Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

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

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

(тема)
Розроблення програмного засобу для моделювання впливу системи автоматичного управління на рух мобільної платформи

Виконав: студент 2 курсу, гр. КІТПВм-22-1
Волков Єгор Юрійович
(прізвище, ініціали)

Спеціальність
151 Автоматизація та компю'терно-інтегровані технології
освітньої програми Комп'ютерно-інтегровані технологічні процеси і виробництва
(код і повна назва напряму)

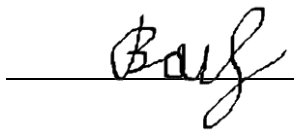
Тип програми освітньо-професійна
(повна назва освітньої програми)

Керівник доц. Чала О. О.
(посада, прізвище, ініціали)

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

2024 р.

Я Волков Єгор Юрійович, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

« 18 » Січня 2024р.  Волков Є. Ю.

Харківський національний університет радіоелектроніки

Факультет	Автоматики і комп'ютеризованих технологій
Кафедра	Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти	другий (магістерський)
Спеціальність	Автоматизація та компю'терно-інтегровані технології
Тип програми	освітньо-професійна
Освітня програма	151 Комп'ютерно-інтегровані технологічні процеси і виробництва

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 01 » 09 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Волкову Єгору Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення програмного засобу для моделювання впливу системи автоматичного управління на рух мобільної платформи
затверджена наказом по університету від 03.11. 2023 р. № 1287 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 16. 01. 2024 р.
3. Вихідні дані до роботи
3.1 Мова програмування для реалізації програми – Java Script
3.2 Середа розробки – Visual Studio Code
3.3 Застосування законів ТАУ та ПІД регулятора для управління рухом мобільної платформи
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Аналіз технічного завдання
4.2 Вступ
4.3 Застосування ТАУ та ПІД регуляторів у робототехніці
4.4 Програмна реалізація ПІД регулятора
4.5 Розробка програмного коду для завдань моделювання
4.6 Експериментальне дослідження програми

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)
Демонстраційний матеріал представлений у форматі презентації PowerPoint.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз літератури за темою роботи	10.10.23 – 17.10.23	Виконав
2	Вплив ТАУ на роботів	18.10.23 – 30.10.23	Виконав
3	Програмна реалізація ПІД	01.11.23 – 16.11.23	Виконав
4	Розробка коду програми	17.11.23 – 19.12.23	Виконав
5	Інструкція з інсталяції програми	20.12.23 – 29.12.23	Виконав
6	Перевірка праці симуляції програми	04.01.24 – 11.01.24	Виконав
7	Оформлення пояснювальної записки	12.01.24 – 18.01.24	Виконав
8	Подання роботи на рецензію	19.01.24 – 21.01.24	Виконав
9	Подання роботи на підпис зав. кафедри	22.01.24 – 23.01.24	Виконав
10	Подання атестаційної роботи в ЕК	24.01.2024	Виконав

Дата видачі завдання 01.09.2023 р.

Студент _____ Волков Є. Ю.
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. Чала О. О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 65 с., 30 рис., 3 дод., 10 джерел.

АВТОМАТИЗАЦІЯ, ПІД, ПРОГРАМНА СИМУЛЯЦІЯ, РОБОТИЗОВАНІ ПЛАТФОРМИ, МОДЕЛЮВАННЯ, КЕРУВАННЯ ПРИСТРОЯМИ.

Мета дослідження – вдосконалення методів налаштування ПІД регуляторів для параметрів руху мобільної платформи шляхом розробки програмного засобу.

Об’єкт дослідження – система автоматичного управління платформи.

Предмет дослідження – вплив системи автоматичного управління та законів ТАУ на рух мобільної платформи.

Для досягнення мети проаналізовано вплив законів ТАУ на рух автоматичних роботів, описані існуючи програми для симуляції руху різноманітних роботизованих платформ, описані їх різні види та застосування ПІД-регулятору у робототехніці. Було викладено як програмно реалізувати ПІД-регулятор. Розроблено код для симуляції впливу регулятора на рух платформи. Описано етапи установки програми та проведено її тестування. Також розписані основні правила поведінки при використанні персонального комп’ютеру. У ході проведених експериментів було доведено успішну роботу розробленої програми.

ABSTRACT

Explanatory note: 65 p., 30 pic., 3 applications, 10 sources.

AUTOMATION, PID, SOFTWARE SIMULATION, ROBOTIC PLATFORMS, SIMULATION, DEVICE CONTROL.

The purpose of the research is to improve the methods of adjusting the PID controllers for the movement parameters of the mobile platform by developing a software tool.

The object of development is the system of automatic control of the platform.

The subject of development is the influence of the automatic control system and TAU laws on the movement of the mobile platform.

To achieve the goal, the impact of TAU laws on the movement of automatic robots was analyzed, existing programs for simulating the movement of various robotic platforms were described, their various types and the application of the PID controller in robotics were described. It was explained how to programmatically implement a PID regulator. A code was developed to simulate the influence of the controller on the movement of the platform. The stages of installing the program are described and its testing is carried out. Basic rules of behavior when using a personal computer are also described. During the conducted experiments, the successful operation of the developed program was proven.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

НДР – науково-дослідна робота;

ПЗ – програмне забезпечення;

ПІД – пропорційно-інтегрально-диференціальний;

ПК – персональний комп’ютер;

ТАУ – теорія автоматичного керування;

ТЗ – технічне завдання.

ВСТУП

На даний момент тема автоматичного керування є дуже актуальною, так як все більше і більше в різних галузях починають використовуватися роботи, від стандартних маніпуляторів до цілих підприємств, які повністю управляються та функціонують за рахунок різноманітних роботів, а для їхнього руху потрібно розробити певну послідовність дій.

Усі роботи рухаються по заданим координатам, які мають бути дуже точно прокладені. Адже якщо маніпулятор відхилиться від заданого маршруту на один градус, він вже опиниться в абсолютно не тому місці, в якому мав опинитися і це спричинить багато наслідків, найкритичніше з усіх буде брак.

Автоматичні машини не змогли б правильно функціонувати без налагодженої системи автоматичного пересування та керування, адже будь-яка помилка в їхньому шляху чи позиціонуванні координат призведе до тотальної дезорієнтації платформи.

Мета дослідження – вдосконалення методів налаштування ПІД регуляторів для параметрів руху мобільної платформи шляхом розробки програмного засобу.

Об’єкт дослідження – система автоматичного управління платформи.

Предмет дослідження – вплив системи автоматичного управління та законів ТАУ на рух мобільної платформи.

Для виконання мети необхідно:

– проаналізувати існуючи аналоги програм;

– розробити код та логіку програми;

– провести декілька експериментів з програмою.

Робота виконана згідно [1-2].

1 ЗАСТОСУВАННЯ ЗАКОІВ ТАУ ДЛЯ УПРАВЛІННЯ РУХОМ РОБОТІВ

1.1 Застосування ТАУ та ПІД регуляторів у робототехніці

1.1.1 Закони ТАУ у автономному руху

Розглянемо як закони автоматики застосовуються для автоматичного управління. За допомогою законів автоматики можна створити системи управління, які підтримують задану швидкість руху. Прикладом цього буде адаптивний круїз-контроль у машинах. Розглянемо як він влаштований та працює.

Адаптивний круїз-контроль - це покращена система круїз-контролю, яка не тільки автоматично підтримує швидкість руху, але і дотримується безпечної дистанції до автомобіля, що йде попереду, адаптуючись під поточну ситуацію на дорозі.

На відміну від неактивної системи, яка лише підтримує задану швидкість руху, система адаптивного круїз-контролю:

– прискорює автомобіль до заданого значення;

– уповільнює його, аж до повної зупинки;

– утримує задану швидкість, забезпечуючи не тільки комфорт у тривалих поїздках, а й значно на 10% і більше скорочуючи витрату палива.

Сама система складається з трьох вузлів.

По-перше, це фронтальні датчики, розташовані в передньому бампері, рідше – за фальш решіткою радіатора. Це – «очі» круїз-контролю, тому мають бути чистими від бруду та снігу, інакше ефективність системи прагнути до нуля.

По-друге, це додаткові датчики, що відстежують, чи їде автомобіль по прямій, в гірку або з неї, на який кут відхилене рульове колесо, що контролюють включену в КПП передачу, навіть інформацію з дорожньої розмітки та придорожніх знаків.

По-третє, це електронний блок управління адаптивного круїз-контролю. Від нього залежить як якість функціонування, та й робота взагалі. Він аналізує інформацію, що надходить, даючи команди:

– системі подачі палива змінити кут нахилу дросельної заслінки;

– гальмівному контуру збільшити або навпаки послабити силу притискання гальмівних колодок до дисків.

Принцип роботи адаптивного круїз-контролю:

– збирається вся інформація з основних та другорядних датчиків;

– блок управління у режимі реального часу безперервно її аналізує;

– залежно від заданої водієм швидкості та ситуації на дорозі – порожня, пряма траса або, навпаки, авто, що йде попереду, пригальмовує/перебудовується, дає команду на прискорення або, навпаки, уповільнення. При цьому система керування може гальмувати двигуном або зменшувати його потужність шляхом короткочасного збіднення повітряно-паливної суміші.

Також регулювання швидкості, тісно пов'язане з проходженням траєкторії руху мобільної платформи. Вона дозволятиме точніше слідувати побудованому маршруту, що підвищить оптимізацію нашого руху.

Роботи і автономні транспортні засоби використовують закони автоматики для слідування заздалегідь визначеним траєкторіям. Це може включати в себе коригування траєкторії на основі зворотного зв’язку від датчиків і камер. Розглянемо використання ПІД-регулятора для відстеження траєкторії роботом, але перед цим розберемося як він працює, а також де він може застосовуватися.

1.1.2 Що таке ПІД-регулятор

Пропорційно-інтегрально-диференціальний (ПІД) регулятор – пристрій в керуючому контурі зі зворотним зв'язком. Використовується в системах автоматичного управління для формування сигналу керуючого з метою отримання необхідних точності і якості перехідного процесу. ПІД-регулятор формує керуючий сигнал, що є сумою трьох доданків, перший з яких є пропорційним різниці вхідного сигналу і сигналу зворотного зв'язку (сигнал

неузгодженості), друге - інтегралу сигналу неузгодженості, третє - похідної сигналу неузгодженості.

ПІД регулятор – один із найпоширеніших автоматичних регуляторів. Він настільки універсальний, що застосовується практично скрізь, де потрібне автоматичне керування.

ПІД регулятор складається з трьох складових (рисунок 1.1): пропорційної P, що інтегрує I і диференціює D, формується просто як сума трьох значень, помножених кожна на свій коефіцієнт. Ця сума після обчислень стає керуючим сигналом, який подається на керуючий пристрій.

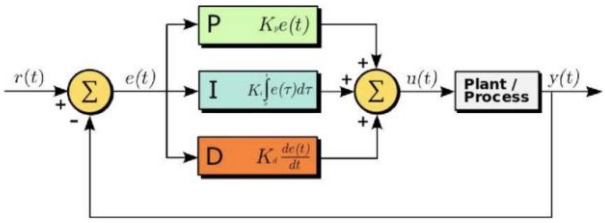


Рисунок 1.1 – Схема принципу роботи ПІД-регулятора [3]

kP, kI і kD це і є ті коефіцієнти, які потрібно налаштувати для роботи ПІД-регулятора. Значення тут можуть бути різні, це залежить від конкретної системи. Будь-який коефіцієнт може дорівнювати нулю, і в такому випадку анулюється вся його компонента. Тобто регулятор можна перетворити на П, ПІ, ПД, та інші поєднання. Різні системи вимагають різного підходу, саме тому ПІД-регулятор такий універсальний.

Пропорційна складова є різницею поточного значення з датчика та установки. Ця різниця називається помилкою регулювання, тобто як далеко знаходиться система від заданого значення. Виходить чим більше помилка, тим більше буде керуючий сигнал і тим швидше система наводитиме керовану величину до заданого значення. П складова є основною в ПІД-регуляторі, регулятор може непогано працювати тільки на ній одній. P складова виправляє помилку в даний час.

Інтегральна складова просто підсумовує в саму себе ту саму помилку, різницю поточного і заданого значення, помножену на період дискретизації системи, тобто на час, що минув з попереднього розрахунку dt - фактично бере інтеграл від помилки за часом. У самому регуляторі це ще множиться на коефіцієнт kI, яким налаштовується різкість цієї складової. В інтегральній складовій буквально накопичується помилка, що дозволяє регулятору з часом повністю її усунути, тобто привести систему до заданого значення з максимальною точністю. I складова виправляє минулі помилки, що накопичилися.

Диференціальна складова являє собою різницю поточної та попередньої помилки, поділену на час між вимірюваннями, тобто на ту ж dt, яка має загальний період регулятора. Іншими словами – це похідна від помилки за часом. Фактично D складова реагує на зміну сигналу з датчика, і чим сильніше відбувається ця зміна, тим більше значення додається до загальної суми. D дозволяє компенсувати різкі зміни в системі і при правильному налаштуванні запобігти сильному перерегулюванню та зменшити розгойдування. Коефіцієнт Д дозволяє налаштувати вагу, чи різкість даної компенсації, як та інші коефіцієнти регулюють свої складові. D складова в першу чергу потрібна для швидких систем, тобто для систем із різкими змінами. D складова виправляє можливі майбутні помилки, аналізуючи швидкість.

Для налаштування регулятора потрібно варіювати коефіцієнти:

При збільшенні kP збільшується швидкість виходу на встановлене значення, збільшується сигнал, що управляє. Чисто математично система не може прийти рівно до заданого значення, так як при наближенні до установки П складник пропорційно зменшується. При подальшому збільшенні kP реальна система втрачає стійкість та починаються коливання.

При збільшенні kI зростає швидкість компенсації помилки, що накопичилася, що дозволяє вивести систему точно до заданого значення з плином часу. Якщо система повільна, а kI занадто великий - інтегральна сума сильно зросте і відбудеться перерегулювання, яке може мати характер коливань, що не

затухають, з великим періодом. Тому інтегральну суму в алгоритмі регулятора часто обмежують, щоб вона не могла збільшуватися та зменшуватись до нескінченності.

При збільшенні kD зростає стабільність системи, вона дає системі змінюватися дуже швидко. У той же час kD може стати причиною неадекватної поведінки системи та постійних стрибків сигналу керуючого, якщо значення з датчика вагається. На кожну різку зміну сигналу датчика Д складова буде реагувати зміною керуючого сигналу, тому сигнал з датчика потрібно фільтрувати.

1.1.3 Застосування ПІД-регулятору у робототехніці

ПІД – це алгоритм керування, який знаходиться в електричному приводі та секції керування мобільної платформи. Для автоматичної навігації та управління зазвичай існує три контури керування: крутний момент, швидкість і положення. Як правило, привід керуватиме контуром крутного моменту, а «контролер» керує швидкістю та положенням.

Погане налаштування може мати низку шкідливих наслідків для програми рухомої платформи, найочевиднішим з яких є те, що система не стабільна чи точна. Погано налаштована система може задавати невірний маршрут і невірно регулювати швидкість руху автоматизованого роботу. Окрім неточності, погане налаштування може призвести до поломки машини, а також до перегріву та пошкодження двигунів і приводів.

Справлятися з мінливими навантаженнями є дуже складним завданням для мобільної платформи. Правильне налаштування важливе, але в екстремальній ситуації може знадобитися динамічна налаштування, щоб впоратися з цією проблемою.

З моменту, коли потрібно перемістити робота в певну точку на тестовій поверхні, від моменту, коли введено потрібну позицію, і доки вона не була змінена регулюванням замкнутого циклу, виникла певна затримка. Ця затримка була матеріалізована тим фактом, що робот не зупинився в потрібному місці і

продовжував рухатися протягом короткого часу. Через це виникали коливання навколо потрібного місця, що проявлялося в переміщенні робота вперед і назад до потрібної контрольної точки. Тому для усунення цієї затримки, а також для точного налаштування вихідних величин (швидкості, крутного моменту) безщіткового двигуна постійного струму використовувався ПІД-регулятор (пропорційно-інтегратор-похідний). Проста структура та реалізація, загальна хороша продуктивність керування та надійна конструкція є кількома перевагами ПІД-регулювання. Алгоритм ПІД складається з трьох параметрів налаштування, які можна інтерпретувати, по відношенню до часу, наступним чином:

– P – пропорційний член, використовується для розрахунку поточних похибок;

– I – інтегральний член, надає інформацію про кількість попередніх помилок;

– D – похідний термін, забезпечує передбачення майбутніх помилок на основі поточної швидкості коливання розміру [3].

На рисунку 1.2 показана схема налаштування системи пересування за допомогою ПІД-регулятора.

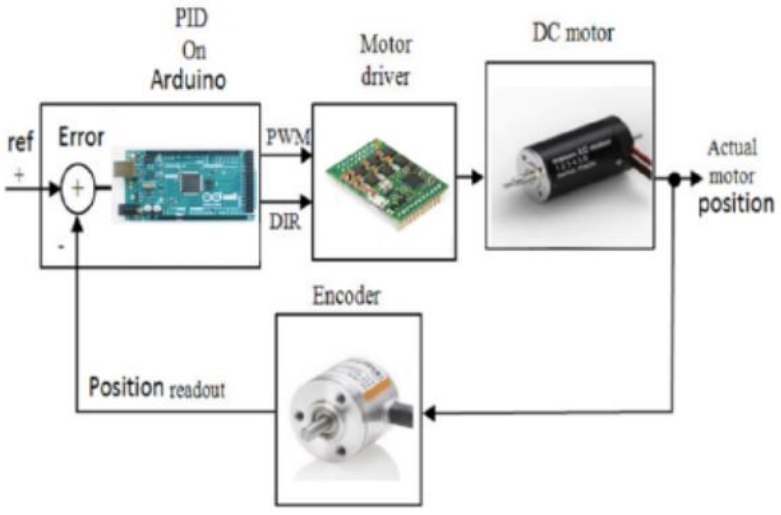


Рисунок 1.2 – Замкнутий цикл реалізації ПІД-регулятора [4]

Реалізація ПІД-регулятора дозволяє регулювати швидкість безщіткових двигунів постійного струму та виконується за допомогою цифрового сигнального процесора плати Arduino. Реакцію системи забезпечують два кодери Kübler, які забезпечують на виході 4096 PPR. Оскільки робот був запрограмований на досягнення певної точки на ігровій поверхні шляхом введення значення позиції, необхідно використовувати два набори констант PID, один набір для руху (переміщення), а інший для орієнтації (обертання). Оскільки найважливішою вимогою до двигунів було досягнення заданого положення за відносно короткий час, стаціонарна похибка кутової швидкості повинна бути менше 1%. Звичайно, цього недостатньо, тому що двигуни повинні розганятися до стаціонарної швидкості якомога швидше, майже одночасно з увімкненням. Однак кутова швидкість, набагато вища за номінальну, може спричинити поломку двигуна, тому ми запровадили перерегулювання менше ніж на 5% [4].

1.1.4 Застосування нечіткого управління в управлінні колісними роботами

З точки зору алгоритму керування, ми зараз використовуємо більш традиційний PID. Алгоритм ПІД-регулювання має такі переваги, як простий принцип, хороша стабільність, надійна робота та проста реалізація. І це стало основною технологією промислового контролю. У фактичному процесі нанесення, коли робот рухається на низькій швидкості, перешкоди для осі нахилу є низькими, коли робот проходить через нерівну дорожню поверхню, оскільки підвіска шасі може відігравати певну роль у поглинанні ударів. У цьому випадку PID-алгоритм може зіграти дуже хороший ефект. Однак, коли робот проїжджає по нерівній дорозі на високій швидкості, механічна підвіска демонструє свої обмеження. Нахильна вісь карданного підвісу робота неминуче зазнає сильних збурень. Діапазон корекції помилок традиційного алгоритму PID обмежений, а параметри, які були встановлені, є фіксованими значеннями, які не можуть впоратися з цією ситуацією. Тому адаптивність низька. У цьому випадку вимоги до конструкції підвіски шасі відносно високі для досягнення кращого ефекту стабільності, але це також має певні обмеження. Нечітке керування — це

алгоритм керування, заснований на досвіді керування людиною. Це вводить нечіткість міркування, як людське судження та прийняття рішень. Він має сильну адаптивність і хорошу ефективність керування складними системами, такими як нелінійність і велика затримка [5].

Однак нечітке керування по суті є методом керування частковим розрядом, і вплив регулювання на похибку в стаціонарному стані не є очевидним під час налаштування. Крім того, хоча для цього не потрібна точна математична модель контрольованого об’єкта, якщо ви хочете використовувати нечітке керування, вам потрібно точно розуміти характеристики об’єкта та ефективно встановлювати правила керування входом і виходом. Метод нечіткого адаптивного ПІД-регулювання поєднує в собі переваги цих двох методів ПІД-регулювання та нечіткого керування. Регулюючи три параметри алгоритму ПІД-регулювання в режимі реального часу, він не тільки має характеристики простої реалізації та делікатного керування ПІД-регулюванням, але також має характеристики нечіткого керування. Контроль має сильну стійкість і здатність адаптуватися, тому карданний підвіс все ще може підтримувати хорошу стабільність у складних дорожніх умовах [5].

Розглянувши як закони ТАУ та ПІД регулятор застосовуються для керування автономними мобільними платформами далі, потрібно визначиться як виглядатиме майбутня програма, а також її функціонал. Для цього розглянемо вже існуючі рішення програм із симуляції руху мобільної платформи, проаналізуємо та на їх основі уявимо, як виглядатиме наш додаток у майбутньому.

Прикладом програми для симуляції роботизованих систем є V-REP від швейцарської компанії Coppelia Robotics.

1.2 Аналіз програми V-REP

V-REP симулятор – це результат спроб відповідати всім вимогам до універсальності та масштабованості середовища моделювання. Поряд із

традиційними підходами до моделювання, які є і в інших тренажерах, V-REP додає кілька додаткових підходів.

Коли є необхідність будувати складні сцени, то поки що немає кращого варіанту, ніж використовувати розподілену систему контролю. Такий підхід спрощує завдання шляхом поділу управління суб'єктами, прискорює моделювання, розподіляє навантаження процесора на кілька ядер чи кілька машин, і це дозволяє контролювати виконання алгоритмів. Існують, однак, вимоги до моделювання, які не повинні бути забуті задля досягнення цієї мети.

Виконання коду симуляції здійснюється за допомогою наступних трьох методів:

Код може виконуватися на окремій машині або роботі, підключеному до симулятора машини через конкретну шину (наприклад, роз'єм, послідовний порт та ін.). Основна перевага такого підходу полягає в оригінальності контролера (керуючий код рідний і працюватиме на оригінальному устаткуванні). Іншою перевагою є зниження обчислювального навантаження на моделювання машини. Проте такий підхід накладає серйозні обмеження щодо синхронізації з циклом моделювання.

Керуючий код виконується на тій же машині, але у відмінному від циклу моделювання процесі (або іншому потоці).

При такому методі ми також можемо скористатися зниженим збалансованим навантаженням на ядра процесора, але це буде супроводжуватися відсутністю синхронізації з циклом моделювання. Більшу частину часу такий процес буде відбуватися в парі з відставанням зв'язку або затримками перемикання. Цей метод часто реалізується через зовнішні програми або плагіни, що виконуються, завантажені в симулятор.

Контрольний код виконується на тій же машині і в тому потоці, що і цикл моделювання.

Основною перевагою такого підходу є синхронізація з циклом моделювання, відсутність затримок. Однак такий метод стає можливим лише зі

збільшенням обчислювального навантаження на процесор. Цей метод часто реалізується через плагіни, завантажені у симулятор.

Найбільш поширеними незручностями перерахованих вище методів є погана переносимість і погане масштабування імітаційних моделей: оскільки код контролю не прив'язаний до відповідної імітаційної моделі, він повинен бути розподілений, складений і встановлений окремо. Це збільшує кількість проблем із сумісністю на різних платформах та кількість конфліктів з іншими бібліотеками. Гнучкість також знижується, тому що потрібно перекомпілювати та перезавантажити виконуваний файл для кожного невеликого коду модифікації. Копія моделі, як і у разі моделювання мульті-роботів, підтримуватиметься за допомогою дротових механізмів, які запускають нові елементи керування для кожного екземпляра імітаційної моделі.

V-REP дозволяє користувачеві використовувати кілька можливостей для моделювання (рис.1.3).

	Embedded script	Add-on	Plug-in	Remote API client	ROS node
API mechanism	Regular API	Regular API	Regular API	Remote API	ROS
Supported programming language	Lua	Lua	C/C++	C/C++, Python, Java, Matlab, Urbi	Depends on ROS support
Available API functions	>280 functions	>270 functions	>400 functions	>100 functions	>150 services, publisher and subscriber types
API is user-extendable	Yes, with custom Lua functions	Yes, with custom Lua functions	Yes, V-REP is open source	Yes, remote API is open source	Yes, ROS interface is open source
Control entity is external and can be native	No	No	No	Yes	Yes
Simulation models are fully portable and scalable	Yes	No	No	No	No
Communication lag	None	None	None	Yes	Yes
Synchronous operation is supported*	Yes, inherent. No delays	Yes, inherent. No delays	Yes, inherent. No delays	Yes. Slow due to comm. lag	Yes. Slow due to comm. lag
Asynchronous operation is supported*	Yes	No	Yes	Yes	Yes
Can start, stop, pause and step a simulation	Stop, pause	Start, stop, pause	Start, stop, pause, step	Start, stop, pause, step	Start, stop, pause, step

Рисунок 1.3 – Порівняння п'яти методів програмування, що підтримуються у V-REP [6]

Вбудовані скрипти є найбільш потужною відмінною особливістю V-REP. Основний цикл моделювання lua скрипт (так званий “основний сценарій”) – опрацьовує загальні функціональні можливості. Наприклад, він викликає різні функції обробки кінематики чи динаміки. Основний сценарій також відповідає за виклик дочірнього скрипту у каскадний спосіб. Дочірній скрипт, на відміну від основного скрипту, прикріплюється до конкретного об'єкта чи конкретної частини моделювання у процесі циклу моделювання. Він є невід'ємною частиною сценарію об'єкта і буде повторюватися разом з ним. Як такий дочірній скрипт є цілком портативним і масштабованим елементом управління: в ньому є один єдиний файл, що містить визначення моделі разом з її контролем або функціоналом, немає проблеми сумісності на різних платформах, немає необхідності в явній компіляції, жодного конфлікту між декількома версіями однієї і тієї ж моделі та ін. Дочірні скрипти можуть бути запущені в потоковій або потокової реалізації [6].

Віддалений інтерфейс API V-REP дозволяє взаємодіяти з V-REP або моделюванням зовнішнього об'єкта за допомогою сокетів. Він складається з дистанційного сервера API служб та віддалених API клієнтів. На стороні клієнта може бути вбудований компактний код (C/C++, Python, Java, Matlab & Urbi) практично на будь-якому обладнанні, в тому числі на реальних роботах, і дозволяє здійснювати віддалений виклик функції, а також швидку потокову передачу даних туди і назад. На стороні клієнта функції називаються майже як звичайні функції, з двома винятками: віддалені функції API приймають додатковий аргумент, який є режимом роботи і повернення коду помилки. Режим роботи дозволяє викликати функції блокування (чекатиме відповіді сервера), або не блокування (читатиме потокові команди з буфера, або починати/зупиняти службу потокової передачі даних на стороні сервера). Зручність використання віддаленого API, доступність на всіх платформах та його невеликі розміри роблять його цікавою альтернативою ROS інтерфейсу [6].

Розглянемо, як виглядає основний інтерфейс V-REP (рис 1.4), і яких роботів там можна симулювати (рис 1.5 – 1.6).

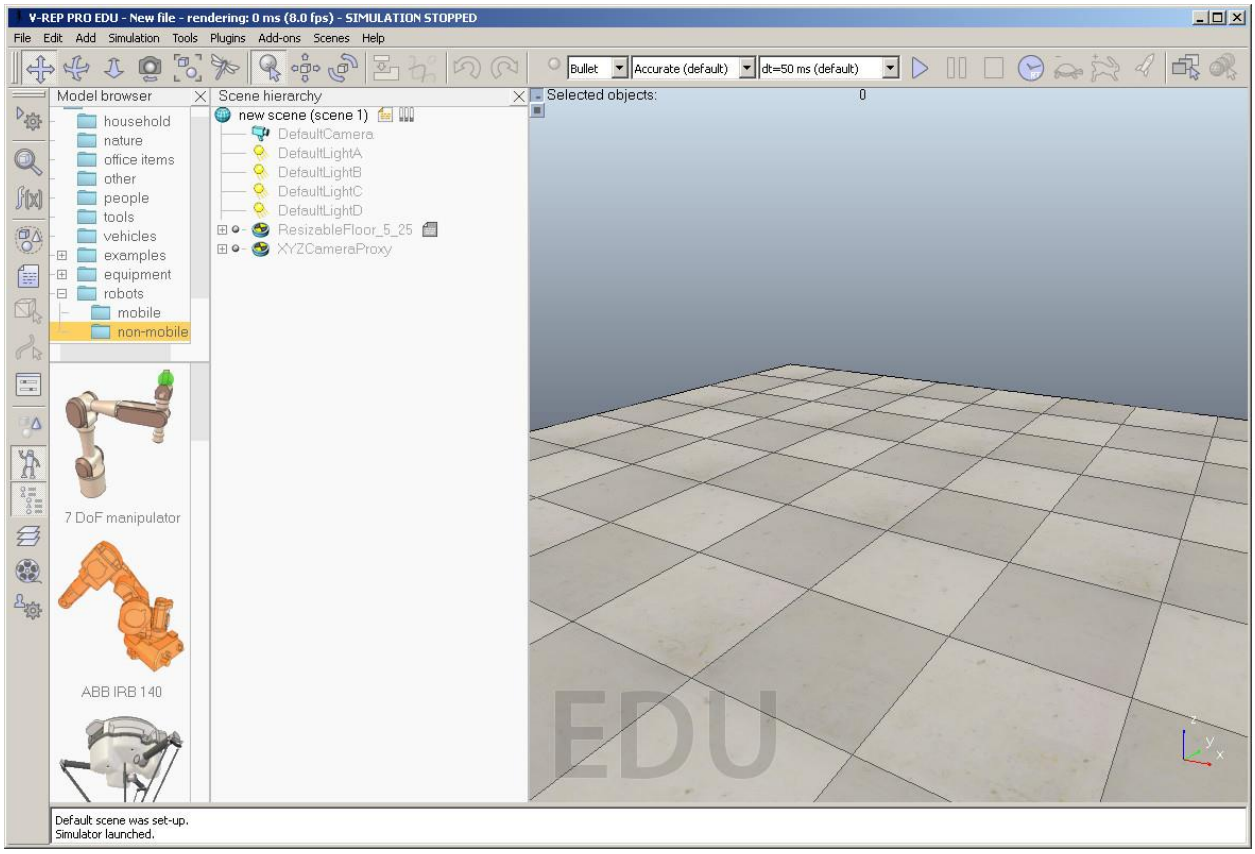


Рисунок 1.4 – Основний інтерфейс V-REP [6]

Рисунок 1.5 – Мобільні роботи представлені у програмі [6]

Рисунок 1.6 – Стаціонарні роботи та маніпулятори представлені у V-REP [6]

Розглянувши 3Д роботосимулятор V-REP, стає зрозуміло, що це досить монументальний додаток для симулювання руху роботів, хоча в ньому і відсутні досить важливі функції для проекту. Тому потрібно буде створювати власний додаток, беручи від V-REP все найкраще.

Для створення подібної програми розглянемо програму LabVIEW, яка допоможе у цьому.

1.3 Застосування LabVIEW для створення програм

LabVIEW - це кросплатформове графічне середовище розробки додатків. LabVIEW – у принципі універсальна мова програмування. І хоча цей продукт часом тісно пов'язаний з апаратним забезпеченням National Instruments, проте він не пов'язаний з конкретною машиною. Існують версії Windows, Linux, MacOS. Вихідні тексти переносяться, а програми виглядатимуть однаково у всіх системах. Код, згенерований LabVIEW, також може бути також виконаний на Windows Mobile або PalmOS. Ця мова може успішно використовуватися для створення великих систем, для обробки текстів, зображень і роботи з базами даних [7].

LabVIEW - дуже високорівнева мова. Однак ніщо не заважає включати «низькорівневі» модулі до LabVIEW-програм. Навіть якщо ви хочете використовувати асемблерні вставки - це також можливо, треба лише згенерувати

DLL і вставити виклики в код. З іншого боку, високорівнева мова дозволяє запросто робити дуже нетривіальні операції з даними, на які в звичайній мові могли піти багато рядків (якщо не десятки рядків) коду. Втім, заради справедливості слід зазначити, що деякі операції низькорівневих мов (наприклад, роботу з покажчиками) не так просто реалізувати в LabVIEW через його «високорівневість». Зрозуміло, мова LabVIEW включає основні конструкції управління, що мають аналоги і в традиційних мовах:

– змінні (локальні чи глобальні);

– розгалуження (case structure);

– for – цикли з перевіркою завершення та без;

– while – цикли;

– угруповання операцій.

У LabVIEW програмні модулі називаються Virtual Instruments (Віртуальні Інструменти) або VI. Вони зберігаються у файлах із розширенням *.vi. VIs – це цегла, з якої складається LabVIEW – програма. Будь-яка LabVIEW програма містить щонайменше один VI. У термінах мови Сі можна досить сміливо провести аналогію з функцією з тією різницею, що в LabVIEW одна функція міститься в одному файлі (можна також створювати бібліотеки інструментів). Зрозуміло, один VI може бути викликаний з іншого VI. У принципі кожен VI складається з двох частин – Блок-Діаграма (Block Diagram) та Передня Панель (Front Panel). Блок-діаграма – це програмний код (точніше візуальне графічне уявлення коду), а Передня панель – це інтерфейс (рис. 1.7).

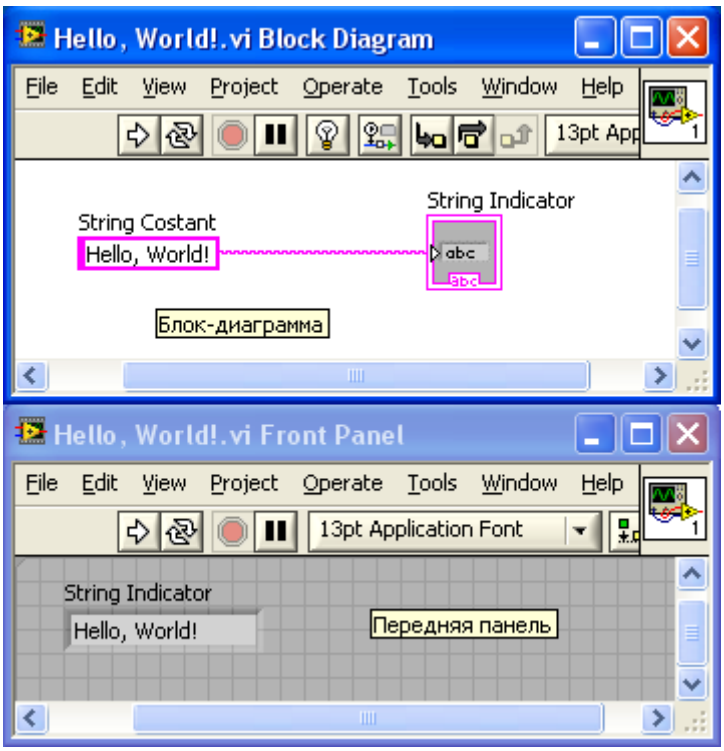


Рисунок 1.7 – Приклад написання «Hello, World!» у LabVIEW [7]

В основі LabVIEW лежить парадигма потоків даних. У наведеному прикладі константа і термінал індикатора з'єднані між собою лінією. Ця лінія називається Wire. Можна назвати її "дротом". По дротах передаються дані від одних елементів іншим. Уся ця концепція називається Data Flow. Суть Блок Діаграми – це вузли (ноди), виходи одних вузлів приєднані до входів інших вузлів. Вузол розпочне виконання лише тоді, коли прибудуть усі необхідні для роботи дані. На діаграмі вгорі дві ноди. Одна з них – константа. Цей вузол самодостатній – він починає виконання негайно. Другий вузол – індикатор. Він відобразить дані, які передає константа (але не відразу, як тільки дані прибудуть від константи) [8].

Стандартне постачання LabVIEW включає також блоки для роботи з ini файлами, реєстром, функції для роботи з двійковими і тестовими файлами, математичні функції, потужні інструменти для побудови графіків (а куди ж без цього в лабораторії-то), а на додаток до вже згаданої можливості викликів DLL, LabVIEW дозволяє працювати з ActiveX компонентами та .net. Починаючи з восьмої версії в LabVIEW було додано підтримку класів — мова стала об'єктно-

орієнтованою. Реалізовану підтримку не можна назвати повною, проте основні риси об'єктно-орієнтованих мов — успадкування та поліморфізм є. Також функціональність мови можна розширити додатковими модулями, наприклад, NI Vision Toolkit – для обробки зображень та машинного зору та інші. А за допомогою модуля Applcation Builder можна згенерувати exe-файл, що виконується [9].

Далі розглянемо які бувають типи колісних роботів, їх переваги та недоліки, а також виберемо яку робитимемо надалі модель робота.

1.4 Типи колісних роботів

Колісні роботи – це роботи, які рухаються по землі за допомогою моторизованих коліс. Ця конструкція простіша, ніж використання проступей або ніжок, а за допомогою коліс їх легше проектувати, будувати та програмувати для пересування по рівній, не дуже пересіченій місцевості. Вони також краще керовані, ніж інші види роботів. Недоліки колісних роботів полягають у тому, що вони не можуть добре орієнтуватися через перешкоди, такі як кам’яниста місцевість, різкі спуски або ділянки з низьким тертям. Найбільшою популярністю на споживчому ринку користуються колісні роботи, їх диференціальне рульове управління забезпечує низьку вартість і простоту. Роботи можуть мати будь-яку кількість коліс, але для статичного та динамічного балансу достатньо трьох коліс. Додаткові колеса можуть додати балансу; однак знадобляться додаткові механізми, щоб утримувати всі колеса в землі, коли місцевість нерівна.

2-колісні роботи. Двоколісним роботам важче втримати рівновагу, ніж іншим типам, тому що вони повинні постійно рухатися, щоб утримуватися у вертикальному положенні. Центр ваги корпусу робота зберігається нижче осі, зазвичай це досягається встановленням батареї під корпусом. Вони можуть мати свої колеса паралельно одне одному, ці транспортні засоби називаються дициклами, або одне колесо перед іншим, тандемом розташовані колеса.

Двоколісні роботи повинні продовжувати рухатися, щоб залишатися у вертикальному положенні, і вони можуть зробити це, рухаючись у напрямку, куди робот падає. Щоб забезпечити рівновагу, основа робота повинна залишатися під його центром ваги. Для робота, який має ліве і праве колесо, потрібно як мінімум два датчики. Датчик нахилу, який використовується для визначення кута нахилу, і датчики коліс, які відстежують положення платформи робота [10].

Прикладом такого робота може бути робот-пилосос Roomba. Це двоколісні пилососи, які автоматично пересуваються, прибираючи кімнату. У них використовується контактний датчик спереду та інфрачервоний датчик у верхній частині (рис. 1.8).

Рисунок 1.8 – Двох колісний робот-пилосос Roomba [11]

3-колісні роботи можуть бути двох типів: з диференційованим керуванням (2 колеса з приводом і додатковим колесом, що вільно обертається, щоб утримувати тіло в рівновазі) або з 2 колесами, що приводяться в рух від одного джерела, і з електроприводом для третього колеса. У випадку диференційовано керованих коліс напрямок робота можна змінити, змінюючи відносну швидкість обертання двох окремо керованих коліс. Якщо обидва колеса рухаються в одному напрямку та з однаковою швидкістю, робот буде їхати прямо. В іншому випадку,

залежно від швидкості обертання та його напрямку, центр обертання може знаходитися де завгодно на лінії, що з’єднує два колеса (рис. 1.9).

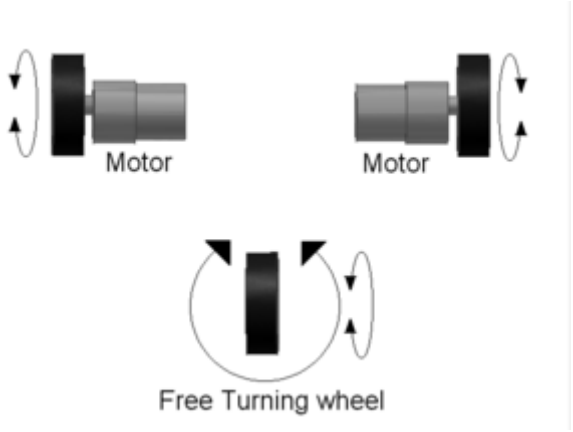


Рисунок 1.9 – 3-х колісний робот з диференціальним керуванням [12]

Центр ваги цього типу робота повинен знаходитися всередині трикутника, утвореного колесами. Якщо збоку від колеса, що вільно обертається, буде встановлено занадто важку масу, робот перекинеться.

4-колісні роботи. 2 колеса з приводом і 2 вільно обертаються. Так само, як диференціально керовані, але з 2 вільно обертовими колесами для додаткового балансу (рис. 1.10).

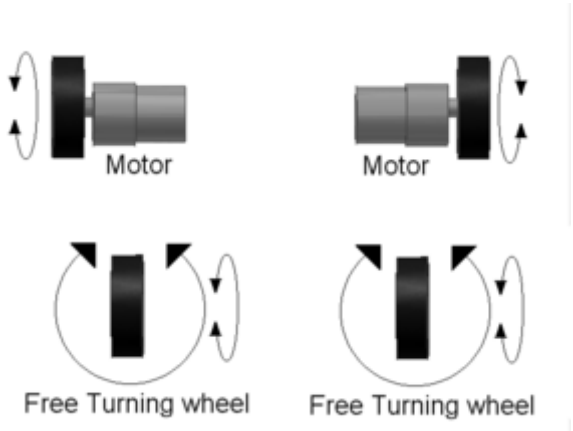


Рисунок 1.10 – Приклад два колеса з приводом, два вільно обертаються [12]

Більш стабільний, ніж триколісна версія, оскільки центр ваги повинен залишатися всередині прямокутника, утвореного чотирма колесами, а не трикутника. Це залишає більший корисний простір. Все ж бажано тримати центр ваги в середині прямокутника, оскільки це найбільш стабільна конфігурація, особливо під час різких поворотів або руху по нерівній поверхні.

Приводні колеса 2 на 2 для пересування, як у танку. Цей вид робота використовує 2 пари приводних коліс. Кожна пара (з'єднана лінією) повертається в одному напрямку. Складна частина цього виду руху полягає в тому, щоб усі колеса оберталися з однаковою швидкістю. Якщо колеса в парі не рухаються з однаковою швидкістю, повільніше буде буксувати (неефективно). Якщо пари не біжать з однаковою швидкістю, робот не зможе їхати прямо. Хороший дизайн повинен включати певну форму автомобільного рульового керування (рис. 1.11).

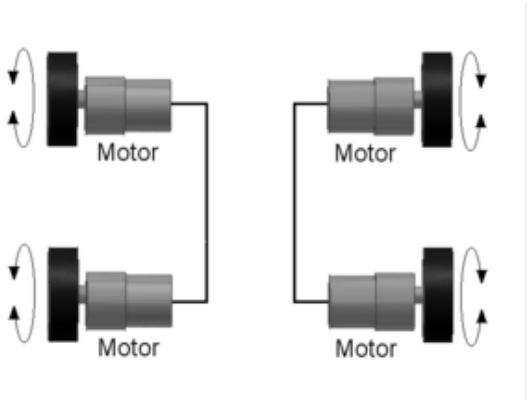


Рисунок 1.11 – Приклад привода на 4 колеса [12]

5 і більше колісні роботи. Для великих роботів. Не завжди дуже практично. Особливо, коли використовуються більш потужні колеса, конструкція стає набагато складнішою, оскільки кожне колесо має обертатися з однаковою швидкістю, коли робот має рухатися вперед. Різниця в швидкості між лівим і правим колесом у роботів з диференційованим керуванням призводить до того, що робот рухається вбік, а не по прямій. Різниця в швидкості між колесами з одного боку призводить до пробуксовування найповільнішого колеса [12].

Іноді до робота додають додаткове колесо вільного обертання з одометрією. Це точніше визначає, як рухається робот. Одометрія на приводних колесах виключає ковзання та інші рухи, тому може бути помилковою.

Mars Rovers (Sojourner, Spirit, Opportunity) — це шість колісних роботів, які після приземлення пересуваються по поверхні Марса. Вони використовуються для огляду території, цікавих орієнтирів і спостереження за поверхнею Марса. Вони мають систему підвіски, яка утримує всі шість коліс у контакті з поверхнею та допомагає їм долати схили та піщану місцевість [10].

Розглянувши всі варіанти колісних робот надалі розглядатиметься, а також імплементуватиметься чотирьох колісний варіант.

1.5 Висновки до розділу

Аналіз літератури показав важливість використання ПІД регуляторів для управління мобільними платформами, а також описано спосіб управління колісними роботами за допомогою нечіткого управління.

Дослідження також показало важливість використання віртуального моделювання роботів перед їхньою фізичною збіркою. Це допомагає заздалегідь побачити неточності складання, дозволяє відразу розуміти, як платформа поводитиметься на різних видах ґрунту, а також допомагає візуалізувати його поведінку на різних нахилах дороги.

Проаналізовано різні види колісних роботів та обрано чотириколісний варіант для його майбутньої реалізації у програмі.

2 ПРОГРАМНА РЕАЛІЗАЦІЯ ПІД РЕГУЛЯТОРА

2.1 Програмна реалізація формул

Для того щоб правильно реалізувати програмно роботу під регулятора, ми повинні розглянути, як виходить керуючий сигнал.

Для того щоб його отримати нам потрібно застосувати формулу:

$$u(t) = P \times K_p + I \times K_i + D \times K_d, \quad (2.1)$$

де *u(t)* – керуючий сигнал;

K_p, K_i, K_d – це пропорційний, інтегральний та диференціальний коефіцієнт регулювання відповідно. Вони допомагають посилити помилку, що дозволяє більш тонко налаштовувати ПІД регулятор;

P, I, D – це пропорційна, інтегральна та диференційна складові регулятора, кожна з яких вираховується за своєю формулою.

Розглянемо формулу Р-складової регулятора:

$$u(t) = P \times K_p, \quad (2.2)$$

де P розраховується за формулою:

$$P = setpoint - input, \quad (2.3)$$

де *setpoint* – це значення, до якого регулятор буде стримуватися;

input – значення датчика, яке виходить після розрахунків регулятора.

На рисунку 2.1 показано роботу П-регулятора та поведінку керуючого сигналу.

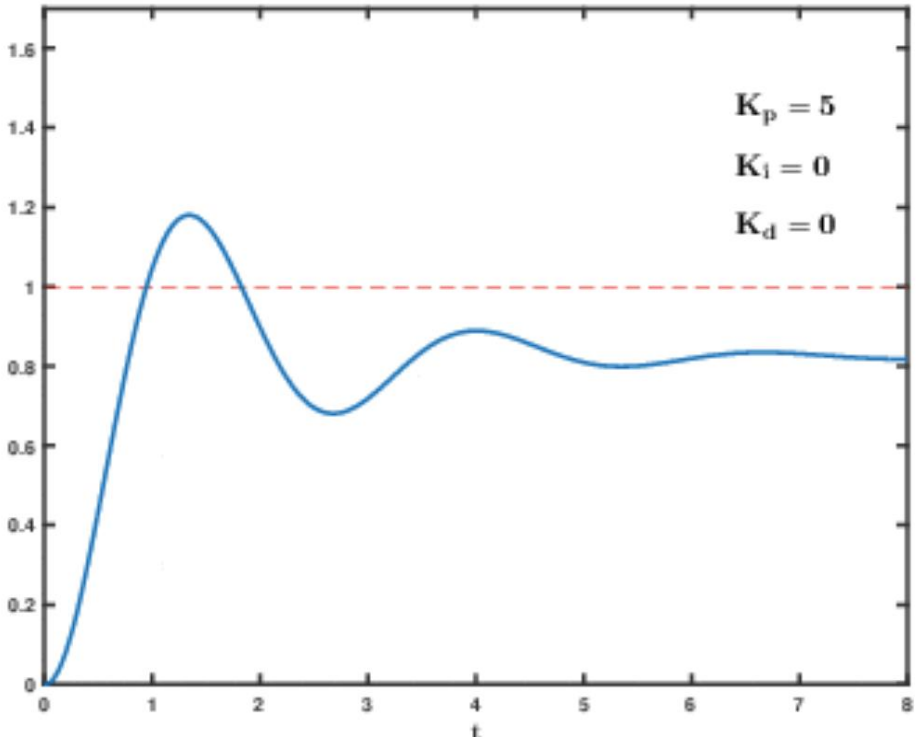


Рисунок 2.1 – Вплив параметра P на сигнал керування

Як видно на рисунку 2.1 керуючий сигнал досить довго підлаштовується до заданого значення, що неприпустимо у системах, де потрібно швидко реагувати на зміни. Із цим допоможе I-регулятор, який зберігає у собі минулі помилки та виправляє їх.

Розглянемо формулу I-регулятора:

$$u(t) = I \times K_i; \tag{2.4}$$

$$I = I + (setpoint\ - input) \times dt, \tag{2.5}$$

де *setpoint* – це значення, до якого регулятор буде стримуватися;

input – значення датчика, яке виходить після розрахунків регулятора;

dt – період обчислення та регулювання.

На рисунку 2.2 показано як P-регулятор та I-регулятор спільно працюють.

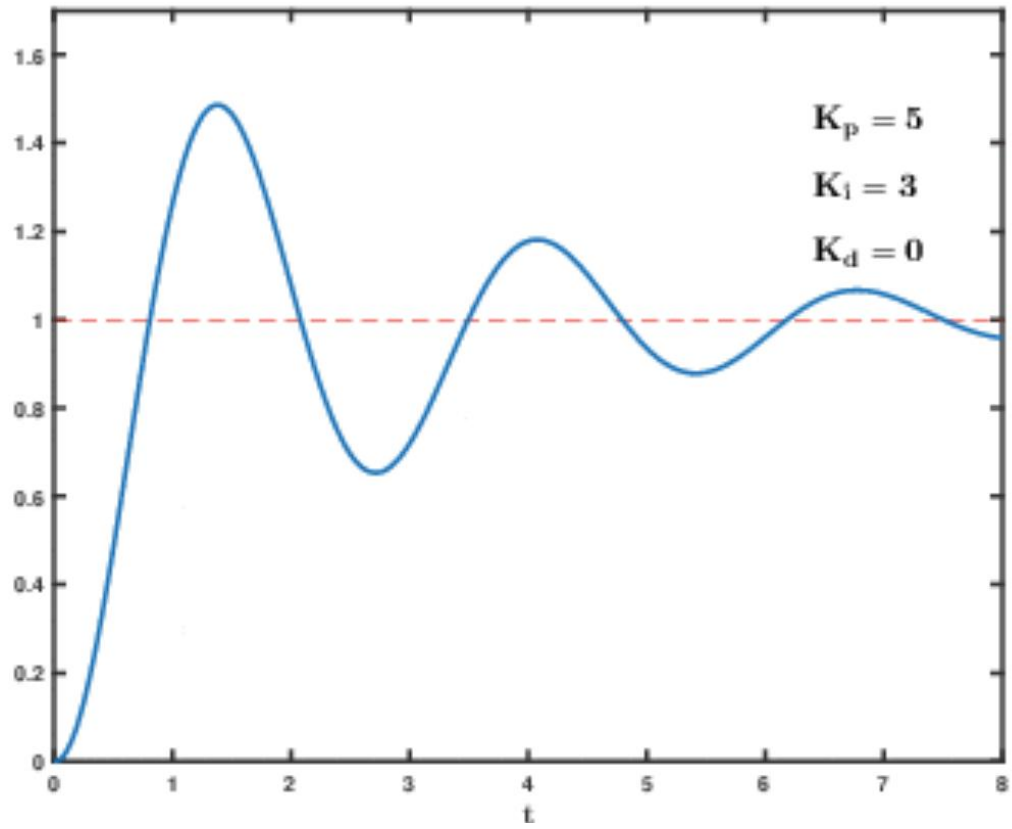


Рисунок 2.2 – Вплив P та I параметрів на керуючий сигнал

Як видно на рисунку 2.2 PI-регулятор вже більш точно підлаштовує сигнал, що управляє, до заданого значення. PI-регулятори чудово підходять для управління бойлерами, батареями і тому подібними пристроями, де не така важлива швидкість, але для автоматичних роботів це не ідеальний варіант, так як платформа повинна швидко змінювати керуючий сигнал і чітко слідувати заданому маршруту. Всі ці речі допоможе реалізувати D-регулятор, що виправляє майбутні помилки системи.

Розглянемо формулу D-регулятора:

$$u(t) = D \times K_d; \tag{2.6}$$

$$D = \frac{error - prevError}{dt}; \tag{2.7}$$

$$error = setpoint - input, \tag{2.8}$$

де *error* – це помилка у нинішній ітерації;

prevError – це помилка попередньої ітерації;

setpoint – це значення, до якого регулятор буде стримуватися;

input – значення датчика, яке виходить після розрахунків регулятора;

dt – період обчислення та регулювання.

На рисунку 2.3 показано роботу PID-регулятора.

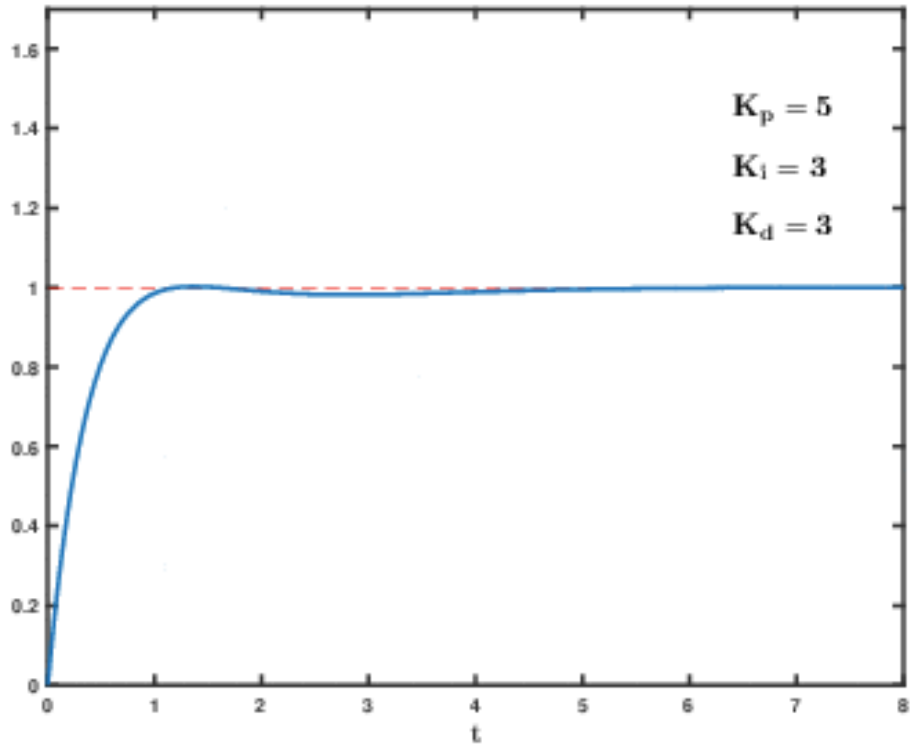


Рисунок 2.3 – Вплив P, I та D параметрів на керуючий сигнал

Для програмної реалізації PID-регулятора необхідно вибрати мову програмування. Після аналізу кількох мов, вибір ліг на Java Script (React). Мова була обрана через легкість і зручність створення на ньому браузерних додатків, простоту написання логіки, а також велику різноманітність бібліотек.

2.2 Висновки до розділу

У цьому розділі були описані всі формули, за допомогою яких PID-регулятор зможе розраховувати число виводу і підтримуватиме частоту. Всі вони будуть використовуватися для написання логіки розрахунків у коді програми.

3 РОЗРОБКА ПРОГРАМИ-СИМУЛЯЦІЇ

3.1 Опис основних елементів програми

Після вибору мови програмування потрібно визначиться з вибором бібліотек, які знадобляться для створення програми.

Використовуватимуться дві бібліотеки. Одна для створення графіків - "Recharts", друга для написання логіки PID-регулятора - "pid-controller".

На рисунку 3.1 показано меню програми. Розглянемо кожен з її елементів, а також, як вони були імплементовані за допомогою коду. Варто уточнити, що програма повністю реалізована та працює у браузері, що полегшує її використання.

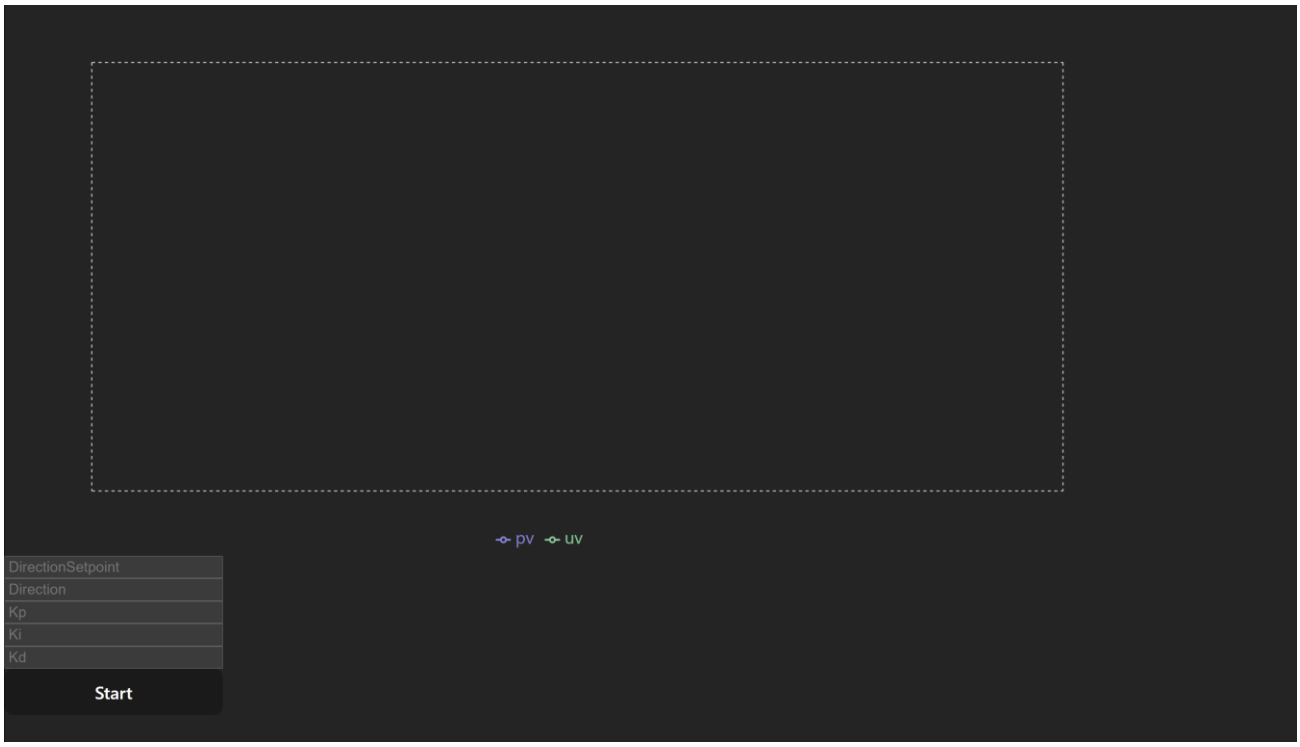


Рисунок 3.1 – Основний екран програми

Як видно з рисунку 3.1, в даний момент програма стоїть на «паузі». Це зроблено для того, щоб користувач міг ввести потрібні значення для тестів, і тільки після перевірки правильності введених значень запускав програму.

Детальніше розглянемо «меню» взаємодії із програмою (рисунок 3.2).

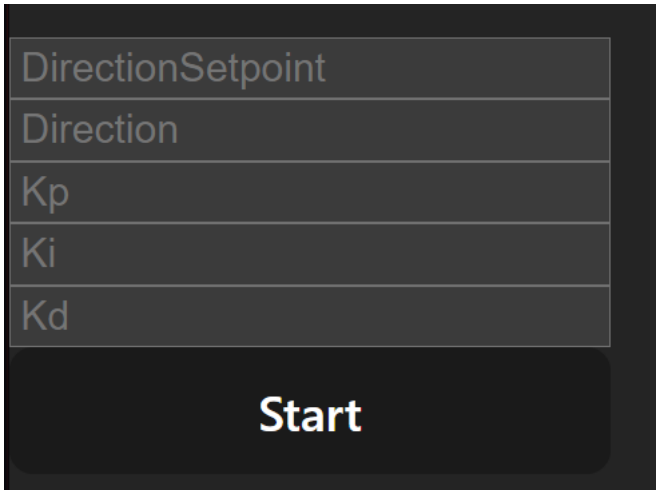


Рисунок 3.2 – Поле для введення значень, що розраховуються

На рисунку 3.2 видно 5 полів для введення значень. DirectionSetpoint – до числа, що вводиться в дане поле, буде підлаштовуватися PID-регулятор. Direction – це число відповідає за стартове становище нашої платформи. Kp, Ki, Kd – це коефіцієнт PID-регулятора.

На рисунку 3.3 та 3.4 показано як за допомогою коду реалізовано функціонал введення, а також запуску та зупинки програми відповідно.

```
  type="text"
  placeholder="DirectionSetpoint"
  onChange={(e) => setDirectionSetpoint(+e.target.value)}
/>
<input
  type="text"
  placeholder="Direction"
  onChange={(e) => setDirection(+e.target.value)}
/>
<input
  type="text"
  placeholder="Kp"
  onChange={(e) => setKp(+e.target.value)}
/>
<input
  type="text"
  placeholder="Ki"
  onChange={(e) => setKi(+e.target.value)}
/>
<input
  type="text"
  placeholder="Kd"
  onChange={(e) => setKd(+e.target.value)}
/>
```

Рисунок 3.3 – Імплементація кодом введення чисел

```
const [isStopped, setIsStopped] = useState(true)
useEffect(() => {
  let intervalId
  if (isStopped === false)
    intervalId = setInterval(DirectionSimulation, timeframe);
  if (isStopped === true)
    clearInterval(intervalId)
  return () => clearInterval(intervalId);
}, [DirectionSimulation, timeframe]);
```

Рисунок 3.4 – Імплементація запуску та зупинки програми

При натисканні кнопки «Start» з'являється ліворуч екрана графік, який відображає роботу регулятора, а справа як це виглядало при роботі з реальним роботом (рисунок 3.5).

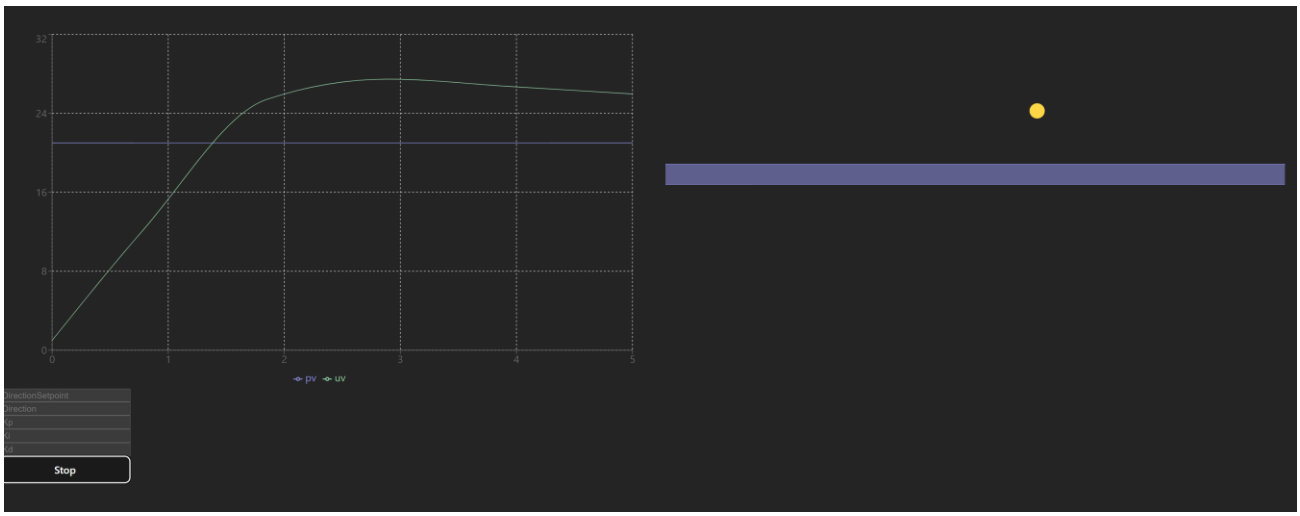


Рисунок 3.5 – Вигляд роботи симуляції

На рисунку 3.5 видно жовту точку, і навіть фіолетову смугу. Крапка – це робот, а фіолетова смуга – дорога, якою він у результаті повинен їхати.

Розглянемо код, який дозволяє виводити розраховані дані у вигляді графіка, зображеного на рисунку 3.6.

```
setData((prev) => {
  return [
    ...prev,
    {
      time: timer,
      uv: direction,
      pv: DirectionSetpoint,
    },
  ];
});
```

Рисунок 3.6 – Функція відтворення графіка

Константа "timer" відповідає за відображення іксової координати на графіку, «direction» та «DirectionSetpoint» відповідають за відображення нинішнього напряму та цільового.

Для симуляції PID-регулятора використовується бібліотека, для користування якої застосовуються деякі команди, зображені на рисунку 3.7.

```
ctr.setInput(direction);
ctr.compute();
```

Рисунок 3.7 – Команди бібліотеки PID-регулятора

На рис. 3.7 прийняті такі позначення:

– setInput це команда, яка відповідає за вибір константи для введення даних;

– Compute – команда, що відповідає за обрахунок початкових даних за формулами PID-регулятора, які були описані у другому розділі роботи.

Також варто зупинитися як встановити всі необхідні компоненти для запуску програми.

3.2 Встановлення та запуск програми

Для використання програми знадобляться дві основні речі – це будь-який браузер, а також програма для редагування та компіляції коду Visual Studio Code, який можна завантажити з головного сайту.

Після завантаження редактора потрібно створити проект. Для початку потрібно створити папку, яка не повинна містити у своїй назві букв кирилиці. Вона міститиме всі необхідні файли для роботи програми моделювання. Після цього відкриваємо створену папку за допомогою Visual Studio Code, або перетягуючи її в поле «Провідник», або натискаючи кнопку «Відкрити папку» (рисунок 3.8)

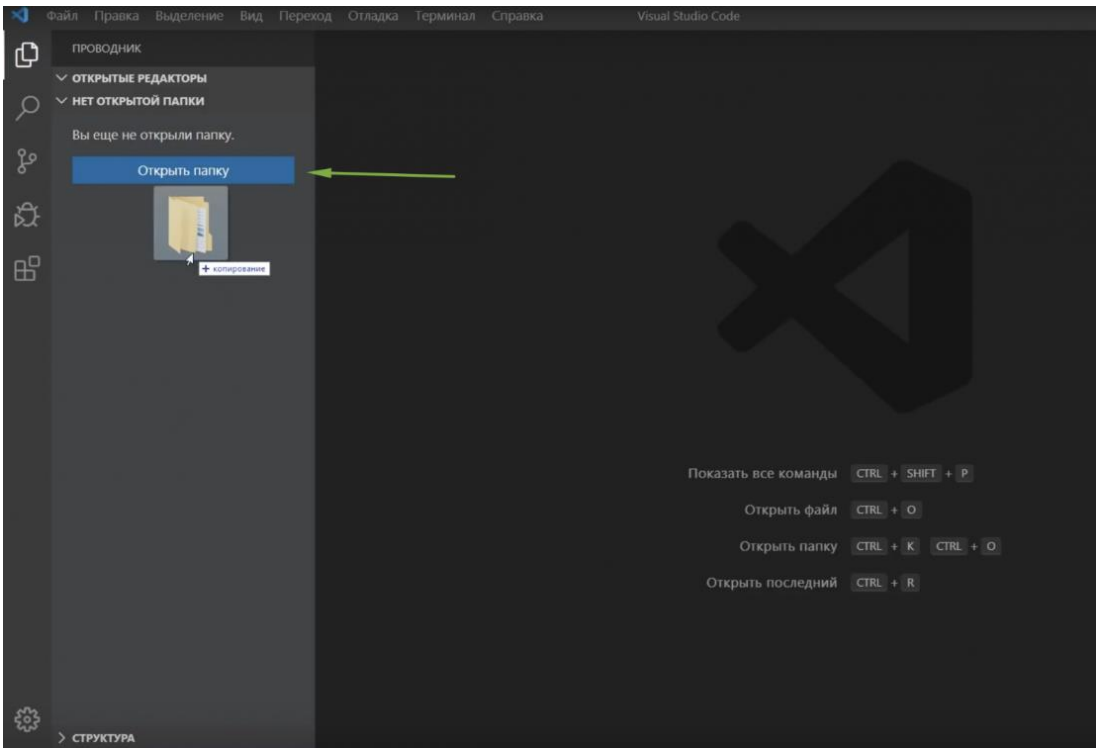


Рисунок 3.8 – Відкриття папки проекту у VS Code

Створимо деякі файли, які потрібні для проекту. Натискаємо Створити файл, пишемо наприклад Index.html. Далі ми можемо створити папку CSS і в ній створити файл style.css.

Для подальшої роботи потрібно обов'язково перевірити встановлену версію Node.js, тому що для встановлення бібліотек, а також локального сервера потрібна певна версія. Для перевірки потрібно відкрити термінал у VS Code та вписати туди команду, як показано на рисунку 3.9.

```
PS D:\Универ\ДИПЛОМ_МАГ\my-project> node -v
v18.16.0
```

Рисунок 3.9 – Перевірка версії Node.js

Для використання програми потрібна версія Node.js 12 або вище, тому оновлювати нічого не потрібно. Далі встановлюємо Vite – це інструмент збирання, створений для забезпечення швидкого та бережливого процесу

розробки сучасних веб-проектів. Для цього потрібно прописати команду у терміналі, показану на рисунку 3.10.

```
$ npm init vite@latest
```

Рисунок 3.10 – Встановлення Vite

Після цього у провіднику з'явиться безліч файлів необхідні правильної роботи Vite (рисунок 3.11).

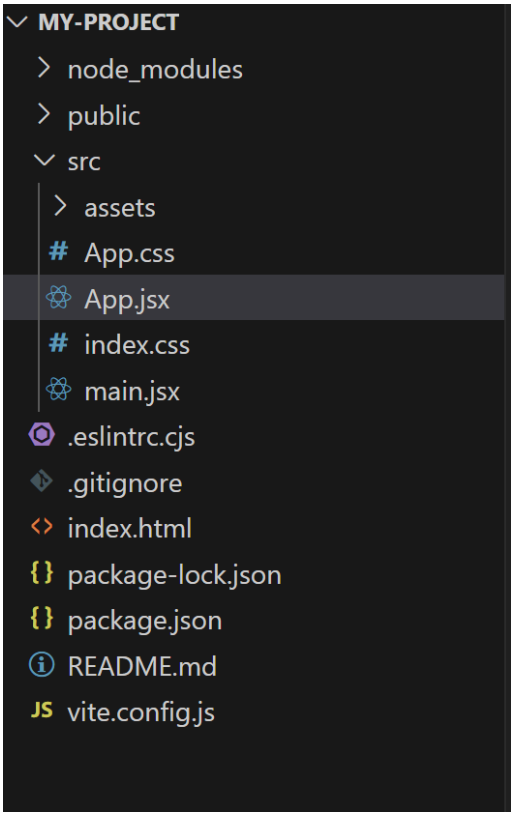


Рисунок 3.11 – Основні файли Vite

Основний код програми буде у файлі App.jsx, де його можна змінювати і підлаштовувати для своєї зручності. Для перевірки правильної працездатності Vite, а також використання програми потрібно написати в термінал команду npm run dev. Якщо з'явиться текст, як показаний рисунку 3.12, отже все працює правильно.

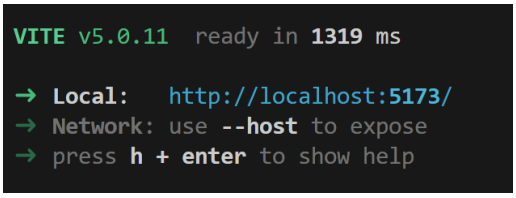


Рисунок 3.12 – Запуск проекту

Після переходу на посилання відкриється браузер, де вже можна працювати з програмою.

Останнє, що потрібно зробити для коректної роботи програми – це встановити дві бібліотеки «recharts» та «pid-controller». Зробити це можна в терміналі за допомогою команди: npm і назва бібліотеки.

3.3 Висновки до розділу

У цьому розділі описано основний функціонал програми, основні частини коду за рахунок яких працює програма. Також було етапно описано встановлення і створення симуляційного докладання.

4 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ ІЗ ПРОГРАМОЮ

4.1 Проведення симуляцій

Розглянемо два приклади симуляції у програмі. Для початку перший розглянемо неправильний, при якому відбуватимуться невгамовні коливання і PID-регулятор не зможе налаштувати правильний висновок. Значення, що вводяться для симуляції $K_p = 1600$ $K_i = 500$ $K_d = 200$ (рисунок 4.1– 4.2).

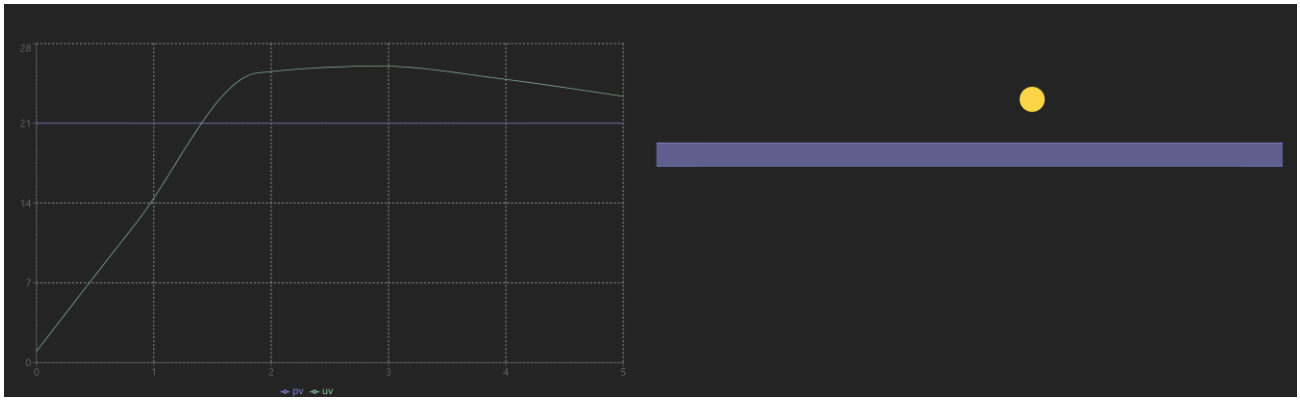


Рисунок 4.1 – Початок симуляції

Як видно з рисунку 4.1, починається все стандартно PID-регулятор починає набирати обертів, відбувається сплеск і з'являється помилка.

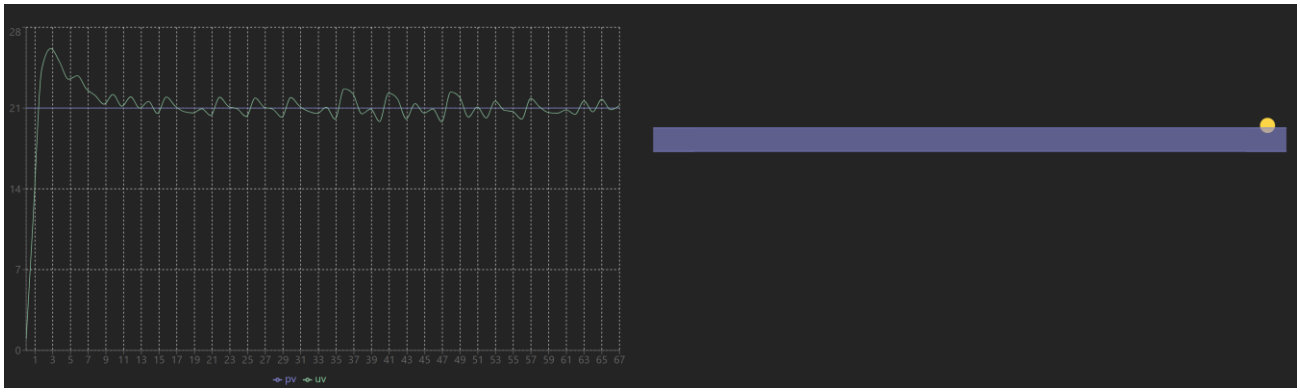


Рисунок 4.2 – Завершення симуляції

PID-регулятор обраховує її і починає зменшувати оберти. Але як видно з рисунка 4.2 він не може визначити потрібну частоту виведення через поганий підбір коефіцієнтів регулятора, що і призводить до коливань, що не загасають.

Далі розглянемо приклад правильної роботи програми, для цього потрібно змінити коефіцієнт регулятора – $K_p = 800$ $K_i = 200$ $K_d = 50$. Зміни в порівнянні з минулим прикладом помітні на рисунках 4.3 – 4.4.

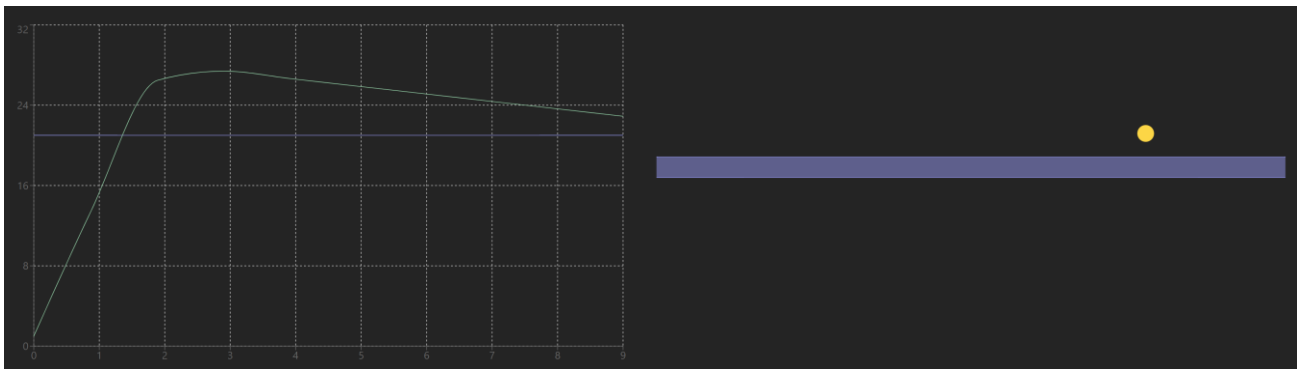


Рисунок 4.3 – Початок другої симуляції

Як видно з рисунку 4.3, початок симуляції проходить так само, як і в попередньому прикладі – PID-регулятор починає набирати обертів, відбувається сплеск і з'являється помилка, але різниця видно вже через кілька секунд. PID-регулятор поступово спускає оберти і тим самим отримує ще одну "помилку", після чого з'ясовує потрібну частоту виведення і підтримує до кінця симулювання, таким чином робот точно їде виставленим маршрутом.

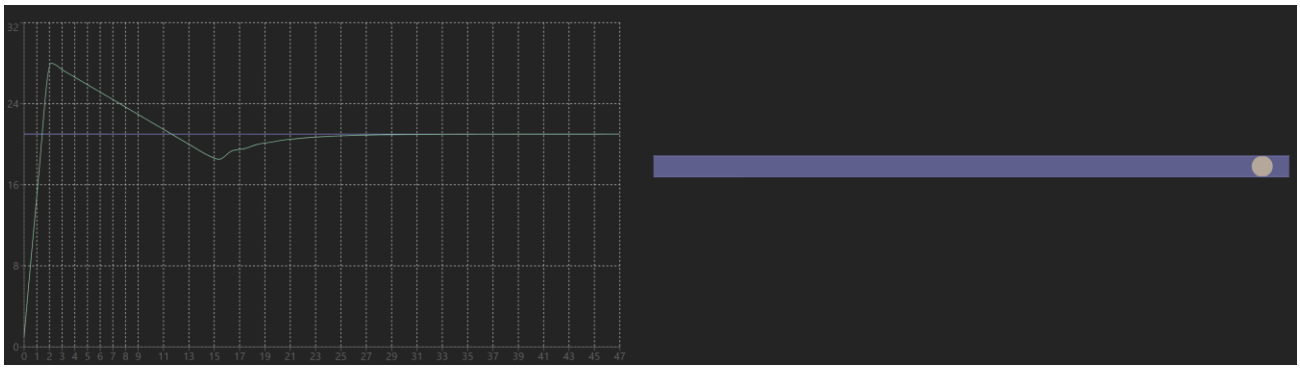


Рисунок 4.4 – Кінець другої симуляції

4.2 Охорона праці

Користувач зобов’язаний дбати про особисту безпеку і здоров`я, а також про безпеку і здоров’я довколишніх при виконанні будь-яких робіт, а також під час перебування на території підприємства.

До роботи на персональному комп’ютері допускають осіб, які пройшли інструктажі з питань охорони праці та пожежної безпеки.

Користувач зобов’язаний:

– виконувати правила внутрішнього трудового розпорядку;

– не допускати за своє робоче місце сторонніх осіб;

– не виконувати вказівок, які суперечать правилам охорони праці та пожежної безпеки;

– знати правила надання домедичної допомоги;

– знати розташування та вміти користуватись первинними засобами пожежогасіння;

– вміти працювати з ПК.

Основні небезпечні та шкідливі виробничі фактори, що можуть впливати на користувача:

– підвищений рівень статичної електрики;

– нерівномірність розподілу яскравості в полі зору;

– підвищена яскравість світлового зображення;

– ураження електричним струмом;

– напруга зору та уваги;

– тривалі статичні навантаження.

У приміщеннях із ПК має бути природне і штучне освітлення. При розміщенні робочих місць необхідно унеможливити пряме засвічування екрана природним освітленням. При природному освітленні слід передбачити наявність сонцезахисних засобів (плівка, жалюзі, штори тощо). Світлові відблиски із клавіатури, екрана та інших частин ПК у напрямку очей користувача

неприпустимі. Основним обладнанням робочого місця є ПК або ноутбук, монітор, клавіатура, маніпулятор, робочий стіл, стілець (крісло).

При розміщенні елементів робочого місця слід враховувати:

– робочу позу користувача;
– простір для розміщення користувача;
– можливість огляду елементів робочого місця;
– можливість огляду простору поза межами робочого місця;
– можливість робити записи, розміщувати на робочому столі документацію та матеріали, які використовує користувач.

Розміщення елементів робочого місця не має заважати рухам та переміщенню для експлуатування ПК. Монітор встановлюють так, щоб відстань від поверхні екрана до очей користувача була 600-700 мм залежно від розміру екрана. Клавіатуру розміщують на робочому або окремому столі на відстані 100-300 мм від краю з боку користувача. Положення клавіатури та кут її нахилу залежить від побажання користувача (як правило, в межах 5-15°). Не допускати хитання клавіатури. Конструкція робочого столу має бути такою, щоб оптимально розмістити на робочій поверхні обладнання, що використовують, з урахуванням кількості, розмірів, конструктивних особливостей і характеру його роботи. Крісло має забезпечувати підтримування раціональної робочої пози під час виконання основних виробничих операцій та можливість зміни пози. Тип робочого крісла обирають залежно від характеру та тривалості роботи.

Раціональна поза користувача:

– ступні розташовані на підлозі або на підставці для ніг;
– стегна зорієнтовані у горизонтальній площині;
– верхні ділянки рук вертикальні;
– кут ліктьового суглоба у межах 70-90°;
– зап’ястя зігнуті під кутом не більше ніж 20°;
– нахил голови у межах 15-20°, а часті її повороти виключені.

Для забезпечення оптимальної робочої пози користувача необхідно: засоби праці, з якими користувач має тривалий або найбільш частий зоровий контакт,

розмістити у центрі зони зорового спостереження та моніторного поля; забезпечити відстань близько 500 мм між найважливішими засобами праці, з якими користувач працює найчастіше. ПК встановлювати на рівній твердій поверхні (столі). Не дозволено встановлювати ПК та оргтехніку на хитких підставках чи на похилій поверхні. ПК не встановлювати впритул до стіни, перегородки тощо. Не допускати загородження вентиляційних отворів ПК сторонніми предметами. Розетка біля ПК має бути в доступному місці, щоб в аварійних випадках можна було своєчасно його відімкнути. Не рекомендовано використовувати подовжувачі. Під час переміщення ПК, периферійних пристроїв витягти вилку живлення з розетки. Не допускати ушкодження чи модифікування шнура живлення. Заборонено ставити важкі речі на шнур живлення, тягнути чи надмірно перегинати його, скручувати та перев’язувати шнур живлення вузлом. ПК під’єднувати до електромережі лише за допомогою справних штепсельних з’єднань та електророзеток заводського виробництва. Штепсельні з’єднання та електророзетки мають бути зі спеціальними контактами для під’єднання нульового захисного провідника. Їхня конструкція має забезпечувати з’єднання нульового захисного провідника раніше, ніж з’єднання фазового та нульового робочого провідників. Порядок роз’єднань при вимкненні має бути зворотнім. Заборонено під’єднувати електрообладнання до звичайної двошнурової електромережі [13].

4.3 Висновки до розділу

У цьому розділі були описані дві симуляції – правильна та неправильна. При неправильній були вказані занадто великі коефіцієнти регулювання, через що PID-регулятор не міг вичислити точну частоту, внаслідок чого з'явилися коливання, що не згасають. При правильному все пройшло в штатному режимі та PID-регулятор зміг налаштувати правильну частоту.

Також були описані правила поведінки під час користування персональним комп'ютером на робочому місці.

ВИСНОВКИ

Аналіз літератури показав важливість використання ПІД регуляторів для управління мобільними платформами, а також описано спосіб управління колісними роботами за допомогою нечіткого управління.

Дослідження також показало важливість використання віртуального моделювання роботів перед їхньою фізичною збіркою. Це допомагає заздалегідь побачити неточності складання, дозволяє відразу розуміти, як платформа поводитиметься на різних видах ґрунту, а також допомагає візуалізувати його поведінку на різних нахилах дороги.

Проаналізовано різні види колісних роботів та обрано чотириколісний варіант для його майбутньої реалізації у програмі.

Були описані всі формули, за допомогою яких PID-регулятор зможе розраховувати число виводу і підтримуватиме керуючий сигнал. Всі вони будуть використовуватися для написання логіки розрахунків у коді програми.

Описано основний функціонал програми, основні частини коду за рахунок яких працює програма. Також було поетапно описано встановлення і створення симуляційного докладання.

Були описані дві симуляції – правильна та неправильна. При неправильній були вказані занадто великі коефіцієнти регулювання, через що PID-регулятор не міг вичислити точну частоту, внаслідок чого з'явилися коливання, що не згасають. При правильному все пройшло в штатному режимі та PID-регулятор зміг налаштувати потрібний керуючий сигнал.

Також були описані правила поведінки під час користування персональним комп'ютером на робочому місці.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

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

2. 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.

3. С.П. Новоселов, Є. Ю. Волков «Завдання автоматичного керування рухом мобільної платформи з застосуванням законів автоматики», матеріали VII-ої Міжнародної конференції «Виробництво & Мехатронні Системи 2023», Харків, 2023 р. – с.: 36 – 39.

4. AplusTopper [Електроний ресурс]: Automation Advantages And Disadvantages | What are The Top Advantages and Disadvantages of Automation? – Режим доступу: www/ URL: https://www.aplustopper.com/automation-advantages-and-disadvantages/

5. В.О. Клебан, В.Г. Парфьонов, А.А. Шалито, «ПОБУДУВАННЯ СИСТЕМИ АВТОМАТИЧНОГО УПРАВЛІННЯ МОБІЛЬНИМ РОБОТОМ НА ОСНОВІ АВТОМАТНОГО ПІДХОДУ», Жовтень 2017.

6. Alexandru BÂRSAN (2019). POSITION CONTROL OF A MOBILE ROBOT THROUGH PID CONTROLLER URL: https://www.researchgate.net/publication/338473586_Position_Control_of_a_Mobile_Robot_through_PID_Controller (the date of access: 13.12.2022)

7. Haocai Luo, Changming Zhang (2022). Application of Fuzzy PID in Wheeled Robot URL: https://www.researchgate.net/publication/366264682_Application_of_Fuzzy_PID_in_Wheeled_Robot (the date of access: 22.11.2022)

8. V-REP - гнучка і масштабована платформа для робомоделювання [Електронний ресурс] – Режим доступу: www/ URL: https://habr.com/ua/articles/383009/

9. LabVIEW - перше знайомство [Електронний ресурс] – Режим доступу: www/ URL: https://habr.com/ua/articles/57859/

10. Robotics/Types of Robots/Wheeld [Електронний ресурс] – Режим доступу: www/ URL: https://en.wikibooks.org/wiki/Robotics/Types_of_Robots/Wheeled

11. PID Controller - Definition and explanations [Електронний ресурс] – Режим доступу: www/ URL: https://www.power-and-beyond.com/pid-controller-definition-and-explanations-a-a69cf75cceed7b82932c0793458f58c9/

12. The PID Controller & Theory Explained [Електронний ресурс] – Режим доступу: www/ URL: https://www.ni.com/en/shop/labview/pid-theory-explained.html

13. Інструкція з охорони праці при роботі на персональному комп’ютері [Електронний ресурс] – Режим доступу: www/ URL: https://services.uteka.ua/ua/publication/zrazky-34-trudovi-vidnosyny-ta-oplata-pratsi-138-instrukciya-po-oxrane-truda-pri-rabote-na-personalnom-kompyutere-obrazec