**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

ННК “Інститут прикладного системного аналізу”
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

“___”_____2016 р.

# Дипломна робота

___першого (бакалаврського)_____**рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності7.05010102, 8.05010102 Інформаційні технології проектування
_____7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: Мобільна навігаційна система з застосуванням маркерів_____
_____
_____

Виконав (-ла): студент (-ка) IV курсу, групи ___ДА-21___
(шифр групи)

_____Круш Ігор Володимирович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____к.т.н., доц. Харченко К. В._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економічний ___д.е.н, проф. Семенченко Н.В._____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____ст. викладач Бритов О.А._____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2016 року

**Національний технічний університет України**
**«Київський політехнічний інститут»**

Факультет (інститут) ННК “Інститут прикладного системного аналізу”
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський) (перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.050102, 8.050102 Інформаційні технології проектування
7.050103, 8.050103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

«___»_____2016 р.

**ЗАВДАННЯ на дипломний проект (роботу) студенту**
Крушу Ігорю Володимировичу
(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) ) Мобільна навігаційна система з застосуванням маркерів

керівник проекту (роботи) ) Харченко К.В., к.т.н., доц.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 2016 р. №_____

2. Строк подання студентом проекту (роботи) 10.06.2016

3. Вихідні дані до проекту (роботи)
Мобільна навігаційна система на базі Raspberry PI 2B з WiFi-модулем, 2 двигуни постійного струму 3V-6B, сервопривід Futaba S3003, Raspberry PI камера, пластмасовий корпус. Бібліотеки RPi GPIO, серверний веб фреймворк Java Spark, OpenCV. Алгоритми розпізнавання та орієнтації: динамічні фільтри Калмана, гістрограмні фільтри, часткові фільтри.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Провести огляд можливих складових компонентів системи
2. Розробити фізичний прототип рухомої платформи
3. Розробити алгоритм вирішення задачі розпізнавання
4. Розробити алгоритм вирішення задачі локалізації
5. Провести функціонально-вартісний аналіз отриманого програмного продукту
6. Проаналізувати результати роботи програми, зробити висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Зовнішній вигляд побудованого прототипу– плакат.
2. Архітектура системи – плакат.
3. Схематична ілюстрація алгоритму локалізації з використанням згортки функцій - плакат.

6. Консультанти розділів проекту (роботи)

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 01.02.2016

Календарний план

| № з/п | Назва етапів виконання дипломного проекту (роботи) | Строк виконання етапів проекту (роботи) | Примітка |
|---|---|---|---|
| 1 | Отримання завдання | 01.02.2016 | |
| 2 | Ознайомлення з технічною літературою і підготовка теоретичної частини роботи | 15.02.2016 | |
| 3 | Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі. | 28.02.2016 | |
| 4 | Проектування модулів пристрою, розробка моделей. | 10.03.2016 | |
| 5 | Тестування розроблених моделей модулів. Перевірка відповідності завданню. | 15.03.2016 | |
| 6 | Захист дипломної роботи | 30.04.2016 | |

| | | | |
|---|---|---|---|
| 7 | Оформлення дипломної роботи | 31.05.2016 | |
| 8 | Отримання допуску до захисту та подача роботи в ДЕК | 10.06.2016 | |

Студент _____ І. В. Круш
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ К. В. Харченко
(підпис) (ініціали, прізвище)

# АНОТАЦІЯ


бакалаврської дипломної роботи Круш Ігоря Володимировича на тему:
“Мобільна навігаційна система з застосуванням маркерів”

Дана дипломна робота присвячена розробці мобільної навігаційної системи, що здатна керуватись сама на основі даних, що приходять із сенсорів.

Приведений огляд існуючих методів отримання даних про стан навколишнього середовища. Було створено фізичну систему на основі мікрокомп’ютера Raspberry PI з General Purpose Input/Output для керування такими периферійними пристроями, як керуюче колесо, двигуни постійного струму, камера, інфрачервоний датчик обходу перешкод, ультразвуковий датчик відстані, проаналізовано можливості обробки даних сенсорів за допомогою типових алгоритмів, виявлено їх переваги та недоліки, реалізовано алгоритм локалізації системи у просторі на наперед відомій карті маркерів та перешкод.

Дану роботу рекомендовано використовувати у якості посібника для розробки мобільних самонавігаційних платформ, що орієнтують за допомогою маркерів.

Загальний обсяг роботи: 104 сторінки, 1 додаток на 24 сторінки, 46 рисунків, 9 таблиць, 24 посилання.

Ключові слова: мобільна навігаційна система, самокеровані системи, комп’ютерне бачення, локалізація, візуальні маркери, QR-коди.

# АННОТАЦИЯ

бакалаврской дипломной работы Круша Игоря Владимировича на тему:
“Мобильная навигационная платформа с использованием маркеров”

Данная дипломная работа посвящена разработке мобильной навигационной системы, которая способна к самоуправлению на основе данных, приходящих с сенсоров.

Проведен обзор существующих методов получения данных о состоянии окружающей среды. Было создано физическую систему на основе микрокомпьютера Raspberry PI с General Purpose Input / Output для управления такими периферийными устройствами, как рулевое колесо, двигатели постоянного тока, камера, инфракрасный датчик обхода препятствий, ультразвуковой датчик расстояния, проанализированы возможности обработки данных сенсоров с помощью типовых алгоритмов, исследованы их преимущества и недостатки, реализирован алгоритм локализации системы в пространстве в наперед известной карте маркеров и помех.

Данную работу рекомендуется использовать в качестве пособия для разработки мобильных самонавигацийних платформ, ориентирующихся с помощью маркеров.

Общий объем работы 104 страницы, из них основная часть - 62 страницы, 1 приложение на 24 страницы, 46 рисунков, 24 библиографических наименований.

Ключевые слова: мобильная навигационная система, самоуправляемые системы, компьютерное видение, локализация, визуальные маркеры, QR-коды.

# ANNOTATION

for the bachelor thesis of Krush Ihor Volodymyrovych on “Mobile navigation system using markers”

Thesis is devoted to the development of mobile navigational system with self-driving ability based on data from sensors.

There was the research of existing methods of receiving data about surrounding. Physical system was created based on microcomputer Raspberry Pi with General Purpose Imput/Output for control of wheels, motors, camera, infrared sensors, hypersonic sensor. The methods of handling the data and their procs and cons from sensors were analyzed. Localization algorithm was implemented for the system to localize in the space with known map of markers and obstacles.

The results of research are encouraged to use as teaching materials for developing of navigational self-driving systems.

Total volume of work: 104 pages, 1 appendix for 24 pages, 46 figures, 9 tables, 24 links.

Keywords: mobile navigational system, self-driving systems, computer vision, localization, visual markers, QR-codes.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КБ – комп’ютерне бачення

CV – Computer Vision

ВВЗП – Входи/виходи загального призначення

GPIO – General Purpose Input/Output

# ВСТУП

Ідея побудови автоматизованих роботів далеко не нова, але саме останнє десятиліття принесло нові технології та методи, що дозволяють побудувати справді потужні та надійні системи. Вони зможуть допомагати людству виконувати будь-які види робіт – від найпростішої рутини до складних процесів, небезпечних для життя людей.

Будь-яка автоматизована робототехніка запрограмована за допомогою алгоритмів, здатних до аналізу та самонавчання. Яскравими прикладами таких пристроїв є проекти, що будують самокеровані автомобілі, розумні будинки.

Найбільш успішним прикладом є проект Google Self-Driving Car Project. Компанія Google працює над автомобілем, що буде здатен довезти вас із пункту А у пункт Б без будь-якої вашої допомоги у керуванні та виборі маршруту. В основу цього пристрою покладено використання SLAM – метод одночасної локалізації і побудови карти, що має на увазі автоматичну побудову карти у невідомому просторі або для оновлення карти в заздалегідь відомому просторі с одночасним контролем поточного місцезнаходження та пройденого шляху.

Локалізація включає одне питання: Де робот зараз? "Тут" є відносною величиною відносно певного орієнтиру (зазвичай пункту відправлення або призначення). Незважаючи на те, що запитання просте, відповісти на нього не так легко, так як відповідь відрізняється в залежності від характеристик вашого робота. Методи локалізації, які працюють відмінно для одного робота в одному середовищі не може добре працювати або взагалі в іншому середовищі. Наприклад, локалізацій, які добре працюють на відкритому повітрі в середовищі може виявитися марним в приміщенні.

Всі методи локалізації зазвичай забезпечують дві основні частини інформації:

1) поточне місце розташування робота в деякому середовищі;
2) яка поточна орієнтація робота в тому ж середовищі.

Тобто для того, щоб мати уявлення, де знаходиться робот, потрібно знайти відповідь на ці запитання програмноє

Мета цієї роботи – дослідити способи побудови рухомих платформ, а також вирішити задачу локалізації за допомогою заданих маркерів різних типів.

Для досягнення поставленої мети необхідно:

- побудувати прототип;
- дослідити можливе використання алгоритмів розпізнавання маркерів різного типу;
- дослідити алгоритми та програмні засоби визначення положення системи;
- побудувати систему передачі поточного відео с рухомої системи.

# 1 ОГЛЯД СКЛАДОВИХ КОМПОНЕНТІВ

Для тестування та оцінки роботи системи в реальних умовах потрібно побудувати її прототип. У цьому розділі буде детально розглянути компоненти прототипу та можливості їх використання.

## 1.1 Raspberry Pi

Raspberry Pi є невеликим комп'ютером, який містить мінімально необхідну кількість компонентів, які забезпечують його функціонування. Розроблений Raspberry Pi Foundation, благодійною організацією Великобританії, з метою надання недорогих комп'ютерів і безкоштовного програмного забезпечення для студентів. Їх кінцева мета полягає у сприянні комп'ютерної природничо-наукової освіти, і вони сподіваються, що цей маленький доступний комп'ютер буде інструментом, який дозволить проводити різноманітні наукові експерименти.

Друкована плата (PBC) містить на собі роз’єми введення та виведення і всі апаратні засоби. В даний час Фонд продає голу PCB - корпус не входить у комплект Raspberry Pi. Загалом, Raspberry PI - навчальне видання з документацію і попередньо завантаженим освітнім програмним забезпечення. Що ж стосується програмного забезпечення, в даний час є три Linux на основі операційних систем, підтримуваних Raspberry Pi.

У першій партії продукції було виготовлено 10,000 пристроїй, і всі вони продані протягом декількох годи. Модель B є версією з ціною у $35 з найбільшою кількістю варіантів підключення, і є моделлю, що цікавить більшість ентузіастів. Модель A  - без мережі Ethernet і має один порт USB, коштує $25.

Що стосується специфікацій, то Raspberry Pi є пристроєм, розміром 82x54x20 мм, побудованим на Broadcom BCM2835 (системі на кристалі). Система включає в себе 32-бітний ARM1176JZFS процесор, що працює на

частоті 700 МГц, а також графічний процесор VideoCore IV. Вона має 256 МБ оперативної пам'яті в пакеті POP. Raspberry Pi отримує енергію від зарядного пристрою Micro USB 5V AC або принаймні 4-х батарейок типу AA.

У той час як ARM процесор забезпечує реальну продуктивність, аналогічну з 300MHz Pentium 2, Broadcom GPU є потужним графічним ядром, здатним до апаратного декодування декількох форматів відео високої чіткості. Однак для того, щоб зберегти кошти на виробництві Raspberry Pi, британська благодійна організація має лише ліцензію на кодек H.264 для апаратного декодування (і неясно, чи зможуть користувачі придбати/активувати додаткові кодеки). У зв'язку з цим VideoCore IV GPU є досить потужним, так як система здатна до апаратного декодування H.264 1080p30 з бітовими швидкостями до 40Mb / s.

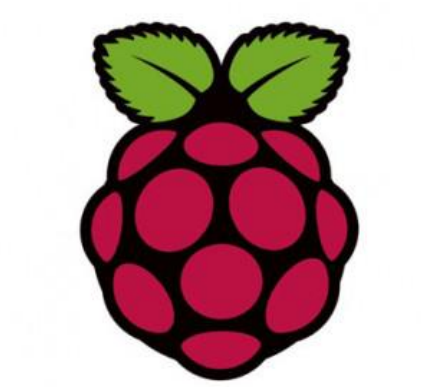

Рисунок 1.1 - Офіційний логотип Raspberry Pi [1]

В даній роботі була використана модель B - вона оснащена HDMI входом і композитним відеовиходом, має два порти USB 2.0, порт 10/100 Ethernet, слот для SD-карти, GPIO (General Purpose I / O) роз'єм і аналоговий аудіо вихід (3,5 мм роз'єм для навушників). Дешевша Модель A не має порту Ethernet і один лише один USB вхід, але має такі ж конфігурації процессора.

Основною перевагою Raspberry Pi є те, що цей пристрій є вдалою комбінацією невеликого розміру комп'ютера і за доступної ціни. Ентузіасти

часто використовують Rapsberry в якості дешевого ПК для домашнього кінотеатру або неосновного робочого комп’ютера з низьким рівнем енергоспоживанням. Невеликий розмір робить дозволяє легко приховати комп'ютер, що може бути встановлений позаду дисплея за потреби. Він також може бути використаний в нішевих додатках, таких як цифрові вивіски або як мікрокомп’ютер для визначення проблем на основній робочій машині.

Raspberry Pi не тільки маленький пристрій в своєму роді - два інших відомих приклади в ентузіастів співтовариства є Arduino і BeagleBoard. Хоча системи аналогічних дуже схожі за призначенням, Raspberry Pi має суттєві відмінності від цих систем. З апаратної точки зору, Raspberry Pi базується навколо ARM системи, у якої дуже закритий вихідний код. З іншого боку, системи Arduino і Beagleboard засновані на апаратних засобах з повністю відкритим вихідним кодом. У BeagleBoards дійсно використовують процесори ARM (TI OMAP 3530 SoC). Плати Arduino є ще більш несхожі в зв'язку з використанням 8-бітних і 16-розрядний Atmel мікроконтролер чіпів.

Найбільша різниця між чимось на зразок Arduino і Raspberry Pi в цілях використання. Arduino призначений для використання в якості плати розвитку з мікро-контролерів, які будуть запрограмовані, а потім інтегровані в більші машини або електроніку і будуть запускатись самостійно. Raspberry Pi ж призначений для використання в якості кінцевого продукту і для роботи в якості традиційного настільного комп'ютера (насправді, дистриб'ютори відмовилися продавати Raspberry Pi, поки він не отримав CE / FCC EM сертифікації перешкод). Слід визнати, що BeagleBoards намагаються захопити частину ринку Raspberry Pi з  такими проектами, як BeagleBoard Ubuntu і підтримку XBMC, але мають перешкоди у вигляди високої ціни [2].

Рисунок 1.2 - Raspberry Pi модель B

Однією з особливістей Raspberry Pi є ряд GPIO (загального призначення вводу / виводу) - шпильки уздовж краю дошки, поруч з жовтим відео входом.

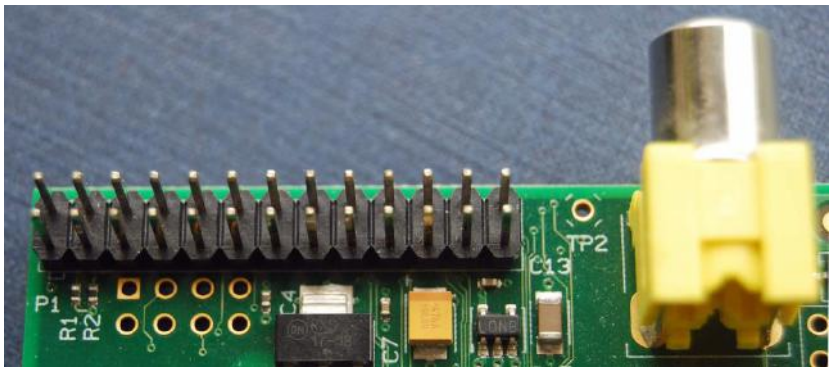


Рисунок 1.3 GPIO на Raspberry PI B

Ці контакти є фізичним інтерфейсом між Pi і зовнішнім світом. На найпростішому рівні, вони можуть розглядатись як перемикачі, що можна увімкнути або вимкнути або які сама Raspberry Pi може увімкнути або вимкнути . Сімнадцять з 26 контактів GPIO є контактами введення/виведення; інші контакти - електричні або контакти заземлення.

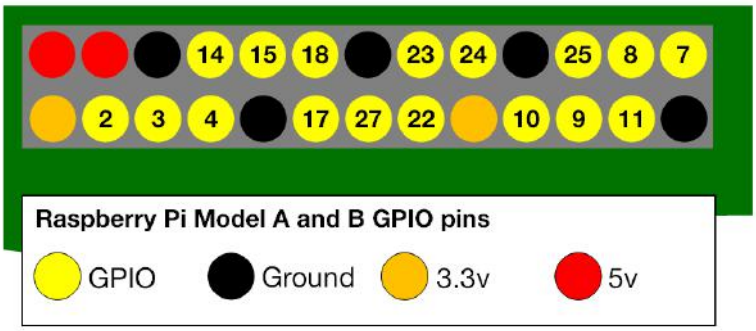


Рисунок 1.4 - Схема GPIO з логічною нумерацією котактів [3]

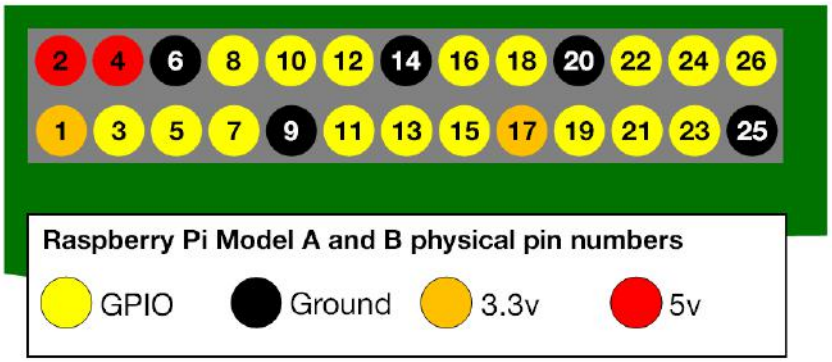


Рисунок 1.5 - Схема GPIO з фізичною нумерацією контактів [4]

GPIO під'єднане до обчислювальної системи процессора BCM2835 трьома контактами. Кожен з 3-х контактів має свій власний VDD вхідний контакт. На Raspberry Pi всі контакти GPIO живляться від 3.3В. Підключення GPIO до напруги вище, ніж 3.3V, може зруйнувати блок GPIO.

Найпростіше електричне коло, яке складене за допомогою GPIO, можна зобразити наступним чином.

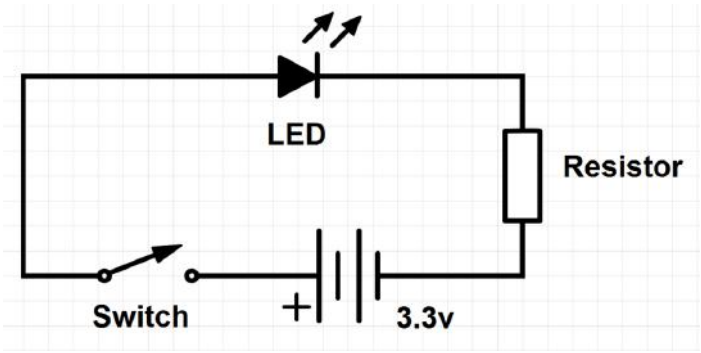


Рисунок 1.6 - Електричне коло з перемикачем та світлодіодом [5]

Коли ми використовуємо GPIO контакти в якості вихідного сигналу, то Raspberry Pi на цій схемі замінює перемикач і батареї живлення. Кожен контакт можна увімкнути або вимкнути, тобто встановити високий або низький сигнал. Коли на контакті високий сигнал - виводить 3,3 вольт; коли на контакті низький сигнал - він вимкнений.

## 1.2 WiFi-модуль

Для того, щоб мати можливість віддалено керувати Rasbperry Pi, скористаємось підключенням до локальної мережі за допомогою WiFi адаптера. Wi-Fi — торгова марка Wi-Fi Alliance та загальновживана назва для стандарту IEEE 802.11 передачі цифрових потоків даних по радіоканалах. Обладнання, що відповідає стандарту IEEE 802.11, може бути протестовано Wi-Fi Alliance та отримати відповідний сертифікат і право нанесення логотипу Wi-Fi. Поширеним на сьогодні є протокол IEEE 802.11n [6].

Встановлення Wireless LAN доцільне для побудови мереж, де розгортання кабельної системи є неможливим або економічно недоцільним. Поточні реалізації Wi-Fi дозволяють отримати швидкість передачі даних понад 100 Мбіт/с, при цьому користувачі можуть переміщуватися між точками доступу на території покриття мережі Wi-Fi, використовуючи мобільні пристрої (КПК, смартфони, PSP і ноутбуки), оснащені клієнтськими приймально-передавальними пристроями Wi-Fi та отримувати доступ в Інтернет.

У даній роботі був використаний адаптер Mediatek MT7610U. Це один з найбільш якісних WiFi модулів зі зйомною антеною, який підтримується всіма операційними системами на базі Linux або Windows. Може працювати в режимах IEEE 802.11b, IEEE 802.11g и IEEE 802.11n.


Рисунок 1.7 - WiFi-модуль Mediatek MT7610U

Для запуску даного драйверу на Raspberry Pi необхідно оновити вбудоване програмне забезпечення та завантажити і встановити файл драйвера:

```
rpi-update
sudo wget
"https://miniboard.com.ua/index.php?controller=attachment&id_attac
hment=25" -O /lib/firmware/mt7601u.bin
sudo reboot
```

Для підключення до мережі через командний рядок було написано утиліти для пошуку всіх доступних мереж та швидкого підключення до потрібної з введенням паролю (додаток А).

## 1.3 Камера

Основним пристроєм, який буде використовуватись для локалізації робота у просторі, буде камера. Зображення будуть оброблятись та на основні цих даних робот-автомобіль буде робити припущення, де саме він знаходиться.

У даній роботі була використана 5-мегапіксельна камера RPI Camera D на базі сенсора OV5647. Камера сумісна з усіма версіями Raspberry. Вона прикріплюється шлейфом, який іде в комплекті, до роз’єму CSI.

Рисунок 1.8 - 5-мегапіксельна RPI Camera D

Камера невелика у розмірах: 25 x 24 x 9 мм. Розширення фото, отриманих з камера - 2592 x 1944 пікселів. Підтримувані формати відео - 1080p з швидкістю 30 кадрів за секунду(fps), 720p з 60 fps і 480p з 90 fps. Фокусна відстань дорівнює 3.37 мм.

Для того, щоб Raspberry Pi розпізнавав камеру, потрібно увімкнути її підтримку на рівні *raspi-config.* Для цього у командному рядку треба ввести

```
$ sudo raspi-config,
```

обрати пункт “Enable Camera”(“Зробити камеру доступною”) і обрати “Enable”(“Зробити доступною”) та натиснути “Enter”.

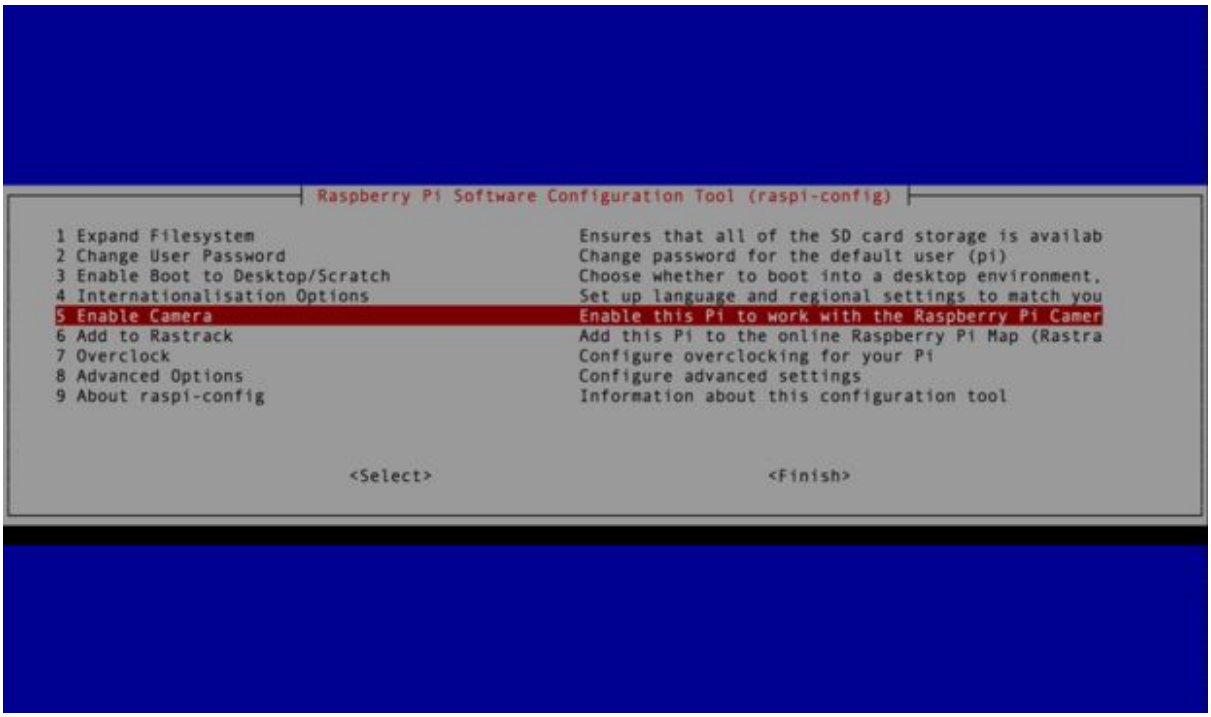


Рисунок 1.9 - Меню raspi-config

Для отримання зображення з камери у роботі була використана системна утиліта Raspberry Pi **raspistill**, загорнута у Bash-скрипт:

```
#!/bin/bash
DATE=$(date +"%Y-%m-%d_%H%M")
raspistill -vf -hf -o /home/pi/camera/$DATE.jpg
```

Для отримання зображення у алгоритмі використана обгортка на мові програмування Java(додаток 2) [7].

## 1.4 Сенсори

У даній роботі досліджувалось використання сенсори двох типів: ультразвукових далекомірів та інфрачервоних датчиків.

Ультразвуковий далекомір генерує звукові імпульси на частоті 40 кГц і слухає відгук. За часом поширення звукової хвилі туди і назад можна однозначно визначити відстань до об'єкта. Цей далекомір може служити датчиком для робота, завдяки якому він зможе визначати відстані до об'єктів або об'їжджати перешкоди. Його можна також використовувати в якості датчика для сигналізації, що спрацьовує при наближенні об'єктів.

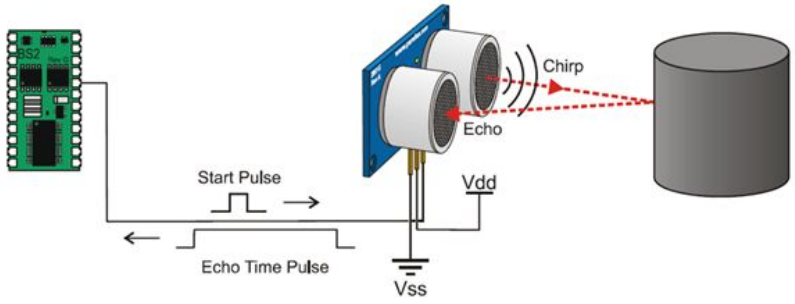


Рисунок 1.10 - Принцип роботи ехолокаторів [8]

Основною перевагою використання ехолокаторів у мобільних платформах є їх простота, дешевизна та ефективність роботи. Якщо для визначення відстані до об’єкту або виявленні перешкод нам довелося б використати кілька камер та застосувати значні алгоритмічні складності, то ехолокатори дозволяють зробити це дуже просто.

Недоліком є бути невеликий радіус дії.

У даній роботі був використаний ультразвуковий датчик відстані HC-SR04. Він підлкючається до системи Raspberry PI через GPIO і відправляє туди

дані про те, що сигнал відправлений або отриманий. Програма була написана на мові програмування Java(додаток 3) [9].

Рисунок 1.11 - Ультразвуковий датчик відстані HC-SR04

Характеристики HC-SR04:

- Напруга живлення: 5 В
- Споживання в режимі тиші: 2 мА
- Споживання при роботі: 15 мА
- Діапазон відстаней: 2-400 см
- Ефективний кут спостереження: 15 °

Інфрачервоний датчик обходу перешкоди необхідний для визначення перешкод на невеликих відстанях. Використовується в нескладних проектах на мікроконтролерах для пересувних роботів (для обходу роботом перешкод) або в системах безпеки (для виявлення в контрольованій зоні руху сторонніх об'єктів), визначення наявності предмета, контроль відстані і багато іншого. На корпусі є два світлодіода.

Зелений світлодіод горить, коли на датчик підключений до джерела живлення. Червоний загоряється, коли в зону дії випромінювача потрапляє

об'єкт, який відображає інфрачервоне випромінювання. Поріг спрацьовування датчика регулюється потенціометром.

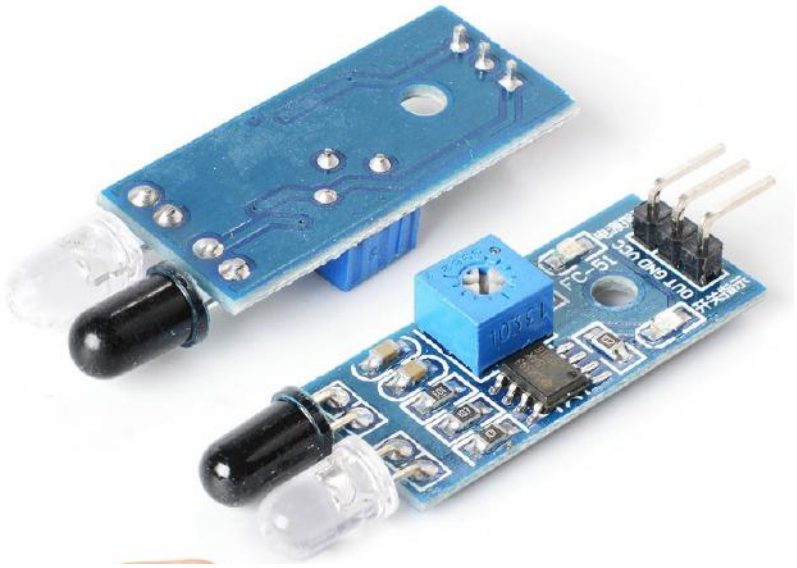

Рисунок 1.12 - Інфрачервоний датчик обходу перешкоди Flying Fish – MH Sensor

Характеристики:

- напруга живлення: 3,3-5В
- споживаний струм: 20 мА
- ефективне вимірювана відстань: 2-30см
- ефективний кут огляду: 35˚
- габарити: 31х15мм

## 1.5 Двигуни

Робот рухає колесами за допомогою трьох двигунів.

Задній привід побудований на двох двигунах постійного струму з робочою напругою від 3 до 6 Вольт.

Рисунок 1.13 - Двигун постійного струму

Швидкість обертання двигуна залежить від величини напруги, що ми подаємо на вхід. Сам двигуни має лише два входи - VCC та GND. Тобто, автоматично, коли ми подаємо напругу, двигун буде працювати. У даній роботі керування двигунами має здійснюватись програмно з Raspberry PI, тож як перемикач напруги використовується модуль L9110 H-bridge [10].


Рисунок 1.14 - Модуль L9110 H-bridge

Модуль дуже простий у використанні - він підтримує одночасно два двигуни для підключення і може керувати ними одночасно. На входи ми подаємо виходи двигунів, на виходи - 4 керуючі сигнали для двигунів, які підуть на I/O контакти GPIO, напруга на модуль подається зі стороннього джерела струму, так як потужності Raspberry Pi не вистачає для живлення двигунів [11].

Передній привід відповідає за повороти робота. У якості двигуна використаний серво Futaba S3003 з аналоговою модуляцією(період пульсу - 30 мс, довжина пульсу - 500-3000 мкс). Має з'єднання типу J. Максимальний кут повороту - $180^{o}$ . Працює у двох режимах - 4.8В та 6В. У першому режимі здатен повернутись на $60^{o}$ за 0.23с, у другому - за 0.19с.

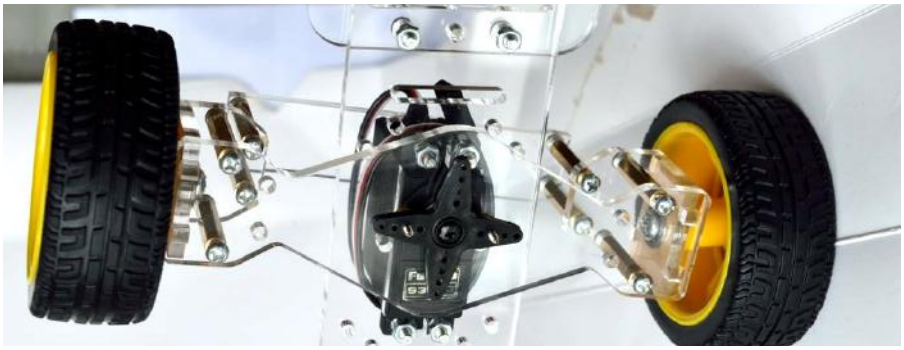

Рисунок 1.15 - Передній привід з використанням крокового двигуна Futaba S3003


Рисунок 1.16 - Серво Futaba S3003

## 1.6 Шасі

Шасі виготовлене китайської компанію Sinoning. Воно складається з пластикових компонентів, що з’єднуються за допомогою болтів, та 4 коліс, виготовлених з гуми та пластику.

Рисунок 1.17 - Загальний вигляд шасі у розібраному вигляді

Всі деталі друкуються на 3Д принтері, макети є у відкритому доступі. Кріплення надійні і тому шасі підходить для збору навігаційної системи та її тестування.

## 1.7 Проміжні компоненти

Для з’єдання усіх електричних компонентів використовується набір провідників-”джемперів” типу “мама-мама” і “тато-тато”. Всі пристрої, що не мають таких входів, спаюються з одним з провідників типу “тато-тато”.

Для проектування використовується макетна плата MB-102 для моделювання з використанням Raspberry Pi, Ardruino без використання

паяльника. Вона виконана з якісного пластику, має 400 отворів і дві лінії живлення(по 50 з кожного боку).

Рисунок 1.18 - Провідники типу “тато-тато”

Рисунок 1.19 - Провідники типу “мама-мама”

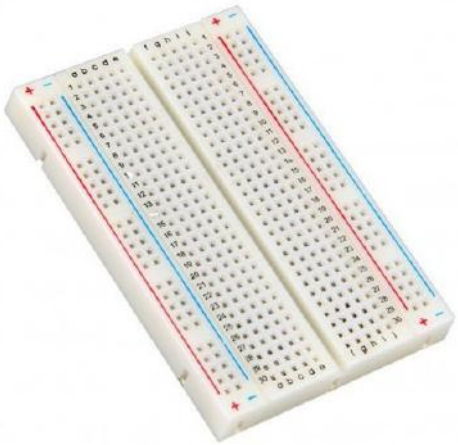

Рисунок 1.20 - Макетна плата MB-102

В якості джерела струму використовується два пристрої. Для живлення серво та двигунів використовується касета для 4 батарейок типу AA, що дає на виході загальну напругу 6В.


Рисунок 1.21 - Касета для 4 батарейок типу AA

Для живлення Raspberry Pi використовується зовнішній акумулятор Defender ExtraLife 5200, що дає на виході напругу 5В і підключається до Raspberry Pi через вхід microUSB.


Рисунок 1.22 - Defender ExtraLife 5200

## 1.8 Висновки

У цьому розділі було розглянуто компоненти, з який можна зібрати рухому платформу та їх основні властивості. Вибір саме цих компонентів дозволяє виконати швидку, надійну та дешеву збірку прототипу для тестування.

# 2 ПОБУДОВА РУХОМОЇ ПЛАТФОРМИ

У попередньому розділі було детально розглянуто компоненти, з яких можна скласти рухому платформу. У цьому розділі буде розглянуто сам процес побудови, починаючи від під’єднань джерел струму закінчуючи з’єднання мікроконтроллерів з GPIO.

Цей розділ детальніше розгляне усі компоненти з точки зору їх взаємодії з навколишнім світом та іншими компонентами системи.

## 2.1 Під’єднання шасі до контролерів та джерела живлення

Для початку для запуску системи нам потрібно зчепити між собою усі частини шасі та підключити до нього електричне обладнання та двигуни.

Шасі виготовлене з пластику і з’єднується болтами та гайками. Використаємо для цієї операції звичайну хрестоподібну викрутку.

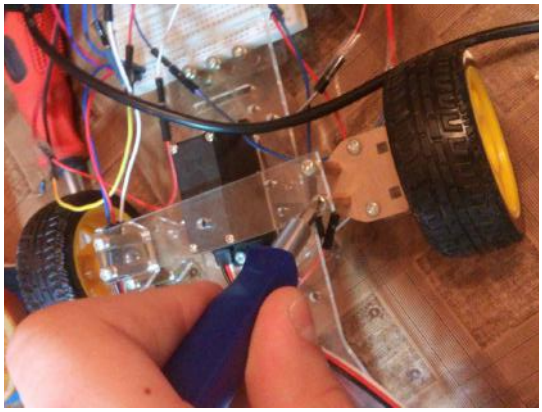

Рисунок 2.1 - З’єздання пластикових частин системи за допомогою болтів та гайок за допомогою викрутки

В результаті, отримаємо конструкцію, зображену на рис. 2.2.

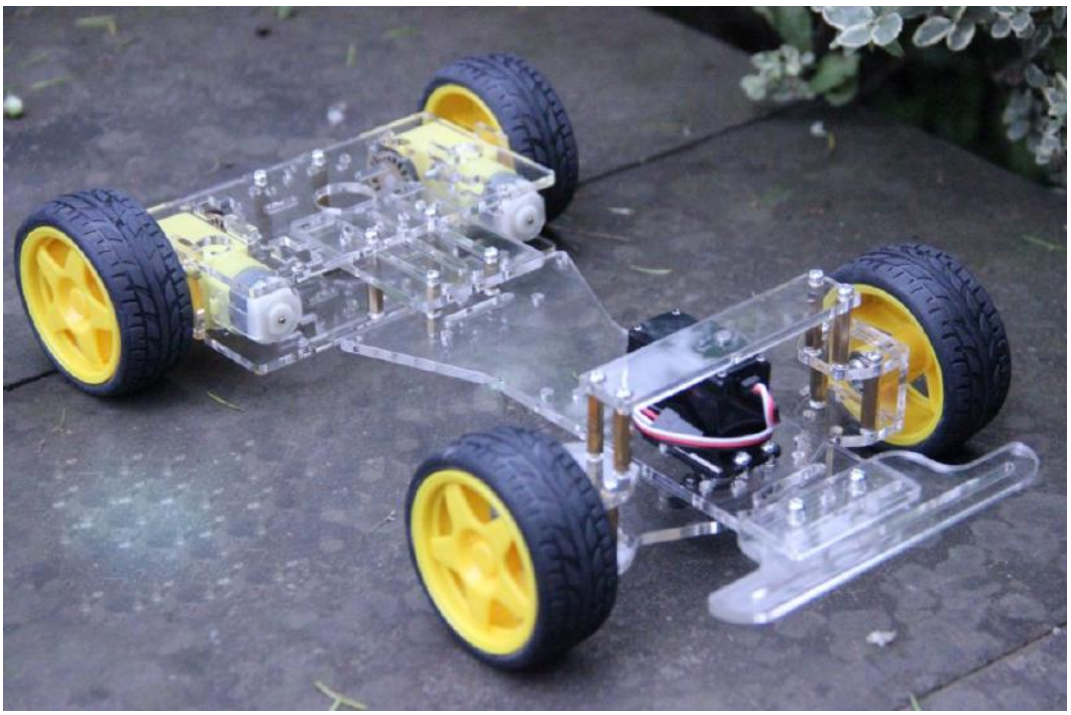

Рисунок 2.2 - Зібрана конструкція

У передній частині конструкції розміщене серво Futaba S3003, яке буде відповідати за повороти системи. У задній - два двигуни, що під’єднані до коліс - це буде слугувати приводом платформи. Всі частини надійно з’єднані болтами.

У середині системи розмістимо макетну плату - через неї буде з’єднано джерело струму з двигунами та серво, а також мікроконтроллер H-Bridge L9110, що буде регулювати подачу струму на двигуни. Саме джерело струму буде розміщено на кормі конструкції.

В задній частині системи розміщено мікрокомп’ютер Raspberry Pi, який є її мозковим центром.

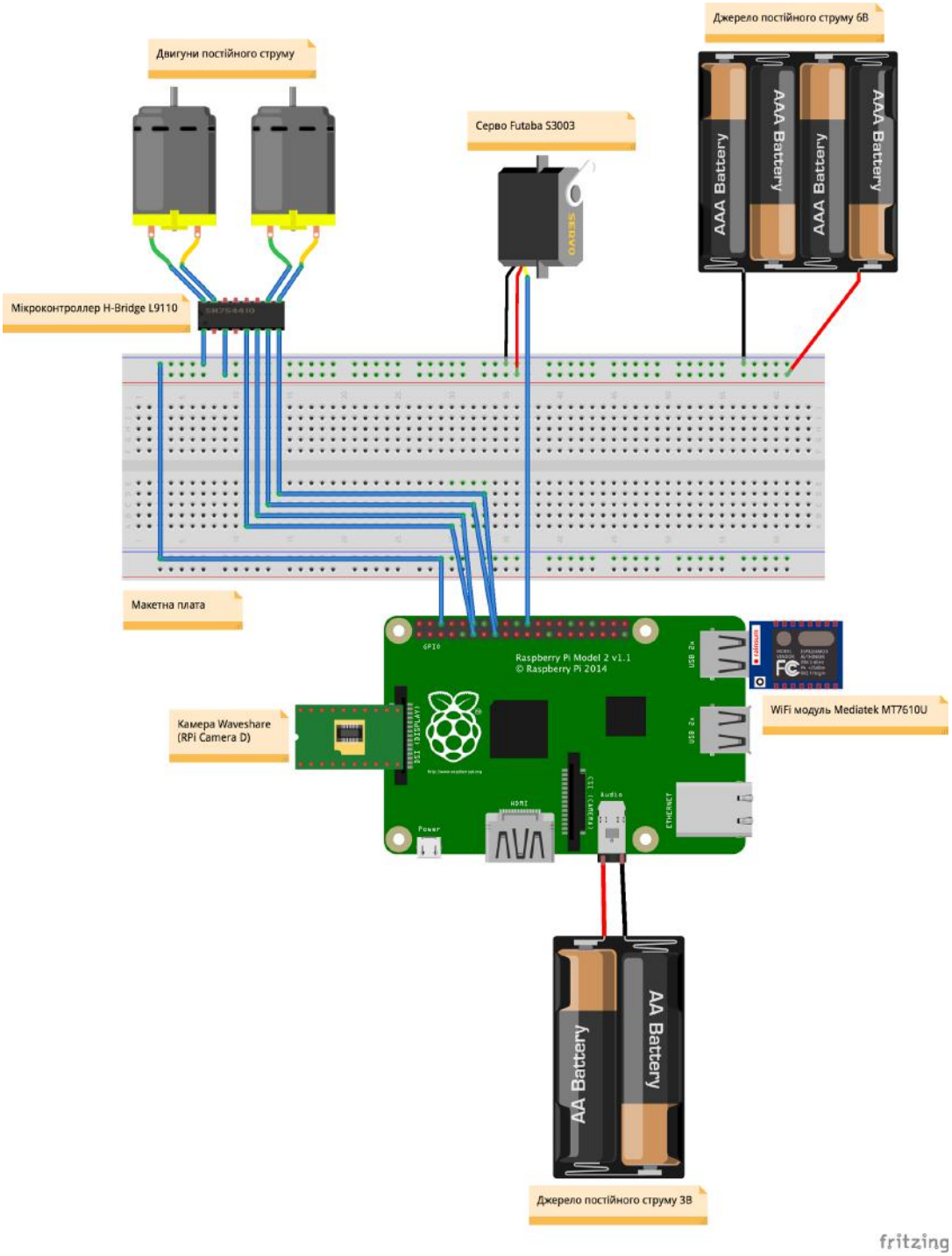


Рисунок 2.3 - Схематичне зображення отриманого електричного кола

Вона також підключається до макетної плати для того, щоб мати спільну “землю” з двигунами та серво, а також відправляє в GPIO на вихід дані з інформацією про те, чи варто вмикати двигуни та серво. З двигунами взаємодія проводиться через мікроконтроллер H-bridge L9110, серво має вбудований контролер.

Розглянемо схематичне зображення отриманого електричного кола. На ньому зображено, як саме з’єднані всі електричні компоненти системи за допомогою проводів-“джемперів”.


Рисунок 2.4 - Кінцевий вигляд системи

## 2.2 Налаштування клієнт-серверу

У цілях відлагодження системи було написано клієнт-серверне ПО для її керування. Воно представляє собою сервер для отримання команд та клієнт у

вигляді браузерного JavaScript додатку, яке реагує на натискання клавіш “вперед”, “назад”, “вліво” та “вправо” та відправляє ці команди серверу на виконання. Сервер знаходиться на Raspberry Pi, клієнт динамічно завантажується у браузер із сервера. Тобто для відлагодження має бути налаштований HTTP-зв’язок у локальній або глобальній мережі.

У якості серверного рішення було використано фреймворк на мові Java: Spark - “міні-фреймворк для створення веб додатків на Java 8 з докладенням мінімальних зусиль”, як стверджують самі розробники. [12]


Рисунок 2.5 - Офіційний логотип Java Spark [13]

Так як сервер керує напряму Raspberry Pi, тобто віддає команди на запуск двигунів та серво, він має бути запущеним в режимі доступу *sudo*, адже тільки в цьому режимі додаток має право доступу до GPIO.

Для керування GPIO на Raspberry Pi використовується бібліотека Pi4J. Цей проект покликаний забезпечити дружній об'єктно-орієнтована API введення / виведення і реалізації бібліотек для надання Java програмістам повного доступу до можливостей введення / виведення платформи Raspberry Pi. Цей проект абстрагує інтеграцію  низького рівня для того, щоб Java-програміст міг зосередитися на реалізації бізнес-логіки програми.


Рисунок 2.6 - Логотип The Pi4j Project [14]

На стороні сервера нам потрібно мати дві точки доступу: одна для віддачі коду клієнту, друга - для отримання запитів на керування зі сторони клієнта. Перша буде знаходитись за адресою *0.0.0.0:4567/* і буде приймати GET запити, друга - *0.0.0.0:4567/api/control* і буде приймати POST запити.

Контроль організовано наступним чином. Коли користувач натискає якусь із клавіш “вперед”, “назад”, “вліво”, “вправо”, на сервері запит обробляється і на відповідний вихід GPIO віддається високий сигнал(для клавіш “вперед” та “назад”) або задається відповідний пульс(для клавіш “вліво” та “вправо”). Коли користувач відпускає клавішу, на сервер відправляється запит і він зупиняє подачу високого сигналу або подає пульс, потрібний для збереження повороту на рівні $0^{o}$.

Для керування двигунами постійного струму використовується мікроконтроллер H-Bridge L9110. Кожен HG7881 (L9110) чіп здатний керувати одним двигуном постійного струму за допомогою двох цифрових входів управління. Один вхід використовується для вибору напрямку обертання двигуна в той час як інший використовується для управління швидкістю обертання двигуна. Швидкість регулюється за допомогою ШІМ(широтно-імпульсної модуляції). Пристрій має свою таблицею істинності, яка визначає результат подачі сигналів на його входи. Знайти її можна на таблиці .

Таблиця 2.1 - L9110 Таблиця істинності

| Вхід | | Вихід | | |
|---|---|---|---|---|
| IA | IB | OA | OB | Опис |
| 0 | 0 | 0 | 0 | Вимкнений |
| 1 | 0 | 1 | 0 | Вперед |
| 0 | 1 | 0 | 1 | Назад |
| 1 | 1 | 1 | 1 | Вимкнений |

Варто звернути увагу, що фактичний напрямок "вперед" і "назад" залежить від того, як двигуни змонтовані і підключені. Завжди можна змінити напрямок двигуна шляхом зміни його проводки.

L9110 використовує двоканальний модуль двигуна використовує. Кожен канальний драйвер призначений для управління одним двигуном, так що наявність двох означає, що цей модуль може управляти двома двигунами незалежно один від одного. Кожен канал двигуна використовує ту ж таблицю істинності, як описано вище. Кожен набір гвинтових клем використовується для підключення двигуна.

Рекомендується використовувати вхід 1A для управління швидкістю обертання кожного двигуна і вхідний 1B для керування напрямком [15].

Код керування, написаний на Java, можна знайти у додатку А.

Таблиця 2.2 - Таблиця з’єднань двигунів до L9110

| Вхід | Опис |
|---|---|
| B-IA | Двигун B вхід A |
| B-IB | Двигун B вхід B |
| GND | “Земля” |
| VCC | Операційна напруга 2.5-12B |
| A-IA | Двигун A вхід A |
| A-IB | Двигун A вхід B |

Керування серво Futaba S3003 побудоване на основі принципу широтно-імпульсної модуляції. Широтно-імпульсна модуляція (ШІМ — англ. pulse-width modulation, PWM), або модуляція за тривалістю імпульсів (англ. pulse-duration modulation, PDM) — процес керування шириною (тривалістю) високочастотних імпульсів за законом, який задає низькочастотний сигнал. В електроніці це може бути керування середнім значенням вихідної напруги шляхом зміни тривалості замкнутого стану електронного (електромеханічного) ключа, наприклад, у схемі ключового стабілізатора напруги. В даному випадку він використовується для контролю напруги, що подається на серво і, а отже і те, як швидко він обертається. На рис. 2.7 показаний сигнал з ШІМ контакту з Raspberry Pi [16].

У випадку Futaba S3003, чим більше напруги подається на серво, тим більший кут повороту робить його двигун. Це дозволяє використовувати пристрій як основу для повороту системи. Для створення ШІМ сигналу з контакту на Raspberry Pi використовується наступний код.

```
softPwmCreate(6, 10, 100);
softPwmWrite(6, cycle);
```

У ньому вказано, що система створить ШІМ сигнал на контакті під номером 6 з довжиною відрізку в 100 з початковим значенням 10 за допомогою методу *softPwmCreate*. Для встановлення нового значення використовується метод *softPwmWrite*, який за аргументи приймає номер контакту та нове значення довжини сигналу, яке треба встановити. У нашому випадку ми емпірично встановили, що поворот для наліво нам потрібно встановити значення 8, для того, щоб лишати серво по центру - 10, для повороту направо - 12(рис. 2.7) [17].

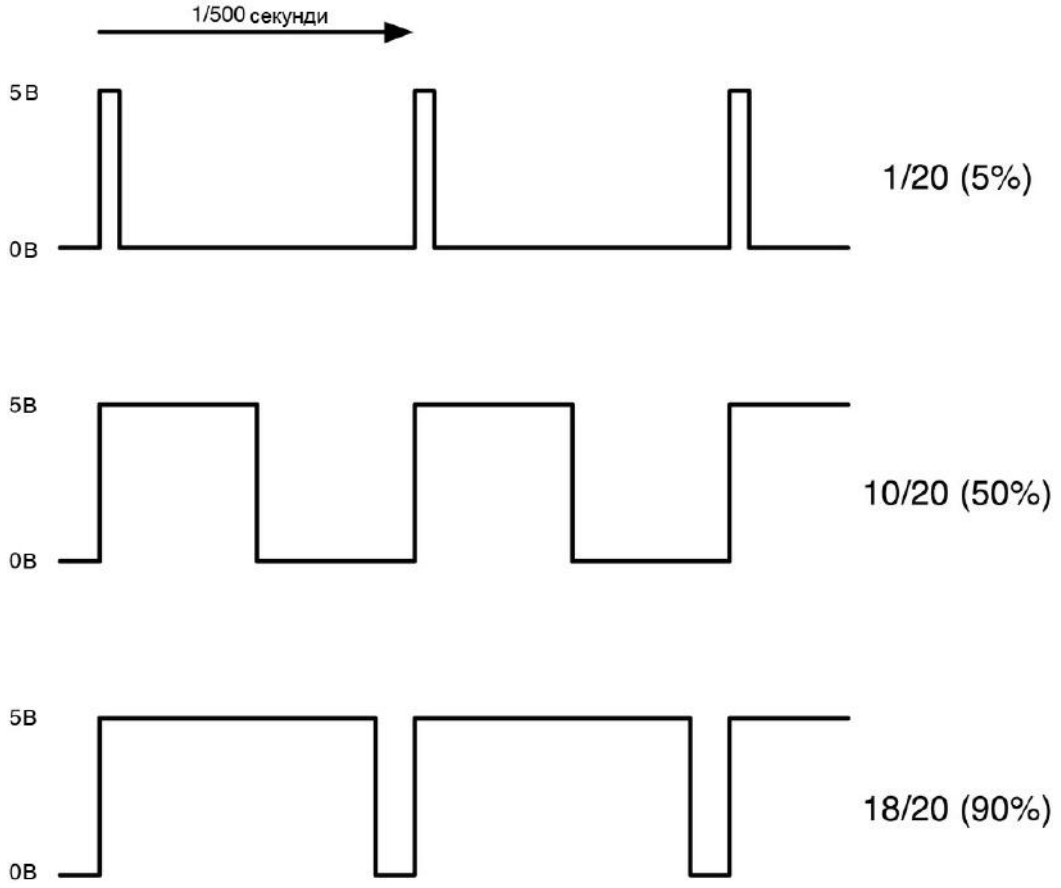


Рисунок 2.7 - Сигнал з ШІМ контакту з Raspberry Pi

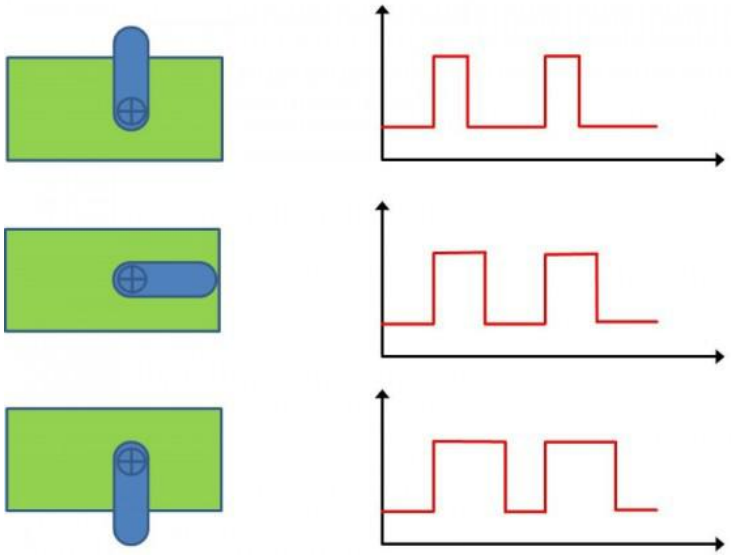

Рисунок 2.8 - Емпірично встановлені залежності між шириною сигналу та поворотом серво

## 2.3 Підключення камери

Для отримання картинки оточення системи використовуємо камеру. У даній роботі це Waveshare RPi Camera D. Вона підключається до Raspberry Pi через звичайний шлейф до роз'єму CSI(Camera Serial Interface - Серійний інтерфейс камери). Для знімання зображення у Raspberry Pi є спеціальна утиліта командного рядка *raspistill*. Для її використання було написано клас, який можна знайти у додатку А. Java-програма створює дочірній процес, що запускає *raspistill* з потрібними параметрами і зберігає зображення. Далі цей клас використовується нашим ядром розпізнавання.

## 2.4 Налаштування потокового відео

Для організації потокового відео використовується Linux утиліта UV4L - User space Video4Linux collection. Для встановлення усіх її частин, потрібних для потокового відео, використовуються наступні команди.

```
$ wget http://www.linux-
projects.org/listing/uv4l_repo/lrkey.asc && sudo apt-key add
./lrkey.asc
```

Додаємо наступний рядок у */etc/apt/sources.list*:

```
$ sudo nano /etc/apt/sources.list
deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/ wheezy main
```

Встановлюємо потрібні утиліти:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install uv4l uv4l-raspicam
$ sudo apt-get install uv4l-raspicam-extras
$ sudo apt-get install uv4l-server
$ sudo apt-get install uv4l-uvc
$ sudo apt-get install uv4l-xscreen
$ sudo apt-get install uv4l-mjpegstream
$ sudo reboot
```

Для того, щоб створити потік відео з камери, використаємо команду:

```
$ sudo uv4l -nopreview --auto-video_nr --driver raspicam --encoding mjpeg --width 640 --height 480 --framerate 20 --server-option '--port=9090' --server-option '--max-queued-connections=30' --server-option '--max-streams=25' --server-option '--max-threads=29'
```

Для зупинки процесу передачі потокового відео скористаємось Linux утилітою **pkill:**

```
$ sudo pkill uv4l
```

Таким чином, ми можемо підключитись за локальною IP-адресою та портом *9090* з клієнта у браузері та виводиит потік відео з камери на екран клієнту. Інтерфейс клієнта зображений на рис. 2.9 [18].

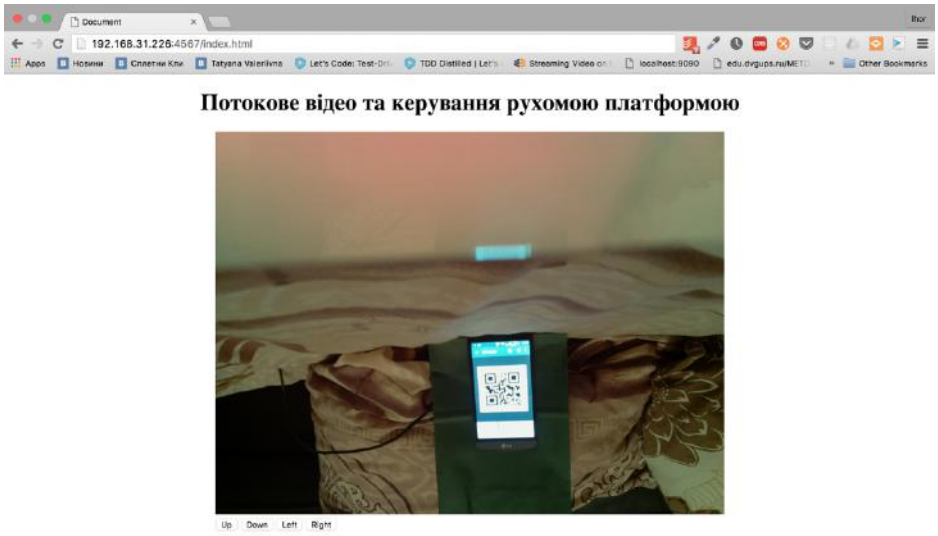


Рисунок 2.9 – Інтерфейс клієнта для перегляду потокового відео та керування рухомою платформою

Одночасно клієнт для перегляду потокового відео з платформи має інтерфейс зовнішнього керування її рухом. Для керування системою вожна використовувати клавіші ВВЕРХ, ВНИЗ, ВЛІВО або ВПРАВО, а також відповідні кнопки у вікні клієнського додатку. При натисканні цих клавіш клієнт відправляє HTTP-запит на сервер системи, яка вирішує, як їй рухатись далі.

Код клієнта знаходиться у додатку А.

## 2.5 Розрахунок затримок на виконання команд

Команди із клієнта доходять до сервера і звідти до апаратного забезпечення із певною затримкою. По-перше, це затримка мережі WiFi. По-друге, це затримка передачі даних від GPIO до контролерів.

Затримку HTTP-запиту до серверу будемо вимірювати за допомогою браузеру Google Chrome.

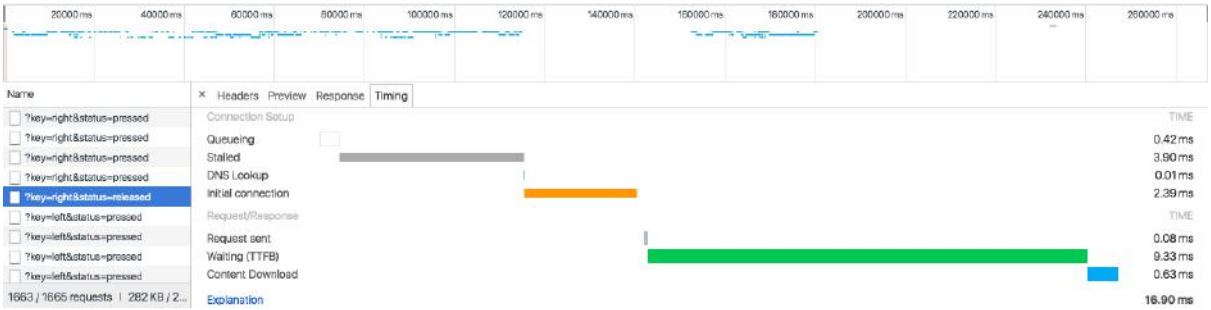


Рисунок 2.10 - Вимірювання швидкості відповіді сервера за допомогою панелі розробника у браузері Google Chrome

Як бачимо на рис. 2.10, час на очікування відповіді від сервера в середньому склав 9мс. Час на передачу запиту по HTTP складає від 1 мс до 5 мс (залежно від = параметрів локальної WiFi мережі), До 4 мс - на виконання команди на стороні сервера: обробка запиту та переведення контакту GPIO у потрібний стан(високий або низький сигнал).

Тестування швидкості виконання команди на стороні GPIO виконувалось таким чином. Тест - перемикання одного з контактів GPIO між нулем і одиницею якомога швидше.

## 2.6 Висновки

У цьому розділі було розглянути процес побудови рухомої платформи прототипу для тестування, клієнта для керування та перегляду потокового відео з рухомої платформи розраховано затримки на виконання команд

Для перевірки системи на коректність роботи було проведено інтеграційне тестування. Всі частини системи були з’єднані включино з джерелами живлення. На стороні Raspberry Pi було запущено сервер, що приймає запити на рух системи та передає зображення з камери на екран клієнту.

Результати тестування показали, що все налаштовано та функціонує коректно та очікувано.

# 3 ВИРІШЕННЯ ЗАДАЧІ РОЗПІЗНАВАННЯ

## 3.1 Технології та підходи комп’ютерного бачення

Computer Vision - теорія та технологія створення машин, які можуть проводити виявлення, стеження та класифікацію об'єктів. Як наукова дисципліна, комп'ютерний зір належить до теорії та технології створення штучних систем, які отримують інформацію у вигляді зображень. Відеодані можуть бути представлені у вигляді багатьох форм, таких як відеопослідовність, зображення з різних камер або тривимірними даними з медичного сканера.

### 3.1.1 Задачі комп’ютерного бачення

Реалізація систем комп’ютерного бачення залежить від області їх застосування, апаратної платформи і вимог до продуктивності. Кожна область застосування комп’ютерного бачення пов’язана з рядом типових задач, таких як задачу вимірів або обробки сирих зображень.

### 3.1.2 Задача розпізнавання

Класична задача в комп’ютерному баченні це визначення, чи містять відеодані певний характерний об’єкт, особливість чи активність. Ця задача може бути легко та достовірно вирішена людиною, але досі не вирішена в комп’ютерному баченні на достатньому рівні в загальному випадку: випадкові об’єкти у випадкових ситуаціях. Існуючі методи вирішення цієї задачі ефективні тільки для окремих об’єктів, таких як прості геометричні об’єкти, людські обличчя, друковані або рукописні символи, автомобілі і тільки в певних умовах – зазвичай це визначене освітлення, фон і положення об’єкта відносно камери.

В літературі описано кілька підвидів задачі розпізнавання.

1) Власне розпізнавання: один чи кілька попередньо заданих або вивчених об’єктів чи класів об’єктів можуть бути розпізнані, зазвичай разом з їх двовимірним положенням на зображенні або трьохвимірним положенням на сцені.
2) Ідентифікація: розпізнається індивідуальний екземпляр об’єкту. Прикладами є ідентифікація людського обличчя, відбитків пальців або автомобіля.
3) Виявлення: відеодані перевіряються на наявність певної умови. Виявлення, базоване на відносно простих і швидких обчисленнях, інколи використовується для пошуку невеликих ділянок в зображенні, що аналізується. Потім ділянку обробляють за допомогою методів, що потребують більше ресурсів для отримання правильної інтерпретації.

### 3.1.3 Задача руху

Задачі руху – це задачі, пов’язані з оцінкою руху, в яких серія зображень (відеодані) обробляються для знаходження швидкості кожної точки зображення або 3D сцени. Прикладами таких задач є:

- Визначення трьохвимірного руху камери
- Спостереження, тобто слідкуванням за переміщенням об’єкту(наприклад, машини або людей)

### 3.1.4 Задача відновлення сцени

Відновлення сцени має за задачу відбудову трьох вимірної моделі сцени. В найпростішому випадку, моделлю може бути набір точок трьохвимірного простору. Більш складні методи відбудовують повну трьохвимірну модель.

### 3.1.5 Задача відновлення зображення

Задача відновлення зображення – це видалення шуму (шуму датчиків, розмитість об’єкта у русі). Найбільш простим підходом до вирішення задачі є різні типи фільтрів, таких як фільтри верхніх або нижніх частот. Більше складні методи використовують представляення того, як мають виглядати ті чи інші

ділянки зображення, і на основі цього їх зміна. Більш високий рівень видалення шумів досягається в ході першочергового аналізу відеоданих на предмет різних структур, таких як ліній чи границі, а далі керування процесом фільтрації на основі цих даних.

### 3.1.6 Використання систем комп'ютерного бачення

Одним з найбільш важливих застосувань систем комп'ютерного бачення є обробка зображень в медицині. Ця область характеризується отриманням інформації з відеоданих для підготовки медицинського діагнозу пацієнтам. В більшості випадків відеодані отримуються за допомогою мікроскопії, рентгенографії, антіографіфї, ультразвукових досліджень і томографії. Прикладом інформації, що може бути отримана з таких відеоданих є виявлення пухлин, атеросклерозу чи інших злоякісних змін. Также прикладом може бути зміна розмірів органів, кровотоки.

Найбільшою областю, де використовується комп'ютерний зір, є воєнна сфера. Очевидними прикладами є виявлення ворожих солдат і транспортних засобів і керування ракетами. Найбільш досконалі системи керування ракетами посилають ракету в задану область замість конкретної цілі, а селекції цілі проводиться на етапі, коли ракета входить в задану область, базуючись на отриманому відеоряді. Сучасне воєнне поняття, таке як "бойова усвідомленість", має на увазі що різні датчики, включає датчики зображення і надає великий набір інформації про поле бою, яка може бути використана для прийняття стратегічних рішень. В цьому випадку автоматична обробка даних використовується для зменшення складності чи збільшення надійності отримуваної інформації.

Однією із нових областей застосування цієї технології є автономні транспортні засоби, включаючи підводні, наземні, повітряні. Рівень автономності варіюється від повністю автономних до траспортних засобів, де

системи, базовані на комп’ютерному зорі підтримують водія чи пілота в різних ситуаціях. Повністю автономні транспортні засоби використовують комп’ютерний зір для навігації, тобто для отримання інформації про місце свого знаходження для створення карти навколишнього середовища для виявлення перешкод.

### 3.1.7 Переваги та недоліки використання систем комп’ютерного зору у рухомих платформах

Основними перевагами використання систем комп’ютерного зору в мобільних системах є, по суті, надання комп’ютеру можливості бачити та розпізнавати навколишній світ. За допомого показань камери легко розпізнавати кольори, зображення за певними шаблонами. Недоліками є те, що вирішувати задачі знаходження відстані до об’єкту є досить складно і вимагає не однієї, а кількох камер [19].

## 3.2 Розпізнавання маркерів

У даній роботі розглядаються два типи маркерів для надання системі інформації, де саме вона знаходиться на карті в даний момент. Це маркери по кольору та маркери по QR коду. Для розпізнавання маркерів по кольору використовується бібліотека *OpenCV* для мови Java. Для розпізнавання QR-кодів використовується бібліотека від Google *ZXing*.

Розглянемо детальніше варіант розпізнавання кольорових маркерів. Для початку, нам потрібно обрати кольори, в які можуть бути замальовані маркери. Далі задамо порогові значення цих кольорів - мінімальні та максимальні. Це робиться методом *Core.inRange* - відфільтрує усі кольори, які не входять в цей інтервал. Далі знаходиться відстань до цього об’єкту і отримані дані передаються у модуль побудови карти [20].

Для розпізнавання QR-з використанням *ZXing* потрібно завантажити зображення в оперативну пам’ять та використати клас *MultiFormatReader*, який

вміє розпізнавати QR-код та повертати закодоване значення та його пложення на зображенні.

Знаходження відстані до маркера – задача нетривіальна. У зв’язку з відстуністю потужних ультразвукових або інфрачервоних сенсорів для вимірювання цієї величини використовується лише камера. Так як нам відомі параметри камери та розміри маркеру, то за допомогою властивості лінзи сенсора камери можемо розрахувати досить точну відстань до об’єкту.

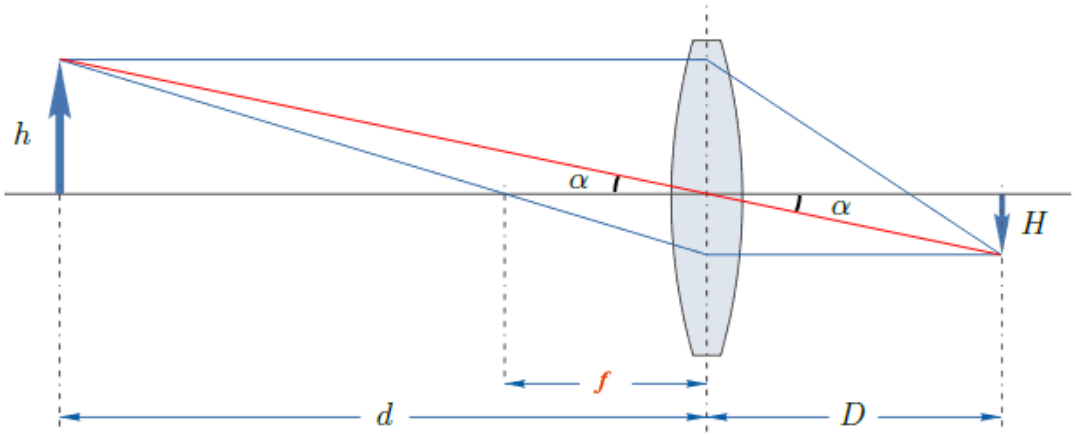


Рисунок 3.1 – Схема плоскої лінзи

На рис. 3.1 зображена схема плоскої лінзи, де d – відстань від лінзи до об’єкту, D – відстань від лінзи до зображення об’єкту(на матриці чи плівці), f – фокальна відстань лінзи.

Формула тонкої лінзи зв’язує ці три величини:

$$\frac{\mathbf{1}}{\boldsymbol{d}} + \frac{\mathbf{1}}{\boldsymbol{D}} = \frac{\mathbf{1}}{\boldsymbol{f}'} \tag{3.1}$$

Так як

$$\boldsymbol{h} = \boldsymbol{d} * \mathbf{tan}\,(\boldsymbol{\alpha}), \tag{3.2}$$

$$\boldsymbol{H = D \ * \ tan(\alpha)}, \tag{3.3}$$

тоді

$$\boldsymbol{d = f + \frac{f \ * \ h}{H}}. \tag{3.4}$$

Так як

$$\boldsymbol{f \ll \frac{f*h}{H}}, \tag{3.5}$$

тоді формулу можна спростити до вигляду

$$\boldsymbol{d \approx \frac{f \ * \ h}{H}}. \tag{3.6}$$

З цієї формули нам відомі значення *h* – реальна висота об’єкту та *H* – висота об’єкту у пікселях на екрані. У нас залишається дві невідомі величини – *d* та *f.* Так як кінцевою метою є універсальне знаходження відстані *d* від камери до об’єкта, потрібно визначити універсальне *f* – фокальну відстань камери.

Нехай маємо маркер 0.085 х 0.11 м (по вертикалі, *h* = 0.085), D = 0.24 м перед камерою, що робить знімок(рис. 3.2). Виміряємо висоту маркеру на екрані у пікселях, *H* = 248px.

Таким чином фокусна відстань для зображення розміром 1296 х 972 px:

$$\boldsymbol{f \approx \frac{248\,px \ * \ 0.24 \text{ м}}{0.11 \text{ м}} = 541\,px}. \tag{3.7}$$

Зафіксуємо це значення для обчислень як константу у коді.

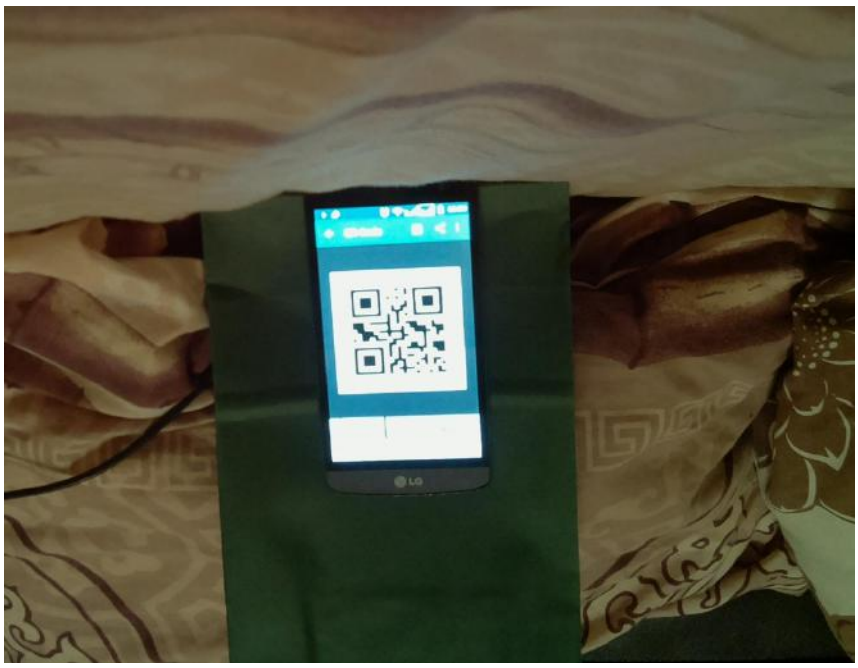

Рисунок 3.2 – Емпіричне визначення фокальної відстані камери у пікселях для заданого розміру фотографії

Далі для знаходження відстані використовуємо відому величину f і виконуємо обчислення за формулою (3.6).

Обчислення дають результат у межах допустимої похибки 5%, як можна бачити з Таблиці 3.1.

Таблиця 3.1 – Порівняння реальної та виміряної відстані

| Реальна відстань, м | Виміряна відстань, м | Похибкам, м | Похибка, % |
|---|---|---|---|
| 0.52 | 0.50 | 0.02 | 4% |
| 1.23 | 1.17 | 0.05 | 4% |
| 4.56 | 4.38 | 0.12 | 3% |

У камері, що використана у даній роботі, фокальна відстань 3.37 мм, висота сенсору - 6.35 мм, висота зображення – 972 px. Реальний розмір маркеру сталий, розмір маркеру на зображенні визначаємо описаним вище алгоритмом розпізнавання маркерів. Із цього можемо зробити висновок, що усі дані для знаходження відстані до об’єкта наявні [21].

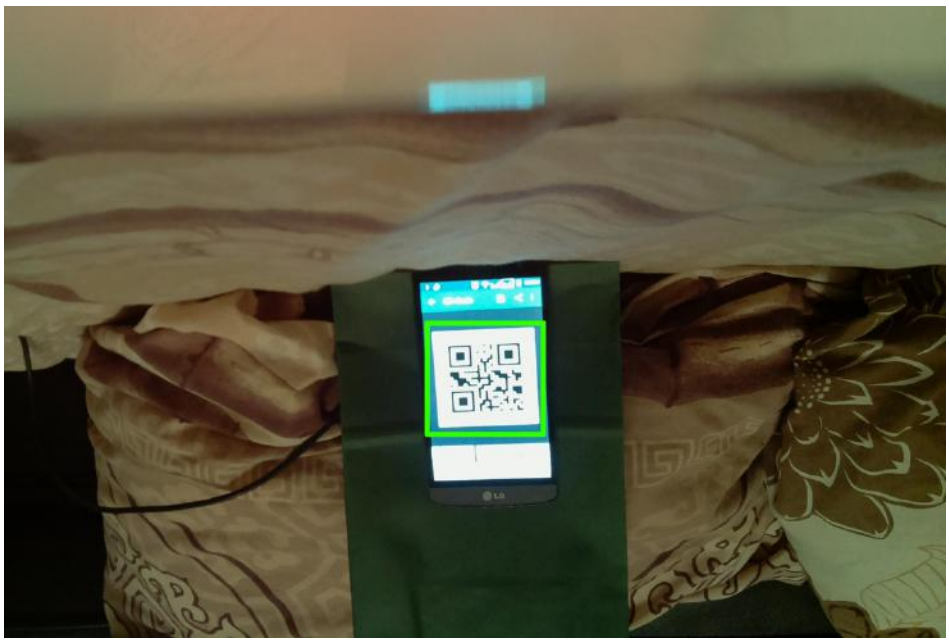

Рисунок 3.3 – Розпізнане положення QR-коду на зображенні обведене границею зеленого кольору

Вважаємо, що маркер та камера знаходяться на одній висоті паралельно один одному.

Для розуміння, на який кут відносно розпізнаного зображення відхилена камера, знайдемо відстань від центра зображення до центра розпізнаного маркера – це буде довжина протилежного катета прямокутного трикутника ACB(рис. 3.4) – BC, довжина прилеглого катета AC – відстань до об’єкта.

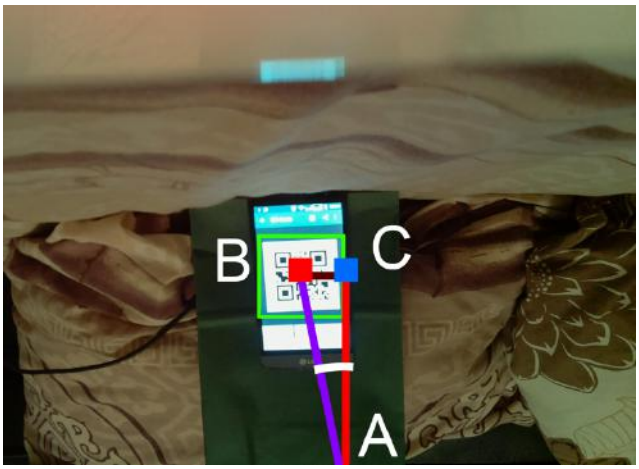


Рисунок 3.4 – Прямокутний трикутник ACB

Так як нам відома реальна фокальна відстань і відома фокальна відстань у пікселях, можемо дізнатися реальну кількість пікселів у метрі і перетворити відстань до об’єкта с метрів у пікселі для визначення кута відхилення об’єкта від камери.

Скористаємось властивостями функції арктангенса:

$$\boldsymbol{\alpha = \arctan\left(\frac{BC}{AD}\right)}. \tag{3.8}$$

Отримаємо відому відстань $d$ до маркера, а також відхилення α – відхилення, що показує, наскільки далеко від центра знаходиться маркер. Ці дані можемо передавати алгоритму локалізації.

Використання маркерів на основі QR-кодів точніше, тому переважно ми будемо використовувати його, а у випадку, коли система не може визначити достатньо точно своє місцезнаходження, буде використовуватись фільтр по кольору. Маркери мають одночасно і колір, і QR-код [22].

Кодова реалізація даного алгоритму знаходиться у додатку А.

## 3.3 Висновки

У даному розділі було розглянуто процес розпізнавання маркеру та знаходженя відстані до нього. Задача була вирішена успішна із застосування властивостей камери як тонкої лінзи. Звісно, результати можна покращити, але для цього потрібно використати більш потужний набір сенсорів. Процес знаходження маркера на зображенні є достатньо оптимальним для коректної роботи системи, але його можна покращати, переписавши алгоритм з використанням більш низькорівневих інструментів.

Для покращення результатів знаходження відстані до об’єкту потрібно мати такі компоненти, як ультразвукові та інфрачервоні датчики, що здатні отримувати інформацію про навколишнє середовище не точково, а на усі $360^{O}$.

# 4 ВИРІШЕННЯ ЗАДАЧІ ЛОКАЛІЗАЦІЇ

Локалізація - здатність машини визначати своє місце у просторі. Отже, вирішенням задачі локалізації буде розташування робота на відомій йому карті за відомими даними сенсора.

## 4.1 Опис вхідних даних

Вхідними даними до алгоритму є дані з сенсорів - камери та ехолокатору і карта з перешкодами та маркерами.

Карту можна представити як набір точок з описаними перешкодами та маркерами.

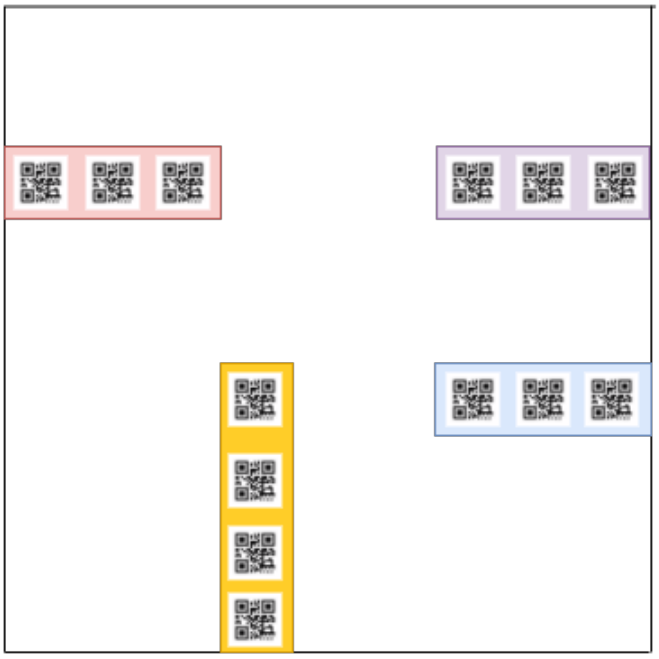

Рисунок 4.1 - Графічне представлення карти

Вона має вигляд блоків різного кольору з QR-кодом для уточнення положення. Кожен QR-код містить зашифрований номер лінії та стовпчика.

Її можна представити у вигляді двовимірного масиву і використовувати у тілі алгоритму локалізації.

Відповісти точно, де знаходиться на карті система в даний момент часу неможливо. Можна лише дати відповідь, що система знаходиться в певному місці з деякою ймовірністю. Для пояснення ідеї розглянемо одновимірний приклад.

Спочатку система, котра потрапила на карту, не має ніяких даних про те, де вона знаходиться, тобто ймовірність перебування на кожній клітині карти буде *однаковою*. Тобто, ми маємо *рівномірно* розподілену випадкову величину. Якщо ми зобразимо на графіку ймовірностей з ймовірністю на вертикальній осі та позицією на горизонтальній вісі, то ми отримаємо пряму лінію(рис. 4.2). Ця лінія описує функцію рівномірного розподілу - це стан максимального нерозуміння [23].

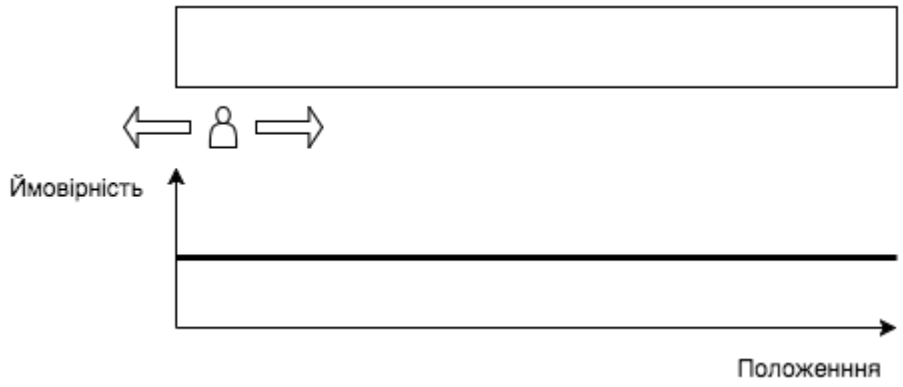


Рисунок 4.2 - Робот у стані максимального нерозуміння

Припустимо, що на карті зображено три приблизно однакові маркери зеленого кольору, і система може відрізнити позиція біля маркеру від позиції без маркеру навпроти (рис. 4.3).

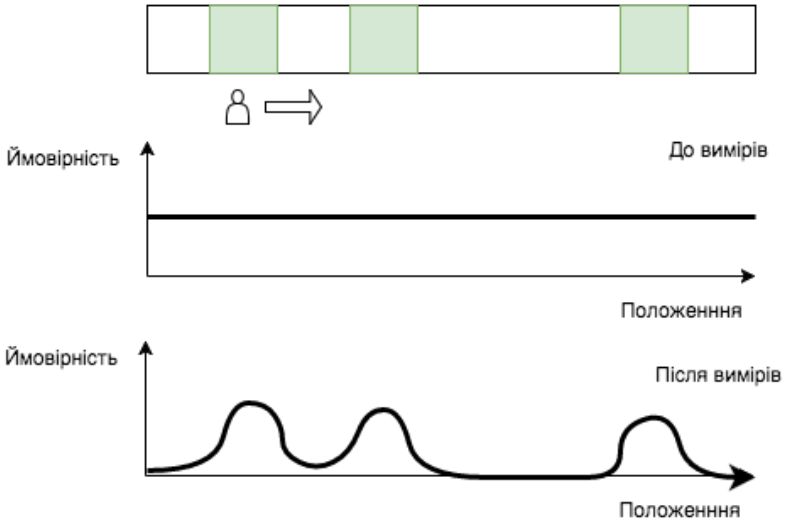


Рисунок 4.3 - Функція розподілу ймовірностей після отримання інформації про середовище

Коли робот зробить заміри сенсорами, він отримає інформацію, що знаходиться навпроти зеленого маркеру. Звичайно, тоді ймовірність бути навпроти конкретного зеленого маркеру буде рівна ⅓, адже система точно знає карту і розуміє, що на карті всього 3 маркери, тому ймовірність ділиться порівну між трьома маркерами.

Після того, як система зробила огляд навколишнього середовища, робот робить рух. Функція розподілу також рухається на відповідний зсув і ми отримаємо графік вигляду, зображеного на рис. 4.4.

Так як робот рухається не на точну відстань, а з деякою похибкою, то отриманий розподіл буде трохи змінений в сторону більш пологої функції і зменшення ймовірності [24].

Розглянемо наступну ситуацію. Робот зупинився біля зеленого маркера, тобто з його точки зору функція розподілу не змінилась. Але так як після першого виміру система перебувала в стані абсолютного нерозуміння того, де вона знаходиться, то зараз вона володіє інформацією, яким був попередній

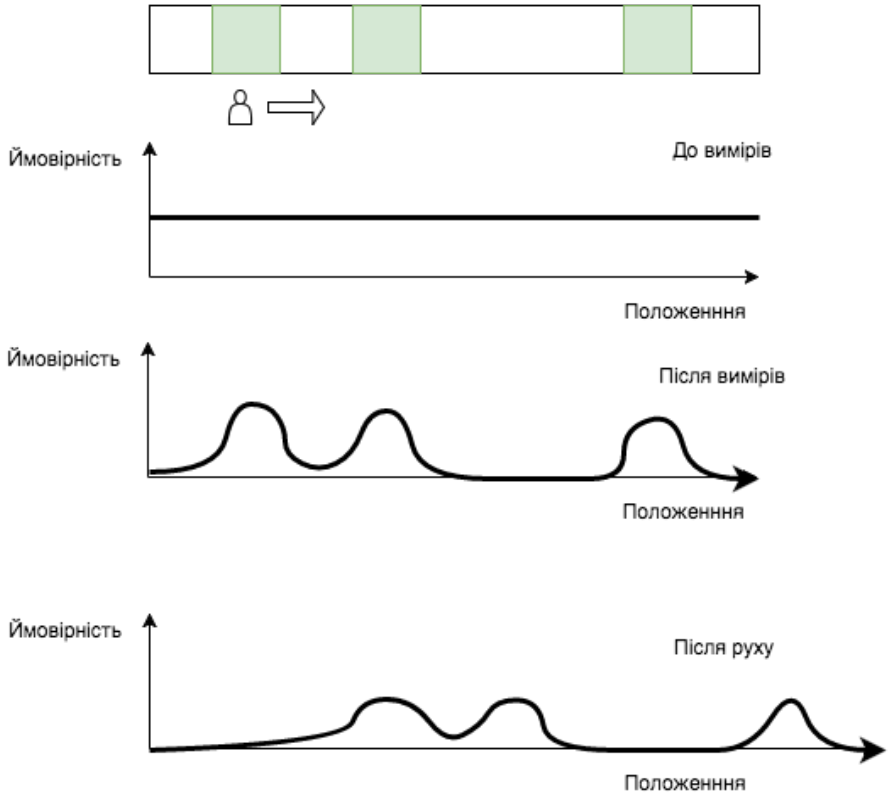


Рисунок 4.4 - Функція розподілу ймовірностей після руху

розподіл ймовірностей. Об’єднавший ці два розподіли за допомогою згортки функцій, отримаємо розподіл, зображений на рис. 4.5. Згортка - це математична операція, що бере дві функції та повертає величину їх перетину. Якщо функції взагалі не пертинаються, значення згортки буде рівним 0, а якщо перетинаються повністю - то 1. У даному обчисленні проходимо між двома граничними випадками, тобто згортка буде набувати значення від 0 до 1.

Як можна бачити з рисунків, в даний момент часу система може однозначно визначити, де вона знаходиться, адже різниця між ймовірностями перебування у різних точках значна. Цей підхід називається *методом гістограмних фільтрів.*

## 4.2 Висновки

Локалізація робота визначає його місцезнаходження за всіма можливими місцях. З точки зору математики, можна стверджувати, що робот постійно

оновлює свою ймовірність перебування в кожному місці карти, тобто кожна точка на карті має свою ймовірність.

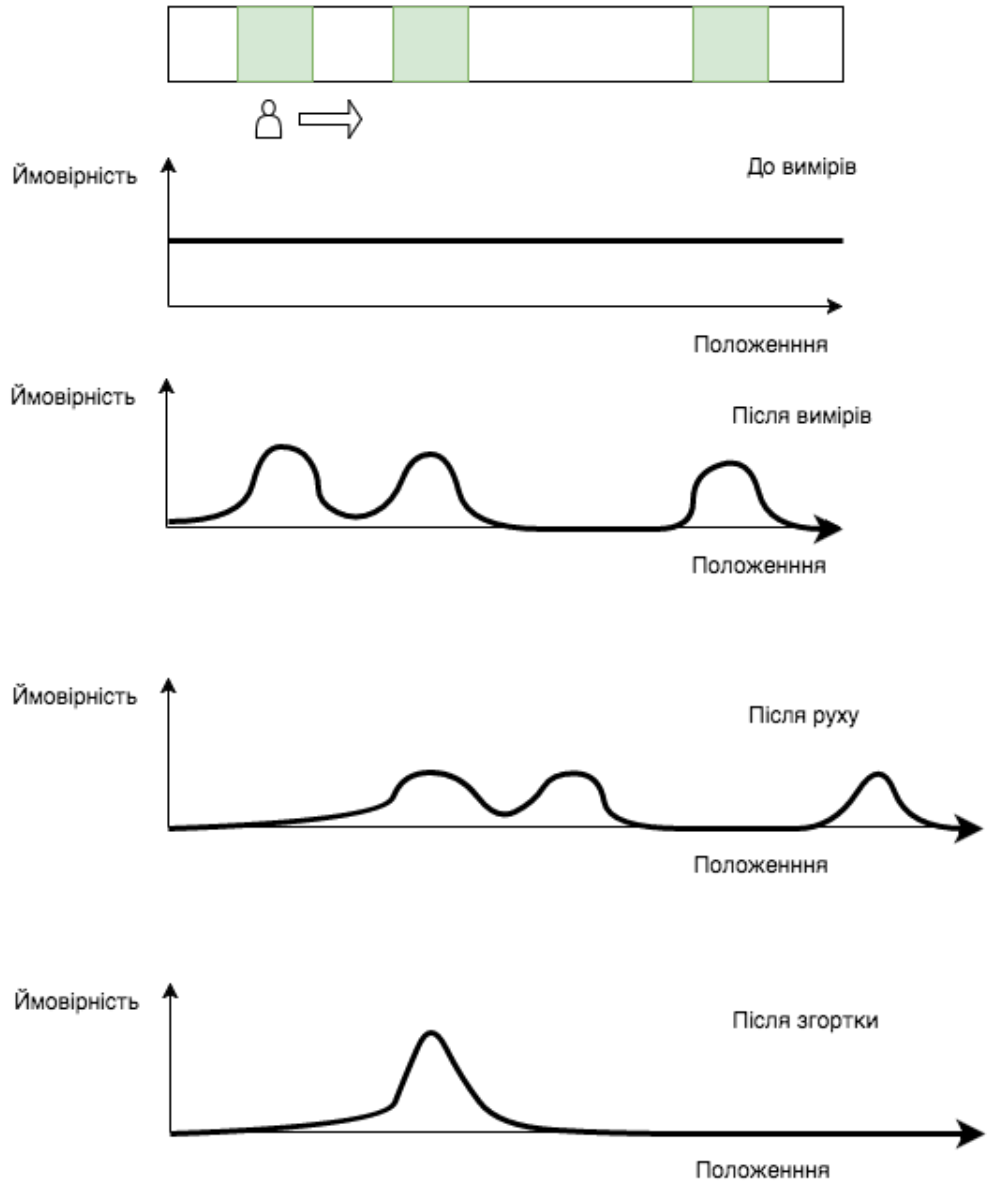


Рисунок 4.5 - Функція розподілу ймовірностей положення системи після згортки

Якщо штучний інтелект системи робить свою роботу належним чином, розподіл цієї ймовірності мусить мати дві властивості:

1) він повинен мати один гострий пік - це вказує на те, що автомобіль має дуже конкретне переконання про його поточне місцезнаходження;
2) пік має бути обраним правильно, адже у противному випадку система зможе потрапити у ситуація, коли буде рухатись зовсім не туди, куди було вказано.

Отже, алгоритм для локалізації, що був використаний у даній роботі, може бути записаний у вигляді послідовності наступнихї кроків:

1) старт з початковими знаннями про навколишнє середовище;
2) множення початкової ймовірності з результатами вимірів сенсорів;
3) нормалізація отриманого результату;
4) рух далі за допомогою використання згортки.

# 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик мобільної навігаційної системи, що за допомогою програмного продукту орієнтується у відомому на карті просторі. Програмне забезпечення було розроблено за допомогою мов програмування Python, Bash та Java у середовищі розробки JetBrains Intellij Idea. Браузерний інтерфейс користувача створений за допомогою браузерної мови програмування JavaScript. Апаратний засіб був побудований на основі Raspberry PI.

Програмний продукт призначено для використання на персональних комп’ютерах під управлінням операційної системи Linux або Mac OS X.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

## 5.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки мобільної навігаційної системи з використанням маркерів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій апаратного засобу та програмного продукту, здатного до самоорієнтації.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на Raspberry PI з визначеним набором компонентів;

– інтерфейс користувача повинен нормально функціонувати на будь-якій операційній системі у нових версіях веб браузерів;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту та апаратних засобів.

### 5.1.1 Обґрунтування функцій програмного продукту

Головна функція $F_0$– розробка мобільної навігаційної системи за допомогою маркерів. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

$F_1$ – вибір мови програмування;

$F_2$ – вибір апаратних компонентів;

$F_3$ – вибір алгоритму алгоритму локалізації.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція $F_1$:

а) мова програмування C++;

б) мова програмування Java;

Функція $F_2$:

а) використання мікрокомп’ютера Raspberry PI;

б) використання електричної платформи Ardruino.

Функція $F_3$:

а) використати готову реалізацію алгоритму локалізації;

б) реалізувати гістограмні, часткові та Калмановий фільтри для локалізації.

### 5.1.2 Варіанти реалізацій основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 5.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 5.1).

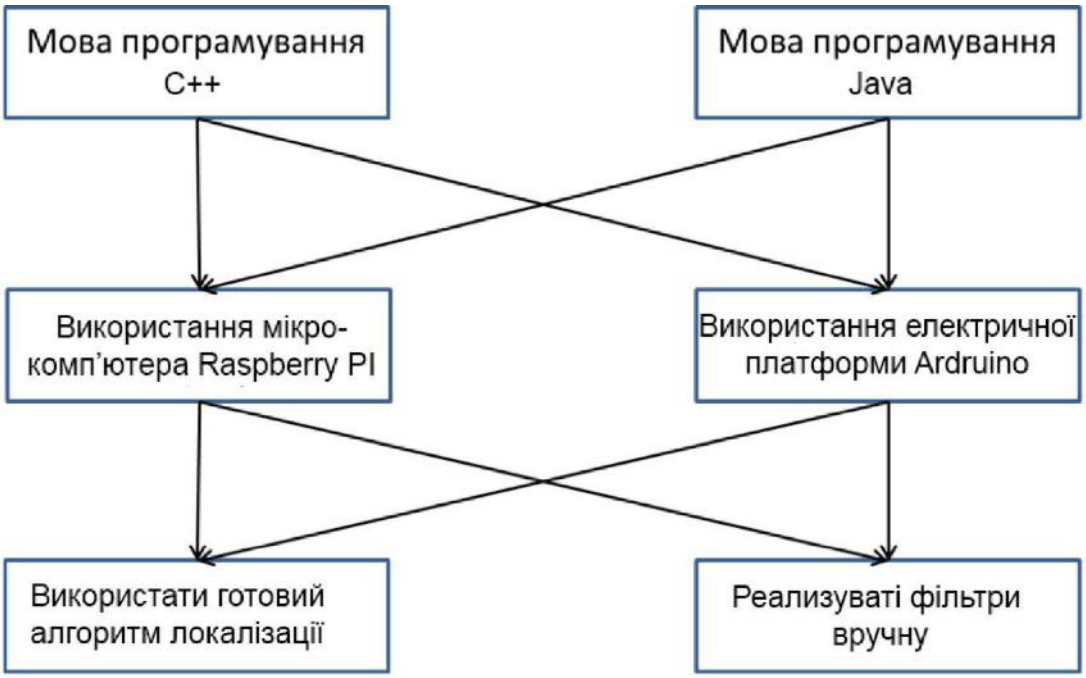


Рисунок 5.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки розрахунки проводяться з на Raspberry PI, нам потрібне швидке швидкість розробки.

Функція *F2:*

Оскільки ми плануємо зробити високорівневу реалізацію і нас не цікавить розробка мікроконтроллерів.

Функція *F3:*

Оскільки нам підходять і варіанти розробки алгоритму, і використання готової реалізації, розглянемо обидва варіанти.

Таблиця 5.1 – Позитивно-негативна матриця

| **Основні функції** | **Варіанти реалізації** | **Переваги** | **Недоліки** |
|---|---|---|---|
| *F1* | *А* | Код швидше виконується | Займає більше часу при написанні коду |
| | *Б* | Кросплатформений, вища швидкість написання | Код довше виконується, |
| *F2* | А | Простий у використанні | Менша точність підрахунків |
| | Б | Найбільше підходить при розробці мікроконтроллерів | Затрачений час |
| *F3* | *А* | Легкий у створенні | Недостатньо широкі можливості персоналізації |
| | *Б* | Широкі можливості персоналізації | Більше часу на розробку |

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1б – F2а – F3а
2. F1б – F2а – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

## 5.2 Обґрунтування системи параметрів ПП

### 5.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – швидкодія мови програмування;

– *X2* – швидкість написання коду;

– *X3* – простота використання апаратного забезпечення;

– *X4* – простота реалізації локалізації.

*X1*: Відображає швидкодію операцій залежно від обраної мови програмування.

*X2:* Відображає швидкість написання коду.

*X3:* Відображає складність у використанні апаратного забезпечення.

*X4:* Показує, наскільки просто і якою кількістю коду може бути вирішена задача локалізації

### 5.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 5.2.

За даними таблиці 5.2 будуються графічні характеристики параметрів – рис. 5.2 – рис. 5.5.

Таблиця 5.2 - Основні параметри ПП

| Назва Параметра | Умовні позначення | Одиниці виміру | Значення параметра | | |
|---|---|---|---|---|---|
| | | | гірші | середні | кращі |
| ***Швидкодія мови програмування*** | X1 | Оп/мс | 19000 | 11000 | 2000 |
| ***Швидкість написання коду*** | X2 | Строк коду/година | 32 | 16 | 8 |
| ***Простота використання апаратного забезпечення*** | X3 | Хвилин на підключення, використання та тестування модулю | 800 | 420 | 60 |
| ***Простота реалізації локалізації*** | X4 | кількість строк коду | 2000 | 1500 | 1000 |

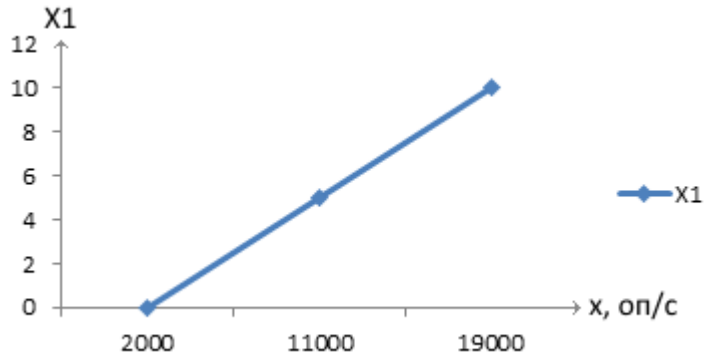


Рисунок 5.2 - X1, швидкодія мови програмування

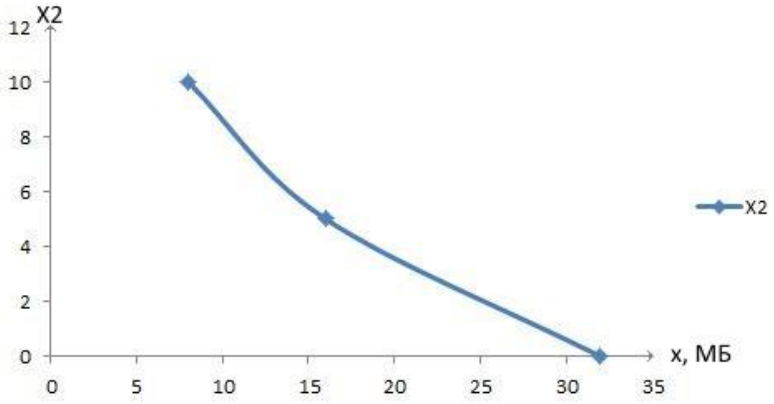


Рисунок 5.3 – X2, швидкість написання коду

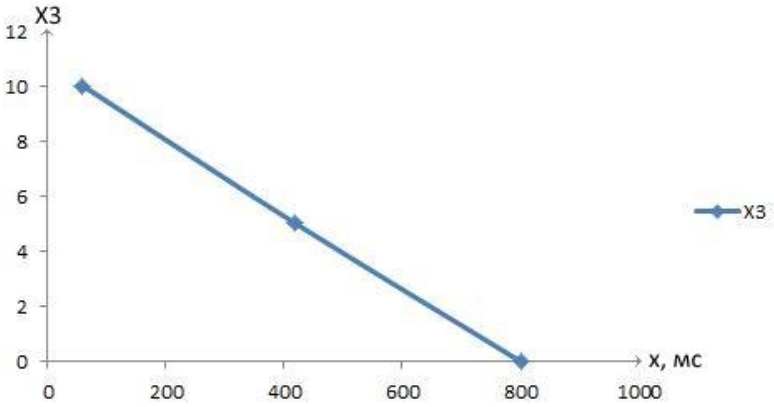


Рисунок 5.4 - X3, простота використання апаратного забезпечення

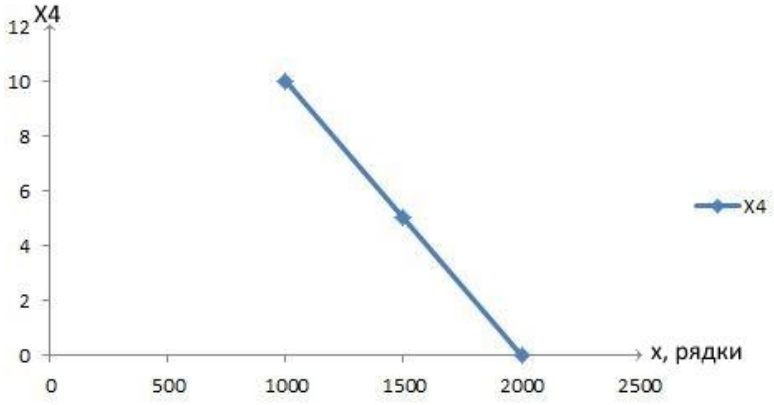


Рисунок 5.5 – X4, простота реалізації локалізації

### 5.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка

програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 5.3.

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105 \quad (5.1)$$

де $N$ – число експертів, $n$ – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26.25 \quad (5.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \quad (5.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

Таблиця 5.3 – Результати ранжування параметрів

| Позначення параметра | Назва параметра | Одиниці виміру | Ранг параметра за оцінкою експерта | | | | | | | Сума рангів $R_i$ | Відхилення $\Delta_i$ | $\Delta_i^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | |
| *X1* | Швидкодія мови програмування | Оп/мс | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 27 | 0,75 | 0,56 |
| *X2* | Швидкість написання коду | Строк коду/година | 4 | 4 | 4 | 3 | 4 | 3 | 3 | 25 | -1,25 | 1,56 |
| *X3* | Простота використання апаратного забезпечення | Хвилин на підключення, використання та тестування модулю | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 12 | -14,25 | 203,06 |
| *X4* | Простота реалізації локалізації | кількість строк коду | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 41 | 14,75 | 217,56 |
| | Разом | | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 105 | 0 | 420,75 |

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420.75 \tag{5.4}$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420.75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0.67 \tag{5.5}$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 5.4.

Таблиця 5.4 – Попарне порівняння параметрів

| Параметри | Експерти | | | | | | | Кінцева оцінка | Числове значення |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | |
| X1 і X2 | = | > | = | < | = | < | < | < | 0,5 |
| X1 і X3 | < | < | < | < | < | < | < | < | 0,5 |
| X1 і X4 | > | > | > | > | > | > | > | > | 1,5 |
| X2 і X3 | < | < | < | < | < | < | < | < | 0,5 |
| X2 і X4 | > | > | > | > | > | > | > | > | 1,5 |
| X3 і X4 | > | > | > | > | > | > | > | > | 1,5 |

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, $a_{ij}$ визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \end{cases} \quad (5.6)$$

0.5 при $X_i < X_j$.

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \quad (5.7)$$

де bi розраховується за наступною формулою:

$$b_i = \sum_{i=1}^{N} a_{ij}. \quad (5.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{5.9}$$

де b`$_i$ розраховується за наступною формулою

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j. \tag{5.10}$$

Як видно з таблиці 5.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5.5 – Розрахунок вагомості параметрів

| **Параметри $x_i$** | **Параметри $x_j$** | | | | **Перша ітер.** | | **Друга ітер.** | | **Третя ітер** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | X1 | X2 | X3 | X4 | $b_i$ | $K_{\text{ві}}$ | $b_i^1$ | $K_{\text{ві}}^1$ | $b_i^2$ | $K_{\text{ві}}^2$ |
| X1 | 1,0 | 0,5 | 0,5 | 1,5 | 3,5 | 0,219 | 22,25 | 0,216 | 100 | 0,215 |
| X2 | 1,5 | 1,0 | 0,5 | 1,5 | 4,5 | 0,281 | 27,25 | 0,282 | 124,25 | 0,283 |
| X3 | 1,5 | 1,5 | 1,0 | 1,5 | 5,5 | 0,344 | 34,25 | 0,347 | 156 | 0,348 |
| X4 | 0,5 | 0,5 | 0,5 | 1,0 | 2,5 | 0,156 | 14,25 | 0,155 | 64,75 | 0,154 |
| **Всього:** | | | | | 16 | 1 | 98 | 1 | 445 | 1 |

## 5.3 Аналіз рівня якості варіантів реалізаціх функції

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(швидкість написання коду) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *ХЗ (*Простота використання апаратного забезпечення) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) Хвилин на підключення, використання та тестування модулю 800 або варіанту б) 80.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 5.6):

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j}, \tag{5.11}$$

де $n$ – кількість параметрів;$K_{\text{в}i}$– коефіцієнт вагомості $i$–го параметра;$B_i$ – оцінка $i$–го параметра в балах.

Таблиця 5.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

| **Основні функції** | **Варіант реалізації функції** | **Абсолютне значення параметра** | **Бальна оцінка параметра** | **Коефіцієнт вагомості параметра** | **Коефіцієнт рівня якості** |
|---|---|---|---|---|---|
| F1(X1) | А | 11000 | 3,6 | 0,215 | 0,774 |
| F2(X2) | А | 16 | 3,4 | 0,283 | 0,962 |
| F3(X3,X4) | А | 800 | 2,4 | 0,348 | 0,835 |
| | Б | 80 | 1 | 0,154 | 0,154 |

За даними з таблиці 5.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \tag{5.12}$$

визначаємо рівень якості кожного з варіантів:

$K_{K1}$ = 0,774 + 0,962 + 0,835 = 2,57

$K_{K2}$ = 0,774 + 0,962 + 0,154 = 1,89

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

## 5.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.13)$$

де $T_P$ – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт;$K_{СК}$ – коефіцієнт на складність вхідної інформації; $K_M$ – коефіцієнт рівня мови програмування;$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_P$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7.

Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді, за формулою 5.13, загальна трудомісткість програмування першого завдання дорівнює:

$Т_1$ = 90·1.7·0.8 = 122.4 людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$Т_2$ = 27 · 0.9 · 0.8 = 19.44 людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I$ = (122.4 + 19.44 + 4.8 + 19.44) · 8 = 1328,64 людино-годин;

$Т_{II}$ = (122.4 + 19.44 + 6.91 + 19.44) · 8 = 1345.52 людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 6000 грн., один розробник апаратного забезпечення з окладом 9000грн. Визначимо зарплату за годину за формулою:

$$СЧ = \frac{М}{T_m \cdot t} \text{ грн.}, \quad (5.14)$$

де М – місячний оклад працівників;$T_m$ – кількість робочих днів тиждень;$t$ – кількість робочих годин в день.

$$СЧ = \frac{6000 + 6000 + 9000}{3 \cdot 21 \cdot 8} = 41{,}67 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$СЗП = С_ч \cdot Т_i \cdot КД, \quad (5.15)$$

де $С_ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 41,67 \cdot 1328.64 \cdot 1.2 = 66437,31$ грн.

II. $С_{ЗП} = 41,67 \cdot 1345.52 \cdot 1.2 = 67281,38$ грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 36,77%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.3677 = 66437,31 \cdot 0.3677 = 24429$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.3677 = 67281,38 \cdot 0.3677 = 24739$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 6000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 6000 \cdot 0,2 = 14400$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 14400 \cdot (1 + 0.2) = 17280$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.3677 = 17280 \cdot 0,3677 = 6353,86$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706{,}4 \cdot 0{,}156 \cdot 0{,}2436 \cdot 2 = 129{,}695$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0{,}67 = 5360$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС}$ = 17280 + 6353,86 + 2300 + 460 + 129,69 + 5360 = 31883,55 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М-Г} = С_{ЕКС} / Т_{ЕФ}$ = 31883,55/1706,4 = 18,68 грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} \cdot Т$$

I. $С_М$ = 18,68· 1328,64 = 24819 грн.;

II. $С_М$ = 18,68· 1345.52 = 25134,52 грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

I. $С_Н$ = 66437,31· 0,67 = 44513 грн.;

II. $С_Н$ = 67281,38· 0,67 = 45078,52 грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП}$ = 66437,31 + 24429 + 24819+ 44513 = 160198,31 грн.;

II. $С_{ПП}$ = 67281,38 + 24739 + 25134,52+ 45078,52 = 162233,42 грн.;

## 5.5 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \quad (5.16)$$

$К_{ТЕР1} = 5,214 / 160198,31 = 0,33 \cdot 10^{-4}$;

$К_{ТЕР2} = 3,569 / 162233,42 = 0,22 \cdot 10^{-4}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0,33 \cdot 10^{-4}$.

## 5.6 Висновки

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0,33 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Java;

– використання мікрокомп’ютера Raspberry PI;

– використання готового алгоритму локалізації.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

# ВИСНОВКИ

В результаті виконання роботи було розглянути необхідні компоненти для побудови мобільної навігаційної системи, їх призначення та принципи роботи; було побудовано повнофункціональний прототип системи на основі розглянутих компонентів. Також було проаналізовано можливості використатння різних типів маркерів та їх розпізнавання. Задача локалізації була вирішена за допомогою алгоритму гістограмних фільтрівю

У ході досліджень було визначено, що найдешевше у плані швидкодії та швижкості написання коду було розпізнавання маркерів у вигляді QR-коду. Це пояснюється тим, що камера системи якісні широкоформатні знімки, а QR-коди дуже стійкі до розпізнавання навіть у дуже поганій якості.

Використаний алгоритм локалізації дає досить точні результати, але з часом руху системи похибка може ставати досить значною. Це залежить не лише від самого алгоритму локалізації, а й від даних, що приходять на його вхід.

Для збільшення точності результатів та швидкості роботи алгоритму за допомогою вхідних потрібно використати потужніші сенсори, що дають повне уявлення про стан навколишнього середовища і тому дають можливість побудови карти навіть без наперед заданої.

Для збільшення точності роботи алгоритму локалізації також варто використати техніки фільтр Калмана та часткових фільтрів. Фільтр Калмана дасть зможу мінімізувати похибку при отриманні нових даних про положення з сенсорів.

В якості демонстрації проведеної роботи маємо робочу систему, що здатна отримувати дані про положення маркерів та карту на вхід і повертає знаходження робота на цій карті. Ці дані далі можуть бути інтегровані з

системою прокладання шляху, яка буде давати системі дані про те, як рухатись далі з метою досягнути певної точки на карті.

Отже, внаслідок проведеного дослідження стало зрозуміло, що потрібно спрямувати подальші дослідження та експерименти на використання більш потужних сенсорів та використанні складніших алгоритмів уточнення похибки. Розвиток сучасної робототехніки сьогодні направлений саме в цю сторону, тому, почавши ці дослідження, можна побудувати конкурентноспроможну систему, здатн до самокервання, яка має великий попит, адже зараз прийшов час інтелектуальних систем, які вміють працювати без втручання людини.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Raspberry Pi: Офіційний логотип Raspberry Pi. – Режим доступа: https://www.raspberrypi.org/wp-content/uploads/2015/08/raspberry-pi-logo.png. Дата доступа: 05.04.2016.
2. ExtremeTech: What is the Raspberry Pi? – Режим доступа: http://www.extremetech.com/computing/124317-what-is-raspberry-pi-2. − Дата доступа: 25.04.2016.
3. Raspberry Pi: Схема GPIO з логічною нумерацією котактів. – Режим доступа: https://www.raspberrypi.org/documentation/usage/gpio/images/a-and-b-gpio-numbers.png. Дата доступа: 10.04.2016.
4. Raspberry Pi: Схема GPIO з фізичною нумерацією котактів. – Режим доступа: https://www.raspberrypi.org/documentation/usage/gpio/images/a-and-b-physical-pin-numbers.png. Дата доступа: 10.04.2016.
5. Raspberry Pi: Електричне коло з перемикачем та світлодіодом. – Режим доступа: https://www.raspberrypi.org/documentation/usage/gpio/images/simple-circuit.png. Дата доступу: 10.04.2016.
6. CCM.net: What is WiFi and How Does it Work? - Режим доступа: http://ccm.net/faq/298-what-is-wifi-and-how-does-it-work. − Дата доступа: 13.04.2016.
7. Microsoft | Developer: Simple Java Wrapper Class for raspistill on the Raspberry Pi 2. – Режим доступа: https://blogs.msdn.microsoft.com/robert_mcmurray/2015/06/12/simple-java-wrapper-class-for-raspistill-on-the-raspberry-pi-2/. - Дата доступа: 18.03.2016.
8. TecBolivia: Принцип роботи ехолокаторів. – Режим доступа: http://tecbolivia.com/images/articulos/sensor_ping.jpg. Дата доступа: 15.04.2016.

9. ModMyPi: HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi. – Режим доступа: http://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi. - Дата доступа: 10.04.2016.
10. RaspberryPi: How can I connect the H-bridge L9110 with the Raspberry Pi? – Режим доступа: http://raspberrypi.stackexchange.com/questions/43102/how-can-i-connect-the-h-bridge-l9110-with-the-raspberry-pi. - Дата доступа: 20.03.2016.
11. Bajdi: L9110 H-bridge module. – Режим доступа: http://www.bajdi.com/l9110-h-bridge-module/. – Дата доступа: 03.03.2016.
12. Spark: Getting started. – Режим доступа: http://sparkjava.com/documentation.html. Дата доступа: 10.05.2016.
13. Sparkjava: Логотип Java Spark. – Режим доступа: http://sparkjava.com/assets/images/logo.svg. Дата доступа: 10.05.2016.
14. Pi4J Project: логотип The Pi4J Project. – Режим доступа: http://pi4j.com/images/logos/pi4j-header-small3.png. Дата доступа: 08.05.2016.
15. Banana Robotics: How to use the HG7881 (L9110) Dual Channel Motor Driver Module. – Режим доступа: https://www.bananarobotics.com/shop/How-to-use-the-HG7881-(L9110)-Dual-Channel-Motor-Driver-Module. Дата доступа: 22.04.2016.
16. Adafruit: Adafruit's Raspberry Pi Lesson 9. Controlling a DC Motor. – Режим доступа: https://learn.adafruit.com/adafruit-raspberry-pi-lesson-9-controlling-a-dc-motor/pwm. Дата доступа: 24.04.2016.
17. Kitronik: How to Control the S3003 Servo with a Microcontroller. – Режим доступа: https://www.kitronik.co.uk/blog/how-to-control-the-s3003-servo-with-a-microcontroller/. Дата доступа: 26.04.2016.
18. Instructables: Raspberry Pi Video Streaming. – Режим доступа: http://www.instructables.com/id/Raspberry-Pi-Video-Streaming/. Дата доступа: 05.05.2016.

19. Szeliski R. Computer Vision. Algorithms and Applications / Szeliski R. – London, UK: Springer-Verlag London Limited, 2011. – С. 3-25.
20. Baggio D. L. OpenCV 3.0 Computer Vision with Java / Baggio D. L. - Birmingham, UK: Packt Publishing Ltd, 2015. – С. 65-69.
21. Pyimagesearch: Find distance from camera to object/marker using Python and OpenCV. – Режим доступа: http://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/. Дата доступа: 11.05.2016.
22. Javapapers: Java QR Code. – Режим доступа: http://javapapers.com/core-java/java-qr-code/. Дата доступа: 12.03.2016
23. Udacity: Artificial Intelligence for Robotics. Lesson 1: Localization. – Режим доступа:https://storage.googleapis.com/supplemental_media/udacityu/487393 81/Lesson1Notes.pdf. Дата доступа: 14.04.2016
24. Круш І.В. Застосування фільтру калмана для вирішення проблеми локалізації робота. / Михалько В.Г., Круш І.В. // // International Scientific Journal. – 2016. – #4. – С. 22-25.

# ДОДАТОК А

## ЛІСТИНГ ПРОГРАМИ

### Утиліти пошуку мереж та підключення до WiFi

*Лістинг файлу utilities/show-wifi-networks*

```
#!/usr/bin/ruby

puts `iwlist wlan0 scan | grep 'ESSID:'`
  .to_s
  .split('\n')
  .map { |essid| essid.delete('ESSID:') }
  .sort
  .join('\n')
```

*Лістинг файлу utilities/start-wifi*

```
#!/usr/bin/ruby

require 'fileutils'

WPA_SUPPLICANT_FILE_PATH = '/etc/wpa_supplicant/wpa_supplicant.conf'
WPA_SUPPLICANT_PREVIOUS_FILE_PATH =
'/etc/wpa_supplicant/wpa_supplicant.conf-previous'

def wpa_supplicant(ssid, password)
  <<-START
    ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
    update_config=1

    network={
      ssid="#{ssid}"
      psk="#{password}"
      key_mgmt=WPA-PSK
    }
  START
```

```
end

args = ARGV.reduce({ args: {}, key: nil }) do |acc, arg|
  if acc[:key] && !arg.start_with?('--')
    acc[:args][acc[:key]] = arg
    acc[:key] = nil
  elsif arg.start_with?('--')
    acc[:key] = arg.delete('--')
  else
    raise ArgumentError, 'wrong arguments list'
  end
  acc
end[:args]

ssid = args['ssid'].to_s.strip
password = args['password'].to_s.strip

if ssid.empty?
  puts 'ssid can\'t be blank'
  exit
end

FileUtils.mv(WPA_SUPPLICANT_FILE_PATH, WPA_SUPPLICANT_PREVIOUS_FILE_PATH)

File.new(WPA_SUPPLICANT_FILE_PATH, 'w').tap do |wpasfile|
  wpasfile.write(wpa_supplicant(ssid, password))
  wpasfile.close
end


`sudo ifdown wlan0 && sudo ifup wlan0`
```

## Сервер управління системою

*Лістинг файлу saprocar-control/src/main/java/org/dodiks/HttpServer.java*

```
package org.dodiks;
```

```
import org.apache.commons.io.FilenameUtils;

import javax.imageio.ImageIO;
import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.*;
import java.util.concurrent.atomic.AtomicLong;

import static spark.Spark.*;

public class HttpServer {

    private static final CarController car =
CarController.createInitialized();

    private static Timer timer = new Timer();

    private static Timer takingPictureTimer = new Timer();
    private static int takingPictureAmount = 20;
    private static int takingPictureInterval = 1000;

    private static Timer imageCleanerTimer = new Timer();

    private static AtomicLong lastRequestTimestamp = new AtomicLong(0);
    private static Path imagesPath;

    public static void main(String[] args) {
        try {
            imagesPath = Files.createTempDirectory("images");
        } catch(IOException e) {
            System.out.println("cant create temp directory");
        }

        int maxThreads = 8;
```

```
            int minThreads = 2;
            int timeOutMillis = 2000;
            threadPool(maxThreads, minThreads, timeOutMillis);

            takingPictureTimer.schedule(
                    new PhotosCapturer(imagesPath, takingPictureAmount,
takingPictureInterval),
                    1000,
                    takingPictureAmount * takingPictureInterval + 5000
            );
            imageCleanerTimer.schedule(new ImagesCleaner(imagesPath), 10000 *
1000, 60 * 1000);

            staticFileLocation("/public");

            get("/", (req, res) -> "Hello");

            get("/pwm/", (req, res) -> {
                int cycle = Integer.parseInt(req.queryParams("cycle"));
                car.setCycle(cycle);
                return "CYCLE WAS SUCCESSFULLY SET";
            });

            get("/api/control/", (req, res) -> {
                long timestamp = new Date().getTime();
                lastRequestTimestamp.set(timestamp);

                timer.schedule(new TimerTask() {
                    @Override
                    public void run() {
                        if (lastRequestTimestamp.get() == timestamp) {
                            car.stop();
                            car.steer(0);
                        }
                    }
                }, 10 * 1000);
```

```
String key = req.queryParams("key");
String status = req.queryParams("status");

boolean isPressed = status.equals("pressed");


switch (key) {
    case "up":
        if (isPressed) {
            car.moveForward();
        } else {
            car.stop();
        }
        break;
    case "down":
        if (isPressed) {
            car.moveBackward();
        } else {
            car.stop();
        }
        break;
    case "left":
        if (isPressed) {
            car.steer(-1);
        } else {
            car.steer(0);
        }
        break;
    case "right":
        if (isPressed) {
            car.steer(1);
        } else {
            car.steer(0);
        }
        break;
    default:
```

```
                    return "WRONG COMMAND!";
                }

                return "ALL RIGHT!";
            });

            get("/stream-image", (req, res) -> {
                File[] files = imagesPath.toFile().listFiles();

                if (files == null || files.length == 0) {
                    halt(405,"server error");
                }

                Arrays.sort(files, new Comparator<File>() {
                    public int compare(File f1, File f2) {
                        return Long.compare(f2.lastModified(),
f1.lastModified());
                    }
                });

                res.raw().setContentType("image/jpeg");
                String ext = FilenameUtils.getExtension(files[1].toString());
                System.out.println("returning image...");
                try (OutputStream out = res.raw().getOutputStream()) {
                    ImageIO.write(ImageIO.read(files[0]), ext, out);
                } catch (Exception e) {
                    System.out.println(e.getMessage());
                    halt(405,"server error");
                }

                return res;
            });
        }
    }
```

*Лістинг файлу saprocar-control/src/main/java/org/dodiks/CarController.java*

```
package org.dodiks;

import com.pi4j.io.gpio.*;

import static com.pi4j.wiringpi.SoftPwm.softPwmCreate;
import static com.pi4j.wiringpi.SoftPwm.softPwmWrite;


public class CarController {

    private final GpioController gpio;

    private final GpioPinDigitalOutput pin0;
    private final GpioPinDigitalOutput pin1;
    private final GpioPinDigitalOutput pin3;
    private final GpioPinDigitalOutput pin4;

    private CarController() {
        gpio = GpioFactory.getInstance();
        pin0 = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_00, PinState.LOW);
        pin1 = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01, PinState.LOW);

        pin3 = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_03, PinState.LOW);
        pin4 = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_04, PinState.LOW);

        // set shutdown state for this pin
        pin0.setShutdownOptions(false, PinState.LOW);
        pin1.setShutdownOptions(false, PinState.LOW);

        pin3.setShutdownOptions(false, PinState.LOW);
        pin4.setShutdownOptions(false, PinState.LOW);

        softPwmCreate(6, 10, 100);
    }

    public static CarController createInitialized() {
```

```
    return new CarController();
}

public void shutdown() {
    gpio.shutdown();
}

public void moveForward() {
    // right wheel
    pin0.low();
    pin1.high();

    // left wheel
    pin3.low();
    pin4.high();
}

public void moveBackward() {
    // right wheel
    pin0.high();
    pin1.low();

    // left wheel
    pin3.high();
    pin4.low();
}

public void stop() {
    // right wheel
    pin0.low();
    pin1.low();

    // left wheel
    pin3.low();
    pin4.low();
}
```

```
    public void steer(int direction) {
        if (direction == -1) {
            setCycle(7);
        } else if (direction == 1) {
            setCycle(12);
        } else {
            setCycle(9);
        }
    }

// should be private (public for debug)
    public void setCycle(int cycle) {
        softPwmWrite(6, cycle);
    }
}
```

*Лістинг файлу saprocar-control/src/main/java/org/dodiks/ImagesCleaner.java*

```
package org.dodiks;

import java.io.File;

import java.nio.file.Path;

import java.util.Arrays;

import java.util.Comparator;

import java.util.TimerTask;


/**
 * Created by ihorkroosh on 6/12/16.
 */
public class ImagesCleaner extends TimerTask {

    private Path path;

    private static final int imagesToLeaveCount = 10;
```

```
ImagesCleaner(Path path) {
    this.path = path;
}

@Override
public void run() {
    if(this.path == null) {
        return;
    }

    File f = new File(this.path.toString());
    File [] files = f.listFiles();

    if(files == null || files.length < 0) {
        return;
    }

    Arrays.sort(files, new Comparator<File>() {
        public int compare(File f1, File f2) {
            return Long.compare(f1.lastModified(), f2.lastModified());
        }
    });
    for (int i = 0; i < files.length - imagesToLeaveCount; ++i) {
        try {
            files[i].delete();
```

```
            } catch (Exception x) {
                // File permission problems are caught here.
                System.err.println(x);
            }
        }
    }
}
```

*Лістинг файлу saprocar-control/src/main/java/org/dodiks/PhotosCapturer.java*

```
package org.dodiks;


import java.nio.file.Path;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimerTask;


/**
 * Created by ihorkroosh on 6/12/16.
 */
public class PhotosCapturer extends TimerTask {
    private Path imagesPath;
    private int takingPictureAmount;
    private int takingPictureInterval;
    private final RaspiStill camera = new RaspiStill();


    PhotosCapturer(Path imagesPath, int takingPictureAmount, int
takingPictureInterval) {
```

```
        this.imagesPath = imagesPath;
        this.takingPictureAmount = takingPictureAmount;
        this.takingPictureInterval = takingPictureInterval;
    }

    @Override
    public void run() {
        Date today;
        String output;
        SimpleDateFormat formatter;
        formatter = new SimpleDateFormat("H_mm_ss_SSS");
        today = new Date();

        if (imagesPath != null) {
            System.out.println("Start(" + formatter.format(today) + "): " + (new
Date()).toString());
            camera.TakeNPicturesWithInterval(
                    imagesPath.toString() + "/" + formatter.format(today) +
"myimage_%04d.jpg",
                    takingPictureAmount,
                    takingPictureInterval,
                    800,
                    600
            );
            System.out.println("Finish("+ formatter.format(today) +"): " + (new
Date()).toString());
        }
```

```
    }
}
```

*Лістинг файлу saprocar-control/src/main/java/org/dodiks/RaspiStill.java*

```
package org.dodiks;
public class RaspiStill {
    // Define the path to the raspistill executable.
    private final String _raspistillPath = "/opt/vc/bin/raspistill";
    // Define the amount of time that the camera will use to take a photo.
    private int _picTimeout = 500;
    private long _picN = 1;
    // Define the image quality.
    private final int _picQuality = 100;

    // Specify a default image width.
    private int _picWidth = 1024;
    // Specify a default image height.
    private int _picHeight = 768;
    // Specify a default image name.
    private String _picName = "example.jpg";
    // Specify a default image encoding.
    private String _picType = "jpg";

    // Default class constructor.
    public void RaspiStill()
    {
        // Do anything else here. For example, you could create another
```

```
        // constructor which accepts an alternate path to raspistill,
        // or defines global parameters like the image quality.
    }

    // Default method to take a photo using the private values for
name/width/height.
    // Note: See the overloaded methods to override the private values.
    public void TakePicture()
    {
        try
        {
            // Determine the image type based on the file extension (or use the
default).
            if (_picName.indexOf('.')!=-1) _picType =
_picName.substring(_picName.lastIndexOf('.')+1);

            // Create a new string builder with the path to raspistill.
            StringBuilder sb = new StringBuilder(_raspistillPath);

            // Add parameters for no preview and burst mode.
            sb.append(" -n -bm");
            // configure amount of photo per _picTimeout
            sb.append(" -tl " + _picTimeout);
            sb.append(" -t " + _picN * _picTimeout);
            // Configure the picture width.
            sb.append(" -w " + _picWidth);
            // Configure the picture height.
```

```
            sb.append(" -h " + _picHeight);
            // Configure the picture quality.
            sb.append(" -q " + _picQuality);
            // Specify the image type.
            sb.append(" -e " + _picType);
            // Specify the name of the image.
            sb.append(" -o " + _picName);
            System.out.println(sb.toString());
            // Invoke raspistill to take the photo.
            Runtime.getRuntime().exec(sb.toString());
            // Pause to allow the camera time to take the photo.
            Thread.sleep(_picTimeout * _picN);
        }
        catch (Exception e)
        {
            // Exit the application with the exception's hash code.
            System.exit(e.hashCode());
        }
    }

    // Overloaded method to take a photo using specific values for the
name/width/height.
    public void TakePicture(String name, int width, int height)
    {
        _picName = name;
        _picWidth = width;
```

```
        _picHeight = height;
        TakePicture();
    }

    public void TakeNPicturesWithInterval(String pattern, long n, int interval,
int width, int height) {
        _picName = pattern;
        _picWidth = width;
        _picHeight = height;
        _picN = n;
        _picTimeout = interval;
        TakePicture();
    }
    // Overloaded method to take a photo using a specific value for the image
name.
    public void TakePicture(String name)
    {
        TakePicture(name, _picWidth, _picHeight);
    }

    // Overloaded method to take a photo using specific values for width/height.
    public void TakePicture(int width, int height)
    {
        TakePicture(_picName, width, height);
    }
}
```

## Сервер розпізнавання та локалізації

*Лістинг файлу saprocar-recognition/src/main/java/org/dodiks/Main.java*

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import org.opencv.core.Core;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root =
FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("Hello World");
        primaryStage.setScene(new Scene(root, 800, 400));
        primaryStage.show();
    }

    public static void main(String[] args) {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        launch(args);
    }
}
```

*Лістинг файлу saprocar-recognition/src/main/java/org/dodiks/Controller.java*

```
package org.dodiks;

import javafx.application.Platform;
import javafx.beans.property.ObjectProperty;
import javafx.beans.property.SimpleObjectProperty;
import javafx.event.ActionEvent;
```

```
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import org.opencv.core.*;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.videoio.VideoCapture;

import java.io.ByteArrayInputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

public class Controller {
    @FXML
    private Button cameraBtn;

    @FXML
    private ImageView realImage;

    @FXML
    private ImageView maskImage;

    @FXML
    private ImageView morphImage;

    @FXML
    private Slider hueStart;

    @FXML
    private Slider hueStop;
```

```
@FXML
private Slider saturationStart;

@FXML
private Slider saturationStop;

@FXML
private Slider valueStart;

@FXML
private Slider valueStop;

@FXML
private Label hsvCurrentValues;


// property for object binding
private ObjectProperty<String> hsvValuesProp;

// a timer for acquiring the video stream
private ScheduledExecutorService timer;
// the OpenCV object that performs the video capture
private VideoCapture capture = new VideoCapture();
// a flag to change the button behavior
private boolean cameraActive;

@FXML
private void startCamera(ActionEvent event) {
    hsvValuesProp = new SimpleObjectProperty<>();
    this.hsvCurrentValues.textProperty().bind(hsvValuesProp);

    this.imageViewProperties(this.realImage, 400);
    this.imageViewProperties(this.maskImage, 200);
    this.imageViewProperties(this.morphImage, 200);
```

```
            if(!this.cameraActive) {
                this.capture.open(0);

                if(this.capture.isOpened()) {
                    this.cameraActive = true;

                    Runnable frameGrabber = new Runnable() {
                        @Override
                        public void run() {
                            Image imageToShow = grabFrame();
                            realImage.setImage(imageToShow);
                        }
                    };

                    this.timer = Executors.newSingleThreadScheduledExecutor();
                    this.timer.scheduleAtFixedRate(frameGrabber, 0, 33,
TimeUnit.MILLISECONDS);
                    this.cameraBtn.setText("Stop Camera");
                } else {
                    // log the error
                    System.err.println("Failed to open the camera
connection...");
                }
            } else {
                // the camera is not active at this point
                this.cameraActive = false;
                // update again the button content
                this.cameraBtn.setText("Start Camera");

                // stop the timer
                try
                {
                    this.timer.shutdown();
                    this.timer.awaitTermination(33, TimeUnit.MILLISECONDS);
                }
                catch (InterruptedException e)
                {
```

```
                    // log the exception
                    System.err.println("Exception in stopping the frame
capture, trying to release the camera now... " + e);
                }

                // release the camera
                this.capture.release();
            }
        }

        private void imageViewProperties(ImageView image, int dimension)
        {
            // set a fixed width for the given ImageView
            image.setFitWidth(dimension);
            // preserve the image ratio
            image.setPreserveRatio(true);
        }

        private Image grabFrame() {
            Image result = null;
            Mat frame = new Mat();

            if(this.capture.isOpened()) {
                try {
                    this.capture.read(frame);

                    if(!frame.empty()) {
                        Mat blurredImage = new Mat();
                        Mat hsvImage = new Mat();
                        Mat mask = new Mat();
                        Mat morph = new Mat();
                        Mat rgbImage = new Mat();

                        Imgproc.blur(frame, blurredImage, new Size(7, 7));
      //                    Imgproc.cvtColor(frame, hsvImage,
Imgproc.COLOR_BGR2HSV);
```

```
                        // get thresholding values from the UI
                        // remember: H ranges 0-180, S and V range 0-255
                        Scalar minValues = new
Scalar(this.hueStart.getValue(), this.saturationStart.getValue(),
                                this.valueStart.getValue());
                        Scalar maxValues = new Scalar(this.hueStop.getValue(),
this.saturationStop.getValue(),
                                this.valueStop.getValue());

                        // show the current selected HSV range
                        String valuesToPrint = "Hue range: " +
minValues.val[0] + "-" + maxValues.val[0]
                                + "\tSaturation range: " + minValues.val[1] +
"-" + maxValues.val[1] + "\tValue range: "
                                + minValues.val[2] + "-" + maxValues.val[2];
                        this.onFXThread(this.hsvValuesProp, valuesToPrint);

                        Core.inRange(frame, minValues, maxValues, mask);
                        this.onFXThread(this.maskImage.imageProperty(),
this.mat2Image(mask));

                        Mat dilateElement =
Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(24, 24));
                        Mat erodeElement =
Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(12, 12));

                        Imgproc.erode(mask, morph, erodeElement);
                        Imgproc.erode(mask, morph, erodeElement);

                        Imgproc.dilate(mask, morph, dilateElement);
                        Imgproc.dilate(mask, morph, dilateElement);

                        this.onFXThread(this.morphImage.imageProperty(),
this.mat2Image(morph));

                        frame = this.findAndDrawMarkers(morph, frame);

                        result = this.mat2Image(frame);
                    }
```

```
                } catch(Exception e) {
                    // log the (full) error
                    System.err.print("ERROR");
                    e.printStackTrace();
                }
            }

            return result;
        }

        private Image mat2Image(Mat frame)
        {
            // create a temporary buffer
            MatOfByte buffer = new MatOfByte();
            // encode the frame in the buffer, according to the PNG format
            Imgcodecs.imencode(".png", frame, buffer);
            // build and return an Image created from the image encoded in the
            // buffer
            return new Image(new ByteArrayInputStream(buffer.toArray()));
        }

        private Mat findAndDrawMarkers(Mat maskedImage, Mat frame)
        {
            // init
            List<MatOfPoint> contours = new ArrayList<>();
            Mat hierarchy = new Mat();

            // find contours
            Imgproc.findContours(maskedImage, contours, hierarchy,
Imgproc.RETR_CCOMP, Imgproc.CHAIN_APPROX_SIMPLE);

            // if any contour exist...
            if (hierarchy.size().height > 0 && hierarchy.size().width > 0)
            {
                // for each contour, display it in blue
                for (int idx = 0; idx >= 0; idx = (int) hierarchy.get(0,
idx)[0])
```

```
                {
                    Imgproc.drawContours(frame, contours, idx, new Scalar(250,
0, 0));
                }
            }

            return frame;
        }

        private <T> void onFXThread(final ObjectProperty<T> property, final T
value)
        {
            Platform.runLater(new Runnable() {
                @Override
                public void run() {
                    property.set(value);
                }
            });
        }
    }
```