Гузик Петро Євгенович

Розробка та дослідження системи для побудови карти з одночасним контролем наявного місцерозташування і пройденого шляху в мобільних автономних засобах

Керівник: доц. Митник М.М.

Development and study of a system for mapping with simultaneous control of current location and distance covered in mobile autonomous devices

АНОТАЦІЯ

Кваліфікаційна робота складається з пояснювальної записки та графічної частини (ілюстративний матеріал – слайди). Об’єм графічної частини кваліфікаційної роботи становить 15 слайдів. Об’єм пояснювальної записки складає 110 друкованих сторінок формату А4 (210×297), об’єм додатків – 14 друкованих сторінок формату А4. Кваліфікаційна робота складається з шести розділів, в яких нараховується 26 рисунків та 8 таблиць з даними. В роботі використано 44 літературних джерел. Метою даної кваліфікаційної роботи була розробка та дослідження системи для побудови карти з одночасним контролем наявного місцерозташування і пройденого шляху в мобільних автономних засобах.

Розроблений автономний мобільний засіб матиме можливість будувати карту з одночасним виявленням наявного місцерозташуванням і пройденого шляху. Розроблено функціонал дистанційного керування в ручному режимі використовуючи програмне забезпечення на ПК. Мобільний засіб оснащено камерою і платою-комп’ютером RaspberryPi 3, драйвером двигуна постійного струму L293d та чотирма двигунами постійного струму. Відеопотік з відеокамери надходить на міні-комп’ютер RaspberryPi 3, де покадровообробляється, співставляється з попередніми кадрами і будується карта місцевості за допомогою виявлених ознак в кожному кадрі відеопотоку.

Ключові слова: КАРТОГРАФІЯ, ЛОКАЛІЗАЦІЯ, МАПУВАННЯ, АВТОНОМНИЙ, МОБІЛЬНИЙ, ЗАСІБ, РОБОТ, ВЕБ-СЕРВІС, БЕЗДРОТОВА МЕРЕЖА.

ЗМІСТ

ВСТУП 5
1. АНАЛІТИЧНА ЧАСТИНА.....6
1.1. Походження проблеми SLAM 6
1.2. Аналіз алгоритмів монокулярного SLAM.....9
1.3. Сучасні та альтернативні підходи до вирішення проблеми SLAM.....13
1.4. Аналіз реалізацій SLAM алгоритмів 15
2. НАУКОВО-ДОСЛІДНА ЧАСТИНА 17
2.1. Рекурсивне Баєсове оцінювання 17
2.1.1. Рекурсивна оцінка.....18
2.1.2. Баєсова оцінка 19
2.2. Представлення просторової карти і стану системи.....20
2.3. Баєсова фільтрація 24
2.4. Фільтр Калмана 26
2.5. Розширений фільтр Калмана 28
2.6. Корпускулярний фільтр 29
3. ТЕХНОЛОГІЧНА ЧАСТИНА.....32
3.1. Опис конструкції прототипу.....32
3.2. Програмна реалізація SLAM алгоритму 35
4. КОНСТРУКТОРСЬКА ЧАСТИНА.....39
4.1. Структурні елементи SLAM та симуляція.....39
4.1.1. Симуляція карти.....41
4.1.2. Симуляція давачів одометрії 44
4.1.3. Симуляція енкодерів 47
4.1.4. Симуляція LiDAR 52
5. СПЕЦІАЛЬНА ЧАСТИНА 57

5.1. Вибір двигунів і енкодерів.....57
5.2. Вибір LiDAR.....59
5.3. Raspberry Pi і операційна система.....64
5.4. Операційна система робота.....67
6. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ...71
6.1. Загальна характеристика приміщення і робочого місця.....72
6.2. Аналіз потенційно небезпечних і шкідливих виробничих факторів на робочому місці.....75
6.3. Безпека в надзвичайних ситуаціях.....77
ВИСНОВКИ.....80
БІБЛІОГРАФІЯ.....81
ДОДАТКИ.....86

ВСТУП

В теперішній час все більшого значення набуває питання розробки, досліджень та виробництвабезпілотних транспортних засобів. Сегмент таких засобів пересування розвивається дуже швидко, хоч сама ідея зародилась ще у XX столітті. Так, у 1939 році на виставці GeneralMotors, Норман БельГеддес створив перший автономний автомобіль, який представляв собою електромобіль, керований радіокерованими електромагнітними полями, генерованими намагніченими металевими шипами,які буливбудованів проїжджу частину.Заразбагато транспортних засобів на дорозі вважаються напівавтономними завдяки таким характеристикам безпеки, як допоміжна система паркування та гальмування, а саме безпілотні автомобілімають можливість самостійно керувати, гальмувати та припарковуватися. При тому багато з них оснащені не тількиGPS-системою, а також вдосконалені системоюзондування, які можуть визначати межі смуги, знаки, сигнали та несподівані перешкоди.Науковці та розробники сподіваються на те, що автономні транспортні засоби принесуть із собою кілька різних переваг, але найважливішим із них є, мабуть, підвищення безпеки на дорогах. Кількість ДТП, спричиненихнеправильним керуванням, ймовірно, значно зменшитьсязавдяки безпілотникам, оскільки автомобілі не мають людського фактору.

Метою цієї роботи є створення прототипу транспортного засобу з системою керування ним і реалізація програмного коду, що забезпечує його автономне переміщення. Необхідною умовою реалізації автономно транспортного засобу є застосування алгоритму SLAM. Який використовуватиметься для побудови траєкторії руху у невідомому середовищі.

1. АНАЛІТИЧНА ЧАСТИНА

1.1. Походження проблеми SLAM

Термін SLAM - це скорочена абревіатура одночасної локалізації та картографування (SimultaneouslyLocalizationAndMapping). Спочатку термін був використаний Х'юДюрантом-Уайтом та Джоном Дж. Леонардом на основі попередніх робіт Сміта, Селфа та Чізмена. Деррант-Уайт та Леонард спочатку називали SMAL, але пізніше його було змінено для кращого сприйняття, оскільки smal у перекладі з англійської означає тонкий.

SLAM вирішує задачу побудови карти невідомого середовища за допомогою автономно керованого робота, одночасно орієнтуючись у середовищі за допомогою карти.

SLAM реалізуєнаступні задачі: вилучення орієнтирів (ознак), об’єднання даних, оцінка стану, оновлення стану та оновлення орієнтиру (ознак). Є багато способів вирішити кожну з цихзадач. Але ефективність вирішення цих задач залежить від виду SLAM алгоритму і апаратних можливостей робота, включаючи збір інформації про місцевість.Найбільшої популярності наразі отримали алгоритми якіаналізуютьвідео потік і результати вимірювання відстані давачів. Так як в таких алгоритмах для локалізації і створення карти використовується відео потік,вони отримали назву візуальний SLAM. В основі SLAM технології використовується: базовий SLAM без використання відеокамери та методи аналізу комп’ютерного зору.

Перші роботи які присвячені SLAM технологіїопублікованіу 1986 року на конференції конференції IEEE RoboticsandAutomationConference.В них

використані теоретично-оціночні методи для локалізації та картографування роботів.

Перші роботи по SLAM здебільшого спирались на одометрію та лазерні/ультразвукові давачі які були основним джерелом вхідних даних. Наприклад, є такі роботи з використанням ультразвуку [3], таз використанням лазера [4] для вимірювання відстані. Теоретичні основи і прототип традиційного баєсівського фільтрування на основі SLAM запропоновані в працях [5][6]у 1990-х роках. Основні теорії та метод реалізації ретельно описано роботі [7].

У цей період також почали розвиватись технології комп'ютерного зору - візуальна одометрія (VO, visualodometry) та структура із руху (SFM, structurefrommotion). Візуальна одометрія - це процес оцінки его-руху робота, використовуючи вхід однієї або декількох підключених до нього камер (рис. 1.1) [8]. Історія VO бере свій початок із ранніх робіт [8] [9], фінансованих програмою національного управління з аеронавтикиі дослідження космічного простору для дослідження поверхні Марсу. З метою створення всюдихідних роботів здатних аналізувативласнепереміщення у просторі.

SFM - це техніка діапазонної візуалізації, яка полягає у процесі оцінювання тривимірної структури із послідовності двовимірних зображень, які пов’язані із сигналами руху. На рис. 1.2 показано як програмне забезпечення відстежує та використовує ознаки для оцінки положення фотографії та побудови тривимірної хмари точок.

SFM задача є більш загальною проблемою, ніж VO, і може розглядатися як перша версія візуального SLAM що не працює в режимі реального часу [10]. Одними з перших роботів що використовували SFM які в автоматичному режимі були здатні визначити місцерозташування камери [11] та побудувати тривимірну модель сканованого об’єкта описані в роботах

[12], [13], [14]. Як VO, так і SFM застосовують принцип візуальної геометрії з використанням декількох послідовних кадрів [15].

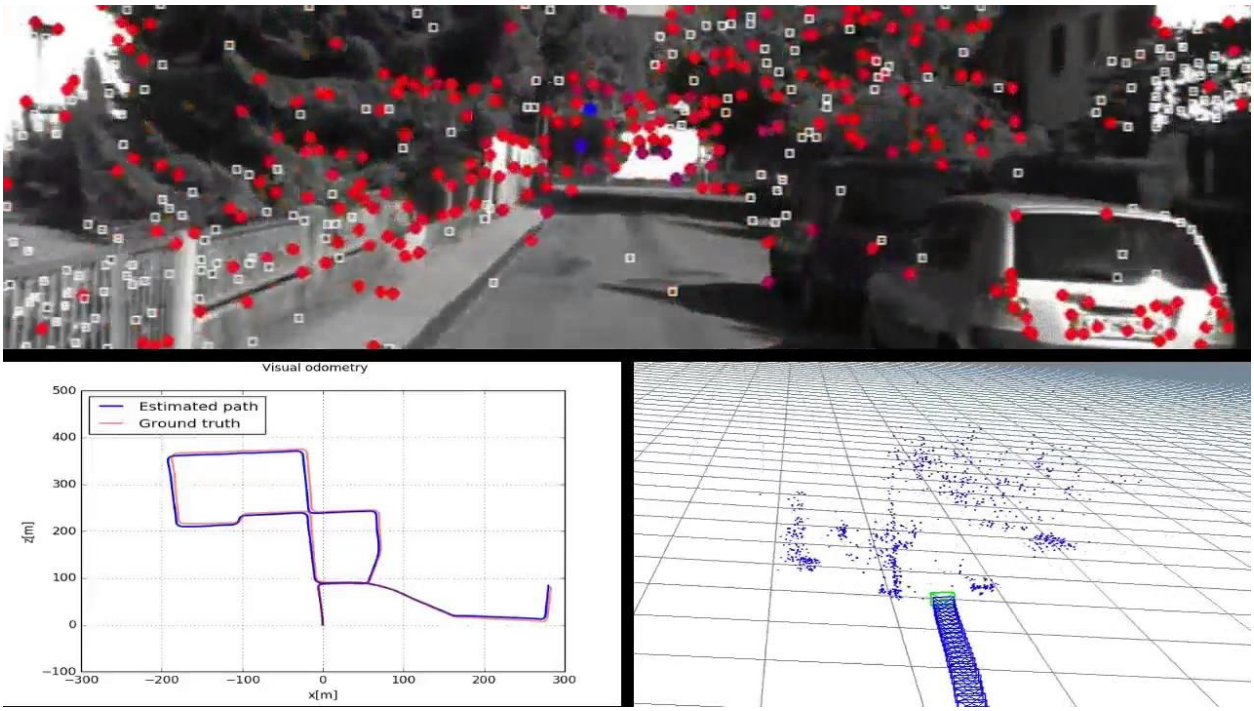


Рисунок 1.1. Демонстрація алгоритму стереовізуальноїодометрії лабораторії в Університеті Мічигану - Дірборн

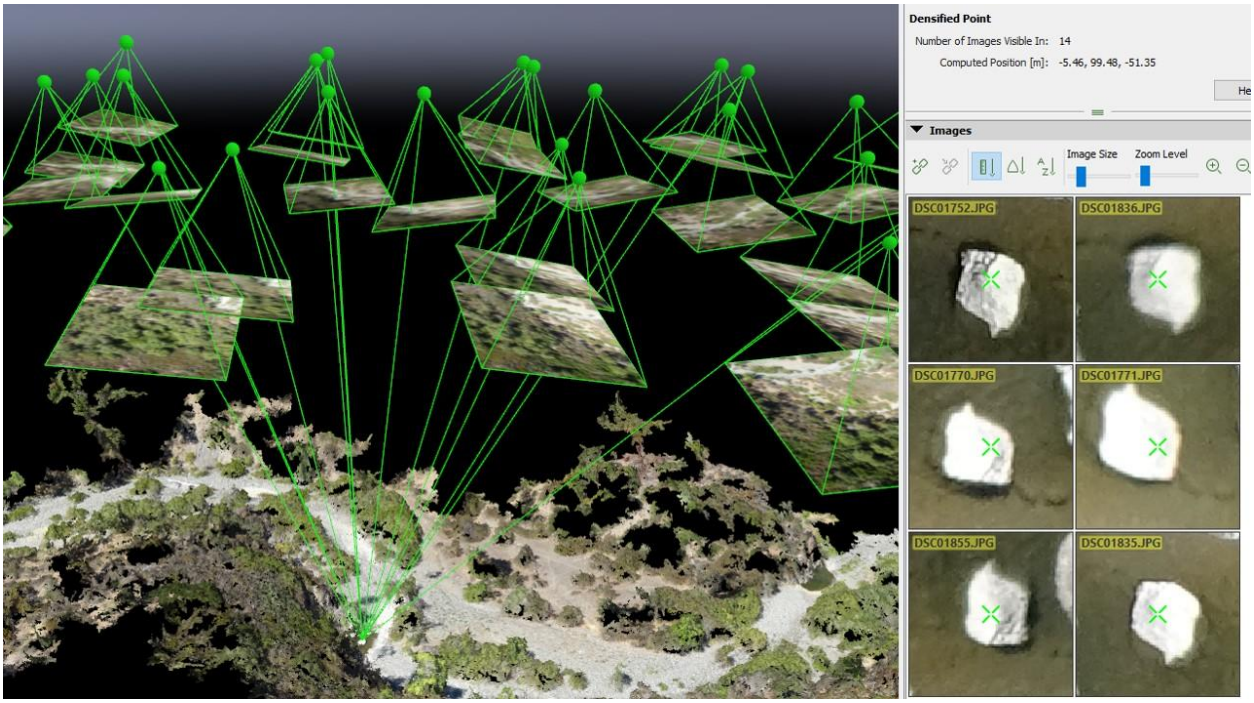


Рисунок 1.2. Результат роботи програми Pix4D, з ідентифікацію одного і того ж самого об’єкта на декількох зображеннях.

До 2000-х років дослідження SLAM у області робототехніки та дослідження SFM у області комп’ютерного зору розвивалися практично не залежно один від одного.

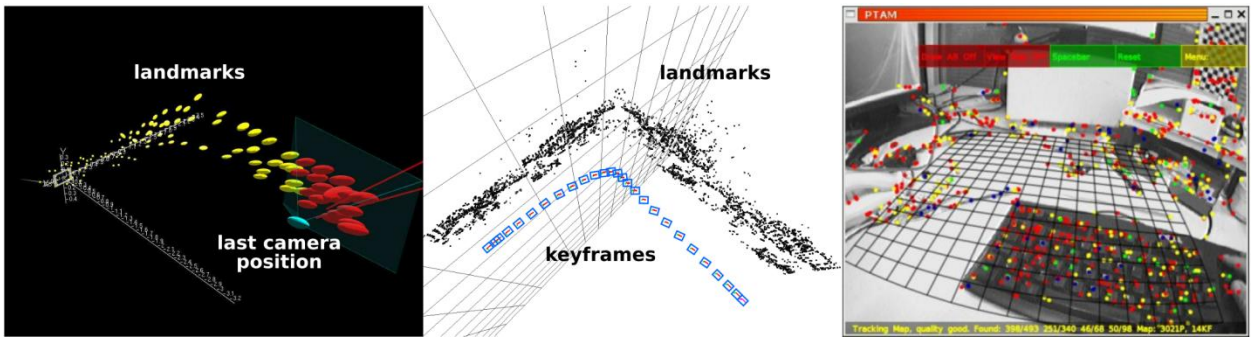
1.2. Аналіз алгоритмів монокулярного SLAM

Відомими дослідженнями є праці [16], [17], [18]у яких вперше використовували комп’ютерний зір iSLAMв автономних роботах.Алгоритм що був розроблений під час вищезгаданих досліджень називається Mono-SLAM. Алгоритм використовує зображення (локальні патчі зображень) для представлення орієнтирів на карті, ітеративноонновлює глибинну карту шляхом пошуку збігу між отриманими кадрами з відеопотоку. Для побудови карти тривимірного середовищавикористовується вектор стану (позиція робота та тривимірні місця розташування) і розширений фільтр Калмана. Ці

дослідження сталиосновою для традиційної реалізації баєсівської фільтрації на основі візуальної реалізації SLAM.

На початку 2000-х років було кілька різновидів SLAM алгоритмів,зокрема досліджували використання корпускулярного фільтрузамість фільтраГауса для визначення позиції камериу візуальному SLAM [19], [20]. Хоча більш відомою роботою на основі корпускулярного фільтру SLAM є FastSLAM[21],де замістьвідеокамер використовували лазерні давачі. В роботі [22], запропоновану методологію, яка реалізується для побудови функціональнихVO та SFM, і до цих пір часто використовується у візуальному SLAM.

Проблема традиційного візуального SLAM, який використовує EKF, полягає в потребі збільшувати обчислювальні потужності у міру зростання складності та розміру карти, оскільки обчислення зростають квадратично із кількістю ознак, що використовуються для побудови карти. Здійснення об'єднання всіх ознак і визначення позиції робота перетворюється у довгий вектор стану та величезну матрицю коваріації. Однак, не завжди є сенс у таких обчисленнях, оскільки не всі кадри отримані з відеопотоку містять спільні об’єкти по яких відбувається співставлення кадрів. Так як матриця коваріації не є розрідженою, здебільшого необхідно багато обчислюваних потужностей і процесорного часу на ітеративне обчислення матриці.



А Б В

Рисунок 1.3. Mono-SLAM і PTAM. А) візуалізований результат MonoSLAM. Б) візуалізований результат відстеження та відображення на основі ключових кадрів за допомогою PTAM. В) демонстрація PTAM.

Досить цікавою роботою, у якійзастосований новий підхід для вирішення проблеми SLAM, була написана дослідниками що працювали над доповненою реальністю (AR). У роботі[24] дослідники запропонували паралельне відстеження та картографування (PTAM) для невеликого робочого простору AR.Алгоритм розроблений та оптимізований для AR додатків і добре працює лише на невеликому робочому просторі без глобального управління картами. PTAM - це один з різновидів алгоритму SLAM, який знаходить, відстежує та відображає ознаки і вважається відносно надійним внаслідок того що може працювати з великою кількістю ознак. Алгоритмзабезпечуєпаралельні обчислення, відстеження та управління картами базуючись на ключових кадрах. Він використовується більшістю сучасних візуальних систем SLAM (наприклад ORB-SLAM [25]) або VO (як SVO [26]).Також він працює в режимі реального часу шляхом розпаралелюванняпроцесів оцінки руху та картографування покладаючись на ефективну корекцію набору ключових кадрів замість баєсівської фільтрації. Такий підхід дозволяє реалізувати точніше визначення позиції камери та побудувати карту.Завдяки цьомуPTAM ефективніший і точніший ніжMonoSLAM.

Також варто згадати про метод побудови карти за допомогою ключових кадрів який належить до так званих методів‘Graph SLAM’. Цей метод розглядає ключові кадри та точки на карті як вузли в графі які оптимізуються для мінімізації помилок вимірювання [27]. Graph SLAM

кращий за EKF SLAM в розрідженості графа і, отже, більш потребує менше обчислювальних потужностей. У [28] запропонований метод графічного SLAM на основі інформаційного фільтру [28]. Метод інформаційного фільтруза реалізацією схожий з методом ключових кадрів, але не набув широкого використання.

PTAM показує досить хороші результати при малому робочому просторі, тоді як проблема побудови великих за обсягомкарт залишається не вирішеною. Рішення для побудови таких карт повинно вирішувати такі задачі: 1) вибір найкращої моделіствоpення карти2) розпізнавання однакових елементів середовища. Друга задача виникає коли карти, зроблені в одному і тому ж місці але з різнимимасштабами, і без використання додаткових алгоритмів,SLAMпомилково вважатиме одинакові елементи з різними масштабами як цілкомрізні,що призведе до неправильної побудови карти. Ця проблема була частково вирішена в роботі FAB-MAP [32]. Таким чином,ефективний вибір найкращої моделі створення картиєнеобхідним дляякісного розпізнавання однакових елементів середовища.

Задача розпізнавання однакових елементів середовища ще далека від вирішення, оскільки проблеми виникають, головним чином, у розпізнаванні в SLAM [33] через зміни освітленості, динамічних або напівдинамічних об'єктів та багатьох інших шумів.

Для роботи з великими за обсягами картами використовується Graph SLAM, що описаний в роботах[29], [30] та [31].В них досліджені теорія та ефективність моделей на основі графів для представлення карти,а такожматематичні операції що дозволять оптимізувати графи для зменшення шумів і фільтрації зайвої інформації.

1.3. Сучасні та альтернативні підходи до вирішення проблеми SLAM

Один з найважливіших компонентів SLAMсистеми є детектор ознак та механізм опису дескрипторів. У більшість сучасних системах використовується вдосконалений ORB [34], що забезпечує виконання алгоритму в реальному часі за допомогою монтованої системи.Однак, якщо немає потреби у виконанні алгоритмів у реальному часі,використовують SIFT або SURF. Ці методи використовують нелінійні бібліотеки для оптимізації методомнайменшихквадратів, такі як g2o [27].

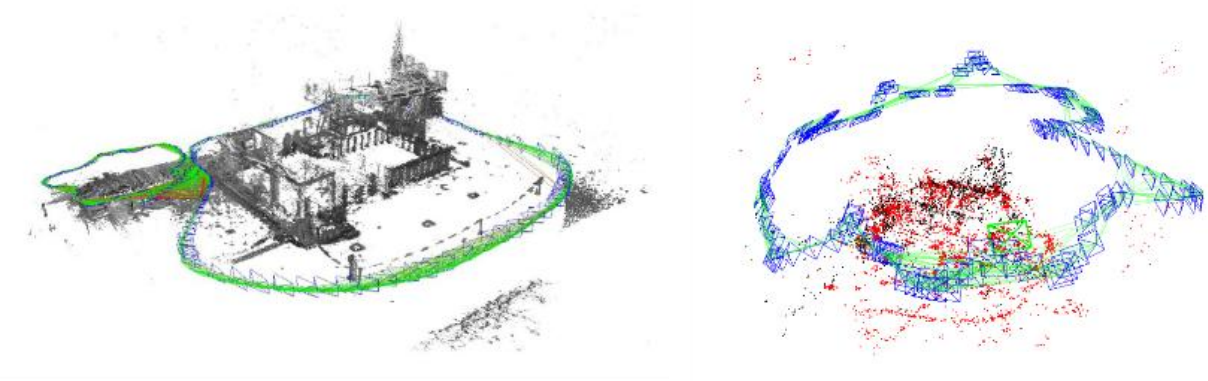
На основі PTAMбуло розробленоORB-SLAM–алгоритм, що базується на ознаках, і концептуально схожий на PTAM, але на практиці демонструє кращі характеристики. Основні удосконалення порівняно з PTAM:

1) реалізація 3 паралельних процесів, а саме відстеження, відображення та розпізнавання однакових елементів середовищадля досягнення послідовної локалізації та відображення.Тоді як у PTAM останній процес відсутній;

2) ініціалізація карти за допомогою моделі яка працює на двох паралельних потоках, що обраховують гомографію та фундаментальний его-рух з використанням RANSAC;

3) використовується детектор та дескриптор ознак ORB, замість патчів зображення у PTAM, що покращує надійністьвідстеження зображень та порівняння ознак при масштабуванні та зміні орієнтації самого зображення;

4) багаторівнева побудова карти, включаючи локальний граф для корекції позиції.



А Б

Рисунок 2.4: Сучасні алгоритми візуального SLAM. А)Карта побудована LSD-SLAM. Б)Карта побудована ORB-SLAM.

Альтернативним рішенням до ORB-SLAM є LSD-SLAM.Його можна віднести до класу так званих прямих методів, тобто алгоритм безпосередньо оцінює стан на основі пікселів зображення, а не на візуальних ознаках що були знайдені у зображенні. Відстеження виконується шляхом вирівнювання зображення із застосуванням алгоритму Губера. Побудова глибинної картиздійснюється за таким самим принципом як і в інших SLAM системах,шляхом використання зворотної параметризації глибини на відносно невеликій парі базових зображень. Оптимізація карт виконується за допомогою оптимізації графів, при цьому існуючі позиції ключових кадрів виражаються в просторі, так як це відбувається у ORB-SLAM. На практиці обробка кожного пікселя в усій послідовності зображень вимагає великих обчислень, саме тому багатьом візуальним системам SLAM які належать до прямих методів, таким як DTAM (DenseTrackingandMapping) [36], потрібен графічний процесор для забезпечення продуктивності в режимі реального часу.

Як вже згадувалося вище, найбільшою проблемою у побудовівеликих за обсягами картзалишається візуальне розпізнавання місця. Цю проблему складно вирішити, використовуючи лише геометричні методи, якізастосовуються у більшості сучасних візуальних SLAM-систем. Подальший розвиток SLAM систем здійснюється у напрямку використання семантичної інформації для аналізу зображення у картографуванні та розпізнанні середовища, наприклад SLAM ++ [38].

Подальше вдосконаленняSLAM алгоритмів дозволяють використовувати їх у комерційнихпродуктах,наприкладGoogleTango та Microsoft Hololens, ARKit, ARCore, 8thWall.

Візуальний SLAM застосовується в багатьох сферах діяльності, особливо у VR / AR. Це висуває вимоги до надійності розпізнавання об’єктів у навколишньому середовищі. Тому роботи у цій сфері є актуальними.

1.4. Аналіз реалізацій SLAMалгоритмів

Останніми роками багато компаній автомобільної промисловості інвестують багато часу і коштів на реалізацію прототипів транспортних засобів з використанням SLAM. Сьогодні на ринку представленіцивільні напівавтономні автомобілі, і представленіпрототипи повністю автономних автомобілів. Очевидно, що для цього необхідно вдосконалити давачі і програмне забезпечення що використовуються в таких автомобілях.

Метою цієї роботи є створенняпрототипу транспортного засобу з системою керування ним і реалізація програмного коду,що забезпечує його автономне переміщення. Необхідною умовою реалізації автономно транспортного засобу є застосування алгоритму SLAM. Який

використовуватиметься для побудови траєкторії руху у невідомому середовищі.

Вибір оптимальногоалгоритму SLAM буде здійснено наоснові проведених теоретичних досліджень. Для досліджень оберемо три різновиди алгоритму: SLAM на основі розширеного фільтра Калмана [12], SLAM на основі корпускулярних фільтрів (FastSLAM) [29] і SLAM на основі графів (GraphSLAM) [11]. FastSLAM реалізований в пакетах ROS gmapping [2] і coreslam [1], а GraphSLAM реалізований в пакетах slam_karto [3]. Алгоритм SLAM повинен одночасно створювати карту навколишнього середовища, а також обчислювати положення транспортного засобу на цій карті. Ці дані будуть транслюватися на інший вузол ROS на ПК, який буде відображати карту та положення автомобіля у реальному часі на графічному інтерфейсі користувача. Для досягнення цілей роботинеобхідно:

1. Порівняти SLAM-алгоритми, щоб прийняти правильне рішення, який з них використовувати.
2. РеалізуватиSLAMалгоритм з використанням обмежених ресурсів (оскільки він буде виконуватися на мобільному процесорі). Перш за все, дослідити алгоритми SLAM, які використовують незначну кількість апаратних ресурсів.
3. Дослідити алгоритми пошуку шляху для SLAM.
4. Здійснити моделюваннясистеми SLAM в Simulinkз урахуванням шумів.

2. НАУКОВО-ДОСЛІДНА ЧАСТИНА

2.1. Рекурсивне Баєсове оцінювання

Ключовим компонентом того, що дозволяє роботу здійснювати правильні переміщення і навігацію, є сканування місцевості. Для навігації роботу потрібна інформація про навколишнє середовище. Цю інформацію надають давачі, якими оснащений автономний транспортний засіб. Однак основна проблема полягає в тому, що дані давача завжди спотворюються

шумом. Це означає, що якщо робот використовує необроблені дані давача, навігація буде неякісною і в результаті будемо отримувати багато помилкових результатів та велику похибку в обчисленнях. Враховуючи усе вищесказане, можна зробити висновок що всі вимірювання необхідно позбавити від наявності шумів івиконати фільтрацію. Один з найбільш популярних підходів до цього - рекурсивне Баєсовеоцінювання, яке використовує імовірнісні моделі для фільтрації шуму.

2.1.1. Рекурсивна оцінка

У рекурсивній оцінці індекс часу позначається як *t*; використовується для позначення стану та вимірювання в моменти, зазначені індексом.Стансистеми позначаєтьсяx, вимірювання позначаємо як z. Наприклад, x_t і z_t позначають значення стану системи які були виміряні в момент часу *t*; $x_{t1:t2}$ та $z_{t1:t2}$ позначають значення стану в момент часу t_1 та t_2відповідно. Стан системи обчислюється за формулою:

$$x_t = g(x_{t-1}, u_t, \epsilon_t) \tag{2.1}$$

де *u*- вхід системи або керуючий сигнал; ε - помилка яку ми отримуємо від присутності шумів; *x*- стан системи.

Вимірювання моделі обчислюються за формулою:

$$z_t = h(x_t, \gamma_t) \tag{2.2}$$

де *x*- стан системи; γ - помилка вимірювання самої моделі.

Використовуючи формулювання теорії імовірності: модель системи та модель вимірювання можуть бути представлені двома умовними розподілами

ймовірностей $p(x_t|x_{t-1})$ (тут ми опускаємо явне представлення вхідного параметра системи) і $p(z_t|x_t)$.

Задача оцінки може бути сформульована наступним чином: при заданих вимірах $z_{1:t}$, обчислюють розподіл, що базується на результатах вимірювання x_t, тобто обчислюють $p(x_t|z_{1:t})$. Таким чином, рекурсивну задачу оцінки можна сформулювати так: враховуючи попередню оцінку $p(x_{t-1}|z_{1:t-1})$ та наступне вимірювання z_t, обчислюємо нову оцінку $p(x_t|z_{1:t})$.

2.1.2. Баєсова оцінка

Розглянемо методологію розв'язання рекурсивної задачі оцінки з використанням баєсової оцінки. Для опису станів застосуємо теорему Маркова: майбутні стани залежать тільки від поточного стану, а не від послідовності подій, які передували цьому. Більш конкретно, припущення Маркова означає, що $x_{(t+1,t+2,\ldots)}$ і $z_{(t+1,t+2,\ldots)}$ не залежать від $x_{1:t-1}$ із заданим x_t (для довільного індексу часу t). Виходячи з теорії Маркова, ми можемо зробити баєсовуоцінкунаступним чином:

$$\begin{aligned} p(x_t|z_{1:t}) &= \frac{p(z_t|x_t, z_{1:t-1})p(x_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} \\ &= \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} \end{aligned} \tag{2.3}$$

де $p(z_t|z_{1:t-1})$ - константа нормалізації, обчислена за формулою:

$$p(z_t|z_{1:t-1}) = \int_{x_t} p(z_t|x_t, z_{1:t-1})p(x_t|z_{1:t-1})\,\mathrm{d}x_t \tag{2.4}$$

Для дискретних випадків інтегральний символ ∫ замінюється символом суми ∑. Вираз $p(x_t|z_{1:t-1})$ у (2.3) ми можемо вивести, застосувавши закон повної ймовірності:

$$\begin{aligned} p(x_t|z_{1:t-1}) &= \int_{x_{t-1}} p(x_t|x_{t-1}, z_{1:t-1}) p(x_{t-1}|z_{1:t-1}) \, \mathrm{d}x_{t-1} \\ &= \int_{x_{t-1}} p(x_t|x_{t-1}) p(x_{t-1}|z_{1:t-1}) \, \mathrm{d}x_{t-1} \end{aligned} \tag{2.5}$$

(2.3) разом з (2.1) можна обчислити нову оцінку$p(x_t|z_{1:t-1})$ зі старої оцінки $p(x_{t-1}|z_{1:t-1})$ та нового вимірюванняz_t. (2.5) зазвичай називають етапом прогнозування поточного стану на основі попередньої оцінки. (2.1) зазвичай називають етапом оновлення, на якомуформується наступна оцінка з урахуванням вимірювання z_t.

Етапи прогнозування і оновлення, є методологією рекурсивної оцінки з використанням баєсовоїоцінки.

2.2. Представлення просторової карти і стану системи

В робототехніці, для визначення станів системи використовується представлення простору-станів. Система яка неперервна у часі і є в представленні простору-станів, тільки лише якщо стан системи можна описати наступними формулами:

$$\begin{aligned} \dot{x} &= f\big(x(t), u(t)\big) \\ \mathrm{y} &= g\big(x(t), u(t)\big) \end{aligned} \tag{2.6}$$

де *x(t)* - вектор стану системи, а *u(t)* та *y(t)* - вхідні та вихідні параметри системи, відповідно. Похідна вектора стану $\dot{x}(t)$ є функцією поточного стану *x(t)* та поточного входу *u(t)*. Аналогічно, вихід системи *y(t)* є функцією поточного стану та поточного входу. Для систем з дискретним часом рівняння простору станів визначаються

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) \\ y_t &= g(x_t, u_t) \end{aligned} \tag{2.7}$$

Для лінійних систем з неперервним часом модель простору-станів може бути описана як

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{2.8}$$

Для лінійних систем з дискретним часом модель простору-станів може бути сформульована як

$$\begin{aligned} x_{t+1} &= Ax_t + Bx_t \\ y_t &= Cx_t + Bx_t \end{aligned} \tag{2.9}$$

Для початку розглянемо загальну процедуру доповнення та перетворення простору-станів у SLAM алгоритмі. Припустимо, що ми маємо існуючу оцінку стану $\hat{x} = [\hat{v}, \hat{m}_1, \ldots, \hat{m}_n]^T$, з n об'єктами на карті, і що ми хочемо доповнити стан новими даними m_{n+1}, базуючись на вимірюванніz_0. Загало

$$\mathbf{P}^{new} = \mathbf{J}\begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_o \end{bmatrix}\mathbf{J}^T \qquad \mathbf{J} = \left[\begin{array}{ccc|c} \mathbf{I} & & & \mathbf{0} \\ \hline \nabla \mathbf{s}_\mathbf{v} & \nabla \mathbf{s}_{\mathbf{m}_1} & \ldots \; \nabla \mathbf{s}_{\mathbf{m}_n} & \nabla \mathbf{s}_{\mathbf{z}_o} \end{array}\right] \qquad (2.10)$$

м,

початкова оцінка буде виведена з комбінації вимірювання та існуючого стану, тобто $\hat{m}_{n+1} = s(\hat{x}, z_0)$, і доповнена коваріація стану має вигляд:

деR_0 - коваріація вимірювання;∇sv = ∂s / ∂v.

Коваріація вводить важливі кореляційні зв'язки між новою точкою та існуючими точками на карті. У випадку появи нової ознаки на карті, лише якобі∇s_v та ∇s_{z0} є ненульовими, і нова точка співвідноситься з тими, що ужекарті. Подібним чином ми також можемо перетворити існуючу точку у представлення$\hat{m}_i^{new} = r(\hat{x})$, і тоді точка визначається як

$$\mathbf{P}^{new} = \mathbf{J}\mathbf{P}\mathbf{J}^T \qquad \mathbf{J} = \left[\begin{array}{c|c|c} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \hline \nabla \mathbf{r}_\mathbf{v} \; \ldots \; \nabla \mathbf{r}_{\mathbf{m}_{i-1}} & \nabla \mathbf{r}_{\mathbf{m}_i} & \nabla \mathbf{r}_{\mathbf{m}_{i+1}} \; \ldots \; \nabla \mathbf{r}_{\mathbf{m}_n} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I} \end{array}\right]$$

де будь-яка зміна розмірності стану відображається на розмірах якобіанаJ.

Процес згортання простору-станів можна легко продемонструвати на прикладі площинної структури що знаходиться у робочому просторі. Дескриптори, що використовуються для виявлення та опису тривимірних точок, найефективніше описують об'єкти, коли їх ініціалізують на площинних, текстурованих поверхнях. Отже, ймовірно, що багато точок на

карті будуть розміщуватись на площинах. Наша ціль визначити набір площин які знаходяться у навколишньому середовищі в якому перебуває транспортний засіб, розташування і орієнтацію цих площин та ввести як додаткові ознаки знайдені площини на карті.

Ми використовуємо алгоритм RANSAC для пошуку площин серед набору точкових об'єктів на карті. Площини генеруються з мінімальних наборів точок, випадково відібраних з підмножини точкових ознак з дисперсією $\sigma_{max}^2 < \sigma_T^2$, де σ_{max}^2 - максимум дисперсії вздовж кожного виміру, а σ_T, відповідним чином вибраний поріг. Мінімальний набір точкових ознак (m_1, m_2, m_3) формує параметри площини наступним чином:

$$\mathbf{p}_o = \mathbf{m}_1 \qquad \mathbf{c}(\theta_1, \phi_1) = \mathbf{m}_2 - \mathbf{m}_1 \qquad \mathbf{c}(\theta_2, \phi_2) = \mathbf{m}_3 - \mathbf{m}_1$$

де точка m_i вважається узгодженою з гіпотезою, якщо її перпендикулярна відстань від площини d, менше d_T, де $d = (m_i - p_0) \cdot n$, а її евклідова відстань від початку площини менше ніж d_{max} .

Другий тест гарантує, що ми ініціюємо площини лише якщо найближчі точки формують з даною точкою залежність у вигляді площини. Відомо, що відносні положення сусідніх точок у системі SLAM зазвичай дозволяють точно визначити площину. Проте якщо похибка у глобальних координатах досить значна, додавання площини до карти на основі набору локальних точок є більш точним способом.

Найбільш придатна площина визначається з урахуванням особливостей внутрішньої точки. Початок координат встановлюється посередині, а параметри орієнтації визначаються з провідних компонентів. Зокрема, якщо вектори положення для l внутрішніх точок із середнім значенням складені в матрицю $l \times 3M$, то власний вектор $M^T M$, що відповідає найменшому власному значенню, єнормаллю до площини.А інші два власні вектори

формують базис в межах площини.Значення λ_{min}, є дисперсією у напрямку нормалі і використовується як показник придатності площини до використання на карті.

Щоб уникнути додавання помилкових площин до карти системи, найкраще підібрана площина, сформована процесом RANSAC, ініціалізується в системі SLAM, якщо $l > l_T$ і $\lambda_{min} < \lambda_T$. Параметри, що найкраще підходять, використовуються для ініціалізації площини у стані, а коваріація оновлюється відповідно до (2.10). Таким чином, і $z_0 = 0$,і$\nabla s_v = 0$.Тоді якобіан відповідно до ознаки внутрішньої точки m_i обчислюється як:

$$\nabla \mathbf{s}_{\mathbf{m}_i} = \left[\frac{\partial \mathbf{p}_o}{\partial \mathbf{m}_i} \;,\; \frac{\partial(\theta_1, \phi_1)}{\partial \mathbf{c}(\theta_1, \phi_1)} \frac{\partial \mathbf{c}(\theta_1, \phi_1)}{\partial \mathbf{m}_i} \;,\; \frac{\partial(\theta_2, \phi_2)}{\partial \mathbf{c}(\theta_2, \phi_2)} \frac{\partial \mathbf{c}(\theta_2, \phi_2)}{\partial \mathbf{m}_i} \right]^T$$

де $\partial c\,(\theta_1, \phi_1)/\partial\, m_i$ та$\partial c\,(\theta_2, \phi_2)/\partial\, m_j$ - якобіан двох власних векторів.

Це гарантує, що параметри площини співвідносяться з рештою стану SLAM через характеристики внутрішньої точки.

2.3. Баєсова фільтрація

Позиція робота зумовлена його поточним станом x_t. Коли система переходить у новий стан x_{t+1}, що обумовлено командою управління u_{t+1}, новий стан буде здійснювати вимірювання z_{t+1}. Зв'язок між елементами управління, станами та вимірами показана на рис. 2.1. Баєсовий фільтр - це алгоритм, який обчислює функцію густини ймовірності для вектора стану x_t. Це робиться у два етапи, етап прогнозування $\overline{bel}(x_t)$ та крок корекції $bel(x_t)$, як показано у формулах (2.14 a, 2.14 b)

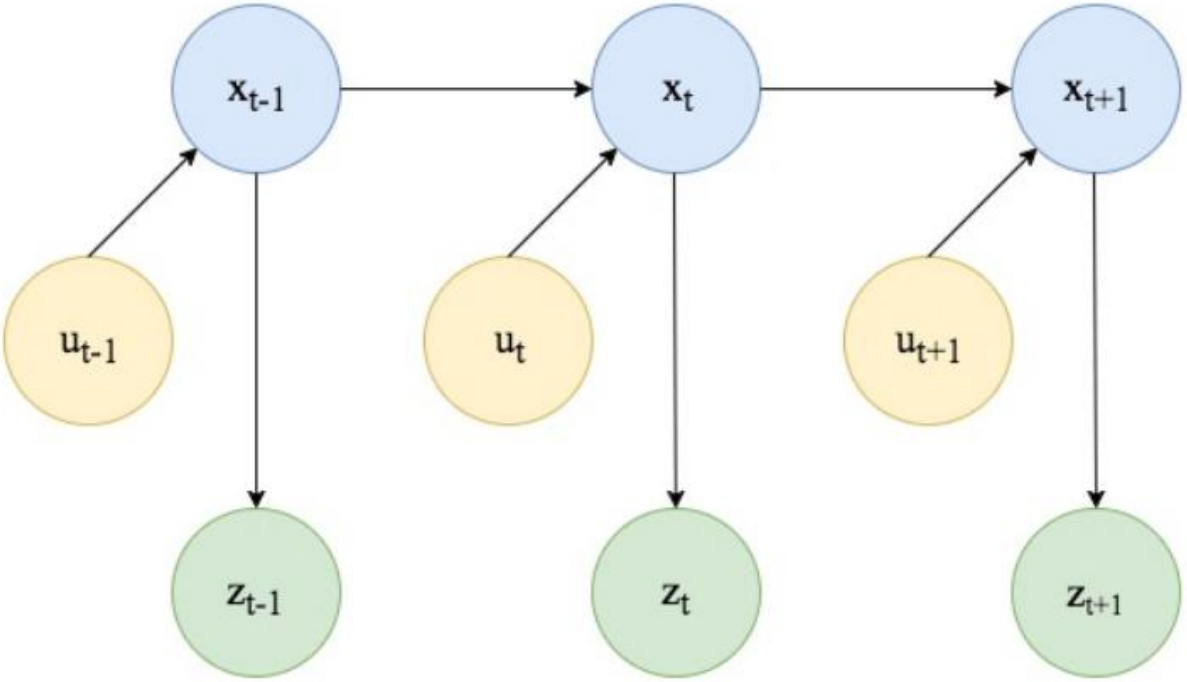


Рис 2.1. Взаємозв'язок між станами, управлінням та вимірюваннями

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx_{t-1} \qquad (2.14\ \text{a})$$

$$bel(x_t) = \eta\ p(z_t \mid x_t)\ bel(x_t) \qquad (2.14\ \text{b})$$

На етапі прогнозування (заданому рівнянням 2.14 а) новий стан x_t прогнозується, використовуючи лише команду управління u_t та попередній стан x_{t-1}. Це прогнозування змінюється на етапі корекції за допомогою даних давачів, з метою отримати точніший результат. Вводиться коефіцієнт масштабування η для масштабування отриманої густини імовірності, щоб скласти інтегральну суму. Псевдокод для виконання алгоритму описаного вище зображено на рис. 2.2.

1: **function** BAYES_FILTER($bel(x_{t-1}), u_t, z_t$)
2: **for** all x_t **do**
3: $\overline{bel}(x_t) = \int p(x_t \mid u_1, x_{t-1})\ bel(x_{t-1})\ dx_{t-1}$
4: $bel(x_t) = \eta\ p(z_t \mid x_t)\ bel(x_t)$
5: **end for**
6: **return** $bel(x_t)$
7: **end function**

Рис. 2.2. Псевдокод алгоритму Баєсової фільтрації

2.4. Фільтр Калмана

Метод, що реалізує фільтр Баєса називають фільтром Калмана, який використовується в лінійних гауссових системах. По суті, він обчислює ймовірність для неперервного стану. Ймовірність стану в конкретний момент часу $bel(x_t)$може бути виражене середнім значенням та його коваріантністю.

Фільтру Кальмана потрібні три передумови для обчислення:

- поточний стан x_t, повинен бути представлений лінійними аргументами, тобто

$$x_t = A_t(x_{t-1}) + B_t u_t + \epsilon_t$$

де A_t і B_t - константи. У рівнянніджерело шуму повинно мати гаусовий розподіл. Поточний стан x_t залежить від попереднього стану x_{t-1}, управління u_t та шуму t.

- Необхідно щоб поточне вимірюванняz_t, та його аргументи мали лінійний характер, тобто

$$z_t = C_t x_t + \delta_t$$

де C_t - константа, а джерело шуму має гауссову природу. Поточні результати обчислень z_t залежить від поточного стану системи x_t та джерела шуму δ_t.

- Останньою передумовою є те, що початковаоцінка$bel(x_0)$ має мати розподіл імовірності по Гаусу.

Сам алгоритм фільтра Калмана, показаний нижче, виражає ймовірність того що отриманий результат точний в поточному стані x_t в певний час t. $bel(x_t)$ виражається середнім значенням μ_t та коваріацією Σ_t, що також є результатом роботи алгоритму.

1: **function** KALMAN_FILTER($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
2: $\overline{\mu_t} = A_t \mu_{t-1} + B_t u_t$
3: $\overline{\Sigma_t} = A_t \Sigma_{t-1} A_t^T + R_t$
4: $K_t = \overline{\Sigma_t} C_t^T (C_t \overline{\Sigma_t} C_t^T + Q_t)^{-1}$
5: $\mu_t = \overline{\mu_t} + K_t (z_t - C_t \overline{\mu_t})$
6: $\Sigma_t = (I - K_t C_t) \overline{\Sigma_t}$
7: **return** μ_t, Σ_t
8: **end function**

Рис. 2.3. Псевдокод алгоритму фільтру Калмана

Алгоритм обчислює середнє значення μ_t та коваріацію Σ_t у рядках 2 та 3. Тоді як A_t та B_t - матриці, помножені на вектор стану x_{t-1}, а вектор керування u_t показує ймовірність переходу стану з лінійними аргументами. Лінійні аргументи є однією з вимог до алгоритму фільтра Калмана. Ймовірність переходу стану використовуються на етапі корекції, див. рядки 4-6. Першим кроком корекції є розрахунок коефіцієнта приросту Калмана, позначеного K_t. Розрахунок середнього μ_t виконується відносно передбачуваного середнього μ_t вимірювання z_t і передбачуваного вимірювання $C_t \mu_t$що помножений на коефіцієнт приросту Калмана.КоваріаціяΣ_t обчислюється з урахуванням коефіцієнта приросту Калмана, а також передбаченої коваріації $\overline{\Sigma_t}$.

2.5. Розширений фільтр Калмана

Розширений фільтр Калмана (РФК) - це підхід до роботи з нелінійними моделями, до яких не можна застосувати звичайний фільтр Калмана. В основному реальні вимірювання та переходи стану часто не є лінійними, тому припускати що всі дані лінійні помилково. Розширений фільтр Калмана описує ймовірність переходу стану та вимірювання ймовірності за допомогою нелінійних функцій

$$x(t) = g(u(t), x(t-1)) + \epsilon(t)$$
$$z(t) = h(x(t)) + \delta(t)$$

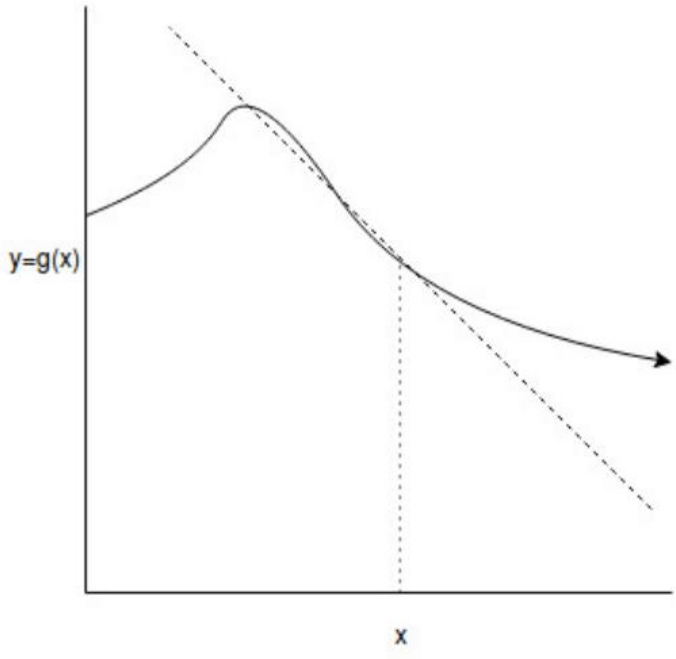


Рисунок 2.4: Лінеаризація нелінійної функції $g(x)$. Пунктирна лінія вказує апроксимацію Тейлора до $g(x)$

Як зазначено вище, РФК використовує нелінійні функції. Результатом роботи алгоритму є апроксимація у вигляді розподілу Гауса з оціненим середнім значенням та коваріацією. Оскільки в алгоритмі використовуються нелінійні функції,вони повинні пройти процес лінеаризації. Яку можна виконати використовуючи, розкладання Тейлора першого порядку навколо точки лінеаризації, як це видно на рис. 2.4. Потім отриманий розподіл Гауса можна обчислити у закритій формі. Це дозволяє отримати характеристики розподілу, такі як середнє значення μ_t та коваріація Σ_t. Враховуючи середнє значення та коваріацію, можна представити оцінку$bel(x_t)$. Псевдокод алгоритму РФК наведено на рис. 2.5.

1: **function** EXTENDED_KALMAN_FILTER($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
2: $\overline{\mu_t} = g(u_t, \mu_{t-1})$
3: $\overline{\Sigma_t} = G_t \Sigma_{t-1} G_t^T + R_t$
4: $K_t = \overline{\Sigma_t} H_t^T (H_t \overline{\Sigma_t} H_t^T + Q_t)^{-1}$
5: $\mu_t = \overline{\mu_t} + K_t(z_t - h(\overline{\mu_t}))$
6: $\Sigma_t = (I - K_t H_t)\overline{\Sigma_t}$
7: **return** μ_t, Σ_t
8: **end function**

Рисунок 2.5. Псевдокод алгоритму розширеного фільтру Калмана

Як видно з рис. 2.5, алгоритм дещо відрізняється від алгоритму фільтра Калмана. Це пов'язано з тим, що EKF використовує нелінійні функції для обчислення середнього значення μ_t та коваріації Σ_t.

2.6. Корпускулярний фільтр

Ще однією реалізацією фільтра Байєса є корпускулярний фільтр. На відміну від розширеного фільтра Калмана, він не використовує параметричну

модель для розподілу ймовірностей. Основна ідея корпускулярногофільтруполягає в тому, що він зберігає набір зразків *M*, де кожен зразок $x_t^{[m]}$ являє собою набір значень потенційного стану.

Корпускулярний фільтр описаний нижче, де $\hat{X}_t$ - прогнозований вектор стану, а X_t - вектор виправленого стану.

1: **function** PARTICLE_FILTER(X_{t-1}, u_t, z_t)
2: $\tilde{X}_t = X_t = \emptyset$
3: **for** m = 1 to M **do**
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\tilde{X}_t = \tilde{X}_t + (x_t^{[m]}, w_t^{[m]})$
7: **end for**
8: **for** m = 1 to M **do**
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to X_t
11: **end for**
12: **return** X_t
13: **end function**

Рисунок 2.6. Псевдокод алгоритму корпускулярного фільтра

Під час проведення вибірки (рядки 3-7 на рис. 2.6) генеруються нові значення для кожного значення $x_t^{[m]}$у фільтрі. Ці значення є тимчасовими, іне будуть оновлені до етапу повторного проведення вибірки. Для кожного$x_t^{[m]}$частинки новий стан обчислюється на основі параметру керування u_t та попереднього стану x_{t-1}. Після цього обчислюється фактор важливості $w_t^{[m]}$. Фактор важливості обумовлений вимірюванням z_t і використовується як міра ймовірності, щоx_tбудевикористаний при повторному проведенні вибірки корпускулярного фільтру.

При повторному проведенні вибірки (рядки 8-10 на рис. 2.6) значення $x_t^{[m]}$оновлюються за допомогою методу випадкового відбору, сформованої на етапі проведення вибірки, де ймовірність вибору $x_t^{[m]}$обумовлена фактором важливості. Повторне проведення вибірки проводиться із заміною, тому кожна частинка може бути вибрана кілька разів.

Існує також інша версія корпускулярного фільтру, яка називається корпускулярний фільтр Rao-Blackwellized. В якому змінні стану представлені величинами що мають Гаусовий розподіл.

3. ТЕХНОЛОГІЧНА ЧАСТИНА

3.1. Опис конструкції прототипу

Основним пристроєм на якому відбувається виконання SLAMалгоритму і який буде знімати показники з усіх давачів у системі є Raspberry PI. Ми обрали цей міні-комп’ютер через доступність, широкий вибір програмного забезпечення,здатністьвиконувати алгоритми які аналізують відеопотік. Для підключення зовнішньої апаратної периферії на комп’ютері є 40-контактний роз’єм GPIO, порти UART, I2C, SPI, а також джерела живлення 3,3 В і 5 В.

Серводвигун - це двигун з давачем зворотного зв’язку,який дозволяє контролювати кутове положення, швидкість і прискорення виконавчого механізму.

LiDAR (LightDetectionandRanging) - це прилад дистанційного вимірювання, який використовує світло у формі імпульсного лазера для вимірювання дальності (змінної відстані) до поверхні як показано на рисунку 3.2. Ці світлові імпульсиу поєднанні з іншими даними, генерують точну тривимірну інформацію про форму поверхні та її поверхневі характеристики.

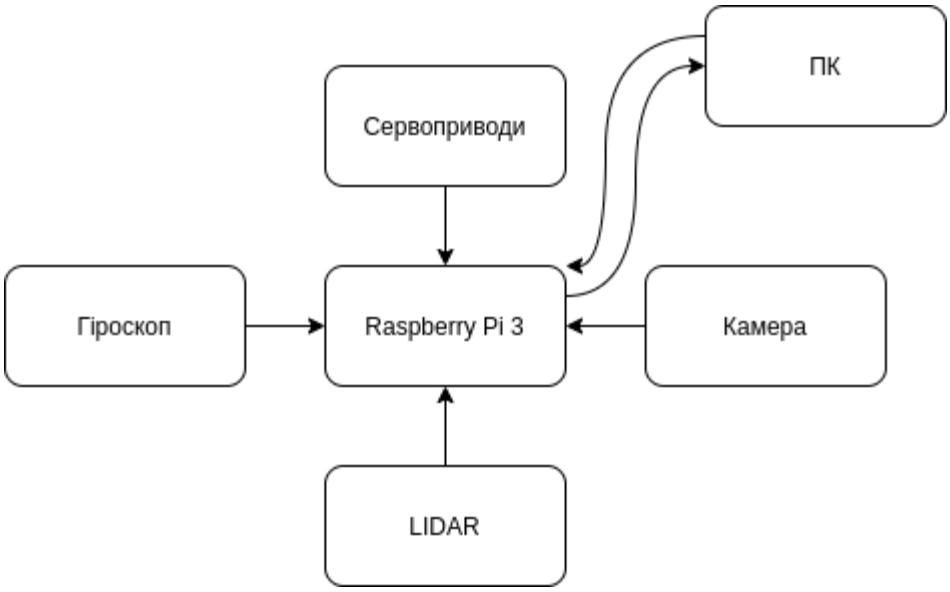


Рис. 3.1. Схема прототипу

Системи LiDARвикористовують в конструкціях у яких необхідно знати глибинну карту навколишнього середовища з високою точністю. У роботі пристрій використовуватиметься для створення глибинних карт при переміщенні робота в невідомому середовищі.

Гіроскоп - це пристрій із обертовим диском або колісним механізмом, що здатний визначати зміну орієнтації відносно інерціального простору. Гіроскоп використовується для визначеннякутового положення робота у невідомому середовищі. У поєднанні з технологією відстеження місцезнаходження можна виявляти та аналізувати рух пристрою в тривимірному просторі.

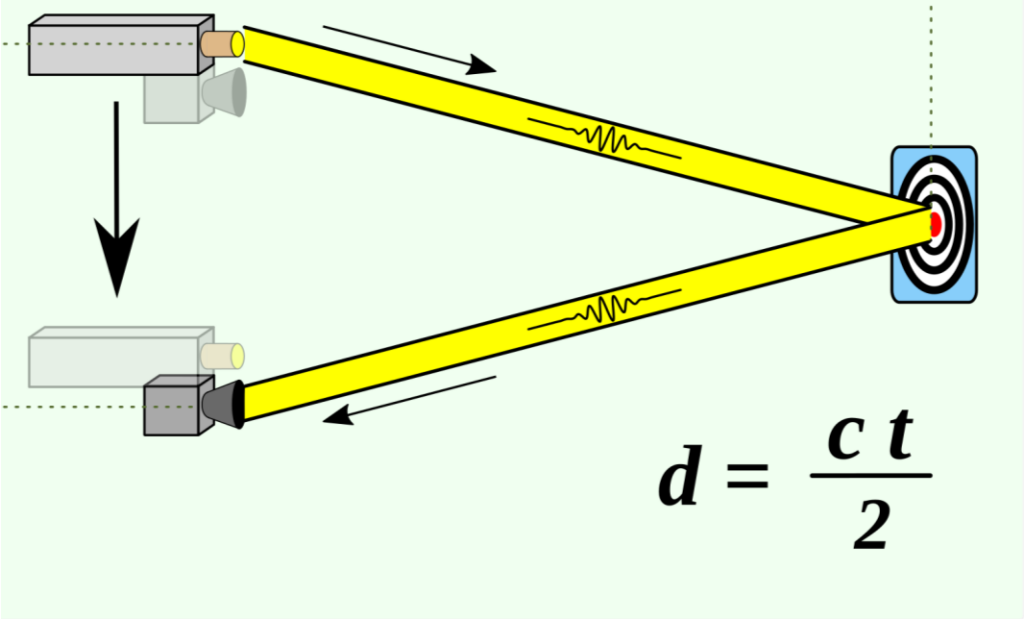


Рис. 3.2. Принцип вимірювання відстані за допомогою лазера

Камера використовується для збору візуальної інформації невідомого середовища. Отримуючи відеопотік з камери на RaspberryPi ми аналізуємо його на наявність ознак що дозволять співставляти наступні отримані кадри з відеопотоку із попередніми. Знаючи величину зміщення між кадрами і прив’язавши це зміщення до фізичного переміщення у невідомому середовищі, це дозволить нам визначити напрямок і дальність переміщення мобільного автономного засобу.

ПК використовується для управління мобільним автономним засобом. На ньому встановлені пакети операційної системи робота (ROS) що містять функціонал підписки/публікації повідомлення на топіки. Кожен топік (тема) відповідає одному пристрою або функції в системі як показано на рисунку 3.3. Наприклад, для того щоб дізнатись інформацію яку нам надає LiDAR потрібно підписатись на тему /sensor_data/lidar. Аналогічно ми отримуємо інформацію від інших пристроїв системи і здійснюємо управління ними.

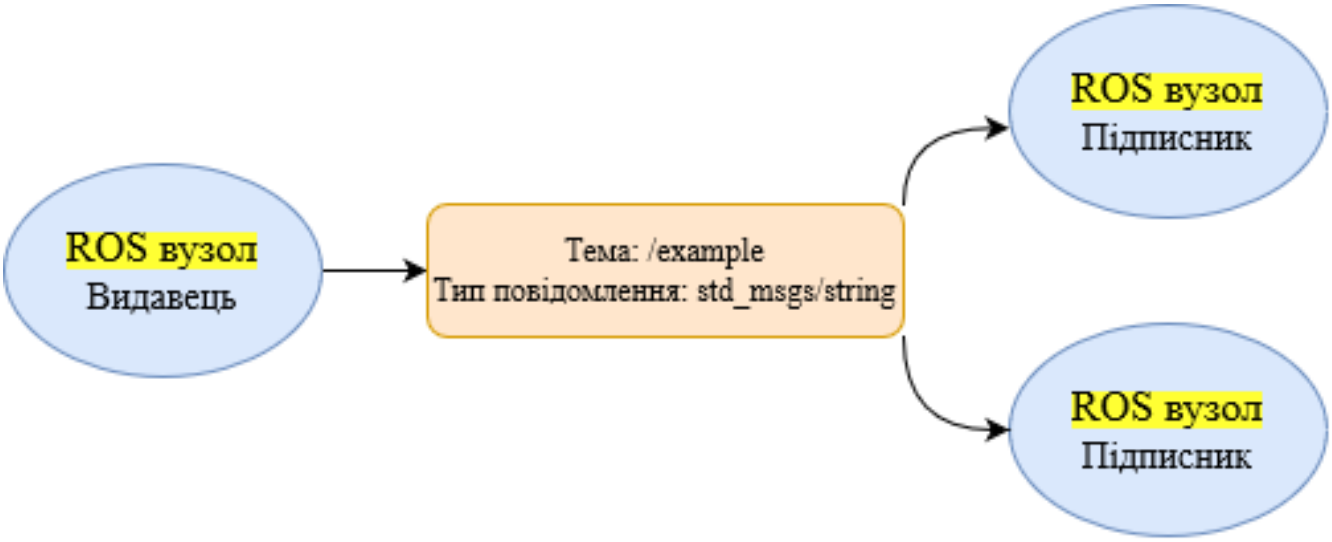


Рис. 3.3. Обмін даними між видавцями та підписниками ROS

Ефективність та точність такої системи ми будемо досліджувати використовуючи програмний продуктSimulink.

3.2. Програмна реалізація SLAM алгоритму

Як буде показано в розділі 4, SLAM алгоритми дуже вимогливі до апаратних ресурсів на яких вонивиконуються. Вимагається як мінімум чотири ядерний процесор з використанням 4 Гб оперативної пам’яті. Так як мобільні пристрої зазвичай не мають таких ресурсів, алгоритм без кардинальної оптимізації буде працювати дуже повільно. Проте, навіть маючи такі ресурси, досить складно знайти баланс між точністю системи та швидкістю її роботи.

З точки зору оптимізації, першочергова задача - зменшити кількість використовуваних ресурсів та зменшити час обробки інформації від давачів під час переміщення робота. Добитись цього результату дозволяють правильні конструктивні рішення про проектуванні. Насамперед це вибір

мови програмування та середовища в якому буде виконуватись програмний код. А також вибір оптимального алгоритму для реалізації. Наступним кроком створені програмних пакети оптимізуютьсяі проводиться подальше тестування системи.

Існує багато концепцій якідопомагають написати оптимальний програмний код. Проте у даній роботі ми створюємо програмний продукт для досить специфічного апаратного забезпечення. Більше того, проектування такої системи ускладнюється наявністю багатьох компонентів що по різному взаємодіютьміж собою. З метою зробити систему логічно розділеною на маленькі самостійні компоненти які не будуть прив'язані до специфічного апаратного забезпечення, ми поділили систему на такі програмні рівні:

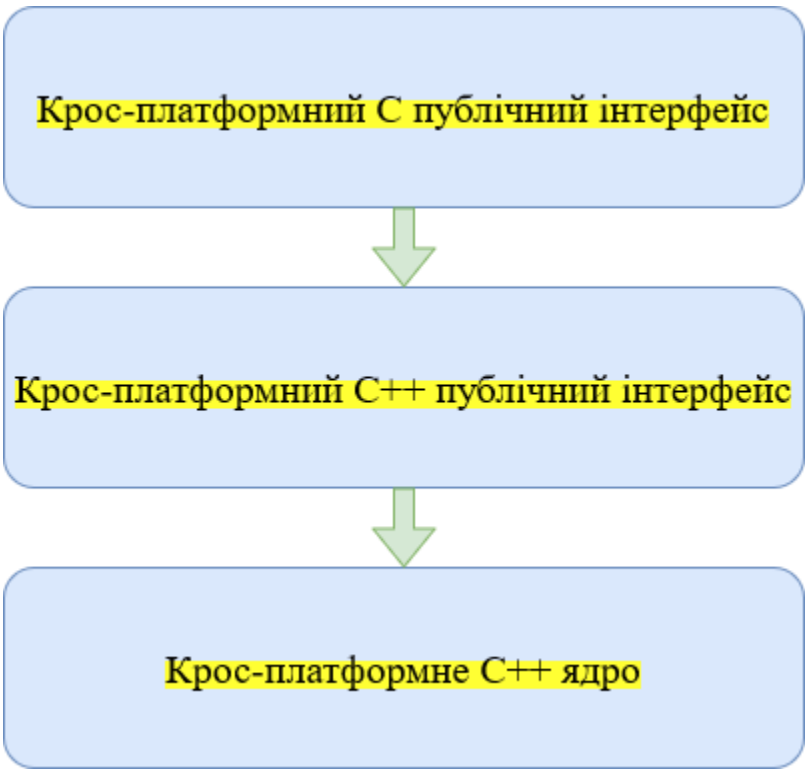


Рис. 3.4. Рівні програмного забезпечення SLAM алгоритму

1. Крос-платформне C++ ядро.

В основі SLAMсистеми лежить крос-платформне ядро C ++. Що містить основу коду на C ++. Оскільки рівень є не залежним від конкретної платформи,код, разом із залежностями, виконується на всіх цільових платформах, використовуючи операційні системи специфічні для кожної платформи. У написаному коді, ядро створене як декілька статичних бібліотек. Ми використали саме статичні бібліотеки для того щоб в майбутньому не встановлювати додаткових залежностей до програмного забезпечення на пристроях де виконуватиметься код.

Також з метою зменшення кількості помилок в коді були реалізовані юніт тести, що перевіряють роботу кожного програмного компонента та виявити помилки ще до запуску програми.

2. Крос-платформний C++ публічний інтерфейс (XAPI)

На цьому рівні ми надаємо загальнодоступний API ядра C++. Застосовано принципи проектування API, що враховують ініціалізацію та деструктори об’єктів, сеанси, конфігурацію, серіалізацію тощо.

Це свого роду SDK, який може бути використаний та виконаний командами без необхідності використання або вивчення повної бази коду ядра C++.

Для кожногофайлу програми на цьому рівні з метою зручного найменування було використаною приставку api у назві. Наприклад, якщо на рівні крос-платформного ядра C++ є файл core.cpp то його інтерфейс на даному рівні описаний у файлі core_api.cpp

3. Крос-платформнийC публічний інтерфейс (XCAPI)

На цьому рівні ми визначаємо інтерфейс у стилі C до загальнодоступного API C++, визначеного у попередньому рівні. Інтерфейс у

стилі C заснований на глобальних автономних функціях без класів C++. Ці функції будуть використовувати API C++ всередині своїх реалізацій .cpp. Іноді API на основі функцій передає типи C++ як аргументи. У API створено функціоналдля управління часом існування об’єкта. Також на даному рівнівирішина проблема перевантажених функцій C++ шляхом створення окремих функцій C для виклику перевантаженої функції C++.

Для найменування файлів програми на цьому рівні було використано приставку c_api. Наприклад, якщо на рівні крос-платформного C++ публічного інтерфейсує файл core_api.cpp то його інтерфейс мовою C на даному рівні описаний у файлі core_c_api.cpp

4. КОНСТРУКТОРСЬКА ЧАСТИНА

4.1. Структурні елементи SLAM та симуляція

Загалом архітектуру можна розділити на програмне забезпечення для одноплатного комп’ютера (RaspberryPi 3) та клієнта (ноутбук), що обмінюються повідомленнями через мережу Інтернет. Головний керуючий пристрій RaspberryPi 3 використовується для управління сервоприводами і зчитування показників із давачів. А клієнт містить графічний інтерфейс який здійснює моніторинг.

Компоненти системи взаємодіють з RaspberryPi 3 через порти за допомогою драйверів та бібліотек. Серводвигуни взаємодіють за допомогою драйвера і каналу ШИМ. Вбудований драйвер длясервоприводів працює таким чином, що користувач може віддалено вказатинапрямок (вперед, назад, ліворуч або праворуч). Веб камера взаємодія через USB порт і для зчитування відеопотоку використовується бібліотека OpenCV.

Транспортний засіб, давачі та навколишнє середовище автомобіля були змодельовані в Simulink. Створена модель у Simulink показана на рис. 4.1. Модель не може бути протестована в режимі реального часу через надто великий час обчислення.

Вхідний модуль забезпечує вхідні параметри для системи. Передача параметрів здійснюється в двох режимах - клавіатура та файл. У режимі клавіатури кут повороту та швидкість руху автомобіля регулюється за допомогою клавіш зі стрілками на клавіатурі. У режимі файлу швидкість та кут нахилу визначаються файлом з розширенням .mat. Файли .mat записуються під час роботи моделі в режимі клавіатури. Типовим робочим процесом під час використання цієї моделі буде першочерговий запис

інструкцій для мобільного транспорту у файл .mat, з метою використання записаного .matфайла для повторних експериментів.

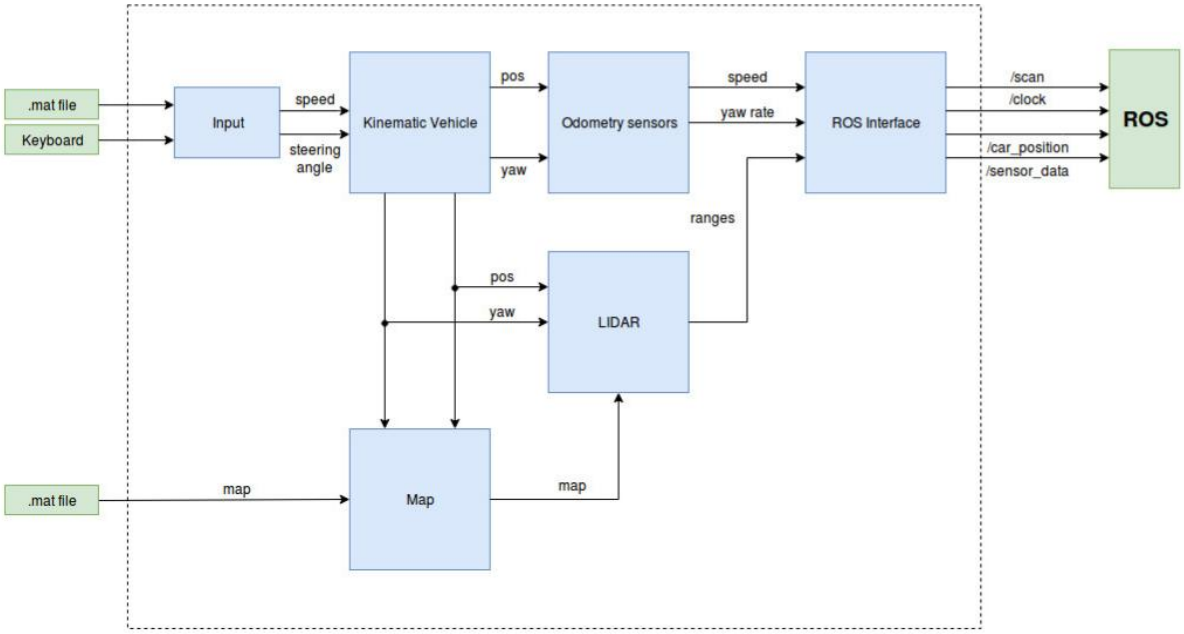


Рисунок 4.1. Блок-схема моделіySimulink

Кінематичний транспортний засіб змодельований у Simulink, показаний на рис. 4.2, складається з п’яти підсистем, кожна з яких описана в розділах нижче.

У моделі вхідним сигналом є бажана швидкість автомобіля. Значення пропускається через фільтр першого порядку для моделювання прискорення та уповільнення автомобіля. Час, необхідний для того, щоб транспортний засіб перейшов із нерухомого стану на повну швидкість, вимірювали для того, щоб обчислити постійну часу τ фільтра першого порядку.

Для сервоприводу була використана модельу якої сигналом керування серводвигуном є значення від -100 до 100, яке потім через таблицю перетворюється в значення яке знаходиться в діапазоні від -45 ° до 45 °, що є максимальним кутом повороту автомобіля. Фільтр першого порядку

використовується для моделювання затримки рульового управління автомобіля, а константа τ обчислювалася шляхом вимірювання часу, необхідного для повороту колеса автомобіля від 0 ° до 45 °.

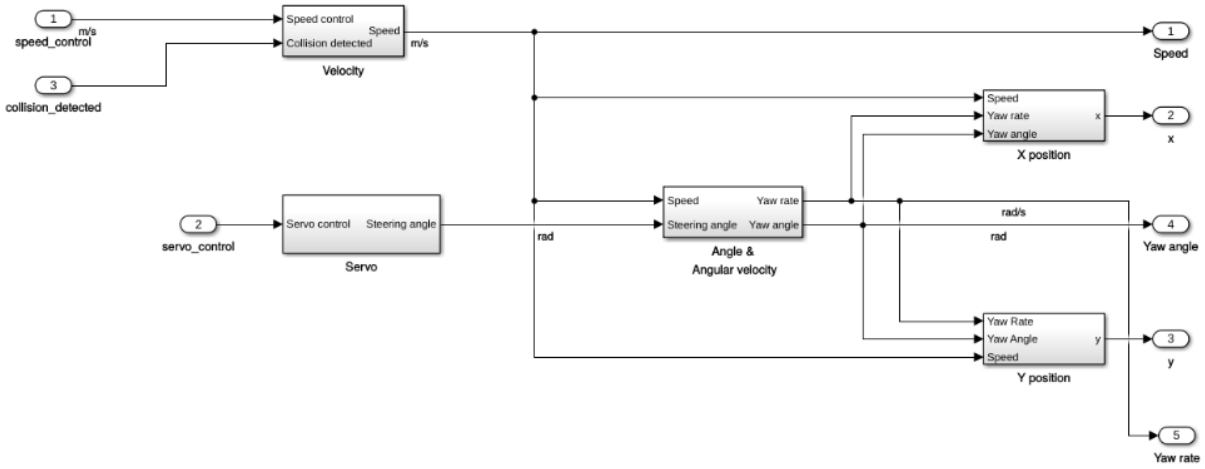


Рисунок 4.2. Модель кінематичного транспортного засобу у Simulink

4.1.1. Симуляція карти

Для того, щоб імітувати середовище у якому перебуватиме транспортний засіб, потрібно створити модель. Для ініціалізації моделі потрібно задати два параметри; розміри та масив стін. Розмір - це матриця розміром 1 на 2, де перший індекс - це ширина карти, а другий - висота карти. Масив стін - це матриця розміром *n* на 4, яка описує стіни на карті, і в якій кожен рядок містить координати однієї стіни. Стіни у симуляції можуть бути діагональними, вертикальними і горизонтальними. Параметри карти зберігаються у файлі .mat, який потім може бути прочитаний файлом моделі ySimulink.

Карта реалізована у двох S-функціях, які називаються vehicleMap та drawMap. Перша функція обчислює всі необхідні значення для відображення поточного положення та переміщення по заданій траекторії транспортного засобу на карті, а друга малює усі дані на графіку MATLAB. На цій ділянці положення та переміщення транспортного засобу представлені трикутником, так як на рис. 4.2. Вхідні дані карти транспортного засобу показані в таблиці 4.1, а діаграма моделі Simulink - на рис. 4.3.

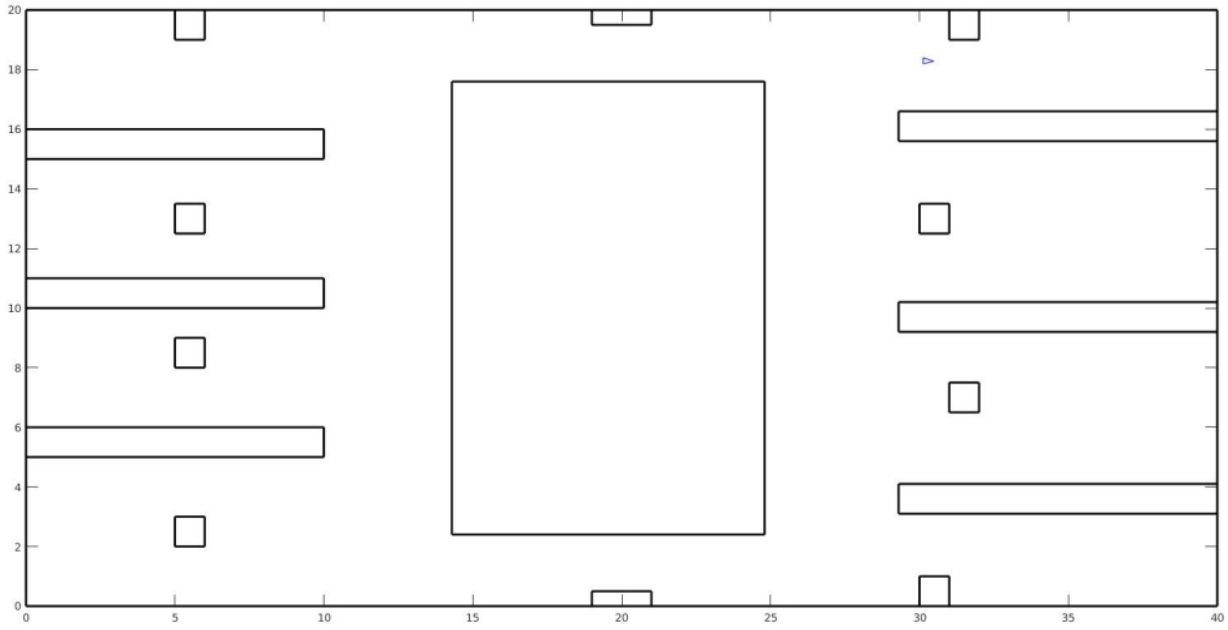


Рисунок 4.2. Карта, сформована S-функцією vehicleMapy Simulink

Таблиця 4.1

Вхідні дані для vehicleMap

Входи	Напрямок	Розмір	Визначення
Координати	Вхід	2	Поточні координати ТЗ
Відхилення	Вхід	1	Відхилення руху ТЗ
Координатитрикутника	Вихід	6	Координати на яких зображається трикутник на мапі
Колізія	Вихід	1	Логічний сигнал що показує наявність зіткнення із перешкодою
Ініціалізація	Вихід	1	Логічний сигнал що показує статус ініціалізації карти

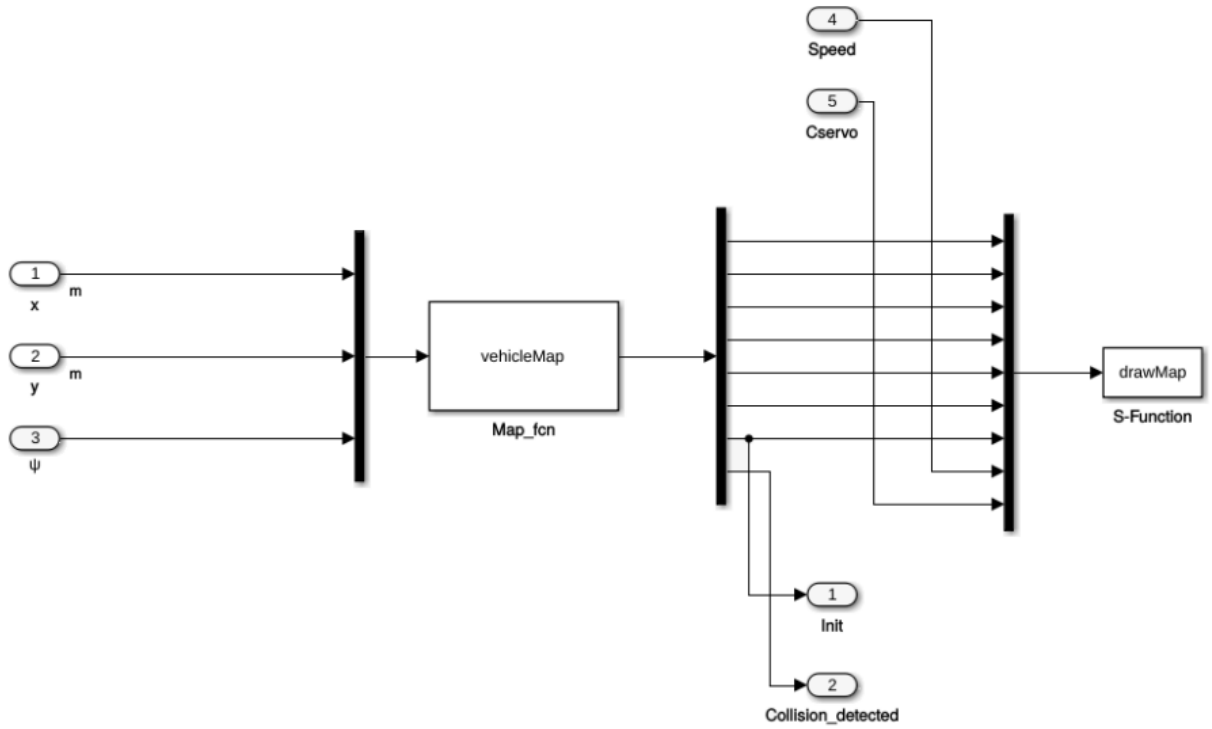


Рисунок 4.3: Підсистема карти у Simulink

4.1.2. Симуляція давачів одометрії

Модель на вході отримує кутову швидкість, отриману від моделі мобільного транспортного засобу, і на виході видає значення яке би вимірявдавач. Вимірювання містять шуми що робить модель наближеною до реальних вимірювань. Модель розроблена з урахуванням фактичних вимірювань значень з гіроскопа. Значення вимірювались в момент коли транспортний засіб стоїть нерухомо на місці. І на основі цих вимірювань змодельована середня похибка. Псевдокод дляалгоритму виведення середньої похибкивимірювання гіроскопа на рис. 4.4.

```
for Gz in array do
    count = count + int(Gz)
    number = number + 1
end for
ψ = (count * 250 * 3.14/(number * 32768))
```

Рисунок 4.4. Псевдокод для алгоритму виведення середньої похибки гіроскопа

Також для симуляції у Simulink реалізовано гауссовий шум і похибку. Це видно на рис. 4.5. Модель на виході видає вимірювання кутової швидкості з шумом і похибкою, виражену в рад/с, з метою симулювати фізичний гіроскоп, реалізований у транспортному засобі.

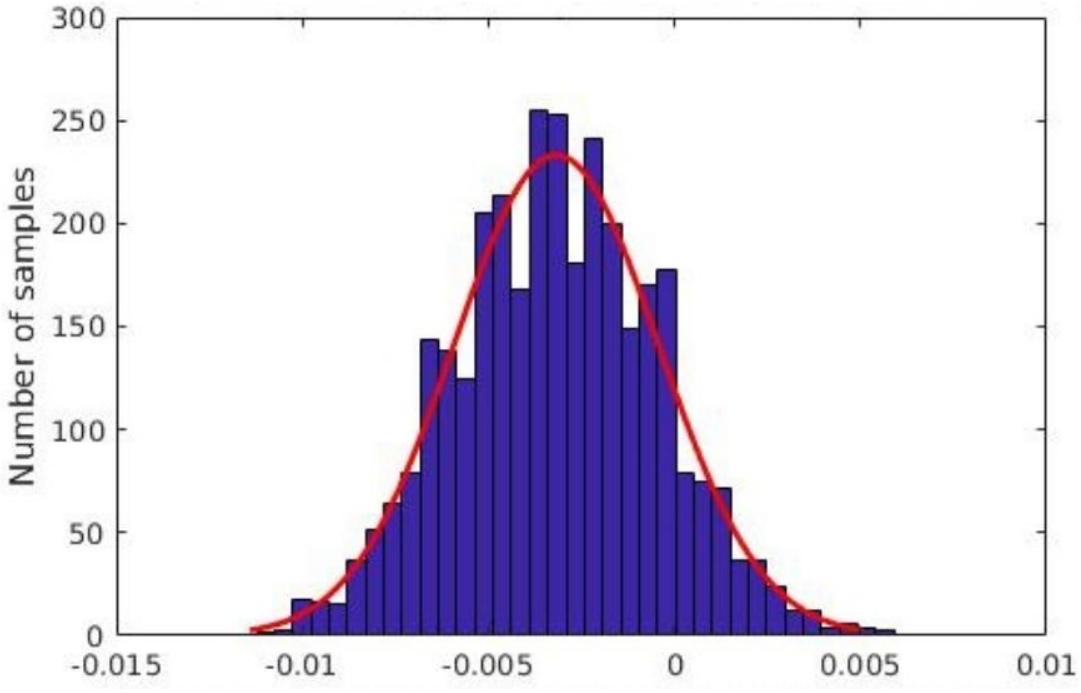


Рисунок 4.4. Стовпчаста діаграма, що відображає шум гіроскопа, крива показує розподіл Гауса

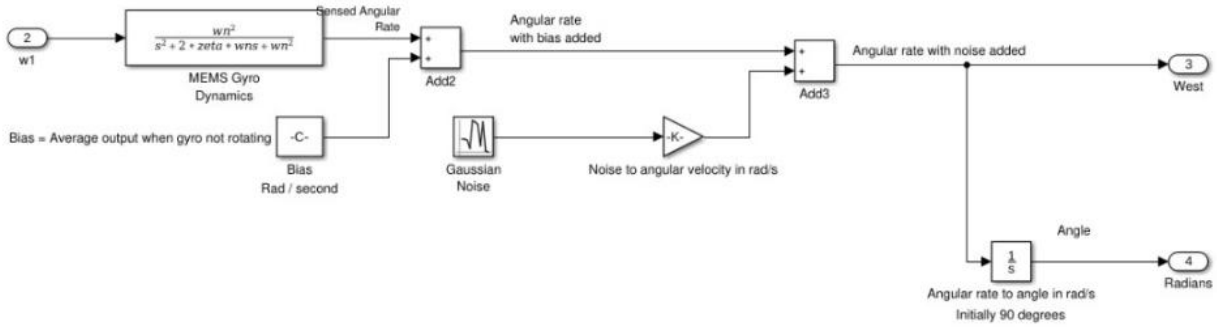


Рисунок 4.5: Реалізація гіроскопа в Simulink

Оцінка моделі була здійснена шляхом використання тестового середовища в Simulink. Перш за все, у Simulink був реалізований користувацький блок ROS, який отримує фактичні значення від фізичного транспортного засобу. Значення осі z гіроскопа вимірюються і обчислюються

як кутова швидкість в рад / с. Порівняння між моделлю давача Simulink та значеннями фізичного давача можна здійснити в середовищі Simulink. На рис. 4.6 нижче показано тестове середовище.

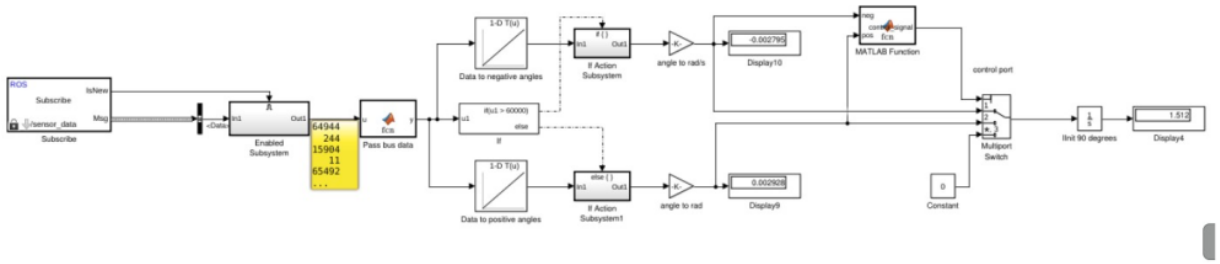


Рисунок 4.6. Тестове середовище гіроскопа ySimulink

На рис. 4.7 показана кутова різниця між моделлю гіроскопа та вимірюваннями у реальному часі на фізичному гіроскопі.

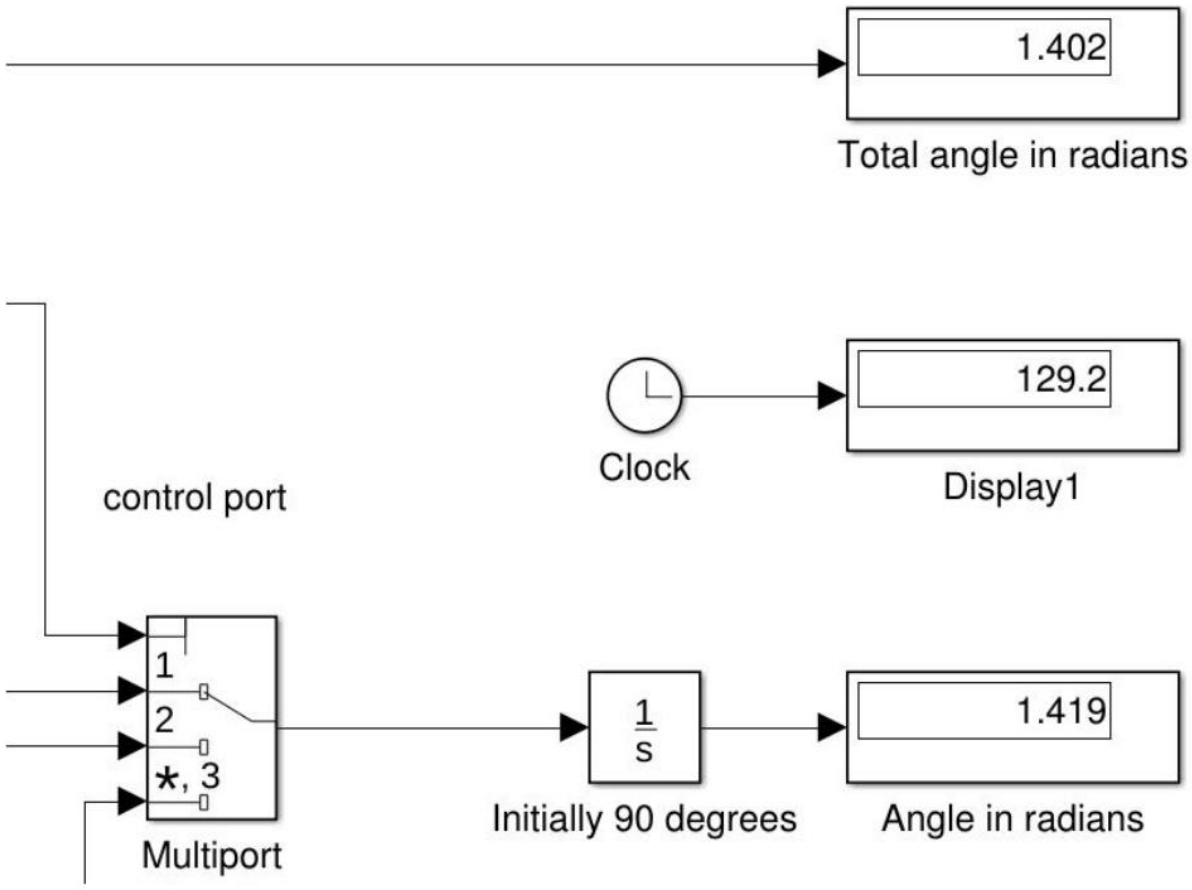


Рисунок 4.7: Порівняння моделі гіроскопа та фізичного гіроскопа у Simulink.

На рис. 4.7, верхній дисплеї відображає виміряне значення на виходіз моделі, нижній дисплей - кут, виміряний фізичним гіроскопом. При виконання вимірювань початкова позиція змодельованого і фізичного гороскопів булаініціалізованана 1,507 рад, 90 градусів.

4.1.3. Симуляція енкодерів

УSimulink,модель енкодеріврealізована в моделі автомобіля. Моделі створені для вимірювання вихідної потужності енкодерів, прикріплених до

транспортного засобу зображені на рис. 4.11. З моделі ми отримуємо дані що містять оцінку швидкості руху транспортного засобу. Для реалізації моделідавача використовувались вимірювання що були зроблені під час випробувань фізичних енкодерів.

У таблиці 4.2 наведені вимірювання, проведені під час тестуванняфізичних енкодерів. Результати кожного вимірювання сильновідрізняються, це здебільшого пов'язано з тим, що двигун на автомобілі перезапускається випадковим чином під час роботи.

У таблиці 4.2, *MI* – це номер надісланої команди двигуну, μ - середня виміряна швидкість; σ - стандартне відхилення; σ^2 – дисперсія (середньо квадратичне відхилення); [s] - час проїзду автомобіля на задану відстань, вимірюється в секундах; [cm] - відстань яку проїхав транспортний засіб під час вимірювання; [tq] - відрізок часу, в якій швидкість транспортного засобу перевищує поріг 90% від максимальної, отже параметр не враховує частину часу, протягом якого транспортний засіб прискорюється; [vq] - відстань яку проїхав автомобіль, ігноруючи дистанцію пройдену під час розгону, до стабільної швидкості (90% від максимальної). На рис. 4.8 показана швидкість, виміряна енкодерами.

Модель була реалізована на основі значень, наведених у таблиці 4.2. Похибка моделювалася за допомогою значень які отримані в результаті вимірювань фізичних енкодерів.У таблиці 4.3, показані результати виміряної швидкості під час фізичних випробувань та результати які були отримані в результаті симуляції.

На рис. 4.10 наведено графік швидкості автомобіля що вимірювалась давачем реалізованого в Simulink. Графік показує швидкість автомобіля в см / с. Також, швидкість автомобіля залежить від параметрів введенихкористувачем у модель. Тому залежно від значення введених

користувачем може мінятись швидкість транспортного засобу. Крім того, реалізована модель давача включає джерело шуму, яке залежить від поточної швидкості моделі автомобіля. Результати моделювання шумівенкодера приведені на рис. 4.10.

Таблиця 4.2

Результати вимірюванняенкодерів

MI	μ	σ	σ^2	$[s]$	$[cm]$	$[tq]$	$[vq]$
5	4.1	0.98	0.96	6.59	58	0.83	0.96
5	4.1	1.1	1.2	4.3	33	0.83	0.95
5	4.4	0.79	0.62	4.6	40	0.83	0.94
5	4.1	0.31	0.1	4.15	34	0.44	0.73
10	10.9	0.34	0.11	5.13	94	0.73	0.89
10	10.7	0.94	0.89	6.4	122	0.76	0.87
10	11	0.91	0.81	5.3	103	0.83	0.87
10	10.7	0.47	0.22	3.55	50	0.38	0.65
15	15.9	0.94	0.89	6.1	177	0.8	0.93
15	15.7	1.24	1.56	3.6	103	0.74	0.85
15	15.9	1.51	2.26	6.35	196	0.75	0.88
15	15.9	1.1	1.21	5.2	144	0.72	0.88
20	22.1	1.32	1.75	4.8	190	0.72	0.84
20	21.5	1.02	1.05	4	142	0.56	0.73
20	20.2	1.99	3.99	3.4	122	0.58	0.80
20	19.8	0.95	0.9	4.65	167	0.61	0.79

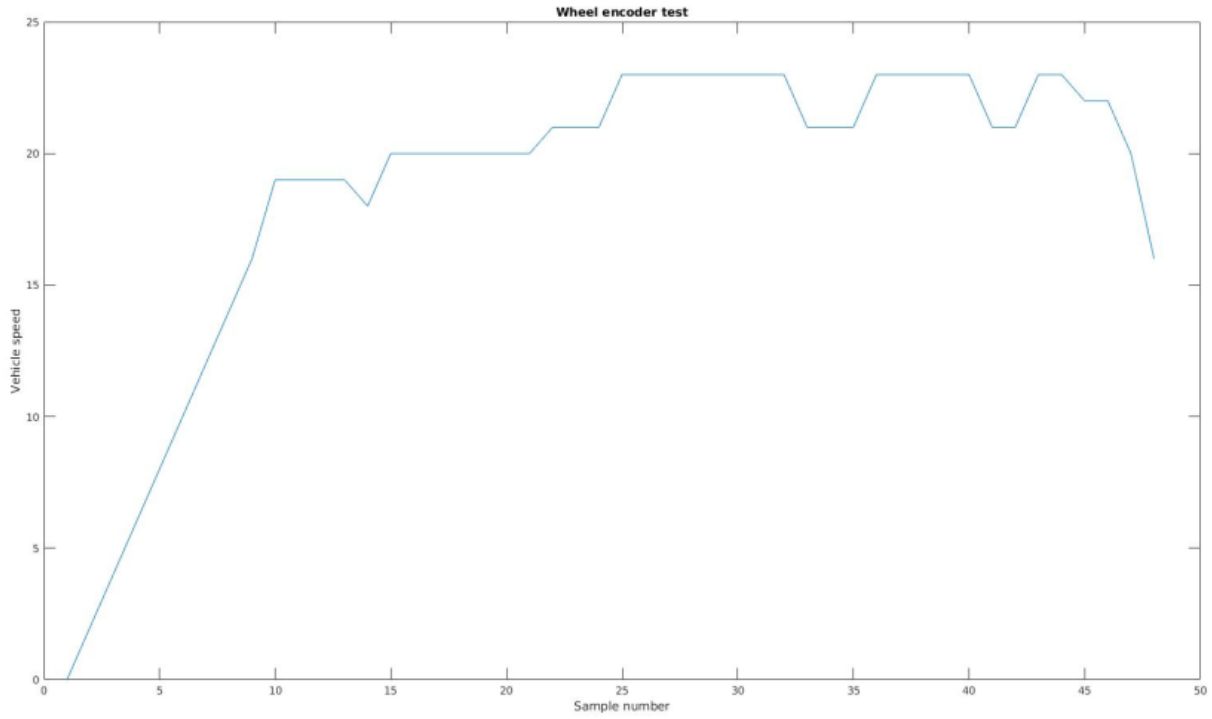


Рисунок 4.8. Графік швидкості виміряноїенкодерами у Simulink

Таблиця 4.3

Значення змодельованої у порівнянні з виміряною швидкістю

Номер команди	0	5	10	15	20
Змодельована швидкість	0	9.12	23	35,6	48
Виміряна швидкість	0	9.82	23.3	36	49

На рис. 4.11 нижче показано реалізація моделі в Simulink. Модель містить блок перетворення виміряної величини з м/с в см/с; блок який генерує шум що має імовірнісний розподіл Гауса.

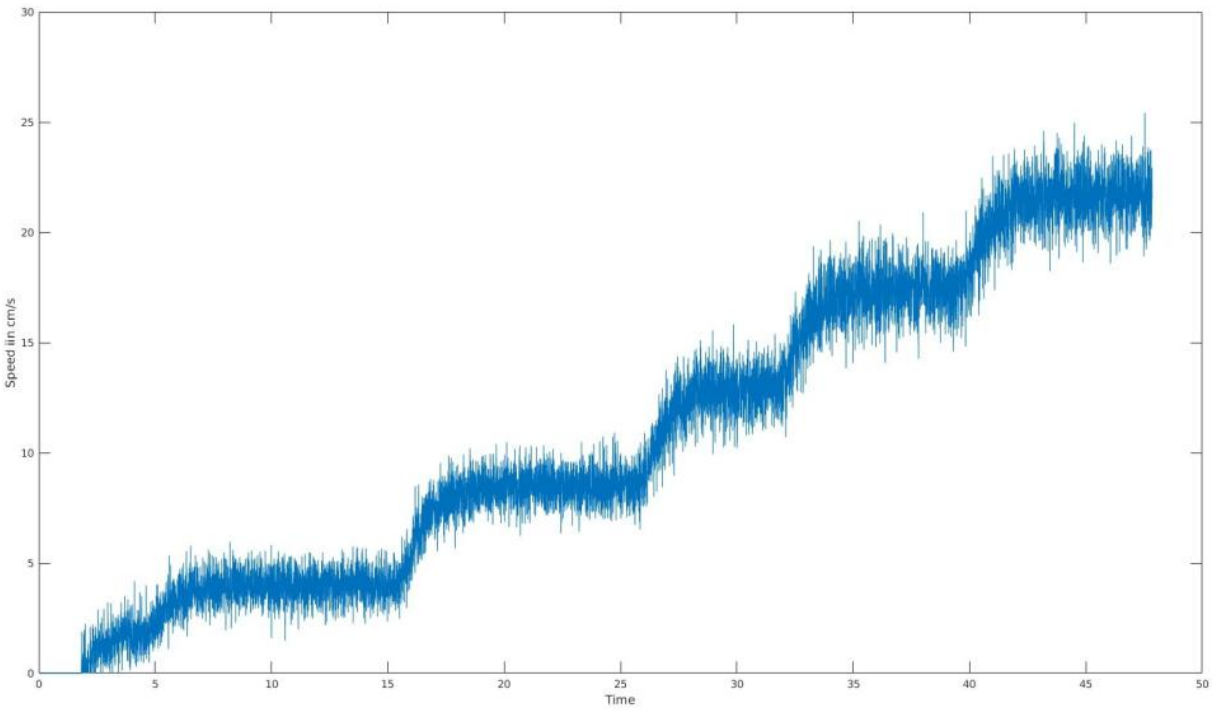


Рисунок 4.10: Шум енкодера побудованого у Simulink

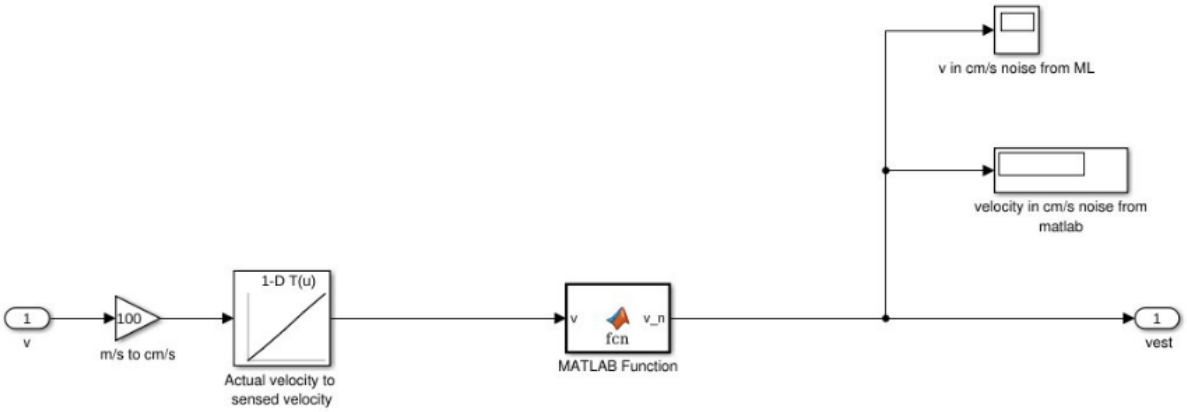


Рисунок 4.11: Модель енкодера реалізована уSimulink

4.1.4. Симуляція LiDAR

LIDAR реалізований як S-функція, яка називається lidar, і моделює пристрійRPLIDAR.S-функції у Simulinkдозволяють описувати блок за допомогою мови програмування.Нам необхідна ця функція для використання алгоритму побудови глибинної карти в Simulink. Входи функції lidarнаведені в таблиці 4.3.

Таблиця 4.3

Входи S-функцій LIDAR

Входи	Напрямок	Розмір	Визначення
Координати	Вхід	2	Поточні координати ТЗ
Відхилення	Вхід	1	Відхилення руху ТЗ
Зміщення LiDAR	Вхід	1	Вертикальне зміщення місцерозташування LiDAR
Колізія	Вихід	1	Логічний сигнал що показує наявність зіткнення із перешкодою
Відстань	Вихід	360	Логічний сигнал що показує статус ініціалізації карти

До того, як VehicleMapініціалізує карту, функція знаходиться в режимі очікування. Наступним етапом вимірюється відстань. До виміряного значення додається шум та додаткові похибки пристрою, щоб результат був близький до фізичного RPLIDAR. Отримане значення записується і зберігається у векторі стану. Алгоритм роботи блоку в Simulinkнаступний: LIDAR повертають на один градус, щоб підготувати його до наступного вимірювання, яке відбудеться після затримки; затримка дорівнює часу,

необхідному для обертання RPLIDAR на один градус;якщо LIDAR досягнув кінця свого циклу обертання, вектор стану подається на вихід блоку, а потім очищується. Блок-схема функції у Simulink представлена на рис. 4.13 нижче, а схема підсистеми - на рис. 4.12.

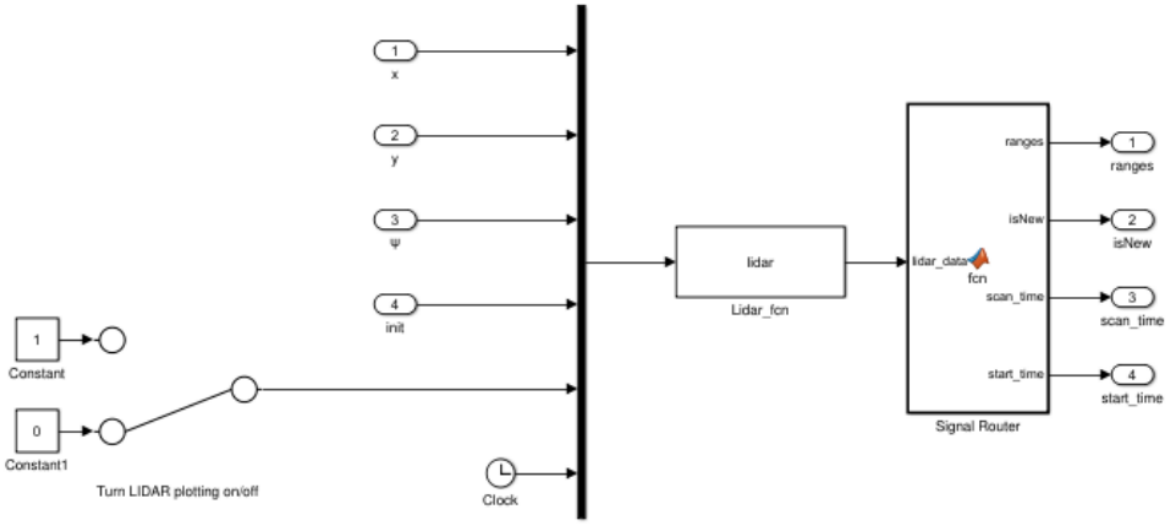


Рисунок 4.12. Схема підсистеми LIDAR у Simulink

На початку процесу моделювання було помічено, що LIDAR не працював належним чином, як очікувалося. При повному обертанні LIDAR не зміг виконати вимірювання на 360 градусів. Було досліджено, що кількість пропущених вимірювань зазвичай зумовлена тим, як промінь лазера потрапляє на об’єкт. Коли LIDAR знаходиться в обмеженому просторі, ми контролюємо два параметри, які будуть відрізнятися при кожному вимірі, відстань до стіни та кут до стіни. Таким чином, було вирішено дослідити, як кожна з цих умов впливала на ймовірність пропущених вимірювань LIDAR.

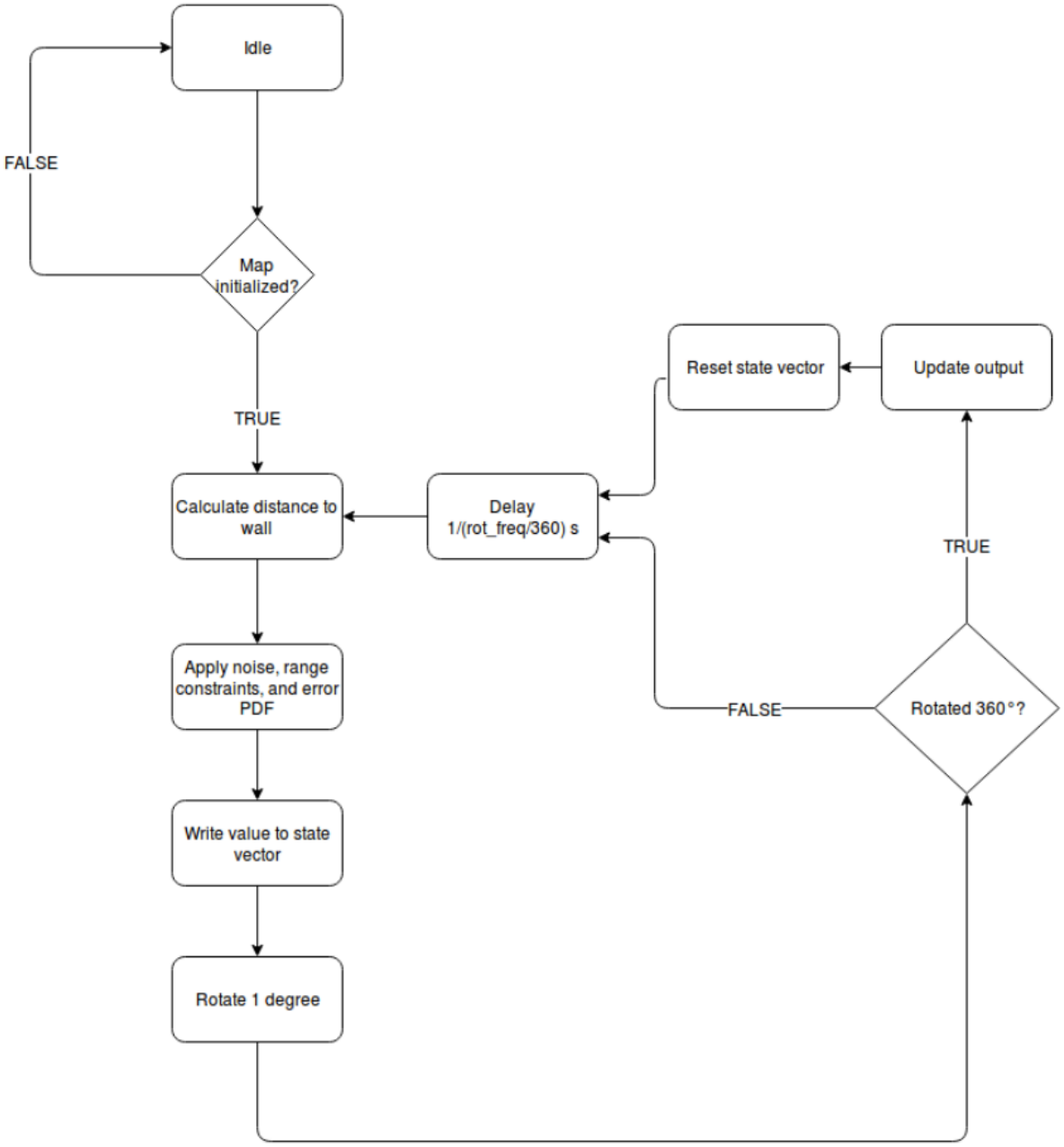


Рисунок 4.13: Блок-схема LIDAR S-функціїу Simulink

Результати експерименту представлені у графіках в Simulinkна рис. 4.14 та рис. 4.15 відповідно. Варто відмітити, що не проводився експеримент щодо впливу різних типів поверхонь на вимірювання.

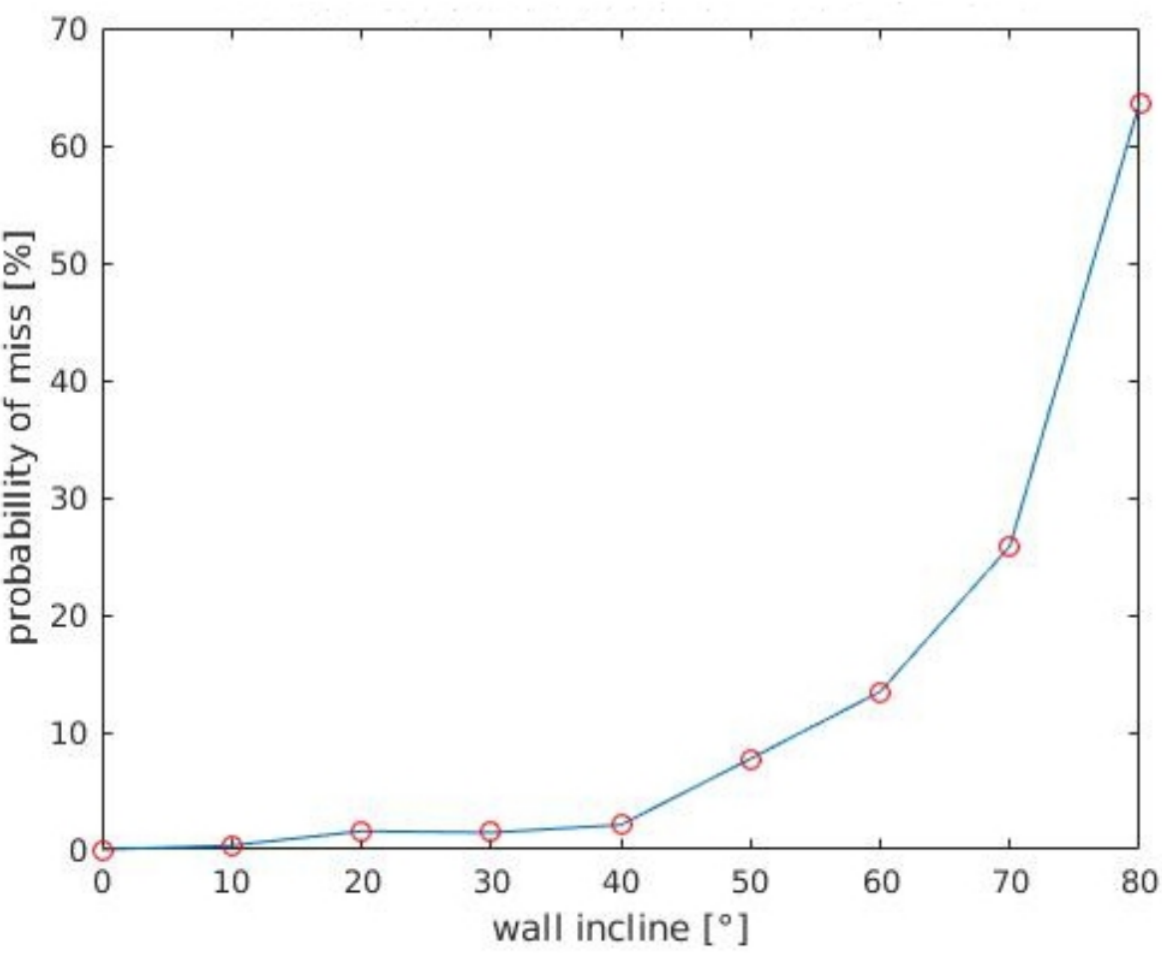
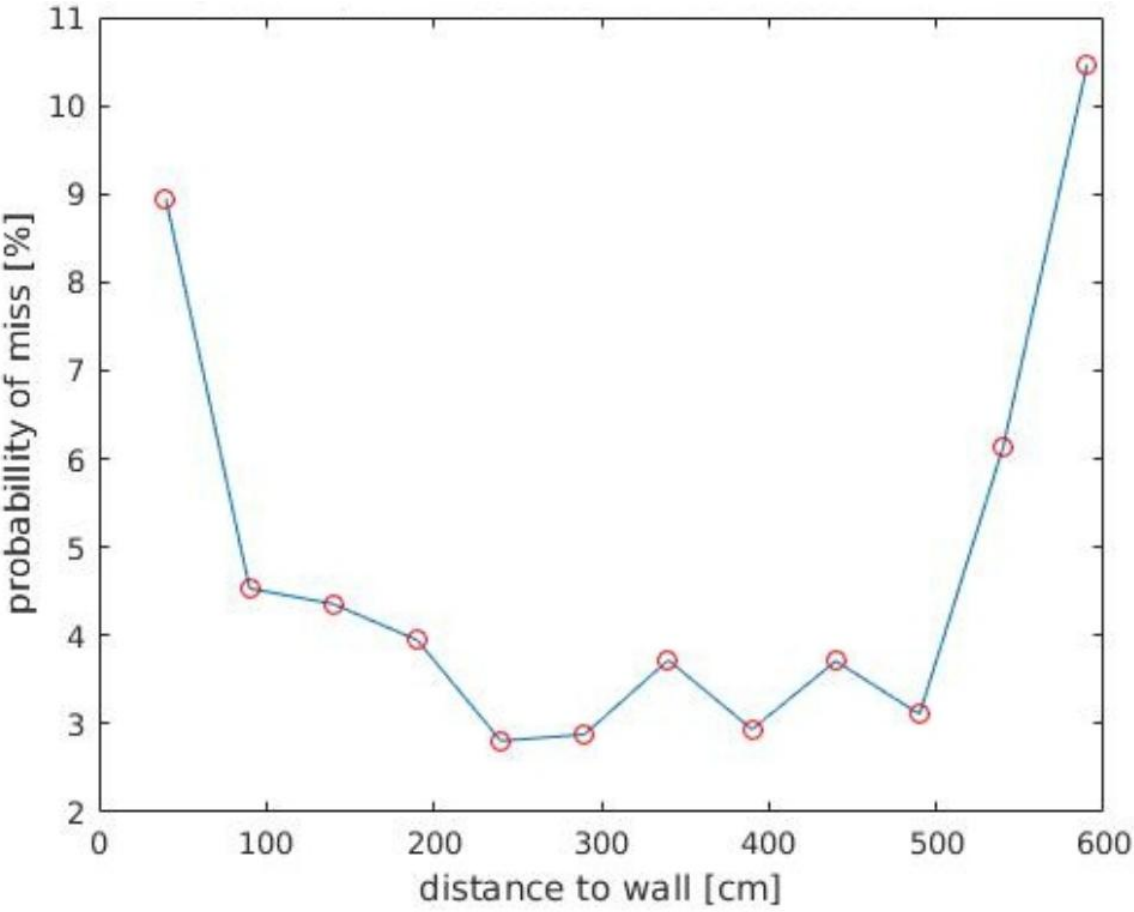


Рисунок 4.14: Відсоток промахів через нахил стіни



Малюнок 4.15: Відсоток промахів через відстань до стіни.

З рис. 4.14 видно, що коли кут відбиття перевищує 40°, відсоток пропущених вимірювань різко зростає. Гіпотетично це через кут нахилу вимірюваного об'єкта, що спричиняє більший кут відбиття лазера. У моделі ці помилки реалізуються як дві таблиці. Коли моделюється вимірювання, кут і відстань до вимірюваного об'єкта передаються до відповідної таблиці, і отримується ймовірність помилки. Потім ця ймовірність використовується у біноміальній випадковій величині, яка множиться на виміряну відстань, що означає, що вимірювання буде дорівнює 0, якщо випадкова величина поза діапазоном.

5. СПЕЦІАЛЬНА ЧАСТИНА

5.1. Вибір двигунів і енкодерів

Зворотній зв'язок відіграє важливу роль у контролі руху. Серводвигуни та крокові двигуни використовують зворотний зв'язок енкодера для точного контролю швидкості та положення. Енкодер - це пристрій, який перетворює рух об'єкта, наприклад, валу двигуна, в аналоговий або цифровий вихід, що відповідає швидкості або положенню. Енкодери є ефективними пристроями зворотного зв'язку лише тоді, коли вони належним чином підібрані для застосування. Вибір потрібного енкодера базується на потребах програмного забезпечення. Ключові фактори, які слід врахувати, включають:

- умови навколишнього середовища, включаючи температури, вологу, удари та вібрацію, забруднення;
- тип руху: односпрямований або двонаправлений тощо;
- величина руху та чутливість до перевезення;
- механічна конструкція, включаючи відповідність системі;
- електричні вимоги до приводів та контролерів;
- фізична конфігурація, включаючи форм-фактор, фізичну відстань між кодером та контролерами;

Характеристики навколишнього середовища обумовлюють вибір типу енкодера. Найпоширенішими давачами є оптичний, магнітний та індуктивний. Але так як у роботі використовуємо магнітні енкодери то розглядатимемо лише їх.

У магнітнихенкодерахвикористовується зубчаста шестерню із чорних металів, барабани або диски із змінними магнітними доменами для збурення магнітного поля. Магнітні енкодери з високою точністю реєструють

проходження магнітних полюсів магнітного елемента, що обертається на малій відстані від чутливого елемента, і перетворюють цю інформацію у відповідний цифровий код.

Магнітні енкодери витримують надзвичайно суворі умови зовнішнього середовища, завдяки чому вони добре підходять для промислового застосування. Вони можуть працювати під водою, покриті пилом і піддаватися сильним вібраціям. Вони досить економічні, що робить їх придатними для бюджетних пристроїв.

Проте, з іншої сторони, магнітний енкодер чутливий до сильних магнітних полів і може вимагати екранування. Дуже високі ударні навантаження можуть розмагнічувати області магніту, як і дуже високі температури; як вже згадувалося, датчики ефекту Холла менш вразливі до ударних навантажень.

Виходячи з міркувань вибору енкодера, енергоспоживань, споживчих розмірів та крутного моменту вибираємо сервоприводHS-M7990TH.

Технологічно сервоприводHitec, "MonsterTorque" HS-M7990TH, оснащений ультраточним магнітним енкодером із високою роздільною здатністю замість звичайного механічного потенціометра. Пристрій HS-M7990TH. Інші особливості HS-M7990TH включають оптимізований без сердечника двигун на 7,4 В, вбудований корпус радіатора та верхній корпус із двома загартованими сталевими штифтами редукторів, підтримуваних латунними осьовими втулками.

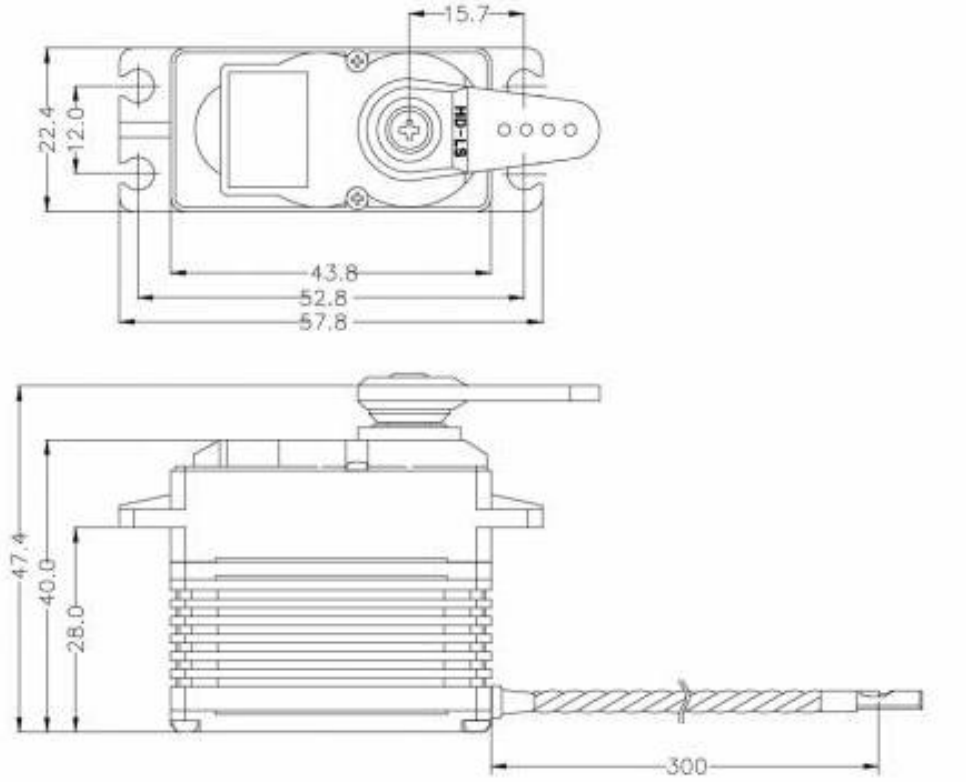


Рисунок 5.1. Фізичні характеристики сервоприводу HS-M7990TH

5.2. Вибір LiDAR

Вибір правильного лазерного сканера (Lidar) забезпечує точну навігацію та дозволяє продовжити роботу навігаційної системи робота навіть у несприятливих умовах. Нижче наведено 5 важливих характеристик щодо вибору лазерного сканера при розробці робота.

1. Стійкість до навколишнього світла.

Однією з ключових проблем, що стоять перед роботами що переміщуються у зовнішньому середовищі, є навколишнє світло. Лідар використовує технологію вимірювання часу відбиття світла, що означає, що

сканер випромінює імпульс світла, який потім відбивається навколишнім об’єктом, якщо він присутній. Час, необхідний для проходження імпульсу між відбивачем і назад, пропорційний відстані. Це дозволяє роботу виявляти та уникати перешкод на своєму шляху.

Інтенсивне сонячне світло може перервати цей процес, недозволяючи приймачу зчитати власні світлові імпульси, що повертаються. Це може призвести до несправності лазерного сканера, що в свою чергу призведе до втрати навігації.

2. Стійкість до навколишнього шуму.

У додатках фактори навколишнього середовища, такі як опади, можуть перешкоджати здатності лазерного сканера точно визначати перешкоди (тобто опади змушують робота виявляти перешкоди, яких немає). Тому важливо вибрати лазерний сканер, який здатний підтримувати високий рівень точності навіть у несприятливих умовах - таких як дощ, сніг, пил тощо. Лазерні сканерикомпанії SICK пом'якшують можливі перешкоди за допомогою технології мульти-ехо, забезпечуючи надійність незалежно від погодних умов та мінімізують помилкові тривоги.

Технологія мульти-ехо важлива, оскільки частина енергії від імпульсу лазера може відбиватись поблизу об'єктами, такими як дощ, тоді як решта променя продовжує поширюватися і відображається фактичною перешкодою. Lidar за допомогою технології мульти-ехо оцінює ці множинні відбивання та ігнорує тісніші і слабкіші відбиття, спричинені факторами зовнішнього середовища. Це усуває шум і допомагає запобігти проблемам навігації Lidar на відкритому повітрі.

3. Стійкість до навколишнього середовища.

Також варто зазначити важливість високого стійкості до несприятливих умов навколишнього середовища при виборі Lidar для

транспортних засобів.Вони піддаються впливу стихії, і лазерний сканер, який не є досить надійний, може зазнати збоїв у роботі або вийти з ладу, якщо на нього потрапитьволога. Якщо лазерний сканер містить якісний корпус то це дозволить скоротити середній час між відмовами (MTBF) та забезпечить довговічність навігаційної системи.

4. Температурний діапазон.

Іншим фактором, що впливає на роботу лазерного сканера, є діапазон температур. Надзвичайно низька або висока температура може пошкодити давач (наприклад, призвести до тріщин у корпусі). Через це варто вибирати лазерні сканери, які мають широкий діапазон робочих температур та вбудовану систему контролю температури.

5. Електромагнітні (EMI) міркування.

Останнє варто врахувати, так це те що транспортний засіб може опинитися в самих різних середовищах, що можуть містити різну силу та типи електричного шуму. Цей шум може індукувати давачі та схеми системи управління, що може спричинити хаотичну поведінку робота.

На основі вищесказаного і вимог до транспортного засобу підібрано LiDAR який найкраще підходить для прототипу. LiDAR виготовлений компанією SLAMTEC і належить до моделей RPLIDAR.

RPLIDAR S1 - це нове покоління 360-градусного двовимірного лазерного сканера (LIDAR), розроблене компанією SLAMTEC. Може приймати до 9200 вимірів лазера в секунд. Оснащений запатентованою технологією OPTMAG SLAMTEC, що збільшує час автономної роботи системи LIDAR.

Система може виконувати 2D 360-градусне сканування в радіусі 40 метрів. Сформовані дані 2D-хмарних точок можна використовувати для картографування, локалізації та моделювання об’єктів / середовищ.

Порівняно з іншими серіями, RPLIDAR S1 має більш кращі показники при виявленні об'єктів на великій відстані, білих і чорних об'єктів або об'єктів під прямими сонячними променями, що ідеально підходить для побудови карт у зовнішньому середовищі в радіусі 40 метрів. Тому його можна широко застосовувати в багатьох бізнес задачах, орієнтованих на користувача.

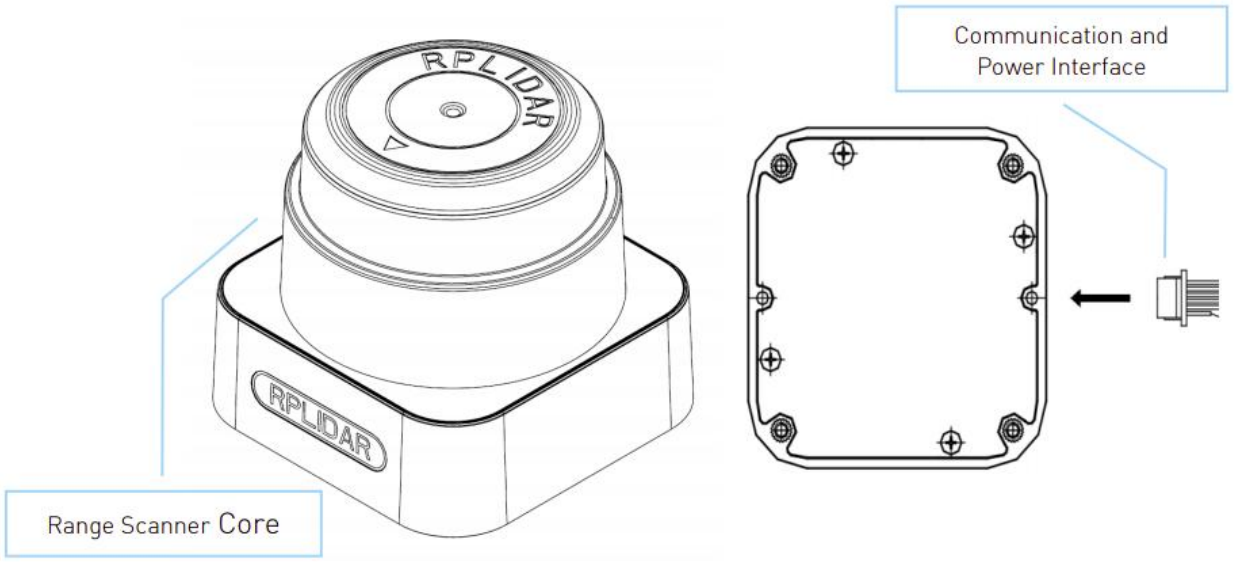


Рисунок 5.2. Система RPLIDAR S1

Таблиця 5.1

Основні характеристики RPLIDAR S1

Характеристика	Опис
Дистанція	Білі об’єкти: 40 м, Чорні об’єкти: 10 м
Сліпа зона	0,1 см
Частота вимірювання	9,2 кГц
Частота сканування	Значення за замовчуванням: 10 кГц (можна налаштовувати в діапазоні 8 кГц – 15 кГц)
Кутова роздільна	Значення за замовчуванням: 0,391°

здатність	(в залежності від частоти сканування 0,313°-0,587°)
Інтерфейс	TTL UART
Швидкість передачі даних по інтерфейсу	256000 байтів за секунду
Точність	±5 см
Роздільна здатність	3 см

Типова частота сканування RPLIDAR S1 становить 10 Гц (600 об / хв), і частоту можна регулювати в межах 8-15 Гц відповідно до конкретних вимог. При частоті сканування 10 Гц частота дискретизації становить 9,2 кГц, а кутова роздільна здатність 0,391 °.

Таблиця 5.2.

Характеристики лазера RPLIDAR S1

Характеристика	Одиниця вимірювання	Мінімальне значення	Типове значення	Максимальне значення	Коментар
Довжина лазерного променю	Нанометр (нм)	895	905	915	Інфрачервоні промені
Потужність	Ват (Вт)	-	28	-	-
Час імпульсу	Наносекунди (нс)	-	10	-	-
Клас безпеки лазера	-	-	IEC-60825 Class 1	-	-

5.3. RaspberryPi і операційна система

Одним із найпопулярніших одноплатних комп’ютерів на ринку є RaspberryPi. Одна з його переваг що пристрій має найкращий баланс між ціною та потужністю. Що більш важливо, що одноплатний комп’ютер використовує лише до 2А з напругою живлення 5V. А також підходить під усі вимоги проектованої системи.

Міні-комп'ютер складається з обчислювальний модуля RaspberryPi 3+ (CM3 +) який міститьоперативну пам’ять DDR2-SODIMMна модулях (SoM), містить процесорBCM2837B0, пам’ять eMMCFlash та схему живлення. Ці модулі дозволяють використовувати апаратно-програмний стек RaspberryPi у прототипі автономного транспортного засобу.

Для коректної роботи модулів CM3 + потрібне зображення програмного забезпечення / мікропрограми від листопада 2018 року або новішої версії.

Список периферії:

- 48x GPIO
- 2x I2C
- 2x SPI
- 2x UART
- 2x SD / SDIO
- 1x HDMI 1.3a
- 1x USB2 HOST / OTG
- 1x DPI (паралельний RGB-дисплей)
- 1x інтерфейс NAND (SMI)
- 1x 4-смуговий інтерфейс камери CSI (до 1 Гбіт / с на смугу)
- 1x 2-смуговий інтерфейс камери CSI (до 1 Гбіт / с на смугу)

- 1x 4-смуговий інтерфейс дисплея DSI (до 1 Гбіт / с на смугу)
- 1x 2-смуговий інтерфейс дисплея DSI (до 1 Гбіт / с на смугу)

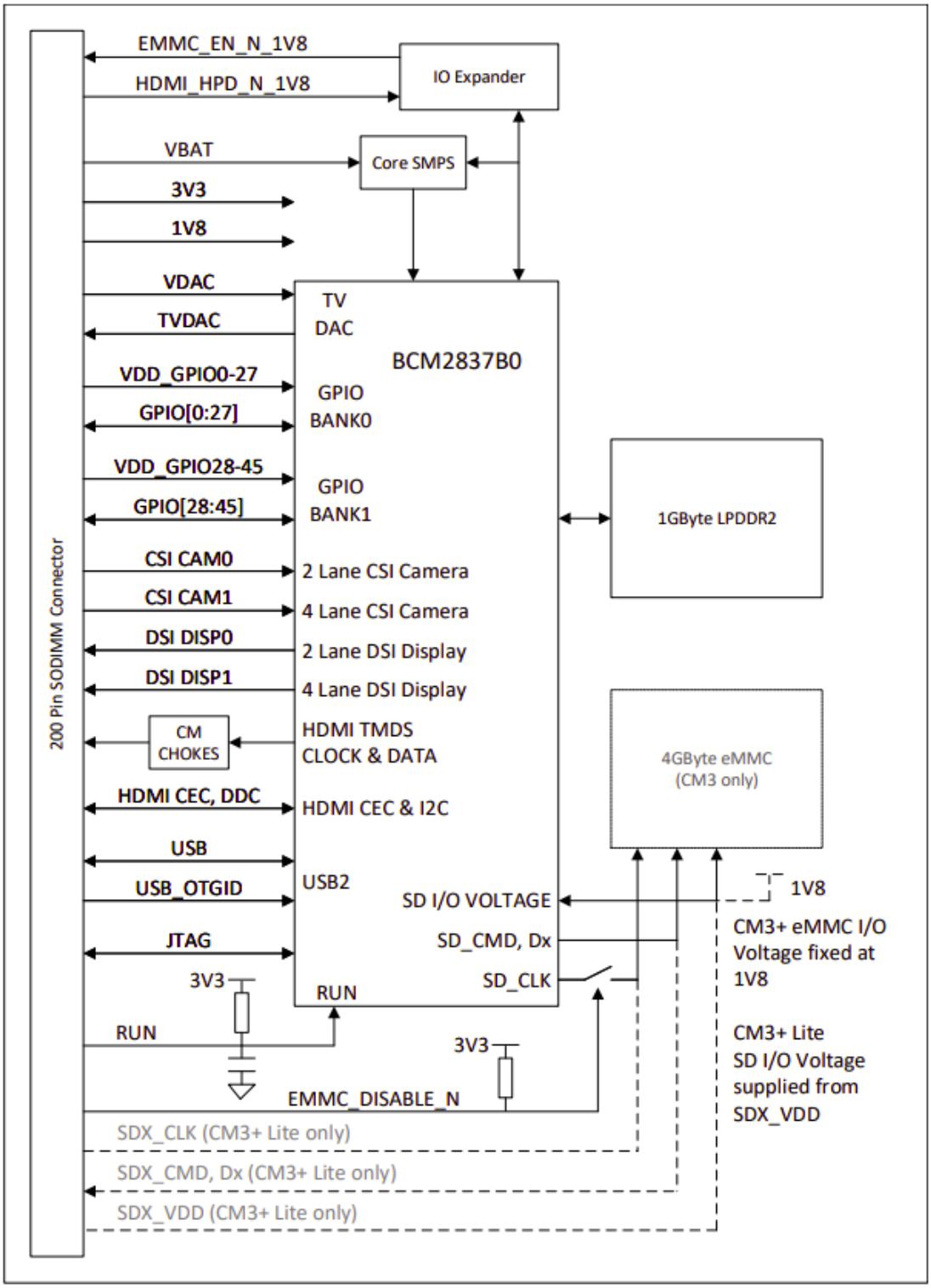


Рисунок 5.3. CM3+ блок діаграма

Опис програмного забезпечення:

- Набір інструкцій ARMv8
- Зрілий і стабільний стек програмного забезпечення Linux
- Остання підтримка ядра Linux
- Повна доступність функцій GPU із використанням стандартних API

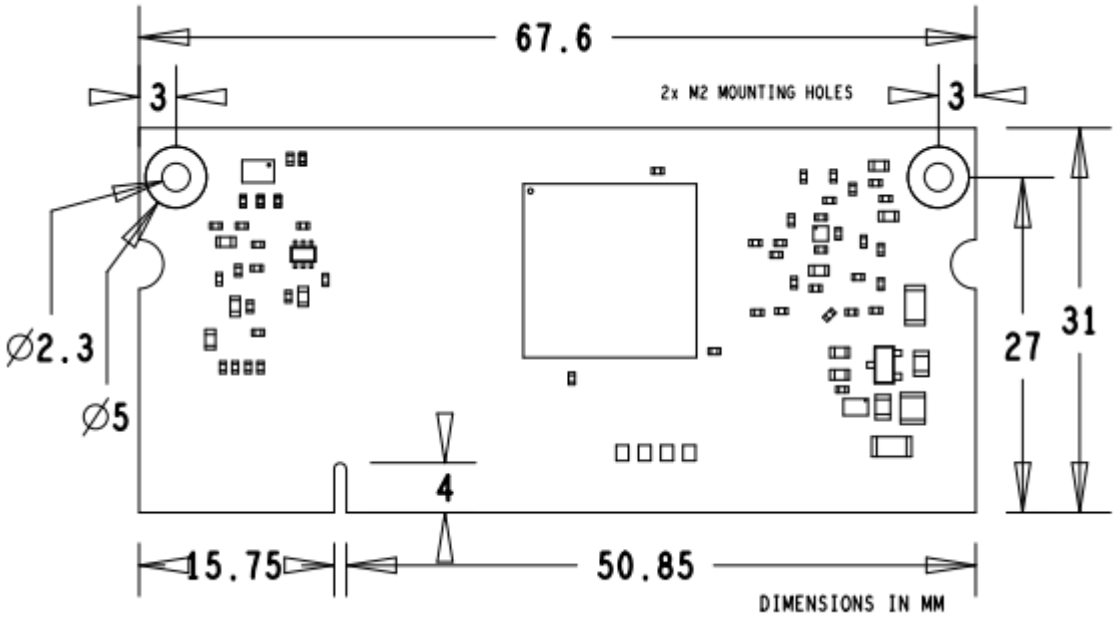


Рисунок 5.4. Механічні розміри RaspberryPi

Модулі CM3 + відповідають механічним специфікаціям JEDEC MO-224, тому можуть працювати з багатьма модулями оперативної пам’яті DDR2 SODIMM, доступними на ринку.

Таблиця 5.3

Максимальні/мінімальні значення напруги для GPIO контактів

Символ	Параметр	Мінімум	Максимум	Одиниця вимірювання
VBAT	Живлення SMPS	-0.5	6.0	В
3V3	Живлення лінії 3V3	-0.5	4.1	В
1V8	Живлення 1V8	-0.5	2.1	В
VDAC	Живлення TV DAC	-0.5	4.1	В
GPIO0-27_VDD	Живлення GPIO0-27 I/O	-0.5	4.1	В
GPIO28-45_VDD	Живлення GPIO28-45 I/O	-0.5	4.1	В
SDX_VDD	Живлення SD/eMCC	-0.5	4.1	В

5.4. Операційна система робота

Операційна система робота (ROS) - це основа для написання програмного забезпечення для робота. Вона містить набір інструментів, бібліотек та конвенцій, спрямованих на спрощення завдання створення складного та надійного алгоритму поведінки робота на різноманітних апаратних платформах що використовуються в робототехніці.

При створенні прототипу транспортного засобу використовували ядро ROS та пакети з бібліотеками в яких реалізовано SLAM.Ядро ROS складається з інтерфейсу передачі повідомлень. Який забезпечує зв’язок між

пристроями і процесами у розробленій системі.Використовуються наступні можливості ядра:

1. Опублікувати / підписатися на передачу анонімного повідомлення. Однією з перевагою використання системи передачі повідомлень є те, що вона дозволяє реалізовувати чіткі інтерфейси між вузлами у системі, тим самим покращуючи інкапсуляцію та сприяючи повторному використанню коду.
2. Запис та відтворення повідомлень. Оскільки система публікації / підписки є анонімною та асинхронною, дані можна збирати та відтворювати без будь-яких змін у коді. Наприклад, є завдання A, яке зчитує дані з датчика, і в той час виконується завдання B, яке обробляє дані, отримані завданням A. ROS зберігає отримані дані, опублікованізавданням A у файл, а потім повторно публікує ці дані з файлу пізніше щоб завдання B обробило дані. Це архітектурне рішеннязменшує зусилля щодо розробки і сприяє гнучкості та модульності системи.
3. Виклик віддаленої процедури запит / відповідь
4. Розподілена систем параметрів. Забезпечує можливістю обмінюватися інформацією про конфігурацію через сховище ключ-значення. Ця система дозволяє легко змінювати налаштування завдань і навіть дозволяє завданням змінювати конфігурацію інших завдань.

Для управління транспортним засобом та отримання данихдавачів був реалізована система яка складається зROS-вузлів, що зображена на рис. 5.5. В системі використовуються такі вузли:

1. CAN зчитувач

Вузол can_reader отримує дані давача, надіслані STM32 через шину CAN, і публікуєотримані дані для інших вузлів ROS. Залежно від

ідентифікатора повідомлення, викликається функція, яка обробляє дані LIDAR, або дані одометрії. Дані LIDAR публікуються на тему, що ку публікується дані давачів називається /sensor_data. На тему публікуназивається /scan, яка має тип повідомлення sensor_msgs/Laserscan. Тема на яються наступні дані: швидкість автомобіля, значення гіроскопа та індекс часу для вимірювання.

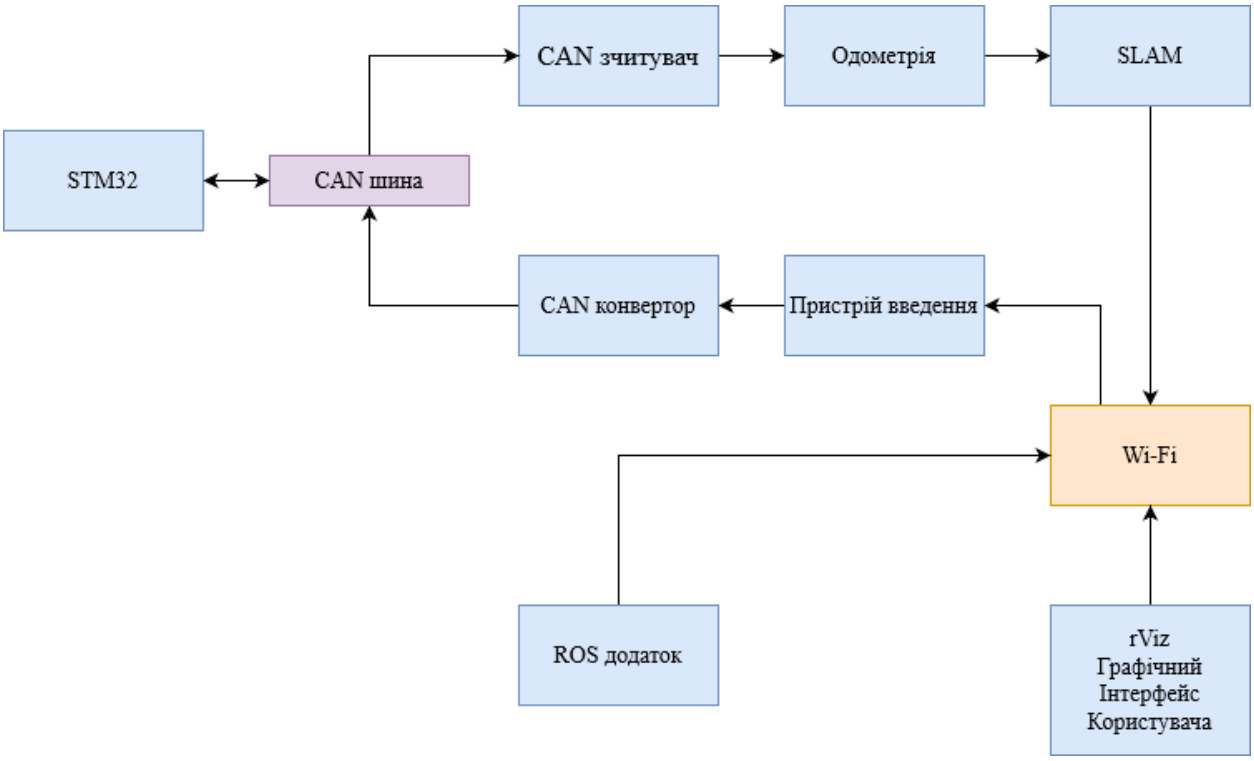


Рисунок 5.5. Блок-схема активних вузлів ROS під час виконання алгоритму SLAM на фізичному транспортному засобі.

2. Одометрія

Завдання вузла одометрії - виміряти кутове розміщення та положення прототипу транспортного засобу. Дані зчитуються у форматі: кутова швидкістьвідповідний індекс часу. За допомогою цих даних можна обчислити швидкість в напрямку x та y. Використовуючи різницю

переміщення між поточним часом та попереднім моментом часу можна обчислити положення x у транспортного засобу та кут, на який транспортний засіб повернув. Обчислене переміщенняпублікуються на тему /position.

3. SLAM

Вузол SLAM містить реалізацію алгоритму SLAM. Для роботи вузол використовує повідомлення опубліковані давачами і вузлом одометріяОтримавши ці дані, він може створювати карту, яка містить інформацію про навколишнє середовище.

6. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Питання охорони праці та безпеки в надзвичайних ситуаціях розглянуті для етапу проектування й розробки системи для побудови карти з одночасним контролем наявного місцерозташування і пройденого шляху в мобільних автономних засобах.

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності. Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам законодавства. Працівник має право відмовитися від дорученої роботи, якщо створилася виробнича ситуація, небезпечна для його життя чи здоров'я або для людей, які його оточують, або для виробничого середовища чи довкілля. Він зобов'язаний негайно повідомити про це безпосереднього керівника або роботодавця. Факт наявності такої ситуації за необхідності підтверджується спеціалістами з охорони праці підприємства за участю представника профспілки, членом якої він є, або уповноваженої працівниками особи з питань охорони праці (якщо професійна спілка на підприємстві не створювалася), а також страхового експерта з охорони праці [12]. Завдання охорони праці – звести до мінімуму ушкодження та захворювання працівника з одночасним забезпеченням комфорту при максимальній продуктивності праці. Основними цілями охорони праці є формування в спеціалістів

необхідних знань і практичних навичок по правових і організаційних питаннях охорони праці, виробничій санітарії, техніці безпеки, пожежній безпеці.

6.1. Загальна характеристика приміщення і робочого місця

Розробка системи для побудови карти з одночасним контролем наявного місцерозташування і пройденого шляху в мобільних автономних засобах виконується в приміщенні, яке знаходиться на четвертому поверсі семиповерхового будинку з загальним та місцевим освітленням. В приміщенні одностороннє освітлення, вікна орієнтовані на схід, на вікнах є ролети. Стеля білого кольору з коефіцієнтом відбиття 0,7, стіни цегляні світлого кольору з коефіцієнтом відбиття 0,5. В приміщенні працює 4 людини, відповідно до цього отримуємо вхідні дані для аналізу потенційно-небезпечних і шкідливих виробничих факторів, які наведено в табл. 7.1.

Таблиця 6.1

Вхідні дані

Параметри приміщення	**Значення**
Довжина х ширина х висота	6,6 х 6,1 х 2,7 м
Номер робочого місця	**Специфіка роботи**
I робоче місце	Front-end програміст (спеціаліст з розробки клієнтської частини веб- застосунків)
II робоче місце	Back-end програміст (спеціаліст з розробки серверної частини веб застосунків та проектування баз даних)
III робоче місце	Бізнес-аналітик (також виконує роль менеджера продукту)
IV робоче місце	UI-UX веб-дизайнер

Технічні засоби (кількість)	**Назва та характеристики**
Монітор (4 шт.)	HP 22Xi/21,5"/1920x1080px/IPS
Комп'ютер (4 шт.)	HP ProBook 440 G6, екран 14" IPS (1920x1080) Full HD, IntelCore i7-8565U (1.8 - 4.6 ГГц)/RAM 16 ГБ/SSD 256 ГБ
Підлоговий кулер (1 шт.)	CRYSTAL YLR3-5V208
Кондиціонер (1 шт.)	DEKKER DSH105R/G/26м^2/2,65кВт-2,9кВт/25x74,5x19,5см/9 кг
Світильники загального призначення (3 шт.)	Світильник растровий вмонтований 4x18W
Світильники місцевого призначення (4 шт.)	DeLuxDécor TF-05 / 1 x 40Вт

Згідно НПАОП 0.00-7.15-18 [14] площа S', виділена для одного робочого місця з персональною ЕОМ, повинна бути не менше 6 м^2 і об'єм – не менше 20 м^3. У приміщенні розташовано 4 робочі місця, що повністю відповідає необхідним нормам.

Розрахуємо фактичні значення цих показників, поділивши об'єм приміщення та загальну площу на кількість працюючих.

Отже, виходячи з отриманих результатів за характеристиками площі та об'єму, приміщення відповідає наступним нормам.

Таблиця 6.2

Характеристики робочого місця

№	Найменування параметру	Значення	
		фактичне	нормативне

1.	Висота робочої поверхні, мм	780	680 – 800
2.	Ширина робочої поверхні, мм	1500	не менше 600
3.	Глибина робочої поверхні, мм	750	не менше 600
4.	Висота простору для ніг, мм	750	не менше 600
5.	Ширина простору для ніг, мм	800	не менше 500
6.	Глибина простору для ніг, мм	750	не менше 450
7.	Висота поверхні сидіння, мм	480	400 – 500
8.	Ширина сидіння, мм	500	не менше 400
9.	Глибина сидіння, мм	500	не менше 400
10.	Висота опорної поверхні спинки, мм	550	не менше 300
11.	Ширина поверхні спинки, мм	470	не менше 380
12.	Довжина підлокітників, мм	300	не менше 250
13.	Ширина підлокітників, мм	60	50 – 70
14.	Відстань від очей до екрану, мм	650	600 – 700

Можна зробити висновок, що розміри робочого місця працівника відповідають встановленим нормам, виходячи з заданих параметрів.

6.2. Аналіз потенційно небезпечних і шкідливих виробничих факторів на робочому місці

При створенні системи для побудови карти з одночасним контролем наявного місцерозташування і пройденого шляху в мобільних автономних засобах виконується сидячи без фізичних зусиль, тому відноситься до категорії легка Іа.

Приміщення для роботи мають бути обладнані системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією відповідно до ДБН В.2.5-67:2013. Нормовані параметри мікроклімату, іонного складу повітря, вмісту шкідливих речовин відповідають вимогам ДСН 3.3.6.042-99, ГН 2152-80, ГОСТ 12.1.005-88, ДСТУ ГОСТ 12.0.230:2008 та ДСТУ ГОСТ 12.4.041:2006. Під вентиляцією розуміють сукупність заходів та засобів, призначених для забезпечення на постійних місцях та зонах обслуговування приміщень метеорологічних умов та чистоти повітряного середовища, що відповідають гігієнічним та технічним вимогам. Основне завдання вентиляції – вилучити із приміщення забруднене, вологе або нагріте повітря та подати чисте свіже повітря.

Джерелами шуму в приміщенні є вентилятор системного блоку персонального комп'ютера, вентилятор ноутбука та кондиціонер. Звук, що створюється вентилятором та кондиціонером, можна класифікувати як постійний.

Відповідно до ДБН В.2.5-28:2018 робота відноситься до розряду зорових робіт. Передбачається використання природного, штучного та змішаного освітлення.

ЕОМ є однофазним споживачем електроенергії, що живиться від змінного струму 220В від мережі із заземленою нейтраллю. IBM PC

відноситься до електроустановок до 1000В закритого виконання, всі струмопровідні частини знаходяться в кожухах. За способом захисту людини від ураження електричним струмом, ЕОМ і периферійна техніка повинні відповідати 1 класу захисту.

Технічні методи захисту від ураження струмом зводиться до застосування струму безпечної напруги, захисту у випадку випадкового доторкання до струмоведучих частин і від надмірних струмів, захисту у випадку переходу напруги на неструмоведучі металеві частини установки.

Безпечну напругу одержують від сітки підвищеної напруги (110-120 В) за допомогою знижувальних трансформаторів.

Захисту від доторкання до струмоведучих частин установки досягають за допомогою ізоляції, відгородження застосування блокуючих пристроїв запобіжної сигналізації та неприступності розташування установок.

Розподільні щитки поміщають у закриті металеві кожухи-ящики.

Запобіжну сигналізацію застосовують у вигляді плакатів і надписів. Найкращими світловими сигналізаціями є подвійні, яких при наявності напруги горить червона лампочка, а при її відсутності - зелена.

Захист від надмірних струмів – короткого замикання і струмів перевантаження, які можуть спричинити займання ізоляції, здійснюється запобіжниками й автоматичними вимикачами, а захист від переходу напруги на струмоведучі частини за допомогою захисного заземлення і захисного вимикання.

Запобігання пожежі досягається виключенням утворення джерел загорянь і горючого середовища.

В цьому приміщенні можливі пожежі таких класів: А – горіння твердих речовин, Е – горіння електроустановок під напругою.

6.3. Безпека в надзвичайних ситуаціях

Великі аварії і катастрофи можуть призвести до загибелі людей і завдати відчутної шкоди. Тому забезпечення безаварійної роботи на підприємстві потрібно розглядати як важливе завдання, що вимагає уваги керівників усіх рівнів та інженерно-технічного персоналу. Аварії можуть відбутися в результаті стихійних лих, допущених прорахунків у проектуванні, будівництві й устаткуванні підприємств; введення в експлуатацію об'єктів з великими недоробками і відступами від проектів; прийняття в експлуатацію вентиляційних систем без випробування їх на ефективність роботи; недоробок з техніки безпеки й охорони праці; незадовільного оснащення контрольно-вимірювальною, захисною, блокуючою апаратурою і недостатньої герметичності технологічного устаткування. Вони можуть бути також наслідком технологічних процесів, несправності електропроводки та відсутності надійних систем пожежогасіння.

Кожна конкретна аварія викликається сукупністю ряду причин і несприятливих факторів у результаті низького рівня обізнаності персоналу, допущеної недбалості, порушені правил техніки безпеки. Вивчення причин аварій і всебічна оцінка ступеня небезпеки дозволяють правильно визначити заходи щодо їх попередження, передбачити необхідні заходи захисту людей і зниження збитків.

Основними заходами щодо ліквідації наслідків великих аварій є: оповіщення про небезпеку робітників та службовців; комплексна розвідка об'єкта, на якому відбулася аварія; порятунок людей з-під завалів, зі зруйнованих і пошкоджених будинків та споруд; надання медичної допомоги

постраждалим і евакуація їх у лікувальні установи; гасіння пожеж; локалізація аварій на комунально-енергетичних мережах, які перешкоджають веденню рятувальних робіт; улаштування проїздів і проходів до місць аварії; обвалування нестійких конструкцій, розбирання завалів, демонтаж збереженого устаткування, якому загрожує небезпека.

Швидке проведення рятувальних робіт і оперативна ліквідація наслідків аварії вимагають значних сил і засобів, для цих цілей залучаються спеціальні та територіальні формування загального призначення.

Рятувальні роботи в місцях аварії проводяться в умовах загазованості, а при пожежах – задимленості і високих температур, щоб забезпечити безперервність роботи з наростаючим темпом, ресурси поділяють на зміни і виділяють резерви.

Рятувальні роботи та допомога потерпілим організовуються негайно після виникнення аварії. До місця аварії першими повинні прибувати протипожежні команди, підрозділи міліції, машини швидкої медичної допомоги.

Ліквідація наслідків аварії може здійснюватися одночасно на всьому об’єкті чи на окремих ділянках у тих випадках, коли мається достатня кількість сил і засобів, роботи проводяться відразу на всій площі. Якщо сил недостатньо, роботи повинні проводитися послідовно, в першу чергу їх починають там, де необхідно надати допомогу людям, і на ділянках, які становлять найбільшу небезпеку.

Перша медична і лікарська допомога надається постраждалим, які знаходяться в стані шоку, а також звільненим з-під завалів і уламків. Витягування людей з-під великих завалів здійснюється з дотриманням заходів безпеки, їм надається невідкладна медична допомога з наступною евакуацію в лікувальні установи.

Для організації робіт з ліквідації наслідків аварій і катастроф на об’єкті створюється постійно діюча надзвичайна оперативна група під керівництвом головного інженера. У надзвичайних умовах вона працює під загальною координацією районної (міської) надзвичайної комісії.

Ліквідація наслідків аварії проводиться в чотири етапи:

1. Вживання екстрених заходів (попередня оцінка обстановки, надання допомоги потерпілим, вживання екстрених заходів по захисту робітників, службовців, населення, локалізація аварії та організація розвідки).

2. Оперативне планування (розвідка, уточнення обстановки, розрахунок необхідних сил і засобів, оцінка масштабів збитків, планування робіт з ліквідації наслідків аварії).

3. Рятувальні роботи (розшук потерпілих, їх витягування з-під завалів, з палаючих будинків, евакуація людей із зони аварії, надання першої медичної й інших видів допомоги постраждалим).

4. Ліквідація наслідків (заходи щодо створення умов для забезпечення життєдіяльності населення в районі аварії, відновлення функціонування.

ВИСНОВКИ

У результаті виконання магістерської роботи було розроблено систему для побудови карти з одночасним контролем наявного місцерозташування і пройденого шляху в мобільних автономних засобах.

Проаналізовано методи для вирішення даної проблеми.Проведені теоретичні дослідження по вибору оптимального SLAM алгоритму. Досліджені базові математичні моделі що використовуються для побудови автоматичних систем локалізації і побудови карти середовища.

Створено симуляції в Simulink які відображають модель системи максимально близько до реальних умов. Показано результати фізичних та змодельованих експериментів кожного компонента проектованої системи.

Вибрано оптимальну архітектуру для забезпечення взаємодії програмних компонентів у системі. Розроблено програмне забезпечення проектованої системи.

Розроблено прототип з використанням енкодерів та LiDAR, на якому випробувано розроблене програмне забезпечення.

Результати даної роботи можна використовувати у промислових автономних транспортних засобах. Що дозволить автоматизувати процеси виробництва які пов’язані з транспортуванням.

БІБЛІОГРАФІЯ

1. Durrant-Whyte, Leonard. Mobilerobotlocalizationbytrackinggeometricbeacons.
2. Smith, etal. Estimatinguncertainspatialrelationshipsinrobotics.
3. H. Durrant-Whyteand T. Bailey. Simultaneouslocalizationandmapping: part i. RoboticsAutomationMagazine, IEEE, 13(2):99–110, June 2006.
4. J. J. Leonardand H. F. Durrant-Whyte. Simultaneousmapbuildingandlocalizationforanautonomousmobilerobot. InIntelligentRobotsandSystems ’91. ’IntelligenceforMechanicalSystems, Proceedings IROS ’91. IEEE/RSJ InternationalWorkshopon, pages 1442–1447 vol.3, Nov 1991.
5. J. S. Gutmannand K. Konolige. Incrementalmappingoflargecyclicenvironments. InComputationalIntelligenceinRoboticsandAutomation, 1999. CIRA ’99. Proceedings. 1999 IEEE InternationalSymposiumon, pages 318–325, 1999.
6. S. Thrun, W. Burgard, and D. Fox. A probabilisticapproachtoconcurrentmappingandlocalizationformobilerobots. AutonomousRobots, 5(3-4):253– 271, 1998.
7. M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solutiontothesimultaneouslocalizationandmapbuilding (slam) problem. IEEE TransactionsonRoboticsandAutomation, 17(3):229– 241, Jun 2001.
8. S. Thrun, W. Burgard, and D. Fox. Probabilisticrobotics. MIT press, 2005.
9. D. Scaramuzzaand F. Fraundorfer. Visualodometry [tutorial]. Robotics&AutomationMagazine, IEEE, 18(4):80–92, 2011.
10. H. P. Moravec. Obstacleavoidanceandnavigationintherealworldby a seeingrobotrover. Technicalreport, DTIC Document, 1980.

11. S. Lacroix, A. Mallet, R. Chatila, and L. Gallo. Roverselflocalizationinplanetary-likeenvironments. InArtificialIntelligence, RoboticsandAutoma- tioninSpace, volume 440, page 433, 1999.
12. T. Malisiewicz. TheFutureofReal-Time SLAM and “DeepLearningvs SLAM” . http://www.computervisionblog.com/2016/01/why-slam-matters-future-of-real-time.html, 2016.
13. A. Fitzgibbonand A. Zisserman. Automaticcamerarecoveryforclosedoropenimagesequences. Computer VisionECCV’98, pages 311–326, 1998.
14. A. Fitzgibbon, G. Cross, and A. Zisserman. Automatic 3d modelconstructionforturn-tablesequences. 3D StructurefromMultipleImagesofLarge-ScaleEnvironments, pages 155–170, 1998.
15. M. Pollefeys, R. Koch, and L. VanGool. A simpleandefficientrectificationmethodforgeneralmotion. InComputerVision, 1999. TheProceedingsoftheSeventh IEEE InternationalConferenceon, volume 1, pages 496–501. IEEE, 1999.
16. M. Pollefeys. Self-calibrationandmetric 3D reconstructionfromuncali- bratedimagesequences. PhDthesis, 1999.
17. R. Hartleyand A. Zisserman. Multipleviewgeometryincomputervision. Cambridgeuniversitypress, 2003.
18. A. J. Davison. Real-timesimultaneouslocalisationandmappingwith a singlecamera. InComputerVision, 2003. Proceedings. Ninth IEEE InternationalConferenceon, pages 1403–1410. IEEE, 2003.
19. A. J. Davisonand D. W. Murray. Simultaneouslocalizationandmap- buildingusingactivevision. PatternAnalysisandMachineIntelligence, IEEE Trans- actionson, 24(7):865–880, 2002.

20. A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-timesinglecameraslam. PatternAnalysisandMachineIntelligence, IEEE Transactionson, 29(6):1052–1067, 2007.
21. M. Pupilliand A. Calway. Real-timecameratrackingusing a particlefilter. In BMVC, 2005.
22. M. Pupilliand A. Calway. Real-timevisualslamwithresiliencetoerraticmotion. In 2006 IEEE ComputerSocietyConferenceonComputerVisionandPatternRecognition (CVPR'06), volume 1, pages 1244–1249, June 2006.
23. M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, etal. Fastslam: A fac-toredsolutiontothesimultaneouslocalizationandmappingproblem. In AAAI/IAAI, pages 593–598, 2002.
24. D. Nister, O. Naroditsky, and J. Bergen. Visualodometry. InComputerVisionandPatternRecognition, 2004. CVPR 2004. Proceedingsofthe 2004 IEEE ComputerSocietyConferenceon, volume 1, pages I–652–I–659 Vol.1, June 2004.
25. J. Montiel, J. Civera, and A. J. Davison. Unifiedinversedepthparametrizationformonocularslam. analysis, 9:1, 2006.
26. G. Kleinand D. Murray. Paralleltrackingandmappingforsmallarworkspaces. InMixedandAugmentedReality, 2007. ISMAR 2007. 6th IEEE and ACM InternationalSymposiumon, pages 225–234. IEEE, 2007.
27. R. Mur-Artal, J. M. M. Montiel, and J. D. Tards. Orb-slam: A versa-tileandaccuratemonocularslamsystem. IEEE TransactionsonRobotics, 31(5):1147–1163, Oct 2015.
28. C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fastsemi-directmonocularvisualodometry. InProc. IEEE Intl. Conf. onRoboticsandAutomation, 2014.

29. G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A generalframe- workforgraphoptimization. In IEEE InternationalConferenceonRoboticsandAutomation, 2011.
30. E. Eadeand T. Drummond. Monocularslamas a graphofcoalescedobservations. In 2007 IEEE 11th InternationalConferenceonComputerVision, pages 1–8, Oct 2007.
31. F. Dellaertand M. Kaess. Squarerootsam: Simultaneouslocalizationandmappingviasquarerootinformationsmoothing. TheInternationalJournalofRoboticsResearch, 25(12):1181–1203, 2006.
32. M. Kaess, A. Ranganathan, and F. Dellaert. isam: Incrementalsmoothingandmapping. Robotics, IEEE Transactionson, 24(6):1365–1378, Dec 2008.
33. M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. isam2: Incrementalsmoothingandmappingwithfluidrelinearizationandincrementalvariablereordering. InRoboticsandAutomation (ICRA), 2011 IEEE InternationalConferenceon, pages 3281–3288, May 2011.
34. M. Cumminsand P. Newman. Fab-map: Probabilisticlocalizationandmappinginthespaceofappearance. TheInternationalJournalofRoboticsRe- search, 27(6):647–665, 2008.
35. S. Lowry, N. Snderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visualplacerecognition: A survey. IEEE TransactionsonRobotics, 32(1):1–19, Feb 2016.
36. E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: Anefficientalternativetosiftorsurf. In 2011 InternationalConferenceonComputerVision, pages 2564–2571, Nov 2011.

37. J. Engel, T. Sch ̈ops, and D. Cremers. Lsd-slam: Large-scaledirectmonocularslam. InComputerVision–ECCV 2014, pages 834–849. Springer, 2014.
38. R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Densetrackingandmappinginreal-time. InComputerVision (ICCV), 2011 IEEE Interna- tionalConferenceon, pages 2320–2327. IEEE, 2011.
39. Australiancentreforroboticvisionpublic. https://roboticvision.atlassian.net/wiki/display/PUB/RVSS2015+Robotic+Vision+Summer+School+Presentations, 2016
40. R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. Slam++: Simultaneouslocalisationandmappingatthelevelofobjects. InComputerVisionandPatternRecognition (CVPR), 2013 IEEE Conferenceon, pages 1352–1359, June 2013.
41. А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник Комп’ютерні мережі. Книга 1. [навчальний посібник] (Лист МОНУ №1/11-8052 від 28.05.12р.) - Львів, "Магнолія 2006", 2013. – 256 с.
42. А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник Комп’ютерні мережі. Книга 2. [навчальний посібник] (Лист МОНУ №1/11-11650 від 16.07.12р.) - Львів, "Магнолія 2006", 2014. – 312 с.
43. Микитишин А.Г., Митник, П.Д. Стухляк. Комплексна безпека інформаційних мережевих систем: навчальний посібник – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016. – 256 с.
44. Микитишин А.Г., Митник М.М., Стухляк П.Д. Телекомунікаційні системи та мережі : навчальний посібник для студентів спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2017 – 384 с.

ДОДАТКИ

Додаток А. Основний файл програмного коду

```
#include<iostream>
#include<algorithm>
#include<fstream>
#include<chrono>

#include<ros/ros.h>
#include <cv_bridge/cv_bridge.h>

#include<opencv2/core/core.hpp>

#include"../../../include/System.h"

usingnamespacestd;

classImageGrabber
{
public:
ImageGrabber(ORB_SLAM2::System* pSLAM):mpSLAM(pSLAM){}

voidGrabImage(constsensor_msgs::ImageConstPtr&msg);

   ORB_SLAM2::System* mpSLAM;
};

intmain(intargc, char **argv)
{
ros::init(argc, argv, "Mono");
ros::start();
```

```
if(argc != 3)
   {
cerr<<endl<< "Usage: rosrun ORB_SLAM2 Monopath_to_vocabularypath_to_settings" <<endl;
ros::shutdown();
return 1;
   }

   // Create SLAM system. Itinitializesallsystemthreadsandgetsreadytoprocessframes.
   ORB_SLAM2::System SLAM(argv[1],argv[2],ORB_SLAM2::System::MONOCULAR,true);

ImageGrabberigb(&SLAM);

ros::NodeHandlenodeHandler;
ros::Subscribersub = nodeHandler.subscribe("/camera/image_raw", 1, &ImageGrabber::GrabImage,&igb);

ros::spin();

   // Stopallthreads
SLAM.Shutdown();

   // Savecameratrajectory
SLAM.SaveKeyFrameTrajectoryTUM("KeyFrameTrajectory.txt");

ros::shutdown();

return 0;
}

voidImageGrabber::GrabImage(constsensor_msgs::ImageConstPtr&msg)
{
```

```
   // Copytherosimagemessageto cv::Mat.
cv_bridge::CvImageConstPtrcv_ptr;
try
   {
cv_ptr = cv_bridge::toCvShare(msg);
   }
catch (cv_bridge::Exception& e)
   {
      ROS_ERROR("cv_bridgeexception: %s", e.what());
return;
   }

   mpSLAM->TrackMonocular(cv_ptr->image,cv_ptr->header.stamp.toSec());
}
```

Додаток Б. Файл в якому обчислюється позиція робота

```
#include <iostream>

#include "PnPsolver.h"

#include <vector>
#include <cmath>
#include <opencv2/core/core.hpp>
#include "Thirdparty/DBoW2/DUtils/Random.h"
#include <algorithm>

usingnamespacestd;

namespace ORB_SLAM3
{
```

```
PnPsolver::PnPsolver(constFrame&F,
constvector<MapPoint*>&vpMapPointMatches):
pws(0), us(0), alphas(0), pcs(0), maximum_number_of_correspondences(0),
number_of_correspondences(0), mnInliersi(0),
mnIterations(0), mnBestInliers(0), N(0)
{
mvpMapPointMatches = vpMapPointMatches;
   mvP2D.reserve(F.mvpMapPoints.size());
   mvSigma2.reserve(F.mvpMapPoints.size());
   mvP3Dw.reserve(F.mvpMapPoints.size());
mvKeyPointIndices.reserve(F.mvpMapPoints.size());
mvAllIndices.reserve(F.mvpMapPoints.size());

intidx=0;
for(size_t i=0, iend=vpMapPointMatches.size(); i<iend; i++)
   {
      MapPoint* pMP = vpMapPointMatches[i];

if(pMP)
      {
if(!pMP->isBad())
         {
const cv::KeyPoint&kp = F.mvKeysUn[i];

            mvP2D.push_back(kp.pt);
            mvSigma2.push_back(F.mvLevelSigma2[kp.octave]);

            cv::MatPos = pMP->GetWorldPos();
            mvP3Dw.push_back(cv::Point3f(Pos.at<float>(0),Pos.at<float>(1),
Pos.at<float>(2)));

mvKeyPointIndices.push_back(i);
mvAllIndices.push_back(idx);

idx++;
```

```
        }
      }
   }

   // Setcameracalibrationparameters
fu = F.fx;
fv = F.fy;
uc = F.cx;
vc = F.cy;

SetRansacParameters();
}

PnPsolver::~PnPsolver()
{
delete [] pws;
delete [] us;
delete [] alphas;
delete [] pcs;
}


voidPnPsolver::SetRansacParameters(doubleprobability,  intminInliers,
intmaxIterations, intminSet, floatepsilon, float th2)
{
mRansacProb = probability;
mRansacMinInliers = minInliers;
mRansacMaxIts = maxIterations;
mRansacEpsilon = epsilon;
mRansacMinSet = minSet;

   N = mvP2D.size(); // numberofcorrespondences

mvbInliersi.resize(N);
```

```
   // AdjustParametersaccordingtonumberofcorrespondences
intnMinInliers = N*mRansacEpsilon;
if(nMinInliers<mRansacMinInliers)
nMinInliers=mRansacMinInliers;
if(nMinInliers<minSet)
nMinInliers=minSet;
mRansacMinInliers = nMinInliers;

if(mRansacEpsilon<(float)mRansacMinInliers/N)
mRansacEpsilon=(float)mRansacMinInliers/N;

   // Set RANSAC iterationsaccordingtoprobability, epsilon, andmaxiterations
intnIterations;

if(mRansacMinInliers==N)
nIterations=1;
else
nIterations = ceil(log(1-mRansacProb)/log(1-pow(mRansacEpsilon,3)));

mRansacMaxIts = max(1,min(nIterations,mRansacMaxIts));

mvMaxError.resize(mvSigma2.size());
for(size_t i=0; i<mvSigma2.size(); i++)
mvMaxError[i] = mvSigma2[i]*th2;
}

cv::MatPnPsolver::find(vector<bool>&vbInliers, int&nInliers)
{
boolbFlag;
returniterate(mRansacMaxIts,bFlag,vbInliers,nInliers);
}

cv::MatPnPsolver::iterate(intnIterations, bool&bNoMore, vector<bool>&vbInliers,
int&nInliers)
{
```

```
bNoMore = false;
vbInliers.clear();
nInliers=0;

set_maximum_number_of_correspondences(mRansacMinSet);

if(N<mRansacMinInliers)
   {
bNoMore = true;
return cv::Mat();
   }

vector<size_t>vAvailableIndices;

intnCurrentIterations = 0;
while(mnIterations<mRansacMaxIts || nCurrentIterations<nIterations)
   {
nCurrentIterations++;
mnIterations++;
reset_correspondences();

vAvailableIndices = mvAllIndices;

      // Getminsetofpoints
for(short i = 0; i <mRansacMinSet; ++i)
      {
intrandi = DUtils::Random::RandomInt(0, vAvailableIndices.size()-1);

intidx = vAvailableIndices[randi];


add_correspondence(mvP3Dw[idx].x,mvP3Dw[idx].y,mvP3Dw[idx].z,mvP2D[idx
].x,mvP2D[idx].y);

vAvailableIndices[randi] = vAvailableIndices.back();
```

```
vAvailableIndices.pop_back();
      }

      // Computecamerapose
compute_pose(mRi, mti);

      // Checkinliers
CheckInliers();

if(mnInliersi>=mRansacMinInliers)
      {
         // Ifitisthebestsolutionsofar, saveit
if(mnInliersi>mnBestInliers)
         {
mvbBestInliers = mvbInliersi;
mnBestInliers = mnInliersi;

            cv::MatRcw(3,3,CV_64F,mRi);
            cv::Mattcw(3,1,CV_64F,mti);
Rcw.convertTo(Rcw,CV_32F);
tcw.convertTo(tcw,CV_32F);
mBestTcw = cv::Mat::eye(4,4,CV_32F);
Rcw.copyTo(mBestTcw.rowRange(0,3).colRange(0,3));
tcw.copyTo(mBestTcw.rowRange(0,3).col(3));
         }

if(Refine())
         {
nInliers = mnRefinedInliers;
vbInliers = vector<bool>(mvpMapPointMatches.size(),false);
for(int i=0; i<N; i++)
            {
if(mvbRefinedInliers[i])
vbInliers[mvKeyPointIndices[i]] = true;
            }
```

```
returnmRefinedTcw.clone();
          }

       }
   }

if(mnIterations>=mRansacMaxIts)
   {
bNoMore=true;
if(mnBestInliers>=mRansacMinInliers)
       {
nInliers=mnBestInliers;
vbInliers = vector<bool>(mvpMapPointMatches.size(),false);
for(int i=0; i<N; i++)
          {
if(mvbBestInliers[i])
vbInliers[mvKeyPointIndices[i]] = true;
          }
returnmBestTcw.clone();
       }
   }

return cv::Mat();
}

boolPnPsolver::Refine()
{
vector<int>vIndices;
vIndices.reserve(mvbBestInliers.size());

for(size_t i=0; i<mvbBestInliers.size(); i++)
   {
if(mvbBestInliers[i])
       {
vIndices.push_back(i);
```

```
        }
    }

set_maximum_number_of_correspondences(vIndices.size());

reset_correspondences();

for(size_t i=0; i<vIndices.size(); i++)
    {
intidx = vIndices[i];

add_correspondence(mvP3Dw[idx].x,mvP3Dw[idx].y,mvP3Dw[idx].z,mvP2D[idx
].x,mvP2D[idx].y);
    }

    // Computecamerapose
compute_pose(mRi, mti);

    // Checkinliers
CheckInliers();

mnRefinedInliers =mnInliersi;
mvbRefinedInliers = mvbInliersi;

if(mnInliersi>mRansacMinInliers)
    {
        cv::MatRcw(3,3,CV_64F,mRi);
        cv::Mattcw(3,1,CV_64F,mti);
Rcw.convertTo(Rcw,CV_32F);
tcw.convertTo(tcw,CV_32F);
mRefinedTcw = cv::Mat::eye(4,4,CV_32F);
Rcw.copyTo(mRefinedTcw.rowRange(0,3).colRange(0,3));
tcw.copyTo(mRefinedTcw.rowRange(0,3).col(3));
returntrue;
    }
```

```
returnfalse;
}


voidPnPsolver::CheckInliers()
{
mnInliersi=0;

for(int i=0; i<N; i++)
   {
      cv::Point3f P3Dw = mvP3Dw[i];
      cv::Point2f P2D = mvP2D[i];

floatXc = mRi[0][0]*P3Dw.x+mRi[0][1]*P3Dw.y+mRi[0][2]*P3Dw.z+mti[0];
floatYc = mRi[1][0]*P3Dw.x+mRi[1][1]*P3Dw.y+mRi[1][2]*P3Dw.z+mti[1];
floatinvZc                                                                    =
1/(mRi[2][0]*P3Dw.x+mRi[2][1]*P3Dw.y+mRi[2][2]*P3Dw.z+mti[2]);

doubleue = uc + fu * Xc * invZc;
doubleve = vc + fv * Yc * invZc;

floatdistX = P2D.x-ue;
floatdistY = P2D.y-ve;

float error2 = distX*distX+distY*distY;

if(error2<mvMaxError[i])
      {
mvbInliersi[i]=true;
mnInliersi++;
      }
else
      {
mvbInliersi[i]=false;
```

```
      }
   }
}


voidPnPsolver::set_maximum_number_of_correspondences(int n)
{
if (maximum_number_of_correspondences< n) {
if (pws != 0) delete [] pws;
if (us != 0) delete [] us;
if (alphas != 0) delete [] alphas;
if (pcs != 0) delete [] pcs;

maximum_number_of_correspondences = n;
pws = newdouble[3 * maximum_number_of_correspondences];
us = newdouble[2 * maximum_number_of_correspondences];
alphas = newdouble[4 * maximum_number_of_correspondences];
pcs = newdouble[3 * maximum_number_of_correspondences];
  }
}

voidPnPsolver::reset_correspondences(void)
{
number_of_correspondences = 0;
}

voidPnPsolver::add_correspondence(double X, double Y, double Z, double u,
double v)
{
pws[3 * number_of_correspondences    ] = X;
pws[3 * number_of_correspondences + 1] = Y;
pws[3 * number_of_correspondences + 2] = Z;

us[2 * number_of_correspondences    ] = u;
us[2 * number_of_correspondences + 1] = v;
```

```
number_of_correspondences++;
}

voidPnPsolver::choose_control_points(void)
{
  // Take C0 asthereferencepointscentroid:
cws[0][0] = cws[0][1] = cws[0][2] = 0;
for(int i = 0; i <number_of_correspondences; i++)
for(int j = 0; j < 3; j++)
cws[0][j] += pws[3 * i + j];

for(int j = 0; j < 3; j++)
cws[0][j] /= number_of_correspondences;


  // Take C1, C2, and C3 from PCA onthereferencepoints:
CvMat * PW0 = cvCreateMat(number_of_correspondences, 3, CV_64F);

double pw0tpw0[3 * 3], dc[3], uct[3 * 3];
CvMat PW0tPW0 = cvMat(3, 3, CV_64F, pw0tpw0);
CvMat DC      = cvMat(3, 1, CV_64F, dc);
CvMat UCt     = cvMat(3, 3, CV_64F, uct);

for(int i = 0; i <number_of_correspondences; i++)
for(int j = 0; j < 3; j++)
     PW0->data.db[3 * i + j] = pws[3 * i + j] - cws[0][j];

cvMulTransposed(PW0, &PW0tPW0, 1);
cvSVD(&PW0tPW0, &DC, &UCt, 0, CV_SVD_MODIFY_A | CV_SVD_U_T);

cvReleaseMat(&PW0);

for(int i = 1; i < 4; i++) {
double k = sqrt(dc[i - 1] / number_of_correspondences);
```

```
for(int j = 0; j < 3; j++)
cws[i][j] = cws[0][j] + k * uct[3 * (i - 1) + j];
  }
}

voidPnPsolver::compute_barycentric_coordinates(void)
{
double cc[3 * 3], cc_inv[3 * 3];
CvMat CC     = cvMat(3, 3, CV_64F, cc);
CvMatCC_inv = cvMat(3, 3, CV_64F, cc_inv);

for(int i = 0; i < 3; i++)
for(int j = 1; j < 4; j++)
     cc[3 * i + j - 1] = cws[j][i] - cws[0][i];

cvInvert(&CC, &CC_inv, CV_SVD);
double * ci = cc_inv;
for(int i = 0; i <number_of_correspondences; i++) {
double * pi = pws + 3 * i;
double * a = alphas + 4 * i;

for(int j = 0; j < 3; j++)
     a[1 + j] =
         ci[3 * j    ] * (pi[0] - cws[0][0]) +
         ci[3 * j + 1] * (pi[1] - cws[0][1]) +
         ci[3 * j + 2] * (pi[2] - cws[0][2]);
   a[0] = 1.0f - a[1] - a[2] - a[3];
  }
}

voidPnPsolver::fill_M(CvMat * M,
                  constintrow, constdouble * as, constdouble u, constdouble v)
{
double * M1 = M->data.db + row * 12;
double * M2 = M1 + 12;
```

```
for(int i = 0; i < 4; i++) {
    M1[3 * i    ] = as[i] * fu;
    M1[3 * i + 1] = 0.0;
    M1[3 * i + 2] = as[i] * (uc - u);

    M2[3 * i    ] = 0.0;
    M2[3 * i + 1] = as[i] * fv;
    M2[3 * i + 2] = as[i] * (vc - v);
  }
}

} //namespace ORB_SLAM
```