Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Медведєв Максим Ігорович

студент групи АС-151

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

Програмна система конструювання карти приміщення за допомогою алгоритму SLAM

Спеціальність:  
121 – Інженерія програмного забезпечення

Спеціалізація:  
Інженерія програмного забезпечення

Керівник:

Тройніна Анастасія Сергіївна,

канд. техн. наук, доцент

Одеса – 2020

Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_ Крісілов В. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

**ЗАВДАННЯ  
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Медведєва Максима Ігоровича, група АС-151

1. Тема роботи: “Програмна система конструювання карти приміщення за допомогою алгоритму SLAM”  
Керівник роботи:Тройніна Анастасія Сергіївна, канд. техн. наук, доцент  
затверджені наказом ректора від «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р. No \_\_\_\_\_\_\_

2. Зміст роботи: аналіз вимог до програмного продукту, план виконання проекту, проектування програмної системи, програмна реалізація системи, тестування програмної системи, розгортання програмного продукту, охорона праці

3. Перелік графічного матеріалу: Згідно до слайдів презентації

4. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, Дата | |
| завдання видав | завдання  прийняв |
| 7 | доц. Москалюк А.Ю. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

5. Дата видачі завдання\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів дипломного  проекту (роботи) | Строк виконання етапів проекту( роботи ) | Примітка |
| 1 | Вимоги до програмної системи | 31.08.20 – 29.09.20 | виконав |
| 2 | Планування розробки програмної системи | 30.09.20 – 06.10.20 | виконав |
| 3 | Проектування веб-системи | 07.10.20 – 30.10.20 | виконав |
| 4 | Програмна реалізація | 02.11.20 – 20.11.20 | виконав |
| 5 | Тестування системи | 23.11.20 – 30.11.20 | виконав |
| 6 | Охорона праці | 01.12.20 – 7.12.20 | виконав |

**Здобувач вищої освіти \_\_\_\_\_\_\_\_\_\_\_\_\_ Медведєв М.І.**

( підпис )

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ Тройніна А.С.**

( підпис )

ЗМІСТ

[ВСТУП 7](#_Toc58835103)

[1. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ПРОДУКТУ 9](#_Toc58835104)

[1.1 Опис предметної області і аналогів системи 9](#_Toc58835105)

[1.2 Існуючі підходи до створення карт місцевості. 9](#_Toc58835106)

[1.3 Алгоритм для створення карт місцевості. 10](#_Toc58835107)

[1.4 Проблеми, що виникають при створенні карт місцевості за допомогою дронів…... 11](#_Toc58835108)

[2. ОСНОВНІ МЕТОДИ ПОБУДОВИ І РОЗПІЗНАВАННЯ ОПИСУ МАРШРУ. 15](#_Toc58835110)

[2.1 Розширений фільтр Калмана 15](#_Toc58835111)

[2.2 Фільтр частиць 16](#_Toc58835112)

[2.3 Алгоритм SLAM на основі особливостей-міток 17](#_Toc58835113)

[3 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ СТВОРЕННЯ КАРТ МІСЦЕВОСТІ 22](#_Toc58835115)

[3.1 Загальні характеристики 22](#_Toc58835116)

[3.1.1 Очікувані результати роботи 22](#_Toc58835117)

[3.2 Функціональні вимоги 22](#_Toc58835118)

[3.3 Нефункціональні вимоги 27](#_Toc58835119)

[3.4 Оцінка тривалості розробки програмного проекту 29](#_Toc58835120)

[3.5 План розробки 33](#_Toc58835121)

[3.6 Аналіз ризиків 35](#_Toc58835122)

[4. ОПИС ПРОЕКТУВАННЯ СИСТЕМИ СТВОРЕННЯ КАРТ МІСЦЕВОСТІ. 40](#_Toc58835124)

[4.1 Програмне проектування системи 40](#_Toc58835125)

[4.1.1 Опис структур даних 41](#_Toc58835126)

[4.1.2 Структура програмних класів 42](#_Toc58835127)

[4.1.3. Зміст програмних класів та опис алгоритмів 44](#_Toc58835128)

[4.2 Проектування інтерфейсу користувача 48](#_Toc58835129)

[5 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУВАННЯ ПЛАНУВАННЯ ЗАХОДІВ ПІД ЧАС ПОДОРОЖІ 53](#_Toc58835131)

[5.1 Набір інструментальних засобів розробки 53](#_Toc58835132)

[5.2 Інструкція розгортання 54](#_Toc58835133)

[6 ТЕСТУВАННЯ СИСТЕМИ 56](#_Toc58835134)

[6.1 Функціональне тестування 56](#_Toc58835135)

[6.2 Модульне тестування 58](#_Toc58835136)

[6.3 Експеримент 59](#_Toc58835137)

[7. ОХОРОНА ПРАЦІ 66](#_Toc58835138)

[ВИСНОВКИ 68](#_Toc58835142)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 69](#_Toc58835143)

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ……………………………….…………….. 71

ДОДАТОК Б…………………………………………………………………………91

**АНОТАЦІЯ**

Пояснювальна записка: 70 с., 17 рисунків, 14 таблиць, 19 джерел, 2 додатки.

Метою роботи є скорочення часу на створення карти кімнати за допомогою алгоритму SLAM та камери Kinect360.

Інструментами розробки є мова програмування Java, камера Kinect360. Операційна система - Linux Ubuntu та роботизована операційна система ROS.

В результаті роботи було створено програмну систему для створення карти кімнати за допомогою вищезазначених інструментів.

Ключові слова: SLAM, Kinect360, Java, room map

**ANNOTATION**

Explanatory note to the thesis: 70 p., 17 figures, 14 tables, 19 sources, 1 supplement.

The aim of the work is to reduce the time to create a room map using the SLAM algorithm and the Kinect360camera.

Development tools are Java programming language, Kinect360 camera. The operating system is Linux Ubuntu, and robotic ROS operating system.

As a result of the work, program system was created to create a map of the room using the above tools.

Keywords: SLAM, Kinect360, Java, room map

# ВСТУП

На сьогодні у галузі робото конструювання, є попит на алгоритми та прилади, які допомагають роботу, або дрону діяти самостійно, визначати своє положення на місцевості, та створювати план навколишнього середовища, та разом з цим передавати це положення оператору. Прикладом таких роботів є: марсоходи, пожежні дрони. Часто при цьому використовують існуючі карти місцевості, наприклад якщо це пожежний робот, і його запускають у приміщення.

Актуальність проблеми пов'язана з тим, що карти, які зазвичай використовуються для навігації, в основному відображають вид простору, зафіксований в момент їх побудови, і зовсім не обов'язково, що вид простору буде тим же в момент використання карт. При цьому складність технічного процесу визначення поточного місцезнаходження з одночасною побудовою точної карти обумовлена ​​низькою точністю приладів, які беруть участь в процесі обчислення поточного місця розташування. До того ж більшість сучасних комп’ютерних систем не можуть працювати в умовах відсутності освітлення, що робить їх використання при обслідуванні наприклад печер, або просто темних приміщень неможливим. У моєму підході я використовую метод одночасної навігації і побудови карти (SLAM), разом з камерою Kinect яка моє інфрачервоний проектор, та може давати картинку навіть за умов повної відсутності освітлення, і пов'язую два незалежних процеса в безперервний цикл послідовних обчислень, при цьому результати одного процесу беруть участь в обчисленнях іншого процесу.

Метою даної роботи є зменшення часу на створення карти приміщення за допомогою алгоритмів SLAM та камери Kinect360.

Для досягнення цієї мети потрібно розв’язати наступні задачі:

* Проаналізувати предметну область.
* Зробити аналіз існуючих рішень та аналогів системи.
* Визначити функціональні та нефункціональні вимоги.
* Виконати план розробки та аналіз ризиків.
* Спроектувати робочу модель.
* Провести експеримент з визначенням швидкості роботи.

Об’єкт дослідження: процес створення карти та орієнтації на місцевості.

Предмет дослідження: система одночасної навігації і побудови карти на базі камери Kinect360.

Гіпотеза дослідження: Передбачається, що використання алгоритму SLAM та камери Kinect360 у зрівнянні з іншими засобами картографування може прискорити процес створення карти місцевості.

У першому розділі наведено аналоги даної системи, та актуальність проблеми. У другому розділі описано використаний мат апарат, загальна структура системи та технічне завдання. У третьому розділі описано проектування системи, планування розробки продукту за допомогою методу UCP, структури декомпозиції робіт, діаграми Ганта, проведено аналіз ризиків за допомогою діаграми Ішикави, описано керівництво користувача по встановленню та використанню системи.

# 1. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ПРОДУКТУ

# 1.1 Опис предметної області і аналогів системи

Предметна область – процес одночасного конструювання карти та орієнтації на місцевості . Дана програмна система, представляє собою реалізацію алгоритму SLAM пазом з камерою Kinect360, яку можна використовувати для розвідки оточуючого простору та знаходження свого місцеположення у ньому.

Системою надається можливість створення 3D карти приміщення, як при наявності освітлення, так і при його відсутності.

В режимі конструювання карті користувач, або дрон, рухаючись по приміщенню повертають камеру, яка в свою чергу знімаючи оточуючий простір створює його 3D карту. Назва програми - SLAMKINECT походить від двох технологій, які у ній використовуються найбільше.

# 1.2 Існуючі підходи до створення карт місцевості.

Аерофотозйомка - фотографування території з певної висоти від поверхні Землі за допомогою аерофотоапарата, встановленого на атмосферному літальному апараті (літаку, вертольоті, дирижаблі та ін. Або їх безпілотний аналогу) з метою отримання, вивчення і наданням об'єктивних просторових даних на ділянках виробленої зйомки [5]. Але у цього метода є такі недоліки як: неможливість створення внутрішніх карт приміщень, або інших місцевостей, які не проглядаються з повітря, та досить велика ціна на повітряні БПЛА.

Дослідження за території за допомогою людей - картографів, які створюють карту місцевості. До переваг такого методу можна високу деталізацію карт, і можливість реагування на зміну обставин, але головним недоліком такого методу дослідження є його низька швидкість, та великі затрати коштів.

Дослідження за допомогою наземних дронів, у тих місцях де неможливо використати наведені віще способи створення карт. Наземні дрони завдяки невеликим розмірам можуть досліджувати недоступні для людей місця, а також створювати більш детальні карти завдяки своїй меншій швидкості аніж летючі та можливості зупинитись де це потрібно, але недоліком такого методу дослідження є також його низька швидкість, найкраще він підходить для дослідження невеликих або важкодоступних територій, куди не можна пустити летючого дрона, або команду людей, такі, як пошкодженні приміщення, або печери.

# 1.3 Алгоритм для створення карт місцевості.

Створення колажу з фото, та з’єднання фото з прив’язкою до GPS навігації. Цей метод зазвичай використовується при фотографуванні території з певної висоти від поверхні Землі за допомогою аерофотоапарата, встановленого на літальному апараті, або камери встановленій на наземному дроні. Прикладом карт створених за допомогою такого метода є Google Maps.

Загальна послідовність роботи алгоритму:

1. Виправлення спотворень, викликане оптикою.

2. Компресія зображень в разі надвисоких дозволів для прискорення оптимізації.

3. Знаходження сполучних точок на зображеннях.

4. Аналіз і зіставлення отриманих на попередньому етапі точок, створення пар точок, які збігаються між собою.

5. Використання початкової інформації про місцезнаходження (дані GPS) і кути зйомки для попередньої орієнтації зображень і визначення початкових параметрів для проведення більш швидкого і надійного способу оптимізації.

6. Розрахунок мінімально необхідних параметрів трансформації для кожної пари точок для досягнення заданого ступеня похибки кінцевої мозаїки за допомогою ітеративного методу градієнтного спуску.

7. Застосування отриманих параметрів по відношенню до точок і остаточне формування єдиної мозаїки.

8. Вирівнювання яскравості зображень і згладжування швів.

Недоліками такого алгоритму є необхідність у великому об’ємі даних та значних затратах ресурсів на їх обчислення. Адже зазвичай камери знімають з частотою приблизно 30 – 60 кадрів на секунду. Що робить задачу одночасного конструювання та відображення карти складною, або неможливою. Оскільки швидко обробити такий об’єм інформації можна лише на потужному комп’ютері.

## **1.4 Проблеми, що виникають при створенні карт місцевості за допомогою дронів**

При взаємодії суб’єкту автоматизації з об’єктом можуть виникати наступні проблеми:

Проблема 1. Не всі можуть дозволити собі дорогий камеру яка дозволяє створювати 3д карти.

Проблема 2. При дослідженні складних місцевостей, такі, як печери, потрібно потім якось повернутися з обладнанням, що може викликати труднощі, якщо немає маршруту слідування.

Проблема 3. Більшість дронів та 3д камер не працюють якщо відсутнє освітлення.

Отже, для того, щоб програмний продукт був реалізований відповідно до вимог до системи, необхідно визначити вимоги до цього програмного продукту:

1. Як найбільше зменшити ціну для мінімального набору, що дасть можливість працювати з повним функціоналом.
2. Дати можливість працювати в нічних умовах, та при повній відсутності освітлення.
3. Дати можливість змінювати платформи не втрачаючи функціонал
4. Підвисити ефективність алгоритмів конструювання 3д карт.

**1.5 Опис аналогів системи.**

В описі аналогів системи, ми будемо розглядати аналоги, котрі мають функціонал для конструювання карт місцевості.

ROS - це екосистема для програмування роботів, що надає функціональність для розподіленої роботи. ROS був спочатку розроблений в 2007 році під назвою switchyard в Лабораторії Штучного Інтелекту Стенфордського Університету для проекту[19]. У собі система має декілька реалізацій алгоритму SLAM на графах, які використовуються для створення карт місцевості.

DroneDeploy. Розглянемо перший аналог нашої системи. DroneDeploy являє собою проект призначенням якого є 3д моделювання за допомогою дронів. Має простий та зрозумілий інтерфейс, та може бути встановлений на майже будь який дрон. Мобільний додаток може працювати в офлайн режимі з попередньо збереженим маршрутом і картами Google Maps. Залежно від необхідного дозволу фотоплана в додатку потрібно встановити висоту польоту дрона. Чим вище дозвіл, тим більше фотографій потрібно зробити, і може вийде так, що для великої території не вистачить одного акумулятора. Додаток дозволяє продовжити політ за маршрутом з перерваного місця після зміни акумулятора або виключення дрона [6]. Для поліпшення якості і точності планів на фотографується майданчику можна встановити на заздалегідь відомих точні координати кілька яскравих візуальних маркерів (наприклад пластикові дорожні конуси). Потім при обробці даних можна вказати точні координати цих маркерів. Також підтримується RTK GPS - (точність 1-2 см). Його мінусом є те, що весь функціонал працює лише з летючими дронами.

Pix4D Mapper. Це проект який дозволяє користувачу створювати 3D карти та моделі за допомогою 2D карт, з прив’язкою до місцевості та географічним особливостям. Програмне забезпечення Pix4D може застосовуватися для оцінки обсягів земляних робіт, створення NDVI-карт для точного землеробства і видобутку корисних копалин. Обробка в Pix4D знаходить застосування і для виявлення змін ландшафту, а також для ліквідації наслідків надзвичайних ситуацій [6]. Сфери застосування ПО Pix4D - це будівництво, кадастр, контроль над станом навколишнього середовища, землеробство, аерофотограмметрія, нерухомість. ПО Pix4Dmapper автоматично перетворює зображення, отримані з БПЛА або зображення наземної зйомки, і забезпечує складання високоточних карт і 3D-моделей з географічною прив'язкою. Але треба зауважити, що він не може працювати у реальному часі.

3DZephyr. Це система, що може працювати з усіма камерами та дронами, створювати 3D конструкції та 3D карти, але функція конструювання маршруту відсутня. Хмарний сервіс і додаток для створення ортофотопланів та 3D моделей місцевості. Додаток Map Pilot For DJI дозволяє планувати маршрут і робити фотографії. Фотографії можна завантажити на хмарний сервер для обробки. Ортофотоплан до 104 зображень можна обробити безкоштовно. Для завантаження фото на сервер необхідно вийняти карту пам'яті з дрона і завантажити фото через комп'ютер[6].

Проаналізуємо, як існуючі програмні продукти вирішують вище перелічені проблеми. На таблиці 1.1 представлений порівняльний аналіз аналогів системи у вирішенні указаних нижче проблем.

Таблиця 1.1 – Порівняльний аналіз програмних продуктів у вирішенні зазначених проблем

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва програмного  продукту  Поставлена  проблема | DroneDeploy | Pix4D | ROS | 3D  Zephyr | SLAMKINECT |
| Створення 3D карти оточуючого простору. | + | + | + | + | + |
| Багатоплатформеність | + | - | - | - | - |
| Створення маршруту руху по приміщенню | - | - | + | - | + |
| Можливість роботи при поганому освітлення | - | + | - | + | + |

Проаналізувавши таблицю, можна зробити висновок, що програмна система SLAMKINECT має деякі переваги порівняно з її аналогами. Це може посприяти гарному попиту на ринку. Особливо це стосується швидкості обробки та конструювання карт, у всіх переведених аналогах цей процес займає декілька годин.

# Висновки

У першому розділі опис предметної області, розглянуто існуючі підходи до створення карт місцевості, наведено опис алгоритму для створення карт та проблем, що виникають при їх створенні, наведено опис аналогів системи та проведений їх порівняльний аналіз.

# ОСНОВНІ МЕТОДИ ПОБУДОВИ І РОЗПІЗНАВАННЯ ОПИСУ МАРШРУТУ

Проведений вище аналіз систем для створення карт місцевості дозволяє виділити два 3 існуючі методи створення карт.

# 2.1 Розширений фільтр Калмана

Ефективний рекурсивний фільтр, що оцінює вектор стану динамічної системи, використовуючи ряд неповних і зашумлених вимірювань.

Фільтр Калмана широко використовується в інженерних і економетричних додатках: від радарів і систем технічного зору до оцінок параметрів макроекономічних моделей. Калмановської фільтрація є важливою частиною теорії управління, грає велику роль в створенні систем управління. Спільно з лінійно-квадратичних регулятором фільтр Калмана дозволяє вирішити задачу лінійно-квадратичного гаусівського управління. Фільтр Калмана і лінійно-квадратичний регулятор - можливе рішення більшості фундаментальних завдань в теорії управління.

У більшості додатків розмірність вектору стану об'єкта перевершує розмірність вектору даних спостереження. І при цьому фільтр Калмана дозволяє оцінювати повний внутрішній стан об'єкта.

Фільтр Калмана призначений для рекурсивного до оцінювання вектору стану апріорно відомої динамічної системи, тобто для розрахунку поточного стану системи необхідно знати поточне вимірювання, а також попередній стан самого фільтра. Таким чином, фільтр Калмана, подібно до інших рекурсивним фільтрам, реалізований в тимчасовому, а не в частотному поданні, але на відміну від інших подібних фільтрів, фільтр Калмана оперує не тільки оцінками стану, а ще й оцінками невизначеності (щільності розподілу) вектору стану, спираючись на формулу Байеса умовної ймовірності.

Алгоритм працює в два етапи. На етапі прогнозування фільтр Калмана екстраполюють значення змінних стану, а також їх невизначеності. На другому етапі за даними вимірювання (отриманого з деякою погрішністю) результат екстраполяції уточняється. Завдяки покрокової природі алгоритму, він може в реальному часі відстежувати стан об'єкта (без заглядання вперед, використовуючи тільки поточні виміри і інформацію про попередньому стані і його невизначеності) [16].

# Фільтр частиць

Фільтр частинок є одним з найпопулярніших методів оптимальної фільтрації.

Фільтр частинок дозволяє отримати оцінку (наближене значення) параметрів системи або об'єкта (позначимо їх як параметри А), які не можна виміряти безпосередньо. Для побудови цієї оцінки фільтр використовує вимірювання інших параметрів (параметри Б) пов'язаних з першими.

Для оцінки параметрів А фільтр створює безліч гіпотез (часток) про поточне значення цих параметрів. У початковий момент часу ці гіпотези абсолютно випадкові, але на кожній ітерації циклу фільтрації фільтр буде прибирати гіпотези, які не пройде перевірку достовірності, засновану на вимірах параметрів.

Таким чином з безлічі гіпотез зрештою залишаться тільки ті, які найбільш близькі до істинного значення параметрів А.

Фільтри частинок в даний час широко використовуються при оцінці моделей для фінансових ринків, зокрема для моделей стохастичною волатильності. Методи фільтрації частинок - це рекурсивні байєсовські фільтри, які забезпечують зручний і привабливий підхід для апроксимації апостеріорного розподілів, коли модель нелінійна і коли шуми не гаусові. Ці методи забезпечують спільні рішення багатьох проблем, де лінеаризація і гаусові наближення важко вирішуваними або призводять до занадто низьким характеристикам. Допущення про негаусівських шумі і включення обмежень на змінні стану також можуть виконуватися природним чином. Крім того, методи фільтрації частинок дуже гнучкі, прості в реалізації, виконуються паралельно і застосовні в найзагальніших умовах. Хоча алгоритм фільтра частинок можна використовувати, як ефективний засіб для вирішення проблеми SLAM, в алгоритмі все ще є деякі проблеми. Основна проблема полягає в тому, що для точного наближення апостеріорної щільності ймовірності системи потрібна велика кількість вибірок. Чим складніше оточення, з яким стикається робот, тим більше зразків потрібно для опису апостеріорного розподілу ймовірностей і тим складніше алгоритм. Тому в центрі уваги алгоритму знаходиться стратегія адаптивної вибірки, яка може ефективно зменшити кількість вибірок. Крім того, фаза повторної вибірки може привести до втрати достовірності і різноманітності вибірки, що призведе до виснаження вибірки. Спосіб підтримки достовірності і різноманітності частинок і подолання виснаження вибірок також повторно аналізує цей алгоритм. У навчанні з підкріпленням фільтр частинок, як учасник глобальних методів пошуку, як правило, вимагає більшої кількості випробувань, щоб зійтися, бо область пошуку є найбільшою з можливих - всього простору політики [17].

# 2.3 Алгоритм SLAM на основі особливостей-міток

Задача алгоритму SLAM стоїть у тому щоб обчислити місцеположення агенту і карти оточуючого простору використовуючи ряд спостережень за деякий час. При цьому усі величини є імовірнісними. Мета задачі стоїть у тому щоб обчислити [7].

Використовуючи правила Байеса та зважаючи на функцію переходу

Де.

Отже для послідовного оновлення карти використовується наступний алгоритм[7].

SLAM можна описати, як послідовність кроків, які повторюються:

1) сканування навколишнього простору. Аналіз і інтеграція нових даних. На цьому етапі проводиться виділення особливостей на знову отриманих даних (на основі RGB-інформації, карти глибини.) [8].

2) визначення зміщення на основі порівняння поточного кадру з попереднім, можна визначити, як змінилося розташування агенту на сцені. Так, як особливості самі по собі стаціонарні – очевидно, що це зрушення - результат зміни положення камери (робота). На основі цієї інформації можна виразити координати камери через систему лінійних рівнянь для вирішення якої використовуються різні методи і їх комбінації.

3) виділення на поточному кадрі особливостей-міток. Особливості - це такі характеристики, які легко можуть бути виділені в середовищі і використовуватися в подальшому для орієнтації в просторі. Бажано щоб їх можна було розпізнати під різним "кутом зору" з точки зору використовуваних сенсорів. Критично щоб вони були стаціонарні. Бажано щоб їх можна було розрізняти один від одного - для того щоб зрозуміти - чи не зустрічається ми їх раніше. Найпростіший приклад особливостей - геометричні - кути, прямі [8].

4) зіставлення міток поточного кадру з мітками отриманими за всю історію спостережень

5) оновлення на основі цієї інформації положення робота за всю історію спостережень, де кожне стан являє собою глобальне положення робота і взаємо-розташування розглянутих особливостей на певному проміжку часу. Останнім часом крім безпосередньо даних про становище, зберігаються також і ймовірні оцінки для кожного значення. На даному етапі проводиться аналіз і додавання отриманих раніше даних в загальну структуру для зберігання інформації про світ за весь час дослідження. Як правило, дані додаються нема на кожному кроці а за умови, що позиції робота (камери) істотно змінилася - наприклад, зсув або поворот в порівнянні з попередньою збереженою позицією склав більше якогось граничного значення[8].

6) перевірка на петлі – чи не проходимо ми повторно по одній і тій самій місцевості

Другою частиною нашої системи є камера Kinect360 яка відстежує рухи або може замірювати дистанцію до об’єктів за допомогою інфрачервоного проектора і камери з сенсором глибини.

Використовуючи ці можливості разом з алгоритмом SLAM ми робимо наступне:

1. на першому кроці робимо серію з трьох знімків оточуючого простору. Так, щоб кожен наступний знімок починався з кінця попереднього
2. за допомогою камери глибини створюємо карту кутів місцевості – це наші орієнтири. Виділення кутів та створення карти глибини виконується за алгоритмом. Наведеним на рисунку 2.1
3. визначивши серію таких орієнтирів на кожному фото ми ставимо зберігаємо знімок з координатами, які дорівнюють середній арифметичній дистанції між усіма орієнтирами на фото, але якщо дистанція до якогось орієнтира більше одного метру то для такого орієнтира робиться окреме фото, яке зберігається саме за його координатами, та обрізається по розмірам орієнтира + 10 см зліва та справа
4. після зіставлення усієї серії фото процес повторюється з місця установки останнього орієнтира

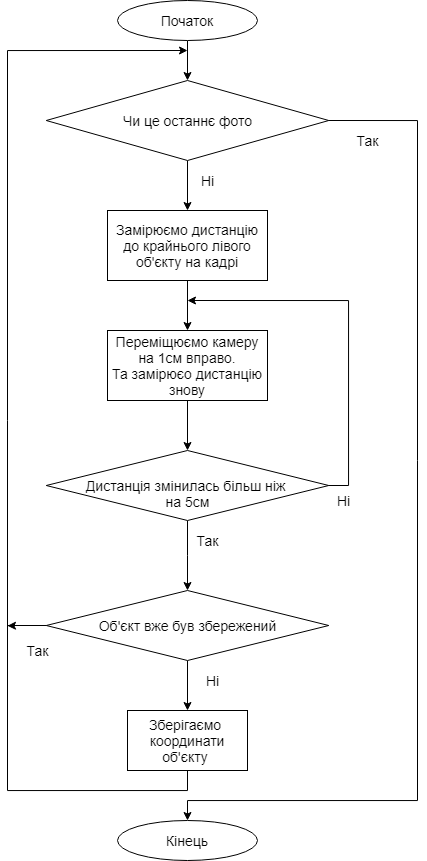


Рисунок 2.1 – Алгоритм пошуку кутів на серії фото

Наявність камери глибини відрізняє наш спосіб створення карти від картографування за допомогою дронів. Де зазвичай, карти не відображають дистанцію між об’єктами. А якщо і відображають, то це часто робиться за допомогою дронів з декількома камерами, або двох дронів, які летять поряд друг з другом і спостерігають об’єкт з різних боків, а далі за допомогою тріангуляції обчислюється дистанція до нього.

Зважаючи на такі умови наш спосіб працює швидше та має більшу точність.

# Висновки.

В другому розділі було проаналізовано основні методи створення карт місцевості. Було виділено наступні способи: розширений фільтр Калмана, фільтр частинок та SLAM на основі особливостей-міток

Для використання в системі, яка розроблюється, було обрано SLAM на основі особливостей-міток, оскільки цей метод дозволяє простіше орієнтуватися в просторі, та швидше створювати карту.

# СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ СТВОРЕННЯ КАРТ МІСЦЕВОСТІ

Специфікація вимог програмного забезпечення відповідно до IEEE 830 (Software requirements specification)[5] - повний опис поведінки програми, яку потрібно розробити. Зокрема, вона містить ряд сценаріїв користувачів, які описують всі варіанти взаємодії між користувачами і програмним забезпеченням.

Призначені для користувача сценарії засобом представлення функціональних вимог. Крім сценаріїв для користувача, специфікація також містить нефункціональні вимоги, які накладають обмеження на дизайн або реалізацію (такі як вимоги продуктивності, стандарти якості, або проектні обмеження).

# 3.1 Загальні характеристики

Далі описані характеристики для проекту програмної системи для створення карт місцевості, яка дозволяє отримувати в результаті роботи детальні карти місцевості, для полегшення орієнтування на ній.

# 3.1.1 Очікувані результати роботи

В результаті роботи буде реалізовано програмну систему для створення карт місцевості. Система включає в себе алгоритм, який дозволяє створити детальну карту оточуючого простору.

# 3.2 Функціональні вимоги

Функціональні вимоги регламентують[6] функціонування або поведінку системи і відповідають на питання «що повинна робити система» в тих чи інших ситуаціях. Вони визначають основний фронт робіт розробника і визначають цілі, завдання та сервіси, що надаються програмної системою. Найбільш поширеними методами специфікації функціональних вимог є створення діаграми варіантів використання і запис сценаріїв варіантів використання.

Програмна система має забезпечувати обмін інформацією між модулями, та можливість зберігати дані про створенні карти.

Користувач повинен мати можливість керувати положенням камери, змінювати алгоритми обробки зображення. Також програма повинна працювати з усіма системами на базі Linux.

На рисунку 3.1 наведені найбільш популярні Linux дистрибутиви світу, виходячи з даних наданих організацією Losst.ru на липень 2019 року.

Рисунок 3.1 – Популярність Linux дистрибутивів в відсотках з 2019 року

Проаналізував таблицю популярності Linux дистрибутивів з 2019 року, можна зробити висновок, що найбільш поширеними є такі системи: Ubuntu, Debian, CentOS та Mint. Але виходячи з того, що Ubuntu користується найбільшим попитом то при розробці ми в першу чергу орієнтувались на нього.

У системі має бути можливість переглядати, поточні дані с камери, та попередні записи.

Отже, функціональні вимоги до системи:

– вибір, алгоритмів конструювання карти приміщення;

– збір та вивід інформації про роботу системи та камер

– інструкція користування;

Для опису конкретних функцій системи була розроблена діаграма варіантів використання [12], або діаграма прецедентів яка представлена на рисунку 3.2.

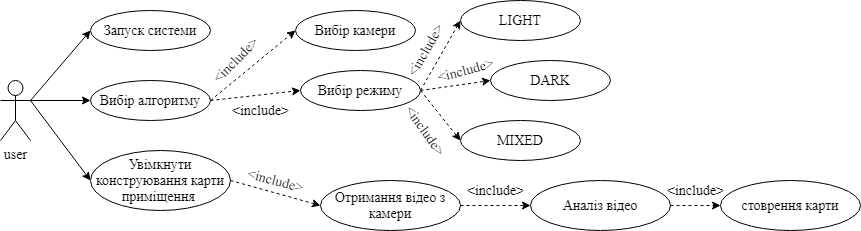


Рисунок 3.2 – Діаграма варіантів використання

Опис сценаріїв варіантів використання:

Скорочення: К – користувач (гравець), С – система..

1) Варіант використання «Вибір алгоритму для роботи з відео».

Передумова: Користувач увійшов у програму

Зацікавлені особи: К

Тригер: К бажає вибрати алгоритм для роботи з відео

Пост умова: К успішно вибрав алгоритм.

1 К відкриває консоль,

1. К вводить назву алгоритму який бажає використовувати для роботи з відео.
2. С виводить на екран меню для цього алгоритму

Альтернативні сценарії.

2а. Введено не існуючу назву алгоритму.

2а1. С виводить повідомлення «Error: wrong algorithm name».

На рисунку 3.3 наведено діаграму послідовності для прецеденту «Вибір алгоритму для роботи з відео»

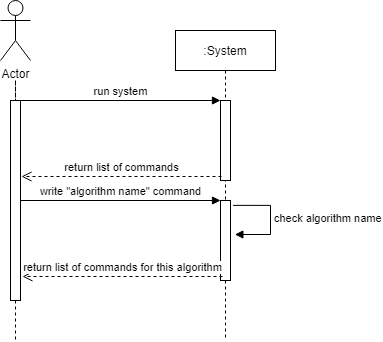


Рисунок 3.3 – Діаграма послідовності для прецеденту «Вибір алгоритму для роботи з відео»

2) Варіант використання «Запуск алгоритму для роботи з відео».

Передумова: Алгоритм обрано

Зацікавлені особи: К

Тригер: К бажає запустити обробку відео з камери

Пост умова: Обробка відео почалась

1. К вводить команду “run”

2. С перевіряє чи підключено камеру

3. С виводить на екран обробку відео з камери.

Альтернативні сценарії.

2а. Камеру не підключено.

2а1. С виводить повідомлення «Error: output device not found».

На рисунку 3.4 наведено діаграму послідовності для прецеденту «Запуск алгоритму для роботи з відео»

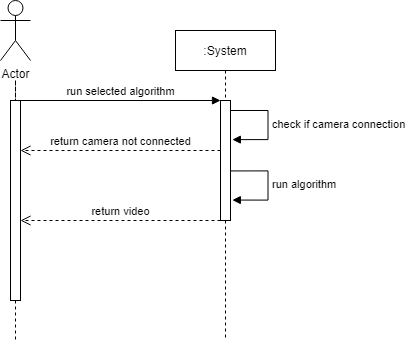


Рисунок 3.4 – Діаграма послідовності для прецеденту «Запуск алгоритму для роботи з відео»

3) Варіант використання «Конструювання 3D моделі приміщення».

Передумова: К запустив алгоритм для роботи з відео

Зацікавлені особи: К

Тригер: К бажає побачити 3Д карту оточуючого простору

Пост умова: вивід 3Д карти

1. К вводить команду для відкриття конструювання карти приміщення “show constructed map”.

2. Система перевіряє чи запущено алгоритм для роботи з відео.

3 .Система виводить на екран створену карту

Альтернативні сценарії

2а. Команду введено не вірно. Система виводить повідомлення «Algorithm is not running».

На рисунку 3.5 наведено діаграма послідовності для прецеденту «Конструювання 3D моделі приміщення»

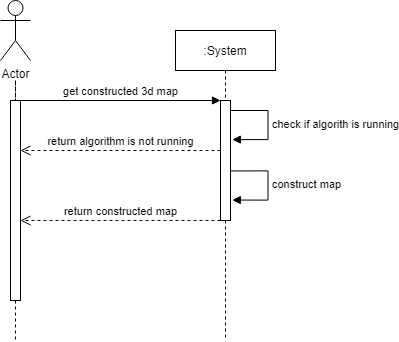


Рисунок 3.5 – Діаграма послідовності для прецеденту «Конструювання 3D моделі приміщення»

# 3.3 Нефункціональні вимоги

Нефункціональні вимоги – опис властивостей або характеристик, які програмне забезпечення повинно демонструвати, або обмеження, яких воно повинно дотримуватися. Розглянемо основні характеристики якості для системи SLAMKINEСT в таблиці 3.1.

Таблиця 3.1 – Нефункціональні вимоги

|  |  |
| --- | --- |
| Набір характеристик | Властивість програмного забезпечення |
| Функціональність | **Здатність до взаємодії:** Програмний продукт повинен взаємодіяти з користувачем для забезпечення роботи  **Надійність:** Програмний продукт має працювати без відказів не менш ніш 2 години. |
| Ефективність | **Часові характеристики:** Програма повинна запускатись завантажуватись не більше як за 5 хвилин  **Використання ресурсів:** Програма не повинна використовувати біль ніж 4гб ОП. |
| Використаність | Запуск будь якого елементу програми має потребувати не більше ніж 3 команд. |
| Переносимість | **Адаптованість:** Програма повинна підтримувати усі сучасні дистрибутиви Linux |

Інтерфейс користувача має бути зручним, простим у використанні та інтуїтивно зрозумілим. Шрифт повинен бути усюди однаковим. Фон – чорний, контраст між фоном та шрифтом – високий. Меню – просте, назви команд – короткі і в той же час чітко відображають пов’язану з ними дію.

Інтерфейс управління. Командна строка

Програмні інтерфейси: обов’язково на комп’ютері має бути встановлений один з дистрибутивів Linux.

Інтерфейси передачі даних: стандарт USB

# 3.4 Оцінка тривалості розробки програмного проекту

Параметричні методи оцінки використовують зв'язок між історичними даними та іншими параметрами, щоб отримати оцінку за допомогою математичної формули. Метод Use Case Point (UCP) розроблений в 1993 році. Він заснований на використанні для оцінки розміру програмного забезпечення прикладів з уніфікованої мови моделювання (Unified Modeling Language - UML). UCP оцінює багато елементів, такі як виконавці, технічна складність і складність середовища. [5]. Цей метод складається з 5 етапів:

1) Оцінка акторів наведена у таблиці 3.2.

Таблиця 3.2 – Визначення типів акторів

|  |  |  |
| --- | --- | --- |
| Назва актора | Тип актора | Вага |
| Користувач | Складний | 6 |

*UAW = 6.*

2) Оцінка варіантів використання наведена у таблиці 3.3.

Таблиця 3.3 – Визначення типів варіантів використання за кількістю класів

|  |  |  |  |
| --- | --- | --- | --- |
| Назва варіанту використання | Тип варіанту використання | Кількість класів | Вага |
| Вибір алгоритму для роботи з відео | Простий | До 5 | 5 |
| Запуск алгоритму для роботи з відео | Простий | До 5 | 5 |
| Зміна персональної інфо-  рмації | Простий | До 5 | 5 |
| Конструювання 3D моделі приміщення | Простий | До 5 | 5 |

*UUCW = 5\*4 = 20.*

*UCP = UAW + UUCW = 6 + 20 = 26.*

3) Оцінка технічних факторів наведена у таблиці 3.3.

Таблиця 3.3 – Технічні фактори

| Фактор | Опис | Вага | Пояснення | Оцінка |
| --- | --- | --- | --- | --- |
| T1 | Розподіленість системи | 2 | Інформує про потребу системи в розподілених обчисленнях | 1 |
| T2 | Час відгуку | 1 | Визначає ефективність системи з точки зору часу відгуку, потоку робіт тощо | 5 |
| T3 | Ефективність кінцевого користувача | 1 | Визначає ефективність користувача з точки зору його (її) сприйняття | 5 |
| T4 | Складність обробки | 1 | Визначає, чи будуть застосовуватись складні алгоритми для обробки даних | 3 |
| T5 | Фокус на повторному використанні коду | 1 | Визначає, чи будуть елементи коду системи використовуватись знову | 2 |
| T6 | Простота інсталяції | 0,5 | Визначає метод інсталяції та простоту інсталяції для кінцевого користувача, чи буде потреба в спеціалісті для інсталяції системи | 1 |
| T7 | Простота використання | 0,5 | Визначає узгодженість інтерфейсу користувача з його потребами | 5 |

Продовження таблиці 3.3 – Технічні фактори

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактор | Опис | Вага | Пояснення | Оцінка |
| T8 | Портативність | 2 | Визначає, чи має застосування працювати в різних середовищах | 3 |
| T9 | Простота змін | 1 | Визначає, чи будуватиметься система в такий спосіб, щоб спростити її модифікації в майбутньому | 3 |
| T10 | Паралельні обчислення | 1 | Інформує, чи будуть мати в системі місце паралельні обчислення | 1 |
| T11 | Засоби захисту | 1 | Визначає чи потребуватиме система спеціальні засоби захисту даних чи системи | 2 |
| T12 | Доступ до третьої сторони | 1 | Визначає ступень використання системи зовнішніми системами або акторами | 3 |
| T13 | Потреби в спеціальному навчанні | 1 | Визначає, чи потрібно організувати тренінги для користувачів | 1 |

*TFactor = 2\*1 + 1\*5 + 1\*5 + 1\*3 + 1\*2 + 0,5\*1 + 0,5\*5 + 2\*3 + 1\*3 + 1\*1 + 1\*2 + 1\*3 + 1\*1 = 36.*

Коефіцієнт технічного фактору: *TCF = 0,6 + (0,01 \* TFactor) = 0,6 + (0,01\*36) = 0,96.*

4) Оцінка зовнішніх факторів наведена у таблиці 3.4.

Таблиця 3.4 – Зовнішні фактори

| Фактор | Опис | Вага | Пояснення | Оцінка |
| --- | --- | --- | --- | --- |
| F1 | Знайомство з  процесом  розробки | 1,5 | Визначає, чи знайома команда з  предметною областю та технічними  аспектами вирішення проблеми клієнта. Особливу увагу слід приділити знанню методології, в якій виконується проект, а також знанню мов моделювання системи | 2 |
| F2 | Досвід подібних  проектів | 0,5 | Загальна уява про досвід команди в  розробці програмного забезпечення | 3 |
| F3 | Досвід об’єктно-  орієнтованої  розробки | 1 | Досвід в проектуванні об’єктно-  орієнтованих застосувань, а також в  підтримці засобів для розробки  інформаційних систем | 3 |
| F4 | Досвідченість  провідного  аналітика | 0,5 | Здатності аналітика отримати вимоги від клієнта та знання щодо задач, які буде вирішувати система | 2 |
| F5 | Мотивація | 1 | Здатність команди займатись призначеною задачею | 3 |
| F6 | Стабільність  вимог | 2 | Визначає, чи не будуть вимоги часто  змінюватись | 3 |
| F7 | Часткова  зайнятість  працівників | -1 | Визначає, наскільки великою є частка  працівників часткової зайнятості | 2 |

Продовження таблиці 3.4 – Зовнішні фактори

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактор | Опис | Вага | Пояснення | Оцінка |
| F8 | Складність мови  програмування | -1 | Визначає, наскільки складно вивчити мову програмування: 0 – легко, за тиждень можна опанувати; 1 – щонайменше 2 тижні потрібно для опанування мови; 2 – щонайменше місяць потрібно для опанування мови; 3 – потрібен спеціальний тренінг з мови; 4 – потрібен спеціальний тренінг та допомога під час виконання проекту; 5 – складно, потрібні лише досвідчені люди. | 2 |

*EFactor = 1,5\*2 + 0,5\*3 + 1\*3 + 0,5\*2 + 1\*3 + 2\*3 + (-1)\*2 + (-1)\*2 = 11,5.*

Коефіцієнт зовнішніх факторів: *EF = 1,4 + (-0,03 \* EFactor) = 1,4 +*

*+ (-0,03\*11,5) = 1,055.*

*5)* Скоректований UCP: *AUCP = UCP \* TCF \* EF =26 \* 0,96 \* 1,055 = 26,716.*

Одному AUCP відповідає 20 робочих годин. Отже, загальний час на розробку дорівнює: *20 \* 26,716.= 402 (годин) = 78 робочих діб.*

# 3.5 План розробки

Структура декомпозиції робіт (WBS) – це ієрархічне розбиття всієї роботи, яку необхідно виконати для досягнення цілей проекту, на більш дрібні операції і дії до такого рівня, на якому способи виконання цих дій цілком ясні і відповідні роботи можуть бути оцінені і сплановані.

Після того, як визначено склад робіт, потрібно оцінити термін їх виконання, а також встановити залежності, що існують між роботами. Структура декомпозиції робіт та строки виконання представлені на рисунку 3.5.

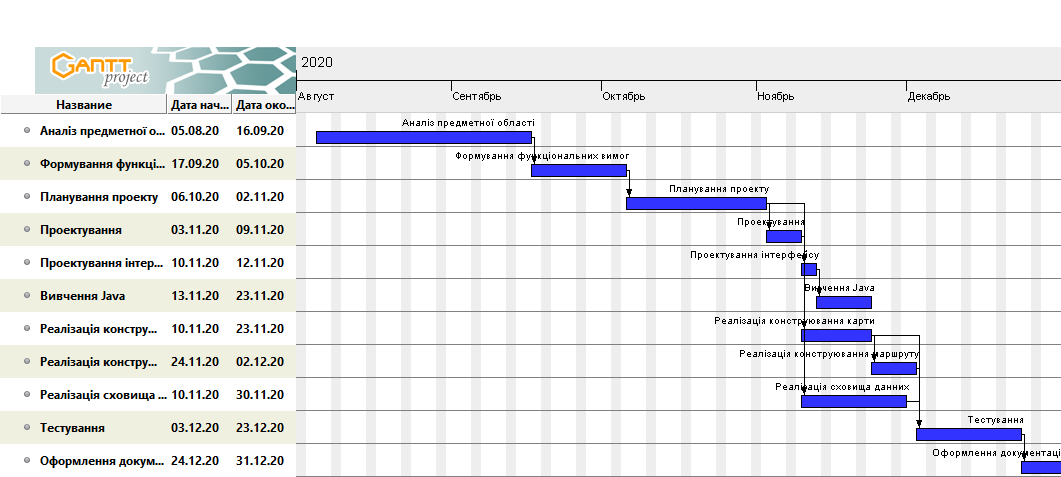
****

Рисунок 3.5 – Структура декомпозиції та строки виконання робіт

Для ілюстрації плану виконання дипломного проекту використаємо діаграму Ганта [10].

Діаграма являє собою відрізки, розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або під задачі. Завдання і під задачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма Ганта представлена на рисунку 3.6.

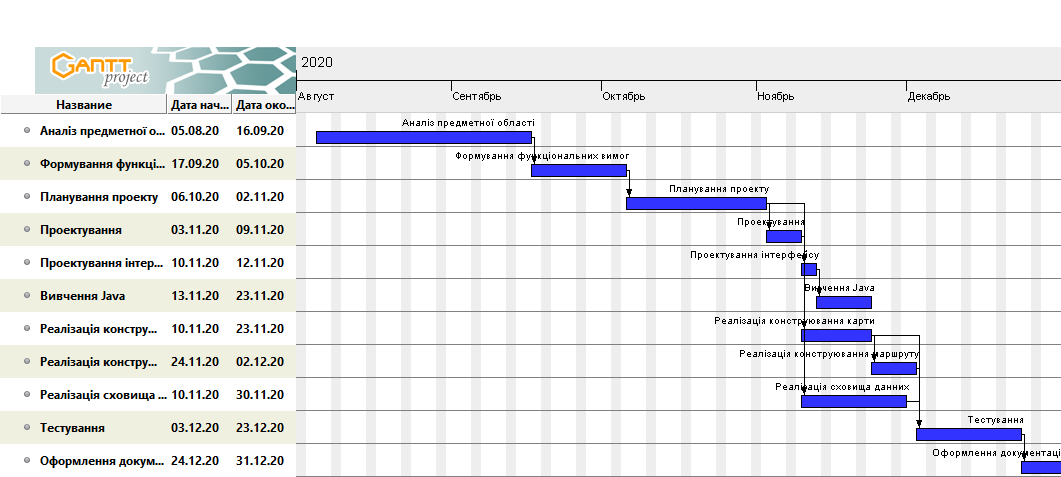


Рисунок 3.6 – Діаграма Ганта

Дана діаграма наглядно відображає тривалість кожної роботи проекту та залежність між ними.

# 3.6 Аналіз ризиків

На Рисунку 3.7 представлена діаграма Ішикави [11], на якій зображено 10 ризиків проекту та наслідок, до якого приводять ці ризики – провал проекту.

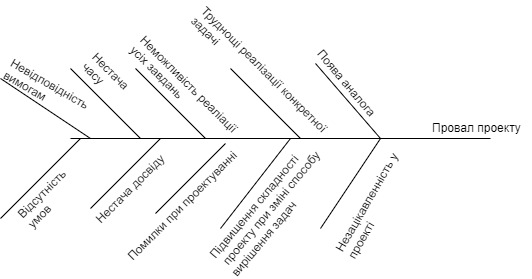


Рисунок 3.7 – Діаграма Ішикави для розроблюваної системи

У таблиці 3.5 представлене визначення рангів ризиків за допомогою матриці ймовірностей:

Таблиця 3.5 – Визначення рангів ризиків

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Ризик | Ймовірність | Вплив | Ранг |
| 1 | Нестача часу | 3 | 3 | 6 |
| 2 | Нестача досвіду розробки подібних програм | 2 | 3 | 6 |
| 3 | Неможливість реалізації всіх поставлених завдань | 2 | 3 | 6 |
| 4 | Помилки проектування | 2 | 3 | 6 |
|  |  |  |  |  |
| 5 | Підвищення складності при зміні способу рішень під задач | 2 | 2 | 4 |
| 6 | Труднощі реалізації конкретної під задачі | 1 | 3 | 3 |
| 7 | Незацікавленість у проекті | 1 | 3 | 3 |
| 8 | Невідповідність проекту вимогам | 1 | 2 | 2 |
| 9 | Поява аналогу розроблюваного проекту | 1 | 2 | 2 |
| 10 | Тимчасова відсутність необхідних умов | 2 | 1 | 2 |

Визначимо п’ять ризиків, що мають найбільший ранг:

– Нестача часу;

– Нестача досвіду розробки подібних програм;

– Неможливість реалізації всіх поставлених завдань;

– Помилки проектування;

– Підвищення складності при зміні способу рішень під задач.

Наведемо в таблицях 2.6-2.10 картки опису для цих п’яти ризиків, що містять також сплановану реакцію на ризики:

Таблиця 3.6 – Картка опису для ризику №1

|  |  |
| --- | --- |
| Номер: 1 | Категорія: Організаційний |
| Причина: Нестача часу | Умови: Стислі терміни |
| Наслідки: затримки у виконанні проекту | Вплив: підвищення термінів виконання проекту, не повна реалізація проекту |
| Імовірність: 2 | Ступінь впливу: 3 |
| Близькість: дуже скоро | Ранг: 6 |
| Стратегія реагування: Зниження ризику. Поставити конкретні строки виконання усіх задач і під задач. | |

Таблиця 3.7 – Картка опису для ризику №2

|  |  |
| --- | --- |
| Номер 2 | Категорія: Технічний |
| Причина: Нестача досвіду розробки подібних програм | Умови: Для розробки буде використовуватись мова програмування Java та операційна система Ubuntu разом з ROS |
| Наслідки: Низька продуктивність розробки | Вплив: Збільшення термінів і трудомісткості розробки |
| Імовірність: 3 | Ступінь впливу: 2 |
| Близькість: дуже скоро | Ранг: 6 |
| Стратегія реагування: Прийняття ризику. | |

Таблиця 3.8 – Картка опису для ризику №3

|  |  |
| --- | --- |
| Номер 3 | Категорія: Технічний |
| Причина: Неможливість реалізації всіх поставлених завдань | Умови: Занадто об'ємний і складний проект |
| Наслідки: Не укладання у терміни | Вплив: Різке збільшення термінів. |
| Номер 3 | Каегорія: Технічний |
| Імовірність: 2 | Ступінь впливу: 3 |

Продовження таблиці 3.8

|  |  |
| --- | --- |
| Номер 3 | Категорія: Технічний |
| Близькість: скоро | Ранг: 6 |
| Стратегія реагування: Ухилення від ризику. Уточнення вимог, зменшення складності проекту | |

Таблиця 3.9 – Картка опису для ризику №4

|  |  |
| --- | --- |
| Номер 4 | Категорія: Технічний |
| Причина: Помилки проектування | Умови: Неможливість спланувати все заздалегідь без помилок |
| Наслідки: Помилки при наступних етапах, необхідність перероблювати вже виконану роботу, зменшення продуктивності | Вплив: Збільшення термінів і трудомісткості розробки |
| Імовірність: 2 | Ступінь впливу: 3 |
| Близькість: не скоро | Ранг: 6 |
| Стратегія реагування: Прийняття ризику. Помилки проектування трапляються дуже часто, і залишається сподіватися, що їх імовірність зменшиться при збільшенні досвіду з плином часу. | |

Таблиця 3.10 – Картка опису для ризику №5

|  |  |
| --- | --- |
| Номер 5 | Категорія: Технічний |
| Причина: Підвищення складності при зміні способу рішень під задач | Умови: При реалізації під задач виявилось, що для їх виконання необхідно більше знань |
| Наслідки: Затримки реалізації підзадачі та усього проекту, спрощення планованої реалізації, зміна вигляду фінального проекту | Вплив: Збільшення термінів розробки або погіршення якості кінцевого продукту |

Продовження таблиці 3.10

|  |  |
| --- | --- |
| Номер 5 | Категорія: Технічний |
| Імовірність: 2 | Ступінь впливу: 3 |
| Близькість: не скоро | Ранг: 6 |
| Стратегія реагування: Зниження ризику. Спроба реалізації підзадач проекту до закінчення планування. Це дасть можливість запланувати більше часу на реалізацію складної підзадачі, а також знання таких складних місць у проекті заздалегідь збільшить ймовірність усунення даного ризику. | |

Отже, була розроблена причинно-наслідкова діаграма Ішикави, визначені ранги ризиків за допомогою матриці ймовірностей. П’ять ризиків з найбільшими рангами були детально розглянуті, та була спланована реакція на них.

# Висновки

В третьому розділі було описано функціональні вимоги до системи за допомогою діаграми функціональних вимог. Були визначені, актори системи, описані сценарії варіантів використання системи. Проведено аналіз алгоритмів, які використовує система.

Описано проектування системи, планування розробки продукту за допомогою методу UCP, структури декомпозиції робіт, діаграми Ганта, проведено аналіз ризиків за допомогою діаграми Ішикави.

# ОПИС ПРОЕКТУВАННЯ СИСТЕМИ СТВОРЕННЯ КАРТ МІСЦЕВОСТІ.

# 4.1 Програмне проектування системи

Програмне проектування системи було вирішено почати зі створення концептуальної діаграми класів для того щоб на її основі створити більш конкретні структури програми, такі, як діаграма програмних класів. Далі буде наведено структуру бази даних та структуру пакетів даних якими обмінюються програмні класи.

На рисунку 4.1 представлена діаграма концептуальних класів [8]. Це структурна діаграма мови моделювання UML, що демонструє загальну структуру ієрархії класів системи, їх кооперацій, атрибутів (полів), методів, інтерфейсів і взаємозв'язків між ними.

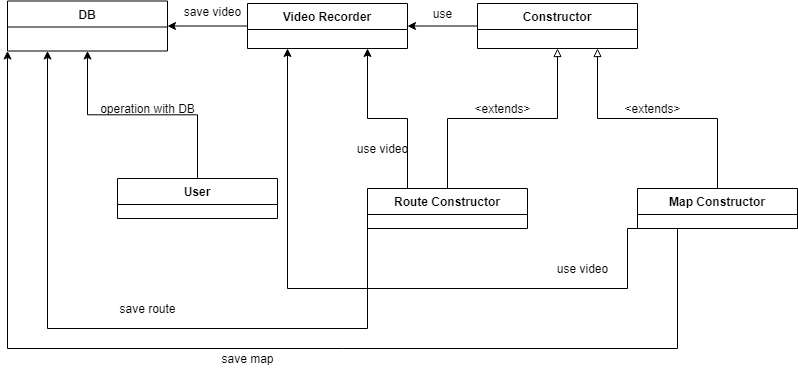


Рисунок 4.1 – Діаграма концептуальних класів

Опис діаграми. На діаграмі представлені основні концептуальні класи:

Клас DB на діаграмі відповідає за роботу з базою даних

Клас Video Recorder відповідає за роботу з камерою та передачу відео.

Клас Constructor містить спільні поля та методи для класів Map Constructor та Route Constructor.

Клас User відповідає за інтерпретацію консольних команд користувача.

Клас Route Constructor відповідає за створення маршруту яким рухалась камера. Цей маршрут потрібен для створення карти

Клас Map Constructor відповідає за створення карти приміщення.

# 4.1.1 Опис структур даних

Нижче представлена структура пакету, який отримують класи Constructors від класу, що відповідає за запис відео:

іd:this.id, - унікальний ідентифікатор кожного кадру

x:this.x, - позиція кадру по Х відносно попередніх кадрів

y:this.y, - позиція кадру по Y відносно попередніх кадрів

z: this.z - позиція кадру по Z відносно попередніх кадрів

img:image – кадр зображення

Структура пакету який отримує клас бази даних для зберігання від класів конструкторів:

id: this.id – унікальний ідентифікатор карти

img[]: imagesMap

name: String

agentId: agentId

На рисунку 4.2 представлено структуру бази даних. На ній зображено таблиці поточного відео, та таблиці з обробленою картою та маршрутом.

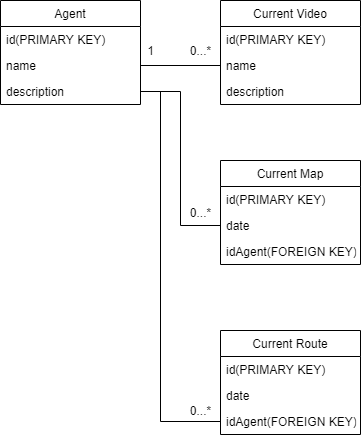


Рисунок.4.2. Структура бази даних

Таблиця Agent відповідає за збереження даних з поточного запису

Таблиця Current Video зберігає необроблені відео.

Таблиця Current Map зберігає поточну поточну оброблену карту місцевості

Таблиця Current Route зберігає поточний оброблений маршрут

## **4.1.2 Структура програмних класів**

На рисунку 4.3 представлена діаграма програмних класів та зв’язки між ними, наведені поля та функції кожного класу. А у нижче наведено опис діаграми та кожного класу.

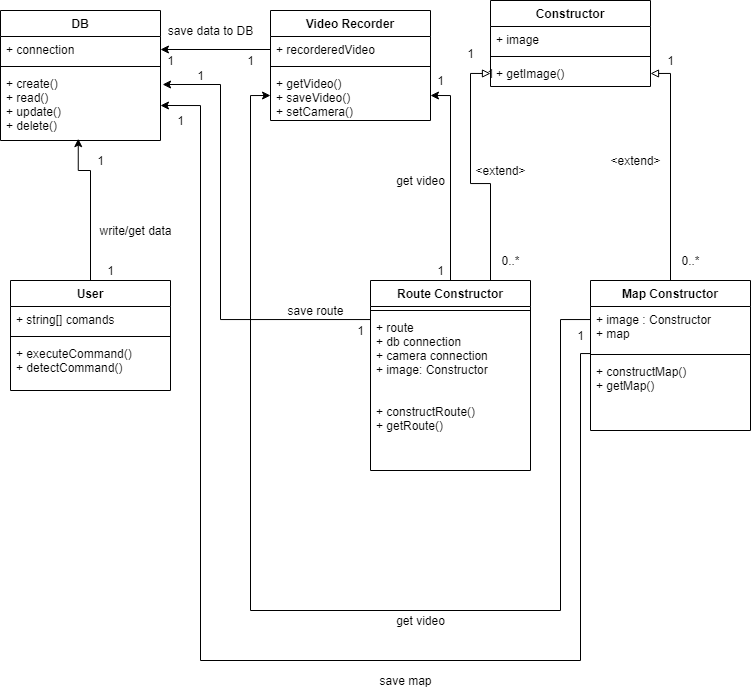


Рисунок 4.3 – Діаграма програмних класів

Клас DB на діаграмі відповідає за роботу з базою даних

Клас Video Recorder відповідає за роботу з камерою та передачу відео. Він присвоює кожному знімку унікальний ідентифікатор та координати відносно попередніх знімків

Клас Constructor містить спільні поля та методи для класів Map Constructor та Route Constructor.

Клас User відповідає за інтерпретацію консольних команд користувача. Та їх перевірку

Клас Route Constructor відповідає за створення маршруту яким рухалась камера. Цей маршрут потрібен для створення карти. Він зберігає у собі усі координати зі знімків

Клас Map Constructor відповідає за створення карти приміщення. Він отримує координати від класу RouteConstructor і використовує їх для того щоб поставити знімки на їх місця на карті.

# 4.1.3. Зміст програмних класів та опис алгоритмів

Клас DB містить у собі поле connection яке відповідає за підключення до бази даних, а також методи для створення, модифікації та зчитування та видалення даних з бази.

Клас Video Recorder містить у собі поле з поточним відео з камери та методи для видачі відео, встановлення камери та збереження відео у базу даних.

Клас User зберігає список команд від користувача у своєму полі commands

Методи executeCommand() та deleteCommand() зчитують команди з консолі та виконують їх, або видаляють.

Клас Route Constructor містить у собі поля: route яке відповідає за поточний маршрут, поле dbconnection яке потрібно для під’єднання до бази даних та роботи з нею, поле camera connection яке відповідає за роботу з камерою, та отримання поточного відео від класу Video Recorder, та поле image яке відповідає за поточну картинку яка оброблюється. Метод constructRoute() відповідає за створення маршруту з поточного зображення та його збереження у поле route, після чого визивається метод getRoute() який зберігає поточний маршрут у базу даних.

Клас Map Constructor містить у собі поля: map яке відповідає за поточну створену карту, та поле image яке відповідає за поточну картинку яка оброблюється. Метод constructMap() відповідає за створення карти з поточного зображення та його збереження у поле map, після чого визивається метод getMap() який зберігає поточну карту у базу даних.

На рис. 4.4 представлена схема алгоритму для функції «Оновлення карти приміщення».



Рисунок 4.4 – Схема алгоритму для функції оновлення карти приміщення

На рис. 4.5 представлена схема алгоритму для функції «Оновлення маршруту».

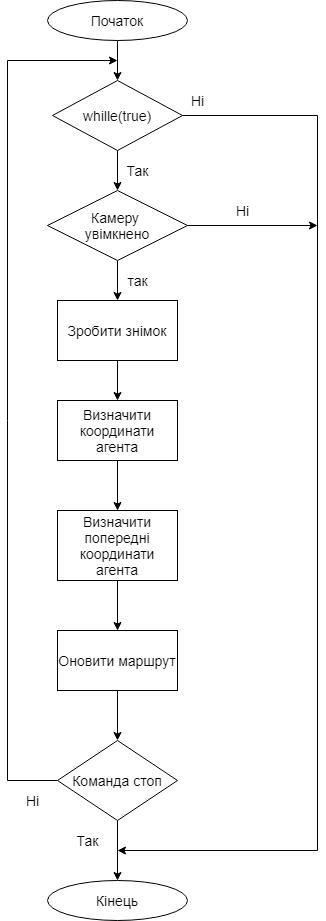


Рисунок 4.5 – Схема алгоритму для функції оновлення маршруту

На рис. 4.6 представлена схема алгоритму для функції «Оновлення координат».

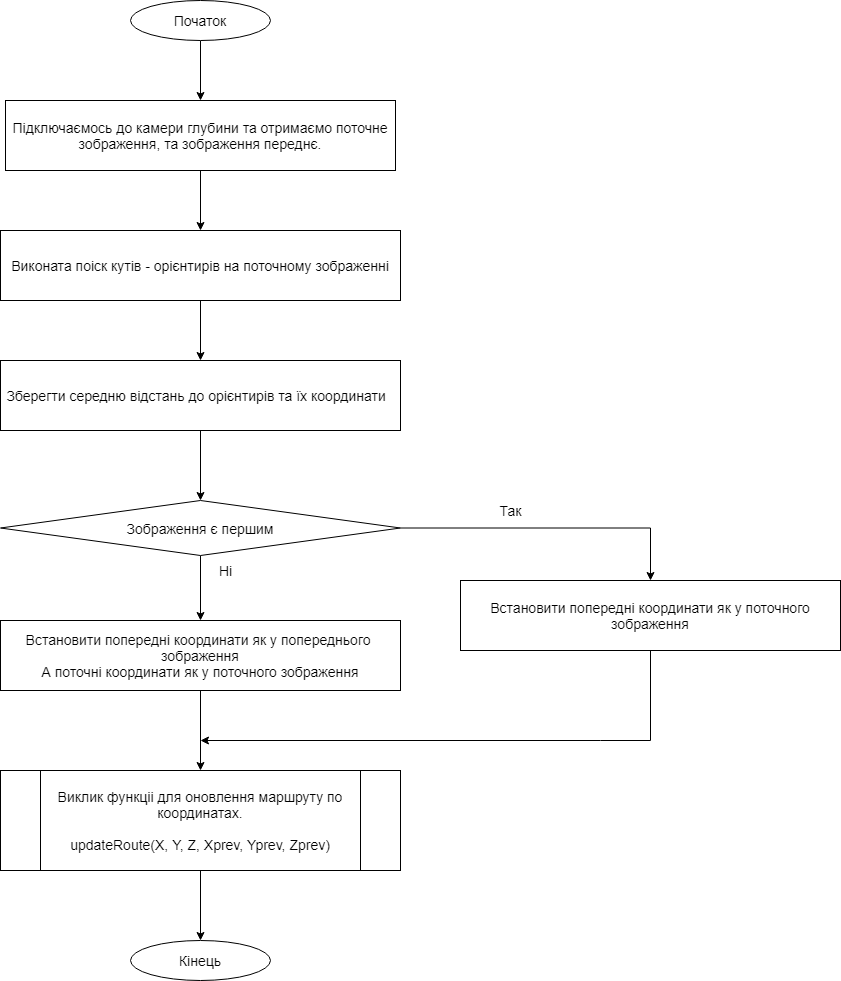


Рисунок 4.6 – Схема алгоритму для функції оновлення координат

На рис. 4.7 представлена схема алгоритму для функції «Оновлення маршруту по координатах».

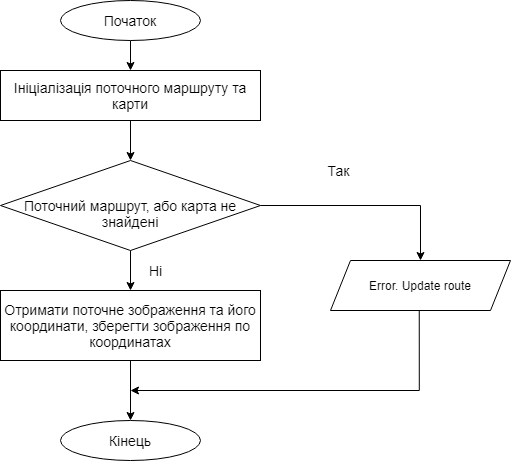


Рисунок 4.7 – Схема алгоритму для функції оновлення маршруту по координатах

## **4.2 Проектування інтерфейсу користувача**

Інтерфейс користувача створений у консольному стилі, оскільки програмну систему планується завантажувати на носії, які мають обмежену кількість пам’яті та ресурсів для обробки інформації. Тому користувач взаємодіє з програмою шляхом введення команд у терміналі, і отримує зображення в іншому вікні. На рисунку 4.8, 4.9, 4.10, 4.11 показано приклад роботи програми. А зеленими кулями помічено орієнтири (кути) які розпізнала камера. А на рисунку 4.12 діаграму огляду взаємодії.



Рисунок 4.8 Колаж зі знімків камери.

На рисунку 4.9 показано початок процесу створення карти кімнати.

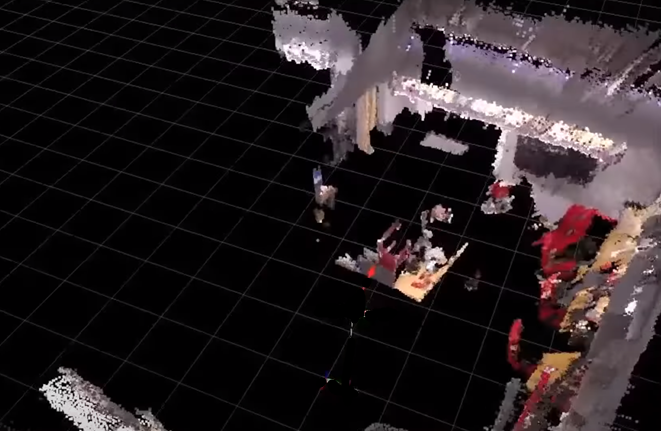


Рисунок 4.9 Початок створення карти кімнати.

На рисунку 4.10 показано поступову деталізацію кімнати. З поворотом камери алгоритм отримує нові знімки які додаються до карти поступово деталізуючи її.

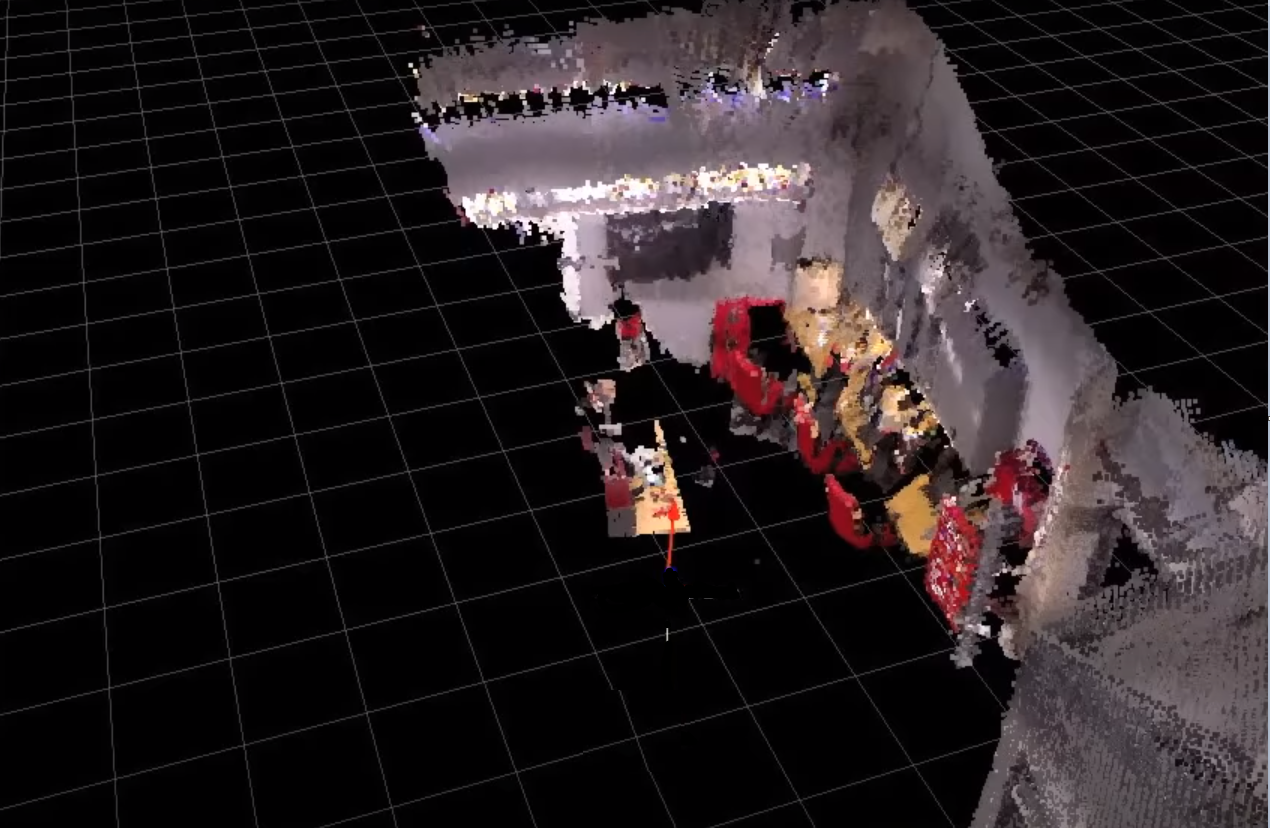


Рисунок 4.10 Більш деталізована карта карта.

На рисунку 4.11 показано повну карту приміщення деталізацію кімнати.

Рисунок 4.11 Фінальна карта кімнати.

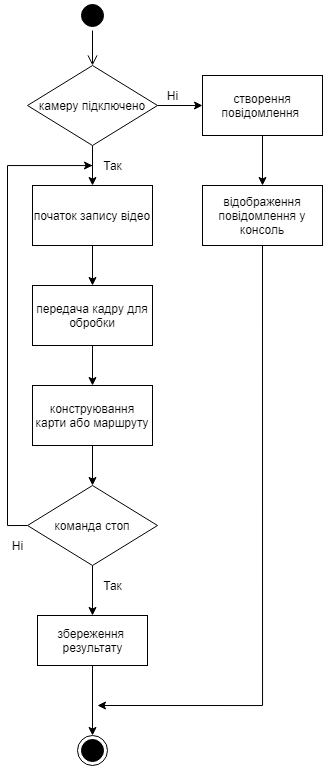


Рисунок 4.12 Діаграма огляду взаємодії роботи програми.

Для керування програмою створено наступні команди:

- run – запускає програму та всі її компоненти на встановленому прострої і відображає відео з камери

- select\_video\_algorithm – ця команда дозволяє вибрати алгоритм для роботи з відео в залежності від умов освітлення. На вибір користувачу надаються 3 алгоритми:

1. select\_video\_alhorithm = LIGHT – для роботи при достатньому освітленні

2. select\_video\_alhorithm = MIXED – для роботи при змішаному освітленні

3. select\_video\_alhorithm = DARK – для роботи при недостатньому освітленні

4. start – запуск конструювання карти

Їх різниця полягає у кількості ресурсів яку потребує робота системи, що важливо в умовах коли ємність батареї дронів обмежена. Так, наприклад, алгоритм DARK потребує увімкнення інфрачервоного ліхтаря, а алгоритм MIXED намагається конструювати карту не використовуючи його.

# Висновки

У цьому наведено діаграми концептуальних та програмних класів, їх опис, та функції, описані основні алгоритми роботи програми та структури даних, наведено схему огляду взаємодії роботи програми та описано інструкцію з користування системою.

# 5 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУВАННЯ ПЛАНУВАННЯ ЗАХОДІВ ПІД ЧАС ПОДОРОЖІ

Після того, як отримано детальний проект з компонентами, в яких визначена структура і методи взаємодії, можна братися за реалізацію компонентів, тобто створення і тестування програмних модулів, визначених при проектуванні.

# 5.1 Набір інструментальних засобів розробки

Для розробки програмного продукту були використані наступні інструменти:

– JetBrains ItelijIDEA – це інтегроване середовище розробки на Java, JavaScript, HTML, CSS та інших мовах програмування, з інтелектуальним редактором, що глибоко розуміє код, забезпечує найкраще в індустрії автодоповнення коду, рефакторинги, запобігання помилок нальоту, підтримує змішування мов.

ItelijIDEA має повнофункціональну підтримку баз даних / SQL [9].

JAVA – строго типізований об'єктно-орієнтована мова програмування загального призначення, розроблений компанією Sun Microsystems (в подальшому придбаної компанією Oracle). Розробка ведеться співтовариством, організованим через Java Community Process, мова і основні реалізують його технології поширюються за ліцензією GPL. Права на торговельну марку належать корпорації Oracle.

Програми Java зазвичай транслюються в спеціальний байт-код, тому вони можуть працювати на будь-якої комп'ютерної архітектурі, для якої існує реалізація віртуальної Java-машини [3].

СУБД MySQL – вільна реляційна система управління базами даних. Розробка та підтримка сайта MySQL здійснює корпорація Oracle, яка отримала права на торговельну марку разом з поглиненої Sun Microsystems, яка раніше придбала шведську компанію MySQL AB. Продукт поширюється, як під GNU General Public License, так і під власною комерційною ліцензією. Крім цього, розробники створюють функціональність за замовленням ліцензійних користувачів. Саме завдяки такому замовленню майже в найперших версіях з'явився механізм реплікації.

Входить до складу серверів WAMP, AppServ, LAMP і в портативні збірки серверів Денвер, XAMPP, VertrigoServ. Зазвичай MySQL використовується, як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати, як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць [13].

Камера Kineсt360, ОС Ubuntu, та система ROS – операційна система для роботів - це кістяк для програмування роботів, що надає функціональність для розподіленої роботи. ROS був спочатку розроблений в 2007 році під назвою switchyard в Лабораторії Штучного Інтелекту Стенфордського Університету для проекту. У 2008 році розвиток триває в Willow Garage, науково-дослідному інституті / інкубаторі робототехніки, спільно з більш ніж двадцятьма співпрацюють інститутами [14].

## **5.2 Інструкція розгортання**

Усі налаштування системи проводяться на операційній системі Linux. Усі наступні команди необхідні для нормальної роботи веб-застосування для планування подорожей:

1) Встановлення ROS: sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

2) Встановлення ROS пакетів: sudo apt install ros-noetic-desktop-full

3) Встановлення оточення: source /opt/ros/noetic/setup.bash

4) Клонування проекту з репозиторію git clone

https://github.com/medmmax/slamkinect.git

7) Перехід до папки проекту та встановлення основних модулів необхідних для роботи сервера cd ~/slamkinect; maven install

8) Запуск системи після усіх проведених вище операцій. Slamkinect run

# 6 ТЕСТУВАННЯ СИСТЕМИ

## **6.1 Функціональне тестування**

Функціональні тести ґрунтуються на функціях, виконуваних системою, і можуть проводитися на всіх рівнях тестування. Як правило, ці функції описуються у вигляді варіантів використання системи.

Розробимо тестові сценарії [15] для тестування функціональності системи на основі варіантів використання таблиця 6.1

Таблиця 6.1 – Тестові сценарії

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Передумова | Порядок дій | Очікуваний результат | Фактичний результат |
| 1 | Користувач запустив консоль та ввів команду slamkinect run | Запустити систему. Перевірити підключення до бази даних | Ініціалізація системи. Та очікування наступних команд | Пройдено |
| 2 | Користувач запустив систему та ввів команду select\_video\_algorithm | Перевірити підключення до камери, вивести меню з типами алгоритму | Користувачу відображено меню з вибором алгоритму | Пройдено |

Продовження таблиці 6.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Передумова | Порялок дій | Очікуваний результат | Фактичний результат |
| 3 | Користувач запустив систему та ввів команду select\_video\_algorithm = LIGHT | Система підстроює камеру під вимоги алгоритму, вимикає інфрачервоний проектор. | Система готова до конструювання карти приміщення з вибраними опціями. | Пройдено |
| 4 | Користувач запустив систему та ввів команду select\_video\_algorithm = MIXED | Система підстроює камеру під вимоги алгоритму, аналізує освітленість на місцевості. | Система готова до конструювання карти приміщення з вибраними опціями. | Пройдено |
| 5 | Користувач запустив систему та ввів команду select\_video\_algorithm = DARK | Система підстроює камеру під вимоги алгоритму, вмикає інфрачервоний проектор. | Система готова до конструювання карти приміщення з вибраними опціями | Пройдено |

Продовження таблиці 6.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Передумова | Порядок дій | Очікуваний результат | Фактичний результат |
| 6 | Користувач обрав алгоритм для роботи та ввів команду start | Отримати зображення з камери, підключитись до бази даних та почати обробку відео | Вивід зображення створеної карти приміщення | Пройдено |

## **6.2 Модульне тестування**

Виконаємо модульне тестування функцій системи «Конструювання карти приміщення». Проведемо тестування методом чорного ящику.

Таблиця 6.2 – Тестування створення нової карти

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип | Опис параметрів | Очікуваний результат | Фактичний результат |
| 1 | Невірний | Користувач вимикає камеру. | Помилка даних | Помилка про відсутність даних |
| 2 | Невірний | Користувач вимикає базу даних | Помилка про неможливість збереження даних | Помилка про відсутність з’єднання з базою даних |

Продовження таблиці 6.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип | Опис параметрів | Очікуваний результат | Фактичний результат |
| 3 | Невірний | Користувач вводить невірну назву алгоритму | Помилка про невірну назву алгоритму | Помилка про невірну назву алгоритму |

## 

## **6.3 Експеримент**

Був проведений експеримент, у якому запроваджено тестування програми тривалістю в 6 днів, у якому систему тестували у 4 різних умовах освітлення, та в наступному порядку: при відсутності світла, при повному освітленні, при змішаному освітлені, та при переході від повної відсутності, до повного освітлення приміщення. Також разом з цим тестувалась робота програми для створення карт від системи ROS.

Була поставлена задача створити карту приміщення площею 16 квадратних метрів.

Час на створення карти займає від 1.5 годин до 2 годин в залежності від освітлення.

А за допомогою розробленого алгоритму приблизно час двадцять для освітленних приміщень і близько часу п’ятидесяти для неосвітленних. Час, який витратив кожен з протестованих алгоритмів на створення карти приміщення, указано у діаграмах на рис. 6.1 – 6.6. А на таблиці 6.3 та 6.4 наведено дані по кожному дні експерименту.

Таблиця 6.3 – Дані експерименту для системи ROS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Умови | День 1 | День 2 | День 3 | День 4 | День 5 | День 6 |
| Повне освітлення | 90 | 100 | 89 | 85 | 95 | 105 |

Продовження таблиці 6.3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Умови | День 1 | День 2 | День 3 | День 4 | День 5 | День 6 |
| Відсутність освітлення | 120 | 130 | 115 | 121 | 125 | 120 |
| Часткове освітлення | 100 | 108 | 95 | 92 | 105 | 115 |
| Перехідне освітлення | 105 | 114 | 101 | 98 | 110 | 120 |

Таблиця 6.4 – Дані експерименту для системи SLAMKINECT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Умови | День 1 | День 2 | День 3 | День 4 | День 5 | День 6 |
| Повне освітлення | 76 | 87 | 75 | 70 | 81 | 90 |
| Відсутність освітлення | 108 | 116 | 105 | 110 | 112 | 109 |
| Часткове освітлення | 95 | 107 | 92 | 89 | 100 | 110 |
| Перехідне освітлення | 103 | 110 | 100 | 92 | 110 | 118 |

Рисунок 6.1. - Порівняння часу витраченого на створення карти приміщення у перший день

Рисунок 6.2 - Порівняння часу витраченого на створення карти приміщення у другий день

Рисунок 6.3 - Порівняння часу витраченого на створення карти приміщення у третій день

Рисунок 6.4 - Порівняння часу витраченого на створення карти приміщення у четвертий день

Рисунок 6.5 - Порівняння часу витраченого на створення карти приміщення у п’ятий день

Рисунок 6.6 - Порівняння часу витраченого на створення карти приміщення у шостий день

На наступному етапі було пораховано середній арифметичний час затрачений кожною з систем на створення карти за усі дні та зведено дані у таблицю 6.5, а на рис. 6.7 зображено їх графік.

Для системи ROS

(90+100+80+85 + 95 + 105)/7 = 79

(120+130+115+121+125+120)/7 = 104

(100+108+95+92+105+115)/7 = 87

(105+114+101+98 +110+120)/7 = 92

Для системи SLAMKINECT

(76+87+75+70+81+90)/7 = 68

(108+116+105+110+112+109)/7 = 94

(95+107+92+89+100+110)/7 = 82

(103+110+100+92+110+118)/7 = 83

Таблиця 6.5 – Зведені дані по усім дням експерименту

|  |  |  |
| --- | --- | --- |
| Умови | ROS | SLAMKINECT |
| Повне освітлення | 79 | 68 |
| Відсутність освітлення | 104 | 94 |
| Часткове освітлення | 87 | 82 |
| Перехідне освітлення | 92 | 83 |

Рисунок 6.6 - Середній арифметичний час витрачений на створення карти приміщення у різних умовах

З наведеної діаграми можна побачити, що час витрачений при використані алгоритму SLAM разом з камерою Kinect360 у середньому менше ніж при використані алгоритмів ROS приблизно на 10%.

# 7. ОХОРОНА ПРАЦІ

Охорона праці це система правових, соціальних, економічних, організаційно-технічних, санітарно-гігієнічних, лікувально-профілактичних заходів та засобів, направлених на збереження життя та здоров’я, працездатності співробітників в процесі трудової діяльності.

Умови праці визначаються відповідно до ГОСТ 19605-74 «Організація праці. Основні поняття, терміни і визначення».

Працівники галузі розробки програмного забезпечення при роботі з комп'ютером наражаються ряду небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон радіочастот: ВЧ, УВЧ і СВЧ), інфрачервоного і іонізуючого випромініювань, шуму і вібрації, статичної електрики і ін.

Вивчення найважливіших заходів з охорони праці та безпеки у надзвичайних ситуаціях для працівників галузі інженерія програмного забезпечення наведено у Додатку Б та висвітлює наступні питання:

* організація та управління охороною праці на фірмі.

## обґрунтування заходів з покращення умов охорони праці

## розрахунок освітлення на робочому місці

* надзвичайні ситуації та шляхи їх запобігання

В ході роботи проаналізовано робоче місце інженера-програміста, вимоги до інтервалів робочих змін, необхідний рівень освітленості при умовах роботи на моніторі високої точності, та режим відпочинку.

# Висновок

У даному розділі було розглянуто управління та організацію охорони праці на робочому місці інженера-програміста, визначено основні параметри умов праці згідно індивідуального завдання.

Проведено розрахунок індивідуального завдання на тему - освітлення та сформовані рекомендації щодо нього.

Проведено розрахунок індивідуального завдання на тему - прогнозування наслідків аварії з викидом хімічно-небезпечних речовин.

Надано рекомендації щодо безпеки у суспільстві в умовах загрози COVID – 19.

# ВИСНОВКИ

В ході роботи було спроектовано та розроблено програмну систему для конструювання карти приміщення за допомогою алгоритму SLAM. Досягнуто мету роботи по зменшенню часу на створення карти. В результаті дослідження встановлено, що розробка програмної системи на 10% прискорить процес створення 3Д карти приміщення у порівнянні з алгоритмом від системи ROS. Визначені функціональні та нефункціональні вимоги до системи. Виконано план розробки та аналіз ризиків. Проведене проектування системи та її алгоритмів. Виконано планування проекту, у ході якого була визначена тривалість повної розробки у 78 днів, розроблено план виконання робіт, ідентифіковано ризики та розглянуті реакції на найбільш імовірні ризики. Наведено опис та обґрунтування технологій̈ розробки, які дозволяють реалізувати архітектуру програмної системи, написано інструкцію користувача.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

* + - 1. Брюс Эккель Философия Java [Текст] /Брюс Эккель -СПб.: Питер1998. – 1168c.
      2. Мартін Фоулер UML Distilled[Текст] / Мартін Фоулер СПб.: Мічіган 2018. – 456c.
      3. Кевін Уейн,Роберт Седжвик Алгоритмы на Java [Електроний ресурс] / Кевін Уейн, Роберт Седжвик – 2018. – 848с.
      4. Форсайт. Компьютерное зрение, современный подход. 2018. – 960с
      5. Аэрофотосъёмка [Електроний ресурс] / Википедия – Режим доступа www/URL: <https://ru.wikipedia.org/wiki/Аэрофотосъёмка> - Загл. с экрана.
      6. Обзор методов для создания карт местности [Електроний ресурс] / Юнек – Режим доступа www/URL: <https://yuneecrussia.ru/10-best-software-3dmapping/>
      7. SLAM [Електроний ресурс] / Википедия – Режим доступа www/URL: https://ru.wikipedia.org/wiki/SLAM\_(метод) - Загл. с экрана.
      8. SLAM – принципы работы [Електроний ресурс] / My-IT-Notes – Режим доступа www/URL: http://my-it-notes.com/2013/01/slam-basis-and-links-at-open-source/) - Загл. с экрана.
      9. Use Case Points [Текст] / Gautam Banerjee. – 2001. – 21 c
      10. Диаграмма Ганта [Электронный ресурс] / Википедия. – Режим доступа: www / URL : https://ru.wikipedia.org/wiki/Диаграмма\_Ганта – 28.04.2016 г. – Загл. с экрана.
      11. Диаграмма Исикавы [Электронный ресурс] / Википедия. – Режим доступа: www / URL : https://ru.wikipedia.org/wiki/Диаграмма\_Исикавы – 30.05.2016 г. – Загл. с экрана.
      12. Буч, Г. Язык UML. Руководство пользователя [Текст] / Г. Буч. – М. : ДМК пресс, 2007. – 496 с.
      13. MySQL [Электронный ресурс] / Википедия. – Режим доступа: www / URL : <https://ru.wikipedia.org/wiki/MySQL> – 30.05.2016 г. – Загл. с экрана.
      14. Kinect [Электронный ресурс] / Википедия. – Режим доступа: www / URL : <https://ru.wikipedia.org/wiki/Kinect> – 30.05.2016 г. – Загл. с экрана.
      15. Функціональне тестування [Электронный ресурс] / Википедия. – Режим доступа: www / URL : <https://ru.wikipedia.org>/wiki/Фильтр\_Калмана – 30.05.2016 г. – Загл. с экрана.
      16. Розширений фільтр Калмана [Электронный ресурс] / Википедия. – Режим доступа: www / URL : <https://ru.wikipedia.org>/wiki/Фильтр\_Калмана – 30.05.2016 г. – Загл. с экрана.
      17. Фільтр частиць [Электронный ресурс] / Википедия. – Режим доступа: www / URL : <https://ru.wikipedia.org>/wiki/Многочастичный\_фильтр – 30.05.2016 г. – Загл. с экрана.
      18. S. Thrun and M. Montemerlo. The graph SLAM algorithm with applications to large-scale mapping of urban structures. Journal of Robotics Research. – 2006. - №25, с.40
      19. ROS [Электронный ресурс] / Википедия. – Режим доступа: www / URL : <https://ru.wikipedia.org>/wiki/ROS\_(операционная\_система)– 30.05.2016 г. – Загл. с экрана.