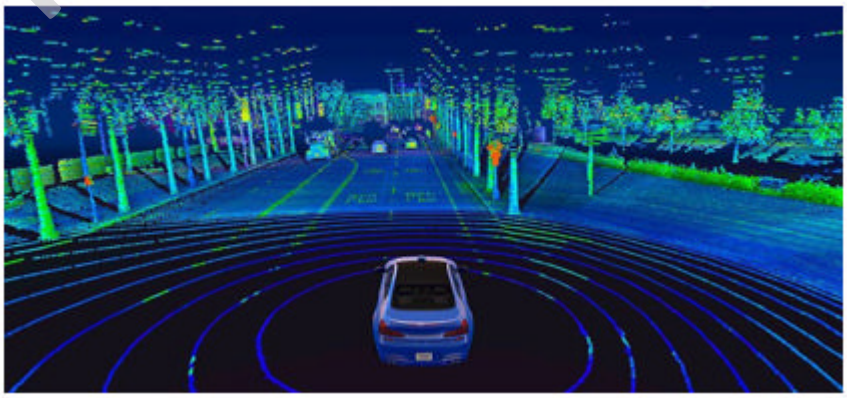

Рисунок 2.10 - Хмара точок, створена датчиком Alpha Prime компанії Velodyne Lidar [29]

Таблиця 2.1 – Порівняльна характеристика датчиків SDC

Функція	LiDAR	RADAR	Camera	Ultrasonic
Первинна технологія	лазерний луч	радіохвиля	світло	звукова хвиля
Діапазон (м)	до 250	0.2-300	до 250	0.02-10
Роздільна здатність	добре	середньо	дуже добре	слабко
Уразливість від погодних умов	так	так	так	так
Впливає на умови освітлення	ні	ні	так	ні
Визначає швидкість	добре	дуже добре	слабко	слабко
Визначає відстань	добре	дуже добре	слабко	добре
Сприйнятливість до перешкод	добре	слабко	дуж. добре	добре
Розмір	великий	малий	малий	малий

Проблеми забезпечення автономності транспортного засобу

Стандарт SAE J3016 дійсно визначає рівні автоматизації водіння, від SAE 0 до SAE 5. Ці рівні визначаються в залежності від того, яку частину керування транспортним засобом відповідає людина, а яку - автоматизована система. Нова ітерація цього стандарту, представлена в 2021 році, може включати оновлення або додаткові розробки, які відображають нові технологічні досягнення та вирішують недоліки попередніх версій.У 2021 р. було подано наступну ітерацію документу SAE J3016 (рисунок 2.11).

Рисунок 2.11 – Рівні автоматизації водіння за SAE J3016 [30]

Рівень 0. На першому рівні відсутня автоматизація водіння, і водій повністю контролює керування транспортним засобом.

Рівень 1. На рівні допомоги водієві, найнижчому рівні автоматизації, маємо єдину автоматизовану систему – адаптивний круїз-контроль.

Рівень 2. На рівні часткової автоматизації водіння ми спостерігаємо впровадження елементів ADAS (Advanced Driver Assistance Systems). Прикладами рівня 2 є такі інновації, як система Bluecruise від Ford, Full Self-Driving від Tesla та Super Cruise від General Motors. (GM) [31].

Рівень 3. У рівні умовної автоматизації водіння, починаючи з третього рівня і вище, автомобілі можуть самостійно відстежувати дорожню ситуацію за допомогою лідару. Транспортні засоби на рівні 3 можуть "читати" (аналізувати) навколишнє середовище, щоб отримати необхідну інформацію для прийняття обґрунтованих рішень щодо подальших дій, зокрема, уникаючи перешкод. Прикладом такого рівня є Audi A8L 2019 року, який став першим серійним

автомобілем на світі рівня 3 [32].

Рівень 4. На рівні високої автоматизації водіння, який становить рівень 4, людське втручання у керування автомобілем не потрібне, оскільки він запрограмований на самостійну зупинку в разі відмови системи. У конструкції автомобіля рівня 4 може відсутнє кермо та педалі. Технологія автоматизації водіння на рівні 4 призначена для застосування в безпілотних таксі та громадському транспорті. У 2021 році Hyundai Motor Group запустила пілотний сервіс автономного водіння на рівні 4 на моделі Hyundai Ioniq 5. [33].

Рівень 5. На рівні повної автоматизації водіння, що є останнім етапом у розвитку автономних транспортних засобів, відсутнє будь-яке втручання людини у динамічні завдання водіння. Транспортні засоби рівня 5 не мають керма, педалей прискорення або гальма. Ці робомобілі працюють усіма погодними умовами, необмежені географічними межами і не потребують втручання людини. Наприклад, декілька років тому компанія Apple запустила проєкт Titan щодо створення повністю автономних транспортних засобів рівня 5. Однак, як стало відомо наприкінці 2022 року, компанія скоротила свої амбітні плани та відклала прогнозовану дату запуску щонайменше до 2026 року. Реалізація проекту зіткнулася з викликами сучасних технологій, оскільки досягнення повної автономії водіння виявилося надзвичайно складним завданням. [34].

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Створення системи кібербезпеки для розробок на Arduino може бути важливим завданням, особливо якщо планується використовувати його у важливих або в системах які піддаються атакам.

Основними метдами захисту можуть бути такі:

- Використовуйте SSL / TLS для зв'язку: Якщо ваша Arduino взаємодіє з іншими пристроями чи серверами через мережу, використовуйте

захищені протоколи, такі як SSL або TLS, для шифрування зв'язку.

- Захист від фізичного доступу: Захистіть вашу Arduino від фізичного доступу, особливо якщо вона використовується в уразливому середовищі.

- Використовуйте паролі та автентифікацію: Реалізуйте механізми автентифікації для доступу до вашої Arduino, наприклад, паролі або ключі доступу.

- Оновлення програмного забезпечення: Регулярно оновлюйте програмне забезпечення вашої Arduino та додатків, щоб усунути виявлені вразливості.

- Використовуйте бібліотеки безпеки: Використовуйте перевірені бібліотеки безпеки для роботи з мережею, шифруванням даних та іншими критичними аспектами безпеки.

- Обмеження прав доступу: Налаштуйте права доступу на вашій Arduino таким чином, щоб обмежити можливість виконання небезпечного коду.

- Моніторинг та журналювання: Додайте механізми моніторингу та журналювання, щоб відстежувати незвичайну активність та інциденти безпеки.

- Використовуйте захист від переповнення буфера: Уникайте переповнення буфера, використовуючи безпечні методи обробки даних та перевірки вводу.

- Шифрування збережених даних: Якщо ви зберігаєте конфіденційні дані на вашій Arduino, використовуйте шифрування для захисту цих даних від несанкціонованого доступу.

- Використовуйте захист від відомих атак: Дослідіть типові атаки на IoT-пристрої, такі як атаки з використанням переповнення буфера, атаки на основі зміни параметрів запитів тощо, і вживайте заходи для захисту від них.

Вибір мови програмування

Мова програмування, що використовується для програмування Arduino, базується на мові C/C++. Однак, для спрощення розробки та взаємодії з

мікроконтролером, Arduino надає спеціальне середовище розробки (IDE), яке включає в себе бібліотеки та функції, спрощуючи роботу з платформою.

Основні особливості мови програмування Arduino:

- функції setup() та loop(): Кожна Arduino-програма має функції setup() і loop(). Функція setup() викликається один раз при запуску програми, а функція loop() викликається безперервно після завершення функції setup(). Це дозволяє створювати код, який виконується циклічно;
- бібліотеки Arduino: Arduino надає багато стандартних бібліотек для спрощення взаємодії з платформою, таких як бібліотека для роботи з GPIO (DigitalInput, DigitalOutput), роботи з аналоговими сигналами, керування серійним портом і багато інших;
- структури даних та функції мови C/C++: Arduino використовує стандартні структури даних та функції мови C/C++, такі як if, for, while, switch, тощо;
- бібліотеки сторонніх розробників: Крім стандартних бібліотек, можна використовувати бібліотеки, розроблені спільнотою Arduino або сторонніми розробниками, для розширення функціональності вашого проекту;
- набір функцій для взаємодії з мікроконтролером: Arduino надає набір функцій для роботи з цифровими та аналоговими пінами, таймерами, інтерфейсами зв'язку (наприклад, UART, I2C, SPI) та іншими функціями, які дозволяють легко керувати мікроконтролером.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке реалізує апаратно-програмний комплекс з впровадженням системи кіберзахисту.

В процесі розробки роботи необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і

негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран повідомлень про некоректні дії користувача та нестандартні ситуації;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

Прототипування на базі Arduino

Arduino - це відкрита електронна платформа, яка базується на простому у використанні апаратному та програмному забезпеченні. Вона призначена для створення різноманітних електронних проектів, включаючи робототехніку. Завдяки мові програмування Arduino, яка базується на мові Wiring, та програмному забезпеченню Arduino, користувачі можуть легко зчитувати вхідні дані (наприклад, з сенсорів) і керувати вихідними пристроями (наприклад, двигунами або світлодіодами).

Процес прототипування з використанням Arduino може бути розділений на кілька етапів, незалежно від типу проекту:

- Ідея (визначення) проблеми: Цей етап передбачає народження ідеї для проекту та розуміння проблем, які потрібно вирішити.
- Концептуалізація: Після виникнення ідеї будується концепція проекту, яка відповідає на питання про те, як саме проект буде працювати та які алгоритми будуть використані.
- Прототипування: На цьому етапі ідея перетворюється на конкретний прототип - апаратно-програмний зразок готового продукту. Спочатку прототип представляє собою альфа-версію проекту, яка потім може перейти в бета-версію для тестування та подальшого вдосконалення.

У робототехніці Arduino широко використовується для керування різними пристроями, такими як двигуни, датчики та виконавчі механізми. Її простота використання, доступність та широкі можливості роблять Arduino популярним вибором для прототипування різноманітних електронних проектів, включаючи робототехніку:

- Економічність та доступність платформи. Платформи Arduino відзначаються низькою вартістю, що робить їх вигідним вибором для

інноваторів та ентузіастів, які самостійно складають свої проекти типу "Зроби сам". Особливо порівняно з іншими мікроконтролерами, такими як Raspberry Pi або Nanode, платформи Arduino є досить економічними. Навіть найдешевший модуль Arduino можна придбати за менше ніж 50 доларів, що робить їх доступними для широкого кола користувачів.

- Підтримка кросплатформенності. Arduino IDE може працювати на різних операційних системах, таких як MacOS, Windows та Linux. Це важлива перевага в порівнянні з іншими платформами мікроконтролерів, багато з яких підтримують лише операційну систему Windows. Доступність Arduino IDE на різних платформах робить його більш універсальним та зручним для розробників, незалежно від їхньої основної операційної системи.Енергоощадність. Arduino вимагає небагато енергії, адже працює з найнижчою напругою, заощаджуючи витрати користувача.

- Швидке прототипування. Платформа Arduino дозволяє швидко створювати прототипи систем завдяки простоті використання та широкому спектру доступних компонентів. Порівняно з альтернативними платформами, процес прототипування на Arduino зазвичай займає значно менше часу. Це сприяє швидкому та ефективному тестуванню ідей та концепцій, що робить Arduino популярним вибором серед розробників для реалізації швидких інженерних прототипів.

- Просте та зрозуміле середовище програмування. Arduino IDE відзначається своєю легкістю використання та швидким освоєнням, що робить його привабливим для новачків. Незважаючи на це, воно також має просунуті функції, які задовольнять потреби професіоналів у розробці. Спрощений інтерфейс Arduino IDE базується на середовищі програмування Processing, що полегшує роботу тим, хто вже має досвід з цією мовою. Наприклад, це зручно для викладачів, які використовують Arduino IDE для навчання студентів, які вже мали досвід з програмуванням в Processing.

- Відкритий вихідний код і розширюване програмне забезпечення.

Arduino IDE та інші компоненти екосистеми Arduino мають відкритий вихідний код, що робить їх доступними для розширень та модифікацій. Це дає досвідченим програмістам можливість додавати нові функції та вдосконалювати платформу. Мова програмування Arduino, що базується на C++, також може бути розширена за допомогою багатьох доступних бібліотек C++; технічні деталі мови також скопійовані з мови програмування AVR C, тому він приймає коди AVR-C, ніби вони були написані в Arduino IDE.

- Інноваційні технології впроваджені в Arduino. При появі Arduino на ринку мікроконтролерів, вже існували деякі інші платформи. Однак завдяки впровадженню передових технологій, зокрема високої швидкості обробки даних, Arduino швидко стала лідером серед конкурентів і визнаною найкращою платформою для професіоналів.
- Простий інтерфейс. Arduino відзначається простим інтерфейсом, завдяки розширюваним контактам, до яких можна підключати зовнішні модулі, такі як USB, що полегшує передачу ресурсів. Крім того, доступний широкий спектр API та інтерфейсів програмного забезпечення, що сприяє зручності в роботі.Широкий набір датчиків. Arduino поставляється з кількома датчиками, що надає йому переваги при виборі користувачів інструментів для прототипування систем, які включають той чи інший тип візуалізації.
- Величезна спільнота користувачів та ентузіастів-новаторів. Arduino об’єднав поціновувачів робототехніки, які обмінюються досвідом, ноу-хау, демонструють свої проєкти, дають рекомендації.
- Підходить для початківців (школярів та студентів). Більшість функцій Arduino можна освоїти, переглянувши відеоролики в Інтернеті або задавши запитання в онлайн-спільноті Arduino. Завдяки легкій у використанні платформі, новачки можуть набути досвіду, отримати нові навички та компетенції в робототехніці та програмуванні.

Водночас, поряд із зазначеними вище перевагами Arduino має низку обмежень:

1) Обчислювальна потужність. Платформи Arduino мають обмежену обчислювальну потужність порівняно з іншими мікроконтролерами та комп'ютерними системами. Це може обмежити їх здатність виконувати складні завдання та обчислення, особливо у додатках реального часу.

2) Пам'ять. Платформи Arduino також обмежені обсягом оперативної та флеш-пам'яті. Це може бути обмеженням для проектів, яким потрібний великий обсяг сховища даних, або для додатків, які потребують багато пам'яті для операцій.

3) Вартість. Хоча самі платформи Arduino є відносно дешевими, вартість додаткових компонентів, таких як датчики, приводи та пристрої зв'язку, може швидко збільшити загальні витрати на проект.

4) Швидкість. Одним з обмежень плат Arduino є обмежена тактова частота, що може вплинути на продуктивність програм реального часу, які вимагають швидкого збору даних або керування двигуном.

Попри зазначені обмеження, платформа Arduino залишається надзвичайно популярною серед винахідників та розробників інноваційних продуктів. Вона широко використовується для реалізації різноманітних ідей і вирішення проблем у різних галузях діяльності. Arduino здійснюється в таких сферах, як проектування систем, розробка загального призначення, апаратний зв'язок, прототипування програмного забезпечення, домашня та загальна автоматизація, сільське господарство, охорона здоров'я, гірничо-добувна промисловість, енергетика та захист довкілля, освіта і багато інших. Широке застосування Arduino можна спостерігати у створенні прототипів автономних транспортних засобів, як показує дослідження, зроблене у роботі [36].

Інструментарій для прототипування мобільних роботів у 2022 році отримав значні покращення, зокрема випуск Arduino IDE 2.0. Ця версія перейшла з бета-версії на стабільну та внесла значні зміни, що зробило її потужним та сучасним редактором. Користувачам тепер легше зосередитися на специфічних для Arduino функціях.

Також нова IDE має безпосередню інтеграцію з Arduino Cloud, що дозволяє продовжити роботу у веб-редакторі на основі браузера з того місця, де вони зупинилися. Це забезпечує зручність та доступність для користувачів, які працюють над проектами з різних пристроїв.

Нова IDE базується на потужному інтерфейсі командного рядка Arduino, що значно спрощує базову бібліотеку та робить розробку більш ефективною. Крім того, було створено новий інтерфейс командного рядка Arduino Cloud.Arduino Cloud CLI - це інструмент, створений новим Arduino для керування пристроями Інтернету речей в масштабі. Він дозволяє ініціалізувати, отримувати оновлення та надсилати дані пристроїв з використанням командного рядка. Це робить процес керування пристроями більш зручним та ефективним, дозволяючи швидко взаємодіяти з ними в реальному часі через командний рядок інтерфейсу. Цей інструмент допомагає спростити розгортання та управління масштабними мережами пристроїв IoT, забезпечуючи швидкий та ефективний доступ до них для виконання різних завдань та операцій.

Постановка задачі

Дослідження зазначає, що розробка системи навігації для самохідної техніки є складною завданням через необхідність створення програмного та апаратного забезпечення, яке дозволить автономному транспортному засобу розуміти навколишнє середовище та ухвалювати рішення в режимі реального часу.

Arduino може виявитися корисною платформою для розробки такого прототипу SDC з наступними характеристиками:

Простота та доступність: Arduino відома своєю простотою використання та доступністю для широкого кола користувачів. Це дозволить дослідникам швидко почати розробку прототипу без значних витрат часу на навчання нових інструментів чи технологій.

Широкий вибір датчиків та модулів: Arduino підтримує велику кількість датчиків, що дозволяє легко інтегрувати різноманітні сенсори, необхідні для

навігації і виявлення перешкод.

Відкритий вихідний код та спільнота користувачів: Багато бібліотек та рішень для Arduino є відкритими та безкоштовними, що сприяє швидкому розвитку та спільнотній підтримці. Користувачі можуть також обмінюватися досвідом та розробками, що полегшує вирішення складних завдань.

Можливість реалізації алгоритмів у реальному часі: Arduino може використовуватися для реалізації алгоритмів обробки даних та ухвалення рішень у режимі реального часу, що важливо для системи навігації самохідного транспортного засобу.

Отже, використання Arduino для розробки прототипу SDC може забезпечити швидке і ефективне розв'язання технологічних проблем, пов'язаних з навігацією та управлінням самохідною технікою.

Це дозволить зробити оцінку обраного підходу та створить підґрунтя для його вдосконалення та проведення подальших досліджень.

Висновки

Згідно з аналізом у наукових працях з автоматизації та роботизації транспортних засобів, популярні терміни для позначення самохідної техніки, яка пересувається без участі людини-водія, включають «автономний автомобіль», «безпілотний автомобіль», «самокерований автомобіль» та «робот-автомобіль». Незважаючи на цю широку лексику, у реальності автономних транспортних засобів рівня 5 наразі не існує, існують лише концептуальні розробки, які перебувають на початковій стадії опрацювання. Такі машини рівня 4 є лише поодинокими зразками, що проходять тестування, тоді як компанії намагаються досягти рівня 3 автономного водіння. Очікується, що протягом найближчих років проєкти зосередяться переважно на рівні 2 та його вдосконаленій версії.

Архітектура самохідних транспортних засобів складається з великих і складних систем, які оснащені різними датчиками для моніторингу як внутрішнього, так і зовнішнього середовища, і здатні генерувати великі обсяги даних за короткий час. Функціональні блоки здійснюються на основі потоку

інформації та етапів обробки, що реалізуються від збору даних до управління транспортним засобом. Система навігації SDC базується на технологічних та функціональних блоках, що забезпечують здатність машини планувати свій маршрут і виконувати його без втручання людини. Це включає розробку навігаційних алгоритмів та використання відповідних апаратних та програмних засобів для уникнення перешкод на шляху.

3.1 Опис функціонування системи

Автономна навігація

Забезпечення автономності пересування самохідних транспортних засобів (SDC) чи мобільних роботів включає в себе використання різноманітних апаратних та програмних компонентів, які визначають особливості їх динаміки і кінематики. Одним з важливих аспектів є автономна навігація, яка означає здатність транспортного засобу планувати свій маршрут та прокладати його без втручання людини. У процесі планування маршруту можуть використовуватися різні методи, включаючи дистанційні навігаційні засоби або дані від датчиків на борту транспортного засобу [38-40].

Українські вчені визначають навігацію як теорію та практику напрямку руху об'єктів за заданою траєкторією. [41].

Мета навігації полягає в знаходженні оптимальних маршрутів переміщення між заданими точками простору з урахуванням перешкод. У робототехніці існує кілька типів навігаційних схем: [41].

У робототехніці виділяють низку навігаційних схем, які поділяють на три категорії [9]:

- глобальна – визначення абсолютних координат пристрою під час руху довгими маршрутами;

- локальна – визначення координат пристрою по відношенню до деякої (зазвичай стартової) точки (ця схема затребувана розробниками наземних роботів,

які виконують місії в межах наперед відомої області);

- персональна – позиціонування роботом частин свого «тіла» та взаємодія з прилеглими предметами, що актуально для пристроїв, забезпечених маніпуляторами.

Кожна з цих схем має свої виклики та переваги в залежності від конкретного застосування робота або транспортного засобу.

Як приклад локальної навігаційної системи на рисунку 3.1 подано структурну схему.

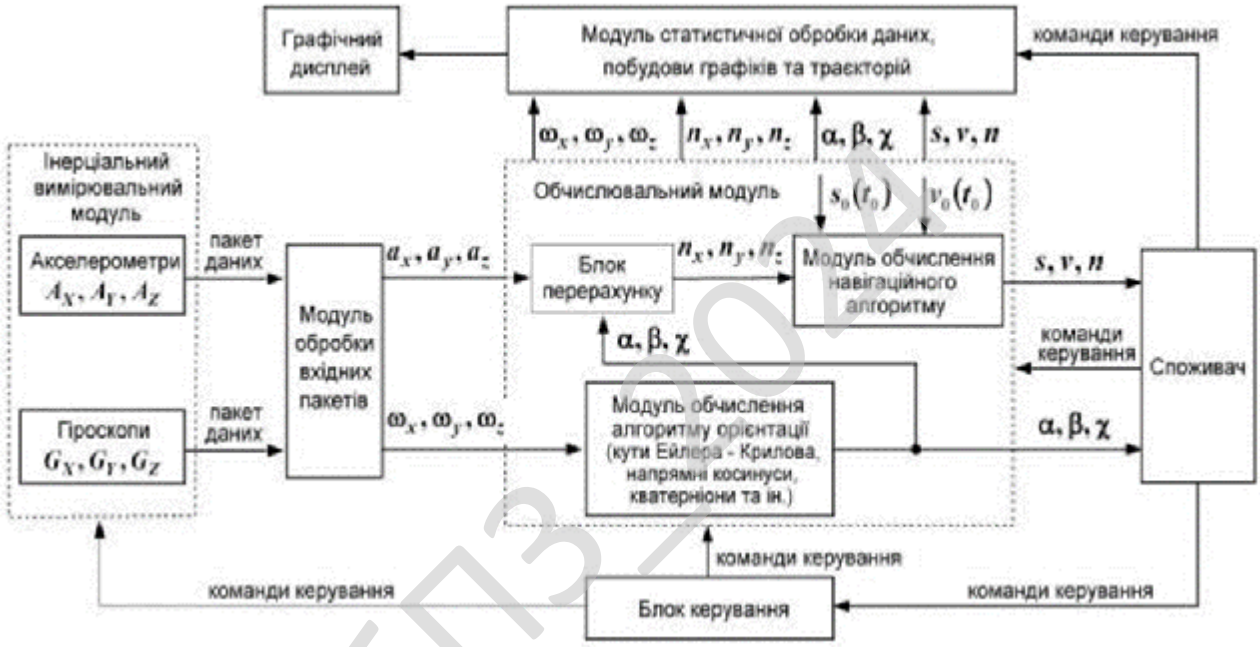

Рисунок 3.1 - Структурна схема локальної навігаційної системи

Так, системи навігації можуть бути класифіковані ще й за ознакою активності. Пасивна система навігації отримує інформацію про власні координати та характеристики руху від зовнішніх джерел, тоді як активна система намагається визначити своє місцезнаходження самостійно, відправляючи сигнал і сприймаючи його відбиття. Зазвичай глобальні схеми навігації є пасивними, локальні можуть бути обох типів, а персональні завжди активні.

Навігація - це ключова особливість, яка відрізняє автономних мобільних роботів. Це методологія, що дозволяє керувати роботом у навколишньому середовищі, використовуючи інформацію з датчиків. Ця здатність робить робота

або транспортний засіб спроможними працювати у різних умовах та виконувати різноманітні завдання [42, с. 77].

Так, компетенції, пов'язані з автономною навігацією, можна узагальнити на чотири основні: побудова карти, самолокалізація, інтерпретація карти та планування шляху.

Побудова карти: Ця компетенція відноситься до здатності робота або транспортного засобу створювати представлення про навколишнє середовище, щоб мати уявлення про територію, яку він вже проінспектував.

Самолокалізація: Це здатність робота або транспортного засобу визначати своє поточне місцезнаходження на побудованій карті. Це може включати в себе використання датчиків, таких як глобальні системи позиціонування (GPS), акселерометри, гіроскопи та інші сенсори.

Інтерпретація карти: Ця компетенція полягає в тому, щоб розуміти та аналізувати інформацію на побудованій карті, таку як перешкоди, дороги, межі території та інші об'єкти, що впливають на маршрут руху.

Планування шляху: Це здатність робота або транспортного засобу розробляти оптимальний маршрут з поточної позиції до заданої точки при врахуванні інформації на карті та обмежень, таких як перешкоди чи обмеження швидкості. Нарешті, Інтерпретація карти дійсно стосується здатності робота використовувати інформацію, яка міститься на карті, для навігаційних завдань, таких як планування шляху. Це включає в себе розпізнавання та аналіз об'єктів на карті, таких як дороги, перешкоди, межі території та інші важливі візуальні елементи, які можуть впливати на шлях руху.

Обчислення шляху та навігація на основі орієнтирів є важливими елементами для багатьох систем автономної навігації. Ці методи дозволяють автономним машинам визначати оптимальний шлях до цілі, враховуючи різні фактори, такі як перешкоди, обмеження шляху та поточні умови дороги.

Для обчислення шляху часто використовуються алгоритми шляхового планування, такі як алгоритм A*, Dijkstra та інші. Ці алгоритми оцінюють шляхи

від поточного положення до цілі та вибирають найоптимальніший шлях з урахуванням різних чинників. Уникання перешкод також важливе для безпеки та ефективності навігації. Автономні машини використовують внутрішні та зовнішні сенсори, такі як лідари, радари та камери, для виявлення перешкод на шляху та планування обходу їх. Урахування обмежень шляху, таких як швидкісні обмеження, заборонені зони та інші фактори, також допомагає забезпечити безпеку та ефективність навігації. У випадку, якщо з'являється необхідність, системи навігації можуть шукати альтернативні маршрути для обходу перешкод або врахування змін у поточних умовах дороги.

На думку автора дослідження [43, с. 351], для забезпечення автономії у навігації найбільш широко використовують модульну структуру для сприйняття та обробки інформації. Складний механізм пересування транспортним засобом реалізується у рамках так званої автономної навігаційної системи (autonomous navigation system) – системи, здатної планувати свій шлях і виконувати свій план без втручання людини. Це комбінація складних систем, що допомагає приймати рішення з урахуванням зондування навколишніх ситуацій. Технології автономної навігаційної системи спираються на такі передові розробки: інерційна навігаційна система, супутникова навігаційна система, радари, камери, ультразвукова та акустична навігація, а також з використанням автономних навігаційних алгоритмів для точної та безпечної навігації транспортних засобів.

У роботі [44] інтерпретація системи навігації роботом як інтелектуальної системи управління (ІСУ) дозволяє розглядати її як складну систему, яка взаємодіє з різними аспектами фізичного середовища для досягнення певних цілей, таких як безпека, ефективність та точність руху.

У такій ієрархічній структурі системи навігації роботом можна виділити три рівні:

1) Високорівневий рівень (планування маршруту): На цьому рівні вирішується стратегічне планування маршруту, вибір цілей та загальних напрямків руху. Система може враховувати глобальні обмеження, такі як шляхи, обмежені

зони або цілі, а також інші фактори, що впливають на ефективність руху.

2) Середньорівневий рівень (траєкторія та обходження перешкод): На цьому рівні система розробляє деталізовані траєкторії руху, враховуючи різноманітні фактори, такі як перешкоди на шляху, обмеження швидкості та інші умови. Тут можуть використовуватися алгоритми обходження перешкод та планування маршруту.

3) Низькорівневий рівень (керування рухом): На цьому рівні відбувається безпосереднє керування рухом робота. Система видає команди моторам, сервоприводам та іншим приводам для реалізації запланованої траєкторії руху, враховуючи реальні умови дороги та динаміку робота.

Ця ієрархічна структура дозволяє розділити завдання навігації на кілька рівнів складності та абстракції, що спрощує розробку та управління системою навігації роботом. (рисунок. 3.2).

Відповідно до [45, с. 89], система контролю ухилення від перешкод є однією з ключових складових автономної навігації транспортних засобів, таких як інтелектуальні транспортні засоби. Її основна мета полягає у забезпеченні безпеки та ефективності руху, униканні зіткнень та забезпеченні стабільного та плавного переміщення навіть у важких умовах дорожнього руху або на території з різноманітними перешкодами.

Дослідження в галузі автономної навігації зосереджене на розробці ефективних алгоритмів долання перешкод. Ці алгоритми повинні дозволити транспортним засобам уникати перешкод, які можуть включати інші автомобілі, пішоходів, дорожні знаки, будівлі тощо. Зазвичай такі алгоритми базуються на обробці даних від різноманітних сенсорів, таких як лідари, радари, камери та ультразвукові датчики.

Основні виклики включають в себе розробку алгоритмів для точного виявлення перешкод, визначення оптимального шляху обходу цих перешкод та врахування різних умов дорожнього руху, таких як швидкість, рух інших транспортних засобів та пішоходів. Дослідники також працюють над розвитком

алгоритмів для управління реагуванням на непередбачені ситуації та забезпеченням безпеки в різних сценаріях дорожнього руху.

Загалом, розвиток ефективних алгоритмів долання перешкод в системах навігації є важливим кроком у напрямку створення безпечних та надійних автономних транспортних засобів [46-48].

Автор [49] Ця архітектурна модель навігації для автономних транспортних засобів (SDC) в статичному середовищі дійсно відображає складність процесу прийняття рішень та управління в умовах, коли середовище не зазнає значних змін.

Стратегічний рівень: на цьому рівні приймаються високорівневі стратегічні рішення, спрямовані на планування та координацію загального напрямку руху. Це може включати вибір загальної маршрутної стратегії та визначення загальної мети подорожі.

Тактичний рівень: на цьому рівні приймаються рішення про вибір конкретних маршрутів та управління рухом відповідно до поточних умов середовища. Тут розробляються тактики обходу перешкод, управління швидкістю, вибір смуг руху тощо.

Виконавчий рівень: це найнижчий рівень, де конкретні команди виконуються для реалізації прийнятих рішень. На цьому рівні відбувається безпосереднє управління рухом автономного транспортного засобу, таке як керування кермом, гальмами та прискорювачами.

У динамічних середовищах однієї з основних проблем є необхідність швидкого прийняття рішень в умовах обмежень часу та обчислювальних ресурсів. Це ускладнюється тим, що інформація про середовище може змінюватися динамічно, а рішення, прийняті на вищих рівнях, можуть бути застарілими або неадекватними через ці зміни. Тому виникає потреба у швидкому та ефективному обміні даними між різними рівнями та системами для забезпечення адаптивності та реагування на зміни в середовищі.

3.2 Розробка структурної схеми

Структурна схема надає інформацію про взаємодію майбутніх частин програми, за якій функціонал вони будуть відповідати, та як будуть взаємодіяти між собою. Не можна сказати що вони є інформативними, оскільки на них не можна відстежити як дані передаються та змінюються. Тому в залежності від технічного завдання, структурні схеми розробляють для окремих частин програм.

Для розробки структурної схеми зазвичай використовують метод покрокової деталізації, щоб вказати всі компоненти з яких складається схема, а також набори підпрограм та підключені бібліотеки.

Структурними компонентами програми можуть називатися підсистеми, бази даних, бібліотеки і т.д. А за допомогою зв'язків можна продемонструвати як вони взаємодіють між собою, обмінюються інформацією, підключати нові бібліотеки. При розробці структурної схеми програмного забезпечення були розглянуті основні модулі програми, та їх взаємодія.

Дуже важливо під час розробки, особливо якщо проект великий і має складну архітектуру, поділити програму на структури, щоб відображати логіку програми, показати місця де підключаються бібліотеки, та за що вони відповідають.

В наведеній схемі, взаємодія модулів, відображена в схемі ієрархії. Де до головного модуля підключаються розроблені модулі, однак схема не відображає порядок функціонування системи. Схема доповнюється розшифровкую функцій які виконують підключення модулів.

У структурній схемі (рисунок. 3.2) позначені зовнішні специфікації програми, які дають розуміння функціональній частині програми, за що вони відповідають, та як вони взаємодіють з вхідними та вихідними даними. Особливо важливо розуміти тип структури даних.

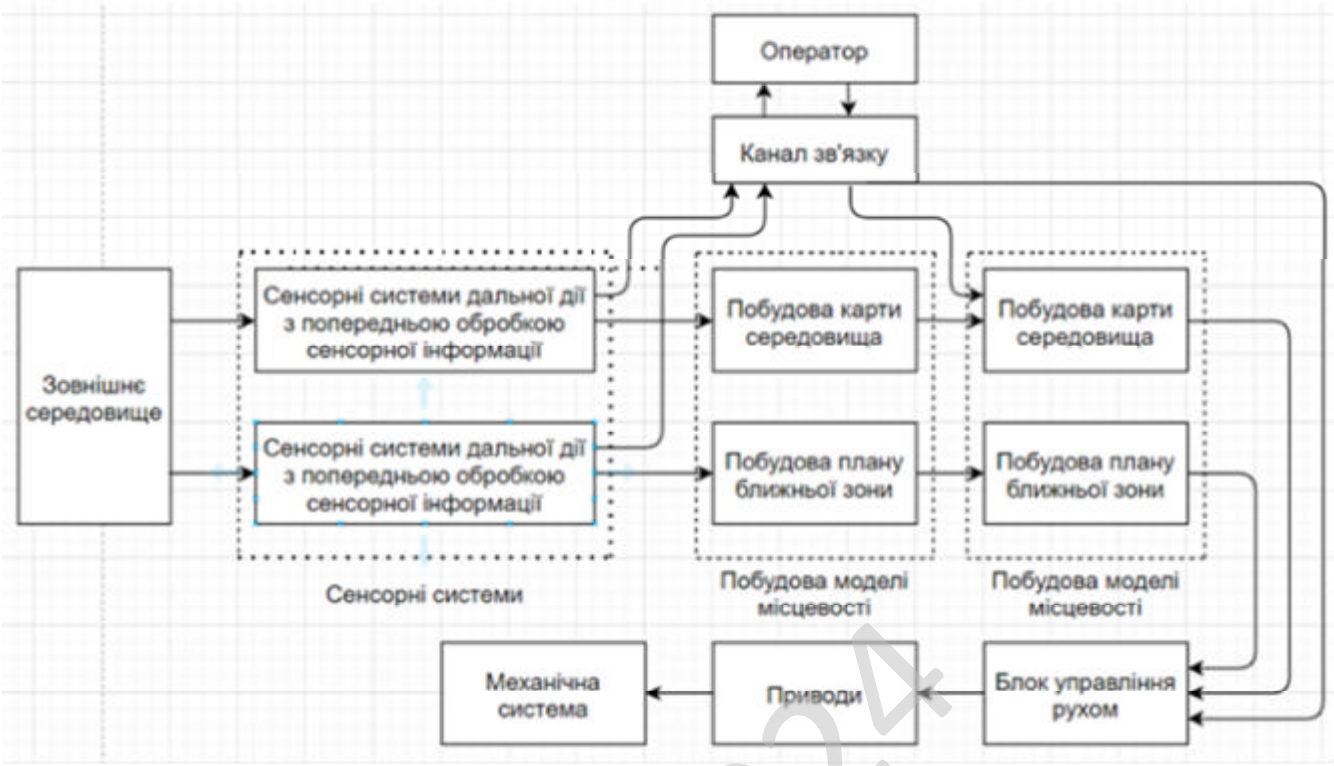

Рисунок 3.2 - Структурна схема

3.3 Розробка функціональної схеми

Пропонована архітектура навігації для системи автономних транспортних засобів (SDC) враховує різні методи керування мобільними роботами та важливі аспекти функціонування в динамічних середовищах. Розглянемо детальніше кожен рівень:

1) Реактивні методи керування: Ці методи дозволяють реагувати на безпосередні події та обставини без значного оброблення інформації. Система навігації може швидко реагувати на зміни в середовищі, використовуючи прості правила або алгоритми.

2) Методи для виконання шаблонних послідовностей діяльності: Ці методи використовуються для виконання стандартних або попередньо вивчених послідовностей дій без значного аналізу поточних умов.

3) Складні обчислювальні методи для прийняття оптимальних рішень: На цьому рівні використовуються складні алгоритми та методи для аналізу та прийняття оптимальних рішень з урахуванням поточних умов та обмежень.

У японській моделі системи навігації реального SDC звертається увага на зчитування (зондування) середовища, розпізнавання об'єктів, прийняття рішень та контроль. Ці етапи дозволяють системі отримувати, аналізувати та реагувати на інформацію з навколишнього середовища для ефективної навігації.

У контексті обох підходів важливо враховувати потреби та характеристики конкретної системи SDC, а також умови її експлуатації. Розробники повинні брати до уваги не лише технічні аспекти, а й правові та етичні питання, пов'язані з автономним рухом транспортних засобів.

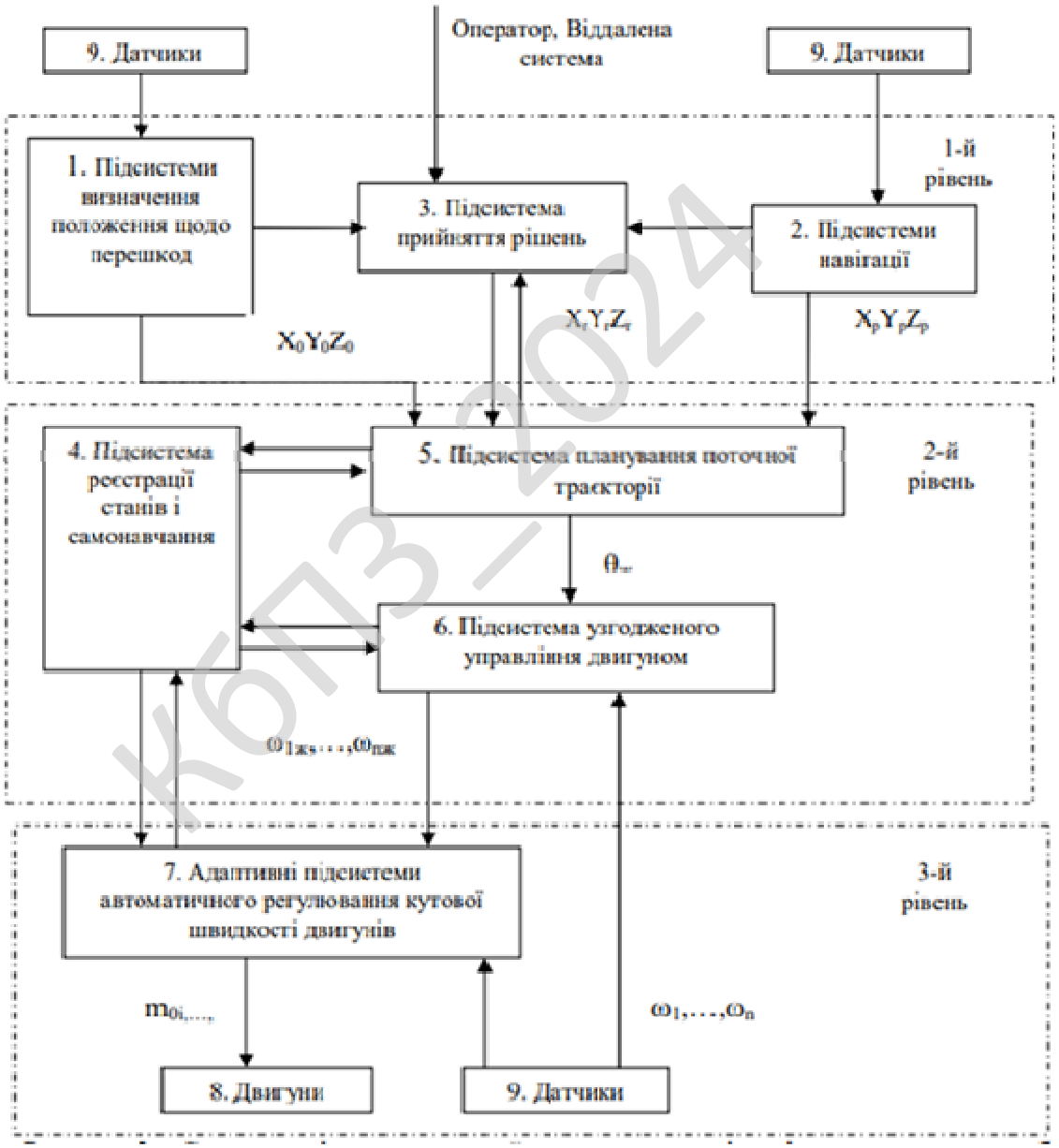

Рисунок 3.3 - Функціональна схема системи

Процес прийняття рішень та контролю, включає планування руху SDC, визначення можливих траєкторій, оцінку траєкторії та навігація SDC обраною траєкторією. "Прогнозування руху об'єктів" полягає у передбаченні майбутньої поведінки транспортних засобів, пішоходів та інших об'єктів навколо автономних автомобілів. Це важливо для виявлення потенційних загроз та уникнення аварій шляхом відповідності руху автомобіля передбачуваним діям інших учасників дорожнього руху. "Формування динамічної карти" включає створення та постійне оновлення просторових карт, які відображають поточну ситуацію на дорозі, у тому числі рух інших транспортних засобів, пішоходів, дорожніх знаків, сигналізацію та інше. Ці карти також містять інформацію про потенційні небезпечні ситуації та ризики. Динамічні карти використовуються для прийняття рішень автономною системою щодо навігації та управління, забезпечуючи безпеку та ефективність руху автономного автомобіля.

Традиційні системи навігації використовують модулі, спрямовані на різні завдання, такі як локалізація, картографування, виявлення об'єктів, планування руху та керування кермовим керуванням. Навігація має велике значення в різних сферах, включаючи створення та відстеження карт, планування маршрутів та виявлення та уникнення перешкод. Сенсорні системи є основою для забезпечення автономності транспортних засобів і забезпечення точної локалізації та планування маршрутів.

3.4 Розробка діаграми процесів

Є різні методи побудови діаграм, які допомагають відстежити логіку та послідовність дії в процесах, але найкраще себе зарекомендував метод погрокового алгоритму побудови діаграм.

З усіх вище перелічених методів, та технічних рішень при проектуванні, та розробки програмного забезпечення, дивлячись на результати, можна зробити висновки, що я обрав оптимальні проектні рішення, які задовольняють головні

потреби, які були поставлені переді мною на початку проекту. Оскільки з результатів можна побачити, що програма виконує всі поставленні задачі.

Ця діаграма представляє основні етапи роботи системи управління роботом на Arduino:

- **Ініціалізація**: Ініціалізація системи та підготовка до роботи.
- **Очікування команд**: Система очікує отримання команди або інструкцій для виконання.
- **Перевірка отриманих команд**: Перевірка та інтерпретація отриманих команд.
- **Керування рухом**: Керування рухом робота на основі отриманих команд.
- **Взаємодія з сенсорами**: Збір інформації від сенсорів, яка може впливати на рух робота.
- **Зчитування даних та аналіз вхідних сигналів**: Обробка і аналіз вхідних даних від сенсорів.
- **Виконання дій на основі отриманих даних**: Виконання відповідних дій роботом на основі результатів аналізу даних.

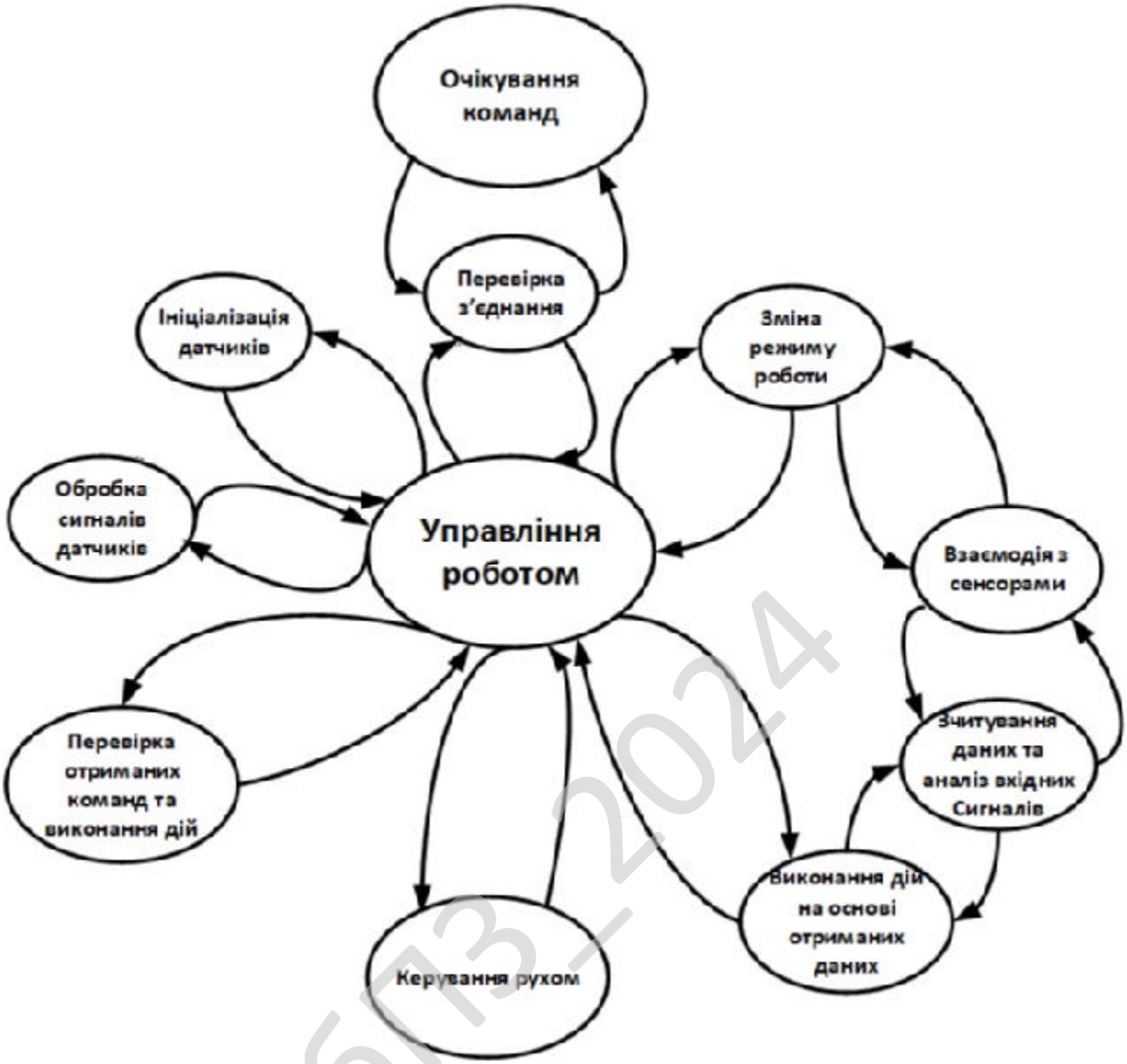

Рисунок 3.4 – Діаграма процесів системи

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

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Алгоритми навігації

У навігації мобільних роботів застосовуються різноманітні методи [53], спрямовані на уникнення перешкод, що є однією з основних проблем у мобільній робототехніці. Ці методи можна розділити на три групи відповідно до базового алгоритму:

- детермінований алгоритм;
- недетермінований (стохастичний) алгоритм;
- еволюційний алгоритм (отриманий гібридизацією зазначених вище двох алгоритмів).

Рисунок 4.1 демонструє загальну класифікацію алгоритмів, які були реалізованими у системах навігації мобільних роботів різними авторами. Загальна класифікація детермінованого алгоритму, недетермінованого (стохастичного) алгоритму та еволюційного алгоритму, які використовуються для навігації мобільного робота.

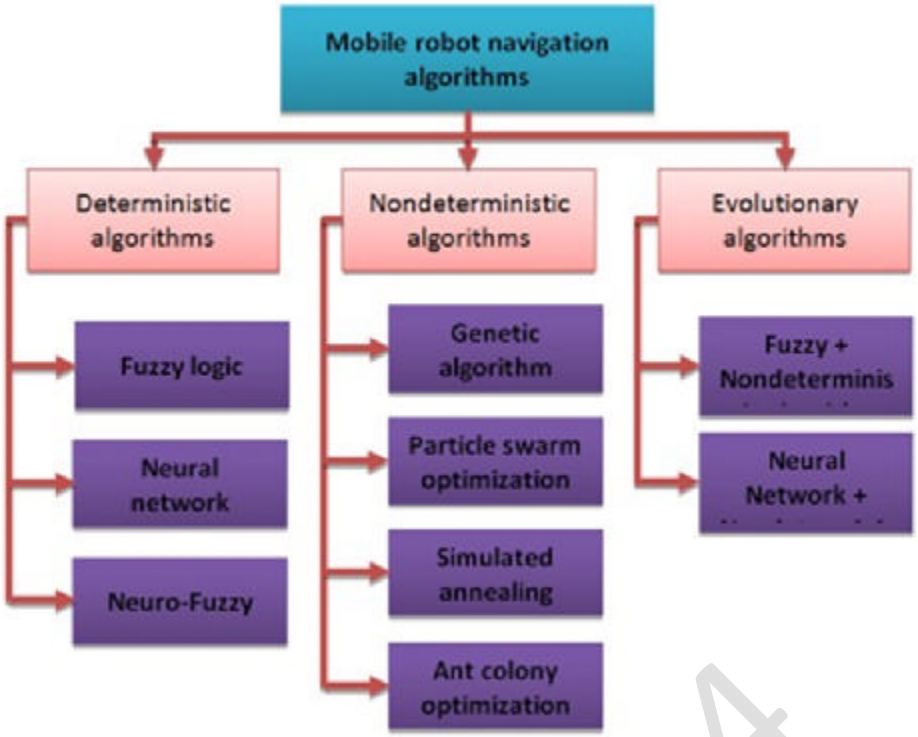

Рисунок 4.1 – Алгоритми навігації мобільних роботів

Останнім часом було запропоновано численні методи програмного забезпечення для розв'язання проблем навігації та уникнення перешкод у різних статичних і динамічних середовищах мобільних роботів. Один із таких методів - це використання нечіткої логіки (Fuzzy logic) для навігації мобільними роботами. Концепція нечіткої логіки, розроблена Заде (Zadeh), широко застосовується в багатьох інженерних сферах, включаючи мобільну робототехніку.Техніка нечіткої логіки була успішно застосована багатьма дослідниками для контролю положення та орієнтації мобільного робота в навколишньому середовищі. Існуючі дослідження детально описують використання інтелектуальних контролерів нечіткої логіки для навігації колісних мобільних роботів у невідомих та змінних середовищах. Вони демонструють ефективність оптимального нечіткого контролера Такагі-Сугено, який налаштовується за допомогою градієнтного методу для налаштування параметрів функцій приналежності. Ці дослідження також пропонують нечітку архітектуру на основі поведінки для навігації мобільних роботів у невідомих середовищах, використовуючи базові моделі

поведінки, такі як прагнення до мети, уникнення перешкод та відстеження. Ці методи тестувалися в різних симуляційних середовищах для підтвердження їх ефективності. Інші дослідники [54] об’єднали нечіткий генетичний алгоритм для розв'язання проблеми планування шляху та керування автономним мобільним роботом, використовуючи інформацію від ультразвукового далекоміра. Дослідники розробили конфігурацію ультразвукових датчиків, побудували динамічну модель та кінетичні рівняння для мобільного робота, і розробили правила уникнення та вибору шляху у відповідності з перешкодами, а також блок-схему навігації робота в середовищі з кількома перешкодами.

Алгоритми планування маршруту

Планування маршруту для автономних транспортних засобів включає в себе пошук оптимального шляху від початкового положення до цілі, враховуючи наявність перешкод на шляху. Це може вимагати розробки стратегій уникнення перешкод та вибору альтернативних шляхів.

На сьогоднішній день існує ряд алгоритмів для управління уникненням перешкод та плануванням маршруту. Вони починаються з простих стратегій, таких як зупинка автомобіля у разі виявлення перешкоди, і закінчуються більш складними методами, що адаптивно реагують на різні типи, розміри та інші особливості виявлених перешкод. Ці алгоритми відрізняються кількістю даних, необхідних для обробки, кількістю датчиків, операційною швидкістю, просторовою складністю, ефективністю та стратегією керування.

У роботі [55], детальний огляд типових алгоритмів, які використовуються в системах керування безпілотним транспортним засобом:

- **Алгоритми помилок**: Ці алгоритми зазвичай використовуються для простих систем і полягають у тому, щоб коригувати шлях або руху автомобіля, якщо виникають помилки у відслідковуванні маршруту.
- **Наївні алгоритми**: Це найпростіші алгоритми, які можуть включати прості стратегії вибору напрямку руху без урахування оточуючого середовища.

- **Алгоритми з використанням датчиків відстані**: Ці алгоритми використовують відстаневі датчики для виявлення перешкод та уникнення ними. Вони дозволяють уникнути зіткнень, враховуючи відстань до об'єктів.

- **Потенційні польові алгоритми**: Ці алгоритми моделюють потенціальне поле навколо автомобіля, де перешкоди визначаються як області з високим потенціалом. Автомобіль рухається в напрямку, що сприяє мінімізації потенціалу.

- **Алгоритми на основі графів**: Вони моделюють середовище як граф, де вершини відповідають можливим положенням, а ребра - можливим рухам. Алгоритми пошуку в графі дозволяють знаходити оптимальний шлях.

- **Формальні алгоритми**: Наприклад, алгоритм Дейкстри і алгоритм Флойда-Воршала. Вони використовуються для пошуку найкоротшого шляху у вагованому графі.

- **Евристичні алгоритми**: Наприклад, пошук по ширині. Ці алгоритми дають швидкі результати, хоча не завжди забезпечують найоптимальніші маршрути.

- **Гібридні алгоритми пошуку**: Наприклад, A* і D*. Вони поєднують у собі переваги різних підходів для досягнення більш ефективного планування маршруту.

Ці алгоритми можуть бути використані для ефективного управління безпілотним транспортним засобом, забезпечуючи оптимальний та безпечний шлях до мети (наприклад, A*, D*).

Алгоритми находження шляху в русі, або алгоритми BUG, є широко використовуваними в системах керування безпілотними транспортними засобами, оскільки вони дозволяють знаходити оптимальний шлях до цілі в умовах, коли перешкоди невідомі заздалегідь. Ось деякі з найпоширеніших модифікацій цих алгоритмів:

- **Alg1 і Alg2**: Ці базові алгоритми використовують прості стратегії навігації, такі як обходження перешкод і знаходження шляху до мети,

використовуючи прості правила.

- **DistBug**: Цей алгоритм використовує відстаневі датчики для визначення розташування перешкод та розрахунку оптимального шляху.
- **Class1, Rev1 і Rev2**: Ці модифікації алгоритмів базуються на класифікації типів перешкод та використанні змінних стратегій навігації в залежності від типу перешкоди та її розташування.
- **OneBug**: Цей алгоритм використовує принцип одного напрямку обходження перешкоди для знаходження шляху до цілі.
- **LeaveBug і TangentBug**: Ці модифікації використовують інші стратегії обходження перешкод, такі як обходження відстані від перешкоди або обходження перешкоди за допомогою каскадних траєкторій.

Ці алгоритми мають свої переваги і недоліки, але їхня популярність полягає в їхній простоті і дешевизні обладнання, що робить їх привабливими для застосування у системах безпілотної навігації. Розглянемо окремі алгоритми цього сімейства.

Оригінальний алгоритм BUG, а точніше дві його модифікації Bug1 (рис. 4.2-4.3) та Bug2 (рис. 4.4-4.5) були запропоновані В. Люмельські та О. Степановим у 1987 р. Суть алгоритму Bug1 полягає в тому, що мобільний робот рухається безпосередньо до мети до тих пір, поки не зустріне перешкоду. Цей метод, який використовується для навігації робота у складних умовах, полягає у послідовному дослідженні контурів перешкод, поки шлях до мети не стане знову доступним. Якщо датчик виявляє перешкоду, робот починає рухатися уздовж межі цієї перешкоди проти годинникової стрілки, доки не знайде точку виходу - це найближча до мети точка на межі перешкоди. Після досягнення точки виходу робот продовжує рух у напрямку мети, доки не досягне її або не зустріне нову перешкоду. У випадку зустрічі нової перешкоди процедура повторюється. Цей метод може мати низьку ефективність через послідовність дій, проте він гарантує, що робот досягне будь-якої доступної точки призначення.

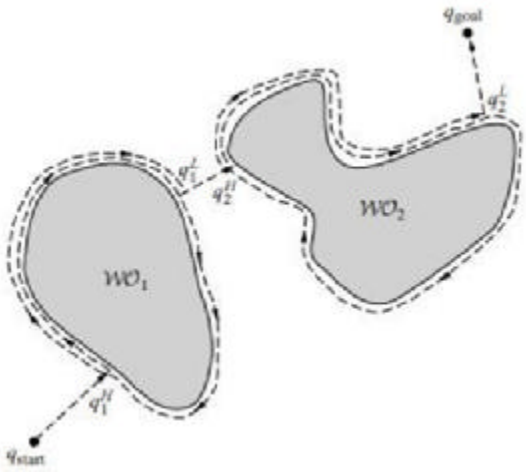

Рисунок 4.2 - Блок-схема головного модулю

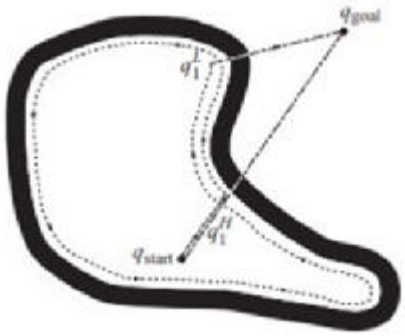

Рисунок 4.3 – Алгоритм Bug1 повідомляє, що мета недосяжна

Algorithm 1 Bug1 Algorithm

Input: A point robot with a tactile sensor

Output: A path to the q_{goal} or a conclusion no such path exists

```
while Forever do
    repeat
        From q_{i-1}^L, move toward q_goal.
    until q_goal is reached or an obstacle is encountered at q_i^H.
    if Goal is reached then
        Exit.
    end if
    repeat
        Follow the obstacle boundary.
    until q_goal is reached or q_i^H is re-encountered.
    Determine the point q_i^L on the perimeter that has the shortest distance to the goal.
    Go to q_i^L.
    if the robot were to move toward the goal then
        Conclude q_goal is not reachable and exit.
    end if
end while
```

Рисунок 4.4 – Алгоритм Bug1

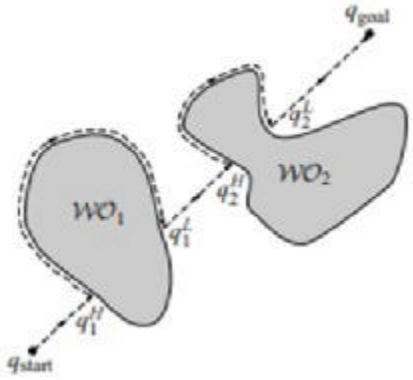

Рисунок 4.5 – Алгоритм Bug2 успішно знаходить мету

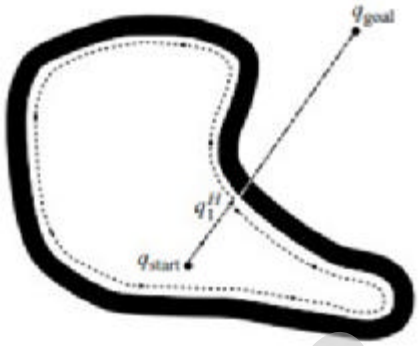

Рисунок 4.6 – Алгоритм Bug 2 повідомляє, що мета недосяжна

Algorithm 2 Bug2 Algorithm

Input: A point robot with a tactile sensor

Output: A path to q_{goal} or a conclusion no such path exists

while True **do**
  repeat
    From q_{i-1}^{L}, move toward q_{goal} along m-line.
  until
  q_{goal} is reached **or**
  an obstacle is encountered at *hit point* q_i^H.
  Turn left (or right).
  repeat
    Follow boundary
  until
   q_{goal} is reached **or**
   q_i^H is re-encountered **or**
   m-line is re-encountered at a point m such that
    $m \neq q_i^H$ (robot did not reach the hit point),
    $d(m, q_{goal}) < d(m, q_i^H)$ (robot is closer), and
    If robot moves toward goal, it would not hit the obstacle
  Let $q_{i+1}^{L} = m$
  Increment i
end while

Рисунок 4.7 – Алгоритм Bug 2

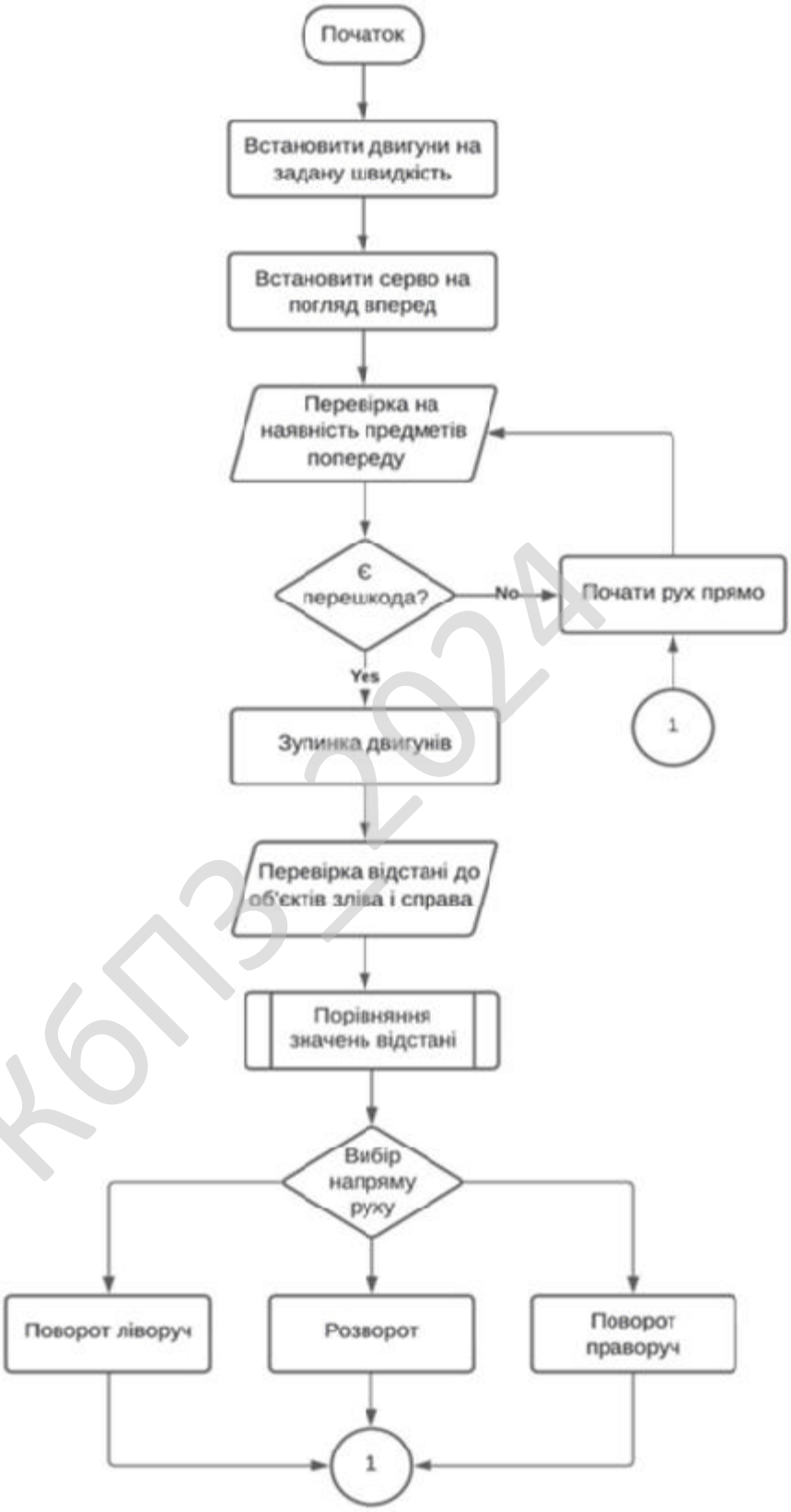

Рисунок 4.8 - Блок-схеми алгоритму роботи робота

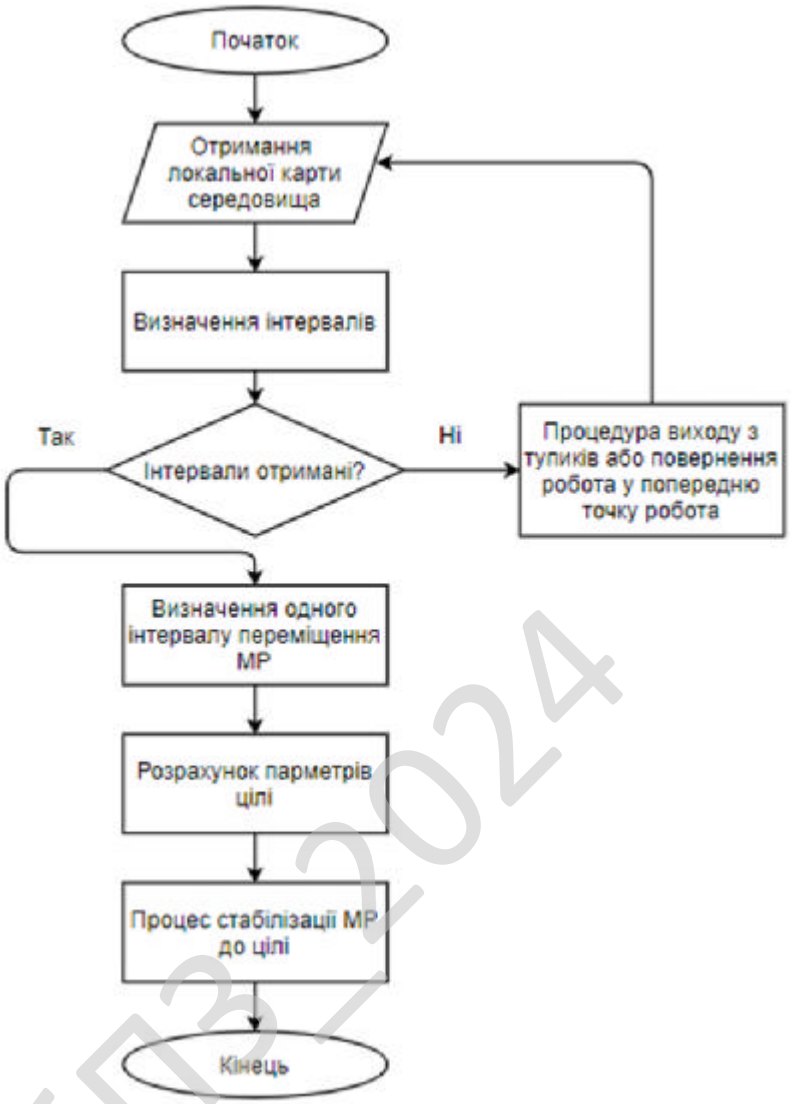

Рисунок 4.9 - Блок-схема роботи підпрограми стабілізації руху робота

4.2 Захист розробленого програмного забезпечення

Для захисту системи розробленої на контролері зазвичай виконуються наступні кроки:

Використання захищених з'єднань: Якщо ваш мікроконтролер взаємодіє з мережею чи іншими пристроями, використовуйте захищені протоколи зв'язку, такі як SSL або TLS, для шифрування комунікації.

Фізичний захист: Захистіть ваш мікроконтролер від фізичного доступу,

наприклад, встановивши його в кейс або корпус, щоб ускладнити несанкціонований доступ.

Використання автентифікації: Реалізуйте механізми автентифікації для перевірки ідентифікації користувачів чи пристроїв, що звертаються до мікроконтролера.

Обмеження доступу: Обмежте доступ до функцій та функціональності мікроконтролера лише для авторизованих користувачів.

Використання цифрових підписів: Підпишіть програмне забезпечення для мікроконтролера цифровим підписом, щоб переконатися, що воно не було змінене після підписання.

Використання бібліотек безпеки: Використовуйте перевірені бібліотеки безпеки для обробки даних, мережевої взаємодії та інших критичних аспектів програми.

Шифрування даних: Шифруйте конфіденційні дані, які зберігаються чи передаються вашим мікроконтролером, для захисту від несанкціонованого доступу.

Моніторинг та журналювання: Додайте механізми моніторингу та журналювання для виявлення незвичайної активності чи потенційних загроз безпеці.

Фірмварний захист: Розгляньте можливість застосування фірмварного захисту, який ускладнює витягання програмного коду з мікроконтролера.

Тестування безпеки: Регулярно проводьте тестування безпеки вашого програмного забезпечення та мікроконтролера для виявлення та виправлення вразливостей.

Розробка захищеного зєднання

```
  #include <WiFiNINA.h>
#include <WiFiSSLClient.h>

char ssid[] = "YourWiFiSSID";
char pass[] = "YourWiFiPassword";
```

```
WiFiSSLClient client;

void setup() {
  Serial.begin(9600);
  while (!Serial);

  // Connect to WiFi
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Connect to secure server
  if (client.connectSSL("example.com", 443)) {
    Serial.println("Connected to server");
    client.println("GET / HTTP/1.1");
    client.println("Host: example.com");
    client.println("Connection: close");
    client.println();
  }
}

void loop() {
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }
  if (!client.connected()) {
    Serial.println();
    Serial.println("Disconnecting from server");
    client.stop();
    while (true);
  }
}
```

Використання автентифікації

```
#include <EEPROM.h>
```

```
#define PASSWORD_ADDRESS 0

void setup() {
  Serial.begin(9600);
  while (!Serial);

  // Write password to EEPROM
  char password[] = "myPassword";
  for (int i = 0; i < strlen(password); i++) {
    EEPROM.write(PASSWORD_ADDRESS + i, password[i]);
  }
  EEPROM.write(PASSWORD_ADDRESS + strlen(password), '\0');
}

void loop() {
  // Code to authenticate user
}
```

Обмеження доступу

```
#define ADMIN_LEVEL 3
#define USER_LEVEL 2
#define GUEST_LEVEL 1

int userLevel = GUEST_LEVEL;

void setup() {
  Serial.begin(9600);
  while (!Serial);

  // Code to set user level (e.g., based on login)
}

void loop() {
  if (userLevel >= ADMIN_LEVEL) {
    // Code for admin users
  } else if (userLevel >= USER_LEVEL) {
    // Code for regular users
  } else {
    // Code for guest users
  }
}
```

Використання бібліотек безпеки

```
#include <SHA256.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  // Example of using SHA256 library
  SHA256 sha256;
  sha256.update("hello", 5);
  uint8_t* hash = sha256.finalize();
  Serial.print("Hash: ");
  for (int i = 0; i < SHA256_SIZE; i++) {
    Serial.print(hash[i], HEX);
  }
  Serial.println();
}

void loop() {
  // Code here
}
```

Шифрування даних

Використання шифрування може включати застосування алгоритмів, таких як AES. Шифрування AES (Advanced Encryption Standard) - це криптографічний алгоритм, який використовується для захисту конфіденційності даних шляхом їх шифрування. На мікроконтролерах, таких як Arduino, можна використовувати алгоритм AES для захисту конфіденційних даних, таких як паролі, ключі або інші важливі інформаційні ресурси.

```
#include <Crypto.h>
#include <AES.h>

AES aes;

void setup() {
  Serial.begin(9600);
  while (!Serial);
```

```
  byte key[16] = "1234567890123456";
  byte iv[16] = "1234567890123456";

  aes.set_key(key, 16);
  aes.set_IV(iv, 16);
}

void loop() {
  char plaintext[] = "Hello, world!";
  byte ciphertext[16];

  aes.do_aes_encrypt((byte*)plaintext,    strlen(plaintext),    ciphertext,
aes.get_size(), aes.get_key(), 128, aes.get_IV());

  Serial.print("Ciphertext: ");
  for (int i = 0; i < 16; i++) {
    Serial.print(ciphertext[i], HEX);
  }
  Serial.println();
}
#include <Crypto.h>
#include <AES.h>

AES aes;

void setup() {
  Serial.begin(9600);
  while (!Serial);

  // 128-bit key (16 bytes)
  byte key[16] = {0x2b, 0x7e, 0x15, 0x16, 0x28, 0xae, 0xd2, 0xa6, 0xab,
0xf7, 0x97, 0x1f, 0x7c, 0x3c, 0x1f, 0x76};

  // 128-bit initialization vector (16 bytes)
  byte iv[16] = {0};

  aes.set_key(key, 16);
  aes.set_IV(iv, 16);
}

void loop() {
  char plaintext[] = "Hello, world!";
```

```
  byte ciphertext[16];

  // Encrypt
  aes.do_aes_encrypt((byte*)plaintext,    strlen(plaintext),    ciphertext,
aes.get_size(), aes.get_key(), 128, aes.get_IV());

  Serial.println("Ciphertext:");
  for (int i = 0; i < 16; i++) {
    Serial.print(ciphertext[i], HEX);
  }
  Serial.println();

  // Decrypt
  char decryptedtext[16];
  aes.do_aes_decrypt(ciphertext,  16,  (byte*)decryptedtext,  aes.get_size(),
aes.get_key(), 128, aes.get_IV());
  decryptedtext[16] = '\0';

  Serial.print("Decrypted text: ");
  Serial.println(decryptedtext);

  while (true);
}
```

Моніторинг та журналювання

```
#include <SD.h>

File logFile;

void setup() {
  Serial.begin(9600);
  while (!Serial);

  if (!SD.begin(4)) {
    Serial.println("SD initialization failed!");
    return;
  }

  logFile = SD.open("log.txt", FILE_WRITE);
  if (logFile) {
    logFile.println("Starting log...");
    logFile.close();
```

```
  } else {
    Serial.println("Error opening log file");
  }
}

void loop() {
  // Log events
  logEvent("Event occurred");
  delay(1000);
}

void logEvent(String event) {
  logFile = SD.open("log.txt", FILE_WRITE);
  if (logFile) {
    logFile.println(event);
    logFile.close();
  } else {
    Serial.println("Error opening log file");
  }
}
```

Фірмварний захист

Використовується функція HAL_FLASHEx_OBProgram() для встановлення захищеного режиму роботи на мікроконтролері. При виклику цієї функції, мікроконтролер переходить в захищений режим, у якому доступ до основних функцій пам'яті та інших системних ресурсів обмежується.

Важливо пам'ятати, що цей код призначений для мікроконтролерів.

```
#include "main.h"
#include "stm32l4xx_hal.h"

/* Private variables */
CRC_HandleTypeDef hcrc;

/* Private function prototypes */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_CRC_Init(void);

int main(void)
```

```
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_CRC_Init();

  /* Enable the CRC Module */
  HAL_CRC_Init(&hcrc);

  /* Enable the Secure Mode */
  HAL_FLASHEx_OBGetConfig(&OBInit);
  OBInit.OptionType = OPTIONBYTE_USER;
  OBInit.USERType = OB_USER_SEC_CFG;
  OBInit.USERConfig = OB_SECURE_MODE_ENABLE;
  HAL_FLASH_Unlock();
  HAL_FLASH_OB_Unlock();
  HAL_FLASHEx_OBProgram(&OBInit);
  HAL_FLASH_OB_Launch();
  HAL_FLASH_Lock();

  while (1)
  {
    /* Your main loop code here */
  }
}

/* System Clock Configuration */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI48;
  RCC_OscInitStruct.HSI48State = RCC_HSI48_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_PCLK1
| RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI48;
```

```
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
}
/* CRC init function */
static void MX_CRC_Init(void)
{
  hcrc.Instance = CRC;
  if (HAL_CRC_Init(&hcrc) != HAL_OK)
  {
    Error_Handler();
  }
}

/* GPIO init function */
static void MX_GPIO_Init(void)
{
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
}
```

5 ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Апаратні засоби проєктування мобільного робота

Мікроконтролер Arduino служить керівним центром для мобільного робота, координуючи його рухи, обробляючи інформацію з сенсорів, та забезпечуючи виконання програм, необхідних для реалізації певних функцій та завдань.

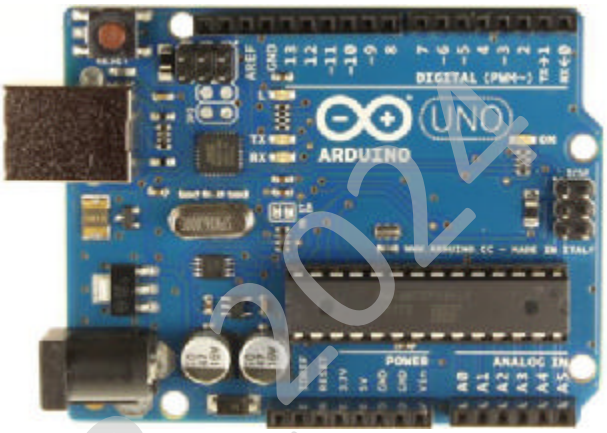

Рисунок 5.1 - Плата Arduino

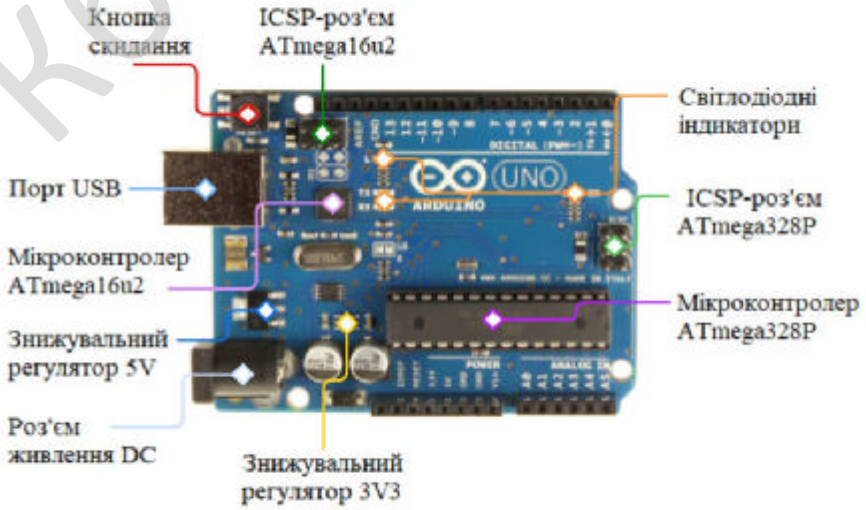

Рисунок 5.2 – Складові плати

Характеристики Arduino Uno наступні

Інтерфейси вводу/виводу: Arduino Uno оснащено 14 цифровими пінами для вводу/виводу, з яких 6 можуть бути налаштовані для використання як виходи для широтно-імпульсної модуляції (ШІМ). Це дозволяє управляти різноманітними пристроями, наприклад, світлодіодами, двигунами та сервоприводами.

Аналогові входи: Модель обладнана шістьма аналоговими входами, що дозволяють зчитувати аналогові сигнали, такі як напруга чи дані з сенсорів.

Пам'ять: Arduino Uno містить 32 кБ флеш-пам'яті, з яких 0,5 кБ займає програмний код. Також є 2 кБ оперативної пам'яті (SRAM) і 1 кБ EEPROM, яка використовується для збереження даних.

Комунікаційні порти: Плата має USB-порт для з'єднання з комп'ютером або іншими пристроями для програмування та обміну даними, а також роз'єм ICSP (In-Circuit Serial Programming) для програмування мікроконтролера.

Інтерфейси: Arduino Uno має USB-порт для підключення до комп'ютера або інших пристроїв для програмування та зв'язку. Вона також має роз'єм ICSP (In- Circuit Serial Programming) для програмування мікроконтролера.

Живлення: Arduino Uno можна живити як від зовнішнього джерела з напругою від 7 до 12 вольт, так і від USB-порту. Плата оснащена вбудованим регулятором напруги, що забезпечує стабільне живлення 5 вольт для мікроконтролера та підключених до нього пристроїв.

Розміри: Arduino Uno відрізняється компактними розмірами, що становлять приблизно 68.6 мм на 53.4 мм, що робить його ідеальним для застосування у різноманітних проектах.

Світлодіодна індикація на Arduino Uno включає:

- **ON** – індикатор, що показує, що плата живлена;
- **L** – це користувацький світлодіод, приєднаний до 13-го піна. Він працює за допомогою константи **LED_BUILTIN** і загоряється при високому рівні напруги на піні, а вимикається при низькому;
- **RX, TX** – ці світлодіоди мигають під час прошивки плати та обміну

даними між Arduino Uno та комп'ютером, а також коли використовуються піни 0 та 1.

Arduino Uno є основним компонентом у цьому проекті, виконуючи функцію мікроконтролерної плати. Як центральний управляючий елемент, вона керує діяльністю робота та відповідає за його основні функції.

У цьому проекті використовується мікроконтролер ATmega328P, встановлений на платі Arduino Uno. ATmega328P містить вбудований процесор та набір програмованих цифрових і аналогових входів/виходів, що дозволяє контролювати різноманітні пристрої та сенсори.

Arduino Uno відіграє ключову роль у координації взаємодії між різними компонентами робота, такими як мотори, сервоприводи та ультразвуковий сенсор. Завдяки своїм програмованим цифровим входам/виходам, плата може управляти рухом моторів, регулювати кут обертання сервоприводів, приймати сигнали з ультразвукового сенсора та виконувати інші дії згідно з завантаженою програмою. Програмування Arduino Uno виконується на мові програмування Arduino, що дозволяє легко адаптувати та реалізувати необхідні функції робота та його взаємодію з додатковими модулями та сенсорами.

Наступний компонент – Arduino Motor Shield L293D (рисунок 5.3) має наступні характеристики.

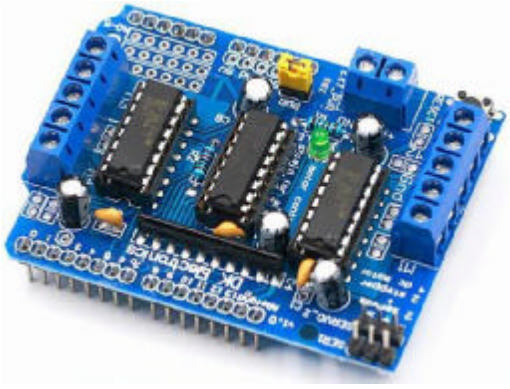

Рисунок 5.3 – Шилд драйверів двигунів на L293D

Інтегрована мікросхема L293D використовується як двоканальний драйвер мотора, що дозволяє контролювати двигуни постійного струму, з максимальним струмом до 600 мА на кожен канал.

Ця мікросхема є елементом шилда, який здатний одночасно управляти кількома двигунами постійного струму, з можливістю керування від одного до чотирьох двигунів.

Це розширює функціональні можливості робота, дозволяючи йому виконувати різноманітні задачі та маневри.

Шилд з інтегрованою мікросхемою L293D підтримує вхідну напругу живлення від 4.5 до 36 вольт, що забезпечує гнучкість у виборі джерела живлення. Логічний рівень керування включає два роз'єми для сервоприводів з напругою 5 В, які підключаються до спеціального таймера на платі Arduino для забезпечення високої точності керування.

Для моніторингу роботи шилда на ньому встановлено LED-індикатори, що відображають стан каналів. Вбудований захист від перевантаження дозволяє запобігти потенційним пошкодженням як самого шилда, так і підключених двигунів.

Крім того, шилд дозволяє керувати швидкістю та напрямком обертання двигунів за допомогою PWM (широтно-імпульсної модуляції) сигналів, забезпечуючи точне і ефективне керування.

L293D Based Arduino Motor Shield пропонує кілька значущих переваг для управління двигунами постійного струму за допомогою платформи Arduino:

- **Простота у використанні**: Шилд має зручний інтерфейс, який спрощує процес підключення та керування двигунами. Від користувачів не вимагається складні налаштування або технічні знання для початку роботи.
- **Висока сумісність з Arduino**: Цей шилд розроблений спеціально для використання з Arduino, що забезпечує легку інтеграцію у проекти із мінімальними змінами чи додатковими налаштуваннями.
- **Можливість керування кількома двигунами одночасно**: L293D

Based Arduino Motor Shield обладнаний роз'ємами для підключення до чотирьох двигунів постійного струму та двох сервоприводів. Це дозволяє одночасно контролювати багато моторів, що істотно розширює можливості управління проектом.

- **Керування швидкістю двигунів**: Шилд дозволяє налаштовувати швидкість кожного двигуна окремо через аналоговий сигнал, надаючи високий рівень контролю і точності для оптимізації роботи двигунів.

Ці характеристики роблять L293D Based Arduino Motor Shield надзвичайно ефективним та зручним рішенням для багатьох проектів, зокрема для складних робототехнічних конструкцій.

L293D Based Arduino Motor Shield забезпечує значні можливості для розширення, що робить його ідеальним для застосування у багатофункціональних робототехнічних проектах. Використовуючи цей шилд, можна додавати додаткові модулі або датчики, тим самим розширюючи його функціональність і адаптивність до різних задач. Ця гнучкість дозволяє розробникам кастомізувати та оптимізувати свої проекти, включаючи до складу системи нові компоненти в залежності від специфіки завдання. Таким чином, шилд може стати основою для розробки різних типів роботів або автоматизованих систем, надаючи міцну та водночас гнучку платформу для експериментів і інновацій.

Підсумовуючи, L293D Based Arduino Motor Shield є чудовим інструментом для розробки проектів, що вимагають керування двигунами постійного струму з використанням платформи Arduino. Цей шилд вирізняється простотою використання, високою сумісністю з Arduino, здатністю одночасно управляти декількома двигунами, підтримкою контролю швидкості, а також можливостями розширення. Ось декілька ключових елементів, що роблять цей шилд таким ефективним:

- **Драйвери двигунів L293D**: Кожен драйвер є двоканальним H-Bridge, що дозволяє керувати двома двигунами постійного струму або одним кроковим двигуном. Завдяки наявності двох таких драйверів на шилді, можливе

управління до чотирьох двигунів постійного струму або двома кроковими двигунами.

- **Регістр зсуву 74HC595**: Цей компонент розширює чотири цифрові контакти Arduino до восьми контактів керування, що забезпечує додаткові можливості управління напрямком для двох мікросхем L293D.

Ці характеристики роблять L293D Based Arduino Motor Shield вельми потужним і гнучким рішенням для управління двигунами у різноманітних робототехнічних і автоматизованих проектах, забезпечуючи точність і надійність у керуванні.

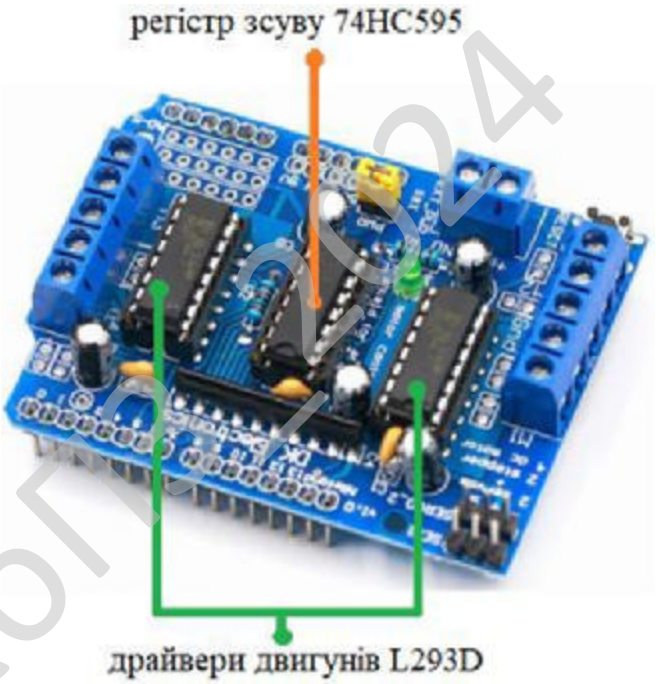

Рисунок 5.4 – Розміщення L293D драйверів двигунів і 74HC595 регістра зсуву

Підключення живлення двигунів на L293D Based Arduino Motor Shield включає гнучкі опції, що дозволяють користувачам адаптувати систему до різних потреб живлення. Шилд підтримує діапазон напруги двигунів від 4.5 до 36 вольт, що робить його сумісним з широким спектром двигунів постійного струму.

Користувачі мають можливість використовувати живлення двигунів

спільно з живленням Arduino або підключати його окремо. Це вибір між спільним і незалежним живленням реалізується за допомогою спеціальної перемички, яка розміщена біля двоконтактного роз'єму живлення. Перемичка маркована як PWR.

Як вибрати режим живлення:

- Спільне живлення з Arduino: Перемикач у положенні, що з'єднує шилд з Arduino, забезпечує живлення двигунів через ту ж саму систему, що живить Arduino. Це може бути корисним для проектів з меншими вимогами до потужності або коли використовується єдине джерело живлення для простоти.
- Окреме живлення: Якщо перемикач встановлено в інше положення, двигуни живляться незалежно від Arduino, дозволяючи підключити до шилда власне джерело живлення з вищою потужністю або стабільністю. Це корисно для проектів, які вимагають більшої потужності або коли є потреба уникнути перевантаження живлення Arduino.

Цей механізм дозволяє легко переключатися між режимами, забезпечуючи гнучкість у розробці та оптимізації енергетичного управління проекту. (рис. 3.5).

Рисунок 5.5 - Розміщення системи живлення шилда

Підключення двигунів до L293D Based Arduino Motor Shield може

відбуватися двома способами, залежно від конфігурації перемички PWR, яка впливає на методи живлення:

a. **Перемичка встановлена (спільне живлення)**:

- Коли перемичка PWR встановлена, живлення двигунів подається через гніздо постійного струму Arduino.
- В цьому режимі двигуни та Arduino не ізольовані один від одного, що означає, що обидва компоненти використовують одне і те ж джерело живлення.
- Цей метод використовується тоді, коли напруга живлення двигуна не перевищує 12 вольт, забезпечуючи простоту підключення, оскільки необхідно тільки одне джерело живлення.

b. **Перемичка видалена (окреме живлення)**:

- Видалення перемички PWR від'єднує живлення двигунів від Arduino, дозволяючи фізично ізолювати двигуни від мікроконтролера.
- У цьому випадку для двигунів необхідно забезпечити окреме джерело живлення, підключене до двополюсного роз’єму живлення, позначеного як EXT_PWR.
- Це дозволяє використовувати вищу напругу живлення для двигунів, не створюючи ризику для Arduino.

Підключення двигунів постійного струму до шилда:

- Вихідні канали для двигунів розташовані на шилді за допомогою двох 5-контактних гвинтових клем. Ці клеми забезпечують міцне та надійне з'єднання, яке легко налаштувати та підтримувати.
- Кожна клема може підключати двигун до відповідних каналів драйверів L293D, забезпечуючи необхідне управління швидкістю та напрямком обертання.

Ця гнучка система підключення дозволяє використовувати шилд в різноманітних проектах з різними вимогами до живлення та управління двигунами. (рис. 3.6), позначених на платі як M1, M2, M3 і M4. До цих клем

можна загалом підключити чотири двигуни постійного струму, що працюють від 4,5 до 25 В.

6 ОСНОВНІ ВИСНОВКИ

Мобільна робототехніка як технологічна галузь та галузь наукових досліджень останні два десятиліття зазнала неймовірних перетворень, перед усім, завдяки інформаційно-комунікаційним технологіям. Це відкрило нові можливості для стрімкої автоматизації транспортних засобів та створення SDC та мобільних роботів різного функціонального призначення, які здатні пересуватися без участі людини-водія. Водночас, проєкти у галузі робототехніки, зокрема такого масштабу як SDC, вимагають чимало ресурсів (фінансових, трудових) та практичних інструментів реалізації ідей та наукових доробок. До того ж винахідники витрачають чимало часу на створення пілотних зразків інженерних рішень і для конкретної конфігурації обладнання часто вимушені «винаходити велосипед» у процесі створення дослідного зразка. Платформа Arduino, маючи низку новацій і переваг у застосуванні (що підтверджує це дослідження), дає можливість достатньо легко створити прототип мобільних роботів заданої конфігурації. Це прискорює отримання результатів дослідницьких проєктів та здійснення їх тестування. Завдяки недорогим, зручним у користуванні та налаштуванні роботизованим рішенням, Arduino сприяє новаторству у робототехніці, відкриваючи широкі можливості для реалізації нових проєктів, зокрема і проєкту

«Система навігації самохідної техніки на базі Arduino». Цей проєкт реалізовано з використанням ультразвукового датчика для виявлення об’єктів (статичних перешкод) у навколишньому середовищі, Motor Driver Shield для керування чотирма двигунами постійного струму для руху коліс робота за допомогою мікроконтролера Arduino.

Серед факторів, які впливають на чітку навігацію сконструйованого робота, перше це середовище, в якому відбувалося тестувався, а також кількість наявних перешкод у тестовому просторі. Ці фактори впливають на датчик (у даному випадку ультразвук), що вказує на те, що точності пересування мобільного

робота залежить від якості датчика.

Запропонований підхід має окремі обмеження: продуктивність і точність створеного мобільного робота значною мірою залежать від моделі ультразвукового датчика, який застосовується, та його якості. Застосований датчик не може точно виміряти відстань між роботом і об'єктом (перешкодою), коли: відстань сягає більше 3 метрів, кут занадто малий, об'єкт занадто малий, поверхня перешкоди погано відбиває (чи зовсім не відбиває) ультразвуковий сигнал.

Для кращих результатів і підвищеної точності пересування створений мобільний робот вимагає додаткових датчиків. Крім того, використання кращих приводів забезпечить більш швидку навігацію та ефективнішу роботу.

Рекомендації для подальших досліджень полягають у наступному. Система навігації буде якіснішою, якщо до проєкту додати:

- камери – це дозволить вивести робота за межі прямої видимості (крім того, камеру можна використовувати для обробки зображень);

- бездротові технології – це забезпечить зовнішній зв'язок і керування роботом із віддаленого комп'ютера.

Ці висновки демонструють, що Arduino є потужним інструментом для розробки роботів та інших електронних пристроїв, який пропонує широкий спектр можливостей для користувачів будь-якого рівня кваліфікації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Koval V., Adamiv O., Proc. of the Third IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2005). – Sofia (Bulgaria). – 2005. – P. 120- 124.

2. Yasin, J.N., Mohamed, S.A.S., Haghbayan, MH. et al. Low-cost ultrasonic based object detection and collision avoidance method for autonomous robots. Int. j. inf. tecnol. 13, 97–107 (2021). https://doi.org/10.1007/s41870-020-00513-w.

3. Sabry F. Self Driving Car: Solving Full Self-driving Need Solving Real-world artificial Intelligence. One Billion Knowledgeable. 2022 . 293 p.

4. Shapiro D. G. Three Anecdotes from the DARPA Autonomous Land Vehicle Project. AI Magazine, 2008. 29(2), 40. https://doi.org/10.1609/aimag.v29i2.2108

5. Young R. Critical Analysis of Prototype Autonomous Vehicle Crash Rates Six Scientific Studies from 2015-2018 SAE International. 2021. 252 p

6. Рудик А. В. Наукові основи та принципи побудови приладової системи вимірювання прискорення мобільного робота : дис. ... д-ра техн. наук : 05.11.01– Прилади та методи вимірювання механічних величин. Київ, 2018. – 460 с.

7. Поліщук М. М., Ткач М.М. Робототехнічні системи: проектування і моделювання: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, 2021. 112 с.

8. Мигаль В. Д. Інтелектуальні системи в технічній експлуатації автомобілів: монографія. Х.: Майдан, 2018. 262 с.

9. Staron M. Automotive Software Architectures: An Introduction. Second Edition Springer International Publishing A&G. 2022. 274 p

10. Vargas J. et al. An Overview of Autonomous Vehicles Sensors and Their Vulnerability to Weather Conditions. Sensors. 2021; 21(16):5397.

11. Wevolver 2020 Autonomous Vehicle Technology Report. URL: https://www.coursehero.com/file/62601304/Wevolver2020AutonomousVehicechnologyReportpdf/ (accessed 2023 June 1)

12. React.js [Електронний ресурс] – Режим доступу до ресурсу: https://https://uk.reactjs.org/

13. AngularJS MVC [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm

14. Vue.js [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/

15. A Look Inside ADAS Modules. URL: https://amkor.com/semiconductor-story/alook.

16. AV and ADAS Sensors. URL https://community.sw.siemens.com/s/article/AVand- ADAS-Sensors (accessed 2023 June 1)

17. Гуржій А. М. Основи автоматики та робототехніки: Навчальний посібник/ А.М. Гуржій, А. Т. Нельга, В. М. Співак, О. С. Ітякін:–Дніпро:«Гарант СВ», 2021.- 243с.

18. Lim, B.S. et al. Autonomous Vehicle Ultrasonic Sensor Vulnerability and Impact Assessment. In Proceedings of the IEEE World Forum on Internet of Things (WFIoT), Singapore, 5–8 February 2018; pp. 231–236.

19. Саліхов М. М. Arduino – перспективний інструмент протитопування у робототехніці // Концептуальні шляхи розвитку науки та освіти: матеріали VIII Міжнародної науково-практичної конференції м. Львів, 9-10 червня 2023 року.– Львів: Львівський науковий форум, 2023. – с. 73-79.

20. Bräunl T. Embedded Robotics: From Mobile Robots to Autonomous Vehicles with RaspberryPi and Arduino Springer Nature, 2022. 519 p.

21. Навігаційні системи [Електронний ресурс] : навч. посіб. для студ. Спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / С.Л. Лакоза; КПІ ім. Ігоря Сікорського, 2021. — 80 с.

22. Imad M. et al. Navigation system for autonomous vehicle: A survey. JCSTS., vol. 2, no. 2, pp. 20–35, 2020.

23. Ковалець І. В. та ін. Технологія планування траєкторій руху мобільних об’єктів з урахуванням перешкод на складній місцевості. Енергетика і автоматика. - 2017. - № 1. - С. 110-122.

24. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web/HTML

25. Aguilo I. et al. Artificial Intelligence Research and Development Hardcover by IOS Press. 2003. 500 p.

26. Гребенюк Б. А. Розробка підсистеми управлення інтелектуальним роботом / Б. А. Гребенюк // «Automation and Development of Electronic Devices» ADED-2023: Collection of Students' Scientific Paper. – Kharkiv : Kind of Kharkiv National Universitety of Radio Elektronics [electronic edition], 2023. – Part 1. –336p. P. 263-269

27. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MongoDB

28. Becker M. et al Obstacle avoidance procedure for mobile robots. ABCM Symposium series in Mechatronics. Vol. 2, 2006. P. 250-257.

29. CORS, XSS [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/maleta/cors-xss-and-csrf-with-examples-in-10-minutes-35k3

30. AJAX [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AJAX

31. Fetch API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/API/Fetch_API

32. Звенігородський О.С. Інтелектуальна система планування тактики руху автономного робота в квазістаціонарному середовищі: Дис... канд. техн. наук: 05.13.23. — Д., 2002. — 127с.

33. Адамів, О.П. Моделі та інтелектуальні засоби адаптивного керування автономним мобільним роботом. Дис. канд.техн. наук. Одеса, 2007. 124 с.

34. Koval V., Adamiv O., Proc. of the Third IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2005). – Sofia (Bulgaria). – 2005. – P. 120- 124.

35. npm [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/

36. Install MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://docs.mongodb.com/manual/administration/install-on-linux/

37. Как установить Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/node-js-ubuntu-18-04-ru

38. Linux [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Linux

39. npm-audit [Електронний ресурс] – Режим доступу до ресурсу: https://docs.npmjs.com/cli/audit

40. Kojima T. et al. Intelligent Technology for More Advanced Autonomous Driving. 2018. Hitachi Review Vol. 67, No. 1. p. 58–63

41. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQL

42. MongoDB Compass [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/products/compass

43. Postman [Електронний ресурс] – Режим доступу до ресурсу: https://www.postman.com/

44. Саліхов М. М. Від автоматизованих до автономних транспортних засобів // Наукові досягнення та інновації: шлях до успіху. X Всеукраїнська мультидисциплінарна науково-практична Інтернет-конференція, 31 травня 2023, Україна, Київ : зб. матеріалів — Електрон. дан. — Київ : Ярочé нко Я. В., 2023. — с. 136-142.

45. SPA (Single-page application) [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/SPA

55. Саліхов М.М. Самокеровані автомобілі та системи їх навігації// Практичні та теоретичні питання розвитку науки та освіти: матеріали VIII Міжнародної науково-практичної конференції: м. Львів, 19-20 червня 2023 року. – Львів: Львівський науковий форум, 2023. – С.53-59.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку програмного забезпечення системи кібербезпеки навігації техніки на базі Arduino.

Підставою для розробки служить завдання на випускну кваліфікаційну роботу, видане на кафедрі програмування та захисту інформації (нак. №135-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної бакалаврської роботи є розробка та програмна реалізація системи керування роботом з впровадженням системи кібербезпеки.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської дипломної роботи є відносна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

- розробку додатку;
- систему підключення та тестування баз даних ;
- цілісність даних у процесі роботи та при зберіганні;
- простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10/11.

5.8.1 Обладнання

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

5.8.2 Мова програмування

При розробці ПЗ потрібно використовувати наступні технології та мови програмування: C/C++.

Середовище програмування – Arduino/

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

– Структурна схема системи керування – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок схема алгоритму програми – 1 аркуш.
– Блок-схема підпрограми – 1 аркуш.
– Пояснювальна записка – 79 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи.

Постановка задачі на виконання кваліфікаційної роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної роботи.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання кваліфікаційної роботи на попередній захист 18.05.2024 р.

9.2 Подання кваліфікаційної роботи на захист 05.06.2024 р.

Додаток Б
(обов'язковий)

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник кваліфікаційної бакалаврської роботи
_____ Босько В.В

Програмне забезпечення системи кібербезпеки навігації техніки на базі Arduino

Лістинг програми

Код документу 12
Носій: CD/DVD-диск

Загальна кількість аркушів: 21

Літера: РП

Кропивницький – 2024 року

AF_DCMotor RBMotor(1); //Створюються об'єкти для керування кожним окремим двигуном

AF_DCMotor RFMotor(2); AF_DCMotor LFMotor(3); AF_DCMotor LBMotor(4);

Servo servoWatch; //Створюється об'єкт для керування сервоприводом

Крок 3: Встановлення параметрів системи. Цей блок коду визначає парамтри, які використовуються для налаштування системи руху робота.

Параметри ультразвукового датчика:

- byte TRIG_PIN = 2; - призначення піна для вихідного сигналу (тригер) ультразвукового датчика.
- byte ECHO_PIN = 13; - призначення піна для вхідного сигналу (exo) ультразвукового датчика.
- byte distWatch = 150; - встановлення максимальної відстані, на яку датчик реагує. Об'єкти, що знаходяться далі, ігноруються.
- byte rangeStop = 50; - встановлення мінімальної відстані до об'єкту, при якій рух зупиняється (у сантиметрах).
- float timeEcho = 2*(distWatch+10)/100/340*1000000; - обчислення максимального часу очікування зворотного сигналу. Цей час використовується для визначення, коли датчик вважає, що немає перешкоди.

Параметри двигуна наступні:

- byte MAX_SPEED = 100; - встановлення максимальної швидкості двигуна.
- int OFFSET_SPEED = 10; - коефіцієнт, який враховує різницю у потужності між двигунами по різних сторонах.

Параметри повороту:

- TURN_SPEED = 50; - сума, яка додається до швидкості двигуна під час повороту. Це дозволяє роботу повертатиї.

Код програми:

byte TRIG_PIN = 2; //Призначення пінів ультразвукового датчика

byte ECHO_PIN = 13;

byte distWatch = 150; //Максимальна відстань, на яку датчик реагує(об’єкти що знаходяться далі ігноруються)

byte rangeStop = 50; //Мінімальна відстань до об'єкту, при якій рух зупиняється (в сантиметрах)

float timeEcho = 2*(distWatch+10)/100/340*1000000; //Максимальний час очікування зворотного сигналу

byte MAX_SPEED = 100; //Максимальна швидкість двигуна

int OFFSET_SPEED = 10; //Коефіцієнт для врахування того, що одна сторона є більш потужною

int TURN_SPEED = 50; //Сума, яка додається до швидкості двигуна під час повороту

Крок 4: Налаштування і початкова ініціалізація.

У цьому блоку коду виконуються деякі початкові налаштування і ініціалізація компонентів системи. Цей код виконується один раз при запуску програми (у функції setup()).

Налаштування швидкостей двигунів: rightBack.setSpeed(motorSpeed); - встановлює швидкість для правого заднього двигуна.

rightFront.setSpeed(motorSpeed); - встановлює швидкість для правого переднього двигуна.

leftFront.setSpeed(motorSpeed+motorOffset); - встановлює швидкість для лівого переднього двигуна з додаванням коефіцієнта motorOffset.

leftBack.setSpeed(motorSpeed+motorOffset); - встановлює швидкість для лівого заднього двигуна з додаванням коефіцієнта motorOffset.

Зупинка всіх моторів:

RBMotor.run(RELEASE); - забезпечує зупинку правого заднього мотора.

RFMotor.run(RELEASE); - забезпечує зупинку правого переднього мотора.

LFMotor.run(RELEASE); - забезпечує зупинку лівого переднього мотора.

LBMotor.run(RELEASE); - забезпечує зупинку лівого заднього мотора.

Ініціалізація сервоприводу:

servoWatch.write(90); - призначає пін 10 для сервоприводу, що дозволяє керувати його положенням.

Налаштування пінів ультразвукового датчика: pinMode(TRIG_PIN,OUTPUT); - встановлює режим вихідного піна trig у режим "OUTPUT" (вихідний).

pinMode(ECHO_PIN,INPUT); - встановлює режим вхідного піна echo у режим "INPUT" (вхідний).

Крок 5: Основний цикл програми.

У цьому кроці я описую функцію loop(), яка є основним циклом програми.

Цей код виконується безперервно після ініціалізації (у функції setup()).

У циклі loop() зазвичай розміщується основний алгоритм керування роботом.

Залежно від контексту програми, цей алгоритм може включати рух, взаємодію з датчиками, прийом команд зовнішнього контролера тощо.

Блок коду 1: Перевірка передньої відстані

servoWatch.write(90); //Встановлення сервоприводу на погляд прямо вперед

delay(750);

int distance = getRange(); //Перевірка, що попереду немає предметів if(distance >= rangeStop) //Якщо в межах шляху немає предметів - рух вперед

```
{
movementForward();
}
Блок коду 2: Зупинка руху
while(distance >= rangeStop)                                    //Продовження
перевірки відстані до об'єкту, поки вона не стане менше мінімальної відстані
зупинки
{
distance = getRange(); delay(250);
}
stopMove();                                                     //Зупинка
двигунів Блок коду 3: Визначення напрямку повороту
int directionTurn = checkDir();                                 //Перевірка
відстані до об’єктів зліва і справа та отримання інструкції щодо повороту
Serial.print(directionTurn);
switch (directionTurn)                                          //Поворот
ліворуч, праворуч або здійснити
розворот залежно від інструкції
{
case 0:                                                         //Поворот ліворуч
turnLeft (400);
break;
case 1:                                                         //Розворот
turnLeft (700); break;
case 2:                                                         //Поворот
праворуч turnRight (400);
break;
}
```

- Функція accelerate(): Прискорення двигунів від 0 до повної швидкості.
- Функція decelerate(): Уповільнення двигунів від повної швидкості до нуля.
- Функція movementForward(): Встановлення всіх двигунів на рух вперед.
- Функція stopMove(): Встановлення всіх двигунів на зупинку.
- Функція turnLeft(): Встановлення двигунів на поворот ліворуч на вказаний час і зупинка їх після повороту.
- Функція turnRight(): Встановлення двигунів на поворот праворуч на вказаний час і зупинка їх після повороту.
- Функція getRange(): Вимірювання відстані до перешкоди за допомогою ультразвукового датчика.

- Функція checkDir(): Перевірка відстаней до об'єктів зліва і справа, прийняття рішення щодо напрямку повороту.

Блок коду 4: Функція accelerate()

```
void accelerate()                                    //Функція для прискорення двигунів від 0 до повної швидкості
{
for (int i=0; i<MAX_SPEED; i++)                      //Цикл від 0 до повної швидкості
{
RBMotor.setSpeed(i);                                 //Встановлення двигунів на поточну швидкість циклу
RFMotor.setSpeed(i); LFMotor.setSpeed(i+OFFSET_SPEED); LBMotor.setSpeed(i+OFFSET_SPEED); delay(10);
}
}
```

Блок коду 5: Функція decelerate()

```
void decelerate()                                    //Функція уповільнення двигунів від повної швидкості до нуля
{
for (int i=MAX_SPEED; i!=0; i--)                     //Цикл від повної швидкості до 0
{
RBMotor.setSpeed(i);                                 //Встановлення двигунів на поточну швидкість циклу
RFMotor.setSpeed(i); LFMotor.setSpeed(i+OFFSET_SPEED); LBMotor.setSpeed(i+OFFSET_SPEED); delay(10);
}
}
```

Блок коду 6: Функція movementForward()

```
void movementForward()                               //Встановлення всіх двигунів на рух вперед
{
RBMotor.run(FORWARD); RFMotor.run(FORWARD); LFMotor.run(FORWARD);
LBMotor.run(FORWARD);
}
```

Блок коду 7: Функція stopMove()

```
void stopMove()                                      //Встановлення всіх двигунів на зупинку
{
```

```
      RBMotor.run(RELEASE); RFMotor.run(RELEASE); LFMotor.run(RELEASE);
LBMotor.run(RELEASE);
      }
```

Блок коду 8: Функція turnLeft()

```
      void turnLeft(int duration)                                    //Встановлення
двигунів на поворот ліворуч на вказаний час і зупинити їх
      {
      RBMotor.setSpeed(MAX_SPEED+TURN_SPEED);                        //Встановлення
всіх двигунів на задану швидкість
      RFMotor.setSpeed(MAX_SPEED+TURN_SPEED);
LFMotor.setSpeed(MAX_SPEED+OFFSET_SPEED+TURN_SPEED);
LBMotor.setSpeed(MAX_SPEED+OFFSET_SPEED+TURN_SPEED);
      RBMotor.run(FORWARD); RFMotor.run(FORWARD); LFMotor.run(BACKWARD);
LBMotor.run(BACKWARD); delay(duration);
      RBMotor.setSpeed(MAX_SPEED);                                   //Встановлення
всіх двигунів на задану швидкість
      RFMotor.setSpeed(MAX_SPEED); LFMotor.setSpeed(MAX_SPEED+OFFSET_SPEED);
      LBMotor.setSpeed(MAX_SPEED+OFFSET_SPEED); RBMotor.run(RELEASE);
      RFMotor.run(RELEASE); LFMotor.run(RELEASE); LBMotor.run(RELEASE);
      }
```

Блок коду 9: Функція turnRight()

```
      void turnRight(int duration)                                   //Встановлення
двигунів на поворот праворуч на вказаний час і зупинити їх
      {
      RBMotor.setSpeed(MAX_SPEED+TURN_SPEED);                        //Встановлення
всіх двигунів на задану швидкість
      RFMotor.setSpeed(MAX_SPEED+TURN_SPEED);
LFMotor.setSpeed(MAX_SPEED+OFFSET_SPEED+TURN_SPEED);
LBMotor.setSpeed(MAX_SPEED+OFFSET_SPEED+TURN_SPEED);
      RBMotor.run(BACKWARD); RFMotor.run(BACKWARD); LFMotor.run(FORWARD);
LBMotor.run(FORWARD); delay(duration);
      RBMotor.setSpeed(MAX_SPEED);                                   //Встановлення
всіх двигунів на задану швидкість
      RFMotor.setSpeed(MAX_SPEED); LFMotor.setSpeed(MAX_SPEED+OFFSET_SPEED);
LBMotor.setSpeed(MAX_SPEED+OFFSET_SPEED); RBMotor.run(RELEASE);
      RFMotor.run(RELEASE); LFMotor.run(RELEASE);
      LBMotor.run(RELEASE);
      }
```

Блок коду 10: Функція getRange()

Вимірює відстань до об'єкта за допомогою ультразвукового сенсора.

```
      int getRange()                                          //Вимір відстані
до предмета
      {
      unsigned long pulseDuration;                            //Створення
змінної для зберігання часу шляху імпульсу
      int distance;                                           //Створення
змінної для зберігання обчисленої відстані
      digitalWrite(TRIG_PIN, HIGH);                           //Генерація
імпульсу тривалістю 10 мікросекунд
      delayMicroseconds(10); digitalWrite(TRIG_PIN, LOW);
      pulseDuration = pulseIn(ECHO_PIN, HIGH, timeEcho);      //Вимір часу
повернення імпульсу
      distance = (float)pulseDuration * 340 / 2 / 10000;      //Обчислення
відстані до об’єкта на основі часу імпульсу
      return distance;
      }
```

Блок коду 11: Функція checkDir ()

Перевіряє відстані в лівому та правому напрямках, щоб визначити, у якому напрямку повернути робота.

```
      int checkDir()                                          //Перевірка
напрямку ліворуч і праворуч, прийняти рішення, у який бік повернути
      {
      int rangeArray [2] = {0,0};                             //Відстань
ліворуч і праворуч
      int directionTurn = 1;                                  //Напрямок
повороту, 0 ліворуч, 1
      розворот, 2 праворуч
      servoWatch.write(180);                                  //Поворот серво,
щоб подивитися ліворуч
      delay(500);
      rangeArray [0] = getRange();                            //Отримати
відстань до об’єкта зліва
      servoWatch.write(0);                                    //Поворот серво,
щоб подивитися праворуч
      delay(1000);
      rangeArray [1] = getRange();                            //Отримати
відстань до об’єкта справа
      if (rangeArray[0]>=200 && rangeArray[1]>=200)           //Якщо обидва
напрямки вільні - поворот ліворуч
      directionTurn = 0;
      else if (rangeArray[0]<=rangeStop && rangeArray[1]<=rangeStop)   //Якщо
обидва напрямки заблоковані, розворот
      directionTurn = 1;
```

```
      else if (rangeArray[0]>=rangeArray[1])                    //Якщо ліворуч
більше місця, поворот ліворуч
      directionTurn = 0;
      else if (rangeArray[0]<rangeArray[1])                     //Якщо праворуч
більше місця, поворот праворуч
      directionTurn = 2; return directionTurn;
      }

      var express = require('express'); var router = express.Router();
      var passport = require('passport'); var User = require('../models/user');
      var Project = require('../models/project'); var csrf = require('csurf');
      var csrfProtection = csrf();
      var config_passport = require('../config/passport.js'); var moment =
require('moment');
      var Leave = require('../models/leave');
      var Attendance = require('../models/attendance');


      router.use('/', isLoggedIn, function isAuthenticated(req, res, next)
      {
      next();
      });
      /**
      *                                                          Description:
      *                                                          Displays home
page to the admin
      *
      *
      *                                                          Known Bugs: None
      */
      router.get('/', function viewHome(req, res, next) {
res.render('Admin/adminHome', {
      title: 'Admin Home', csrfToken: req.csrfToken(),
      userName: req.session.user.name
      });
      });
      /**
      *                                                          Description:
      *                                                          First it gets
attributes of the logged in admin from the User Schema.
      *                                                          Attributes are
get with the help of id of logged in admin stored in session.
      */
      router.get('/view-profile', function viewProfile(req, res, next) {
User.findById(req.session.user._id, function getUser(err, user)
      {
```

```
      if (err) {
      console.log(err);

      }
      res.render('Admin/viewProfile', { title: 'Profile',
      csrfToken: req.csrfToken(), employee: user,
      moment: moment,
      userName: req.session.user.name
      });
      });

      });
      /**
      *                                                    Description:
      *                                                    Sorts the list of
employees in User Schema.

      *                                                    Known Bugs: None
      */

      router.get('/view-all-employees', function viewAllEmployees(req, res,
next) {
      var userChunks = []; var chunkSize = 3;
      //find is asynchronous function
      User.find({$or: [{type: 'employee'}, {type: 'project_manager'},
      {type: 'accounts_manager'}]}).sort({_id: -1}).exec(function getUsers(err,
docs) {
      for (var i = 0; i < docs.length; i++) { userChunks.push(docs[i]);
      }
      res.render('Admin/viewAllEmployee', { title: 'All Employees', csrfToken:
req.csrfToken(), users: userChunks,
      userName: req.session.user.name
      });
      });


      });
      /**
      *                                                    Description:
      *                                                    Displays add
employee form to the admin.
      */
      router.get('/add-employee', function addEmployee(req, res, next) { var
messages = req.flash('error');
      var newUser = new User(); res.render('Admin/addEmployee', {
```

```
});


});

title: 'Add Employee', csrfToken: req.csrfToken(), user:
config_passport.User, messages: messages,
hasErrors: messages.length > 0, userName: req.session.user.name

/**
*                                                    Description:
*                                                    First it gets the
id of the given employee from the parameters.
*/
router.get('/all-employee-projects/:id', function
getAllEmployeePojects(req, res, next) {
var employeeId = req.params.id; var projectChunks = [];

//find is asynchronous function Project.find({employeeID:
employeeId}).sort({_id: -
1}).exec(function findProjectOfEmployee(err, docs) { var hasProject = 0;
if (docs.length > 0) { hasProject = 1;
}
for (var i = 0; i < docs.length; i++) { projectChunks.push(docs[i]);
}
User.findById(employeeId, function getUser(err, user) { if (err) {
console.log(err);
}
res.render('Admin/employeeAllProjects', { title: 'List Of Employee
Projects', hasProject: hasProject,
projects: projectChunks, csrfToken: req.csrfToken(), user: user,
userName: req.session.user.name




});
/**
```

```
*/



});

});

});

router.get('/leave-applications', function getLeaveApplications(req, res,
next) {

var leaveChunks = [];

var employeeChunks = []; var temp;
//find is asynchronous function
Leave.find({}).sort({_id: -1}).exec(function findAllLeaves(err, docs) {
var hasLeave = 0;
if (docs.length > 0) { hasLeave = 1;
}
for (var i = 0; i < docs.length; i++) { leaveChunks.push(docs[i])
}
for (var i = 0; i < leaveChunks.length; i++) {
User.findById(leaveChunks[i].applicantID, function getUser(err, user) {
if (err) {
console.log(err);
}
employeeChunks.push(user);
})
}



seconds

// call the rest of the code and have it execute after 3
setTimeout(render_view, 900); function render_view() {
res.render('Admin/allApplications', { title: 'List Of Leave Applications',
csrfToken: req.csrfToken(), hasLeave: hasLeave,
leaves: leaveChunks,
employees: employeeChunks, moment: moment, userName:

req.session.user.name
});
```

```
});

}
});


/**
*                                                          Description:
*/
router.get('/respond-application/:leave_id/:employee_id', function
respondApplication(req, res, next) {
var leaveID = req.params.leave_id;
var employeeID = req.params.employee_id; Leave.findById(leaveID, function
getLeave(err, leave) {
if (err) {
console.log(err);
}

User.findById(employeeID, function getUser(err, user) { if (err) {
console.log(err);
}
res.render('Admin/applicationResponse', { title: 'Respond Leave
Application', csrfToken: req.csrfToken(),
leave: leave, employee: user,
moment: moment, userName: req.session.user.name
});


})


});


});

/**
*                                                          Description:
*/
router.get('/employee-profile/:id', function getEmployeeProfile(req, res,
next) {
var employeeId = req.params.id; User.findById(employeeId, function
getUser(err, user) {
```

```
      if (err) {
      console.log(err);
      }
      res.render('Admin/employeeProfile', { title: 'Employee Profile', employee:
user,
      csrfToken: req.csrfToken(), moment: moment,
      userName: req.session.user.name
      });

      });
      });

      /**
      *                                                          Description:
      */
      router.get('/edit-employee/:id', function editEmployee(req, res, next) {
      var employeeId = req.params.id; User.findById(employeeId, function
getUser(err, user) {
      if (err) {

      res.redirect('/admin/');
      }
      res.render('Admin/editEmployee', { title: 'Edit Employee', csrfToken:
req.csrfToken(), employee: user,
      moment: moment, message: '',
      userName: req.session.user.name
      });



      });
      });

      /**
      *                                                          Description:

      *
      *                                                          Known Bugs: None
      */
      router.get('/edit-employee-project/:id', function editEmployeeProject(req,
res, next) {
      var projectId = req.params.id;
      Project.findById(projectId, function getProject(err, project) { if (err) {
      console.log(err);
      }
      res.render('Admin/editProject', { title: 'Edit Employee', csrfToken:
req.csrfToken(), project: project,
```

```
      moment: moment, message: '',
      userName: req.session.user.name
      });


      });

      });
      /**
      *                                                          Description:
      *                                                          Gets the id of
the employee from parameters.
      *                                                          Displays the add
employee project form to the admin.
      *
      *                                                          Known Bugs: None
      */
      router.get('/add-employee-project/:id', function addEmployeeProject(req,
res, next) {

      var employeeId = req.params.id; User.findById(employeeId, function
getUser(err, user) {
      if (err) {
      res.redirect('/admin/');
      }
      res.render('Admin/addProject', { title: 'Add Employee Project', csrfToken:
req.csrfToken(), employee: user,
      moment: moment, message: '',
      userName: req.session.user.name
      });
      });

      });


      /**
      *                                                          Description:
      *                                                          First finds
project in the Project Schema with the help of id from the parameters.
      *                                                          Gets the Employee
of the project.
      *
      *                                                          Known Bugs: None
      */
      router.get('/employee-project-info/:id', function
viewEmployeeProjectInfo(req, res, next) {
```

```
var projectId = req.params.id;
Project.findById(projectId, function getProject(err, project) { if (err) {
console.log(err);


user) {

}
User.findById(project.employeeID, function getUser(err,
if (err) {
console.log(err);

}
res.render('Admin/projectInfo', {
title: 'Employee Project Information', project: project,
employee: user, moment: moment, message: '',
userName: req.session.user.name, csrfToken: req.csrfToken()
});
})
});

});

/**
*                                                    Description:
*                                                    Redirects admin
to the employee profile page.
*                                                    Known Bugs: None
*/
router.get('/redirect-employee-profile', function viewEmployeeProfile(req,
res, next) {
var employeeId = req.user.id;
User.findById(employeeId, function getUser(err, user) { if (err) {
console.log(err);
}
res.redirect('/admin/employee-profile/' + employeeId);
});

});
/**
*                                                    Description:
*                                                    Displays the
admin its own attendance sheet
*                                                    Known Bugs: None
*/
router.post('/view-attendance', function viewAttendance(req, res, next) {
```

```
      var attendanceChunks = []; Attendance.find({
      employeeID: req.session.user._id, month: req.body.month,
      year: req.body.year
      }).sort({_id: -1}).exec(function viewAttendanceSheet(err, docs)
      {
      var found = 0;
      if (docs.length > 0) { found = 1;
      }
      for (var i = 0; i < docs.length; i++) { attendanceChunks.push(docs[i]);
      }
      res.render('Admin/viewAttendanceSheet', { title: 'Attendance Sheet',
      month: req.body.month, csrfToken: req.csrfToken(), found: found,
      attendance: attendanceChunks, userName: req.session.user.name, moment:
moment
      });
      });

      });
      /**
      *                                                    Description:
      *                                                    Known Bugs: None
      */
      router.get('/view-attendance-current', function
viewCurrentlyMarkedAttendance(req, res, next) {
      var attendanceChunks = [];
      Attendance.find({
      employeeID: req.session.user._id, month: new Date().getMonth() + 1, year:
new Date().getFullYear()
      }).sort({_id: -1}).exec(function getAttendanceSheet(err, docs) { var found
= 0;
      if (docs.length > 0) { found = 1;
      }
      for (var i = 0; i < docs.length; i++) { attendanceChunks.push(docs[i]);
      }
      res.render('Admin/viewAttendanceSheet', { title: 'Attendance Sheet',
      month: new Date().getMonth() + 1, csrfToken: req.csrfToken(), found:
found,
      attendance: attendanceChunks, moment: moment,
      userName: req.session.user.name
      });
      });

      });
      /**
      *                                                    Description:
```

```
      *                                                    Displays the
attendance sheet of the given employee to the admin.
      */
      router.get('/view-employee-attendance/:id', function
viewEmployeeAttendance(req, res, next) {
      var attendanceChunks = [];
      Attendance.find({employeeID: req.params.id}).sort({_id: -
1}).exec(function getAttendanceSheet(err, docs) {
      var found = 0;
      if (docs.length > 0) { found = 1;
      }
      for (var i = 0; i < docs.length; i++) { attendanceChunks.push(docs[i]);
      }

      User.findById(req.params.id, function getUser(err, user) {
      res.render('Admin/employeeAttendanceSheet', { title: 'Employee Attendance
Sheet', month: req.body.month,
      csrfToken: req.csrfToken(), found: found,
      attendance: attendanceChunks, moment: moment,
      userName: req.session.user.name
      ,
      'employee_name': user.name
      })
      });
      });



      });
      /**
      *                                                    Description:

      */
      router.post('/add-employee', passport.authenticate('local.add- employee',
{
      successRedirect: '/admin/redirect-employee-profile', failureRedirect:
'/admin/add-employee', failureFlash: true,
      }));
      /**
      *                                                    Description:

      */
      router.post('/respond-application', function respondApplication(req, res)
{
      Leave.findById(req.body.leave_id, function getLeave(err, leave)
      {
```

```
      leave.adminResponse = req.body.status; leave.save(function saveLeave(err)
{
      if (err) {
      console.log(err);
      }
      res.redirect('/admin/leave-applications');
      })
      })

      });
      router.post('/edit-employee/:id', function editEmployee(req, res) { var
employeeId = req.params.id;

      var newUser = new User(); newUser.email = req.body.email;
      if (req.body.designation == "Accounts Manager") { newUser.type =
"accounts_manager";
      }
      else if (req.body.designation == "Project Manager") { newUser.type =
"project_manager";
      }
      else {
      newUser.type = "employee";
      }
      newUser.name = req.body.name,
      newUser.dateOfBirth = new Date(req.body.DOB), newUser.contactNumber =
req.body.number, newUser.department = req.body.department;
      newUser.Skills = req.body['skills[]']; newUser.designation =
req.body.designation;

      User.findById(employeeId, function getUser(err, user) { if (err) {
      res.redirect('/admin/');
      }
      if (user.email != req.body.email) {
      User.findOne({'email': req.body.email}, function getUser(err, user) {
      if (err) {
      res.redirect('/admin/');
      }
      if (user) {
      res.render('Admin/editEmployee', { title: 'Edit Employee', csrfToken:
req.csrfToken(), employee: newUser,
      moment: moment,
      message: 'Email is already in use', userName: req.session.user.name
      });
      }
      });
      }
```

```
      user.email = req.body.email;
      if (req.body.designation == "Accounts Manager") { user.type =
"accounts_manager";
      }
      else if (req.body.designation == "Project Manager") { user.type =
"project_manager";
      }
      else {
      user.type = "employee";
      }
      user.name = req.body.name,
      user.dateOfBirth = new Date(req.body.DOB), user.contactNumber =
req.body.number,

      user.department = req.body.department; user.Skills = req.body['skills[]'];
user.designation = req.body.designation;

      user.save(function saveUser(err) { if (err) {
      console.log(error);
      }
      res.redirect('/admin/employee-profile/' + employeeId);

      });
      });

      });
      router.post('/add-employee-project/:id', function addEmployeeProject(req,
res) {
      var newProject = new Project(); newProject.employeeID = req.params.id;
newProject.title = req.body.title; newProject.type = req.body.type;
      newProject.startDate = new Date(req.body.start_date), newProject.endDate =
new Date(req.body.end_date), newProject.description = req.body.description,
newProject.status = req.body.status;
      newProject.save(function saveProject(err) { if (err) {
      console.log(err);
      }
      res.redirect('/admin/employee-project-info/' + newProject._id);
      });
      });
      router.post('/edit-employee-project/:id', function
editEmployeeProject(req, res) {
      var projectId = req.params.id; var newProject = new Project();
      Project.findById(projectId, function (err, project) { if (err) {
      console.log(err);
      }
      project.title = req.body.title; project.type = req.body.type;
```

```
      project.startDate = new Date(req.body.start_date), project.endDate = new
Date(req.body.end_date), project.description = req.body.description,
project.status = req.body.status;
      project.save(function saveProject(err) { if (err) {
      console.log(err);



      projectId);
      });

      }
      res.redirect('/admin/employee-project-info/' +

      });
      });


      router.post('/delete-employee/:id', function deleteEmployee(req, res) {
      var id = req.params.id;
      User.findByIdAndRemove({_id: id}, function deleteUser(err) { if (err) {
      console.log('unable to delete employee');
      }
      else {
      res.redirect('/admin/view-all-employees');



      });

      }
      });

      router.post('/mark-attendance', function markAttendance(req, res, next) {

      Attendance.find({
      employeeID: req.session.user._id, date: new Date().getDate(), month: new
Date().getMonth() + 1, year: new Date().getFullYear()
      }, function getAttendance(err, docs) { var found = 0;
      if (docs.length > 0) { found = 1;
      }
      else {
      var newAttendance = new Attendance(); newAttendance.employeeID =
req.session.user._id; newAttendance.year = new Date().getFullYear();
newAttendance.month = new Date().getMonth() + 1; newAttendance.date = new
```

```
Date().getDate(); newAttendance.present = 1; newAttendance.save(function
saveAttendance(err) {
      if (err) {
      console.log(err);
      }
      });
      }
      res.redirect('/admin/view-attendance-current');
      });

      });
      module.exports = router;
      function isLoggedIn(req, res, next) { if (req.isAuthenticated()) {
      return next();
      }
      res.redirect('/');
      }


      function notLoggedIn(req, res, next) { if (!req.isAuthenticated()) {
      return next();
      }
      res.redirect('/');
      }
```