Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Розроблення робототехнічної платформи для моніторингу виробничого середовища на основі IoT Hab Azure
(тема)

Виконав:
студент 2 курсу, групи КІТПВм-20-1

Гаврик С. С.
(прізвище, ініціали)

Спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)

Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва
(повна назва освітньої програми)

Керівник проф. Невлюдов І. Ш.
(посада, прізвище, ініціали)

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

2021р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАМ
Рівень вищої освіти Другий (магістерський)
Спеціальність 151 Автоматизація та комп`ютерно-інтегровані технології
Тип програми Освітньо-професійна
Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАМ _____
(підпис)
« ___»_____2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Гаврику Станіславу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення робототехнічної платформи для моніторингу виробничого середовища на основі IoT Hab Azure
Затверджена наказом по університету від 08.11.2021 р. № 1697 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 15.12.2021 р.
3. Вихідні дані до роботи 3.1 Робототехнічна платформа являє собою шести-колісну платформу з чотирма приводами, розташованими симетрично; ;
3.2 Основними компонентами системи є РП, сервер і клієнт;
3.3 Ескіз РП (вид зверху та спереду

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз технічного завдання; Вступ; Аналіз об'єкту розробки; Аналіз існуючих апаратних рішень необхідних для розробки платформи (системи пересування, модулі керування і обробки інформації та сенсорні системи); Аналіз та визначення набору програмних інструментів для забезпечення роботи системи; Проектування схеми апаратного забезпечення та 3D-моделі прототипу; Розробка програмного забезпечення для пересування платформи, алгоритми комунікації сенсорів один з одним та із сервером; Забезпечення безпечних умов праці при створенні моделі захватного пристрою маніпулятора, Висновки; Додатки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Графічний матеріал у вигляді презентації – 11 арк. ф. А 4

6. Консультанти розділів роботи

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз об'єкту розробки	1.09	виконано
2	Аналіз існуючих апаратних рішень	20.09	виконано
3	Аналіз та визначення набору програмних інструментів	10.10	виконано
4	Проектування схеми апаратного забезпечення. Проектування 3D-моделі прототипу	25.10	виконано
5	Розробка програмного забезпечення для пересування платформи	25.11	виконано
6	Подання роботи на перевірку Інтернет-сервісом Unicheck	01.12	виконано
7	Оформлення пояснювальної записки	05.12	виконано
8	Подання роботи на рецензію	09.12	виконано
9	Подання роботи на підпис зав. кафедри	10.12	виконано
11	Подання кваліфікаційної роботи в ЕК	15.12	виконано

Дата видачі завдання 01.09.2021 р.

Студент _____ Гаврик С. С.
(підпис)

Керівник роботи _____ проф. Невлюдов І. Ш.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 67 с., 2 табл., 70 рис., 1 дод., 32 джерел.

МОБІЛЬНІ ПЛАТФОРМИ, СИСТЕМИ ПЕРЕСУВАННЯ, АРХИТЕКТУРА СИСТЕМИ, АЛГОРИТМ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МОДУЛЬ ВІДОБРАЖЕННЯ

Об’єктом дослідження є процес проектування апаратної платформи, а також процес її використання.

Предмет дослідження – компонентна база, технології розробки та ефективність використання рухомої охоронної платформи.

Метою кваліфікаційної роботи є вирішення задачі проектування моделі автономної рухомої охоронної платформи, покращення цінової політики завдяки правильно підібраній компонентній базі

Проведено аналіз існуючих апаратних рішень необхідних для розробки платформи (системи пересування, модулі керування і обробки інформації та сенсорні системи).

За результатами аналізу було спроектовано модуль авторизації для веб-платформи, розроблено модулі відображення прямої трансляції інформації з системи, а також записаних даних та розроблено код для передачі і зберігання інформації на сервері.

ABSTRACT

Explanatory note: 67 pages, 2 tables, 70 figures, 1 appendix, 32 sources

MOBILE PLATFORMS, MOVEMENT SYSTEMS, ARCHITECTURE OF SYSTEM, ALGORITHM, SOFTWARE, DISPLAY MODYLE

The object of research is the process of designing a hardware platform, as well as the process of its use.

The subject of research is the component base, development technologies and efficiency of mobile security platform use.

The purpose of the qualification work is to solve the problem of designing a model of an autonomous mobile security platform, improving pricing policy through a properly selected component base.

The analysis of the existing hardware solutions necessary for the development of the platform (mobile systems, control and information processing modules and sensor systems) is carried out.

Based on the results of the analysis, an authorization module for the web platform was designed, modules for displaying live information from the system, as well as recorded data were developed, and code for transmitting and storing information on the server was developed.

ЗМІСТ

Перелік скорочень 6
Вступ 7
1 Аналіз технічного завдання 9
1.1 Класифікація роботів 9
1.2 Системи управління мобільними платформами 11
1.3 Аналіз принципу роботи систем пересування 15
2 Розробка програмно-апаратної моделі тактично-рефлексивного рівня мобільного робота 19
2.1 Математична модель мобільного робота 19
2.2 Розрахунок приводів МР 21
2.3 Опис коліс 22
2.3.1 Фіксовані колеса 23
2.3.2 Центрально орієнтовані колеса 24
2.3.3 Нецентрально-орієнтовані колеса (кастор колеса) 24
3 Вибір технологій та рішень для розробки апаратної платформи 26
3.1 Архітектура системи 26
3.1.1 Огляд одноплатних комп’ютерів для керування РП 27
3.2 Вибір платформи, ескізування та створення 3D-моделі РП 33
3.2.1 Розробка 3D-моделі 34
3.3 Програмне забезпечення компонентів рухомої платформи 38
3.4 Реалізація алгоритму роботи рухомої платформи 44
3.5 Програмне забезпечення веб-платформи 49
3.6 Заходи із забезпечення умов праці розробника роботизовано платформи 60
Висновки 63
Перелік посилань 64
Додаток А Демонстраційний матеріал 68

ПЕРЕЛІК СКОРОЧЕНЬ

ІРД – інформаційно-рухових дій;

МР – мобільний робот;

СУ – системи управління.

ВСТУП

В даний час все частіше на виробництві та в багатьох інших сферах діяльності замінюють роботи. У промисловості використовують стаціонарні роботи для складання, зварювання та фарбування та інших технологічних операцій. Здебільшого промислові роботи є стаціонарними. В інших же сферах, наприклад, розбір завалів, військових дій, розвідки, роботи в особливо небезпечній зоні використовуються мобільні роботи [1]. Таким чином, машина виконує безліч функцій, які раніше виконувала людина, а людині нічого не загрожує.

Роботизовані охоронні платформи особливо ефективні на великих ділянках, де приміщення можна розділити на місцеві охоронні зони і де мобільні роботи можуть здійснювати відео спостереження [2]. Кожен робот патрулює власну місцевість, забезпечуючи всебічний контроль над найвіддаленішими частинами. У той же час мобільний підрозділ співробітників служби безпеки може бути готовий швидко дістатися до місць вторгнення у відповідь на сигнал тривоги.

Поєднуючи можливості різних моделей роботів, можна зменшити кількість персоналу на охоронюваних об'єктах, автоматизувати деякі загальні функції безпеки, негайно посилити безпеку, коли це необхідно, та уникнути змін у якості робіт із забезпечення безпеки, що виконуються персоналом. Використовуючи охоронних роботів, компанії, що надають послуги охорони з працівниками служби безпеки, можуть швидко збільшити кількість своїх охоронюваних місць.

Дослідження проводиться в області проектування апаратної та програмної частини рухомої охоронної платформи, та вибору необхідних компонентів. Дослідження у цій області допоможуть визначити переваги вибору окремих технічних рішень та доцільність використання рухомих охоронних платформ у порівнянні із стаціонарними.

Об’єктом дослідження є процес проектування апаратної платформи, а також процес її використання.

Предмет дослідження – компонентна база, технології розробки та ефективність використання рухомої охоронної платформи.

Метою кваліфікаційної роботи є вирішення задачі проектування моделі автономної рухомої охоронної платформи, покращення цінової політики завдяки правильно підібраній компонентній базі.

Для досягнення поставленої мети в роботі поставлені такі задачі:

– проаналізувати існуючі апаратні рішення необхідні для розробки платформи (системи пересування, модулі керування і обробки інформації та сенсорні системи);

– проаналізувати та визначити набір програмних інструментів для забезпечення роботи системи;

– спроектувати схему апаратного забезпечення та 3D-модель прототипу;

– розробити програмне забезпечення для пересування платформи, алгоритми комунікації сенсорів один з одним та із сервером.

Робота виконана згідно з [3–5]. Результати дослідження опубліковані в [6].

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Класифікація роботів

Роботи можна класифікувати за [8, 9]:

– сферою застосування: виробничі (промислові), військові (бойові, що забезпечують), дослідницькі, медичні;

– середовищем експлуатації: наземні, підземні, надводні, підводні, повітряні, космічні;

– ступенем рухливості: стаціонарні, мобільні;

– типом системи управління: програмні, адаптивні, інтелектуальні;

– функціональним призначенням: маніпуляційні, транспортні, інформаційні, комбіновані;

– рівнем універсальності: спеціальні, спеціалізовані, універсальні;

– конструктивним ознакам;

– типом виконавчих приводів: електричні, гідравлічні, пневматичні;

– типом рушія: гусеничні, колісні, колісно-гусеничні, напівгусеничні, крокуючі, колісно-крокові, роторні, з петлевим, гвинтовим, водометним та реактивним рушіями;

– конструктивним особливостям технологічного обладнання: за кількістю маніпуляторів, за вантажопідйомністю маніпуляторів, системою координат робочої зони (лінійна, кутова);

– типом джерел первинних керуючих сигналів: електричні, біоелектричні, акустичні;

– способом управління: автоматичні, дистанційно керовані (копіюючі, командні, інтерактивні, супервізорні, діалогові), ручні (шарнірно-балансні, екзоскелетонні).

Умови функціонування роботів, що визначаються типом середовища експлуатації та характером робочого процесу, можна розділити на дві

категорії: детерміновані (певні) та недетерміновані (невизначені). До детермінованих середовищ відносяться середовища, спроектовані та створені людиною [10].

Відповідно, детермінованим процесом є кожен процес, перебіг якого повністю залежить від цілеспрямованої діяльності людини (діяльності з безпосереднього здійснення процесу, управління процесом тощо). У детермінованих середовищах вже є високий рівень організації, або необхідний рівень організації можна досягти при порівняно невеликих витратах. Визначеність середовища обумовлена апріорним знанням точного становища всіх об'єктів, із якими може взаємодіяти робот [11]. Для маніпуляційного робота [12] це означає точне знання розташування та орієнтації об'єктів, розташованих у його робочій зоні. Для транспортного робота детермінованим середовищем є, наприклад, рейкова траса цеху . До першої категорії відносяться також середовища, які можна організувати необхідним чином, хоч і ціною значних витрат (не повністю організовані середовища). І тут окремі об'єкти можуть мати заздалегідь невідомі відхилення від еталону. До цих середовищ можна віднести польові склади боєприпасів, паливно-мастильних матеріалів, технологічні позиції та ін [13].

У середовищі другої категорії практично неможливо здійснити їх організацію. Такі середовища називаються повністю неорганізованими (недетермінованими). До них відносяться, зокрема, природні середовища проживання і середовища, створювані аварійними ситуаціями як і природних умовах, і при руйнації середовищ, спроектованих і створених людиною, тобто, при руйнуваннях будівель та споруд. До дій робота в природних середовищах відносяться дії в польових умовах: розвідка на місцевості, військові дії, розмінування та патрулювання, підводні та підземні роботи тощо (у тому числі у випадках радіоактивного, хімічного та бактеріологічного зараження місцевості), а також дії з розчищення завалів, рятувальних робіт у зруйнованих спорудах тощо. До недетермінованих процесів відноситься кожен процес, протікання та результат якого повністю

не залежить від цілеспрямованої діяльності людини. Недетермінованими процесами є природні процеси (землетруси, виверження вулканів тощо), пожежі, вибухи (як результати техногенних аварій) тощо [13-15].

1.2 Системи управління мобільними платформами

Процес управління мобільним роботом (МР) реалізований, переважно, виконанням апріорно обмеженої безлічі команд зовнішнього інтерфейсу бортової системи управління (СУ). Це регламентує основне обмеження методів управління, що розробляються, створення переважно інформаційного забезпечення для управління кінцевим положенням мультиструктури МР в умовах протидій, що надаються зовнішнім середовищем. Системи управління даним класом об'єктів характеризуються суттєвою нестаціонарністю, оскільки зазвичай включають нестаціонарний контур зворотного зв'язку [16].

Інформаційна структура контуру зворотного зв'язку рівня пропріоцепції апаратно фіксована, на відміну від рівня екстероцепції, на якому вона має тенденцію до трансформації у більш комплексні інформаційно-керуючі форми, при яких модель процесу навігації переймає властивості моделей ієрархічних систем підтримки та прийняття рішень (СППР) [2] з широким використанням математичних структур як метричного, і топологічного просторів [17]. Насамперед, це викликано збільшенням числа та вдосконаленням мехатронних вузлів МР, що передбачає їх асинхронну роботу, для якої, як правило, потрібні процедури узгодження часткових обчислювальних процесів [20].

Розглядаючи абстракцію МР [2] – об'єкта, що автономно переміщається з безліччю інформаційно-рухових дій (ІРД), можна відзначити, що, на відміну від стаціонарних роботів, в управлінні якими є апріорна інформація [17] (цільова конфігурація виконавчих підсистем), управління МР пов'язано з відсутністю у явній формі такої інформації. Наявність у ній конфігурацій як

перешкод так і МР, як активного об'єкта середовища [10], значно ускладнює її використання. До теперішнього часу в сучасній робототехніці загальновідома схема структурної організації МР (рисунок 1.1), яка розподілена на прикладний, стратегічний та тактично-рефлексивний рівні взаємодії [2].

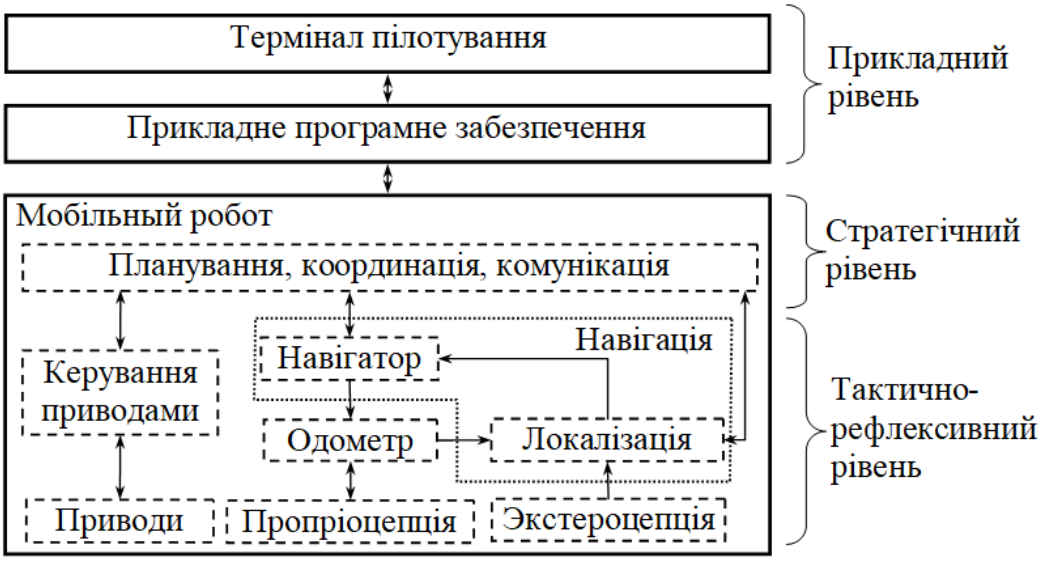

Рисунок 1.1 – Узагальнена структура організації МР

Тактично-рефлексивний рівень, включаючи повну групу виконавчих вузлів, узагальнює апаратно фіксовану структуру взаємодіючих модулів, що забезпечують з фіксованим ступенем свободи її переміщення [18]. До них, в залежності від призначення МР, можуть входити приводи, сенсори пропріоцепції та екстероцепції, одометри, модуль навігації. При цьому, основна схема взаємодії заснована на отриманні навігатором деякої послідовності, що управляє, дотримання якої забезпечує введення МР в цільовий стан; відповідно до продукованих одометрами даними верифікації поточного розташування МР; та локалізація власної конфігурації з використанням даних екстероцепції, результатом чого є визначення мети та формування траєкторії.

Первинні обмеження загального завдання навігації можна назвати

досліджуючи механічну зв'язність виконавчих вузлів МР (рисунок 1.2). Формально вони визначаються математичною моделлю руху МР як механічної системи взаємопов'язаних абсолютно твердих тіл, що реалізується на структурі орграфа [17]. Рівняння руху при цьому отримують виходячи із загальних теорем динаміки [2] та нелономної механіки [19]. Причому методи нелономної механіки виявляються тут більш переважними, оскільки в такому випадку модель може бути представлена у векторно-матричній формі [20], що допускає її реалізацію методами комп'ютерної алгебри [21].

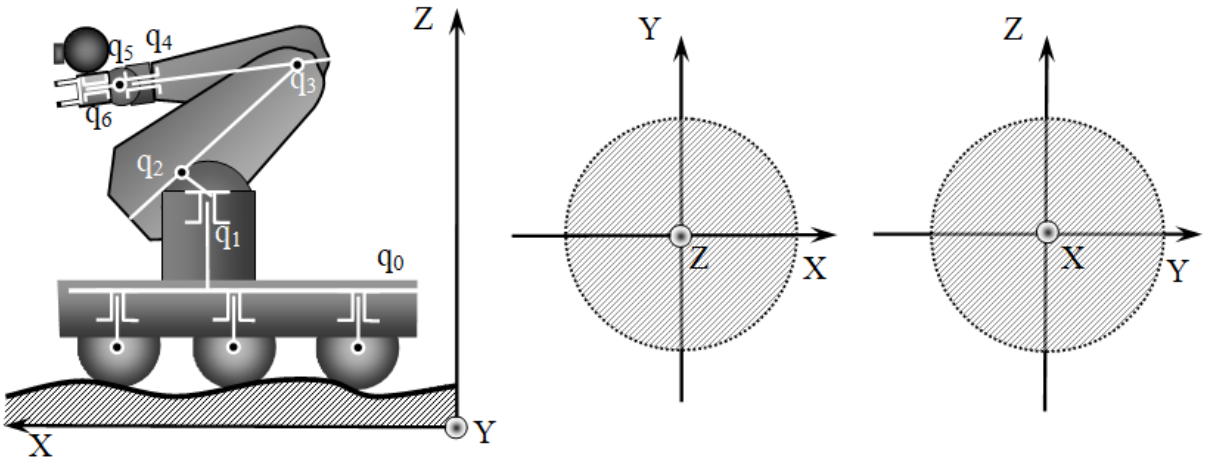

Рисунок 1.2 – Зв’язність виконавчих вузлів МР

Математично, зазначені обмеження можуть формувати безліч управляючих змінних $\boldsymbol{Q} = \{q_0, q_1, \ldots, q_N\}$ – векторів узагальнених координат конфігураційного простору станів МР. У навігації структура вектора на вході обмежена фазовими координатами СУ, відповідних розташування МР у зовнішньому середовищі [10] з деякою введеною у неї системою координат (СК). У зв'язку з цим, існуючі навігаційні системи є, головним чином, інформаційною надбудовою над СУ без будь-яких модифікацій наявної апаратної структури, і таким чином прикладний і стратегічний рівні реалізуються інформаційним забезпеченням у певному кінцевому конфігураційному просторі станів.

Слід зазначити, що прикладний рівень з інтерфейсом пілотування МР прийнято відокремлювати від процесу навігації через відсутність у відомих методах навігації схем аналізу досяжності цілей на етапі їх встановлення пілотом [22] або хоча б прийнятності створюваної моделі поведінки МР при виконанні конкретної місії [11]. Постулювання цього рівня вище стратегічного [22] призводить до виникнення складно вирішуваних завдань на узгодження елементів інтерфейсу, якими декларуються цілі, з виконавчо-руховими можливостями, що реалізуються. Очевидно, у майбутньому складність подібних завдань підвищитися через те, що інтерфейс має тенденцію еволюції до форм комунікації людей [11].

Істотне значення тут має асинхронний характер розгортання локальних СУ, що відповідає вимогам функціонування МР у масштабі «жорсткого» реального часу [23]. Відсутність методології побудови уніфікованої моделі об'єктів/перешкод [22] на основі СІ довільної форми, частково викликана відносністю реєстрованих сенсорами даних [17], як і відмінності використовуваних дескрипторів у значно обмежених методах обробки СІ, призводить до загальної неоднорідності структури навігаційних даних та моделі процесу навігації в цілому.

Аналіз архітектур існуючих МР [8] дає можливість виділити типові реляції (рис. 1.3) між рівнями організації МР. Насамперед, вони обумовлені алгоритмічними зв'язками взаємодіючих підпроцесів процесу навігації, що «розгортається» у часі як процесу функціонування абстрактної СППР, що включає стратегічний (глобальна навігація), тактичний (локальна навігація) та реактивний (оптимальне проходження спланованої траєкторії) вертикальні рівні взаємодій [10, 15].

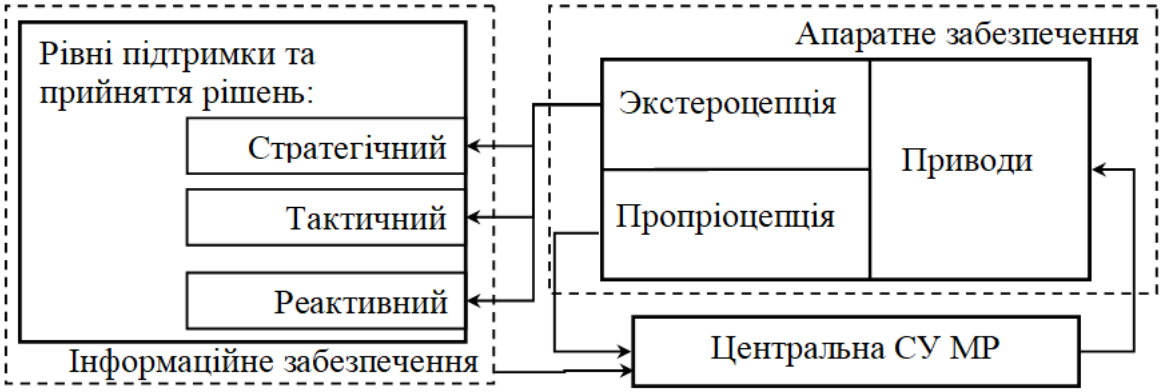

Рисунок 1.3 – Тактико-технічна організація МР

1.3 Аналіз принципу роботи систем пересування

Мобільному роботу необхідні певні механізми пересування, щоб він міг рухатися по своєму оточенню. Існує декілька механізмів для досягнення цієї мети. Механізми руху ніг роботів часто надихаються біологічними системами, які є дуже успішними у пересуванні по різних типах поверхонь.

Для кожного виду пересування, без різниці, використовуватиметься колесо, ноги чи інша концепція – є три основні проблеми: стабільність, характеристики контакту з землею та тип середовища пересування.

Основними атрибутами стабільності є кількість і геометрія точок контакту, сила тяжіння і нахил місцевості. Характеристики контакту із землею залежать від типу точки контакту (у випадку з ногами – це відбиток стопи), кут контакту з землею і тертя між роботом і поверхнею. Атрибутами типу середовища є структура середовища (наприклад, у випадку жорсткого середовища), і саме середовище (наприклад, вода, повітря, тверда або м’яка земля) [24].

У нашому випадку платформа буде використовуватися для роботи у приміщенні, на жорсткій та рівній поверхні. Виключення становлять різні покриття, такі як килими, лінолеум, стики між приміщеннями. Через це будемо використовувати колісну або гусеничну платформу пересування.

Пересувні колісні та гусеничні роботи мають мінімум два двигуни, які використовуються для приведення в рух і керування роботом [25]. Зазвичай обирають ковзаюче рульове через простоту інсталяції, конфігурації та управління. Використання третього заднього колеса, як правило, запобігає падінню робота. Чотириколісні роботи мають два, або чотири двигуна і використовують рульове управління. Шестиколісні роботи найчастіше мають два, чотири або шість двигунів. Збільшення кількості приводних двигунів допомагає роботу підніматися на більш круті нахили, за допомогою підвищеного крутного моменту [24].

Додавання коліс без двигуна часто призводить до втрати ваги з ведучих коліс, що призводить до ковзання та втрати тяги [26, 27]. На рис. 1.4 центральне колесо, обране помилково як приводне колесо, часто втрачає контакт із землею.

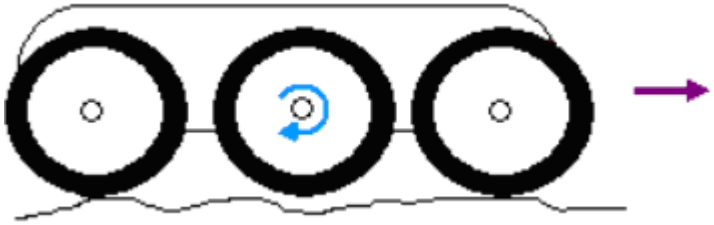

Рисунок 1.4 – Колісна база із центральним ведучим колесом

Система коліс (або зв’язування всіх коліс із шестернями або приводними ременями) також допомагає запобігти цьому. Колії – не обов'язково кращий вибір, ніж використання кількох приводних коліс. Виробники танків у всьому світі виробляють як колісні, так і гусеничні моделі танків, і обидві концепції демонструють майже однакові показники. Більшість користувачів погоджуються, що танкові колії – далеко не самий вишуканий спосіб транспортування і мають тенденцію "рвати" землю під собою. Роблячи щільний поворот або повертаючись на місці, протектори танка стикаються з значно більшим опором, ніж колеса, оскільки обидві

половини притискаються до землі перпендикулярно радіусу повороту. Саме через це такі платформи споживають дуже багато електроенергії.

Можливо розробити самобалансуючого робота, використовуючи два колеса. Або можливо розробити більш стабільну платформу, використовуючи чотири або більше [28].

Два ведучих колеса використовуються для приведення в рух і повороту робота (рульове керування), а одне або два холості колеса запобігають падінню робота вперед або назад. Колесо «холостого ходу» може бути роликом або кулею (рис. 1.5).

Рисунок 1.5 – Платформи на двох ведучих колесах та одному колесі «холостого ходу»

Омні-платформи (або Omniwheels) зазвичай мають три [29], розташовані трикутно, або чотири, розташовані під 90 градусів колеса (рис. 1.6). Omniwheels мають невеликі роликові колеса, які вільно обертаються. Зазвичай ведучим є центральне колесо. Це дозволяє рухати колесо вперед, але ковзати вбік.

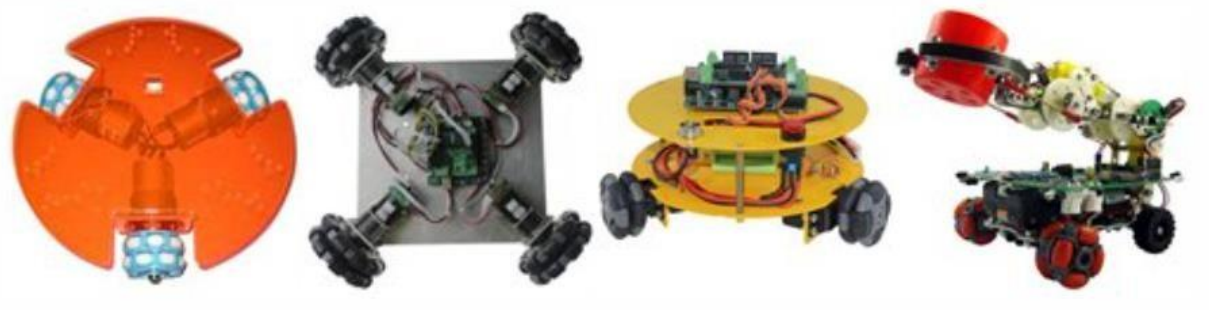

Рисунок 1.6 – Платформи Omniwheels

Правильний розмір колеса дуже важливий, і слід подбати, щоб вибрати правильний для кожного конкретного робота. Потрібно врахувати два основних рівняння:

– швидкість = кутова швидкість (колеса) х радіус (колеса)

Це означає, що і радіус колеса, і кутова швидкість, з якою воно обертається, впливатимуть на швидкість руху вперед.

– сила (що діє колесом на поверхню) = крутний момент (двигуна) / радіус (колеса)

Щоб колісний робот рухався або піднімався по нахилу, колесо прикладає горизонтальну силу на поверхню. Якщо вам потрібно прикласти велику силу (оскільки ваш робот важкий), то вам потрібно збільшити крутний момент або зменшити радіус колеса. Збільшення крутного моменту, як правило, дороге, а збільшення радіуса зазвичай означає важче колесо, що вимагає більше крутного моменту, але також зменшує максимальну швидкість (оскільки воно повинне обертатися швидше). Дуже малий діаметр коліс також може бути недоцільним [24].

2 РОЗРОБКА ПРОГРАМНО-АПАРАТНОЇ МОДЕЛІ ТАКТИЧНО-РЕФЛЕКСИВНОГО РІВНЯ МОБІЛЬНОГО РОБОТА

Мобільний робот – складна керована мехатронна система, що утворюється безліччю локальних інформаційно-керуючих підсистем [11], з одночасним протіканням як механічних процесів, що реалізують переміщення його механічної мультиструктури – деякої активної системи, так і інформаційних процесів, що реалізують інтерпретацію сенсорної інформації локалізації місця розташування перешкоджаючих і цільових об'єктів у протидіючому середовищу руху – деякої пасивної системи A_0. Репрезентація системи A_0, за умов апріорної невизначеності зовнішніх протидій, становить сутність проблемного середовища (ПС).

2.1 Математична модель мобільного робота

Нехтуючи прослизанням рушійних опор системи $\Re$ в точках їх дотику з поверхнею руху (неголономні зв'язки), математичну модель пересування МР можна представити рівняннями Лагранжа-Феррерса, що описує механічні реакції нелономних зв'язків [30]:

$$\frac{d}{dt}\left(\frac{\partial T_K}{\partial \dot{q}}\right)^T - \left(\frac{\partial T_K}{\partial q}\right)^T = Q + G^T\lambda, \qquad (2.1)$$

де $q = (q_0, q_1, \ldots, q_s)^T$ – s-мірний вектор узагальнених координат положення МР (верхній індекс T позначає операцію транспонування);

T_K – кінетична енергія системи: $\dfrac{\partial T_K}{\partial q} = \left|\dfrac{\partial T_K}{\partial q_0}\dfrac{\partial T_K}{\partial q_1}\ldots\dfrac{\partial T_K}{\partial q_s}\right|$;

Q - вектор узагальнених сил;

G – прямокутна матриця (l x s), елементи якої є функціями узагальнених координат, задовольняючи рівняння $G\dot{q} = 0_{vx1}$ з числом диференціальних нелономних стаціонарних зв'язків, що визначають набір псевдошвидкостей МР як механічної системи зв'язкових твердих тіл;

$\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_v)$ - мірний вектор множників Лагранжа.

Рівняння (2.1) у вихідній формі дозволяють математично описати взаємопов'язаність механічних характеристик МР (рис. 2.1) з тими інтелектуальними функціями, які можуть реалізовуватись у просторі узагальнених (керованих) координат інформаційним забезпеченням. Відомі різні спрощення розв'язання цих рівнянь [30] поряд з ефективними алгоритмами його розв'язання у масштабі реального часу [11].

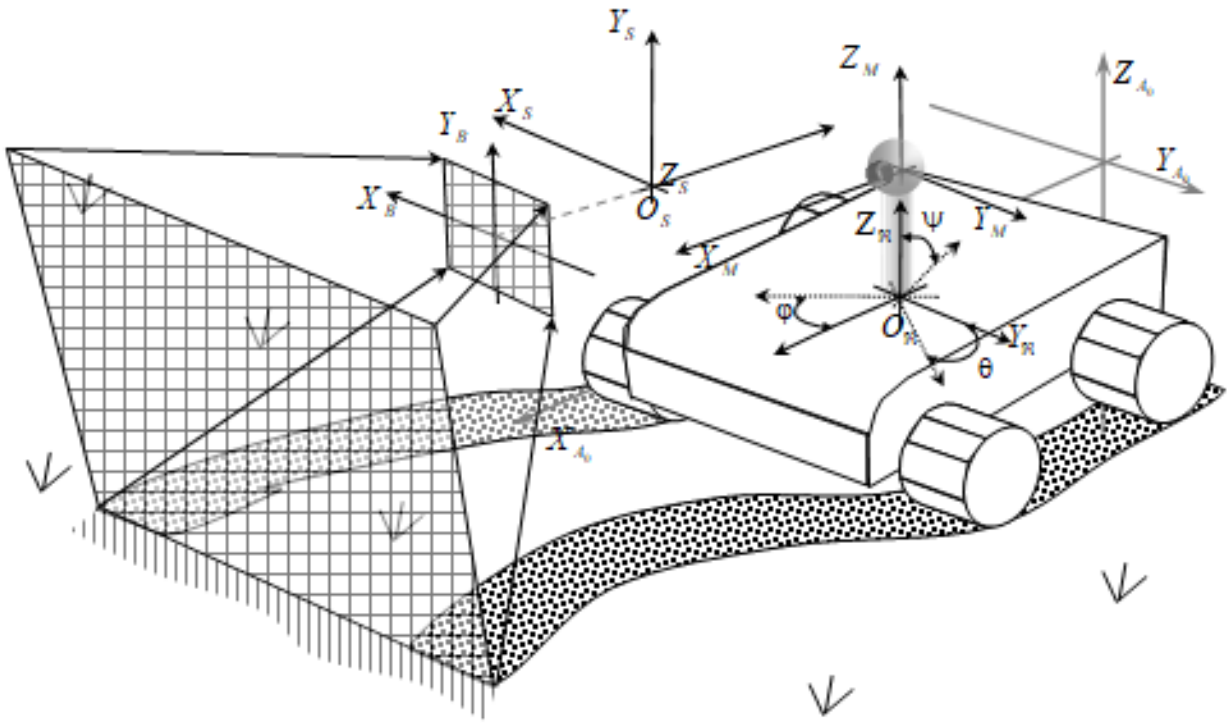

Рисунок 2.1 – Простір керованих координат у конфігурації МР

2.2 Розрахунок приводів МР

Колісний мобільний робот – це засіб пересування, з колесами і здатний здійснювати автономний рух (без зовнішнього впливу з боку людини). Ця здатність досягається тим, що на роботі встановлено кілька двигунів, керованих за допомогою комп'ютера, також розміщеного на корпусі робота. Припустимо, що мобільні роботи, що розглядаються в даній роботі, є твердою платформою (корпусом), забезпеченою колесами, що не деформуються. Рух роботів відбувається по нерухомій горизонтальній площині.

В такому випадку положення робота на площині описується як показано на рис. 2.2 [26, 27].

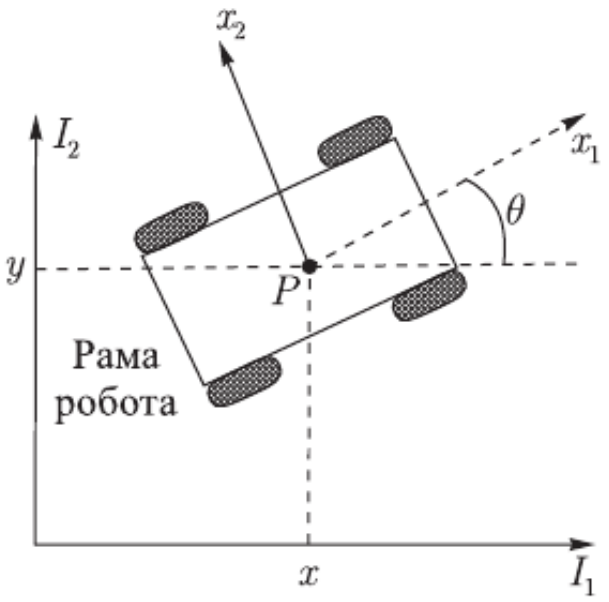

Рисунок 2.2 – Визначення положення [26, 27]

Довільна ортогональна нерухома система координат $\{O, \vec{I}_1, \vec{I}_2\}$ обрана у площині руху. На корпусі робота обрана довільна точка P, яка є початком рухомої системи координат $\{\vec{x}_1, \vec{x}_2\}$ жорстко пов'язаної з корпусом робота. Положення робота у разі повністю характеризується трьома змінними x, y, θ, де x, y – координати точки P в нерухомій системі координат, тобто

$\{OP = x\,\vec{I}_1 + y\,\vec{I}_2\}$, θ – визначає орієнтацію рухомого базису $\{\vec{x}_1, \vec{x}_2\}$ по відношенню до нерухомого базису $\{\vec{I}_1, \vec{I}_2\}$.

Визначаємо трикомпонентний вектор ξ, який описує положення робота:

$$\zeta \cong \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{2.2}$$

Визначимо також ортогональну матрицю повороту рухомої системи координат щодо нерухомої:

$$R(\theta) \cong \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

2.3 Опис коліс

Ми припускатимемо, що в процесі руху площина кожного колеса залишається вертикальною по відношенню до площини руху, і обертання кожного колеса відбувається навколо відповідної горизонтальної осі, що проходить через центр колеса. Однак орієнтація цієї осі щодо корпусу може бути фіксованою, так і не фіксованою. Відразу ж розділимо всілякі типи ідеальних (тобто недеформованих) коліс на два класи: традиційні колеса та омніколеса. Припустимо, що у кожному випадку контакт між колесом і площиною руху відбувається у одній точці, тобто, будемо надалі розглядати лише одноточковий контакт [26, 27].

Для традиційного колеса контакт між колесом і площиною руху відповідає умовам кочення без прослизання. Це означає, що швидкість точки колеса, що знаходиться в даний час у дотику до площини руху, дорівнює нулю. Звідси випливає, що компоненти швидкості цієї точки колеса

дорівнюють нулю як у проекції на напрям, що лежить у площині колеса, так і в проекції на напрям, ортогональне площині колеса.

2.3.1 Фіксовані колеса

Центр колеса, позначений *A*, є нерухомим у системі координат $\{\vec{x}_1, \vec{x}_2\}$ (рис. 2.3). Його положення щодо цієї системи характеризується за допомогою полярних координат, а саме довжини радіус-вектора $PA = l$ та кута α. Орієнтація площини колеса щодо напрямку *PA* характеризується постійним кутом β.

Кут повороту колеса щодо його горизонтальної осі позначимо $\varphi(t)$, а радіус колеса позначимо r [26, 27].

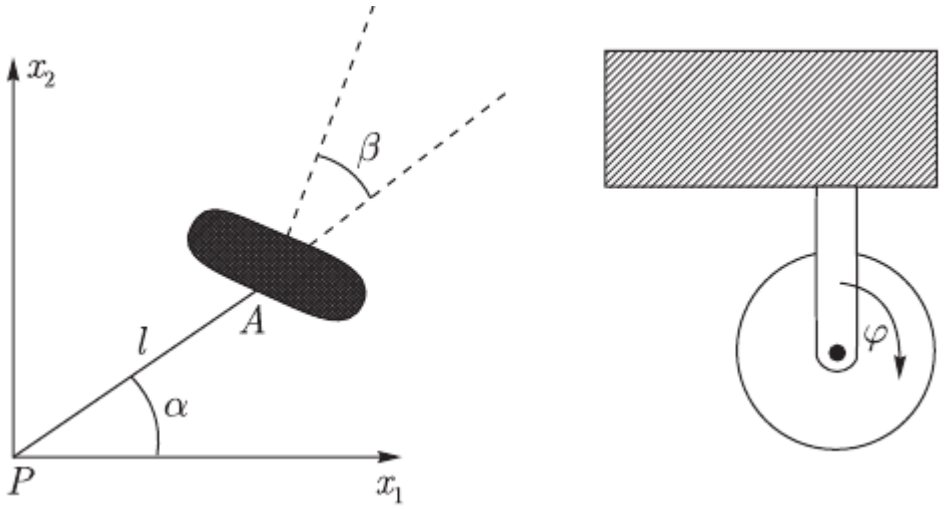

Рисунок 2.3 – Традиційне фіксоване та традиційне центрально орієнтоване колеса [26, 27]

Отже, положення колеса у разі характеризується чотирма постійними: $\alpha, \beta, l, r, \varphi$ його рух описується однією змінною $\varphi(t)$. При такому описі компоненти швидкості точки контакту легко обчислюються, і ми записуємо два зв'язки: вздовж площини колеса (2.4)

$$[-\sin(\alpha+\beta)\cos(\alpha+\beta)\, l\cos\beta]R(\theta)\zeta + r\varphi = 0, \qquad (2.4)$$

ортогонально площині колеса (2.5):

$$\left[\cos(\alpha+\beta)\ \sin(\alpha+\beta)\ l\ \sin\beta\right]R(\theta)\zeta = 0. \tag{2.5}$$

2.3.2 Центрально орієнтовані колеса

Центрально орієнтоване колесо – це таке колесо, що його площина під час руху робота може повертатися навколо вертикальної осі, що проходить через центр колеса (рис. 2.3). Опис такого колеса аналогічно до опису фіксованого колеса, за винятком тієї обставини, що тепер кут β не є постійним, а залежить від часу: $\beta = \beta(t)$. Положення колеса тепер характеризується трьома постійними величинами l, α, r, яке рух описується двома змінними кутами $\beta(t)$ і $\varphi(t)$. Зв'язки мають ту ж форму, що і (2.4), (2.5):

$$\left[-\sin(\alpha+\beta)\ \cos(\alpha+\beta)\ l\ \cos\beta\right]R(\theta)\zeta + r\varphi = 0, \tag{2.6}$$

$$\left[\cos(\alpha+\beta)\ \sin(\alpha+\beta)\ l\ \sin\beta\right]R(\theta)\zeta = 0. \tag{2.7}$$

2.3.3 Нецентрально орієнтовані колеса (кастор колеса)

Нецентрально орієнтоване колесо – це колесо, яке також може змінювати свою орієнтацію щодо корпусу робота, однак у цьому випадку поворот площини колеса відбувається навколо вертикальної осі, що не проходить через центр колеса (рис. 2.4).

В цьому випадку для опису колеса потрібно більше параметрів, ніж у випадках, описаних вище. Позначимо центр колеса через B і нехай цей центр пов'язаний з корпусом робота твердим стрижнем AB постійної довжини d, який може обертатися навколо вертикальної осі, що проходить через точку A.

Точка A сама собою є нерухомою у системі координат $\{\bar{x}_1, \bar{x}_2\}$ і, як і вище, характеризується полярними координатами l і α. Площина колеса завжди буде спрямована вздовж AB. У цьому випадку положення колеса

визначається чотирма постійними *α, l, r, d*, рух яких описується двома кутами *β(t)* і $\varphi(t)$.

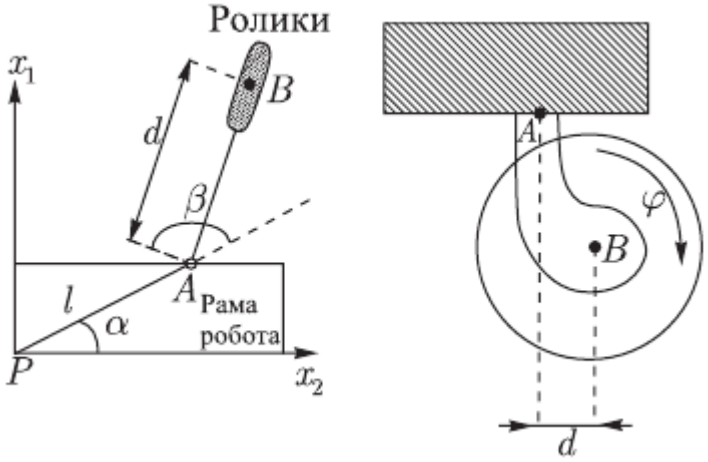

Рисунок 2.4 – Традиційні нецентрально орієнтовані колеса [26, 27]

З урахуванням цих позначень відповідні зв'язки мають вигляд:

$$[-\sin(\alpha+\beta)\cos(\alpha+\beta)\, l\cos\beta]R(\theta)\zeta + r\varphi = 0, \tag{2.8}$$

$$[\cos(\alpha+\beta)\sin(\alpha+\beta)\, d + l\sin\beta]R(\theta)\zeta + d\beta = 0. \tag{2.9}$$

3 ВИБІР ТЕХНОЛОГІЙ ТА РІШЕНЬ ДЛЯ РОЗРОБКИ АПАРАТНОЇ ПЛАТФОРМИ

3.1 Архітектура системи

Трьома основними компонентами системи є робототехнічна платформа (РП), сервер і клієнт (рис. 3.1).

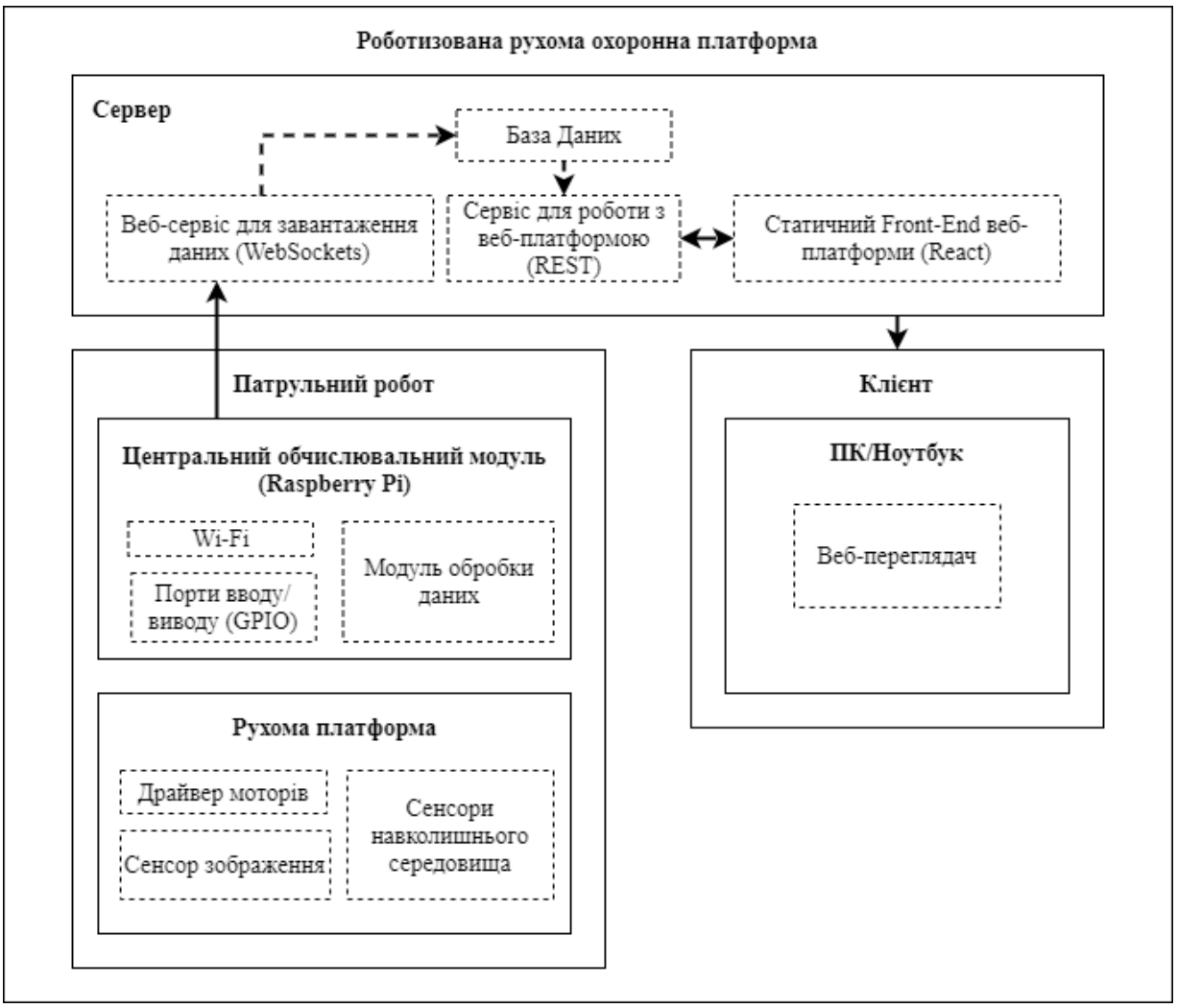

Рисунок 3.1 – Архітектура системи

Першим і одним з найголовніших модулів буде модуль камери (рис. 3.2). Модуль камери оснащений двома інтенсивними інфрачервоними

світлодіодними прожекторами для нічного запису. Інфрачервоні світлодіоди живляться безпосередньо від порту CSI і здатні освітлювати територію на відстані до 8м. Камера також має регульовану фокусну відстань 3,6 мм і кут огляду 75,7 градусів.

Рисунок 3.2 – Модуль камери

ІЧ-світлодіоди можна прикрутити до корпусу камери за допомогою доданих в комплекті болтів. Це підключає живлення 3,3 В до плат і фіксує їх на місці. Якщо потрібно налаштувати поріг навколишнього освітлення (щоб змінити, коли інфрачервоний світлодіод вмикається), на кожній платі світлодіода є крихітний регульований потенціометр, можна повернути його, щоб змінити поріг.

Далі необхідно обрати плату, яка буде виконувати роль центрального обчислювального модуля РП. Усі сенсори та додаткові пристрої будуть підключені до цієї плати [23].

3.1.1 Огляд одноплатних комп’ютерів для керування РП

Odroid-XU4 (рис. 3.3) – це економічно вигідний варіант з високою потужністю, з одним гігабітним портом Ethernet для швидкого підключення, HDMI 1.4a для дісплея, а плата навіть постачається з активним кулером та адаптером живлення. Він оснащений потужним восьмиядерним процесором Samsung Exynos 5422 Cortex-A15 з тактовою частотою 2 ГГц та графічним процесором Mali-T628 MP6 з підтримкою OpenGL ES 3.1 та OpenCL 1.2. ODROID-XU4 також має 2 ГБ пам'яті, два порти USB 3.0, але немає Bluetooth

для бездротового підключення. На платі можна запустити останню версію Ubuntu, Android 4.4 KitKat, 5.0 Lollipop та 7.1 Nougat. ODROID-XU4 працює на основі технології ARM big.LITTLE та рішення гетерогенної багатопроцесорної обробки (HMP). Впроваджуючи інтерфейси eMMC 5.0, USB 3.0 та Gigabit Ethernet, ODROID-XU4 може похвалитися вражаючою швидкістю передачі даних – функцією, яка все частіше потрібна для підтримки розширеної потужності обробки на ARM- пристроях. Для завантаження ОС потрібна карта MicroSD або модуль eMMC.

Рисунок 3.3 – Odroid-XU4

Так само, як ODROID-XU4, ASUS Tinker Board (рис. 3.4) можна використовувати як щоденний комп'ютер з більш ніж достатньою потужністю для базового редагування зображень, потокового передавання відео у форматі Full HD, перегляду веб-сторінок, прослуховування музики та навіть деяких ігор. На платі встановлений Rockchip RK3288, який є сучасним чотирьохядерним процесором на базі ARM, який можна знайти в багатьох планшетах та мультимедійних програвачах. Завдяки 2 ГБ пам’яті та графічному процесору Mali-T764, плата може відтворювати HD та UHD відео зі швидкістю 30 кадрів в секунду за допомогою включеного медіаплеєра з підтримкою апаратного прискорення. Плата також включає 40-контактний інтерфейс GPIO, гігабітне підключення до локальної мережі, підключення DSI MIPI для дисплеїв та сенсорних екранів, а також підключення CSI MIPI

для підключення до сумісних камер, що робить його чудовим для Інтернету речей.

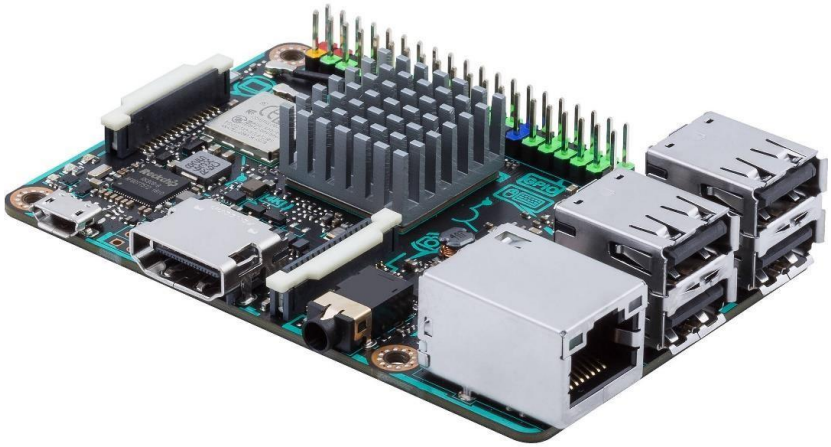

Рисунок 3.4 – ASUS Tinker Board RK3288

Якщо потрібний недорогий одноплатний комп’ютер для налаштування мережевого сховища, файлового сервера чи чогось іншого, одноплатний комп’ютер Libre Renegade (рис. 3.5) буде гарним вибором.

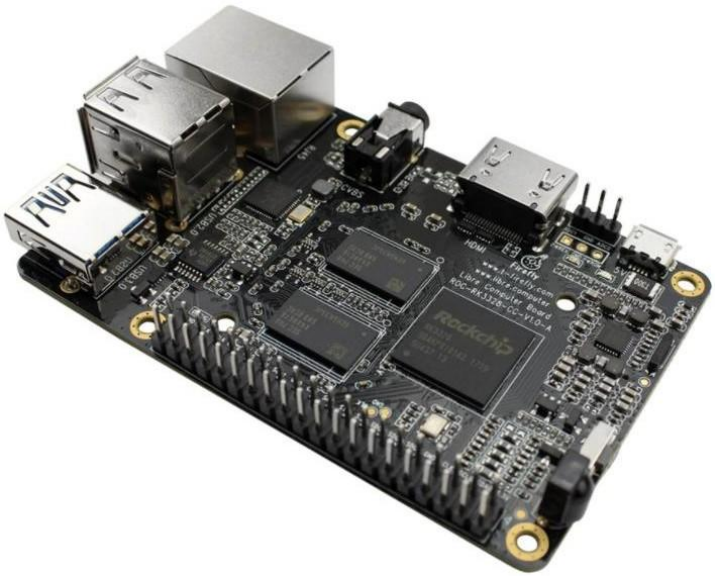

Рисунок 3.5 – Libre Renegade

Він має чотирибітний 64- розрядний процесор ARM Cortex-A53 1,4 ГГц, графічний процесор 4K Ultra HD ARM Mali-450 та 2 Гб оперативної пам'яті DDR4. Ця система насправді на 40% швидше, ніж Raspberry Pi.

Комп’ютер Libre’s Renegade навіть оснащений швидким гігабітним Ethernet, якого не зустрінеш на багатьох платах.

Плата Raspberry Pi Zero W (рис. 3.6) розширює функціональність плати Pi Zero завдяки бездротовій локальній мережі та Bluetooth. Багато в чому це ідеальний одноплатний комп’ютер для епохи Інтернету речей. Беручи до уваги його крихітний розмір, ви можете бути здивовані, дізнавшись, що він поставляється з процесором Broadcom BCM2835 з тактовою частотою 1 ГГц та 512 МБ LPDDR2 SDRAM. За надзвичайно низьку вартість ви можете отримати неймовірно універсальний Wi-Fi одноплатний комп’ютер, який стоїть на плечах найактивнішого та найкориснішого співтовариства існуючих ентузіастів одноплатних комп’ютерів. Розміри Raspberry Pi Zero W мають розмір всього 6,5 см × 3 см × 0,5 см, і він споживає настільки мало енергії, що навіть помірно великий акумулятор може тримати його в роботі дуже довго.

Рисунок 3.6 – Raspberry Pi Zero W

Pi – це комп’ютер розміром з кредитну картку з чудовими функціями [23]. Він вироблений китайською компанією Shenzhen co, Ltd. Raspberry Pi вплинув на дизайн комп'ютерного обладнання banana-Pi. Banana-Pi може працювати в багатьох операційних системах, таких як Linux, Android, Debian та Ubuntu. Banana-Pi BPI-M4 використовує чіп-структуру Realtek RTD1395, це 64- бітний зменшений одноплатний ПК A53. Він має 1 ГБ / 2 ГБ оперативної пам'яті та 8 ГБ eMMC. BPI-M4 має 4 порти USB 2.0, 1 порт USB

TYPE C, 1 порт HDMI, звуковий роз'єм f1. Є підтримка M.2 Key E інтерфейсу PCI-E 2.0.

Arduino – це програмована плата з відкритим кодом, яка може бути інтегрована в найрізноманітніші проекти (рис. 3.7). Ця плата містить мікроконтролер, який можна запрограмувати для управління різними пристроями та сприйняття великої кількості інформації. Arduino здатний взаємодіяти з великим набором вихідних пристроїв, таких як світлодіоди, двигуни та дисплеї. Завдяки своїй гнучкості та низькій вартості, Arduino став дуже популярним вибором для інженерів для створення інтерактивних апаратних проектів [23].

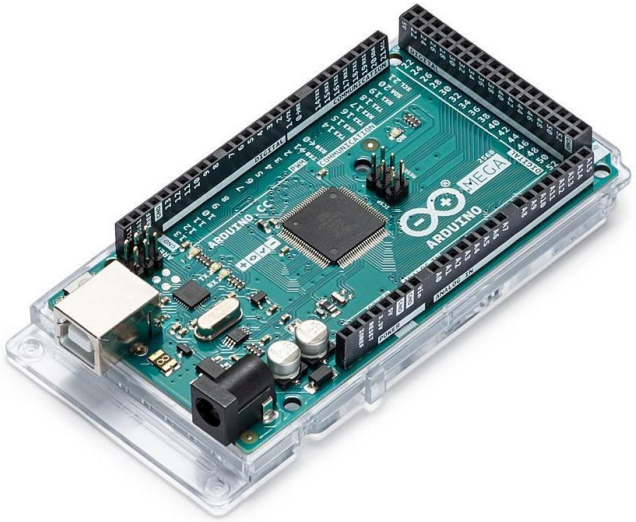

Рисунок 3.7 – Arduino Mega 2560

Мова програмування Arduino відрізняється від інших мов програмування, оскільки її легко вивчити та вона приховує всі труднощі написання програм у мікроконтролерах, надаючи пакет, який спрощує та обгортає всі важливі деталі. Деякі ключові особливості та переваги Arduino:

- зручне для користувача середовище;
- економічно вигідний;
- відкритий і розширюванний код;
- крос-платформа.

Raspberry Pi 4 Model B (рис. 3.8) оснащена найновішим високопродуктивним чотирьохядерним 64-розрядним процесором Broadcom 2711, Cortex A72 з тактовою частотою 1,5 ГГц. Плата розроблена на 20% більш енергоефективною та забезпечує на 90% більшу продуктивність, ніж її стара версія.

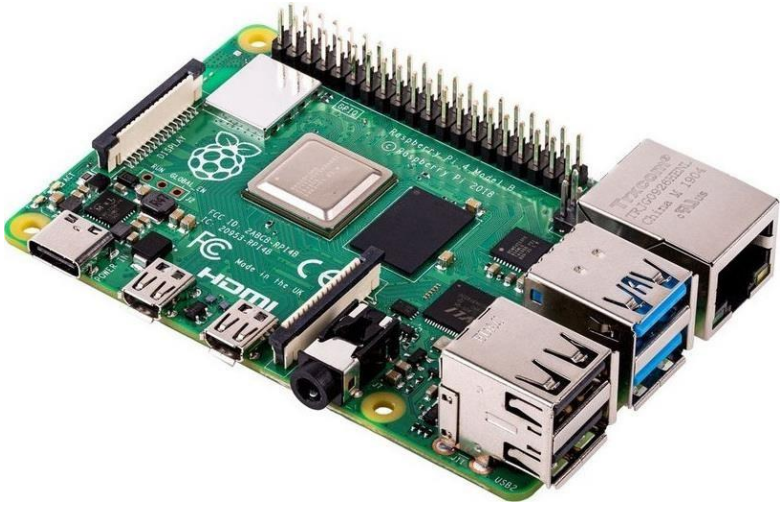

Рисунок 3.8 – Raspberry Pi 4 Model B

Оновлення апаратного забезпечення на RPi4 розроблено для швидшої продуктивності не тільки за рахунок нових варіантів SDRAM 1 ГБ / 2 ГБ та 4 ГБ SDRAM, але також за рахунок дводіапазонної 2,4 ГГц та 5 ГГц бездротові локальні мережі 802,11 b/ g / n / ac та можливості PoE через окремий PoE HAT.

Другим необхідним модулем є модуль контролю температури, вологості і тиску. Таким модулем було обрано BME280. Пристрій BME280 є цифровим барометричним датчиком тиску і є модернізованою версією BMP180. Датчик умістився на дуже малій платі, яка забезпечує доступ до датчика через інтерфейс I2C. Це дозволяє легко підключити його до Raspberry Pi і читати дані за допомогою Python. BME280 забезпечує контроль температури, тиску і вологості. BME280 виготовлений компанією Bosch, а офіційна специфікація BME280 включає всі технічні деталі. Їх пристрій може пропонувати як інтерфейси SPI, так і I2C, тому вам потрібно переконатися, що ваш модуль використовує бажаний інтерфейс [23].

Шина I2C (Inter-IC) – це двонаправлена двопровідна послідовна шина, яка забезпечує комунікаційний зв'язок між інтегральними схемами (IC). Філіпс представив шину I2C 20 років тому для серійних виробів, таких як телевізори, відеомагнітофони та аудіообладнання. Сьогодні I2C є фактичним рішенням для вбудованих додатків.

Існує три швидкості передачі даних для шини I2C: стандартний, швидкий режим і високошвидкісний режим. Стандартний – 100 Кбіт/с. Швидкий режим – 400 Кбіт/с, а високошвидкісний режим підтримує швидкість до 3,4 Мбіт / с.

Всі вони сумісні один з одним. Шина I2C підтримує 7-розрядні та 10-розрядні пристрої адресного простору та пристрої, які працюють під різною напругою.

3.2 Вибір платформи, ескізування та створення 3D-моделі РП

Для ґрунтової поверхні оптимально використовувати шестиколісну платформу з чотирма приводами, розташованими симетрично (рисунок 3.9).

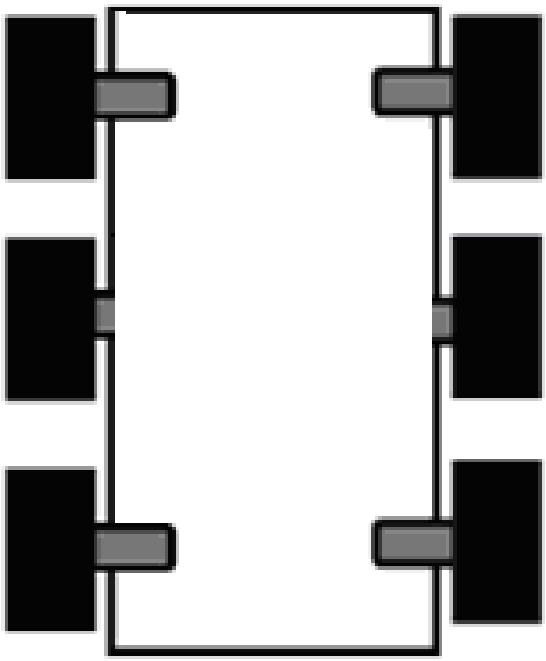

Рисунок 3.9 – Ескіз платформи РП

Розрахунки по орієнтованій масі готової РП наведено в табл. 3.1.

Кермо планується реалізувати перемикання напрямків обертання чотирьох двигунів.

Таблиця 3.1 – Орієнтована маса РП

Параметр	Значення, кг
Орієнтована маса приводів	4 х (0,25–0,4) = 1,0–1,6
Орієнтована маса акумуляторів	2 х (0,3–0,4) = 0,6–0,8
Маса коліс	6 х (0,25–0,3) = 1,5–1,8
Маса електроніки и радіаторів	0,3–0,4
Маса корпуса и шасі	1–1,3
Загалом	4,4–5,9

На рисунку 3.10 представлений ескіз РП (вид зверху та спереду). За даними ескізами можна оцінити габаритні розміри робота.

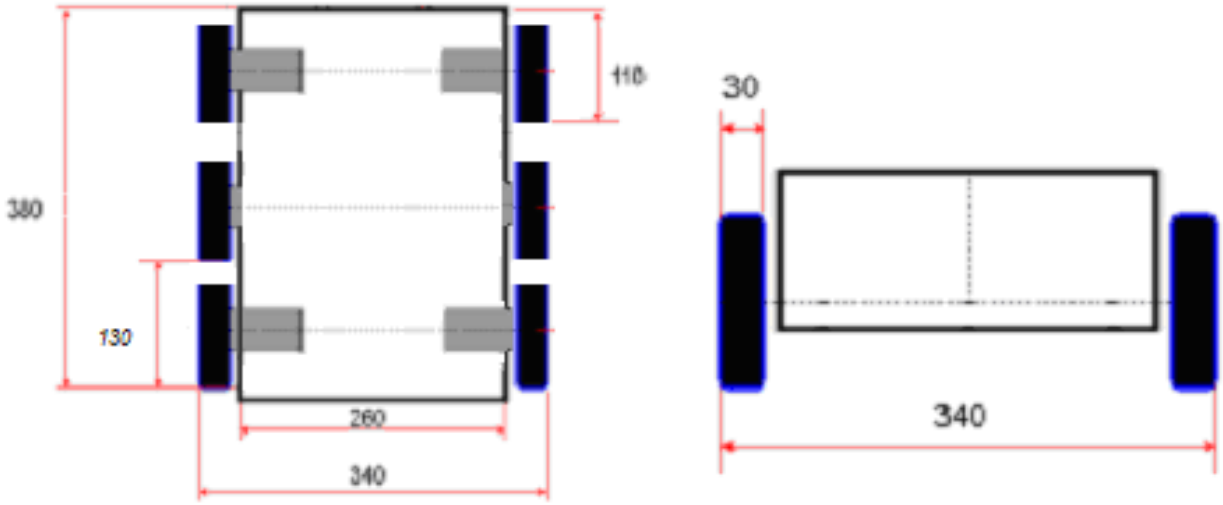

Рисунок 3.10 – Ескіз РП з габаритними розмірами

3.2.1 Розробка 3D-моделі

Для розробки 3D-моделі прототипу було обрано програму Fusion 360. Fusion 360 – це хмарна платформа САПР [31, 32], яка є доступною,

високопродуктивною альтернативою іншим основним гравцям галузі. Цей додаток простий у використанні та має всі загальні функції, яких можна очікувати від популярних пакетів САПР. Fusion 360 був побудований з нуля, щоб бути всеохоплюючим рішенням для розробки продуктів і може запропонувати простий робочий процес, починаючи від концептуального дизайну і закінчуючи виробництвом.

Fusion 360 має дуже велику базу знань, яка ретельно охоплює всі особливості програмного забезпечення; ці навчальні посібники можна отримати через Fusion 360, а також через веб-сайт Autodesk.

Fusion 360 може виконувати ресурсоємні операції в хмарі, включаючи візуалізацію, моделювання, оптимізацію фігури та генеративний дизайн. Це означає, що робота може тривати, поки весь важкий підйом робиться на хмарі.

У Fusion 360 можна перемикатися між шістьма різними робочими областями. Кожна робоча область має власний набір інструментів та функцій:

– Design: Для створення 3D-моделей та поверхонь за допомогою ескізів, видавлень, обертів та багатьох інших стандартних інструментів САПР;

– Render: Створення фотореалістичних візуалізацій компонентів та продуктів;

– Anumation: Анімація збірки, щоб побачити, чи функціонують вони належним чином, або показати функціональність потенційним клієнтам.

– Simulation: автоматизована інженерія для проведення різних аналізів напружень на конструкціях, щоб переконатися, що вони здатні відповідати умовам експлуатації;

– Manufacture: автоматизоване виробництво (CAM) для надання допомоги у виготовленні деталі на різних цифрових інструментах виготовлення, таких як фрези з ЧПУ, токарні верстати з ЧПУ, лазерні різаки та гідроабразивні різаки;

– Drawing: Створіть креслення конструкцій для виготовлення на ЧПУ.

Створення 3D-моделі складається з кількох простих етапів. Перший – створення ескізу (рис. 3.11).

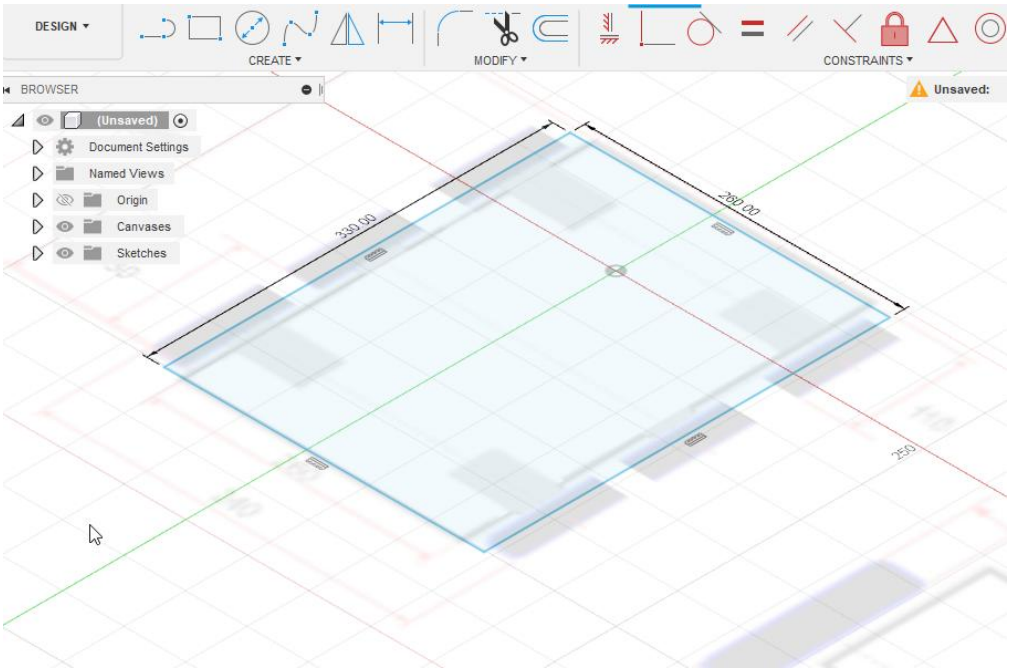

Рисунок 3.11 – Створення ескізу у Fusion 360

Ескізи представляють собою лінії та криві на 2D-площині. Вигляд переключиться з тривимірної на 2D-перспективу.

Другий етап – видавлювання (рис. 3.12).

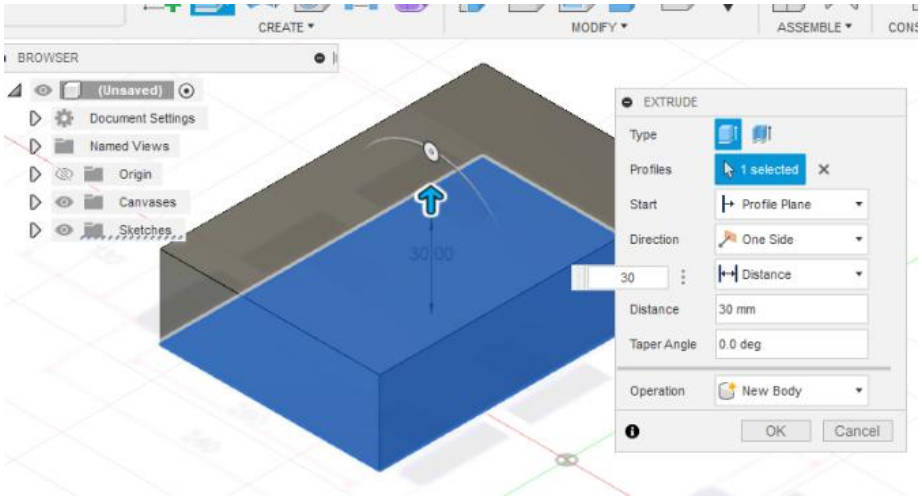

Рисунок 3.12 – Операція видавлювання у Fusion 360

Після цього достатньо застосувати деякі косметичні зміни у об’єкті на експортувати модель (рис. 3.13). Після цього її можна передати у програму для перетворення моделі на GCODE-файл. Він використовується 3D-принтером при друці.

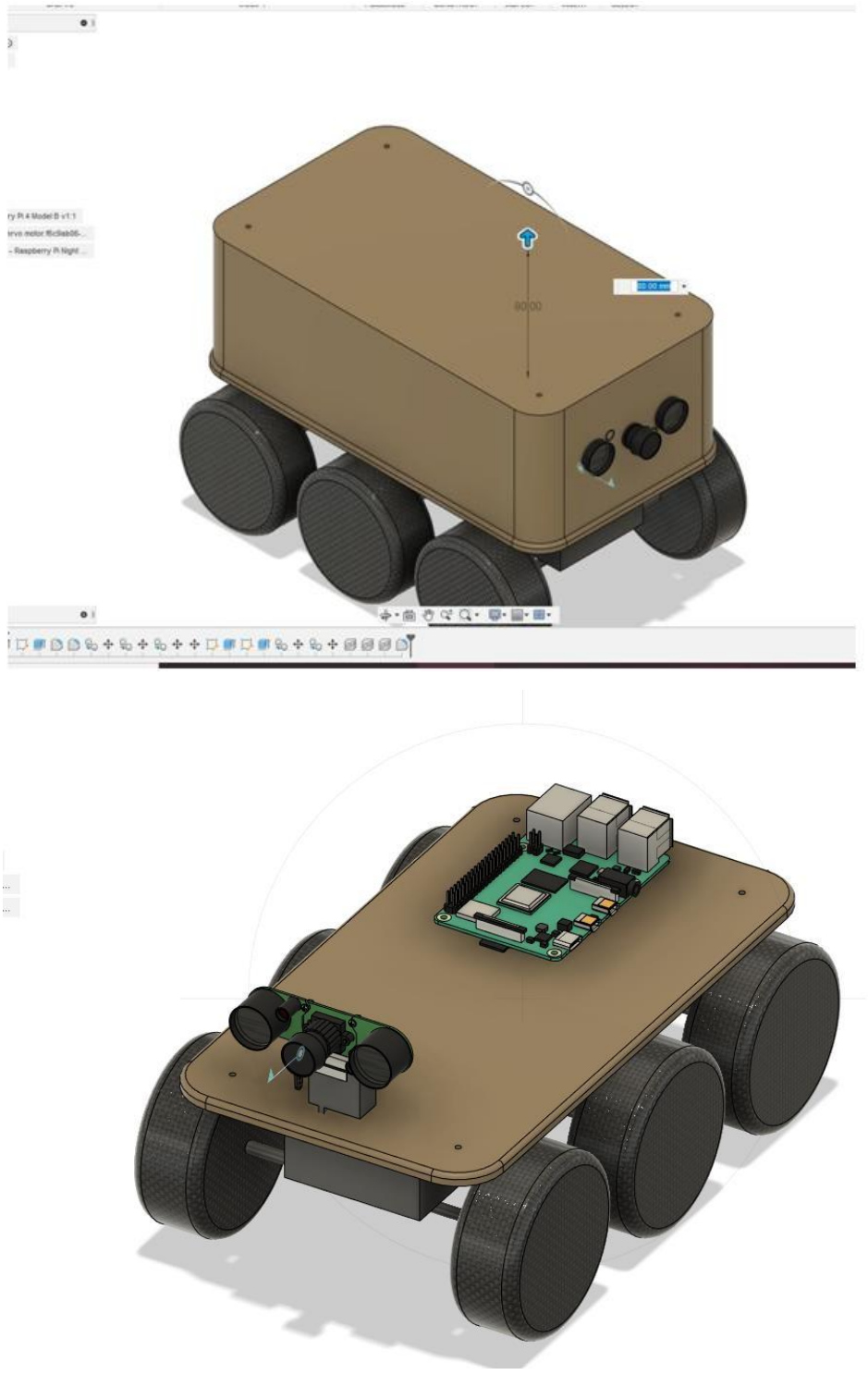

Рисунок 3.13 – 3D-модель РП у Fusion 360

3.3 Програмне забезпечення компонентів рухомої платформи

Для використання BME280 потрібно ввімкнути інтерфейс I2C на Raspberry Pi, оскільки він не ввімкнений за замовчуванням. Спочатку треба запустити sudo raspi-config і дотримуватися вказівок, щоб встановити підтримку i2c для ядра ARM та ядра Linux (рис. 3.14).

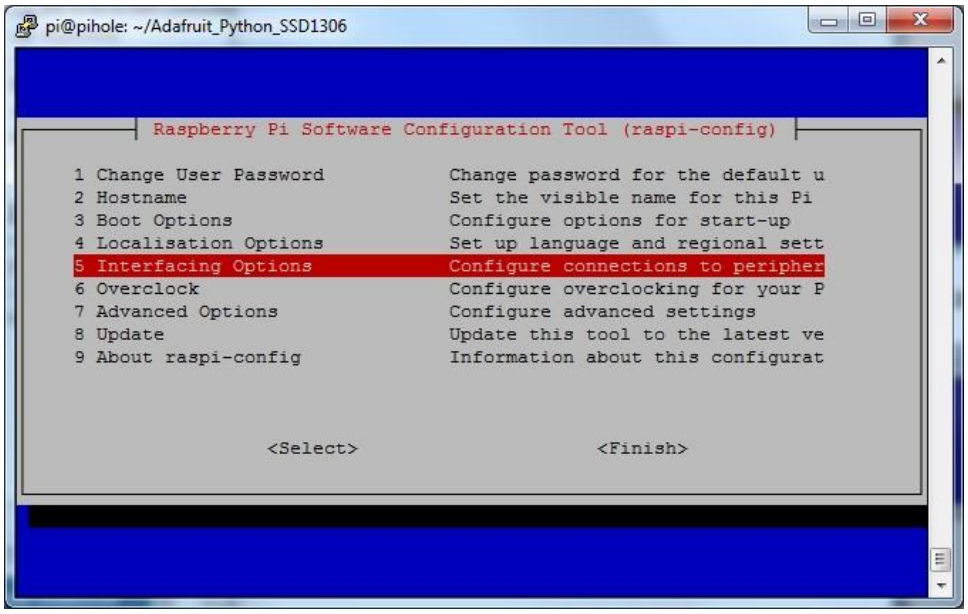

Рисунок 3.14 – Перехід до пункту «Interfacing options»

Для того, щоб протестувати роботу I2C – можна ввести команду sudo i2cdetect -y 1. На рис. 3.15. видно, що використовуються дві адреси I2C – bx30 та 8x60.

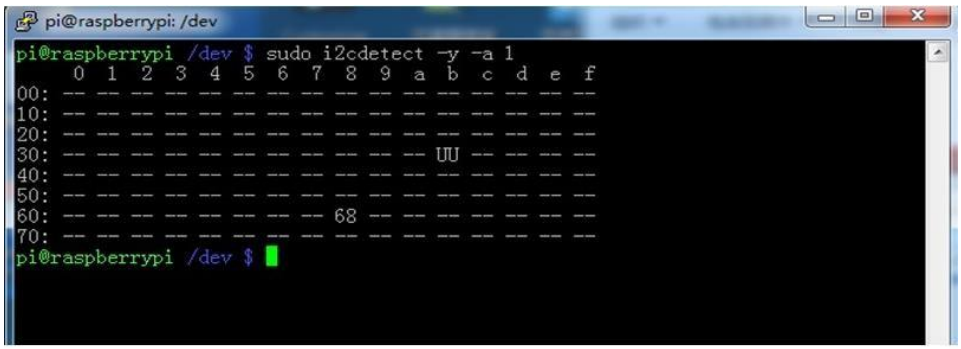

Рисунок 3.15 – Тестування роботи I2C

Ці значення будуть різними залежно від того, що в даний час приєднано до роз'ємів I2C RPi.

Інтерфейс I2C досить простий (рис. 3.16). Він має лише чотири дроти: SCL (послідовний годинник), SDA (послідовні дані), GND (земля) та VIN (вхід напруги). Адреса I2C датчика зазвичай визначається виробником. Плати одного типу, виготовлені різними виробниками, зазвичай мають однакову адресу для забезпечення сумісності.

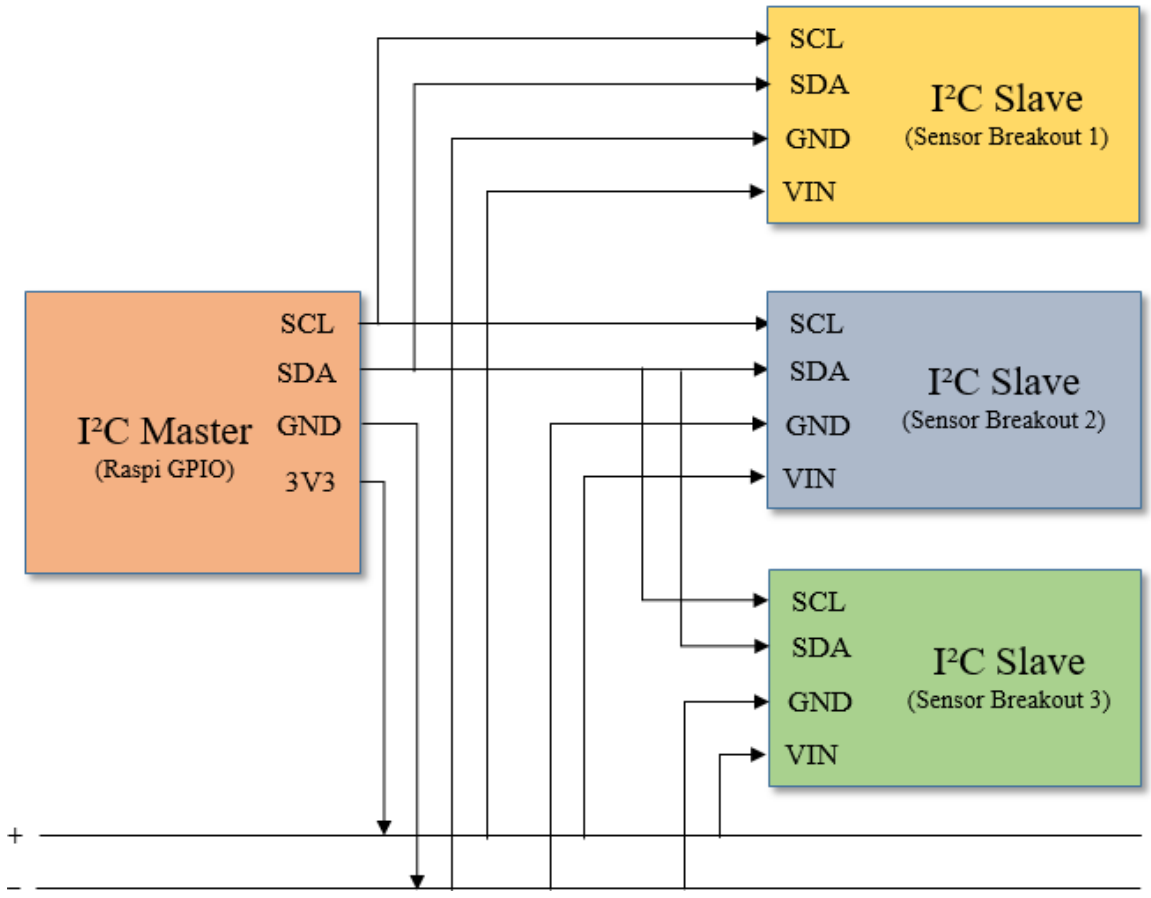

Рисунок 3.16 – Схема шини I2C, яка з’єднує декілька сенсорів

Багато з сенсорів мають драйвери з відкритим кодом, написані на Python та Java, доступні в Raspberry Pi. Тож ви можете легко розпочати роботу.

Єдина незручність цих плат полягає в тому, що користувачеві доводиться самостійно займатися пайкою (дуже мало плат постачаються з вже запаяними коннекторами). Але як тільки користувач пройде цей крок,

все повинно бути відносно легким. Давайте подивимося, що деякі з цих проривів датчиків можуть зробити для нас.

Розглянемо схему підключення BME280 до RasbperryPi (3.17).

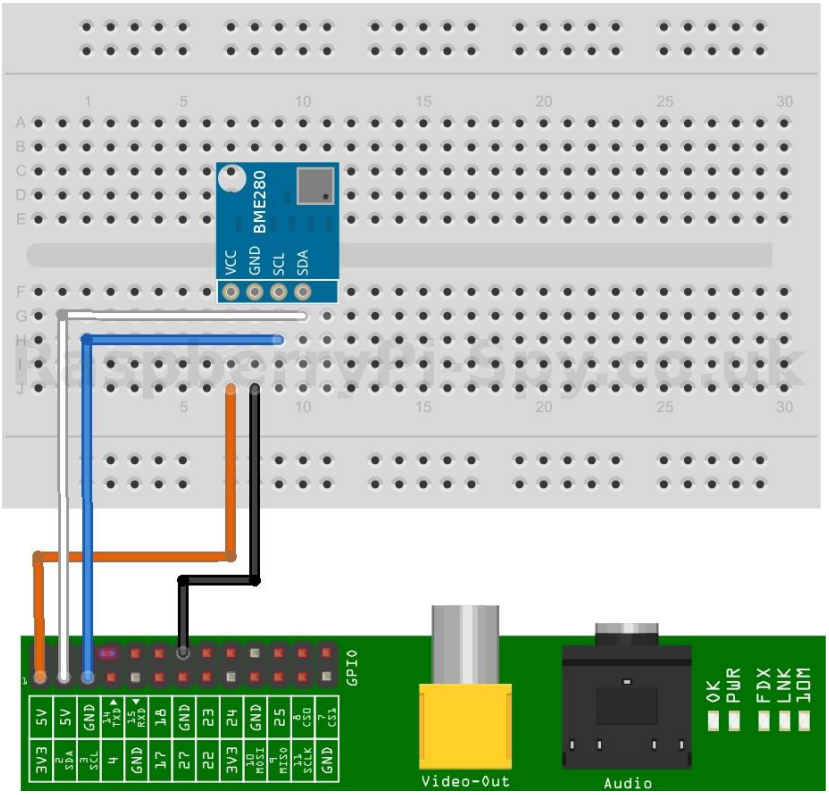

Рисунок 3.17 – Схема підключення BME280 до Rasbperry Pi

В якості сенсора світла застосуємо звичайний фоторезистор (рис. 3.18).

Рисунок 3.18 – Фоторезистор

Світлозалежний резистор або також відомий як датчик LDR у світлій кімнаті матиме опір лише кілька сотень Ом, а в темряві може мати опір у кілька мегаомів.

Для того, щоб виміряти опір фоторезистора ми застосуємо звичайний конденсатор (рис 3.19). Конденсатор по суті діє як акумулятор, заряджаючись під час отримання енергії, а потім розряджаючись, коли більше не отримує живлення. Використовуючи його послідовно з фоторезистором, ми можемо визначити, скільки опору надає фоторезистор, таким чином маючи змогу визначити, наскільки світло чи темно в приміщені.

Рисунок 3.19 – Конденсатор на 1мкФ

Щоб правильно під’єднати фоторезистор до RaspberryPi при прототипуванні, треба виконати наведені нижче дії (рис. 3.20):

– підключити GPIO №1 (3v3) до позитивної шини (+) на макетній дошці;

– підключити GPIO №6 (земля) до шини заземлення (-) на макетній дошці;

– помістити фоторезистор на плату і під’єднати провід від одного з контактів до позитивного контакту конденсатора;

– до іншого контакта фоторезистора підключити дріт, що веде назад до Raspberry Pi, до GPIO №7;

– під’єднати негативний контакт конденсатора до шини заземлення на макетній дошці.

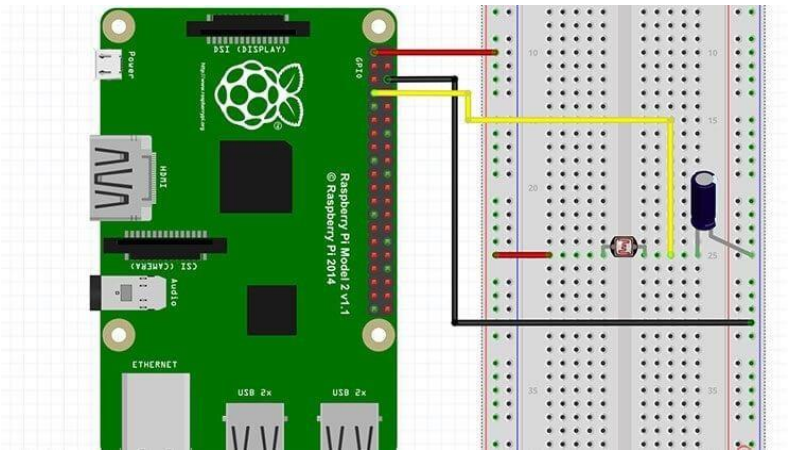

Рисунок 3.20 – Схема під’єднання фоторезистора до Raspberry Pi

Найбільша проблема, полягає в тому, що RPi не має жодного аналогового контакту. Усі вони цифрові, тому ми не можемо точно виміряти опір на вході.

Натомість ми виміряємо час, який потрібно конденсатору, щоб зарядитися і надіслати на контакт HIGH сигнал. Цей метод – простий, але неточний спосіб визначити освітленість.

Наведемо лістинг коду для зчитування інформації з датчика освітленості на Raspberry Pi. Для початку ми імпортуємо пакет GPIO, який нам знадобиться, щоб ми могли зв’язатися із контактами GPIO.

Також імпортуємо пакет time (рис. 3.21), що ми могли перевести сценарій у режим сну, коли нам потрібно.

```
#!/usr/local/bin/python

import RPi.GPIO as GPIO
import time
```

Рисунок 3.31 – Імпорт пакету time

Потім ми встановлюємо режим GPIO (рис. 3.32) на GPIO.BOARD, і це означає, що вся нумерація, яку ми використовуємо в цьому сценарії, буде стосуватися фізичної нумерації контактів.

Оскільки ми використовуємо лише один контакт на вхід/вихід, нам потрібно оголосити лише одну змінну. Встановлюємо для цієї змінної номер контакту, який використовуємо як вхідний / вихідний.

```
GPIO.setmode(GPIO.BOARD)

#define the pin that goes to the circuit
pin_to_circuit = 7
```

Рисунок 3.32 – Задання режиму роботи GPIO

Далі створюємо функцію, яка називається rc_time, і яка вимагає одного параметра, який є номером контакту на платі. У цій функції ми ініціалізуємо змінну, яка називається count, і повернемо значення цієї змінної, коли контакт перейде у HIGH.

Потім ми встановлюємо наш контакт, як вихідний і встановлюємо його значення LOW. Далі виконується сценарій сну на 10мс.

Після цього встановлюємо контакт, як вхідний і вводимо цикл while. Залишаємось у цьому циклі поки контакт не перейде у значення HIGH (коли конденсатор заряджається приблизно до 3/4).

Як тільки контакт переходить у значення HIGH, повертаємо значення підрахунку до основної функції (рис. 3.33).

```
def rc_time (pin_to_circuit):
    count = 0

    GPIO.setup(pin_to_circuit, GPIO.OUT)
    GPIO.output(pin_to_circuit, GPIO.LOW)
    time.sleep(0.1)

    GPIO.setup(pin_to_circuit, GPIO.IN)

    while (GPIO.input(pin_to_circuit) == GPIO.LOW):
        count += 1

    return count

try:
    while True:
        print(rc_time(pin_to_circuit))
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
```

Рисунок 3.33 – Визначення часу переходу стану конденсатора

Системою контролю двигунів виступатиме Adafruit Motor Shield V2 (рис. 3.34).

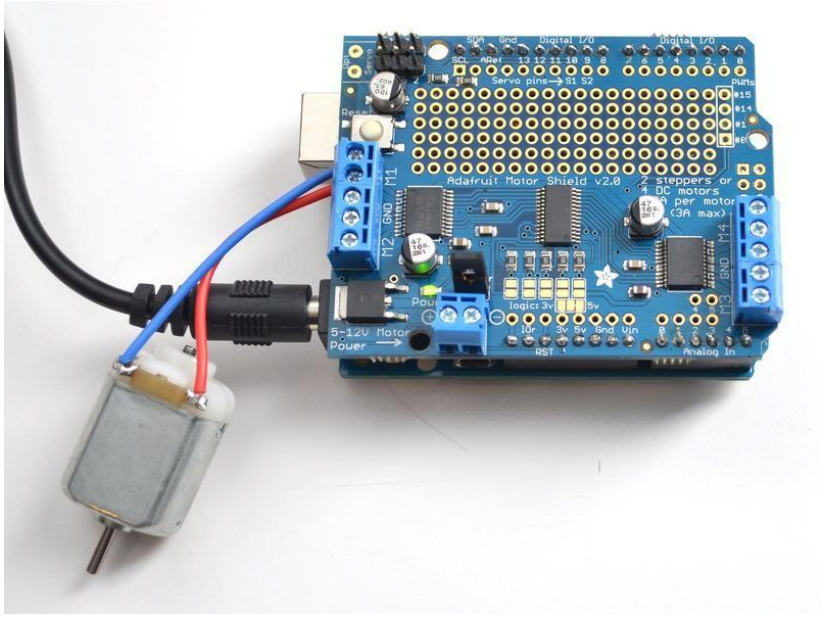

Рисунок 3.34 – Adafruit Motor Shield V2

3.4 Реалізація алгоритму роботи рухомої платформи

Спочатку робот буде шукати прохідний шлях, використовуючи інформацію про глибину сцени з камери. Робот підходить до всіх виявлених пунктів огляду та підписується електронним способом, після чого завантажується номер пункту огляду та час входу. Коли всі пункти огляду пройдено, це конкретне завдання патрулювання буде виконано. На рис. 3.35 представлена блок-схема, що демонструє процес автономного патрулювання.

На сервері використовується два веб-сервіси (сервіс інформації у реальному часі та сервіс завантаження/відображення збережених даних) для віддаленого резервного копіювання даних та моніторингу. Робот створює резервну копію інформації під час проходження контрольних точок та при виявленні виключних ситуацій. Центральний модуль RPi підключається до бездротової точки доступу за допомогою модуля Wi-Fi.

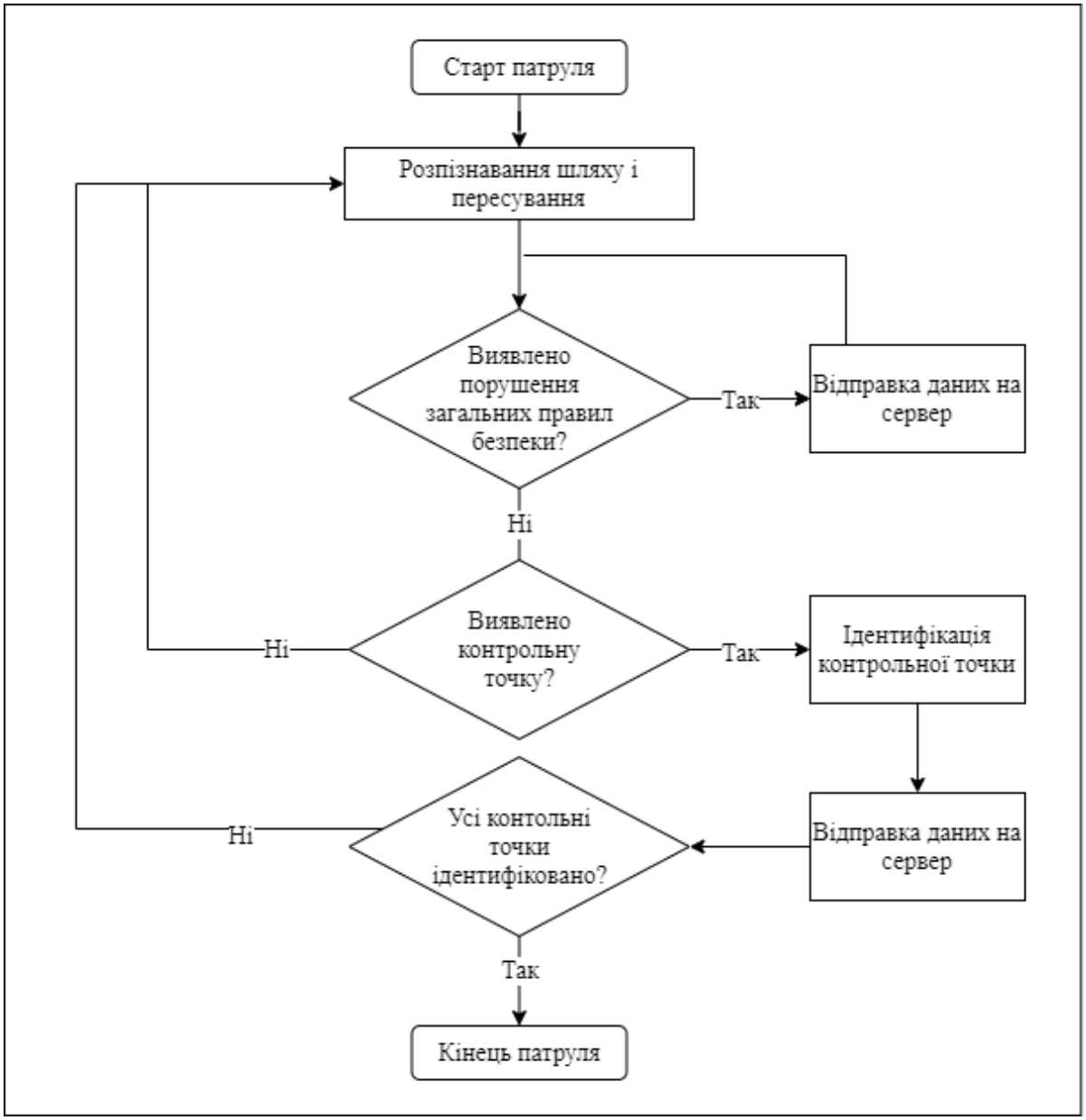

Рисунок 3.35 – Алгоритм патрулювання

Уся інформація про стан контрольних точок передається у вигляді JSON-об’єкта, в якому знаходиться інформація про температуру, вологість повітря, тиск, освітлення і шум, а також фото контрольної точки, закодоване у форматі base64. Також робот на постійній основі передає зображення з камери на WebSocket-сервер. Користувач може зайти на веб-платформу та переглянути зображення та інформацію з робота у реальному часі, або дані за певний період часу.

Насамперед, необхідно здійснити підключення моторів до драйвера моторів. Для цього потрібно:

– підключити два дроти від першого та другого двигуна до клем M1 та M2 на платі;

– підключити позитивну і негативну клему живлення до позитивного і негативного терміналу плати.

Потім необхідно встановити останню версію Adafruit CurcuitPython. Потім для керування двигунами застосовуємо цей код на рис. 3.36.

```
import time
import board
from adafruit_motorkit import MotorKit

kit = MotorKit(i2c=board.I2C())

kit.motor1.throttle = 1.0
time.sleep(0.5)
kit.motor1.throttle = 0
```

Рисунок 3.36 – Код для керування двигунами

Для отримання і збереження інформації з датчика BME280 будемо використовувати Azure IoT Hub. Спочатку створимо новий IoT Hub (рис. 3.37). Для цього необхідно мати аккаунт Microsoft Azure.

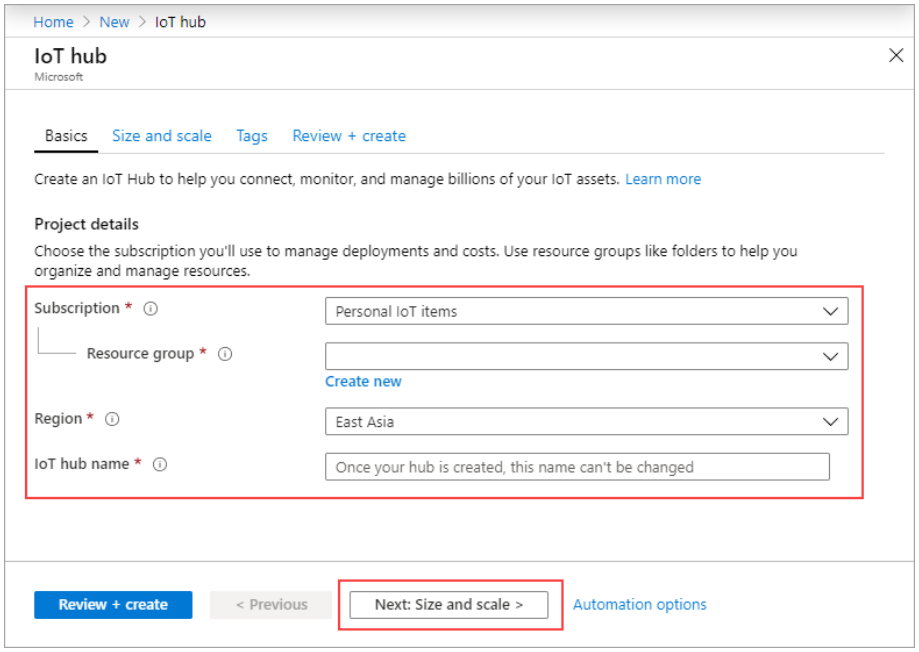

Рисунок 3.37 – Створення Microsoft Azure IoT Hub

Далі необхідно додати новий пристрій до IoT Hub. Для цього треба:

– у меню навігації IoT Hub відкрити IoT Devices, а потім виберати New, щоб додати пристрій у IoT Hub;

– у розділі «Створення пристрою» ввести ім’я нового пристрою, наприклад myDeviceId, і вибрати «Зберегти». Ця дія створює ідентифікацію пристрою для IoT Hub;

– після створення пристрою відкрити його зі списку на панелі пристроїв IoT. Скопіювати рядок основного підключення, щоб використовувати пізніше.

Для зчитування даних з BME280 застосуємо код на рис. 3.38 – 3.40.

```
smbus
import time</p><p># Get I2C bus
bus = smbus.SMBus(1)</p><p># BME280 address, 0x76(118)
# Read data back from 0x88(136), 24 bytes
b1 = bus.read_i2c_block_data(0x76, 0x88, 24)</p><p># Convert the data
```

Рисунок 3.38 – Код для зчитування даних з BME280

```
# Temp coefficients
dig_T1 = b1[1] * 256 + b1[0]
dig_T2 = b1[3] * 256 + b1[2]
if dig_T2 > 32767 :
    dig_T2 -= 65536
dig_T3 = b1[5] * 256 + b1[4]
if dig_T3 > 32767 :
    dig_T3 -= 65536</p><p># Pressure coefficients
dig_P1 = b1[7] * 256 + b1[6]
dig_P2 = b1[9] * 256 + b1[8]
if dig_P2 > 32767 :
    dig_P2 -= 65536
dig_P3 = b1[11] * 256 + b1[10]
if dig_P3 > 32767 :
    dig_P3 -= 65536
dig_P4 = b1[13] * 256 + b1[12]
if dig_P4 > 32767 :
    dig_P4 -= 65536
dig_P5 = b1[15] * 256 + b1[14]
if dig_P5 > 32767 :
    dig_P5 -= 65536
dig_P6 = b1[17] * 256 + b1[16]
if dig_P6 > 32767 :
    dig_P6 -= 65536
dig_P7 = b1[19] * 256 + b1[18]
if dig_P7 > 32767 :
    dig_P7 -= 65536
dig_P8 = b1[21] * 256 + b1[20]
if dig_P8 > 32767 :
    dig_P8 -= 65536
dig_P9 = b1[23] * 256 + b1[22]
if dig_P9 > 32767 :
    dig_P9 -= 65536</p><p># BME280 address, 0x76(118)
# Read data back from 0xA1(161), 1 byte
dig_H1 = bus.read_byte_data(0x76, 0xA1)</p><p># BME280 address, 0x76(118)
# Read data back from 0xE1(225), 7 bytes
b1 = bus.read_i2c_block_data(0x76, 0xE1, 7)</p><p># Convert the data
# Humidity coefficients
dig_H2 = b1[1] * 256 + b1[0]
if dig_H2 > 32767 :
    dig_H2 -= 65536
dig_H3 = (b1[2] &  0xFF)
dig_H4 = (b1[3] * 16) + (b1[4] & 0xF)
if dig_H4 > 32767 :
    dig_H4 -= 65536
dig_H5 = (b1[4] / 16) + (b1[5] * 16)
if dig_H5 > 32767 :
    dig_H5 -= 65536
dig_H6 = b1[6]
if dig_H6 > 127 :
    dig_H6 -= 256</p><p># BME280 address, 0x76(118)
```

Рисунок 3.39 – Код для зчитування даних з BME280

```
# Select control humidity register, 0xF2(242)
#       0x01(01)    Humidity Oversampling = 1
bus.write_byte_data(0x76, 0xF2, 0x01)
# BME280 address, 0x76(118)
# Select Control measurement register, 0xF4(244)
#       0x27(39)    Pressure and Temperature Oversampling rate = 1
#                   Normal mode
bus.write_byte_data(0x76, 0xF4, 0x27)
# BME280 address, 0x76(118)
# Select Configuration register, 0xF5(245)
#       0xA0(00)    Stand_by time = 1000 ms
bus.write_byte_data(0x76, 0xF5, 0xA0)</p><p>time.sleep(0.5)</p><p>
# Read data back from 0xF7(247), 8 bytes
# Pressure MSB, Pressure LSB, Pressure xLSB, Temperature MSB, Temperature LSB
# Temperature xLSB, Humidity MSB, Humidity LSB
data = bus.read_i2c_block_data(0x76, 0xF7, 8)</p><p>
adc_p = ((data[0] * 65536) + (data[1] * 256) + (data[2] & 0xF0)) / 16
adc_t = ((data[3] * 65536) + (data[4] * 256) + (data[5] & 0xF0)) / 16</p><p>
adc_h = data[6] * 256 + data[7]</p><p># Temperature offset calculations
var1 = ((adc_t) / 16384.0 - (dig_T1) / 1024.0) * (dig_T2)
var2 = (((adc_t) / 131072.0 - (dig_T1) / 8192.0) * ((adc_t)/131072.0
    - (dig_T1)/8192.0)) * (dig_T3)
t_fine = (var1 + var2)
cTemp = (var1 + var2) / 5120.0
fTemp = cTemp * 1.8 + 32</p><p># Pressure offset calculations
var1 = (t_fine / 2.0) - 64000.0
var2 = var1 * var1 * (dig_P6) / 32768.0
var2 = var2 + var1 * (dig_P5) * 2.0
var2 = (var2 / 4.0) + ((dig_P4) * 65536.0)
var1 = ((dig_P3) * var1 * var1 / 524288.0 + ( dig_P2) * var1) / 524288.0
var1 = (1.0 + var1 / 32768.0) * (dig_P1)
p = 1048576.0 - adc_p
p = (p - (var2 / 4096.0)) * 6250.0 / var1
var1 = (dig_P9) * p * p / 2147483648.0
var2 = p * (dig_P8) / 32768.0
pressure = (p + (var1 + var2 + (dig_P7)) / 16.0) / 100</p><p>
var_H = ((t_fine) - 76800.0)
var_H = (adc_h - (dig_H4 * 64.0 + dig_H5 / 16384.0 * var_H)) * (dig_H2 / 65536.0
    * (1.0 + dig_H6 / 67108864.0 * var_H * (1.0 + dig_H3 / 67108864.0 * var_H)))
humidity = var_H * (1.0 -  dig_H1 * var_H / 524288.0)
if humidity > 100.0 :
    humidity = 100.0
elif humidity < 0.0 :
    humidity = 0.0</p><p># Output data to screen
print "Temperature in Celsius : %.2f C" %cTemp
print "Temperature in Fahrenheit : %.2f F" %fTemp
print "Pressure : %.2f hPa " %pressure
print "Relative Humidity : %.2f %%" %humidity</p>
```

Рисунок 3.40 – Код для зчитування даних з BME280

Як платформу для руху системи було обрано платформу Wild Thumper (рис. 3.41). Кожен із шести моторів, прикріплених до Wild Thumper, управляє одним колесом. Ці мотори мають такі властивості при напрузі 7 В: 130 обертів на хвилину, 450 мА струм без навантаження і 0,96 Н м крутного моменту.

Рисунок 3.41 – Мотор Wild Thumper

Два біти, або цифрові сигнали, використовуються для визначення напрямку роботи двигуна. У табл. 3.2 пояснюється, як можливі комбінації цих бітів визначатимуть напрямок обертання двигуна.

Таблиця 3.2 – Комбінації упрявляючих бітів на мотор

Біт 1	Біт 2	Напрямок обертання
0	0	Не обертається
0	1	Назад
1	0	Вперед
1	1	Не обертається

В залежності від положення двигуна користувач може вибрати, в якому напрямку будуть відбуватися рухи "вперед" і "назад". Логічне значення цих бітів, наприклад LOW або HIGH, визначає напрямок обертання двигунів; а отже і колес, прикріплених до них. Потужність джерела напруги визначає швидкість обертання двигунів. Під час використання картографічного робота висока швидкість не потрібна. Більш низькі напруги призводять до лінійно менших швидкостей обертання в хвилину. Найнижчий темп, необхідний для управління двигунами, становить приблизно 60 об / хв.

3.5 Програмне забезпечення веб-платформи

Програмне забезпечення платформи буде базуватися на технологіях NodeJS + Express та React. Node – це середовище JavaScript, побудоване на тому самому механізмі JavaScript, який використовується у веб-браузері Google Chrome. Він має кілька чудових функцій, які роблять його привабливим вибором для створення додатків середнього рівня на стороні сервера, включаючи веб-сервери та веб-служби для API платформ.

Модель вводу-виводу, керована подіями, що не блокуються, надає їй дуже привабливу продуктивність, легко перевершуючи потокові серверні

середовища, такі як PHP та Ruby on Rails, які блокують введення / виведення та обробляють одночасно запити декількох користувачів, використовуючи окремі потоки для кожного.

Деякі особливості NodeJS:

– швидка робота (Неблокуючий ввід / вивід за замовчуванням);

– події;

– швидкий обмін данними;

– добре спроектований потоковий API;

– зручні стандартні бібліотеки для взаємодії з ОС, файловою системою тощо;

– підтримка скомпільованих двійкових модулів на випадок, коли вам потрібно розширити можливості Node за допомогою мови нижчого рівня, такої як C ++;

– NodeJS довіряють і підтримують великі підприємства, що використовують критично важливі програми. (Adobe, Google, Microsoft, Netflix, PayPal, Uber, Walmart тощо.);

– низький поріг входу.

Для того, щоб запустити найпростіший сервер на NodeJS достатньо коду на рис. 3.42.

```
const express = require('express');

const app = express();
const port = process.env.PORT || 3000;

app.get('/', (req, res) => {
  res.send('\n\nHello, world!\n\n');
});

app.listen(port, () => {
  console.log(`listening on port ${ port }`);
});
```

Рисунок 3.42 – Код для запуску сервера NodeJS

Express.js – це фреймворк веб-додатків, який побудований поверх Node.js. Він забезпечує мінімальний інтерфейс з усіма необхідними

інструментами, необхідними для створення веб-додатків. Express.js покращує гнучкість додатку за допомогою великого набору модулів, доступних на npm, які можна безпосередньо підключити до Expres. Це допомагає легко керувати потоком даних між сервером і маршрутами (routes) в серверних додатках. Він був розроблений TJ Holowaychuk і випущений на ринок 22 травня 2010 року. Раніше ним керував IBM, але в даний час він знаходиться під керівництвом інкубатора Node.js Foundation.

Express в основному відповідає за обробку внутрішньої частини в стеку MEAN. MEAN Stack – це програмний стек з відкритим кодом JavaScript, який активно використовується для створення динамічних веб-сайтів та веб-додатків на ринку. Тут MEAN розшифровується як MongoDB, Express.js, та Node.js.

Для роботи із користувачем нам спочатку необхідно описати MongoDB модель. На рис. 3.43 наведено код для найпростішої моделі користувача:

```
const mongoose = require('mongoose');
const UserSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});
module.exports = mongoose.model('User', UserSchema);
```

Рисунок 3.43 – Найпростіша модель користувача

Цей код дозволяє створювати об'єкти користувачів, які мають власні унікальні поля електронної пошти та пароля, які потім можна зберегти та отримати з MongoDB для аутентифікації користувачів.

Паролі не можна зберігати в простому тексті, тому для захисту паролів використовується невелика бібліотека bcrypt, це дозволить хешувати паролі (рис. 3.44).

Після встановлення bcrypt можна додати в схему користувача hook для хешування паролів, перш ніж зберігати їх у базі даних:

```
const mongoose = require('mongoose');
const bcrypt = require('bcrypt');
const saltRounds = 10;
const UserSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});
UserSchema.pre('save', function(next) {
  if (this.isNew || this.isModified('password')) {
    const document = this;
    bcrypt.hash(document.password, saltRounds,
      function(err, hashedPassword) {
      if (err) {
        next(err);
      }
      else {
        document.password = hashedPassword;
        next();
      }
    });
  } else {
    next();
  }
});
module.exports = mongoose.model('User', UserSchema);
```

Рисунок 3.44 – Хешування паролів

Тепер, коли об’єкт User створено, можна перевірити його та створити деяких користувачів (рис. 3.45). Для цього створюємо ще один Express-маршрут так:

```
app.post('/api/register', function(req, res) {
  const { email, password } = req.body;
  const user = new User({ email, password });
  user.save(function(err) {
    if (err) {
      res.status(500)
        .send("Error registering new user please try again.");
    } else {
      res.status(200).send("Welcome to the club!");
    }
  });
});
```

Рисунок 3.45 – Маршрут регістрації користувачів

Тепер можна перевірити цей ендпоїнт за допомогою URL запиту (рис. 3.46):

```
curl -X POST \
  http://localhost:3000/api/register \
  -H 'Content-Type: application/json' \
  -d '{
 "email": "me@example.com",
 "password": "mypassword"
}'
```

Рисунок 3.46 – Перевірка ендпоїнта

Тепер, коли декілька користувачів збережено у базі даних, необхідно створити спосіб їх аутентифікації використовуючи базу даних.

Для додаємо метод до схеми користувача, який прийме пароль у вигляді рядка, і за допомогою bcrypt повідомить, чи це правильний пароль для цього користувача (рис. 3.47):

```
UserSchema.methods.isCorrectPassword = function(password, callback){
  bcrypt.compare(password, this.password, function(err, same) {
    if (err) {
      callback(err);
    } else {
      callback(err, same);
    }
  });
}
```

Рисунок 3.47 – Метод перевірки правильності пароля

Тепер, можна почати видавати токени клієнтам. Спочатку треба створити секретний рядок, який слід використовувати під час підписання токенів.

Далі потрібно встановити бібліотеку jsonwebtoken, яка дозволить видавати та перевіряти веб-маркери JSON.

Нарешті, можна створити новий Express-маршрут, який, отримавши електронну адресу та пароль, знайде користувача із вказаною адресою електронної пошти та переконається, що вказаний пароль правильний (рис. 3.48). Якщо пароль правильний, ми видамо запитуваний токен користувачу:

```
const jwt = require('jsonwebtoken');
app.post('/api/authenticate', function(req, res) {
  const { email, password } = req.body;
  User.findOne({ email }, function(err, user) {
    if (err) {
      console.error(err);
      res.status(500)
        .json({
        error: 'Internal error please try again'
      });
    } else if (!user) {
      res.status(401)
        .json({
          error: 'Incorrect email or password'
        });
    } else {
      user.isCorrectPassword(password, function(err, same) {
        if (err) {
          res.status(500)
            .json({
              error: 'Internal error please try again'
          });
        } else if (!same) {
          res.status(401)
            .json({
              error: 'Incorrect email or password'
          });
        } else {
          // Issue token
          const payload = { email };
          const token = jwt.sign(payload, secret, {
            expiresIn: '1h'
          });
          res.cookie('token', token, { httpOnly: true })
            .sendStatus(200);
        }
      });
    }
  });
});
```

Рисунок 3.48 – Видача токену користувачу

Тепер, було створено спосіб видавати підписаний токен уповноваженим користувачам, потрібно визначити, які маршрути в додатку не підходять для неавторизованих користувачів. У цьому випадку необхідно, щоб певний маршрут був доступний лише за умови, що клієнт-запитувач має дійсний токен. По-перше, треба переконатися, що у на сервері встановлений пакет cookie- parser, щоб можна було реалізувати можливість аналізу файлів cookie, переданих браузером (рис. 3.49).

Після цього додамо middleware cookie-parser до конфігурації Express:

```
// server.js
const cookieParser = require('cookie-parser');
...
app.use(cookieParser());
```

Рисунок 3.49 – Ініціалізація cookie-parser

Тепер можна створити власний Express middleware, який розміщуватиметься між запитом та захищеним маршрутом та перевірятиме, чи є запит авторизованим. Middleware буде шукати токен у cookie із запиту, а потім перевіряти його (рис. 3.50).

```
// middleware.js
const jwt = require('jsonwebtoken');
const secret = 'mysecretsshhh';
const withAuth = function(req, res, next) {
  const token = req.cookies.token;
  if (!token) {
    res.status(401).send('Unauthorized: No token provided');
  } else {
    jwt.verify(token, secret, function(err, decoded) {
      if (err) {
        res.status(401).send('Unauthorized: Invalid token');
      } else {
        req.email = decoded.email;
        next();
      }
    });
  }
}
module.exports = withAuth;
```

Рисунок 3.50 – Створення Express middleware

Нарешті, можна використовувати цей middleware, коли є необхідність захистити маршрут. Достатньо просто відредагувати його конфігурацію, щоб використовувати новий middleware (рис. 3.51):

```
// server.js
const withAuth = require('./middleware');
...
app.get('/api/secret', withAuth, function(req, res) {
  res.send('The password is potato');
});
```

Рисунок 3.51 – Використання middleware у маршрутах Express

Пізніше стане в нагоді спосіб просто запитати сервер, чи є дійсним маркер, збережений у файлах cookie браузера.

Для цього достатньо створити простий маршрут, який поверне статус HTTP 200, якщо користувач має дійсний маркер (рис. 3.52):

```
app.get('/checkToken', withAuth, function(req, res) {
  res.sendStatus(200);
});
```

Рисунок 3.52 – Метод перевірки токена

Після створення серверної частини, яка може реєструвати та аутентифікувати користувачів, можна працювати над front-end частиною додатку. Для початку створимо простий компонент React з формою, яка буде використовуватися для аутентифікації користувача (рис. 3.53):

```
import React, { Component } from 'react';
export default class Login extends Component {
  constructor(props) {
    super(props)
    this.state = {
      email : '',
      password: ''
    };
  }
  handleInputChange = (event) => {
    const { value, name } = event.target;
    this.setState({
      [name]: value
    });
  }
  onSubmit = (event) => {
    event.preventDefault();
    alert('Authentication coming soon!');
  }
  render() {
    return (
      <form onSubmit={this.onSubmit}>
        <h1>Login Below!</h1>
        <input
          type="email"
          name="email"
          placeholder="Enter email"
          value={this.state.email}
          onChange={this.handleInputChange}
          required
        />
        <input
          type="password"
          name="password"
          placeholder="Enter password"
          value={this.state.password}
          onChange={this.handleInputChange}
          required
        />
       <input type="submit" value="Submit"/>
      </form>
    );
  }
}
```

Рисунок 3.53 – Форма авторизації користувача

Наш експрес-додаток встановлює для нас cookie httpOnly із підписаним JSON веб-токеном, тому все, що потрібно зробити для аутентифікації, це правильно перенаправити користувача, якщо він отримає статус код 200 HTTP під час виклику методу аутентифікації із додатку React (рис. 3.54):

```
onSubmit = (event) => {
  event.preventDefault();
  fetch('/api/authenticate', {
    method: 'POST',
    body: JSON.stringify(this.state),
    headers: {
      'Content-Type': 'application/json'
    }
  })
  .then(res => {
    if (res.status === 200) {
      this.props.history.push('/');
    } else {
      const error = new Error(res.error);
      throw error;
    }
  })
  .catch(err => {
    console.error(err);
    alert('Error logging in please try again');
  });
}
```

Рисунок 3.54 – Переадресація користувача після авторизації

Тепер можна додати компонент Login до конфігурації маршруту (рис. 3.55):

```
import Login from './Login';
...
<Route path="/login" component={Login} />
```

Рисунок 3.55 – Додання компоненту до конфігурації маршруту

Отже, налаштовано процес аутентифікації, який отримує підписаний токен із сервера, зберігає його в cookie і згодом використовує цей маркер для доступу до захищених маршрутів на сервері.

Наостанок, необхідний спосіб вказати маршрути для захисту у інтерфейсі, щоб користувач не бачив компонент, якщо він не ввійшов у

систему. Натомість необхідно перенаправити його на сторінку входу, як це слід робити в добре спроектованому веб-додатку.

Для цього використовується концепція, яка називається компонентами вищого порядку, щоб обернути маршрути компонентів, які необхідно захистити.

Компонент вищого порядку – це не що інше, як функція, яка приймає компонент і повертає компонент. Отже, необхідно створити компонент вищого порядку, withAuth, який буде приймати компонент, який необхідно захистити, наприклад <Secret />, і трохи модифікувати його, щоб користувачі не мали до нього доступу, якщо вони не ввійшли в систему (рис. 3.56):

```
import React, { Component } from 'react';
import { Redirect } from 'react-router-dom';
export default function withAuth(ComponentToProtect) {
  return class extends Component {
    constructor() {
      super();
      this.state = {
        loading: true,
        redirect: false,
      };
    }
    componentDidMount() {
      fetch('/checkToken')
        .then(res => {
          if (res.status === 200) {
            this.setState({ loading: false });
          } else {
            const error = new Error(res.error);
            throw error;
          }
        })
        .catch(err => {
          console.error(err);
          this.setState({ loading: false, redirect: true });
        });
    }
    render() {
      const { loading, redirect } = this.state;
      if (loading) {
        return null;
      }
      if (redirect) {
        return <Redirect to="/login" />;
      }
      return <ComponentToProtect {...this.props} />;
    }
  }
}
```

Рисунок 3.56 – Захищені маршрути React

Після цього створюємо модель сутності «Configuration», розробляємо додаткові методи для відтворення і збереження інформації, створюємо React-інтерфейс для відтворення інформації. Як результат отримуємо інтерфейс авторизації (рис. 3.57), інтерфейс трансляції відео потоку (рис. 3.58) та інтерфейс відображення вмісту сховища (рис. 3.59).

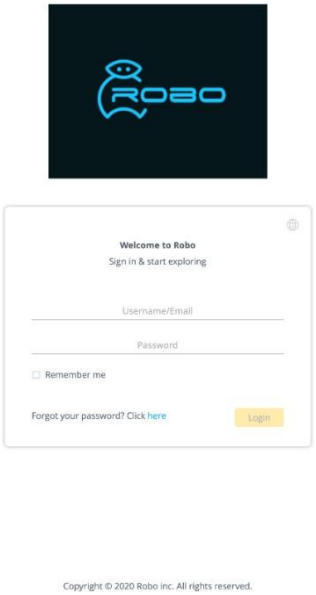

Рисунок 3.57- Інтерфейс авторизації

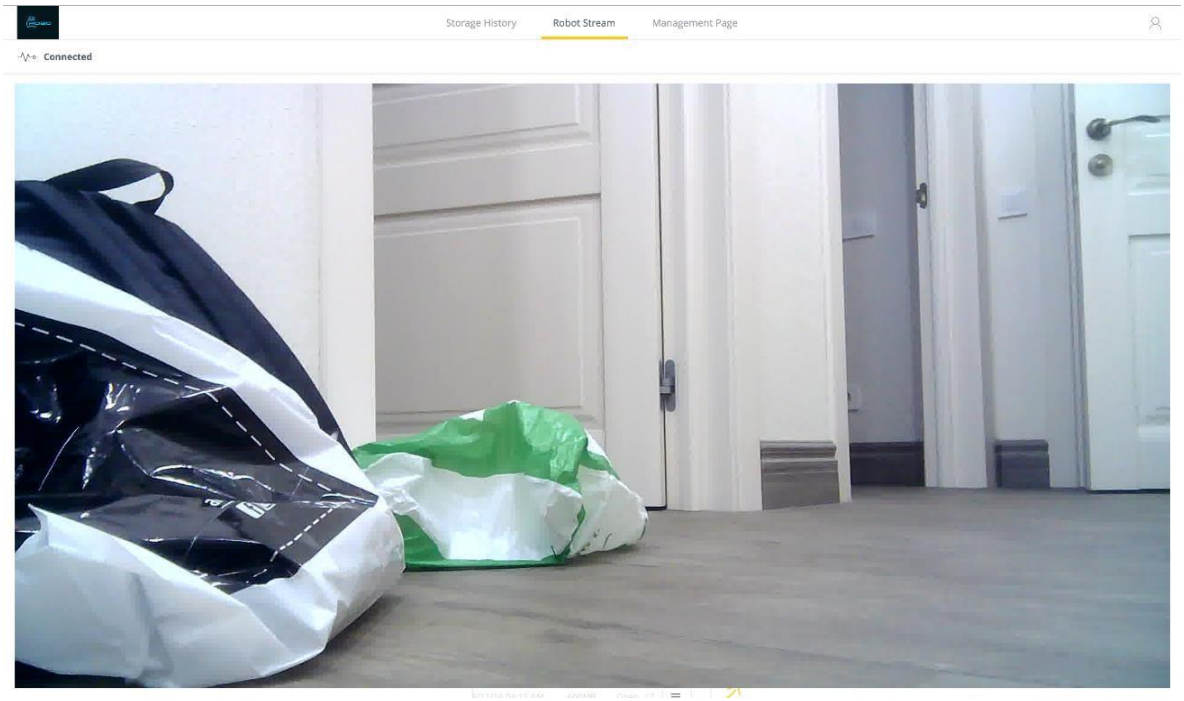

Рисунок 3.58 – Інтерфейс трансляції відео-потоку

Рисунок 3.59 – Інтерфейс відображення вмісту сховища

3.6 Заходи із забезпечення умов праці розробника роботизовано платформи

Кожне робоче місце відповідає ДНАОП 0.00-1.31-99. Розміщення робочих місць показане на рисунку 3.1. Висота робочої поверхні столу для ПК складає 690 мм, ширину столу - 1200 мм, глибину столу - 1000 мм Ці розміри відповідають розмірам робочих місць, показаних на рисунку 3.60. Робочий стіл має простір для ніг заввишки 620 мм і шириною 510 мм Сидіння під’йомно-поворотне, регульоване по висоті, куту нахилу сидіння і спинки, по відстані спинки до переднього краю сидіння і по висоті підлокітників.

Для зменшення перевантаження зорових аналізаторів екран відеотерміналу розташувати на оптимальній відстані від очей: при розмірі екрану по діагоналі 17" - 700.800 мм(монітор LG - Flatron F700P).

Робота з ПК відноситься до групи В (відладка програм, переклад і редагування та ін.) і встановлена 8-годинна робоча зміна, інакше - тривалість робіт групи В перевищує 4 год і виконувані роботи відносяться до III

категорії робіт. Для зменшення дії психофізіологічних ОВПФ (розумове перенапруження, монотонність праці і емоційні перевантаження) для цієї категорії робіт слід встановити перерви по 20 хв кожен через 2 год після початку робіт, 10 хв через кожну годину роботи. Загальна тривалість перерв за 8-годинний день повинна складати 60 хв.

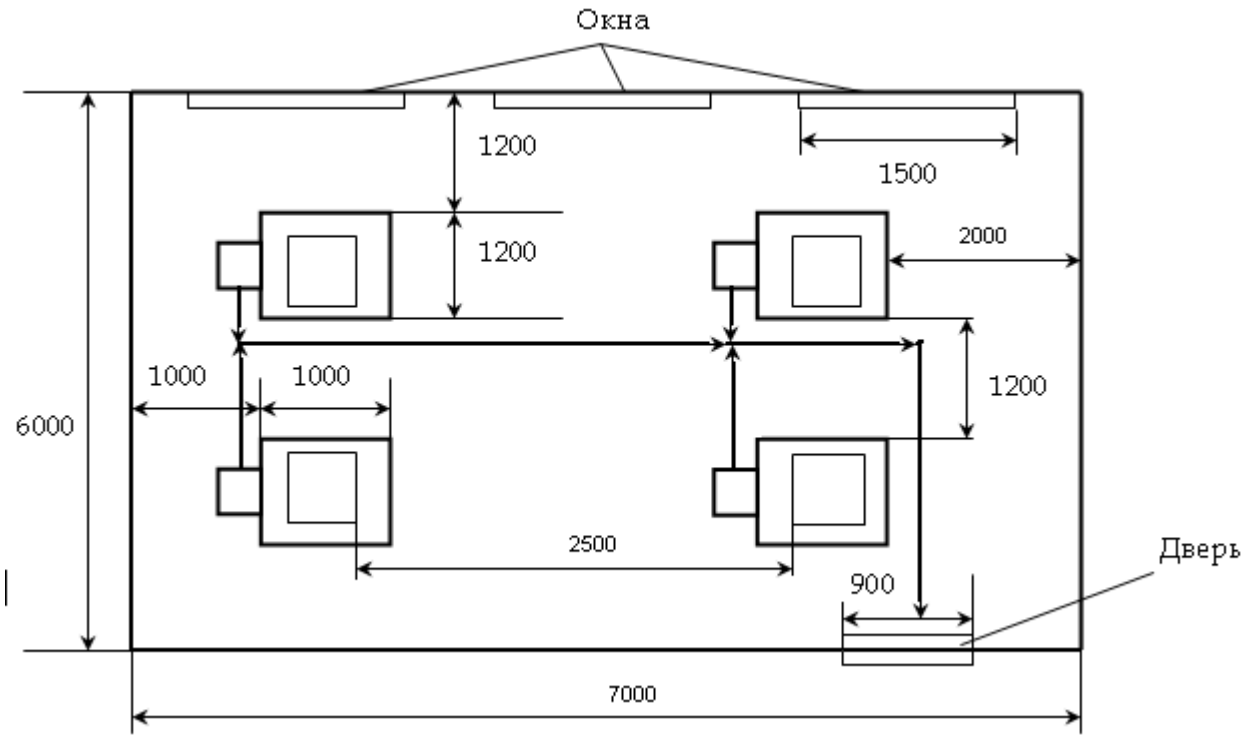

Рисунок 3.60 – Розміщення робочих місць і маршрут евакуації при пожежі

Робота розробника виконується сидячи і не вимагає систематичної фізичної напруги. Згідно ГОСТ 12.1.005-88, робота належить до категорії легкої 1а і для неї встановлені параметри мікроклімату, приведені в табл. 3.2.

Таблиця 3.2 – Оптимальні параметри мікроклімату

Пора року	Температура повітря, град. С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодне	22-24	40-60	≤ 0.1
Тепле	23-25	40-60	≤ 0.1

Для забезпечення встановлених параметрів мікроклімату в приміщенні слід застосовувати кондиціонування.

Згідно СНиП 2.09.02-85 в лабораторії використовуються тверді горючі матеріали з температурою спалаху зверху 61 ^{o}C, тому по вибухопожежній та пожежній небезпеці це приміщення слід віднести до категорії В. Згідно ПУЭ-85 клас пожежної небезпеки приміщення П-IIа, оскільки в лабораторії знаходяться тверді і волокнисті пальні речовини. Будівля виконана із залізобетонних конструкцій і цеглини і відноситься до I міри вогнестійкості, згідно ДБН В.1.1.7-2002.

Причиною пожежі в приміщенні можуть бути коротке замикання електропроводки, несправність електроустаткування, нагріваючи провідників, порушення правил пожежної безпеки, а також підвищена температура усередині приміщення.

Пожежна безпека в лабораторії забезпечується системою відвертання пожежі, протипожежного захисту і організаційно-технічними заходами.

ВИСНОВКИ

У роботі було розглянуто основні методології проектування 3D- моделей прототипів, наведено переваги використання пакету САПР Fusion 360, спроектовано модель роботизованої рухомої охоронної платформи, а також розглянуто процес 3D-друку та процесу підготовки моделі до 3D-друку.

Було розглянуто роботу із інтерфейсом I2C у Rasbperry Pi, проведено підключення усіх модулів і кодування алгоритмів роботи із ними.

Створено модуль авторизації для веб-платформи, розроблено модулі відображення прямої трансляції інформації з системи, а також записаних даних. Розроблено код для передачі і зберігання інформації на сервері.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Nevliudov, I., Yevsieiev, V., Demska, N., & Novoselov, S. (2020). Development of a software module for operational dispatch control of production based on cyber-physical control systems. Innovative Technologies and Scientific Solutions for Industries, (4 (14)), 155–168.

2. Nevliudov I., Tsymbal O., Bronnikov A., Mordyk O. Internet of things for robotic projects. Сучасний стан наукових досліджень та технологій в промисловості. 2020. № 3 (13). С. 58-64.

3. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2021. – 55 с.

4. Дипломне проектування для студентів усіх форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології»: довід. / І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, Г.В. Пономарьова. –Київ, 2018. – 320 с.

5. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. Введ. 2015-06-22. К. Держстандарт України, 2017 – 29 с.

6. Гаврик С. С., Кострова Г. Ю. Моделювання корпусу багатоцільової мобільної робототехнічної платформи з удосконаленими маніпуляційними можливостями. Автоматизація та приладобудування («Automation and Development of Electronic Devices» ADED-2021) [Електронний ресурс]: збірник студентських наукових статей [редкол.: І.Ш. Невлюдов та ін.]. Харків

: ХНУРЕ, 2021. Вип. 2. С. 79–84.

7. Робототехніка. Підручник / [В. І. Костюк, Г. О. Спину, Л. С. Ямпольський, М. М. Ткач.] К.: Вища школа. - 1994. - 447 с.

8. Робототехніка та мехатроніка: навч. посіб. / Л.І. Цвіркун, Г. Грулер ; під заг. ред. Л.І. Цвіркуна ; М-во освіти і науки України, Нац. гірн. ун-т. 3-тє вид., переробл. і доповн. Дніпро: НГУ, 2017. 224 с.

10. Невлюдов І. Ш., Демська Н. П., Скрипник К. Є. Аналіз технології побудови локальної карти середовища мобільного робота. Технология приборостроения. 2019, №2. с. 10–13

11. Невлюдов І.Ш., Євсєєв В.В., Максимова С.С. Автоматизована система керування технологічними процесами в SCADA системі TRACE MODE 6: Навчальний посібник. Харків: 2018. 316 с.

12. Михайлов Є.П. Маніпулятори та промислові роботи [Текст]: підручник / Михайлов Є.П., Лінгур В.М. Одеса: ОНПУ, 2019, 233 с.

13. Цимбал О.М., Бронніков А.І. Адаптивність у прийнятті рішень роботів. Восточно-Европейский журнал передовых технологий, Харьков, 2011, № 4/4 (52), С. 40-43.

14. Цимбал О.М., Бронніков А.І. Адаптивні процеси у завданнях робототехніки // Системи обробки інформації, Харків. – 2012. – Вип. 3 (101), том 1, С. 68 -73.

15. Бронніков А.І., Методи пошуку оптимального керування. Технология приборостроения. Харьков, 2015, № 2 , С. 53 - 55.

16. Nevlyudov I., Tsymbal O., Bronnikov A. Intelligent means in the system of managing a manufacturing agent. Сучасний стан наукових досліджень та технологій в промисловості. 2018. № 1 (3). С. 33-47.

17. Цымбал А.М., Бронников А.И. Моделирование адаптивного принятия решений в ИСУ роботом. Вестник БГТУ им. В.Г. Шухова, Белгород. 2013. №4, С. 173-176.

18. Невлюдов І.Ш., Демська Н.П., Чала О.О., Демська А.І. / Групове управління гнучкими виробничими системами у виготовленні МЕМС

виробів. Міжнародна науково-практична конференція «Математичне моделювання процесів в економіці та управлінні проектами і програмами (ММП-2018)», Коблево, 10-14 вересня 2018 р. Харків: ХНУРЕ, 2018. С. 101 - 103

19. Теоретична механіка. Статика. Кінематика: посіб. для студ. вищ. навч. закл. / І. В. Кузьо, Т. М. Ванькович, Я. А. Зінько. Л.: Вид-во «Растр-7», 2010. 324 с.

20. Дудюк Д. Л. Гнучке автоматизоване виробництво і роботизовані комплекси: Навч. посібник / Д. Л. Дудюк, С. С. Мазепа, М. М. Мисик. Львів: "Магнолія плюс" СПД ФО В. М. Піча, 2005. 278 с.

21. Писаренко А. В., Цвєлодуб А. О. Розробка інструментальних засобів MATLAB/Simulink для проектування керуючих пристроїв систем управління на базі ПЛІС. Вісник Національного технічного університету України Київський політехнічний інститут. Сер.: Інформатика, управління та обчислювальна техніка. №52. 2010. С 53-58.

22. Лоторев, П. В., Курочкин, А. Г., & Гривачев, А. В. (2016). Математическая модель динамической коррекции маршрута подвижного робота. Наукоемкие технологии, 17(3), 21-25.

23. Невлюдов І.Ш., Євсєєв В.В., Андрусевич А.О. Проектування мобільних роботів на базі одноплатних комп'ютерів (Raspberry Pi та мови Python 3.6). Харків: ФОП Панов А.М. 2020. 264 с.

24. Мешковский, Е. О., & Курмашев, А. Д. (2020). Построение математической модели четырёхколёсного мобильного робота с двумя дифференциальными приводными блоками. Инновации и инвестиции, (2), 113-118.

25. Колесниченко, Е. Ю., Павловский, В. Е., Орлов, И. А., Алисейчик, А. П., Грибков, Д. А., & Подопросветов, А. В. (2018). Математическая модель робота на омни-колесах, расположенных в вершинах прямоугольного треугольника. Мехатроника, автоматизация, управление, 19(5), 327-330.

26. Берестова С. А., Мисюра Н. Е., Митюшов Е. А. Кинематическое

управление движением колесных транспортных средств, Вестн. Удмуртск. ун-та. Матем. Мех. Компьют. науки, 2015, том 25, выпуск 2, 254–266

27. Кампион Г., Бастен Ж., д’Андреа-Новель Б. Структурные свойства и классификация кинематических и динамических моделей колесных мобильных роботов, Нелинейная динам., 2011, том 7, номер 4, 733–769

28. Бартенев, В. В., Яцун, С. Ф., & Аль-Еззи, А. С. (2011). Математическая модель движения мобильного робота с двумя независимыми ведущими колесами по горизонтальной плоскости. Известия Самарского научного центра Российской академии наук, 13(4-1).

29. Сизых, В. Н., Баканов, М. В., & Мухопад, Ю. Ф. (2018). Математическая модель для адаптивного управления трёхколёсным мобильным роботом. In Транспортное, горное и строительное машиностроение: наука и производство (pp. 9-18).

30. Крахмалев, О. Н. (2012). Математическое моделирование динамики манипуляционных роботов. В мире научных открытий, (8-1), 51-59.

31. Сімута Р.Р., Петраков Ю.В. Моделювання поверхонь деталей в CAD/CAM системах. Процеси механічної обробки в машинобудування. № 12. 2012. С 149-158

32. Бучинський М. Я., Горик О. В., Чернявський А. М., Яхін С. В. Основи творення машин. Харків: Вид-во «НТМТ», 2017. 448 с.