МАГІСТЕРСЬКА РОБОТА

~~01.09~~. – МР 585 «С». 2022.02.04. 011 ПЗ

ОЛЕКСІЙКО ОЛЕКСАНДР ГРИГОРОВИЧ

2023

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет конструювання та дизайну

УДК 531+62.50

ПОГОДЖЕНО:
Декан факультету
конетруювання та дизайну
_____**Ружило З.В.**
(підпис)
«___» _____ 2023 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ:
Завідувач кафедри конструювання
машин і обладнання
_____ **Ловейкін В.С.**
(підпис)
«___» _____ 2023 р.

МАГІСТЕРСЬКА РОБОТА

на тему: **«ОПТИМІЗАЦІЯ КЕРУВАННЯ ТА СТАБІЛИЗАЦІЯ РУХУ МОБІЛЬНОГО РОБОТА»**

01.09. – МР.585 «С». 2022.02.04. 011 ПЗ

Спеціальність - 133 «Галузеве машинобудування»
Освітня програма - «Машини та обладнання сільськогосподарського виробництва»
Орієнтація освітньої програми - освітньо-наукова

Керівники магістерської роботи:
д.т.н. проф. _____ **Ромасевич Ю.О.**
(підпис)

Виконав: _____ **Олексійко О.Г.**
(підпис)

Київ-2023

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет конструювання та дизайну

ЗАТВЕРДЖУЮ:
Завідувач кафедри конструювання машин і обладнання, д.т.н.

_____ **Ловейкін В.С.**
(підпис)

«___» _____ 20___ р.

ЗАВДАННЯ

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТУ

ЗАРІВНОМУ ОЛЕКСАНДРУ ЮРІЙОВИЧУ

Спеціальність - 133 «Галузеве машинобудування»

Освітня програма - «Машини та обладнання сільськогосподарського виробництва»

Орієнтація освітньої програми - освітньо-наукова

Тема магістерської роботи: «Оптимізація керування та стабілізація руху мобільного робота», затверджена наказом ректора від «04» лютого 2022 р. № 204 «С».

Термін подання завершеної роботи на кафедру 15.05.2023 р.

Вихідні дані до магістерської роботи:

1. Вебсайти Укрпатент, патентне відомство Німеччини, Google Академію, Scopus, Web of Science.
2. Технічні характеристики Keyestudio Self-balancing Car (KS0193).
3. Функціональні можливості Arduino IDE.
4. Правила користування пристроєм Keyestudio Self-balancing Car.

Перелік питань, що підлягають дослідженню:

1. Провести аналіз технічних документів та науковик публікацій за темою оптимізації керування та стабілізації руху мобільного робота.
2. Зібрати та описати апаратну частину приструю в даній моделі.
3. Описати середовище в якому розробляється код, написати та розробити код для Self-balancing Car.
4. Правила безпеки та поводження з пристроєм Keyestudio Self-balancing Car.

Перелік графічних документів: Використовується 12 рисунків та 8 таблиць.

Дата видачі завдання: 27.10.2021 р.

Керівник магістерської роботи:
д.т.н. проф. _____ **Ромасевич Ю.О.**
(підпис)

Завдання прийняв до виконання: _____ **Олексійко О.Г.**
(підпис)

ЗМІСТ

ВСТУП

У сучасному світі мобільні роботи стають все більш важливими в різних галузях людської діяльності, включаючи промисловість, медицину, транспорт, аграрний сектор та інші. Оптимізація керування та стабілізація руху мобільних роботів є ключовими аспектами, які впливають на їх ефективність та безпеку функціонування. Аналіз технічних документів та наукових публікацій з цієї теми виявив, що існує певна кількість патентів, що стосуються керування та стабілізації руху мобільних роботів.

Однак, було помічено, що більшість патентів мають обмежену кількість особливостей та не враховують повністю потенціал оптимізації цих процесів. Водночас, необхідно врахувати, що кількість патентів в даній області була обмеженою, ймовірно, через швидкі технологічні зміни та постійні дослідження у цій галузі.

Окрім того, проведено огляд використовуваних алгоритмів та методів, що забезпечують оптимальне керування та стабілізацію руху мобільного робота. Дослідження спрямоване на оцінку ефективності та придатності цих алгоритмів для конкретного пристрою, а також їх можливостей для оптимізації руху та підвищення стабільності. Програмне забезпечення безперечно відіграє важливу роль у забезпеченні ефективності та надійності робота, а також у забезпеченні його можливостей для виконання різноманітних завдань. Починаючи з аналізу вимог до програмного забезпечення, розробляється структура програми, визначаються основні модулі та їх функціональні можливості. Для забезпечення ефективності та швидкої дії робота важливо враховувати особливості апаратної платформи та використовувати оптимізовані алгоритми керування та стабілізації.

Оптимізація керування та стабілізація руху робота передбачає не лише розробку програмної частини та апаратних рішень, але й усвідомлення правильного використання пристрою з метою досягнення найкращих результатів. Пояснюється необхідність виконання певних заходів безпеки, таких

як захисне обладнання, правильне розміщення пристрою у просторі та дотримання встановлених обмежень щодо ваги та розмірів робота.

РЕФЕРАТ

Актуальність теми. В сучасному світі робототехніка та автоматизація стають все більш важливими та невід'ємними складовими нашого повсякденного життя. Одним з ключових напрямків в цій галузі є розвиток мобільних роботів - універсальних механізмів, здатних переміщуватися та виконувати завдання у різних середовищах.

Оптимізація керування та стабілізація руху мобільного робота є актуальною проблемою, яка вимагає глибокого аналізу та розробки нових методів для покращення його ефективності та точності. Правильне керування мобільним роботом впливає на його здатність уникати перешкод, навігацію ускладненими середовищами, а також виконання поставлених завдань з високою точністю.

Мета і задачі дослідження. Ця дипломна робота має на меті вивчення, аналіз та оптимізацію методів керування та стабілізації руху мобільного робота з використанням сучасних технологій та алгоритмів. Для досягнення поставленої мети необхідно вирішити такі завдання:

- отримати комплексне розуміння технічних аспектів теми дослідження, а також виявити наявні патентні розробки, які вже існують у відповідній галузі;
- дослідити різні аспекти цієї проблеми, такі як техніка навігації, визначення місцезнаходження, планування маршрутів та інші аспекти, які впливають на ефективність робота;
- оптимізувати стабілізацію гіроборда;
- здійснити покращення у взаємодії гіроборда з блютуз системою для поліпшення балансування під час руху;
- розглянути заходи безпеки: забезпечення належного використання захисного обладнання, коректного розташування пристрою в просторі і дотримання встановлених обмежень щодо ваги та розмірів робота.

Об'єкт дослідження – процес руху мобільного робота та його особливості;

Предмет дослідження – процес оптимізації керування та стабілізація руху мобільного робота;

Методика дослідження - дослідження грунтуються на аналізі технічних аспектів та патентів, дослідженні апаратної та програмної частин пристрою Keyestudio Self-balancing Car та програмного середовище Arduino ID та інших.

Загальна характеристика роботи. Магістерська робота складається з розрахунково-пояснювальної записки загальним обсягом 139 сторінки машинописного тексту, 12 рисунків, 8 таблиць, 22 літературних джерел, 12 додатки.

Публікації:

Стаття в фаховому виданні категорії «Б»: Ю. О. Ромасевич, В. С. Ловейкін, О. Ю. Зарівний, О. Г. Олексійко. Керування рухом перевернутого маятника: розробка установки, ідентифікація системи та синтез оптимального регулятора її руху. Наукові доповіді НУБіП України. № 1 (95), 2022. DOI:10.31548/dopovidi2022.01.016. Додаток З

Тези:

1. Ю. О. Ромасевич, О. Г. Олексійко. Нестійкі засоби індивідуального переміщення: гіроборд і гіроскутер.

Збірник тез доповідей IX Міжнародної науково-технічної конференції «Крамаровські читання» з нагоди 115-ї річниці від дня народження доктора технічних наук, професора, члена-кореспондента ВАСГНІЛ, віцепрезидента УАСГН Крамарова Володимира Савовича (1906-1987) 24-25 лют. 2022 р., м. Київ / МОН України, Національний університет біоресурсів і природокористування України. К.: Видавничий центр НУБіП України, 2022. 279 – 280 с. Додаток И.

2. Ю. О. Ромасевич, О. Г. Олексійко. Кількісний аналіз науково-технічних документів у галузі розробки сегвеїв.

Збірник наукових праць / Вісник студентів факультету конструювання та дизайну Національного університету біоресурсів і природокористування України. – Вип. 10 – К., 2022. 28 – 29 с. Додаток І.

РОЗДІЛ 1
АНАЛІЗ ТЕХНІЧНИХ ДОКУМЕНТІВ ТА НАУКОВИХ ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДОСЛІДЖЕННЯ

1.1 Патентні документи України. Змістовний аналіз

Цей розділ допомагає отримати комплексне розуміння технічних аспектів теми дослідження, а також виявити наявні патентні розробки, які вже існують у відповідній галузі, див. табл. 1.1.

Актуальність даного розділу полягає у зосереджені уваги на схожості продемонстрованих патентів. Перш за все потрібно відзначити, що усі патенти є транспортуючими засобами, які використовуються для перевозки. Найголовнішими з схожестей є самобалансування, динамічність та нестійкість.

Усі ці патенти відображають проблеми та недоліки, які можуть бути властивими конкретним виробам та конструкціям, і вказують на можливості для подальшого вдосконалення.

Наведені винаходи мають свої недоліки, такі як складні конструкції, високі витрати, обмежене використання на нерівних поверхнях, проблеми зарядки акумулятора та обмеженість програмування. Ці недоліки можуть ускладнити експлуатацію, ремонт та знизити функціональні можливості велосипедів.

Таблиця 1.1 — Розгляд патентного відомства України.

Посилання	Формула	Технічний результат
[1]	У велосипедах з одним колесом існує моносфера, де кривошип зв'язаний з колесом таким чином, що вісь обертання колеса є самим кривошипом. Цей велосипед рухається за допомогою фізичної сили людини, яка використовує педалі на моносфері. Моносфера містить надувне колесо, розташоване в обоймі сфери, пружинну підвіску, пов'язану з штоком, діелектричну запчастину, яка використовується для закріплення шарнірного з'єднання, а також сідло, на якому сидить водій. Додатково, на вершині пружинної підвіски розміщений лінійний двигун, а на корпусі є датчик.	Результатом корисної моделі є креслення,яке зображене на фіг.1, на якій відтворено загальний вигляд даної моделі. Моносфера містить основну надувну сферу 2, яка розміщена в обоймі сфери 4, завдяки чому і відбувається переміщення самої моносфери, діелектрична запчастина 3 служить захистом від механічних пошкоджень і травм, саме це і робить її більш безпечною для велосипедиста, пружинна підвіска 5 пов'язана зі штоком 6, на якому закріплене 15 шарнірне з'єднання 8, та сідло 9 на якому сидить сам велосипедист, лінійний двигун 7 розміщений зверху пружинної підвіски, який в свою чергу служить для накопичення кінетичної енергії в пружині з подальшим її вивільненням для подолання перешкод. На корпусі розташований датчик 10, який використовується для виявлення перешкод спереду.
[2]	1. Електровелосипед має ведене та ведуче колеса.У електровелосипеда є рама, м'язовий привід разом з електричним приводом ведучого колеса, який в свою чергу відрізняється тим, що електричний привід складається з джерела електричної енергії у вигляді акумуляторної батареї, яка підключена до безколекторного постійного струму електродвигуна, а також контролера. Електродвигун може бути кріплений до переднього або заднього колеса або обох одночасно. 2. У електровелосипеда згаданого у пункті 1,відмінним є те, що електропривід вже оснащено контролером, який управляється ручкою "газа," що винесена на рукоятку електровелосипеда, який регулює крутний момент даного електродвигуна для управління тягою електровелосипеда.	На кресленні показано винахід корисної моделі, який представляє електровелосипед згідно з формулою, згаданою у пункті 1. Цей електровелосипед має назву "ЕКОСТАР". Він включає раму 1, рульове ведуче колесо 2, на якому розміщений електродвигун, ведуче колесо 3 з декількома передачами та м'язовий ланцюговий привід 4. М'язовий привід складається з кареткового механізму 5, педалей, ведучої зірочки, шатунів, ланцюжка та веденої зірочки (не зображених на кресленні). Також велосипед оснащений електроприводом, який включає акумуляторну батарею та контролер 6, мотор-колесо 7 та ручку "газа" 8.

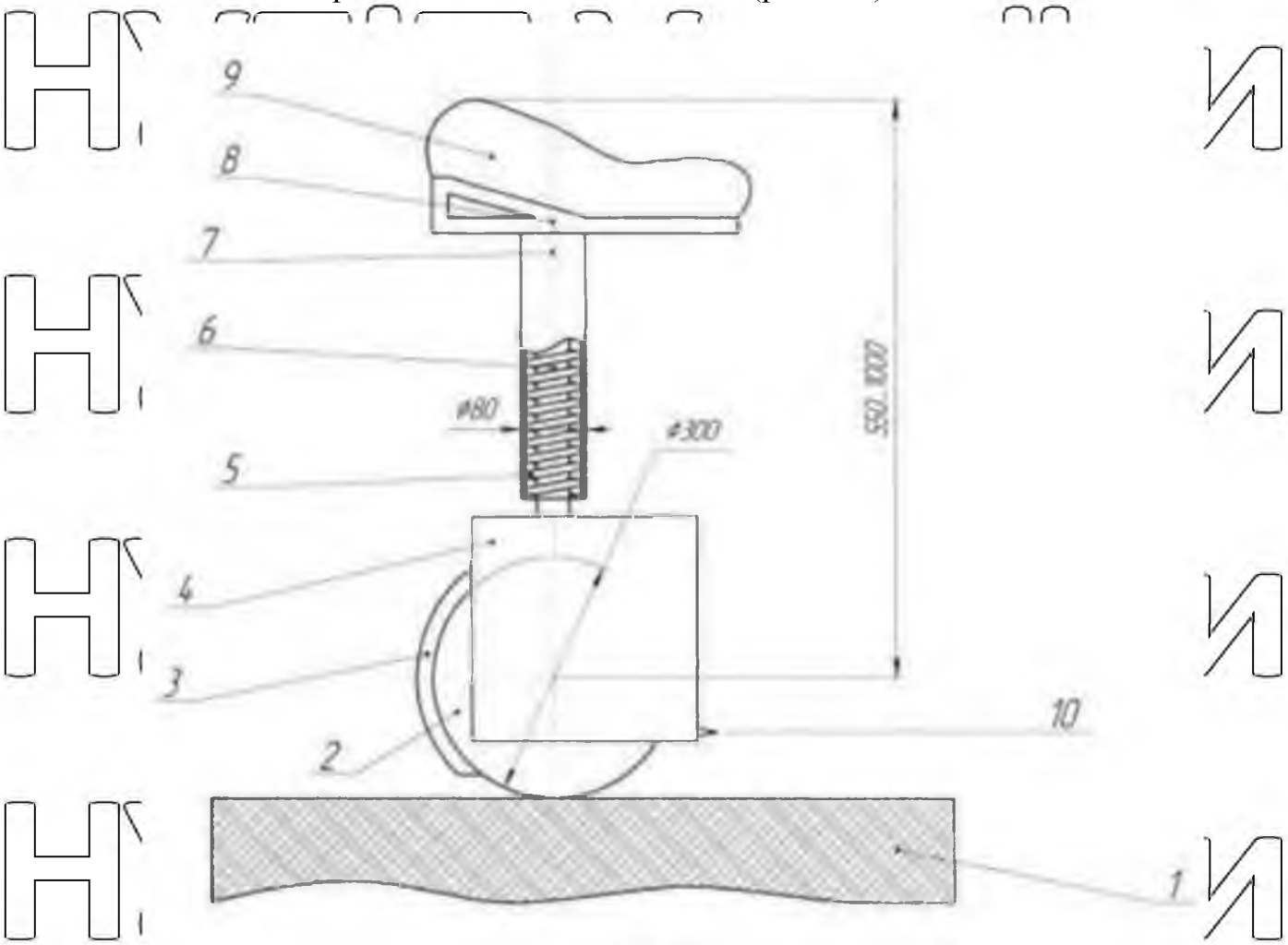
	3. Електровелосипед відрізняється від двох попередніх тим, що електропривід оснащений педальним контролером для регулювання крутного моменту електродвигуна та управління тягою електровелосипеда.	
[3]	Звичайні двоколісні велосипеди використовують привідні механізми, які спеціально призначені для цих типів велосипедів. Основним типом велосипеда в світі є звичайний двоколісний, який приводиться в рух за допомогою кривошипа. Кривошипи на цих велосипедах оснащені педалями, які при натисканні на каретковий механізм спричиняють обертання цього механізму. За допомогою ланцюга кінетична енергія переноситься, пройшовши повністю першу зірочку, і потім переходить до другої зірочки, яка статично прикріплена до заднього колеса. Це стається завдяки наявності механізму вільного ходу. В технічному плані також існують велосипеди з постійним приводом, в яких ведена зірочка постійно жорстко прив'язана до заднього колеса для переміщення велосипеда. Одноколісні велосипеди відрізняються від звичайних тільки в тому, що кривошипи безпосередньо зв'язані з одним колесом, і вісь обертання кривошипів співпадає з віссю обертання заднього колеса.	Задача даної моделі полягала в розробці привідного механізму, призначеного виключно для одноколісних велосипедів, де вісь обертання цього механізму повинна бути паралельна та відмінна від осі обертання веденого колеса. Оскільки більшість одноколісних велосипедів мають зв'язок з єдиним веденим колесом, було важливо змонтувати раму велосипеда на такому колесі, щоб вона займала постійне положення відносно осі обертання веденого колеса під час руху. Ця задача була вирішена шляхом створення нового привідного механізму, особливість якого полягає в використанні виключно веденого колеса, яке може обертатися у спеціальній втулці, жорстко прикріпленій до рами велосипеда, щоб забезпечити механізм вільного ходу для одноколісного велосипеда. Правий та лівий кривошипи повинні бути зв'язані між собою, дозволяючи обертання рами велосипеда, при цьому вісь обертання правого та лівого кривошипів повинна бути паралельна осі обертання веденого колеса і знаходитися на відстані, рівній або меншій радіусу веденого колеса. Особливістю привідного механізму є фіксація вісі обертання правого та лівого кривошипів до осі обертання веденого колеса. Привідний механізм має праву передачу, пов'язану з правим кривошипом і веденим колесом, та ліву передачу, пов'язану з лівим кривошипом і веденим колесом, при цьому права та ліва передачі виконані у вигляді силового кінематичного зв'язку, який проходить через ведене колесо.
[4]	Цей електро-байк має раму, до якої прикріплені сидіння, кермо, педалі, важіль, заднє та переднє колесо.	Модель даного винаходу спрямована на розширення функціональних можливостей електробаків та покращення їх

	Переднє колесо є ведучим і має храповий механізм. Також на рамі встановлена шарнірна передня вилка, до якої закріплене переднє ведуче колесо, яке є редукторним мотор-колесом. Що стосується заднього колеса, до рами прикріплена шарнірна задня вилка, на якій закріплене заднє колесо та амортизатор. Переднє ведуче колесо приводиться в рух за допомогою мотор-колеса, до якого також прикріплений амортизатор. Мотор-колесо підключене через перший контролер до акумуляторної батареї, яка закріплена на рамі. Кермо встановлене на осі з можливістю обертання в заданому секторі, а на його нижній частині знаходиться консоль перекидного блока, який підключений через другий контролер до акумуляторної батареї. Також на байку є бампер, обтічник та передній амортизатор, який закріплений з одного кінця до консолі перекидного блока. До консолі перекидного блока шарнірно закріплені важелі з педалями. Через гнучку тягу, перекидний блок, ланцюг та храповий механізм педалі можуть взаємодіяти з пристроєм перемикання передаточного співвідношення. Додатковий ланцюг з'єднує пристрій перемикання передаточного співвідношення з другим храповим механізмом, який встановлений на передньому ведучому редукторному мотор-колесі. Інший кінець ланцюга, через гнучку тягу, з'єднаний зі спіральною пружиною, яка закріплена на осі.	експлуатаційних характеристик шляхом впровадження пристрою перемикання переднього ведучого колеса. Цей пристрій включає редукторне мотор-колесо, яке додається до заднього ведучого колеса разом з мотор-колесом і амортизаторами для забезпечення м'якого їзди на електробайку. Крім того, винахід включає бампер і обтічник для покращення динаміки та стійкості руху електробайку, а також забезпечує можливість регулювання фізичних навантажень для водія, розширює діапазон швидкостей та забезпечує здатність подолати круті підйоми. Це підвищує безпеку та комфорт водія, а також зменшує витрати енергії.
[5]	Ця інтелектуальна система керування призначена для електровелосипеда і включає контролер, який управляє двигуном електровелосипеда. Що робить її відмінною від інших систем, це наявність датчика, ручки газу, акумулятора і датчика, а також модуля зв'язку для передачі даних про рух та стан електровелосипеда та його компонентів. Цей зв'язок також забезпечує обмін даними з мобільним додатком і можливість його дистанційного керування, якщо така	Схема інтелектуальної системи керування електровелосипедом, включаючи контролер, представлена та пояснена на кресленні моделі. Ця система складається з різних електронних модулів з відокремленими функціями, кожен з яких відповідає за певне завдання. Ці модулі з'єднані послідовною шиною CAN і передають інформацію на смартфон або пристрій електровелосипеда. Вся система контролюється та керується одноплатним комп'ютером, який

	можливість передбачена комплектацією електровелосипеда. Крім того, система має сигналізацію, світловий контролер для керування потужністю світла, сигналами повороту, зупинки та звуковим сигналом, оскільки цей виріб використовується у дорожньому русі. Також вона має контролер запобігання зіткнень і велокомп'ютер для відображення основних показників системи. Особливістю цієї системи є те, що основні компоненти, включаючи згаданий контролер, зв'язані між собою за допомогою колеса та шини.	базується на ARM архітектурі, щоб працювати як єдиний комплексний механізм.
[6]	Електровелосипед включає в себе раму, кермо, яким можна керувати переднім колесом, заднє ведуче колесо, педалі та вал каретки, які приводяться в рух за допомогою фізичної сили. Він також має електричний привід, який приводить у рух заднє ведуче колесо, а також привід електромотора, який повинен бути підключений до джерела електричної енергії та вимикача, утворюючи замкнутий електричний контур. Особливістю електричного контуру цього електровелосипеда є те, що джерело електричної енергії - акумулятор - додатково обладнано геліо-енергетичною установкою. Ця установка розташована на рамі електровелосипеда і має автоматичні компаратори. Сонячні панелі повністю покривають горизонтальну та вертикальну поверхні рами. Вони накопичують більше енергії, ніж одна панель, яка була б встановлена горизонтально, і разом з оборотною електричною машиною генерують більше енергії та запасу приводу для електровелосипеда.	Розглянута задача полягала в тому, щоб створити електровелосипед, який би мав всі стандартні компоненти, такі як рама, кермо для керування переднім колесом, заднє ведуче колесо, педалі та вал каретки, що приводяться в рух за допомогою фізичної сили. У цьому електровелосипеді також присутній електричний привід на задньому ведучому колесі, який повинен бути підключений до джерела електричної енергії та вимикача для створення замкнутого електричного контуру. Однак, особливістю цього електровелосипеда є те, що джерело електричної енергії - акумулятор - додатково обладнано геліоенергетичною установкою. Ця установка розташована на рамі електровелосипеда і має автоматичні компаратори. Сонячні панелі повністю покривають горизонтальну та вертикальну поверхні рами. Вони накопичують більше енергії, ніж одна панель, яка була б встановлена горизонтально, і разом з оборотною електричною машиною генерують більше енергії та запасу приводу для електровелосипеда.

Моносфера [1]

Недоліком пристрою в патенті [1] є складність його конструкції оскільки конструкція являється не стандартною, також одним з не достатків є використанням фізичної сили людини, що безпосередньо залежить від швидкості та вправності їзди на моноколесі (рис. 1.1).



Фіг. 1

Рис. 1.1 — Конструкція моноколеса [1]

Прив'язаність до рівної поверхні є одною з основних правил для даного виду велосипедів, оскільки на даному пристрої все залежить від балансу, тому й, поверхня переважно повинна бути рівною, щоб забезпечити повне використання цього пристрою.

Електровелосипед "ЕКОСТАР" [2].

Недоліком в такого електровелосипеда [2] є саме додавання електроприводу, оскільки виникає досить складна кінематична схема у зв'язку

з валом електродвигуна з обгінною муфтою, проблема виникає з значною втратою частини енергії акумулятора (рис.1.2).

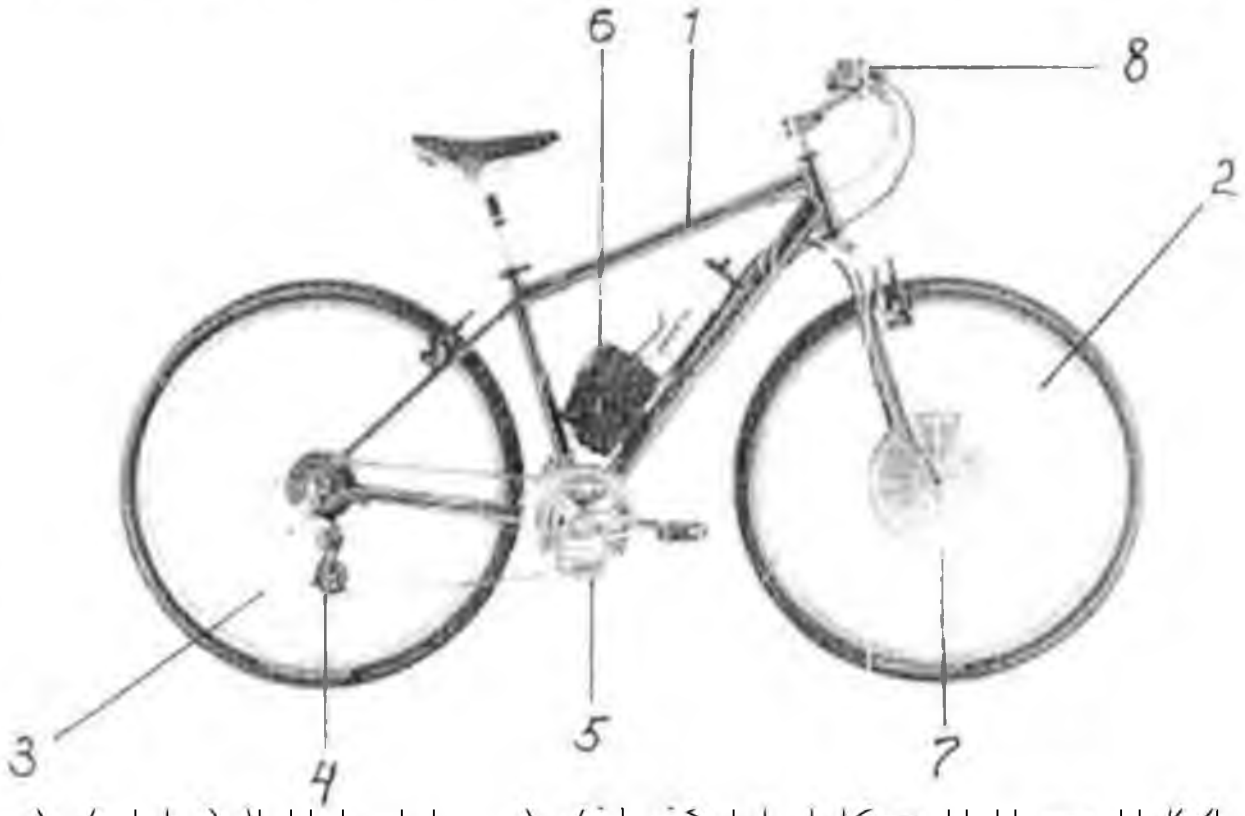


Рис 1.2 — Електровелосипед "Екостар" [2]

Також, одною з недоліком такого електровелосипед дорогостоящим його створення, та знос батарей в акумуляторі, потрібно брати до уваги також його навантаженням, оскільки при досить великому великому навантажені енергія акумулятора значно збільшується та зменшує її експлуатаційні здібності.

Привідний механізм, особливо для одноколісних велосипедів [3].

Проаналізувавши винахід механізму для одноколісних велосипедів [3], замічені певні недоліком в цій моделі є складність даного виробу, оскільки, виріб обумовлений використанням фізичної сили людини, а також гарного балансу на ньому (рис.1.3).

Рис.1.3 — Механізм одноколіного велосипедів [3]

Та основною з проблем є прив'язаність до рівної поверхні, оскільки досить тяжко минати різні перешкоди які можуть зустрітися на шляху цього одноколісного велосипеда, а також, піднімання чи спускання на ньому з пагорба. Також, певним недоліком є не популярність одноколісних велосипедів, тому його створення також можна вважати дорогостоящим.

Електробайк [4]

В даного електро-байка [4] є основний недолік, це використання електрики для заряджання акамулятора, що додає певних незручностей його власнику, оскільки в ньому напріч відсутня можливість використання додаткової енергії для забезпечення початкового руху необхідного прискорення на початку руху, а також підвищеної швидкості для подолання затяжних підйомів на шляху, що в свою чергу, потрібно бути зменшувати на вантаження та підіймати даний виріб вручну, цей недолік додає певних труднощів тому потрібно його використовувати в певних місцях, що обмежує функціональні можливості електро-байка, так збільшує використання енергії акумулятора (рис.1.4).

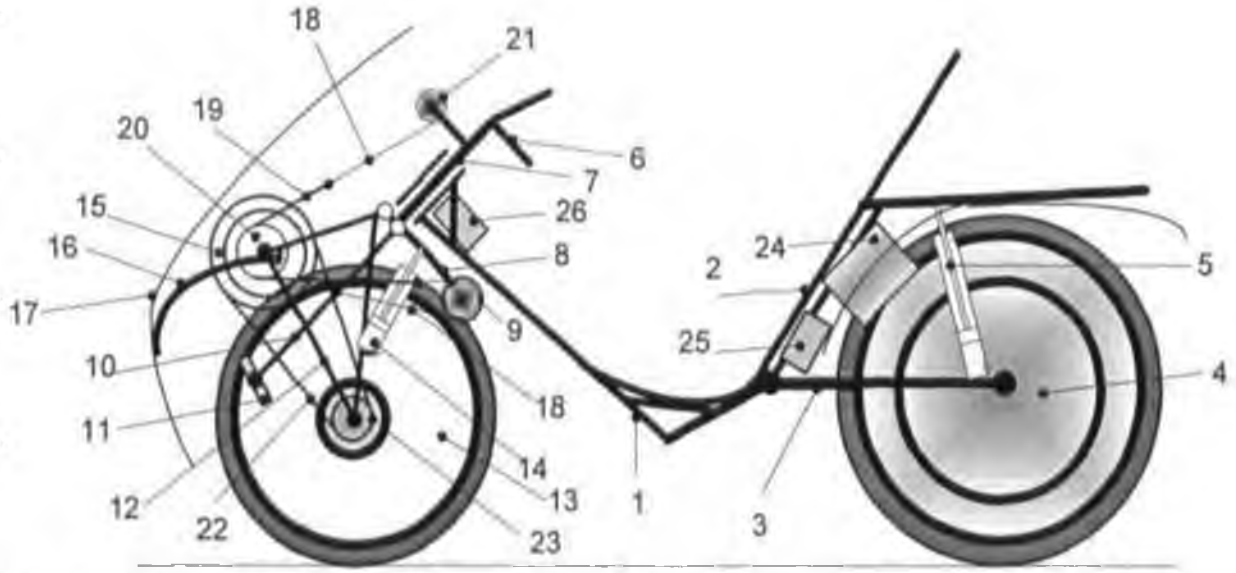


Рис 1.4 — Електробайк [4]

До недоліком цього електро-байка, також потрібно віднести привід який виконаний так що він не має можливості перемикання передаточного співвідношення, оскільки власник цього електро-байка відносить його мотоциклий, недолік який виникає, це відсутність обтічника тому збільшується опір руху і відповідно зменшує швидкість, при проходженні поворотів виникають значні бокові навантаження на колеса.

Інтелектуальна система керування електровелосипедом [5].

Інтелектуальна ~~систем~~ [5] має свої недоліки, оскільки контролер керує виключно електродвигуном, тому програмується виключно на ті характеристики кожного електро-велосипеда які він має, та підходить ця програма. Оскільки програма не являється вікритою і застосування та поширення її в широкі маси затрудняє його масового поширенню в маси (рис.1.5).

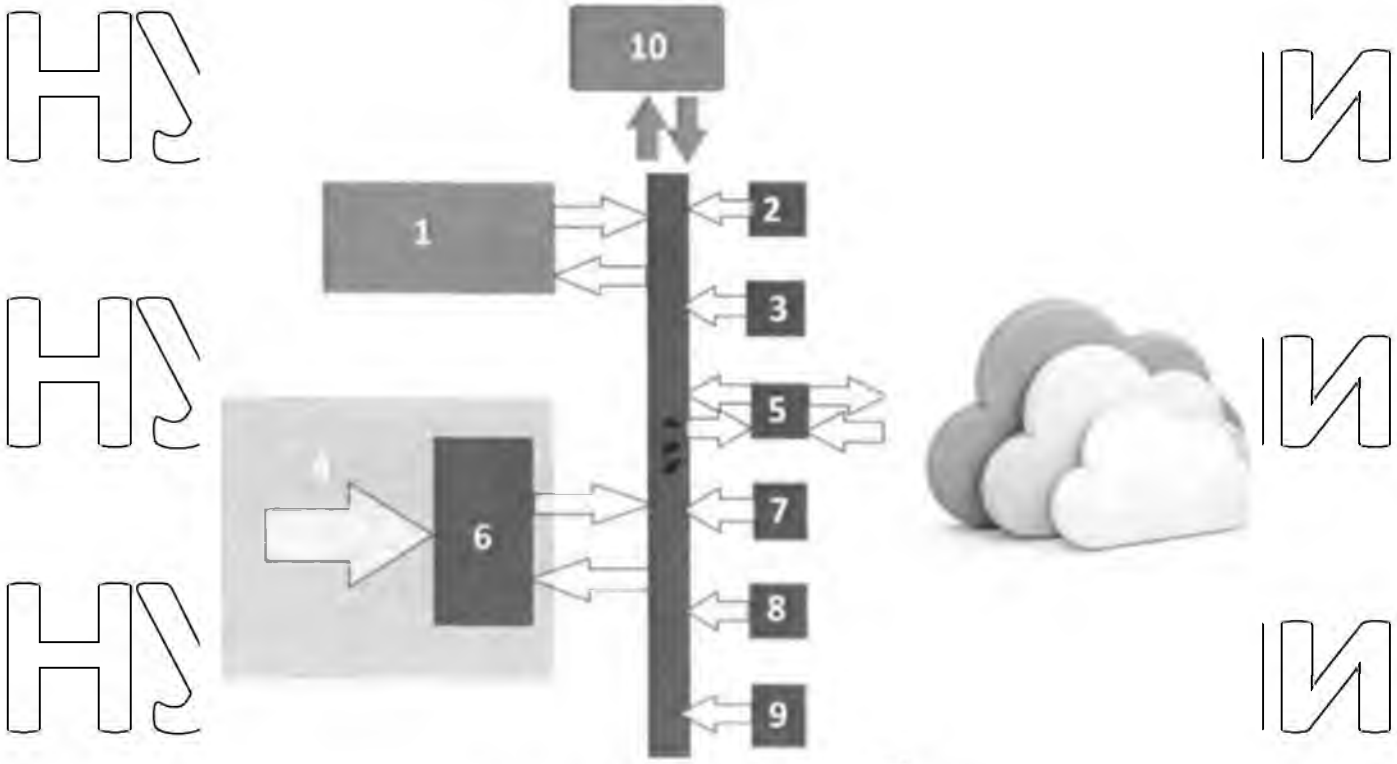


Рис.1.5. Система керування [5]

Також з недоліків, щоб використовувати цю програму в повному обсязі потрібно придбати стороннє обладнання, як то BMS, контролер заряду, різноманітні датчики працюють незалежно один від одного. Потрібно братии до уваги, що додавання в електро-велосипеда додаткового контролера та полеса з шиною, додає додаткових затрат та подальше навантаження на акумулятор, він буде сідати ще швидше оскільки, буде постійне зчитування інформації з вік контролерів та переданням їх до смартфону чи екрану на велосипеді.

Електровелосипед [6].

Недоліком в даній системі електровелосипеда [6] є, наявність джерела електроенергії яке в свою чергу досить обмеженої ємності у вигляді конденсатора, що не дає змоги повністю значно збільшувати запас ходу стандартного оборотного електричного мотора машини. З подвійним електричним шаром який використовується даному патенті, він виключає активне використання електропривода та істотно обмежує експлуатаційні можливості електро-велосипеда (рис.1.6).

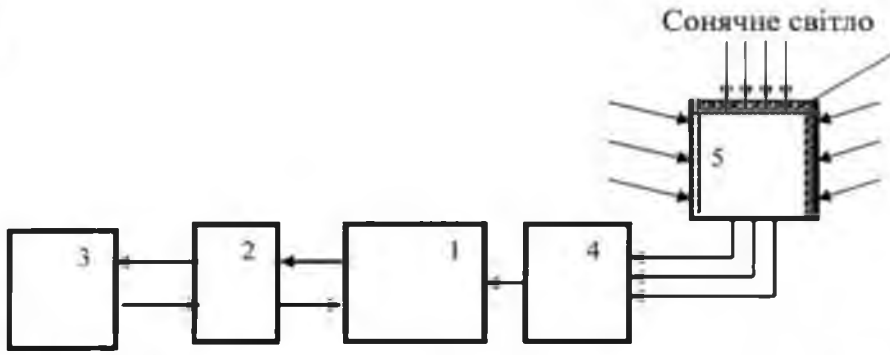


Рис.1.6. — Електровелосипед [6]

Незначний, недостаток це доростоюще обладнання для електро-велосипеда, та додаткове навантаження на велосипед, оскільки потрібно раму горизонтально та вертикально обладнати гелиоенергетичною установкою.

1.2 Патентні документи світу. Кількісний аналіз

На основі кількісного аналізу патентних документів зі світової патентної бази Німеччини та статів Всесвітньої організації інтелектуальної власності (WIPO), було проведено дослідження одноколісних та двоколісних велосипедів. За отриманими результатами, які актуальні станом на 1 лютого 2023 року, можна зробити висновок про інноваційну активність у світі див. табл. 1.2 та 1.3.[7]

Детальну інформацію та звіти щодо даного дослідження можна знайти на веб-сайті Всесвітньої організації інтелектуальної власності (WIPO) за наступним посиланням [8].

Табл. 1.2 — Міжнародна Патентна кваліфікація B62K 3/00

Рік	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
Кількість патентів у світі	373	433	420	561	1133	1136	789	520	442	494	479

Табл. 1.3 — Міжнародна Патентна кваліфікація B62K 1/00

Рік	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
Кількість патентів у світі	131	226	211	238	336	232	189	152	95	87	81

В патентному відомстві Німеччини відбувався великий приріст наукових робіт, на даних графіках було значені виключно патенти які стосувалися моєї наукової роботи. Потрібно виявити, чому відбувся цей науковий стрибок тому і було проведено дослідження цієї теми.

Аналіз проведено щоб виявити актуальність даної теми в світі. Вибраний період з 2012 по 2022 рік, потрібно дізнатися наскільки світ зацікавлений в цьому напрямку, тому для більшої зручності буду розглядати сумірну кількість МПК B62K 1/00 та B62K 3/00.

Як бачимо, на період 2012 року почався ріст патентних заявок, поданих у всьому світі, після найбільшого зниження яке відбулося в 2009 році на 3,9%, однак в 2012 зросла на 9,2% що є найшвидшим зростанням за останні 18 років, як зазначає Всесвітня організація інтелектуальної власності або ВОІВ в своєму щорічному звіті. Основне поясненням цього явища стало значне збільшення кількості заявок, поданих до ВОІВ. В Європі відзначалися змішані тенденції в тому, що стосується подання заявок. Наприклад, Європейське патентне відомство (ЄПВ, +4%), відомства Німеччини (+3,2%) і Сполученого Королівства (СК, +4,4%) зареєстрували зростання [9].

Стосовно 2013 та 2014 будем розглядати разом, оскільки певного зросту та падіння не виявлено, так в 2014 році зменшилось на 13-15 патентних заявок, але це не так істинно впливає на розвиток цього направлення в світі загалом, оскільки на графіку представлена передова країна Європи та Світу [10].

Тенденція до зростання кількості поданих у всьому світі патентних заявок збереглася як 2013 так 2014 році: загальна кількість поданих новаторами заявок склала близько 600 одиниць інтелектуальної власності, в той час як Китай випередив наступні за ним за кількістю заявок Сполучені Штати Америки і Японію разом узяті. Згідно з доповіддю "Всесвітніх показників інтелектуальної власності" 2013-2014 років, щорічними звітами ВОІВ про останні глобальні тенденції у сфері інтелектуальної власності (ІВ). У 2013-2014 роках, кількість зареєстрованих у світі заявок поданих вперше за 20 років, скоротилася на 8,1% стосовно 2014 р. в порівняно з потужним ростом на 2012 -2013 рік. Це зменшення обумовлено різким зменшенням кількості заявок, поданих мешканцями Китаю, за ними слідують Європейського Союзу, Республіка Корея, Німеччина і Туреччина. Порівняно з попереднім роком, кількість заявок на реєстрацію промислових зразків у 2014 році зменшилася в Китаї (-14,4%), Туреччині (-4,5%) і Республіці Корея (-2,3%), але зросла в Німеччині (+6,6%) і Великобританії (+1,3%) [11].

У 2015 році новатори у всьому світі подали загалом близько 800 патентних заявок, що на 7,8% більше, ніж у 2014 році, в зазначається в щорічній доповіді ВОІВ «Світові показники діяльності в галузі інтелектуальної власності». В 2015 році кількість поданих у світі заявок на реєстрацію промислових зразків зросла на 2,3% і нормалізувалася після різкого спаду, зафіксованого в 2014 році внаслідок масштабного зниження темпів подання заявок в Китаї. Зростання кількості заявок пов'язано головним чином з збільшенням темпів подання в Китаї, Республіці Корея і США. В порівнянні з 2014 р. у 2015 р. було відзначено зростання числа заявок у відомствах Республіки Корея (+5,9%) і Китаю (+0,8%), тоді як в Німеччині (-7,5%), Туреччині (-6%) і ЄС (-0,1%) зафіксовано зниження темпів подання [12].

Частка Азії в глобальних поданнях збільшилася з 49,7% в 2006 році до 64,6% в 2016 році, головним чином через швидке зростання економіки в Китаї. Всього в 2016 році новаторами у всьому світі було подано на 8,3% більше, ніж роком раніше, причому ця позитивна динаміка спостерігалась останні сім років, зазначається в щорічній доповіді ВОІВ «Світові показники діяльності в галузі інтелектуальної власності». Кількість зазначених у заявках зразків у всьому світі зросла на 8,3%, переважно в результаті динамічного зростання в Китаї що становить 52% від загальносвітового показника. Далі в списку фігурують Ісламській Республіці Іран (+34,8%), за якою слідують Україна (+17,4%), Китай (+14,3%) і США (+12,1%) [13].

У 2017 році винахідники по всьому світу подали достатню кількість заявок що показує зростання показників вже восьмий рік поспіль. В цілому у 2017 році налічувалося десять провідних агентств, Тільки Китай становив 50,6% світового загального обсягу. ЄС (8,9%) і Республіка Корея (5,4%) взяли відповідно, Друге і третє місця по загальному обсягу заявок, що впали на їх частку це Італії (+14,2%), Іспанії (+23,5%) і СК (+92,1%) зафіксовано впевнений річний приріст зразків у заявках, отриманих у 2017 р. з 10 провідних відомств лише у двох – відомствах Німеччини (–18,5%) і Республіки Корея (–2,6%) – кількість заявлених зразків у 2017 р. зменшилася в порівнянні з 2016 р., що й видно на зазначеному графіку[14].

У 2018 році більше двох третин всіх заявок на реєстрацію патентів, припадає на Азію, загальне зростання попиту на інтелектуальну власності (ІВ) визначалося динамікою в Китаї, тоді як Сполучені Штати зберігали першість за кількістю патентних заявок, поданих на експортних ринках, згідно з щорічною доповіддю Всесвітньої організації інтелектуальної власності (ВОІВ) [15].

У 2019 році розпочалась пандемія COVID-19, таке падіння можна порівняти з фінансовою кризи 2009 року. Оскільки пандемія COVID-19 розпочалась наприкінці року, тому кількість патентних заявок знизилася лише на -3%. В тому році майже 73% всіх поданих у світі заявок на реєстрацію

промислових зразків, що враховуються за кількістю зразків, припадало на п'ять провідних відомств, причому одне відомство Китаю отримало 52,3% від загальносвітового обсягу. Серед десяти провідних відомств найбільший річний приріст кількість промислових зразків, зазначених у заявках зразка 2019 року, зафіксували Туреччина (+9,1%), Сполучене Королівство (С.К., +8,7%), США (+5,7%) і ВІС ЄС (+4,4%), тоді як в Німеччина (-0.9%), Італії (-13,6%) і Франції (–12%) було відзначено різке скорочення цього показника в порівнянні з попереднім роком [16].

Оскільки подовжилась пандемія COVID-19 у 2020 році, але, незважаючи на глобальний економічний спад, спостерігалося стрімке зростання кількості заявок на реєстрацію товарних знаків. У 2020 році рівень патентних заявок, поданих по всьому світу, повернувся в зону зростання близько (+1,6%).

Більше половини всіх поданих у світі заявок на реєстрацію промислових зразків припадали на відомство Китаю, попри жорсткі умови карантину. В 2020 р. у світі було подано досить велика кількість заявок на реєстрацію зазначених у них звіту знизилась на - 2%. У першій десятці відомств впевнений річне зростання кількості зразків у заявках, поданих у 2020 р., відзначили Сполучене Королівство (+9,5%) і Китай (+8,3%), тоді як країни Німеччина (-7,8%), Італії та Франції спостерігалося різке річне зниження аналогічного показника (–18,5% і –16,6% відповідно), як і в минулому році відбувається погіршена ситуація, але не критична [17].

У 2021 році інноваційні підприємства та інноватори по всьому світу подолали виклики, викликані пандемією COVID-19, а глобальні послуги ВОІВ з інтелектуальної власності щодо патентів відносно почало зростати. Азіатські країни зайняли передову позиції: на них припадало 54,1% від усіх заявок, поданих в 2021 році. Для порівняння, у 2011 році ця цифра для регіону становила лише 38,5 відсотка. Тому зростання числа місцевих патентних заявок які лідирують 2021 році, це Китаї (+5,5%) Республіці Корея (+2,5%) і Індії (+3,5%) зумовлена глобальне зростання числа патентних заявок в 2021 році, внаслідок чого доля заявок, поданих в країнах Азії, перевищила дві

третини загальносвітового числа заявок. Серед країн патентна активність в 2021 році знизилася в США (- 1,2%) Японії (-1,7%) та Німеччині (-3,9%).

Якщо брати загаліну картину, зростання патентів за 2021 рік знизилося на 7,54%. 2021 рік став першим роком, коли темпи приросту сповільнилися. До пандемією COVID-19 кілька років поспіль спостерігався рівномірний і стійке зростання [18].

Сайт патентного відомство Німеччини за даними на 01.02.2023 надав певну інформацію стосовно патентів випущених за 2022 рік. За двома таблицями ми бачимо зацікавленість спала незначно оскільки розпочалась світова криза із-за війни яка виникла в Україні, але зацікавленість всього світу не припинилась тому видно незначний спад в 6% [19].

Досить складний період для науки оскільки:

1) Пандемією COVID-19, що досить сильно вдарило по світу в економічному плані та викликало кризу вже в середині 2021 році. Однак науковці в цілому світі продовжили працювати та створювати різні винаходи, пристрої вдосконалювати те що є, та патентувати свої праці в даному напрямку.
2) Продовження пандемією COVID-19, але світ почав відновлюватися після впроваджених карантинів в цілому світі, однак розпочата Війна Росією проти України почала знову загаяти світ в економічну «яму». Тому криза продовжуються вже в 2023 році в цілому світі, що не досить позитивно сприяє розвитку науки в цілому.
3) Відновлення світу після кризи яка розпочалась на прикінці 2019 році, чим довше триває криза тим гірше для науки та світу, тому наразі відбувається «стогнація» даного напрямк та науки в цілому.

Незважаючи на те, що приріст патентів зменшився, не варто думати, що наука увійшла у фазу застою в цілому. Коли пандемія пандемією COVID-19 закінчиться та Війна в Україні, все повернеться до стабільного зростання та економічного процвітання/

1.3 Кількісний аналіз наукових публікацій

Проведений кількісний аналіз наукових публікацій, що стосуються оптимального контролю перевернутого маятника та керування самобалансуючим скутером, з 14 листопада 2022 року до 1 лютого 2023 року, свідчить про значну кількість проведених досліджень в цій області. Використання оптимального контролю для управління перевернутим маятником та самобалансуючим скутером є актуальною темою досліджень, яка привертає великий інтерес у наукових колах. Існує розмаїття підходів та методів, що застосовуються для вирішення задач оптимального контролю цих систем.

Дослідження в цих напрямках є активними і продовжують зростати з часом. Загалом, кількісний аналіз наукових публікацій свідчить про широкий інтерес до оптимального контролю перевернутого маятника та керування самобалансуючим скутером у наукових дослідженнях, що підтверджує актуальність цих тем і потенціал для подальшого розвитку.

Табл. 1.4 — Аналіз Google Academy за публікацією

Табл. 1.5 — Аналіз Google Academy за публікацією

Графік аналізу кількості публікації проведений з 2012 року по 2022 рік. На 2012 рік було зроблено 1930 публікацій на тему «перевернутий маятник оптимальний контроль», однак вже в 2013 році зацікавленість до даної теми зросла тож кількість публікацій зросла 5%. Порівняно 2013 та 2014 рік ріст відбувся на 6%, але в 2015 році ріст трохи призупинився на 3% що свідчать дані по публікаціях за 2015 рік. Стосовно 2016 ріст відбувся на 9% і продовжу зростати на 2017 вже в 8%, на кінці 2018 році становив 9,8% однак в 2019 ріст перестав бути таким стрімким тому становив 5,6%, але він продовжується та само і в 2020 році вже підріс 4,3%. В 2021 році відбувся самий великий приріст публікацій в 10%, що і видну на графіку та на лінії тренда. Не дивлючись на війну яка виникла в світі, науковці були зацікавлені про що свідчать данні Google Academy кількість публікацій на дану тему не змінилась та скала 0%, це добре /оскільки люди продовжують цікавитися темою не дивлячись на кризу [19].

Розглянувши таблицю спостерігається поступовий ріст, наприкінці 2012 році становила 105 публікація, однак в 2013 та 2014 році становив негативне зростання спочатку на 25% потім на 34% в порівняно з 2012 роком. В 2015 році відбувся самий великий приріст з даного графіку в 265% на тему «самобалансування управління скутером» що свідчило досить велику

зацікавленість. Однак з 2016 де відбувся бум на тему публікацій почався поступовий спад до 2019 року який становить в -46%, після такого спаду відбувся незначний ріст 2020 році в 10%, але спад продовжився вже в 2021 який становив -22% в порівняно з 2020 роком. В 2022 році розпочалась світова криза із-за війни, але науковці все ж були зацікавлені даною сферою тому кількість публікацій становила 122 в 2022 році що на 7% [19].

Висновок який ми можем зробити стосовно цих 2 діаграмних таблиць, це те що науковцям цікавить цей напрямок та розвиток даного напрямку, не дивлячись на всі проблеми які виникають в світі вони продовжують працювати та створювати різні криативні пристрої.

1.4 Оглядовий аналіз доцільних та перспективних сфер застосування пристрою

Оглядовий аналіз, гіроскутер/сигвеє - це електричний транспортний засіб, який складається з двох коліс та платформи для стояння, що забезпечує рух за рахунок різниці в швидкості обертання коліс. Гіроскутери часто використовуються як альтернативний вид транспорту в міських умовах, оскільки вони легкі, компактні та екологічні.

Однією з головних переваг гіроскутерів є їх маневреність і простота у використанні. Оскільки користувачі керують рухом за допомогою рівноваги свого тіла, не потрібно ніяких додаткових зусиль для прискорення чи гальмування. Крім того, більшість гіроскутерів мають можливість регулювання швидкості, що дозволяє їх використовувати як для повільних переміщень по переповненому місту, так і для швидкого проїзду на відкритих ділянках доріг.

Однак, недоліки також в гіроскутерах є, вони можуть бути нестійкими на нерівних дорогах чи при різких змінах швидкості, що може спричинити небезпеку для користувачів. Також, гіроскутери мають обмежений запас ходу, що може становити проблему для довгих поїздок.

Якщо розглядати в загальному, гіроскутери є цікавим та корисним транспортним засобом для коротких відстаней в міських умовах. Однак, перед використанням гіроскутера, користувачам необхідно дотримуватися правил безпеки та ознайомитися з локальними законами та регуляціями щодо їх використання.

Стосовно преспективного розвитку, проаналізувавши актуальність теми електро-платформи (гіроскутера) виявиляється що напрямок є досить преспективний, оскільки науковці зацікавлені в даному нарямку і це видно на графіках вище які зазначені в пунті (1.3).

Приспективи застосування в майбутньому можуть бути досить значними. З одного боку, ці транспортні засоби можуть стати популярними як альтернатива автомобілям та мотоциклам, зменшуючи транспортну навантаженість на міста і зменшуючи викиди в атмосферу. Крім того, вони можуть бути особливо привабливими для коротких відстаней від дому до роботи чи до місцевих магазинів.

З іншого боку, існує можливість, що гіроскутери стануть більш популярними для розваг та туризму. Вони можуть бути використані для відвідування пам'яток і визначних місць у містах, а також для пересування по території туристичних комплексів. Що також сильно вплине ро розвиток даного напрямку.

Крім того, можливо, що гіроскутери стануть більш продуктивними транспортними засобами для робітників на великих заводах та складах. Їх можна використовувати для швидкої доставки товарів на відстані від станції до складу або між різними ділянками заводу.

З розвитком технологій, можливо, що гіроскутери стануть більш автономними, здатними до взаємодії з іншими транспортними засобами та датчиками, що забезпечує безпеку на дорозі. Вони також можуть стати більш ефективними з точки зору енергоспоживання та пробігу, що зробить їх ще більш привабливими для користувачів.

Таким чином, перспективи застосування гіроскутерів в майбутньому можуть бути досить значними і залежатимуть від розвитку технологій, регуляторних рамок та попиту на такий вид транспорту.

Варто зазначити, що гіроскутери можуть бути використані для забезпечення мобільності людей з обмеженими можливостями, а також для доставки продуктів та медичних препаратів до місць, до яких важко дістатися за допомогою інших транспортних засобів.

Загалом, гіроскутери можуть стати додатковим інструментом у глобальній боротьбі зі забрудненням навколишнього середовища, забезпеченні мобільності та забезпеченні більш ефективного використання транспортної інфраструктури.

Проте, необхідно також враховувати потенційні ризики використання гіроскутерів, такі як можливість травм для користувачів та небезпека на дорогах. Тому необхідно продовжувати роботу над безпекою та розвитком регуляторних рамок для гіроскутерів.

У цілому, перспективи застосування гіроскутерів сигвеє у майбутньому можуть бути досить значними, і вони можуть стати важливим елементом майбутньої транспортної системи, але залежатимуть від розвитку технологій, регуляторних рамок та попиту на такий вид транспорту.

РОЗДІЛ 2
АПАРАТНА ЧАСТИНА ПРИСТРОЮ

2.1 Платформа

Платформа Keyestudio Self-balancing Car (KS0193) - це набір для збірки робота-машинки з функцією автоматичного балансування. Ця платформа складається з двох коліс і працює завдяки мікроконтролера Arduino Uno R3 [21].

Платформа має вбудовані 6-осей гіроскоп MPU6050 та 3-осей акселерометр MPU6050. Ці датчики є досить точними та надійними, що дозволяє роботу підтримувати баланс під час руху. Гіроскоп дозволяє визначати кутову швидкість обертання робота навколо своєї осі, тоді як акселерометр визначає прискорення, з яким рухається робот. Завдяки цим даним, мікроконтролер може аналізувати рух робота та відповідно реагувати на нього, що дозволяє роботу підтримувати баланс під час руху. Також він має два мотора з енкодерами, що дозволяють точно контролювати швидкість та напрямок руху.

Ця платформа має багато можливостей для розширення, так як має вільні порти для додаткових датчиків та модулів. Також вона може керуватися з допомогою Bluetooth керування, який дозволяє змінювати напрямок руху та швидкість.

Платформа Keyestudio Self-balancing Car складається з наступних компонентів:

1. Плата управління: це основний мікроконтролер, який відповідає за управління всіма датчиками та актуаторами робота. Мікроконтролером є Arduino Uno, який має вбудований USB-порт для програмування та налагодження.
2. Двигуни: KS0193 має два DC-мотори з редукторами та вбудованими енкодерами. Дані мотори мають потужність 12 В та швидкість обертання 150-250 обертів за хвилину з редукційним відношенням 1:30.

3. Датчики: KS0193 має вбудовані 6-осей гіроскоп MPU6050 та 3-осей акселерометр MPU6050. Ці датчики допомагають роботу підтримувати баланс під час руху.
4. Датчик Холла: він використовується для вимірювання обертів моторів та забезпечення контролю швидкості та напрямку руху
5. Колеса: KS0193 має два колеса з резиновими покришками, які забезпечують рух робота.
6. Батарея: KS0193 поставляється з літій-іонною батареєю 11.1V 1800mAh, яка забезпечує живлення робота.
7. Корпус: KS0193 має корпус з акрилу, який складається зі збірних деталей та кріплень. Цей корпус захищає всі компоненти робота від пошкоджень та дозволяє збирати та розбирати робота за потреби.
8. Інші додаткові компоненти: KS0193 може бути доповнений різними додатковими модулями та датчиками, такими як сенсори відстані, голосові модулі та інші.

Дозволено виконувати наступні функції/операції на даній платформі:

1. Самобалансування: завдяки вбудованим гіроскопу та акселерометру, робот може самостійно балансуватися на двох колесах.
2. Керування: KS0193 може бути керований з допомогою пульта дистанційного керування, який поставляється разом з роботом. Робот також може бути керований за допомогою мобільного додатку через Bluetooth-з'єднання.
3. Рух по прямій та повороти: KS0193 може їхати вперед, назад та повертати. Кут повороту може бути налаштований.
4. Слідування за об'єктом: робот може слідувати за об'єктом, використовуючи додатковий сенсор відстані.
5. Автономна навігація: KS0193 може рухатися автономно за допомогою вбудованих сенсорів, таких як гіроскоп та акселерометр.

6. Навчання програмування: KS0193 може бути використаний для навчання програмування на мові Arduino. Розробник може програмувати робота, використовуючи різні сенсори та актуатори.
7. Експерименти: розробник може використовувати KS0193 для проведення різних експериментів, таких як вимірювання швидкості, визначення відстані та інших параметрів руху.

Тобто, дана платформа може бути корисною платформою для навчання та дослідження робототехніки для початківців та досвідчених розробників. Оскільки, має вбудовані гіроскоп та акселерометр для самобалансування та робот може рухатися вперед, назад, повертати, слідувати за об'єктом та рухатися автономно. Крім того, ця платформа може бути використана для проведення різних експериментів та для навчання програмування на мові Arduino. Також добре продуманою та легко збирається з компонентів, що дозволяє розробникам швидко розпочати роботу з платформою. Крім того, вона є доступною за ціною та має гнучкі можливості для подальшого розвитку та модифікації.

Тому, вона є цікавою та корисною платформою для тих, хто бажає навчитися програмуванню роботів та робототехніки. Ця платформа дозволяє створювати самобалансуючі роботи на базі Arduino з можливістю керування за допомогою пульта дистанційного керування або мобільного додатку через Bluetooth-з'єднання.

2.2 Приводи

Кінематична схема приводів коліс описує, як рухається робот та як приводяться в рух його колеса. Для KS0193 Keyestudio Self-balancing Car, кінематична схема приводів коліс виглядає наступним чином:

- Робот має два колеса, кожне з яких приводиться в рух окремим мотором з енкодером. Мотори з енкодером дозволяють точно визначати кількість обертів колеса та його швидкість.

- Для керування роботом, використовується система самобалансування з використанням гіроскопа та акселерометра. Рух робота керується за допомогою двох моторів, які забезпечують рух вперед або назад та повороти на місці.
- Таким чином, кінематична схема приводів коліс KS0193 Keyestudio Self-balancing Car є досить простою та зрозумілою, що дозволяє легко програмувати рух робота та керувати ним за допомогою пульта дистанційного керування або мобільного додатку.

До кінематичної схеми приводів коліс також можна віднести систему розподілу потужності між моторами, яка забезпечує рівномірний рух робота та його стабільність. Крім того, платформа має декілька датчиків, таких як датчик відстані та датчик світла, які можуть використовуватись для програмування робота та його взаємодії з навколишнім середовищем.

Описана кінематична схема приводів коліс дозволяє KS0193 Keyestudio Self-balancing Car рухатись у різні напрямки, повороти на місці, підтримувати стабільність та самобалансування, що робить його досить універсальним для використання у різних сферах, таких як розваги, освіта, наука тощо. Крім того, простота та зрозумілість кінематичної схеми приводів коліс дозволяє легко програмувати рух робота та налаштовувати його під різні задачі.

До кінематичної схеми приводів коліс також можна віднести систему розподілу потужності між моторами, яка забезпечує рівномірний рух робота та його стабільність. Крім того, платформа має декілька датчиків, таких як датчик відстані та датчик світла, які можуть використовуватись для програмування робота та його взаємодії з навколишнім середовищем.

Описана кінематична схема приводів коліс дозволяє KS0193 Keyestudio Self-balancing Car рухатись у різні напрямки, повороти на місці, підтримувати стабільність та самобалансування, що робить його досить універсальним для використання у різних сферах, таких як розваги, освіта, наука тощо. Крім того, простота та зрозумілість кінематичної схеми приводів коліс дозволяє легко програмувати рух робота та налаштовувати його під різні задачі.

GM37-520 Мотор-редуктор постійного струму з датчиком Холла - це електродвигун-редуктор з вбудованим датчиком Холла, який дозволяє вимірювати оберти вала. Цей мотор-редуктор може працювати з напругою живлення 12 В постійного струму і має тримачі типу L, що дозволяє легко кріпити його до різних поверхонь. (Рис.2.7)

Рис.2.1 — GM37-520 Мотор-редуктор постійного струму з датчиком Холла, 12 В, з тримачами типу L

Параметри двигунів:

- Діаметр: 25 мм
- Висота: 45 мм
- Гнучкий вал для з'єднання з колесом
- Максимальна напруга живлення: 9 - 12 В
- Коефіцієнт уповільнення: 30:1
- Струм холостого ходу: не більше 100 мА
- Оберти холостого ходу: 247 об/хв.
- Крутний момент: 1,4 кг/см

- Обороти: 160 об/хв
- Середній споживаний струм: не більше 450 мА
- Момент утримання: 5,5 кг/см
- Струм у загальмованому режимі: 2,4 А
- Довжина редуктора: 22 мм
- Тип двигуна: DC з двома фазами

GM37-520 Мотор-редуктор постійного струму з датчиком Холла та тримачами типу L може бути використаний в різноманітних проектах, що потребують зменшення швидкості руху та вимірювання обертів. Цей мотор-редуктор є доволі потужним і гнучким інструментом, який може використовуватись у робототехніці, автоматизації та інших галузях.

2.3 Датчики

Датчик - це пристрій, який вимірює певний параметр та перетворює його на сигнал, який може бути використаний для подальшої обробки або керування системою. Датчики дозволяють отримувати дані, які можуть бути використані для керування системами, контролю якості, діагностики та моніторингу.

Датчики використовуються для визначення кута нахилу, руху, відстані та швидкості для забезпечення стійкості і руху самобалансуючої машини. Це дозволяє автоматизувати процеси виробництва та забезпечувати ефективну роботу устаткування.

1. Датчик 6-осевого гіроскопа MPU-6050: він вимірює прискорення та обертання в трьох вимірах (гіроскопічні кути руху), дозволяючи здійснювати стабілізацію платформи та керувати рухом машини.
2. Датчик 3-осевого акселерометра MMA7361: він вимірює прискорення в трьох вимірах (вперед-назад, вправо-вліво та вгору-вниз), дозволяючи визначати нахил платформи.
3. Датчик Холла: він використовується для вимірювання обертів моторів та забезпечення контролю швидкості та напрямку руху.

Датчик MPU-6050 є 6-осевим гіроскопом та жирометром, який може вимірювати прискорення та обертання в трьох вимірах (осі X, Y та Z). Цей датчик здатний забезпечувати точне визначення кутів нахилу, руху та орієнтації для забезпечення стійкості та руху самобалансуючої машини.

Тобто, він вимірює рух по трьом вісям з допомогою трьох обертових гіроскопів, які визначають зміну кутової швидкості тіла вздовж кожної з осей. Отримана інформація дозволяє здійснювати стабілізацію платформи та керувати рухом машини.

В основі роботи датчика лежить принцип користування технології MEMS (Micro-Electro-Mechanical Systems), яка використовується в багатьох сучасних електронних пристроях. Датчик містить мікроелектромеханічні компоненти, які взаємодіють з зовнішнім середовищем, і конвертують ці взаємодії в електричні сигнали. За допомогою цих сигналів, MPU-6050 може визначати прискорення та обертання тіла в трьох вимірах.

Цей датчик забезпечує високу точність вимірювання та швидкий час реакції, що робить його ідеальним варіантом для використання в самобалансуючих системах, таких як роботи, дрони, самобалансуючі машини, тощо. Тому, він має одним з найбільш поширених і доступних гіроскопів для використання в робототехніці та інших пристроях з механічною стабілізацією.

Датчик MMA7361 є 3-осевим акселерометром, який використовується для вимірювання прискорення в трьох вимірах (осі x, y та z). Він виявляє силу, яка діє на нього у напрямку, що перпендикулярний до площини датчика, і генерує відповідний електричний сигнал.

Датчик має різні режими роботи та рівні чутливості, які можна налаштовувати. Він має вбудовану підтримку для калібрування, що дозволяє забезпечувати більш точні вимірювання прискорення. Датчик забезпечує широкий діапазон вимірювання прискорення, від +/- 1,5 g до +/- 6 g.

Датчик MMA7361 можна використовувати в різних застосуваннях, таких як контроль руху, стеження за рухом об'єктів, вимірювання вібрацій тощо. Він може бути підключений до різних пристроїв, таких як

мікроконтролери або вбудовані системи, що дозволяє використовувати його для реалізації різних проектів, які потребують вимірювання прискорення в трьох вимірах.

Датчик Холла, що використовується в в цій платформі, є безконтактним датчиком, який вимірює магнітне поле. Він базується на ефекті Холла - з'явлення електричного струму у провіднику, який рухається в магнітному полі.

Він використовується для вимірювання кута нахилу платформи, на якій розташована машинка. Складається безпосередньо з магнітного датчика та електронного модуля, який обробляє сигнали з датчика та передає їх до мікроконтролера.

Коли машинка нахиляється, магніт у датчику зміщується, що змінює магнітне поле. Датчик Холла вимірює зміну магнітного поля та генерує відповідний електричний сигнал, який передається до електронного модуля. Електронний модуль обробляє сигнал та перетворює його в відповідний кут нахилу платформи.

Цей кут нахилу потім використовується для керування рухом машинки. Мікроконтролер, який керує рухом машинки, отримує сигнал з датчика Холла та виконує відповідні дії, щоб зберегти машинку у стабільному положенні. Датчик Холла є важливим компонентом в системі стабілізації та керування рухом машинки, що дозволяє їй рухатися з більшою точністю та стабільністю.

2.4 Мікроконтролер

Можливість поєднати цей щит із платою керування UNO R3, щоб створити балансувальну машину на основі платформи розробки Arduino.(Рис. 2.2)

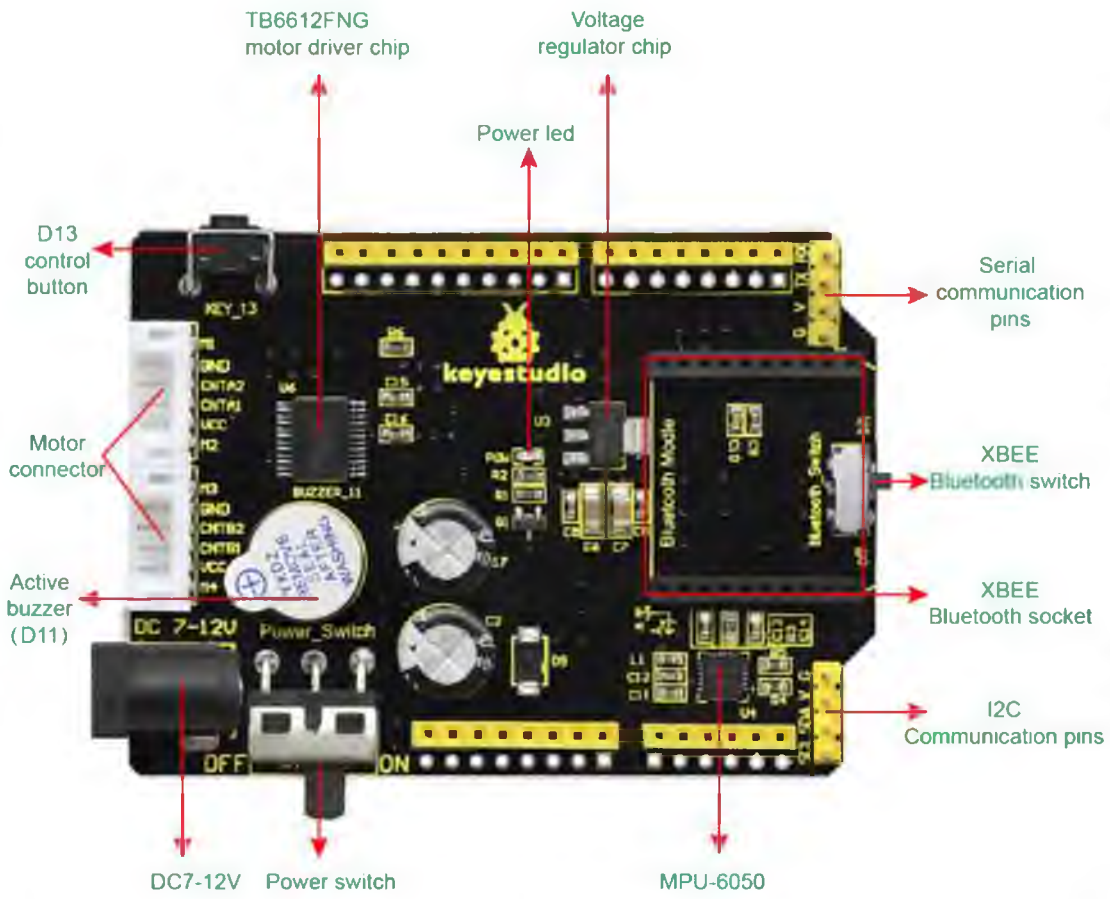


Рис. 2.2. Keyestudio Balance Shield V3

Щит постачається з роз'ємом XBEE Bluetooth, тому ви можете підключити Bluetooth APP, щоб легко керувати машиною-балансиром. За допомогою APP ви можете вибрати як режим керування клавішами, так і режим керування гравітацією, крім того, ви також можете регулювати кут балансу та змінювати параметри PID.

Щит також сумісний із двигуном щітки з постійним магнітом 12 В із коефіцієнтом зменшення 1:30 і зафіксованим крутним моментом 7,5 кг.см.

Після випуску металевої шестерні він має потужнішу потужність, ніж звичайні двигуни. Він має низьке енергоспоживання, високий крутний момент і швидкий старт і т.д.

Для легкого налаштування цинток балансирної машини містить перемикач живлення, комунікаційний перемикач XBEE, вбудований активний

зумер (керування D11), пристрій керування кнопкою D13, послідовний зв'язок і комунікаційний контакт I2C.

Технічні деталі

- Робоча напруга: DC 7-12V
- Робочий струм: 800 мА
- Сумісний з платою UNO R3
- Відповідає ROHS
- Розміри: 80мм*60мм*28мм
- Вага: 28,6 г

Keyestudio Uno R3 — це плата мікроконтролера на основі ATmega328, повністю сумісна з ARDUINO UNO REV3. (Рис 2.3)

Рис 2.3 — Плата контролера Keyestudio UNO R3 REV3

Він має 14 цифрових входів/виходів (6 з яких можна використовувати як ШІМ-виходи), 6 аналогових входів, кварцовий кристал 16 МГц, USB-з'єднання, роз'єм живлення, 2 роз'єми ICSP і кнопку скидання. Містить все необхідне для підтримки мікроконтролера; просто підключіть його до

комп'ютера за допомогою кабелю USB або живіть його за допомогою адаптера змінного струму в постійний струм або акумулятора, щоб почати табл.2.1..

Зауважте, що два заголовки ICSP окремо використовуються для програмування мікропрограми для ATMEGA16U2-MU та ATMEGA328P-PU, але загалом дві мікросхеми запрограмовано добре.

Uno R3 відрізняється від усіх попередніх плат тим, що не використовує чіп драйвера FTDI USB-to-serial. Натомість він містить Atmega16U2, запрограмований як USB-послідовний перетворювач. Більш точна характеристика наведена в табл.2.1..

Табл. 2.1 — Технічні характеристики

Мікроконтролер	ATmega328P-PU
Робоча напруга	5В
Вхідна напруга (рекомендовано)	7-12В
Цифрові контакти введення/виведення	14 (з яких 6 забезпечують вихід ШІМ)
Виводи цифрового входу/виводу ШІМ	6 (D3, D5, D6, D9, D10, D11)
Аналогові входи	6 (A0-A5)
Постійний струм на контакт введення/виведення	20 мА
Постійний струм для контакту 3,3 В	50 мА
Флеш-пам'ять	32 КБ (ATmega328), з яких 0,5 КБ використовується завантажувачем
SRAM	2 Кб (ATmega328P-PU)
EEPROM	1 КБ (ATmega328P-PU)
Тактова швидкість	16 МГц
LED_BUILTIN	D13

Розпиновка (англ. pinout) — це опис того, які контакти (піни) з якими функціями знаходяться на електронному пристрої, наприклад, на мікроконтролерній платі.

У разі Keyestudio UNO R3 REV3 розпиновка описує те, які контакти плати призначені для підключення додаткових модулів, сенсорів, інших пристроїв або для взаємодії з платою в цілому (Рис. 2.4.).

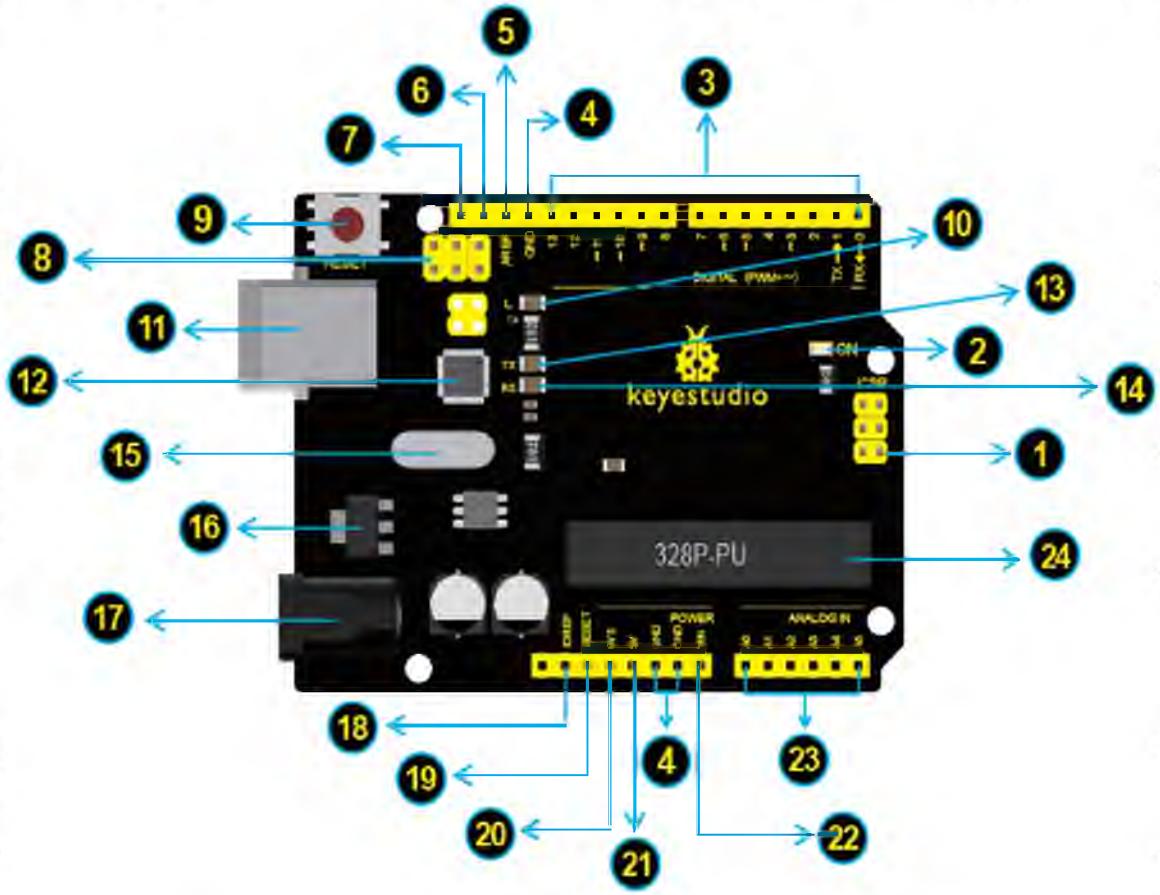


Рис. 2.4 — Інтерфейси елементів і контактів

Розпиновка Keyestudio UNO R3 REV3 описує 14 цифрових входів / виходів, 6 аналогових входів, живлення та інші контакти, що доступні для використання на цій платі. Це дає можливість розширювати функціональні можливості плати за допомогою підключення різноманітних модулів та додаткових компонентів. Більш детальна інформація в табл. 2.2.

Табл. 2.2. — пояснення до рис.2.4.

1	Заголовок ICSP (In-Circuit Serial Programming). У більшості випадків ICSP — це AVR， заголовок мікропрограми Arduino, що складається з MOSI, MISO, SCK, RESET, VCC і GND. Його часто називають SPI (послідовний периферійний інтерфейс) і можна вважати «розширенням» вихідного сигналу. Насправді підлеглі пристрої виводу під хостом шини SPI. При підключенні до ПК запрограмуйте мікропрограму на ATMEGA328P-PU.
2	Світлодіодний індикатор живлення Під час живлення Arduino світлодіодний індикатор означає, що вашу друковану плату правильно увімкнено. Якщо світлодіодний індикатор не горить, підключення неправильне.
3	Цифровий вхід/вихід Arduino UNO має 14 цифрових входів/виходів (з яких 6 можна використовувати як ШІМ-виходи). Ці контакти можуть бути налаштовані як контакти цифрового входу для зчитування логічного значення (0 або 1).Або використовується як цифровий вихідний штифт для управління різними модулями, такими як світлодіод, реле тощо. Штифт з позначкою «～» можна використовувати для генерації ШІМ.
4	GND (роз'єми контактів заземлення) Використовується для заземлення ланцюга
5	AREF Опорна напруга (0-5 В) для аналогових входів. Використовується з analogReference() .

6	ПДР контакт IIC
7	SCL контакт IIC
8	Заголовок ICSP (In-Circuit Serial Programming). У більшості випадків ICSP — це AVR， заголовок мікропрограми Arduino, що складається з MOSI, MISO, SCK, RESET, VCC і GND. Підключений до ATMEGA 16U2-MU. При підключенні до ПК запрограмуйте прошивку на ATMEGA 16U2-MU.
9	Кнопка RESET Ви можете скинути плату Arduino, наприклад, запустити програму з початкового стану. Ви можете скористатися кнопкою RESET.
10	Світлодіод D13 Існує вбудований світлодіод, керований цифровим виводом 13. Коли на виводі встановлено ВИСОКЕ значення, світлодіод світиться, коли на виводі є НИЗЬКЕ, він вимкнений.
11	Підключення USB Плату Arduino можна живити через роз'єм USB. Все, що вам потрібно зробити, це підключити USB-порт до ПК за допомогою USB-кабелю.
12	АТМЕГА 16У2-МУ USB на послідовний чіп, може перетворювати сигнал USB у сигнал послідовного порту.

13	TX LED На борту ви можете знайти мітку: TX (передавання) Коли плата Arduino спілкується через послідовний порт, надішліть повідомлення, індикатор TX блимає.
14	RX LED На борту ви можете знайти мітку: RX(receive) Коли плата Arduino спілкується через послідовний порт, приймає повідомлення, індикатор RX блимає.
15	Кристалічний осцилятор Допомога Arduino впоратися з проблемами часу. Як Arduino обчислює час? за допомогою кристалічного генератора. Число, надруковане на верхній частині кристала Arduino, — 16.000 H 9 H. Він говорить нам, що частота становить 16 000 000 герц або 16 МГц.
16	Регулятор напруги Для контролю напруги, що подається на плату Arduino, а також для стабілізації напруги постійного струму, що використовується процесором та іншими компонентами. Перетворіть зовнішню вхідну напругу DC7-12V у DC 5V, а потім перемкніть DC 5V на процесор та інші компоненти.
17	Гніздо живлення постійного струму Плата Arduino може бути забезпечена зовнішнім джерелом живлення DC7-12V від гнізда живлення постійного струму.
18	IOREF Використовується для налаштування робочої напруги мікроконтролерів. Використовуйте менше.

19	RESET Заголовок Підключіть зовнішню кнопку для скидання плати. Функція така ж, як кнопка скидання (позначена 9)
20	Штифт живлення 3V3 Живлення 3,3 вольта генерується бортовим регулятором. Максимальний споживаний струм становить 50 мА.
21	Штифт живлення 5В Забезпечує вихідну напругу 5 В
22	прийти Ви можете подавати зовнішнє джерело живлення DC7-12V через цей контакт до плати Arduino.
23	Аналогові контакти Плата Arduino UNO має 6 аналогових входів, позначених від A0 до A5. Ці контакти можуть зчитувати сигнал від аналогових датчиків (таких як датчик вологості або датчик температури) і перетворювати його в цифрове значення, яке може читати мікроконтролер. Також можуть використовуватися як цифрові контакти, A0=D14, A1=D15, A2=D16 , A3=D17, A4=D18, A5=D19.
24	Мікроконтролер Кожна плата Arduino має власний мікроконтролер. Ви можете розглядати це як мозок вашої дошки. Основна IC (інтегральна схема) на Arduino трохи відрізняється від панелі. Мікроконтролери зазвичай від ATMEL. Перш ніж завантажити нову програму в Arduino IDE, ви повинні знати, що таке IC на вашій платі. Цю інформацію можна перевірити у верхній частині IC.

Keyestudio Bluetooth Xbee модуль радіо Bluetooth Hc- 06 приймає з щитом Xbee. (Рис.2.4)

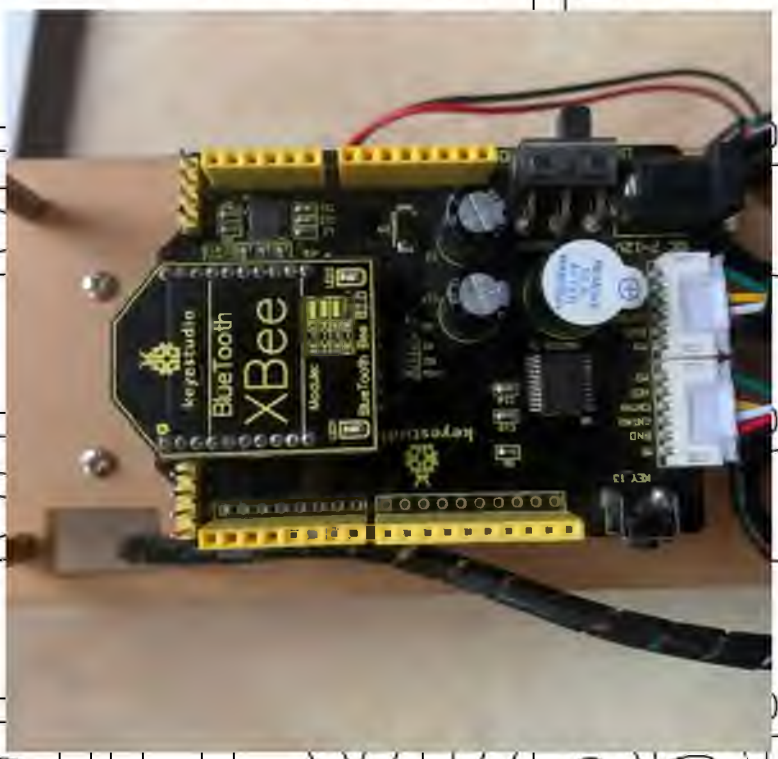


Рис.2.4 — Keyestudio Bluetooth модуль XBee HC-06

Він має особливості компактного розміру, compatible з щитом Xbee, і відповідний для різних 3.3v систем MCU.

Модуль може використовувати AT команді, щоб встановити швидкість у бодах і головний/непривілейований режим. Значення, встановлювані за умовчанням - швидкість у бодах 9600, підрізуючи пароль 1234, непривілейований режим. Це приходить з ефективною on-board антеною. Виставлена антена гарантує кращу сигнальну якість і довше передаючи в дстань. Прозорий послідовний порт може бути використаний, щоб розташуватися пара с з різними адаптерами Bluetooth, телефони Bluetooth.

Гуманізований дизайн пропонує зручність для вторинного розвитку. Після випробування, модуль, відомо, відповідний у використанні з усіма адаптерами Bluetooth на ринку (ПК і телефони з Bluetooth).

Специфікація:

- Гнучко напруга: 3.3v (напруга вище за 3.3v заборонена. Це пошкодить модуль)

- Bluetooth, що підрізує ім'я користувача: Hc- 06
- Bluetooth, що підрізує пароль: 1234
- Задана по умовчанню швидкість у бодах: 9600
- Модифікована швидкість у бодах набуває чинності після module reboot

2.5 Система живлення пристрою

Акумулятори Li - ion Liitokala NCR18650B 3400 mAh – високоякісні і надійні літієві акумулятори без плати захисту і великою реальною місткістю. (Рис.2.5)

Акумулятор NCR18650B відмінно підходить для живлення портативних пристроїв на базі Arduino та інших мікроконтролерів, робоплатформ, ліхтарів, а також заміни акумуляторів в ноутбуках та іншої електроніки.

Особливістю акумуляторів NCR18650B є хімічний склад елемента, який дозволяє запобігти загоряння навіть при механічному пошкодженні. Незахищені Li-Ion акумулятори можна заряджати понад 4.2В і розряджати нижче 2.7В. Порушення цих умов приведе акумулятор до виходу з ладу.

Рис.2.5 — 3 батарейки NCR18650B з блоком

Характеристики:

- Виробник: Liitokala (Китай)
- Модель: NCR18650B
- Тип: Li-ion 18650 без захисту
- Плата захисту: Немає
- Номінальна ємність 3400 мАг (перевірено)
- Макс. струм розряду 4.8 А
- Номінальний струм заряду: 0.5с * (1 625 мА)
- Номінальна напруга: 3.6 В
- Мінімальна напруга: 2.7 В
- Максимальна напруга: 4.2 В
- Метод заряду: CC / CV
- Температурні режими: заряд: 0-45 °C / розряд: від -20 до 60 °C
- Довжина: 67.5 мм
- Діаметр: 18.5 мм
- Вага: 46 г

Блок для трьох батарейок АА, даний тримач призначений для батарейок типу 18650B, також має дріт 15 см з штекер який підключається до плати Arduino.

Для забезпечення стабільної роботи платформи потрібно використовувати ПІД-регулювання (PID control). Для цього можна використовувати мікроконтролер Arduino, який входить до складу комплекту KS0193.

Щоб реалізувати стабілізацію платформи за допомогою ПІД-регулювання, можна використовувати бібліотеку Arduino PID. Ця бібліотека дозволяє легко налаштувати параметри ПІД-регулятора та забезпечити стабільну роботу платформи.

Також можна використовувати готові програми та бібліотеки, що доступні в Інтернеті для самобалансуючих автомобілів на базі Arduino,

наприклад, програму MPU6050_DMP6 з бібліотекою I2Cdev та драйвером MPU6050, яку можна знайти на GitHub.

Стабілізація платформи є складним завданням, оскільки воно вимагає певних значних навичків в даному питанні та досвіду. Тому, рекомендується ознайомитися з теорією ПІД-регулювання та дослідити та провести достатньо часу на налагодження параметрів ПІД-регулятора, щоб досягти стабільної роботи платформи.

Додатково, для стабілізації платформи KS0193 Keyestudio Self-balancing Car можна використовувати такі методики:

1. Калібрування сенсорів: перед початком роботи потрібно калібрувати сенсори гіроскопа та акселерометра, щоб забезпечити точність вимірювань та уникнути похибок в роботі.
2. Регулювання кута нахилу: для забезпечення стабільної роботи платформи потрібно підтримувати кут нахилу платформи в межах певного діапазону. Якщо кут нахилу перевищує максимально допустиме значення, то потрібно виконати корекцію, щоб платформа повернулась до стабільного стану.
3. Регулювання швидкості: для забезпечення стабільної роботи платформи потрібно регулювати швидкість її руху. Якщо швидкість занадто висока або занадто низька, то це може призвести до втрати стабільності.
4. Використання додаткових сенсорів: для поліпшення стабільності платформи можна використовувати додаткові сенсори, такі як магнітометр, щоб враховувати вплив магнітного поля на рух платформи.

Загалом, стабілізація платформи KS0193 Keyestudio Self-balancing Car вимагає технічних знань та досвіду у програмуванні та робототехніці. Рекомендується ознайомитися зі специфікацією комплекту та зв'язатися з фахівцями з питань робототехніки, якщо виникають складнощі з роботою платформи.

РОЗДІЛ 3
ПРОГРАМНА ЧАСТИНА ПРИСТРОЮ

3.1 Опис середовища розробки коду

Arduino ID - це програмне середовище (IDE) для програмування контролерів Arduino. Воно дозволяє створювати, збирати та завантажувати програми для контролерів Arduino. Arduino IDE забезпечує зручний інтерфейс для програмування, який включає текстовий редактор з підсвічуванням синтаксису, компілятор, засіб завантаження програми в контролер та серійний монітор для відладки [22].

Крім того, дана програма має велику популярність в світі, та потужну спільноту користувачів, яка надає безкоштовні бібліотеки та додаткові інструменти для розширення функціональності контролерів Arduino. Arduino IDE підтримує багато різних контролерів Arduino, включаючи Arduino Uno, Arduino Nano, Arduino Mega, Arduino Due та інші.

Основа Arduino IDE полягають у наданні зручного інтерфейсу для програмування та налаштування контролерів Arduino. Основні функції Arduino IDE включають:

1. Текстовий редактор: Arduino IDE має текстовий редактор, який дозволяє користувачам створювати та редагувати програмний код для контролерів Arduino. Редактор має підсвічування синтаксису для полегшення читання та редагування коду.
2. Компілятор: Arduino IDE включає компілятор, який перетворює програмний код, написаний користувачами, у машинний код, який може бути виконаний контролером Arduino.
3. Засіб завантаження програми: Arduino IDE надає засіб для завантаження скомпільованої програми на контролер Arduino. Цей засіб дозволяє користувачам зручно та легко завантажувати програми на контролер.
4. Серійний монітор: Arduino IDE має серійний монітор, який дозволяє користувачам відслідковувати виведення даних з контролера в режимі

реального часу. Цей інструмент дуже корисний при відлагодженні програм.

5. Бібліотеки: Arduino IDE надає безкоштовні бібліотеки, які допомагають користувачам розширювати функціональність контролерів Arduino. Бібліотеки містять код, який може бути використаний для різних завдань, таких як робота з датчиками, керування сервоприводами тощо.

Ці основні функції Arduino IDE роблять цю програму дуже корисною для розробників, які працюють з контролерами Arduino. Вона дозволяє легко програмувати та налаштовувати контролери, що робить розробку електронних проектів доступнішою для багатьох людей. Додаткові функції та особливості Arduino IDE включають:

6. Платформа: Arduino IDE працює на різних платформах, включаючи Windows, Mac та Linux, що дозволяє користувачам працювати з програмою на будь-якому комп'ютері.
7. Підтримка різних контролерів: Arduino IDE підтримує багато різних моделей контролерів Arduino, що дозволяє розробникам працювати з різними моделями та використовувати їх для різних завдань.
8. Інструменти для відлагодження: Arduino IDE має вбудовані інструменти для відлагодження програм, такі як можливість встановлювати точки зупинки та відстежувати значення змінних.
9. Вбудовані приклади: Arduino IDE містить багато вбудованих прикладів програм для різних контролерів та сенсорів, що дозволяє користувачам швидко почати роботу з цими пристроями.
10. Розширення: Arduino IDE підтримує розширення, що дозволяє користувачам додавати нові функції та інструменти до програми.
11. Можливість створення власних бібліотек: Користувачі можуть створювати власні бібліотеки, що дозволяє зберігати і використовувати свій код для різних проектів.

12. Легкість використання: Arduino IDE має дуже простий та зрозумілий інтерфейс, що дозволяє користувачам легко розпочати роботу з програмою та контролерами Arduino.

Як бачимо, програма є досить корисна для розробки проектів з контролерами Arduino. Вона має велику кількість функцій та інструментів, що дозволяють користувачам легко програмувати та налаштовувати ці пристрої.

Arduino IDE має кілька переваг для розробки проектів з контролерами Arduino:

1. Простота використання: Arduino IDE має дуже простий та зрозумілий інтерфейс, що дозволяє користувачам легко розпочати роботу з програмою та контролерами Arduino.
2. Безкоштовний: Arduino IDE є безкоштовною програмою з відкритим кодом, що дозволяє розробникам та студентам з економією коштів використовувати її для своїх проектів.
3. Крос-платформений: Arduino IDE працює на різних платформах, включаючи Windows, Mac та Linux, що дозволяє користувачам працювати з програмою на будь-якому комп'ютері.
4. Вбудовані бібліотеки: Arduino IDE має вбудовані бібліотеки для роботи з різними пристроями, що дозволяє користувачам легко використовувати ці пристрої у своїх проектах.
5. Велика спільнота користувачів: Arduino IDE має велику спільноту користувачів, що дозволяє отримувати допомогу та поради від інших користувачів у разі потреби.
6. Вбудовані інструменти відлагодження: Arduino IDE має вбудовані інструменти для відлагодження програм, такі як можливість встановлювати точки зупинки та відстежувати значення змінних.
7. Розширення: Arduino IDE підтримує розширення, що дозволяє користувачам додавати нові функції та інструменти до програми.

Arduino IDE - це потужний та зручний інструмент для розробки проектів з контролерами Arduino, який має декілька переваг для розробників та студентів.

Існують деякі недоліки Arduino IDE, які можуть впливати на його використання:

1. Обмежена можливість редагування: Arduino IDE має обмежені можливості редагування коду порівняно з іншими текстовими редакторами, що може бути незручним для деяких користувачів.
2. Відсутність автоматичної підказки: Arduino IDE не має автоматичної підказки для функцій та методів, що може бути проблемою для початківців.
3. Низька продуктивність: Arduino IDE може бути повільним на старих комп'ютерах або при роботі з великими проектами.
4. Обмежена підтримка мов програмування: Arduino IDE має обмежену підтримку мов програмування порівняно з іншими інтегрованими середовищами розробки, що може бути незручним для деяких користувачів.
5. Відсутність інтеграції з системами контролю версій: Arduino IDE не має вбудованої підтримки систем контролю версій, що може бути проблемою для роботи в команді над великими проектами.
6. Обмеження на кількість використовуваних бібліотек: Arduino IDE має обмеження на кількість бібліотек, які можна використовувати в проекті, що може бути обмеженням для деяких проектів.

Хоча Arduino IDE має кілька недоліків, вона все ще є потужним та зручним інструментом для розробки проектів з контролерами Arduino, який дозволяє розробникам та студентам ефективно працювати з мікроконтролерами та реалізувати свої ідеї.

Написання коду для програмування контролерів Arduino має свої правила та рекомендації, щоб зробити його більш зрозумілим та ефективним, загальні правила:

- Використовуйте зрозумілі та описові назви для змінних, функцій та інших елементів коду. Назви мають бути короткими, але інформативними.
- Використовуйте коментарі, щоб пояснити, що робить кожен елемент коду. Коментарі мають бути короткими та зрозумілими.
- Розділяйте код на логічні блоки з допомогою відступів. Це допомагає зрозуміти структуру коду та легше знайти помилки.
- Використовуйте бібліотеки та фреймворки для спрощення програмування різноманітних функцій та пристроїв.
- Перевіряйте код на помилки перед завантаженням на контролер. Arduino IDE має вбудовану функцію перевірки коду, яка допомагає знайти синтаксичні та логічні помилки.
- Намагайтеся використовувати змінні з відповідним типом даних, наприклад, цілочисельні змінні для цілих чисел та дійсні для чисел з плаваючою точкою.
- Використовуйте функції для повторного використання коду та спрощення структури програми.
- Використовуйте команди затримки з обережністю, оскільки вони можуть спричинити затримку в роботі програми та погіршити її продуктивність.
- Завжди слідкуйте за тим, щоб зберігати копії свого коду та регулярно зберігати його на зовнішній носій або в хмарному сховищі.

Ці Правила допоможуть зробити код більш зрозумілим, читабельним та ефективним, що значно спростить процес розробки та підтримки програмного забезпечення для контролерів Arduino.

Звичайно є і основні принципи програмування, такі як декомпозиція, абстракція та модульність. Ці принципи допоможуть створювати більш складні програми, які легше зрозуміти та підтримувати.

Ще один важливий аспект програмування контролерів Arduino - це дотримання правил безпеки. Перед підключенням будь-яких пристроїв та

сенсорів, переконайтеся, що вони сумісні з контролером та не пошкодять його. Тому не потрібно притримуватися до правила електробезпеки, такі як відключення живлення перед підключенням або відключенням дротів.

Основні коди, які можуть бути використані в програмі Arduino IDE для контролерів Arduino, включають [22]:

1. Код для ініціалізації портів введення/виведення:

```
pinMode(pin, mode); // дозволяє встановити режим вводу/виводу піна
digitalWrite(pin, value); // встановлює вказане значення для цифрового піна
analogWrite(pin, value); // встановлює аналогове значення для піна
digitalRead(pin); // зчитує цифрове значення з піна
analogRead(pin); // зчитує аналогове значення з піна
```

2. Код для роботи з часом та затримками:

```
delay(ms); // затримка в мілісекундах
millis(); // повертає час в мілісекундах, що пройшов з моменту запуску програми
micros(); // повертає час в мікросекундах, що пройшов з моменту запуску програми
```

3. Код для роботи зі змінними та операторами:

```
int a = 10; // оголошення цілочисельної змінної зі значенням
float b = 3.14; // оголошення дійсної змінної зі значенням
a = a + 1; // додавання 1 до змінної 'a'
b = b * 2; // множення 'b' на 2
if (a > b) { // умовний оператор // код, який виконується, якщо 'a' більше 'b'}
else { // код, який виконується, якщо 'a' менше або дорівнює 'b'}
```

4. Код для роботи з функціями та циклами:

```
void setup() { // код, який виконується при запуску програми }
void loop() { // код, який виконується в кожній ітерації циклу }
for (int i = 0; i < 10; i++) { // цикл for // код, який виконується 10 разів }
```

while (condition) { // цикл while // код, який виконується, доки умова виконується }

do { // код, який виконує }

while (condition); // цикл do-while, код виконується хоча б один раз, після чого перевіряється умова

Ці коди не є повним списком можливостей Arduino IDE, але вони представляють основні функції, які можуть бути використані для програмування контролерів Arduino. Крім цього, є багато бібліотек та фреймворків, які можуть спростити роботу з різними сенсорами, пристроями та комунікаційними протоколами.

3.2 Розробка коду

Основні принципи написання коду - це зрозумілість, ефективність та структурованість.

Структурований код повинен мати чітко визначену організацію та логіку, так щоб його було легко змінювати та розширювати в майбутньому.

Для написання коду використовують спеціальні редактори коду або інтегровані середовища розробки (IDE). Після написання коду він компілюється або інтерпретується для виконання на комп'ютері.

Кнопка та звуковий сигнал

Кнопка може використовуватися для запуску, зупинки та перезапуску гіроскопа. Вона може також використовуватися для зміни режимів роботи гіроскопа, наприклад, для включення або виключення режиму самобалансування.

Звуковий сигнал може використовуватися для інформування користувача про певні події, які відбуваються під час роботи гіроскопа, такі як запуск, зупинка, перезавантаження або помилки. Він може також використовуватися для підтвердження виконання деяких команд, наприклад, для повідомлення користувача про зміну режиму роботи гіроскопа.

```
const int buz = 11;    // set the buzzer pin
const int btn = 13;    //set the button pin
```

```
int button;    //button variable
void setup()
{
  pinMode(btn,INPUT);       //set to INPUT state
  pinMode(buz,OUTPUT);      //set to OUTPUT state
}
void loop()
{
  button = digitalRead(btn);      //assign the button value to variable button
  if(button == 0)    //if press the button
  {
    delay(10);    //delay time
    if(button == 0)     //judge again, if the button is pressed
    {
      buzzer();  // execute the subfunction of buzzer
    }
  }
  else       // button not pressed
  {
    digitalWrite(buz,LOW);    // buzzer not sounds
  }
}
//buzzer makes tick sound
void buzzer()
{
    for(int i=0;i<50;i++)
    {
    digitalWrite(buz,HIGH);
    delay(1);
    digitalWrite(buz,LOW);
```

```
        delay(1);}
        delay(50);
        for(int i=0;i<50;i++)
        {
        digitalWrite(buz,HIGH);
        delay(1);
        digitalWrite(buz,LOW);
        delay(1);
        }
    }
```

Цей код описує програмування мікроконтролера Arduino для звукового сигналу (так званий біпер) при натисканні кнопки.

Перші дві рядки визначають піни Arduino, які використовуються для кнопки і біпера. Наступні рядки установлюють піни як вхідні та вихідні для їх подальшої роботи.

У функції loop() програма постійно перевіряє стан кнопки. Якщо кнопка натиснута (button == 0), то викликається функція buzzer(), яка генерує звуковий сигнал. Якщо кнопка не натиснута, біпер не звучить.

Функція buzzer() відтворює звуковий сигнал за допомогою змінної під назвою i у двох циклах, які виконуються 50 разів кожен. Кожен цикл включає біпер на короткий час, а потім вимикає його протягом того ж самого часу. Після першого циклу виконується додатковий перерву на 50 мс, після чого виконується другий цикл звукового сигналу, який аналогічний першому.

Ми перевірили кнопку щита балансу і гудок. Зараз, в коді буде перевірятися моторна здатність, що рухає платформу.

В даному випадку, ми управляємо напрямом двигуна права pin D8 D12 на REV4; швидкістю управляє pin D10. Напрям двигуна лівої сторони pin D7 D6 на REV4; швидкістю управляє pin D9, за інформацією звернутися за табл. 3.1..

Таблиця 3.1. — Вказаний контрольний метод

D8	D12	D10/PWM	Right motor	D7	D6	D9/PWM	Left motor
HIGH	LOW	0-255	Go front	HIGH	LOW	0-255	Go front
LOW	HIGH	0-255	Go back	LOW	HIGH	0-255	Go back
HIGH	HIGH	/	stop	HIGH	HIGH	/	stop
LOW	LOW	/	stop	LOW	LOW	/	stop

D9 D10 - шпильки Pwm, які можуть бути використані як Цифрова продукція або Аналога продукція.

Якщо використано як продукція Аналога, вимагається назвати analogwrite () функцію Arduino, і ця analogwrite () функція може управлятися в ряду 0 - 255.

Виклик до analogwrite () на масштабі 0 - 255, таке, що analogwrite (255) просить 100uty цикл (завжди на), і analogwrite (127) - 50uty цикл (наполовину на часі).

У коді чим більше значення ШІМ, встановлене контактами D9 і D10, тим вище швидкість двох двигунів. У наступному коді, ми тільки просто встановлюємо два двигуни, щоб прокрутити мотор вперед і назад.

```
//TB6612 pins
const int right_R1=8;
const int right_R2=12;
const int PWM_R=10;
const int left_L1=7;
const int left_L2=6;
const int PWM_L=9;
void setup()
{
  Serial.begin(9600);        //set the baud rate to 9600
  pinMode(right_R1,OUTPUT);    // set all TB6612pins to OUTPUT
  pinMode(right_R2,OUTPUT);
```

```
  pinMode(PWM_R,OUTPUT);
  pinMode(left_L1,OUTPUT);
  pinMode(left_L2,OUTPUT);
  pinMode(PWM_L,OUTPUT);
}
void loop()
{
  // two motors turn forward
  digitalWrite(right_R1,HIGH);
  digitalWrite(right_R2,LOW);
  digitalWrite(left_L1,HIGH);
  digitalWrite(left_L2,LOW);
  analogWrite(PWM_R,100);   // write into PWM value 0~255 (speed)
  analogWrite(PWM_L,100);
}
```

Цей код описує програмування мікроконтролера Arduino для приводу двигунів двохколісного робота за допомогою драйвера моторів TB6612.

Спочатку визначаються піни TB6612 і встановлюється їхній режим на вихідні.

У функції loop() встановлюються напрямки обертання двигунів. У цьому коді два мотори будуть рухатися вперед, оскільки для кожного з них піни R1 і L1 встановлені в режим HIGH, а піни R2 і L2 в режим LOW.

Далі встановлюються значення ШИМ для кожного з двигунів (в даному випадку 100). Це дозволяє регулювати швидкість руху двигунів, збільшуючи або зменшуючи значення від 0 до 255.

Таким чином, коли програма запущена, двигуни будуть рухатися вперед з однаковою швидкістю. Щоб змінити напрямок руху або швидкість, потрібно змінити відповідні піни і значення ШИМ.

Результати коду: Добре встановіть балансир, завантажте вихідний код і ввімкніть; увімкніть повзунок. І лівий, і правий двигуни починають обертатися вперед.

Тест кодера Холла

З встановленням швидкість правого та лівого двигуна за допомогою ШІМ значеннями D9 D10, було розглянуто, але для щоб отримати задану швидкість потрібно перевірити її за допомогою вбудованого в двигун датчика Холла.

Датчик Холла - це електронний датчик, який використовується для вимірювання обертання вала гіроскопа. В коді гіроскопа тест датчик Холла дозволяє перевірити правильність роботи датчик Холла та визначити напрямок обертання вала гіроскопа.

Тест датчик Холла може бути корисним в декількох випадках. Наприклад, якщо гіроскоп працює некоректно або нестабільно, то тест датчик Холла допоможе визначити, чи є проблема у датчик Холла, і потребується його заміна. Крім того, тест датчик Холла може бути корисним для відлагодження коду гіроскопа, оскільки він дозволяє перевірити правильність роботи датчика обертання вала гіроскопа в різних ситуаціях.

Він є досить важливим елементом в коді гіроскопа, оскільки він дозволяє перевірити правильність роботи кодера Холла та визначити напрямок обертання вала гіроскопа, що в свою чергу покращує стабільність та точність роботи гіроскопа. Тому ми цьому коді ми зможем обчислювати задану швидкість за кількістю імпульсів, отриманих від енкодера Холла протягом 100 мс. Код наведено в Додатку А.

Цей код працює з двигунами, які підключені до контролера двигуна TB6612. Крім того, він використовує оптичний енкодер, щоб вимірювати кількість обертів двигуна та контролювати його швидкість.

Код встановлює всі піни TB6612 як вихідні, а піни енкодера як вхідні. Після цього він задає напрямок руху двигунів, встановлює значення ШІМ для керування швидкістю та чекає деякий час. Після цього код виконується в циклі

while для вимірювання кількості імпульсів, що надходять з енкодера, протягом певного інтервалу часу. Результати вимірювання виводяться на моніторі швидкості та обертів для кожного двигуна.

Тобто, код забезпечує керування швидкістю двигунів та вимірювання їх швидкості та кількості обертів з використанням енкодера.

Тест Bluetooth

Bluetooth - це бездротовий протокол зв'язку, який дозволяє передавати дані між різними пристроями на невеликій відстані. Він часто використовується для з'єднання смартфонів, наушників, колонок, датчиків, годинників та інших пристроїв між собою. Bluetooth дозволяє обмінюватися даними між пристроями без необхідності використання кабелів або Інтернет-з'єднання. Це робить його дуже зручним і популярним серед користувачів різних пристроїв. Код наведено в Додатку Б.

Цей код дозволяє керувати двома моторами за допомогою модуля TB6612 та зчитувати команди через Bluetooth.

Задаються піни TB6612 та налаштовуються на вихід. Після цього програма чекає, коли буде доступне значення в буфері послідовного порту, тобто коли буде прийматися команда через Bluetooth. Значення зчитується у змінну val, після чого виконується команда в залежності від отриманої команди (F - рух вперед, B - рух назад, S - зупинка).

Функції front(), back() та Stop() відповідають за керування моторами в залежності від отриманої команди. В них виконуються відповідні дії з пінами TB6612 та PWM, що дозволяє контролювати швидкість руху моторів. Рух вперед та назад відрізняється за допомогою зміни напрямку обертання моторів (HIGH/LOW сигнали на відповідних пінах). Зупинка виконується шляхом встановлення PWM сигналу на 0 та встановлення пінів для напрямку обертання в однакове стан.

Тест MPU6050

Щоб забезпечити положення платформи використовують акселерометра і гіроскопа.

Теоретично, потрібен лише двовісний акселерометр (вісь Z у прямому напрямку та вісь Y вздовж напрямку руху автомобіля) та одновісний гіроскоп (обчислюється кутова швидкість по осі X у напрямку колеса автомобіля).

У цьому коді ми використовуємо вбудовану мікросхему MPU-6050 балансувального щита keyestudio для тестування даних три-координатного акселерометра і три-координатного гіроскопа, роздруковуючи їх на послідовний монітор. Діапазон прискорення +-2g; діапазон гіроскопа +-250°/S. Код наведено в Додатку В.

Цей код програмує Мікроконтролер для зчитування даних з акселерометра та гіроскопа, що знаходяться на платі MPU6050 за допомогою I2C інтерфейсу. Значення розміщуються в змінних і виводяться на моніторі порту серійного зв'язку (Serial Monitor) за допомогою функції "printData()". Код також містить функції для налаштування плати MPU6050 та обробки даних, які отримані з акселерометра та гіроскопа. Цей код можна використовувати як основу для будь-якої програми, яка використовує датчики руху для збору даних та реалізації функціональності.

Перш за все, у функції `setup()` відбувається ініціалізація I2C та MPU-6050 за допомогою функції `setupMPU()`. У функції `loop()` програма викликає функції `recordAccelRegisters()` та `recordGyroRegisters()`, які записують значення гіроскопа та акселерометра у відповідні змінні.

Функції `processAccelData()` та `processGyroData()` обробляють ці значення та переводять їх у одиниці виміру градусів на секунду та гравітаційного прискорення відповідно. У функції `printData()` значення гіроскопа та акселерометра виводяться на порт Serial.

Програма затримується на 100 мілісекунд за допомогою функції `delay(100)` у функції `loop()`.

Обчислення значень кута нахилу та кутової швидкості

У попередньому коді ми використовували акселерометр і гіроскоп для отримання положення автомобіля. У цьому коді ми використаємо дані, виміряні мікросхемою MPU-6050, щоб безпосередньо отримати положення

автомобіля, що балансує. В наступному коді дізнаємось, як обчислити значення кута нахилу та кутової швидкості, виявлені балансувальним автомобілем:

```
#include <MPU6050.h>  //MPU6050 library
#include <Wire.h>  //IIC communication library
MPU6050 mpu6050;  //Instantiate an MPU6050 object; name mpu6050
int16_t ax, ay, az, gx, gy, gz;  //Define three-axis acceleration, three-axis gyroscope variables
float Angle;  //angle variable
int16_t Gyro_x;  //Angular velocity variable
void setup()
{
// Join the I2C bus
Wire.begin();          //Join the I2C bus sequence
Serial.begin(9600);    //open serial monitor and set the baud rate to 9600
delay(1500);
mpu6050.initialize();          //initialize MPU6050
delay(2);
}
void loop()
{
 mpu6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);    //IIC to get MPU6050 six-axis data ax ay az gx gy gz
Angle = -atan2(ay , az) * (180/ PI);    //Radial rotation angle calculation formula; the negative sign is direction processing
Gyro_x = -gx / 131;    //The X-axis angular velocity calculated by the gyroscope; the negative sign is the direction processing
 Serial.print("Angle = ");
 Serial.print(Angle);
 Serial.print("Gyro_x = ");
```

```
    Serial.println(Gyro_x);
}
```

Цей код використовує бібліотеку MPU6050 для отримання даних про рухомий пристрій MPU6050, який містить 3-осевий акселерометр та 3-осевий гіроскоп. Код ініціалізує MPU6050 та отримує дані про рух у циклі. Для отримання кута нахилу використовується формула радіального обертання з акселерометра. Для отримання швидкості обертання використовується значення з гіроскопа. Отримані дані відображаються в моніторі порту за допомогою функції Serial.print(). Код наведено в Додатку Г.

Цей код використовує MPU6050, який є шестипівденним гіроскопом та акселерометром, для вимірювання кутів нахилу. Код реалізує фільтр Калмана для покращення точності вимірювання кутів.

Спочатку він ініціалізує MPU6050 та встановлює параметри фільтра Калмана, такі як коефіцієнти шуму гіроскопа та акселерометра, коефіцієнт кореляції акселерометра та інші.

У функції loop () він зчитує значення кутів нахилу з функції angle_calculate (), яка використовує значення з акселерометра та гіроскопа для розрахунку кутів нахилу.

Функція Kalman_Filter () реалізує сам фільтр Калмана для покращення точності вимірювання кутів нахилу. Ця функція використовує вже розраховані кути нахилу та швидкість обертання, які він зберігає у змінних angle та angle_speed, відповідно. За допомогою фільтра Калмана він обчислює оптимальні значення кута нахилу та швидкості обертання, що використовуються в функції angle_calculate ().

Регулювання вертикальної петлі

Регулювання вертикальної петлі для гіроскопа є важливою частиною коду для створення стабілізації системи, яка використовує гіроскоп. Вертикальна петля, або вертикальна обертова петля, використовується для вимірювання відхилення гіроскопа від звичайної вертикалі. За допомогою

цього вимірювання можна визначити кут нахилу або кутову швидкість системи.

Регулювання вертикальної петлі дозволяє забезпечити правильну роботу гіроскопа в системі стабілізації. Це досягається за допомогою регулювання коефіцієнтів управління в коді програми. Якщо коефіцієнти налаштовані неправильно, то можуть виникнути проблеми зі стабілізацією системи, що може призвести до пошкодження обладнання або несправності у пристрої, який використовує гіроскоп.

Тобто, регулювання вертикальної петлі для гіроскопа є важливою частиною коду для забезпечення правильної роботи системи стабілізації з гіроскопом. Він дозволяє забезпечити точність вимірювання кутів нахилу або кутових швидкостей і запобігає можливим проблемам зі стабілізацією системи. Код наведено в Додатку Ґ.

Цей код реалізує контроль моторів на основі платформи MPU6050 з використанням фільтра Калмана та ПД-регулятора. Платформа MPU6050 забезпечує датчики гіроскопу та акселерометра, які використовуються для вимірювання кута нахилу і кутової швидкості. Фільтр Калмана використовує ці вимірювання для підрахунку кута нахилу з меншою похибкою та плавнішим згладжуванням значень.

Контроль моторів забезпечується за допомогою драйверів моторів TB6612, підключених до платформи Arduino REV4. Вихідні сигнали кута нахилу обробляються ПД-регулятором, який визначає вихідний сигнал управління, щоб забезпечити балансування платформи.

Регулювання контуру швидкості

Регулювання швидкості в гіроскопі є важливою частиною коду для досягнення більш точної та стабільної роботи гіроскопа. Контур швидкості визначає швидкість зміни кутової швидкості гіроскопа, або диференціальний сигнал вихідного сигналу гіроскопа, який використовується для контролю швидкості зміни кутової швидкості.

Регулювання дозволяє забезпечити більш точну вимірювання кутової швидкості та більш точне управління системою стабілізації з гіроскопом. Наприклад, якщо швидкість зміни кутової швидкості занадто висока, то це може призвести до похибок у вимірюванні кутів нахилу та до нестабільної роботи системи стабілізації.

Регулювання є важливою частиною коду для забезпечення більш точного та стабільного вимірювання кутової швидкості та управління системою стабілізації з гіроскопом. Воно дозволяє забезпечити більш точну роботу системи та запобігає можливим проблемам, що можуть виникнути при використанні неправильної настройки контуру швидкості. Код наведено в Додатку Д.

Цей код використовується для керування двигунами робота на базі MPU6050, електронного гіроскопа та акселерометра. Він включає в себе керування кутом нахилу та швидкістю, а також вимірювання швидкості за допомогою енкодера.

В основі коду лежить бібліотека MPU6050, яка використовується для отримання даних про кут нахилу та швидкість з електронного гіроскопа та акселерометра. Код також використовує бібліотеку PinChangeInt для того, щоб всі піни на платі REV4 стали зовнішніми перериваннями.

Для керування двигунами використовуються піни TB6612. Код включає в себе параметри кута нахилу, які вимірюються, і фільтр Калмана, який використовується для обробки даних про кут нахилу і швидкість. Код також включає в себе параметри PID для керування кутом нахилу та швидкістю.

Щодо вимірювання швидкості, код використовує зовнішні переривання, щоб розрахувати значення пульса, який обчислюється за допомогою енкодера. Код включає в себе також змінні PI для керування швидкістю.

Код складається з функцій setup() та loop(), які використовуються для ініціалізації та роботи програми відповідно.

Керування через Bluetooth

Керування через Bluetooth для гіроскопа в коді є важливим елементом, який дозволяє користувачу віддалено контролювати гіроскоп та його функції, використовуючи мобільний телефон або інший пристрій з підтримкою Bluetooth. Це дає можливість змінювати настройки гіроскопа, які можуть бути важливими для його роботи в конкретних умовах, а також віддалено управляти процесом збору даних та передачі їх на інший пристрій. Керування через Bluetooth для гіроскопа в коді є важливим елементом, який забезпечує більш гнучкий та зручний спосіб керування гіроскопом та контролю над ним. Код наведено в Додатку Е.

Цей код використовує декілька бібліотек для зчитування даних з датчика MPU6050 та керування двома двигунами постійного струму за допомогою драйвера двигуна TB6612.

Перші кілька рядків коду містять необхідні бібліотеки для цього проекту, зокрема MsTimer2 для внутрішнього таймера, PinChangeInt для зовнішніх переривань, MPU6050 для зчитування даних з датчика та Wire для зв'язку з IIC. Потім створюється об'єкт з ім'ям mpu6050 і визначаються декілька цілих і плаваючих змінних для зберігання даних з датчика.

Наступні рядки визначають виводи для драйвера двигуна TB6612 та ініціалізують їх до початкового стану. Далі йдуть декілька змінних, що використовуються для обчислення кутів, зокрема angle_X, angle_Y, angle0, Gyro_x, Gyro_y та Gyro_z. Ці змінні використовуються для обчислення фільтра Калмана, який описано у наступному розділі коду.

Фільтр Калмана визначається кількома параметрами, зокрема Q_angle, Q_gyro та R_angle, та кількома змінними, зокрема K_0, K_1, t_0, t_1, angle_err, q_bias, accelz, angle, angleY_one, angle_speed, Pdot, P, PCt_0, PCt_1 та E. Метою фільтра Калмана є оцінка істинного кута датчика з урахуванням вимірювань від акселерометра та гіроскопа.

Далі визначено декілька змінних для контуру ПІД (пропорційного, інтегрального, похідного) керування, який використовується для керування

кутом, швидкістю та рухом робота. Серед визначених змінних: kp, ki, kd, kp_speed, ki_speed, kd_speed, kp_turn, ki_turn, kd_turn, setp0, PD_pwm, pwm1 і pwm2.

Наступний розділ коду використовується для підрахунку імпульсів з датчиків швидкості, які використовуються для обчислення швидкості робота. Змінні, визначені для цієї секції, включають PinA_left, PinA_right, count_right, count_left, speedcc, lz, rz, rpluse, lpluse, pulseright і pulseleft.

У наступному розділі визначено змінні, які використовуються для ПІ (пропорційно-інтегрального) контуру керування, який використовується для керування швидкістю робота. Серед визначених змінних: speeds_filterold, positions, flag1, PI_pwm, cc, speedout та speeds_filter.

Нарешті, є частина коду, яка використовується для керування рульовим керуванням робота. Змінні, визначені для цієї секції, включають turnmax, turnmin, turnout, Turn_pwm, zz і turncc. Функція циклу не містить значного коду, а в основному виводить значення змінних angle_speed і Gyro_x на послідовний монітор.

Код для контролю за рухом автомобіля на платформі з двома колесами. Код використовує акселерометр та гіроскоп MPU6050 для визначення кута нахилу платформи, а також два енкодери для вимірювання швидкості кожного з коліс. Код також містить PID-регулятори для управління кутом нахилу та швидкістю, які забезпечують рівномірний рух автомобіля. Крім того, код містить функції для визначення значень пульсації для кожного з коліс, а також для зчитування даних з Bluetooth-модуля. Код наведено в розділі Є.

Це програма для Arduino, яка використовує різні датчики та алгоритми керування, щоб збалансувати робота і дозволити йому рухатися вперед і повертати. Ось огляд програми:

До складу програми входять наступні бібліотеки:

– MsTimer2.h: для налаштування внутрішнього переривання таймера

- PinChangeInt.h: для перетворення всіх виводів Arduino на зовнішні переривання
- MPU6050.h: для взаємодії з датчиком акселерометра та гіроскопа MPU6050
- Wire.h: для зв'язку через I2C

Програма визначає різні виводи, які керують двигунами та енкодерами робота:

- right_R1, right_R2, PWM_R, left_L1, left_L2 та PWM_L: виводи керування двома двигунами робота
- PinA_left і PinA_right: виводи зовнішніх переривань для зчитування даних енкодерів з коліс робота

У програмі визначено різні змінні, які використовуються для обчислення кута і швидкості руху робота, а також для керування його рухами:

- ax, ay, az, gx, gy, gz: змінні тривісного прискорення та тривісного гіроскопа
- angle_X та angle_Y: змінні нахилу осей X та Y відповідно, обчислені за допомогою акселерометра
- Gyro_x, Gyro_y, Gyro_z: кутова швидкість для розрахунку гіроскопа
- Q_angle, Q_gyro, R_angle - значення коваріації для фільтра Калмана
- C_0, dt, K1, K_0, K_1, t_0, t_1, angle_err, q_bias, accelz, angle, angleY_one, angle_speed, Pdot, P, PCt_0, PCt_1, E: різні параметри для фільтрації Калмана
- kp, ki, kd: PID-параметри для кутового контуру
- kp_speed, ki_speed, kd_speed: PID-параметри для контуру швидкості
- kp_turn, ki_turn, kd_turn: PID-параметри для контуру повороту
- setp0: точка балансування кута
- PD_pwm - вихідне значення кута

- pwm1 та pwm2 - змінні для керування двигуном
- count_right та count_left - змінні для підрахунку імпульсів енкодера від правого та лівого колеса відповідно
- pulseright та pulseleft: кількість імпульсів для правого та лівого коліс відповідно
- speeds_filterold, positions, flag1, PI_pwm, cc, speedout, speeds_filter - змінні для керування швидкістю
- turnmax, turnmin, turnout: змінні для керування кермом
- Turn_pwm: вихідне значення керма

Функція setup ініціалізує контакти керування для двигунів та енкодерів, а також налаштовує датчик MPU6050. Вона також налаштовує переривання внутрішнього таймера, який виконує функцію DSzhongduan кожні 5 мілісекунд.

Функція циклу зчитує дані з датчика, обчислює кут і швидкість робота та керує його рухами за допомогою PID-алгоритмів та алгоритмів керування. Вона також отримує вхідні дані від модуля Bluetooth, що дозволяє дистанційно керувати рухами робота.

Регулювання кута балансу та керування через Bluetooth

Регулювання кута балансу та керування через Bluetooth для гіроскопа в коді є важливими функціями, які дозволяють досягти стійкого та точного руху гіроскопа.

Регулювання кута використовується для того, щоб забезпечити оптимальне розташування гіроскопа у просторі, що дозволяє досягти максимальної стійкості та точності руху. Керування кутом балансу здійснюється за допомогою сигналів, які надходять від датчиків, що вимірюють кут нахилу.

Керування через Bluetooth дозволяє віддалено керувати гіроскопом з використанням мобільного телефону або іншого пристрою з підтримкою Bluetooth. Це дає можливість змінювати параметри руху гіроскопа, такі як швидкість, кут нахилу, напрямок руху та інші параметри. Керування через

Bluetooth також дає можливість зчитувати дані з гіроскопа та передавати їх на інший пристрій для подальшої обробки.

Регулювання кута балансу та керування через Bluetooth для гіроскопа в коді є важливими функціями, які дозволяють забезпечити оптимальний рух гіроскопа, підвищити його стійкість та точність руху, а також забезпечити зручний та гнучкий спосіб керування гіроскопом з використанням мобільного телефону або іншого пристрою з підтримкою Bluetooth. Код наведено в Додатку Ж.

Цей код призначений для керування двигунами та сенсорами руху робота. Він використовує MPU6050 для отримання даних про прискорення та швидкість обертання робота, а також інші компоненти, такі як таймери, регулятори швидкості і енкодери швидкості, для контролю руху робота. Код містить реалізацію алгоритмів калманового фільтра та ПІД-регуляторів для стабілізації кута нахилу робота та його швидкості, а також ПД-регулятора для руху по кривих. В коді також є реалізація зчитування вхідних сигналів через блютуз.

До складу коду входить декілька бібліотек: MsTimer2, PinChangeInt, MPU6050 та Wire. Бібліотека MsTimer2 використовується для встановлення переривання таймера, бібліотека PinChangeInt використовується для визначення всіх виводів на платі REV4 як зовнішніх переривань, бібліотека MPU6050 використовується для взаємодії з гіроскопом MPU6050, а бібліотека Wire використовується для зв'язку з IIC.

Код визначає декілька змінних і констант для драйвера двигуна TB6612, таких як виводи, що використовуються для правого і лівого двигунів, виводи ШІМ, вивід зумера і вивід кнопки. Він також визначає змінні для даних прискорення і гіроскопа від MPU6050, а також змінні для кута, дрейфу гіроскопа і параметрів фільтра Калмана.

Код також визначає параметри PID-регулятора для керування кутом, швидкістю та рульовим керуванням. Він містить змінні для підрахунку

імпульсів і вимірювання швидкості за допомогою датчиків на основі ефекту Холла, а також змінні для Bluetooth-зв'язку.

У функції setup() виводи керування двигуном встановлюються у стан OUTPUT і призначаються початкові значення стану. Також визначаються вхідні та вихідні виводи і підключається шина I2C. MPU6050 ініціалізовано, а переривання таймера налаштовано на запуск функції DSzhongduan кожні 5 мілісекунд.

Функція loop() зчитує дані прискорення і гіроскопа з MPU6050, обчислює кути і дрейф гіроскопа та застосовує фільтр Калмана для отримання більш стабільного вимірювання кутів. Потім вона використовує ПІД-регулятор для регулювання швидкості двигуна і кута повороту на основі вимірів кута і швидкості.

Нарешті, функція loop() також зчитує вхідні дані через Bluetooth і відповідно коригує напрямок і швидкість робота. Підрахунок імпульсів і вимірювання швидкості за допомогою датчиків на основі ефекту Холла також виконується у функції loop().

РОЗДІЛ 4
ЗАГАЛЬНІ ПРАВИЛА ВИКОРИСТАННЯ ПРИСТРОЮ

Гіроскутер (іноді також відомий як самобалансуючий скейт) є високотехнологічним пристроєм, який дозволяє людині перевозити вантажі та переміщуватися на певній відстані швидко та комфортно. Він працює на основі принципу динамічної стабільності, використовуючи вбудований високошвидкісний центральний мікропроцесор, високоточні електронні гіроскопи та електронно керований приводний двигун.

Однією з головних особливостей гіроскутера є його спроможність реагувати на дії користувача та керувати рухом відповідно до цих дій. Це означає, що, певною програмою людина може керувати платформою гіроскутера, вона може легко керувати напрямком руху, швидкістю, поворотами та зупинкою, просто відхиляючи своє тіло в різних напрямках. Гіроскутери даного плану є простими у використанні, мають гнучке керування та легку конструкцію, що може посприяти їхній популярності серед користувачів.

Важливо зазначити, що існує декілька різних моделей гіроскутерів, які можуть мати різні характеристики та можливості. Деякі гіроскутери можуть мати додаткові функції, такі як вбудовані динаміки для програвання музики або ультразвуковий пристрій який буде оминати певну перешкоду на своєму шляху, прикладів вдосконалення та варіацій є багацько. Отже, при вдосконаленні гіроскутера потрібно враховувати індивідуальні потреби та вимоги.

Основні технічні характеристики самобалансуючої машини включають наступне:

1. Максимальна швидкість: зазвичай мають від 2 до 6 км / год.
2. Максимальна вага вантажу: вантажу до 1 кг.
3. Максимальний нахил: можуть пересуватися по нахилу до 15-20 градусів.

4. Ємність батареї: однієї батерейки 3400 мАг.
5. Час зарядки: час 3 годин.
6. Дальність керування: зазвичай до 10-15 метрів через модуль Bluetooth.
7. Система стабілізації: має вбудовану систему стабілізації, яка дозволяє їй автоматично коригувати рівень стабільності та уникати падінь.
8. Розміри: вона має довжину 25 см, ширину 7,5 см та висоту 12 см.

Однак, технічні характеристики можуть варіюватися в залежності від конкретної моделі самобалансуючої машини Keyestudio.

Перед початком використанні самобалансуючої машини перш за все, встановіть її на рівну поверхню і підключіть до живлення за допомогою зарядного пристрою. Завантажте додаток на свій смартфон або планшет, який буде використовуватися для керування машиною.

Перед початком експлуатації перевірте роботу системи стабілізації і рівень заряду батареї. Перед керуванням машиною, переконайтеся, що ви одягнені відповідно до правил техніки безпеки, які описані в інструкції.

Використовуйте машину тільки на рівних твердих поверхнях та уникайте перешкод, щоб уникнути падіння та пошкодження самобалансуючої машини. Керування може бути складним на початкових етапах, тому починайте зі спокійної їзди, а потім поступово збільшуйте швидкість.

Після використання машину необхідно вимкнути, відключити від живлення та зарядити батарею до повної ємності. Також рекомендується перевіряти машину на наявність пошкоджень та регулярно проводити технічне обслуговування для забезпечення безпечної та надійної роботи.

Детальна інструкція з підключення до Bluetooth на самобалансуючій машині Keyestudio може виглядати наступним чином:

1. Підключіть блок живлення до самобалансуючої машини Keyestudio, переконайтеся, що живлення встановлено вірно і має відповідний параметр напруги та струму, щоб запустити пристрій.

2. Перевірить чи модуль Bluetooth є в роз'єм на самобалансуючій машині, що модуль вставлено в правильний роз'єм, щоб уникнути можливих проблем з підключенням.

3. Увімкніть живлення на самобалансуючій машині, переконайтеся, що пристрій запускається і готовий до підключення до Bluetooth, та подайте на плату XBee HC-06 питання на неї.

4. На пристрої, з якого ви збираєтеся керувати самобалансуючою машиною, відкрийте налаштування Bluetooth, переконайтеся, що функція Bluetooth увімкнена і пошук доступних пристроїв включений. Встановити потрібний APK file для даної машини, знайдіть модуль Bluetooth на самобалансуючій машині у списку доступних пристроїв.

5. Після успішного знайдення модуля Bluetooth на самобалансуючій машині, встановіть підключення до модуля Bluetooth на самобалансуючій машині, використовуючи встановлення паролю, який вказаний в інструкції.

6. Після успішного підключення, відкрийте додаток на своєму пристрої, який дозволяє керувати самобалансуючою машиною, і відправляйте команди на самобалансуючу машину через Bluetooth.

Виконання цих кроків у правильній послідовності є важливим для успішного підключення до Bluetooth на самобалансуючій машині Keyestudio.

Основна безпека при роботі з гіроборда полягає в дотриманні правил електробезпеки та правил експлуатації електронної техніки.

Основні правила безпеки включають:

- Не допускайте потрапляння води та інших рідин на електронні компоненти машини;
- Не розбирайте машину та не змінюйте її конструкцію без належної кваліфікації;
- Не допускайте потрапляння металевих предметів до електронних компонентів машини;
- Не допускайте перевищення максимальної ваги, яку може нести самобалансуюча машина;

- Не використовуйте машину на дорозі або у небезпечних місцях;
- Не залишайте машину під напругою після використання;
- Не допускайте дітей або тварин поблизу пристрою, коли він увімкнений.

Також рекомендується користуватися самобалансуючою машиною на рівній твердій поверхні, використовувати захисний екран та захисні шоломи при необхідності, дотримуватися правил дорожнього руху та не перевищувати дозволену швидкість.

Найважливіше - завжди дотримуйтесь інструкції та правил експлуатації, щоб забезпечити безпеку вам та оточуючим.

Додатково до порад з безпеки, важливо також дотримуватися інструкції зі збирання та монтажу самобалансуючої машини, щоб забезпечити правильну роботу та функціонування пристрою.

Крім того, під час використання машини важливо слідкувати за рівнем заряду батареї, щоб уникнути непередбачуваних випадків та падінь з машини. Для ефективності використання машини, рекомендується тренуватися та навчатися її управлінню на рівній та безпечній поверхні перед переходом до більш складних умов. Також важливо бути уважним та пильним на дорозі та уникати небезпечних місць та ситуацій.

Загалом, з правильним підходом та дотриманням необхідних заходів безпеки, використання самобалансуючої машини може стати захоплюючим та корисним досвідом для розвитку навичок програмування та управління, а також для активного та здорового життєвого стилю.

Таким чином, самобалансуюча машина є досить цікавим та корисним електронним пристроєм для розвитку навичок програмування та управління. Однак , її використання потребує дотримання основних правил безпеки, таких як правильна постановка ніг на платформу, уникання гострих поворотів та раптових рухів, не перевантажування машини та не допускання перевищення

максимальної швидкості. Такі заходи допоможуть зменшити ризик травм та нещасних випадків та забезпечити безпечне та комфортне використання самобалансуючої машини Keyestudio.

ВИСНОВКИОшибка! Закладка не определена.

1. В даній дипломній роботі проведений детальний аналіз технічних документів та наукових публікацій, що стосуються теми дослідження. В першому розділі роботи проведений змістовний аналіз патентних документів України, зокрема проведено кількісний аналіз патентних документів світу. Також був здійснений кількісний аналіз наукових публікацій. Оглядовий аналіз був проведений з метою виявлення доцільних та перспективних сфер застосування розглянутого пристрою. Цей факт підтверджується тим ,що кількість публікацій та патентів з кожним роком зростає на 3-3,2%.
2. Процес дослідження та розробки включав розбір та аналіз всіх складових компонентів набору Keyestudio KS0193, включаючи Arduino-сумісну плату управління, двигуни, акселерометр, гіроскоп. Було проведено дослідження алгоритмів керування і регулювання, що використовуються для забезпечення самобалансування.
3. На основі отриманих знань було розроблено програмне забезпечення для управління самобалансуючим роботом, використовуючи мову програмування Arduino та налагоджено взаємодію між всіма компонентами системи. Було проведено ряд експериментів та тестів для оцінки функціональності, стійкості та ефективності системи. Отримані результати підтверджують ефективність розробленої системи самобалансуючого робота на основі набору Keyestudio KS0193. Система демонструє стабільну самобалансуючу поведінку та здатність до керування і маневрування. Вона може бути успішно використана як платформа для подальших досліджень у галузі робототехніки та автоматичного керування.

Розроблено код для балансу машини, та проведено аналіз балансу машини. Для балансування машини Keyestudio використовуються наступні дані з MPU-6050:

1. Прискорення по осям X, Y і Z - дані використовуються для визначення нахилу самобалансуючої машини в просторі.

2. Кут нахилу по осях X та Y - визначення стану самобалансуючої машини і коригування руху моторів для збереження балансу.

3. Гіроскопічне значення по осях X, Y і Z - визначення швидкості обертання самобалансуючої машини навколо вісей.

Зазначені дані зчитуються з MPU-6050 за допомогою вбудованих сенсорів і використовуються для аналізу машини та прийняття рішень щодо корекції руху за допомогою коду баланс здійснюється автоматично.

Оскільки в різних випадках при переміщені данні різні, тому числове значення постійно змінюється, данні які можуть бути отримані з MPU-6050:

1. Прискорення по осях X, Y і Z:

- X: -2.37 м/с²
- Y: 9.81 м/с²
- Z: 0.52 м/с²

2. Кут нахилу по осях X та Y:

- X: -4.57 градусів
- Y: 1.23 градусів

3. Гіроскопічні значення по осях X, Y і Z:

- X: 15.62 °/с
- Y: -8.94 °/с
- Z: 3.78 °/с

4. Дана машина є високотехнологічним пристроєм, який дозволяє переміщуватися на відстані швидко та комфортно, працюючи на основі принципу динамічної стабільності. Гіроскутери є простими у використанні, мають гнучке керування та легку конструкцію, що посприяє їхній популярності серед користувачів. При використанні гіроскутера потрібно дотримуватися загальних правил безпеки та використовувати його відповідно до інструкції виробника.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Пат. №105738 Україна, МПК (2016.01)B62K 1/00, заявник та власник БОВДУР АНТОН МАКСИМОВИЧ, — № u201504603. заявл 13.05.2015; опубл. 11.04.2016. Бюл. № 7. МОНОСФЕРА Бовдур Антон Максимович.
2. Пат. №69046 Україна, МПК (2012.01)B62K 11/00, заявник та власник Дмитрієнко Павло Павлович, — № u201109234. заявл 25.07.2011; опубл. 25.04.2012. Бюл. № 8. ЕЛЕКТРОВЕЛОСИПЕД "ЕКОСТАР" Бовдур Антон Максимович.
3. Пат. №21261 Україна, МПК (2006)B62K 1/00, заявник та власник ДАЧЕВ ХРІСТО АТАНАСОВ, – № u200608118. заявл 16.12.2004; опубл. 15.03.2007. Бюл. №3. ПРИВІДНИЙ МЕХАНІЗМ, ОСОБЛИВО ДЛЯ ОДНОКОЛІСНИХ ВЕЛОСИПЕДІВ Дачев Х.А., Бешеньйо І.Й., Деметер М., Гіалуш Й. Кьосегі А. Летак Л., Менніч Л., Мольняр М.К., Ракач Й., Рожа М., Шчеілі Р., Татош З, Віг Ш.
4. Пат. №135879 Україна, МПК (2006) B62M 1/24 (2013.01) B62K 3/00 B60K 7/00, заявник та власник Горенюк Віктор Васильович, – № u201901129. заявл 04.02.2019; опубл. 25.07.2019. Бюл. № 14. ЕЛЕКТРОБАЙК Горенюк Віктор Васильович.
5. Пат. №137655 Україна, МПК (2006) B60W 30/00, B62K 23/00, заявник та власник ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ "ДЕЛФАСТ", — № u201905663. заявл 24.05.2019; опубл. 25.10.2019. Бюл. № 20. ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЕРУВАННЯ ЕЛЕКТРОВЕЛОСИПЕДОМ Денисенко Сергій Вікторович; Даніель Тонкопій
6. Пат. №99141 Україна, МПК B62K 11/00, заявник та власник ДНІПРОДЗЕРЖИНСЬКИЙ ДЕРЖАВНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ, — № u201411340. заявл 17.10.2014; опубл. 25.05.2015. Бюл. № 10. ЕЛЕКТРОВЕЛОСИПЕД Пабат Анатолій Іванович; Попіль Олег Ігорович.

7. Патентне відомство Німеччини: веб-сайт URL:https://depatisnet.dpma.de/DepatisNet/depatisnet?window=1&space=menu&content=index&action=basis (дата звернення: 01.2.2023).
8. Щорічний огляд PCT: Міжнародна патентна система: веб-сайт. URL: https://www.wipo.int/pct/ru/activity/index.html (дата звернення: 01.2.2023).
9. Серія BOIB з економіки та статистики 2012 р.: веб-сайт. URL:https://www.wipo.int/edocs/pubdocs/en/patents/901/wipo_pub_901_2013.pdf (дата звернення: 01.2.2023).
10. Серія BOIB з економіки та статистики 2013 р.: веб-сайт. URL:https://www.wipo.int/edocs/pubdocs/en/patents/901/wipo_pub_901_2014.pdf (дата звернення: 02.2.2023).
11. Серія BOIB з економіки та статистики 2014 р.: веб-сайт. URL: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_901_2015.pdf (дата звернення: 02.2.2023).
12. Серія BOIB з економіки та статистики 2015 р.: веб-сайт. URL: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_901_2016.pdf (дата звернення: 03.2.2023).
13. Серія BOIB з економіки та статистики 2016 р.: веб-сайт. URL: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_901_2017.pdf (дата звернення: 03.2.2023).
14. Серія BOIB з економіки та статистики 2017 р.: веб-сайт. URL: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_901_2018.pdf (дата звернення: 03.2.2023).
15. Серія BOIB з економіки та статистики 2018 р.: веб-сайт. URL: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_901_2019.pdf (дата звернення: 05.2.2023).
16. Серія BOIB з економіки та статистики 2019 р.: веб-сайт. URL: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_901_2020.pdf (дата звернення: 05.2.20223.

17. Серія BOIB з економіки та статистики 2020 р.: веб-сайт. URL: https://www.wipo.int/edocs/pubdocs/en/wipo_pub_901_2021.pdf (дата звернення: 05.2.2023).
18. Серія BOIB з економіки та статистики 2021 р.: веб-сайт. URL: https://www.wipo.int/edocs/pubdocs/en/wipo-pub-901-2022-en-patent-cooperation-treaty-yearly-review-2022.pdf (дата звернення: 09.02.2023).
19. Серія BOIB з економіки та статистики 2022 р.: веб-сайт. URL: https://www.wipo.int/edocs/pubdocs/en/wipo-pub-901-2022-en-patent-cooperation-treaty-yearly-review-2022.pdf (дата звернення: 15.02.2023).
20. Гугл Академія: веб-сайт. URL: https://scholar.google.com.ua/schhp?hl=uk (дата звернення: 22.02.2023).
21. Ks0193 keyestudio Автомобіль, що самобалансується: веб-сайт. URL: https://wiki.keyestudio.com/Ks0193_keyestudio_Self-balancing_Car#Kit_List (дата звернення: 05.03.2023).
22. URL: https://doc.arduino.ua/ru/prog/ (дата звернення: 02.04.2023).

НУБіП України

НУБіП України

НУБіП України

ДОДАТКИ

НУБіП України

НУБіП України

НУБіП України

НУБіП України

НУБіП України

```
//TB6612 pins
const int right_R1=8;
const int right_R2=12;
const int PWM_R=10;
const int left_L1=7;
const int left_L2=6;
const int PWM_L=9;
const int PinA_left = 5;    // set the left motor's pulse pin to D5
const int PinA_right = 4;    //set the right motor's pulse pin to D4
int times=0,newtime=0,d_time=100;   // time, new time, time interval
int valA=0,valB=0,flagA=0,flagB=0;    //variable valA and valB for
calculating the number of pulse

void setup()
{
  Serial.begin(9600);

  pinMode(right_R1,OUTPUT);    // set the TB6612 pins to OUTPUT
  pinMode(right_R2,OUTPUT);
  pinMode(PWM_R,OUTPUT);
  pinMode(left_L1,OUTPUT);
  pinMode(left_L2,OUTPUT);
  pinMode(PWM_L,OUTPUT);
  pinMode(PinA_left,INPUT);    // set the pulse pin to INPUT
  pinMode(PinA_right,INPUT);
}
void loop()
{
  //both motors turn forward
  digitalWrite(right_R1,HIGH);
  digitalWrite(right_R2,LOW);
  digitalWrite(left_L1,HIGH);
  digitalWrite(left_L2,LOW);
  analogWrite(PWM_R,100);   //write into PWM value 0~255 (speed)
  analogWrite(PWM_L,200);
  newtime=times=millis();     //make newtime and times equal to the time
the program runs to here
  while((newtime-times)<d_time)    //if less than the setting d_time,
always loop
  {
    if(digitalRead(PinA_left)==HIGH&&flagA==0)   // if detects HIGH
    {
      valA++;      //valA plus 1
      flagA=1;
    }
    if(digitalRead(PinA_left)==LOW&&flagA==1)    // if LOW
    {
      valA++;     //valA plus 1
      flagA=0;
    }

    if(digitalRead(PinA_right)==HIGH&&flagB==0)
    {
      valB++;
      flagB=1;
    }
    if(digitalRead(PinA_right)==LOW&&flagB==1)
```

```
    {
      valB++;
      flagB=0;
    }
    newtime=millis();        //newtime equals to the time the program runs
to here
  }
  Serial.println(valA);      // print out the value of valA and B
  Serial.println(valB);
  valA=valB=0;             //set to 0
}
```

Додаток Б

```
//TB6612 pins
const int right_R1=8;
const int right_R2=12;
const int PWM_R=10;
const int left_L1=7;
const int left_L2=6;
const int PWM_L=9;
char val;   // Bluetooth variable
void setup()
{
  Serial.begin(9600);
    pinMode(right_R1,OUTPUT);   //set TB6612 pins OUTPUT
  pinMode(right_R2,OUTPUT);
  pinMode(PWM_R,OUTPUT);
  pinMode(left_L1,OUTPUT);
  pinMode(left_L2,OUTPUT);
  pinMode(PWM_L,OUTPUT);
}
void loop()
{
  if(Serial.available())    //if serial buffer value is available
  {
    val = Serial.read();      //assign the value read from serial port to
val
    Serial.println(val);
    switch(val)             //switch statement
    {
      case 'F': front(); break;     //motor turns front
      case 'B': back(); break;       //turn back
      case 'S': Stop();break;    //stop
    }
  }
}
//turn front
void front()
{
  digitalWrite(right_R1,HIGH);
  digitalWrite(right_R2,LOW);
  digitalWrite(left_L1,HIGH);
  digitalWrite(left_L2,LOW);
  analogWrite(PWM_R,100);
  analogWrite(PWM_L,100);
}
//turn back
```

```
void back()
{
  digitalWrite(right_R1,LOW);
  digitalWrite(right_R2,HIGH);
  digitalWrite(left_L1,LOW);
  digitalWrite(left_L2,HIGH);
  analogWrite(PWM_R,100);
  analogWrite(PWM_L,100);
}
//stop
void Stop()
{
  digitalWrite(right_R1,LOW);
  digitalWrite(right_R2,HIGH);
  digitalWrite(left_L1,LOW);
  digitalWrite(left_L2,HIGH);
  analogWrite(PWM_R,0);
  analogWrite(PWM_L,0);
}
```

Додаток В

```
#include <Wire.h>
 long accelX, accelY, accelZ;      // set to overall variable; can be used
directly inside the function.
float gForceX, gForceY, gForceZ;
 long gyroX, gyroY, gyroZ;
float rotX, rotY, rotZ;
 void setup() {
  Serial.begin(9600);
  Wire.begin();
  setupMPU();
}
 void loop() {
  recordAccelRegisters();
  recordGyroRegisters();
  printData();
  delay(100);
}
 void setupMPU(){
  // REGISTER 0x6B/REGISTER 107:Power Management 1
  Wire.beginTransmission(0b1101000); //This is the I2C address of the MPU
(b1101000/b1101001 for AC0 low/high datasheet Sec. 9.2)
  Wire.write(0x6B); //Accessing the register 6B/107 - Power Management
(Sec. 4.30)
  Wire.write(0b00000000); //Setting SLEEP register to 0, using the
internal 8 Mhz oscillator
  Wire.endTransmission();
   // REGISTER 0x1b/REGISTER 27:Gyroscope Configuration
  Wire.beginTransmission(0b1101000); //I2C address of the MPU
  Wire.write(0x1B); //Accessing the register 1B - Gyroscope Configuration
(Sec. 4.4)
  Wire.write(0x00000000); //Setting the gyro to full scale +/- 250deg./s (
转化为 rpm:250/360 * 60 = 41.67rpm) ;The highest can be converted to
2000deg./s
  Wire.endTransmission();
    // REGISTER 0x1C/REGISTER 28:ACCELEROMETER CONFIGURATION
```

```
  Wire.beginTransmission(0b1101000); //I2C address of the MPU
  Wire.write(0x1C); //Accessing the register 1C - Acccelerometer
Configuration (Sec. 4.5)
  Wire.write(0b00000000); //Setting the accel to +/- 2g (if choose +/- 16g
, the value would be 0b00011000)
  Wire.endTransmission();
}
void recordAccelRegisters() {
  // REGISTER 0x3B~0x40/REGISTER 59~64
  Wire.beginTransmission(0b1101000); //I2C address of the MPU
  Wire.write(0x3B); //Starting register for Accel Readings
  Wire.endTransmission();
  Wire.requestFrom(0b1101000,6); //Request Accel Registers (3B - 40)
   // Use left shift << and bit operations |  Wire.read() read once 1bytes
, and automatically read the data of the next address on the next call.
  while(Wire.available() < 6);  // Waiting for all the 6 bytes data to be
sent from the slave machine  (Must wait for all data to be stored in the
buffer before reading)
  accelX = Wire.read()<<8|Wire.read(); //Store first two bytes into accelX
 (Automatically stored as a defined long value)
  accelY = Wire.read()<<8|Wire.read(); //Store middle two bytes into
accelY
  accelZ = Wire.read()<<8|Wire.read(); //Store last two bytes into accelZ
  processAccelData();
}
 void processAccelData(){
  gForceX = accelX / 16384.0;     //float = long / float
  gForceY = accelY / 16384.0;
  gForceZ = accelZ / 16384.0;
}
 void recordGyroRegisters()
{
  // REGISTER 0x43~0x48/REGISTER 67~72
  Wire.beginTransmission(0b1101000); //I2C address of the MPU
  Wire.write(0x43); //Starting register for Gyro Readings
  Wire.endTransmission();
  Wire.requestFrom(0b1101000,6); //Request Gyro Registers (43 ~ 48)
  while(Wire.available() < 6);
  gyroX = Wire.read()<<8|Wire.read(); //Store first two bytes into accelX
  gyroY = Wire.read()<<8|Wire.read(); //Store middle two bytes into accelY
  gyroZ = Wire.read()<<8|Wire.read(); //Store last two bytes into accelZ
  processGyroData();
}
 void processGyroData() {
  rotX = gyroX / 131.0;
  rotY = gyroY / 131.0;
  rotZ = gyroZ / 131.0;}
 void printData() {
  Serial.print("Gyro (deg)");
  Serial.print(" X=");
  Serial.print(rotX);
  Serial.print(" Y=");
  Serial.print(rotY);
  Serial.print(" Z=");
  Serial.print(rotZ);
  Serial.print(" Accel (g)");
```

```
  Serial.print(" X=");
  Serial.print(gForceX);
  Serial.print(" Y=");
  Serial.print(gForceY);
  Serial.print(" Z=");
  Serial.println(gForceZ);}
```

Додаток Г

```
#include <MPU6050.h>      //MPU6050 library
#include <Wire.h>        //IIC communication library
MPU6050 mpu6050;     //Instantiate an MPU6050 object; name mpu60500
int16_t ax, ay, az, gx, gy, gz;     //Define three-axis acceleration,
three-axis gyroscope variables
float Angle;
float Gyro_x,Gyro_y,Gyro_z;  //calculate angular velocity of each axis by
gyroscope
//////////////////////Kalman_Filter/////////////////////////////
float Q_angle = 0.001;  //Covariance of gyroscope noise
float Q_gyro = 0.003;    //Covariance of gyroscope drift noise
float R_angle = 0.5;    //Covariance of accelerometer
char C_0 = 1;
float dt = 0.005; //The value of dt is the filter sampling time.
float K1 = 0.05; // a function containing the Kalman gain is used to
calculate the deviation of the optimal estimate.
float K_0,K_1,t_0,t_1;
float angle_err;
float q_bias;    //gyroscope drift
float accelz = 0;
float angle;
float angle_speed;
float Pdot[4] = { 0, 0, 0, 0};
float P[2][2] = {{ 1, 0 }, { 0, 1 }};
float  PCt_0, PCt_1, E;
//////////////////////Kalman_Filter/////////////////////////////
void setup()
{
  // Join the I2C bus
  Wire.begin();                                //Join the I2C bus sequence
  Serial.begin(9600);                           //open serial monitor and set
the baud rate to 9600
  delay(1500);
  mpu6050.initialize();                          //initialize MPU6050
  delay(2);
}
void loop()
{
  Serial.print("Angle = ");
  Serial.print(Angle);
  Serial.print("  K_angle = ");
  Serial.println(angle);
  Serial.print("Gyro_x = ");
  Serial.print(Gyro_x);
  Serial.print("  K_Gyro_x = ");
  Serial.println(angle_speed);
  mpu6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);     //IIC to get
MPU6050 six-axis ax ay az gx gy gz
```

```
  angle_calculate(ax, ay, az, gx, gy, gz, dt, Q_angle, Q_gyro, R_angle,
C_0, K1);      //obtain angle and KalmanFilter
}
//////////////////////////////angle calculate//////////////////////
void angle_calculate(int16_t ax,int16_t ay,int16_t az,int16_t gx,int16_t
gy,int16_t gz,float dt,float Q_angle,float Q_gyro,float R_angle,float
C_0,float K1)
{
  Angle = -atan2(ay , az) * (180/ PI);           //Radial rotation angle
calculation formula; negative sign is direction processing
  Gyro_x = -gx / 131;              //The X-axis angular velocity
calculated by the gyroscope; the negative sign is the direction processing
  Kalman_Filter(Angle, Gyro_x);            //KalmanFilter
}
//////////////////////////////////////////////////////////////////////
//////////////////////////////KalmanFilter//////////////////////
void Kalman_Filter(double angle_m, double gyro_m)
{
  angle += (gyro_m - q_bias) * dt;          //Prior estimate
  angle_err = angle_m - angle;

  Pdot[0] = Q_angle - P[0][1] - P[1][0];    //Differential of azimuth
error covariance
  Pdot[1] = - P[1][1];
  Pdot[2] = - P[1][1];
  Pdot[3] = Q_gyro;
    P[0][0] += Pdot[0] * dt;    //The integral of the covariance
differential of the prior estimate error
  P[0][1] += Pdot[1] * dt;
  P[1][0] += Pdot[2] * dt;
  P[1][1] += Pdot[3] * dt;
    //Intermediate variable of matrix multiplication
  PCt_0 = C_0 * P[0][0];
  PCt_1 = C_0 * P[1][0];
  //Denominator
  E = R_angle + C_0 * PCt_0;
  //Gain value
  K_0 = PCt_0 / E;
  K_1 = PCt_1 / E;
    t_0 = PCt_0;  //Intermediate variable of matrix multiplication
  t_1 = C_0 * P[0][1];
    P[0][0] -= K_0 * t_0;    //Posterior estimation error covariance
  P[0][1] -= K_0 * t_1;
  P[1][0] -= K_1 * t_0;
  P[1][1] -= K_1 * t_1;
    q_bias += K_1 * angle_err;    //Posterior estimation
  angle_speed = gyro_m - q_bias;   //The differential value of the output
value; work out the optimal angular velocity
  angle += K_0 * angle_err; ////Posterior estimation; work out the optimal
angle
}
```

Додаток Г

```
#include <MsTimer2.h>        //internal timer 2
#include <PinChangeInt.h>    //this library can make all pins of arduino
REV4 as external interrupt
#include <MPU6050.h>      //MPU6050 library
```

```
#include <Wire.h>        //IIC communication library
MPU6050 mpu6050;     //Instantiate an MPU6050 object; name mpu6050
int16_t ax, ay, az, gx, gy, gz;     //Instantiate an MPU6050 object; name
mpu6050
//TB6612 pins
const int right_R1=8;
const int right_R2=12;
const int PWM_R=10;
const int left_L1=7;
const int left_L2=6;
const int PWM_L=9;
//////////////////////angle parameters/////////////////////////////
float Angle;
float angle_X; //calculate the inclined angle variable of X-axis by
accelerometer
float angle_Y; //calculate the inclined angle variable of Y-axis by
accelerometer
float angle0 = 1; //Actual measured angle (ideally 0 degrees)
float Gyro_x,Gyro_y,Gyro_z;  //Angular angular velocity for gyroscope
calculation
//////////////////////angle parameters/////////////////////////////
//////////////////////Kalman_Filter/////////////////////////////
float Q_angle = 0.001;  //Covariance of gyroscope noise
float Q_gyro = 0.003;    //Covariance of gyroscope drift noise
float R_angle = 0.5;    //Covariance of accelerometer
char C_0 = 1;
float dt = 0.005; // The value of dt is the filter sampling time.
float K1 = 0.05; // a function containing the Kalman gain is used to
calculate the deviation of the optimal estimate
float K_0,K_1,t_0,t_1;
float angle_err;
float q_bias;    //gyroscope drift
float accelz = 0;
float angle;
float angleY_one;
float angle_speed;
float Pdot[4] = { 0, 0, 0, 0};
float P[2][2] = {{ 1, 0 }, { 0, 1 }};
float  PCt_0, PCt_1, E;
//////////////////////Kalman_Filter/////////////////////////
//////////////////////PD parameters/////////////////////////////
double kp = 34, ki = 0, kd = 0.62;                    //Angle loop
parameter
double setp0 = 0; //Angle balance point
int PD_pwm;  //angle output
float pwm1=0,pwm2=0;

//void anglePWM();
void setup()
{
  //set the control motor's pin to OUTPUT
  pinMode(right_R1,OUTPUT);
  pinMode(right_R2,OUTPUT);
  pinMode(left_L1,OUTPUT);
  pinMode(left_L2,OUTPUT);
  pinMode(PWM_R,OUTPUT);
  pinMode(PWM_L,OUTPUT);
  //Initial state value
  digitalWrite(right_R1,1);
```

```
  digitalWrite(right_R2,0);
  digitalWrite(left_L1,0);
  digitalWrite(left_L2,1);
  analogWrite(PWM_R,0);
  analogWrite(PWM_L,0);
  // Join I2C bus
  Wire.begin();                              //Join the I2C bus sequence
  Serial.begin(9600);                         //open serial monitor, set the
baud rate to 9600
  delay(1500);
  mpu6050.initialize();                        //initialize MPU6050
  delay(2);
  //5ms  use timer2 to set the timer interrupt (Note: using timer2 will
affect the PWM output of pin3 pin11.)
  MsTimer2::set(5, DSzhongduan);    //5ms execute the function DSzhongduan
once
  MsTimer2::start();    // start the interrupt
}
void loop()
{
  Serial.print("angle = ");
  Serial.println(angle);
  Serial.print("Angle = ");
  Serial.println(Angle);
  /*Serial.print("Gyro_x = ");
  Serial.println(Gyro_x);
  Serial.print("K_Gyro_x = ");
  Serial.println(angle_speed);*/
    //Serial.println(PD_pwm);
  //Serial.println(pwm1);
  //Serial.println(pwm2);
}
//////////////////////////////////interrupt/////////////////////////////
void DSzhongduan()
{
  sei();  //Allow overall interrupt
  mpu6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);     //IIC to get
MPU6050 six-axis data ax ay az gx gy gz
  angle_calculate(ax, ay, az, gx, gy, gz, dt, Q_angle, Q_gyro, R_angle,
C_0, K1);      //get angle and Kalman_Filter
  PD();         // angle loop of PD control
  anglePWM();
}
/////////////////////////////////////////////////////////////////
//////////////////////////////angle calculation/////////////////////////
void angle_calculate(int16_t ax,int16_t ay,int16_t az,int16_t gx,int16_t
gy,int16_t gz,float dt,float Q_angle,float Q_gyro,float R_angle,float
C_0,float K1)
{
  Angle = -atan2(ay , az) * (180/ PI);           //Radial rotation angle
calculation formula; negative sign is direction processing
  Gyro_x = -gx / 131;              //The X-axis angular velocity
calculated by the gyroscope; the negative sign is the direction processing
  Kalman_Filter(Angle, Gyro_x);             //  Kalman Filter
  //Rotation Angle Z axis parameter
  Gyro_z = -gz / 131;                       //Z-axis angular velocity
  //accelz = az / 16.4;
  float angleAx = -atan2(ax, az) * (180 / PI); //Calculate the angle with
the x-axis
```

```
  Gyro_y = -gy / 131.00; //Y-axis angular velocity
  Yiorderfilter(angleAx, Gyro_y); //first-order filter
}
/////////////////////////////////////////////////////////////////
//////////////////////////////KalmanFilter/////////////////////
void Kalman_Filter(double angle_m, double gyro_m)
{
  angle += (gyro_m - q_bias) * dt;          //Prior estimate
  angle_err = angle_m - angle;
    Pdot[0] = Q_angle - P[0][1] - P[1][0];    //Differential of azimuth
error covariance
  Pdot[1] = - P[1][1];
  Pdot[2] = - P[1][1];
  Pdot[3] = Q_gyro;
    P[0][0] += Pdot[0] * dt;    //A priori estimation error covariance
differential integral
  P[0][1] += Pdot[1] * dt;
  P[1][0] += Pdot[2] * dt;
  P[1][1] += Pdot[3] * dt;
    //Intermediate variable of matrix multiplication
  PCt_0 = C_0 * P[0][0];
  PCt_1 = C_0 * P[1][0];
  //Denominator
  E = R_angle + C_0 * PCt_0;
  //gain value
  K_0 = PCt_0 / E;
  K_1 = PCt_1 / E;
    t_0 = PCt_0;  //Intermediate variable of matrix multiplication
  t_1 = C_0 * P[0][1];
    P[0][0] -= K_0 * t_0;    //Posterior estimation error covariance
  P[0][1] -= K_0 * t_1;
  P[1][0] -= K_1 * t_0;
  P[1][1] -= K_1 * t_1;
    q_bias += K_1 * angle_err;    //Posterior estimate
  angle_speed = gyro_m - q_bias;   //The differential of the output value
gives the optimal angular velocity
  angle += K_0 * angle_err; ////Posterior estimation to get the optimal
angle
}
//////////////////////first-order Filter/////////////////
void Yiorderfilter(float angle_m, float gyro_m)
{
  angleY_one = K1 * angle_m + (1 - K1) * (angleY_one + gyro_m * dt);
}
//////////////////angle PD////////////////////
void PD()
{
  PD_pwm = kp * (angle + angle0) + kd * angle_speed; //PD angle loop
control
}
/////////////////////////////PWM end value/////////////////////////////
void anglePWM()
{
  pwm2=-PD_pwm;            //The final value assigned to the motor PWM
  pwm1=-PD_pwm;

  if(pwm1>255)             //limit PWM value not greater than 255
  {
    pwm1=255;
```

```
  }
  if(pwm1<-255)
  {
    pwm1=-255;
  }
  if(pwm2>255)
  {
    pwm2=255;
  }
  if(pwm2<-255)
  {
    pwm2=-255;
}
  if(angle>80 || angle<-80)      //When the self-balancing trolley's tilt
angle is greater than 45 degrees, the motor will stop.
  {
    pwm1=pwm2=0;}
  if(pwm2>=0)          //determine the motor's steering and speed by the
positive and negative of PWM
  {
    digitalWrite(left_L1,LOW);
    digitalWrite(left_L2,HIGH);
    analogWrite(PWM_L,pwm2);}
  else
  {
    digitalWrite(left_L1,HIGH);
    digitalWrite(left_L2,LOW);
    analogWrite(PWM_L,-pwm2);}
  if(pwm1>=0)
  {
    digitalWrite(right_R1,LOW);
    digitalWrite(right_R2,HIGH);
    analogWrite(PWM_R,pwm1);}
  else
  {
    digitalWrite(right_R1,HIGH);
    digitalWrite(right_R2,LOW);
    analogWrite(PWM_R,-pwm1);}
}
```

Додаток Д

```
#include <MsTimer2.h>        //internal timer 2
#include <PinChangeInt.h>    //This library file can make all pins on the
REV4 board as external interrupts.
#include <MPU6050.h>      //MPU6050 library
#include <Wire.h>        //IIC library
MPU6050 mpu6050;     //Instantiate an MPU6050 object; name mpu6050
int16_t ax, ay, az, gx, gy, gz;     // Define three-axis acceleration,
three-axis gyroscope variables
//TB6612 pins
const int right_R1=8;
const int right_R2=12;
const int PWM_R=10;
const int left_L1=7;
const int left_L2=6;
const int PWM_L=9;
//////////////////////angle parameters/////////////////////////////
```

```
float angle_X; //Calculate the tilt angle variable about the X axis from
the acceleration
float angle_Y; //Calculate the tilt angle variable about the Y axis from
the acceleration
float angle0 = 1; //Actual measured angle (ideally 0 degrees)
float Gyro_x,Gyro_y,Gyro_z;  //Angular angular velocity by gyroscope
calculation
//////////////////////angle parameters/////////////////////////////
//////////////////////Kalman_Filter/////////////////////////////
float Q_angle = 0.001;  //Covariance of gyroscope noise
float Q_gyro = 0.003;    //Covariance of gyroscope drift noise
float R_angle = 0.5;    //Covariance of accelerometer
char C_0 = 1;
float dt = 0.005; // The value of dt is the filter sampling time.
float K1 = 0.05; //a function containing the Kalman gain is used to
calculate the deviation of the optimal estimate
float K_0,K_1,t_0,t_1;
float angle_err;
float q_bias;    //Gyro drift
float accelz = 0;
float angle;
float angleY_one;
float angle_speed;
float Pdot[4] = { 0, 0, 0, 0};
float P[2][2] = {{ 1, 0 }, { 0, 1 }};
float  PCt_0, PCt_1, E;
//////////////////////Kalman_Filter/////////////////////////
//////////////////////PID parameters/////////////////////////////////
double kp = 34, ki = 0, kd = 0.62;                    //angle loop
parameters
double kp_speed = 3.6, ki_speed = 0.080, kd_speed = 0;   // speed loop
parameters
double setp0 = 0; //angle balance point
int PD_pwm;  //angle output
float pwm1=0,pwm2=0;
//////////////////Interrupt speed measurement//////////////////////////////
#define PinA_left 5  //external interrupts
#define PinA_right 4   //external interrupts
volatile long count_right = 0;//Used to calculate the pulse value
calculated by the Hall encoder (the volatile long type is to ensure the
value is valid)
volatile long count_left = 0;
int speedcc = 0;
//////////////////////pulse calculation//////////////////////////
int lz = 0;
int rz = 0;
int rpluse = 0;
int lpluse = 0;
int pulseright,pulseleft;
////////////////////////////////PI variable
parameters/////////////////////////
float speeds_filterold=0;
float positions=0;
int flag1;
double PI_pwm;
int cc;
int speedout;
float speeds_filter;
void setup()
```

```
{
  // set the pins of motor to OUTPUT
  pinMode(right_R1,OUTPUT);
  pinMode(right_R2,OUTPUT);
  pinMode(left_L1,OUTPUT);
  pinMode(left_L2,OUTPUT);
  pinMode(PWM_R,OUTPUT);
  pinMode(PWM_L,OUTPUT);
  //assign initial state value
  digitalWrite(right_R1,1);
  digitalWrite(right_R2,0);
  digitalWrite(left_L1,0);
  digitalWrite(left_L2,1);
  analogWrite(PWM_R,0);
  analogWrite(PWM_L,0);
  pinMode(PinA_left, INPUT);  //speed code wheel input
  pinMode(PinA_right, INPUT);
  // join I2C bus
  Wire.begin();                              //join I2C bus sequence
  Serial.begin(9600);                         //open the serial monitor to
set the baud rate to 9600
  delay(1500);
  mpu6050.initialize();                        //initialize MPU6050
  delay(2);
  //5ms; use timer2 to set timer interruption (note：using timer2 will
affect the PWM output of pin3 pin11)
  MsTimer2::set(5, DSzhongduan);    //5ms ; execute the function
DSzhongduan once
  MsTimer2::start();    //start interrupt
}
void loop()
{
  Serial.println(angle);
  delay(100);
  //Serial.println(PD_pwm);
  //Serial.println(pwm1);
  //Serial.println(pwm2);
  //Serial.print("pulseright = ");
  //Serial.println(pulseright);
  //Serial.print("pulseleft = ");
  //Serial.println(pulseleft);
  //Serial.println(PI_pwm);
  //Serial.println(speeds_filter);
  //Serial.println (positions);
    //External interrupt for calculating wheel speed
  attachPinChangeInterrupt(PinA_left, Code_left, CHANGE);
//PinA_left Level change triggers external interrupt; execute subfunction
Code_left
  attachPinChangeInterrupt(PinA_right, Code_right, CHANGE);
//PinA_right Level change triggers external interrupt; execute subfunction
Code_right
}
//////////////////////Hall calculation//////////////////////////
//left speed code wheel count
void Code_left()
{
  count_left ++;
}
//Right speed code wheel count
```

```
void Code_right()
{
  count_right ++;
}
//////////////////////pulse calculation//////////////////////
void countpluse()
{
  lz = count_left;     //Assign the value counted by the code wheel to lz
  rz = count_right;
  count_left = 0;     //Clear the code counter count
  count_right = 0;
  lpluse = lz;
  rpluse = rz;
  if ((pwm1 < 0) && (pwm2 < 0))                    //judge the moving
direction; if backwards (PWM, namely motor voltage is negative) , pulse
number is a negative number
  {
    rpluse = -rpluse;
    lpluse = -lpluse;
  }
  else if ((pwm1 > 0) && (pwm2 > 0))                // if backwards (PWM,
namely motor voltage is positive) , pulse number is a positive number
  {
    rpluse = rpluse;
    lpluse = lpluse;
  }
  else if ((pwm1 < 0) && (pwm2 > 0))                //Judge turning
direction of the car;  turn left; Right pulse number is a positive number;
Left pulse number is a negative number.
  {
    rpluse = rpluse;
    lpluse = -lpluse;
  }
  else if ((pwm1 > 0) && (pwm2 < 0))               //Judge turning
direction of the car;  turn right; Right pulse number is a negative
number; Left pulse number is a positive number.
  {
    rpluse = -rpluse;
    lpluse = lpluse;
  }
  //enter interrupts per 5ms; pulse number superposes
  pulseright += rpluse;
  pulseleft += lpluse;
}
//////////////////////////////////interrupts////////////////////////////
void DSzhongduan()
{
  sei();  //Allow global interrupts
  countpluse();        //Pulse superposition subfunction
  mpu6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);     //IIC to get
MPU6050 six-axis data  ax ay az gx gy gz
  angle_calculate(ax, ay, az, gx, gy, gz, dt, Q_angle, Q_gyro, R_angle,
C_0, K1);      //get angle and Kalman filtering
  PD();          //angle loop PD control
  anglePWM();
  cc++;
  if(cc>=8)     //5*8=40, 40ms entering once speed PI algorithm
```

```
  {
    speedpiout();
    cc=0;  //Clear
  }
}
/////////////////////////////////////////////////////////
//////////////////////////////angle calculation/////////////////////////
void angle_calculate(int16_t ax,int16_t ay,int16_t az,int16_t gx,int16_t
gy,int16_t gz,float dt,float Q_angle,float Q_gyro,float R_angle,float
C_0,float K1)
{
  float Angle = -atan2(ay , az) * (180/ PI);           //Radial rotation
angle calculation formula; negative sign is direction processing
  Gyro_x = -gx / 131;               //The X-axis angular velocity
calculated by the gyroscope; the negative sign is the direction processing
  Kalman_Filter(Angle, Gyro_x);            //Kalman Filtering
  //Rotation angle Z-axis parameter
  Gyro_z = -gz / 131;                      //Z-axis angular velocity
  //accelz = az / 16.4;
  float angleAx = -atan2(ax, az) * (180 / PI); //Calculate the angle with
the x-axis
  Gyro_y = -gy / 131.00; //Y-axis angular velocity
  Yiorderfilter(angleAx, Gyro_y); //first-order filtering
}
/////////////////////////////////////////////////////////////////
//////////////////////////////KalmanFilter/////////////////////////
void Kalman_Filter(double angle_m, double gyro_m)
{
  angle += (gyro_m - q_bias) * dt;          //Prior estimate
  angle_err = angle_m - angle;
    Pdot[0] = Q_angle - P[0][1] - P[1][0];    //Differential of azimuth
error covariance
  Pdot[1] = - P[1][1];
  Pdot[2] = - P[1][1];
  Pdot[3] = Q_gyro;
    P[0][0] += Pdot[0] * dt;    //A priori estimation error covariance
differential integral
  P[0][1] += Pdot[1] * dt;
  P[1][0] += Pdot[2] * dt;
  P[1][1] += Pdot[3] * dt;
    //Intermediate variable of matrix multiplication
  PCt_0 = C_0 * P[0][0];
  PCt_1 = C_0 * P[1][0];
  //Denominator
  E = R_angle + C_0 * PCt_0;
  //gain value
  K_0 = PCt_0 / E;
  K_1 = PCt_1 / E;
    t_0 = PCt_0;  //Intermediate variable of matrix multiplication
  t_1 = C_0 * P[0][1];
    P[0][0] -= K_0 * t_0;    //Posterior estimation error covariance
  P[0][1] -= K_0 * t_1;
  P[1][0] -= K_1 * t_0;
  P[1][1] -= K_1 * t_1;
    q_bias += K_1 * angle_err;    //Posterior estimate
  angle_speed = gyro_m - q_bias;   //The differential of the output value
gives the optimal angular velocity
  angle += K_0 * angle_err; ////Posterior estimation to get the optimal
angle
```

```
}
//////////////////////first-order filtering/////////////////
void Yiorderfilter(float angle_m, float gyro_m)
{
  angleY_one = K1 * angle_m + (1 - K1) * (angleY_one + gyro_m * dt);
}
///////////////////angle PD///////////////////
void PD()
{
  PD_pwm = kp * (angle + angle0) + kd * angle_speed; //PD angle loop
control
}
///////////////////speed PI///////////////////
void speedpiout()
{
  float speeds = (pulseleft + pulseright) * 1.0;      //Vehicle speed
pulse value
  pulseright = pulseleft = 0;      //Clear
  speeds_filterold *= 0.7;         //first-order complementary filtering
  speeds_filter = speeds_filterold + speeds * 0.3;
  speeds_filterold = speeds_filter;
  positions += speeds_filter;
  positions = constrain(positions, -3550,3550);    //Anti-integral
saturation
  PI_pwm = ki_speed * (setp0 - positions) + kp_speed * (setp0 -
speeds_filter);      //speed loop control PI
}
///////////////////speed PI///////////////////
//////////////////////////////PWM end value////////////////////////////
void anglePWM()
{
  pwm2=-PD_pwm - PI_pwm ;           //assign the final value of PWM to
motor
  pwm1=-PD_pwm - PI_pwm ;
    if(pwm1>255)             //limit PWM value not greater than 255
  {
    pwm1=255;
  }
  if(pwm1<-255)
  {
    pwm1=-255;
  }
  if(pwm2>255)
  {
    pwm2=255;
  }
  if(pwm2<-255)
  {
    pwm2=-255;
  }
  if(angle>80 || angle<-80)      // the inclined angle of balance car is
greater than 45°, motor will stop.
  {
    pwm1=pwm2=0;
  }
 if(pwm2>=0)         // determine the motor's steering and speed according
to the positive and negative of PWM
  {
    digitalWrite(left_L1,LOW);
```

```
    digitalWrite(left_L2,HIGH);
    analogWrite(PWM_L,pwm2);
  }
  else
  {
    digitalWrite(left_L1,HIGH);
    digitalWrite(left_L2,LOW);
    analogWrite(PWM_L,-pwm2);
  }
  if(pwm1>=0)
  {
    digitalWrite(right_R1,LOW);
    digitalWrite(right_R2,HIGH);
    analogWrite(PWM_R,pwm1);
  }
  else
  {
    digitalWrite(right_R1,HIGH);
    digitalWrite(right_R2,LOW);
    analogWrite(PWM_R,-pwm1);
  }
}
```

Додаток Е

```
#include <MsTimer2.h>        //internal timer 2
#include <PinChangeInt.h>    //this library can make all pins of arduino
REV4 as external interrupt
#include <MPU6050.h>      //MPU6050 library
#include <Wire.h>        //IIC communication library
MPU6050 mpu6050;     //Instantiate an MPU6050 object; name mpu6050
int16_t ax, ay, az, gx, gy, gz;     //Define three-axis acceleration,
three-axis gyroscope variables
//TB6612 pins
const int right_R1=8;
const int right_R2=12;
const int PWM_R=10;
const int left_L1=7;
const int left_L2=6;
const int PWM_L=9;
//////////////////////angle parameters/////////////////////////////
float angle_X; //calculate the inclined angle variable of X-axis by
accelerometer
float angle_Y; //calculate the inclined angle variable of Y-axis by
accelerometer
float angle0 = 1; //Actual measured angle (ideally 0 degrees)
float Gyro_x,Gyro_y,Gyro_z;  //Angular angular velocity for gyroscope
calculation
//////////////////////angle parameters/////////////////////////////
//////////////////////Kalman_Filter/////////////////////////////
float Q_angle = 0.001;  //Covariance of gyroscope noise
float Q_gyro = 0.003;    //Covariance of gyroscope drift noise
float R_angle = 0.5;    //Covariance of accelerometer
char C_0 = 1;
float dt = 0.005; //The value of dt is the filter sampling time
float K1 = 0.05; //  a function containing the Kalman gain; used to
calculate the deviation of the optimal estimate
float K_0,K_1,t_0,t_1;
```

```
float angle_err;
float q_bias;    //gyroscope drift
float accelz = 0;
float angle;
float angleY_one;
float angle_speed;
float Pdot[4] = { 0, 0, 0, 0};
float P[2][2] = {{ 1, 0 }, { 0, 1 }};
float  PCt_0, PCt_1, E;
//////////////////////Kalman_Filter/////////////////////////
//////////////////////PID parameters/////////////////////////////////
double kp = 34, ki = 0, kd = 0.62;                    //angle loop
parameters
double kp_speed = 3.6, ki_speed = 0.080, kd_speed = 0;   // speed loop
parameters
double kp_turn = 24, ki_turn = 0, kd_turn = 0.08;       // steering loop
parameters
double setp0 = 0; //Angle balance point
int PD_pwm;  //angle output
float pwm1=0,pwm2=0;
//////////////////Interrupt speed count//////////////////////////////
#define PinA_left 5  //external interrupt
#define PinA_right 4   //external interrupt
volatile long count_right = 0;//Used to calculate the pulse value
calculated by the Hall encoder (the volatile long type is to ensure the
value is valid)
volatile long count_left = 0;
int speedcc = 0;
//////////////////////pulse count//////////////////////////
int lz = 0;
int rz = 0;
int rpluse = 0;
int lpluse = 0;
int pulseright,pulseleft;
////////////////////////////////PI variable
parameter/////////////////////////
float speeds_filterold=0;
float positions=0;
int flag1;
double PI_pwm;
int cc;
int speedout;
float speeds_filter;
////////////////////////////////steering PD//////////////////
int turnmax,turnmin,turnout;
float Turn_pwm = 0;
int zz=0;
int turncc=0;
//Bluetooth//
int front = 0;//go front variable
int back = 0;//go back variable
int left = 0;//turn left
int right = 0;//turn right
char val;
void setup()
{
  //set the motor control pins to OUTPUT
  pinMode(right_R1,OUTPUT);
  pinMode(right_R2,OUTPUT);
```

```
  pinMode(left_L1,OUTPUT);
  pinMode(left_L2,OUTPUT);
  pinMode(PWM_R,OUTPUT);
  pinMode(PWM_L,OUTPUT);
  //assign initial state value
  digitalWrite(right_R1,1);
  digitalWrite(right_R2,0);
  digitalWrite(left_L1,0);
  digitalWrite(left_L2,1);
  analogWrite(PWM_R,0);
  analogWrite(PWM_L,0);
  pinMode(PinA_left, INPUT);  //Speed encoder input
  pinMode(PinA_right, INPUT);
  // join I2C bus
  Wire.begin();                              //join I2C bus sequence
  Serial.begin(9600);                         //open the serial monitor and
set the baud rate to 9600
  delay(1500);
  mpu6050.initialize();                         //initialize MPU6050
  delay(2);
  //5ms; use timer2 to set timer interruption (note:using timer2 will
affect the PWM output of pin3 pin11)
  MsTimer2::set(5, DSzhongduan);    //5ms ; execute the function
DSzhongduan once
  MsTimer2::start();    //start interrupt
}
void loop()
{
  Serial.print(angle_speed);
  Serial.print("     ");
  Serial.println(Gyro_x);
    //Serial.println(angle);
  //delay(100);
  //Serial.println(PD_pwm);
  //Serial.println(pwm1);
  //Serial.println(pwm2);
  //Serial.print("pulseright = ");
  //Serial.println(pulseright);
  //Serial.print("pulseleft = ");
  //Serial.println(pulseleft);
  //Serial.println(PI_pwm);
  //Serial.println(speeds_filter);
  //Serial.println (positions);
  //Serial.println(Turn_pwm);
  //Serial.println(Gyro_z);
  //Serial.println(Turn_pwm);
    if(Serial.available())
  {
    val = Serial.read();      //assign the value read from serial port to
val
    //Serial.println(val);
    switch(val)             //switch statement
    {
      case 'F': front=250; break;       //if val equals to F, front=250,
balance robot goes front.
      case 'B': back=-250; break;       //go back
      case 'L': left=1; break;    //turn left
      case 'R': right=1; break;                         // turn right
```

```
      case 'S': front=0,back=0,left=0,right=0;break;    //stop
      case 'D': Serial.print(angle);break;    //when receiving 'D', send
value of angle to APP
    }
  }
    //external interrupt; used to calculate the wheel speed
  attachPinChangeInterrupt(PinA_left, Code_left, CHANGE);
//PinA_left  Level change triggers external interrupt;  execute
subfunction Code_left
  attachPinChangeInterrupt(PinA_right, Code_right, CHANGE);
//PinA_right Level change triggers external interrupt;  execute
subfunction Code_right
}
//////////////////////Hall count/////////////////////////
//left speed encoder count
void Code_left()
{
  count_left ++;
}
//right speed encoder count
void Code_right()
{
  count_right ++;
}
//////////////////////pulse count/////////////////////////
void countpluse()
{
  lz = count_left;     //assign the value counted by encoder to lz
  rz = count_right;
  count_left = 0;     //clear the count quantity
  count_right = 0;
  lpluse = lz;
  rpluse = rz;
  if ((pwm1 < 0) && (pwm2 < 0))                      //judge the moving
direction of balance robot; if go back (PWM the motor voltage is
negative), the number of pulses is negative.
  {
    rpluse = -rpluse;
    lpluse = -lpluse;
  }
  else if ((pwm1 > 0) && (pwm2 > 0))                  // if go back (PWM
the motor voltage is positive) , the number of pulses is positive.
  {
    rpluse = rpluse;
    lpluse = lpluse;
  }
  else if ((pwm1 < 0) && (pwm2 > 0))                  //judge the moving
direction of balance robot; if turn left, right pulse is positive but left
pulse is negative.
  {
    rpluse = rpluse;
    lpluse = -lpluse;
  }
  else if ((pwm1 > 0) && (pwm2 < 0))                //judge the moving
direction of balance robot; if turn right , right pulse is negative but
left pulse is positive.
  {
    rpluse = -rpluse;
    lpluse = lpluse;
```

```
  }
  //entering interrupt per 5ms, pulse will plus
  pulseright += rpluse;
  pulseleft += lpluse;
}
//////////////////////////////////interrupt /////////////////////////////
void DSzhongduan()
{
  sei();  //allow overall interrupt
  countpluse();        //pulse count subfunction
  mpu6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);     //IIC to get
MPU6050 six-axis data ax ay az gx gy gz
  angle_calculate(ax, ay, az, gx, gy, gz, dt, Q_angle, Q_gyro, R_angle,
C_0, K1);      //get the angle and Kalman filtering
  PD();         //PD control of angle loop
  anglePWM();
  cc++;
  if(cc>=8)     //5*8=40, 40ms entering PI count of speed once
  {
    speedpiout();
    cc=0;  // Clear
  }
  turncc++;
  if(turncc>4)       //20ms entering PI count of steering once
  {
    turnspin();
    turncc=0;     //Clear
  }
}
/////////////////////////////////////////////////////////////
//////////////////////////////tilt angle calculation//////////////////////
void angle_calculate(int16_t ax,int16_t ay,int16_t az,int16_t gx,int16_t
gy,int16_t gz,float dt,float Q_angle,float Q_gyro,float R_angle,float
C_0,float K1)
{
  float Angle = -atan2(ay , az) * (180/ PI);           //Radial rotation
angle calculation formula; the negative sign is direction processing.
  Gyro_x = -gx / 131;              //The X-axis angular velocity
calculated by the gyroscope ; the negative sign is the direction
processing.
  Kalman_Filter(Angle, Gyro_x);            // Kalman Filter
  // Z-axis angular velocity
  Gyro_z = -gz / 131;                      //speed of Z-axis
  //accelz = az / 16.4;
  float angleAx = -atan2(ax, az) * (180 / PI); //calculate the included
angle of X-axis
  Gyro_y = -gy / 131.00; //angle speed of Y-axis
  Yiorderfilter(angleAx, Gyro_y); //first-order filtering
}
/////////////////////////////////////////////////////////////////
//////////////////////////////KalmanFilter////////////////////
void Kalman_Filter(double angle_m, double gyro_m)
{
  angle += (gyro_m - q_bias) * dt;          //prior estimate
  angle_err = angle_m - angle;
    Pdot[0] = Q_angle - P[0][1] - P[1][0];    //The differential of the
covariance of the prior estimate error
  Pdot[1] = - P[1][1];
```

```
  Pdot[2] = - P[1][1];
  Pdot[3] = Q_gyro;
    P[0][0] += Pdot[0] * dt;    //The integral of the covariance
differential of the prior estimate error
  P[0][1] += Pdot[1] * dt;
  P[1][0] += Pdot[2] * dt;
  P[1][1] += Pdot[3] * dt;
    //Intermediate variables in matrix multiplication
  PCt_0 = C_0 * P[0][0];
  PCt_1 = C_0 * P[1][0];
  //denominator
  E = R_angle + C_0 * PCt_0;
  //gain value
  K_0 = PCt_0 / E;
  K_1 = PCt_1 / E;
    t_0 = PCt_0;  //Intermediate variables in matrix multiplication
  t_1 = C_0 * P[0][1];
    P[0][0] -= K_0 * t_0;    //the covariance of the prior estimate error
  P[0][1] -= K_0 * t_1;
  P[1][0] -= K_1 * t_0;
  P[1][1] -= K_1 * t_1;
    q_bias += K_1 * angle_err;    //posterior estimate
  angle_speed = gyro_m - q_bias;   //The differential of the output value;
get the optimal angular velocity
  angle += K_0 * angle_err; ////posterior estimate; get the optimal
angular velocity
}
//////////////////////first-order filtering/////////////////
void Yiorderfilter(float angle_m, float gyro_m)
{
  angleY_one = K1 * angle_m + (1 - K1) * (angleY_one + gyro_m * dt);
}
//////////////////angle PD////////////////////
void PD()
{
  PD_pwm = kp * (angle + angle0) + kd * angle_speed; //PD angle loop
control
}
//////////////////speed PI////////////////////
void speedpiout()
{
  float speeds = (pulseleft + pulseright) * 1.0;      //pulse value of
speed
  pulseright = pulseleft = 0;      //Clear
  speeds_filterold *= 0.7;         //first-order complementary filtering
  speeds_filter = speeds_filterold + speeds * 0.3;
  speeds_filterold = speeds_filter;
  positions += speeds_filter;
  positions += front;             //Forward control fusion
  positions += back;              //backward control fusion
  positions = constrain(positions, -3550,3550);    //Anti-integral
saturation
  PI_pwm = ki_speed * (setp0 - positions) + kp_speed * (setp0 -
speeds_filter);      //speed loop controlling PI
}
//////////////////speed PI////////////////////
////////////////////////////steering/////////////////////////////////
void turnspin()
{
```

```
  int flag = 0;       //
  float turnspeed = 0;
  float rotationratio = 0;
    if (left == 1 || right == 1)
  {
    if (flag == 0)                                  //judge the speed before
turning; increase the car's flexibility.
    {
      turnspeed = ( pulseright + pulseleft);
//current speed of car; pulse expression
      flag=1;
    }
    if (turnspeed < 0)                                    //Absolute value of
the car's current speed
    {
      turnspeed = -turnspeed;
    }
    if(left==1||right==1)         //if press left key or right key
    {
     turnmax=3;           //maximum value of turning
     turnmin=-3;          // minimum value of turning
    }
    rotationratio = 5 / turnspeed;          //set the value by speed
    if (rotationratio < 0.5)
    {
      rotationratio = 0.5;
    }
        if (rotationratio > 5)
    {
      rotationratio = 5;
    }
  }
  else
  {
    rotationratio = 0.5;
    flag = 0;
    turnspeed = 0;
  }
  if (left ==1)//plus by direction parameter
  {
    turnout += rotationratio;
  }
  else if (right == 1 )//plus by direction parameter
  {
    turnout -= rotationratio;
  }
  else turnout = 0;
  if (turnout > turnmax)   turnout = turnmax;//the max value setting of
amplitude
  if (turnout < turnmin)   turnout = turnmin;//the min value setting of
amplitude
  Turn_pwm = -turnout * kp_turn - Gyro_z * kd_turn;//The rotation PD
algorithm controls the fusion speed and Z axis rotation positioning
}
/////////////////////////turning//////////////////////////////////
/////////////////////////PWM end value////////////////////////////
void anglePWM()
{
```

```
  pwm2=-PD_pwm - PI_pwm + Turn_pwm;           //assign the end value of
PWM to motor
  pwm1=-PD_pwm - PI_pwm - Turn_pwm;

  if(pwm1>255)             //limit the PWM value not more than 255
  {
    pwm1=255;
  }
  if(pwm1<-255)
  {
    pwm1=-255;
  }
  if(pwm2>255)
  {
    pwm2=255;
  }
  if(pwm2<-255)
  {
    pwm2=-255;
  }
  if(angle>80 || angle<-80)      //if tilt angle is greater than 45° ,
motor will stop.
  {
    pwm1=pwm2=0;
  }
 if(pwm2>=0)         //motor's turning and speed are determined by the
positive or negative of PWM
  {
    digitalWrite(left_L1,LOW);
    digitalWrite(left_L2,HIGH);
    analogWrite(PWM_L,pwm2);
  }
  else
  {
    digitalWrite(left_L1,HIGH);
    digitalWrite(left_L2,LOW);
    analogWrite(PWM_L,-pwm2);
  }
  if(pwm1>=0)
  {
    digitalWrite(right_R1,LOW);
    digitalWrite(right_R2,HIGH);
    analogWrite(PWM_R,pwm1);
  }
  else
  {
    digitalWrite(right_R1,HIGH);
    digitalWrite(right_R2,LOW);
    analogWrite(PWM_R,-pwm1);
  }
}
```

Додаток Є

```
#include <MsTimer2.h>        //internal timer 2
#include <PinChangeInt.h>    //this library can make all pins of arduino
REV4 as external interrupt
#include <MPU6050.h>      //MPU6050 library
```

```
#include <Wire.h>        //IIC communication library
MPU6050 mpu6050;     //Instantiate an MPU6050 object; name mpu6050
int16_t ax, ay, az, gx, gy, gz;     //Define three-axis acceleration,
three-axis gyroscope variables
//TB6612 pins
const int right_R1=8;
const int right_R2=12;
const int PWM_R=10;
const int left_L1=7;
const int left_L2=6;
const int PWM_L=9;
//////////////////////angle parameters/////////////////////////////
float angle_X; //calculate the inclined angle variable of X-axis by
accelerometer
float angle_Y; //calculate the inclined angle variable of Y-axis by
accelerometer
int angle0 = 0; //Actual measured angle (ideally 0 degrees)
float Gyro_x,Gyro_y,Gyro_z;  //Angular angular velocity for gyroscope
calculation
//////////////////////angle parameters/////////////////////////////
//////////////////////Kalman_Filter/////////////////////////////
float Q_angle = 0.001;  //Covariance of gyroscope noise
float Q_gyro = 0.003;    //Covariance of gyroscope drift noise
float R_angle = 0.5;    //Covariance of accelerometer
char C_0 = 1;
float dt = 0.005; //The value of dt is the filter sampling time
float K1 = 0.05; // a function containing the Kalman gain; used to
calculate the deviation of the optimal estimate
float K_0,K_1,t_0,t_1;
float angle_err;
float q_bias;    //gyroscope drift
float accelz = 0;
float angle;
float angleY_one;
float angle_speed;
float Pdot[4] = { 0, 0, 0, 0};
float P[2][2] = {{ 1, 0 }, { 0, 1 }};
float  PCt_0, PCt_1, E;
//////////////////////Kalman_Filter/////////////////////////
//////////////////////PID parameters/////////////////////////////////
double kp = 34.00, ki = 0, kd = 0.62;                    //angle loop
parameter
double kp_speed = 3.60, ki_speed = 0.08, kd_speed = 0;   // speed loop
parameter
double kp_turn = 24.00, ki_turn = 0, kd_turn = 0.08;                  //
steering loop parameter
double setp0 = 0; //angle balance point
int PD_pwm;  //angle output
float pwm1=0,pwm2=0;
//////////////////interrupt speed count/////////////////////////////
#define PinA_left 5  //external interrupt
#define PinA_right 4   //external interrupt
volatile long count_right = 0;//Used to calculate the pulse value
calculated by the Hall encoder (the volatile long type is to ensure that
the value is valid)
volatile long count_left = 0;
int speedcc = 0;
//////////////////////pulse count/////////////////////////
int lz = 0;
```

```
int rz = 0;
int rpluse = 0;
int lpluse = 0;
int pulseright,pulseleft;
////////////////////////////////PI variable
parameter/////////////////////////
float speeds_filterold=0;
float positions=0;
int flag1;
double PI_pwm;
int cc;
int speedout;
float speeds_filter;
//////////////////////////////steering PD//////////////////
int turnmax,turnmin,turnout;
float Turn_pwm = 0;
int zz=0;
int turncc=0;
//Bluetooth//
int front = 0;//backward variable
int back = 0;//forward variable
int left = 0;//turn left
int right = 0;//turn right
char val;
int TT;
void setup()
{
  //set the control pins of motor to OUTPUT
  pinMode(right_R1,OUTPUT);
  pinMode(right_R2,OUTPUT);
  pinMode(left_L1,OUTPUT);
  pinMode(left_L2,OUTPUT);
  pinMode(PWM_R,OUTPUT);
  pinMode(PWM_L,OUTPUT);
  //assign the initial value
  digitalWrite(right_R1,1);
  digitalWrite(right_R2,0);
  digitalWrite(left_L1,0);
  digitalWrite(left_L2,1);
  analogWrite(PWM_R,0);
  analogWrite(PWM_L,0);
  pinMode(PinA_left, INPUT);  //speed encoder input
  pinMode(PinA_right, INPUT);
  // join I2C bus
  Wire.begin();                              //join I2C bus sequence
  Serial.begin(9600);                         //open the serial monitor, set
the baud rate to 9600
  delay(1500);
  mpu6050.initialize();                         //initialize MPU6050
  delay(2);
  //5ms; use timer2 to set timer interruption (note：using timer2 will
affect the PWM output of pin3 pin11)
  MsTimer2::set(5, DSzhongduan);    //5ms;  execute the function
DSzhongduan once
  MsTimer2::start();    //start interrupt
}
void loop()
{
  //Serial.println(angle);
```

```
  //delay(100);
  //Serial.println(PD_pwm);
  //Serial.println(pwm1);
  //Serial.println(pwm2);
  //Serial.print("pulseright = ");
  //Serial.println(pulseright);
  //Serial.print("pulseleft = ");
  //Serial.println(pulseleft);
  //Serial.println(PI_pwm);
  //Serial.println(speeds_filter);
  //Serial.println (positions);
  //Serial.println(Turn_pwm);
  //Serial.println(Gyro_z);
  //Serial.println(Turn_pwm);
    if(Serial.available())
  {
    val = Serial.read();      //assign the value read from serial port to
val
    //Serial.println(val);
    switch(val)             //switch statement
    {
      case 'F': front=250; break;       //if vale equals F, front=250, go
front
      case 'B': back=-250; break;       //go back
      case 'L': left=1; break;    //turn left
      case 'R': right=1; break;                          //turn right
      case 'S': front=0,back=0,left=0,right=0;break;    //stop
      case 'Q': Serial.print(angle);break;    //receiving'D', send the
value of variable angle to APP
      case 'P': kp=kp+0.5,Serial.print(kp);break;
      case 'O': kp=kp-0.5,Serial.print(kp);break;
      case 'I': kd=kd+0.02,Serial.print(kd);break;
      case 'U': kd=kd-0.02,Serial.print(kd);break;
      case 'Y': kp_speed=kp_speed+0.05,Serial.print(kp_speed);break;
      case 'T': kp_speed=kp_speed-0.05,Serial.print(kp_speed);break;
      case 'G': ki_speed=ki_speed+0.01,Serial.print(ki_speed);break;
      case 'H': ki_speed=ki_speed-0.01,Serial.print(ki_speed);break;
      case 'J': kp_turn=kp_turn+0.4,Serial.print(kp_turn);break;
      case 'K': kp_turn=kp_turn-0.4,Serial.print(kp_turn);break;
      case 'N': kd_turn=kd_turn+0.01,Serial.print(kd_turn);break;
      case 'M': kd_turn=kd_turn-0.01,Serial.print(kd_turn);break;
    }
if(val=='F'||val=='B'||val=='L'||val=='R'||val=='S'||val=='Q'||val=='P'||v
al=='O'||val=='I'||val=='U'||val=='Y'||val=='T'||val=='G'||val=='H'||val==
'J'||val=='K'||val=='N'||val=='M')
    {
      TT = angle0;
    }
    else
    {
      TT = val;
      angle0=TT;
    }
    //Serial.println(angle0);
   }
    //external interrupt; used to calculate the wheel speed
```

```
  attachPinChangeInterrupt(PinA_left, Code_left, CHANGE);
//PinA_left Level change triggers external interrupt; execute subfunction
Code_left
  attachPinChangeInterrupt(PinA_right, Code_right, CHANGE);
//PinA_right Level change triggers external interrupt; execute Code_right
}
//////////////////////Hall count//////////////////////////
//left speed encoder count
void Code_left()
{
  count_left ++;
}
//right speed encoder count
void Code_right()
{
  count_right ++;
}
//////////////////////pulse count//////////////////////
void countpluse()
{
  lz = count_left;     //assign the value counted by encoder to lz
  rz = count_right;
  count_left = 0;     //Clear the count quantity
  count_right = 0;
  lpluse = lz;
  rpluse = rz;
  if ((pwm1 < 0) && (pwm2 < 0))                    //judge the car's
moving direction; if backward (PWM /motor voltage is negative), pulse is a
negative number.
  {
    rpluse = -rpluse;
    lpluse = -lpluse;
  }
  else if ((pwm1 > 0) && (pwm2 > 0))                //if backward (PWM
/motor voltage is positive), pulse is a positive number.
  {
    rpluse = rpluse;
    lpluse = lpluse;
  }
  else if ((pwm1 < 0) && (pwm2 > 0))                //judge the car's
moving direction; if turn left, the right pulse is positive; left pulse is
negative.
  {
    rpluse = rpluse;
    lpluse = -lpluse;
  }
  else if ((pwm1 > 0) && (pwm2 < 0))               //judge the car's
moving direction; if turn right, the right pulse is negative ; left pulse
is positive.
  {
    rpluse = -rpluse;
    lpluse = lpluse;
  }

  //entering interrupt per, pulse plus
  pulseright += rpluse;
  pulseleft += lpluse;
}
////////////////////////////////////interrupt////////////////////////////
```

```
void DSzhongduan()
{
  sei();  //allow overall interrupt
  countpluse();        //pulse plus subfunction
  mpu6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);     //IIC to get
MPU6050 six-axis data ax ay az gx gy gz
  angle_calculate(ax, ay, az, gx, gy, gz, dt, Q_angle, Q_gyro, R_angle,
C_0, K1);      //get angle and Kalman filtering
  PD();         // PD control of angle loop
  anglePWM();
  cc++;
  if(cc>=8)     //5*8=40, that is, execute the PI calculation of speed
once per 40ms
  {
    speedpiout();
    cc=0;  //Clear
  }
  turncc++;
  if(turncc>4)       //20ms, that is, execute the PD calculation of
steering once per 40ms
  {
    turnspin();
    turncc=0;     //Clear
  }
}
////////////////////////////////////////////////////////////
//////////////////////////////tilt angle calculation
//////////////////////
void angle_calculate(int16_t ax,int16_t ay,int16_t az,int16_t gx,int16_t
gy,int16_t gz,float dt,float Q_angle,float Q_gyro,float R_angle,float
C_0,float K1)
{
  float Angle = -atan2(ay , az) * (180/ PI);           //Radial rotation
angle calculation formula ; negative sign is direction processing
  Gyro_x = -gx / 131;              //The X-axis angular velocity
calculated by the gyroscope ; the negative sign is the direction
processing
  Kalman_Filter(Angle, Gyro_x);            //Kalman Filter
  //Rotation angle Z-axis parameter
  Gyro_z = -gz / 131;                      //angle speed of Z-axis
  //accelz = az / 16.4;
  float angleAx = -atan2(ax, az) * (180 / PI); //calculate the inclined
angle of X-axis
  Gyro_y = -gy / 131.00; //angle speed of Y-axis
  Yiorderfilter(angleAx, Gyro_y); //first-order filtering
}
////////////////////////////////////////////////////////////////
//////////////////////////////KalmanFilter////////////////////
void Kalman_Filter(double angle_m, double gyro_m)
{
  angle += (gyro_m - q_bias) * dt;          //prior estimate
  angle_err = angle_m - angle;
    Pdot[0] = Q_angle - P[0][1] - P[1][0];    //The differential of the
covariance of the prior estimate error
  Pdot[1] = - P[1][1];
  Pdot[2] = - P[1][1];
  Pdot[3] = Q_gyro;
    P[0][0] += Pdot[0] * dt;    //A priori estimation error covariance
differential integral
```

```
  P[0][1] += Pdot[1] * dt;
  P[1][0] += Pdot[2] * dt;
  P[1][1] += Pdot[3] * dt;
    //Intermediate variables in matrix multiplication
  PCt_0 = C_0 * P[0][0];
  PCt_1 = C_0 * P[1][0];
  //denominator
  E = R_angle + C_0 * PCt_0;
  //gain value
  K_0 = PCt_0 / E;
  K_1 = PCt_1 / E;
    t_0 = PCt_0;  //Intermediate variables in matrix multiplication
  t_1 = C_0 * P[0][1];
    P[0][0] -= K_0 * t_0;    //Posterior estimation error covariance
  P[0][1] -= K_0 * t_1;
  P[1][0] -= K_1 * t_0;
  P[1][1] -= K_1 * t_1;
    q_bias += K_1 * angle_err;    //Posterior estimate
  angle_speed = gyro_m - q_bias;   //The differential of the output value;
get the optimal angular velocity
  angle += K_0 * angle_err; ////Posterior estimation; get the optimal
angle
}
//////////////////////first-order filtering/////////////////
void Yiorderfilter(float angle_m, float gyro_m)
{
  angleY_one = K1 * angle_m + (1 - K1) * (angleY_one + gyro_m * dt);
}
///////////////////angle PD////////////////////
void PD()
{
  PD_pwm = kp * (angle + angle0) + kd * angle_speed; //PD angle loop
control
}
///////////////////speed PI////////////////////
void speedpiout()
{
  float speeds = (pulseleft + pulseright) * 1.0;      //speed; pulse value
  pulseright = pulseleft = 0;      //Clear
  speeds_filterold *= 0.7;         //first-order complementary filtering
  speeds_filter = speeds_filterold + speeds * 0.3;
  speeds_filterold = speeds_filter;
  positions += speeds_filter;
  positions += front;             //Forward control fusion
  positions += back;              //Backward control fusion
  positions = constrain(positions, -3550,3550);    //Anti-integral
saturation
  PI_pwm = ki_speed * (setp0 - positions) + kp_speed * (setp0 -
speeds_filter);      //speed loop control PI
}
///////////////////speed PI////////////////////
////////////////////////////turning////////////////////////////////
void turnspin()
{
  int flag = 0;      //
  float turnspeed = 0;
  float rotationratio = 0;
    if (left == 1 || right == 1)
  {
```

```
    if (flag == 0)                                //judge the current speed
before turning, to increase the flexibility.
    {
      turnspeed = ( pulseright + pulseleft);
//current speed; pulse expression
      flag=1;
    }
    if (turnspeed < 0)                                    //absolute value of
current speed
    {
      turnspeed = -turnspeed;
    }
    if(left==1||right==1)         //if press the left key or right key
    {
     turnmax=5;           //the maximum value of turning
     turnmin=-5;          //the minimum value of turning
    }
    rotationratio = 5 / turnspeed;          //set by the speed of car
    if (rotationratio < 0.5)
    {
      rotationratio = 0.5;
    }
        if (rotationratio > 5)
    {
      rotationratio = 5;
    }
  }
  else
  {
    rotationratio = 0.5;
    flag = 0;
    turnspeed = 0;
  }
  if (left ==1)//add according to orientation parameter
  {
    turnout += rotationratio;
  }
  else if (right == 1 )//add according to orientation parameter
  {
    turnout -= rotationratio;
  }
  else turnout = 0;
  if (turnout > turnmax)   turnout = turnmax;//the max value setting of
amplitude
  if (turnout < turnmin)   turnout = turnmin;//the min value setting of
amplitude
  Turn_pwm = -turnout * kp_turn - Gyro_z * kd_turn;//The rotation PD
algorithm controls the fusion speed and Z axis rotation positioning
}
/////////////////////////////steering//////////////////////////////////
/////////////////////////////PWM end value//////////////////////////////
void anglePWM()
{
  pwm2=-PD_pwm - PI_pwm + Turn_pwm;           //assign the end value of
PWM to motor
  pwm1=-PD_pwm - PI_pwm - Turn_pwm;
    if(pwm1>255)             //limit PWM value not more than 255
  {
    pwm1=255;
```

```
  }
  if(pwm1<-255)
  {
    pwm1=-255;
  }
  if(pwm2>255)
  {
    pwm2=255;
  }
  if(pwm2<-255)
  {
    pwm2=-255;
  }
  if(angle>80 || angle<-80)      // if the tilt angle is greater than 45°
, motor will stop turning.
  {
    pwm1=pwm2=0;
  }
 if(pwm2>=0)          //determine the motor's turning and speed by the
negative and positive of PWM
  {
    digitalWrite(left_L1,LOW);
    digitalWrite(left_L2,HIGH);
    analogWrite(PWM_L,pwm2);
  }
  else
  {
    digitalWrite(left_L1,HIGH);
    digitalWrite(left_L2,LOW);
    analogWrite(PWM_L,-pwm2);
  }
  if(pwm1>=0)
  {
    digitalWrite(right_R1,LOW);
    digitalWrite(right_R2,HIGH);
    analogWrite(PWM_R,pwm1);
  }
  else
  {
    digitalWrite(right_R1,HIGH);
    digitalWrite(right_R2,LOW);
    analogWrite(PWM_R,-pwm1);
  }
}
```

Додаток Ж

```
#include <MsTimer2.h>        //Internal timer2
#include <PinChangeInt.h>    //This library file can make all pins on the
REV4 board as external interrupts.  Define three-axis acceleration, three-
axis gyroscope variables
#include <MPU6050.h>       //MPU6050 Library
#include <Wire.h>         //IIC communication library
MPU6050 mpu6050;     // Instantiate an MPU6050 object; name mpu6050
int16_t ax, ay, az, gx, gy, gz;     //Define three-axis acceleration,
three-axis gyroscope variables
//TB6612 pins definition
const int right_R1=8;
```

```
const int right_R2=12;
const int PWM_R=10;
const int left_L1=7;
const int left_L2=6;
const int PWM_L=9;
const int buz = 11;
const int btn = 13;
//////////////////////angle parameters/////////////////////////////
float angle_X; // calculate the inclined angle variable of X-axis by
accelerometer
float angle_Y; //calculate the inclined angle variable of Y-axis by
accelerometer
float angle0 = 0; //mechanical balance angle (ideally 0 degrees)
float Gyro_x,Gyro_y,Gyro_z;  //Angular angular velocity by gyroscope
calculation
//////////////////////angle parameter/////////////////////////////
//////////////////////Kalman_Filter/////////////////////////////
float Q_angle = 0.001;  //Covariance of gyroscope noise
float Q_gyro = 0.003;    // Covariance of gyroscope drift noise
float R_angle = 0.5;    //Covariance of accelerometer
char C_0 = 1;
float dt = 0.005; //The value of dt is the filter sampling time
float K1 = 0.05; // a function containing the Kalman gain is used to
calculate the deviation of the optimal estimate
float K_0,K_1,t_0,t_1;
float angle_err;
float q_bias;    //gyroscope drift
float accelz = 0;
float angle;
float angleY_one;
float angle_speed;
float Pdot[4] = { 0, 0, 0, 0};
float P[2][2] = {{ 1, 0 }, { 0, 1 }};
float  PCt_0, PCt_1, E;
//////////////////////Kalman_Filter/////////////////////////
//////////////////////PID parameter/////////////////////////////////
double kp = 34, ki = 0, kd = 0.62;                    //angle loop
parameter
double kp_speed = 3.56, ki_speed = 0.072, kd_speed = 0;   // speed loop
parameter
double kp_turn = 24, ki_turn = 0, kd_turn = 0.08;                  //
steering loop parameter
double setp0 = 0; //angle balance point
int PD_pwm;  //angle output
float pwm1=0,pwm2=0;
//////////////////interrupt speed count/////////////////////////////
#define PinA_left 5  //external interrupt
#define PinA_right 4   //external interrupt
volatile long count_right = 0;//Used to calculate the pulse value
calculated by the Hall encoder (the volatile long type is to ensure the
value is valid)
volatile long count_left = 0;
int speedcc = 0;
//////////////////////pulse count/////////////////////////
int lz = 0;
int rz = 0;
int rpluse = 0;
int lpluse = 0;
int pulseright,pulseleft;
```

```
////////////////////////////////PI variable
parameter//////////////////////////
float speeds_filterold=0;
float positions=0;
int flag1;
double PI_pwm;
int cc;
int speedout;
float speeds_filter;
////////////////////////////////turning PD//////////////////
int turnmax,turnmin,turnout;
float Turn_pwm = 0;
int zz=0;
int turncc=0;
//Bluetooth//
int front = 0;//forward variable
int back = 0;//backward
int left = 0;//turn left
int right = 0;//turn right
char val;
int i,button;
void setup()
{
  //set the motor control pins to OUTPUT
  pinMode(right_R1,OUTPUT);
  pinMode(right_R2,OUTPUT);
  pinMode(left_L1,OUTPUT);
  pinMode(left_L2,OUTPUT);
  pinMode(PWM_R,OUTPUT);
  pinMode(PWM_L,OUTPUT);
  //assign the initial state value
  digitalWrite(right_R1,1);
  digitalWrite(right_R2,0);
  digitalWrite(left_L1,0);
  digitalWrite(left_L2,1);
  analogWrite(PWM_R,0);
  analogWrite(PWM_L,0);
  pinMode(PinA_left, INPUT);  //speed encoder input
  pinMode(PinA_right, INPUT);
  pinMode(btn,INPUT);
  pinMode(buz,OUTPUT);
  // join I2C bus
  Wire.begin();                              //join I2C bus sequence
  Serial.begin(9600);                         //open the serial monitor and
set the baud rate to 9600
  delay(1500);
  mpu6050.initialize();                        //initialize MPU6050
  delay(2);
  //5ms; use timer2 to set the timer interrupt (note: using timer2 may
affects the PWM output of pin3 pin11)
  MsTimer2::set(5, DSzhongduan);    //5ms; execute the function
DSzhongduan once
  MsTimer2::start();    //start interrupt
}
 //buzzer
void buzzer()
{
    for(int i=0;i<50;i++)
    {
```

```
    digitalWrite(buz,HIGH);
    delay(1);
    digitalWrite(buz,LOW);
    delay(1);
    }
    delay(50);
    for(int i=0;i<50;i++)
    {
    digitalWrite(buz,HIGH);
    delay(1);
    digitalWrite(buz,LOW);
    delay(1);
    }
}
void loop()
{
  //Serial.println(angle0);
  //Serial.print("angle= ");
  //Serial.println(angle);
  //delay(1);
  //Serial.println(PD_pwm);
  //Serial.println(pwm1);
  //Serial.println(pwm2);
  //Serial.print("pulseright = ");
  //Serial.println(pulseright);
  //Serial.print("pulseleft = ");
  //Serial.println(pulseleft);
  //Serial.println(PI_pwm);
  //Serial.println(speeds_filter);
  //Serial.println (positions);
  //Serial.println(Turn_pwm);
  //Serial.println(Gyro_z);
  //Serial.println(Turn_pwm);
  while(i<1)
  {
    button = digitalRead(btn);
    if(button == 0)
    {
      angle0=-angle;
      //Serial.println(angle0);
      buzzer();
      i++;
    }
  }
  if(Serial.available())
  {
    val = Serial.read();      //assign the value read from the serial port
to val
    //Serial.println(val);
    switch(val)             //switch statement
    {
      case 'F': front=250; break;       //if val equals F, front=250, car
will move forward
      case 'B': back=-250; break;       //go back
      case 'L': left=1; break;    //turn left
      case 'R': right=1; break;                          //turn right
      case 'S': front=0,back=0,left=0,right=0;break;    //stop
      case 'D': Serial.print(angle);break;
    }
```

```
  }
    //external interrupt; used to calculate the wheel speed
  attachPinChangeInterrupt(PinA_left, Code_left, CHANGE);
//PinA_left Level change triggers the external interrupt; execute the
subfunction Code_left
  attachPinChangeInterrupt(PinA_right, Code_right, CHANGE);
//PinA_right Level change triggers the external interrupt; execute the
subfunction Code_right
}
//////////////////////Hall count/////////////////////////
//left speed encoder count
void Code_left()
{
  count_left ++;
}
//right speed encoder count
void Code_right()
{
  count_right ++;
}
//////////////////////pulse count/////////////////////////
void countpluse()
{
  lz = count_left;     //assign the value counted by encoder to lz
  rz = count_right;
  count_left = 0;     //Clear count quantity
  count_right = 0;
  lpluse = lz;
  rpluse = rz;
  if ((pwm1 < 0) && (pwm2 < 0))                    //judge the car's
moving direction; if backward (PWM namely motor voltage is negative),
pulse is a negative number.
  {
    rpluse = -rpluse;
    lpluse = -lpluse;
  }
  else if ((pwm1 > 0) && (pwm2 > 0))                 //if backward (PWM
namely motor voltage is positive), pulse is a positive number.
  {
    rpluse = rpluse;
    lpluse = lpluse;
  }
  else if ((pwm1 < 0) && (pwm2 > 0))                 //judge the car's
moving direction; if turn left, right pulse is a positive number; left
pulse is a negative number.
  {
    rpluse = rpluse;
    lpluse = -lpluse;
  }
  else if ((pwm1 > 0) && (pwm2 < 0))               //judge the car's
moving direction; if turn right, right pulse is a negative number; left
pulse is a positive number.
  {
    rpluse = -rpluse;
    lpluse = lpluse;
  }
  // enter interrupt per 5ms, pulse number plus
  pulseright += rpluse;
  pulseleft += lpluse;
```

```
}
////////////////////////////////interrupt ////////////////////////////
void DSzhongduan()
{
  sei();  //allow overall interrupt
  countpluse();        //pulse plus subfunction
  mpu6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);     //IIC to get
MPU6050 six-axis data  ax ay az gx gy gz
  angle_calculate(ax, ay, az, gx, gy, gz, dt, Q_angle, Q_gyro, R_angle,
C_0, K1);      //get angle and Kalmam filtering
  PD();         //angle loop PD control
  anglePWM();
  cc++;
  if(cc>=8)     //5*8=40, enter PI algorithm of speed per 40ms
  {
    speedpiout();
    cc=0;  //Clear
  }
  turncc++;
  if(turncc>4)       //20ms; enter PD algorithm of steering
  {
    turnspin();
    turncc=0;     //Clear
  }
}
/////////////////////////////////////////////////////////////
////////////////////////////tilt calculation//////////////////////
void angle_calculate(int16_t ax,int16_t ay,int16_t az,int16_t gx,int16_t
gy,int16_t gz,float dt,float Q_angle,float Q_gyro,float R_angle,float
C_0,float K1)
{
  float Angle = -atan2(ay , az) * (180/ PI);           //Radial rotation
angle calculation formula ; negative sign is direction processing
  Gyro_x = -gx / 131;              //The X-axis angular velocity
calculated by the gyroscope;  the negative sign is the direction
processing
  Kalman_Filter(Angle, Gyro_x);            //Kalman Filter
  //rotating angle Z-axis parameter
  Gyro_z = -gz / 131;                      //angle speed of Z-axis
  //accelz = az / 1604;
  float angleAx = -atan2(ax, az) * (180 / PI); //calculate the inclined
angle with x-axis
  Gyro_y = -gy / 131.00; //angle speed of Y-axis
  Yiorderfilter(angleAx, Gyro_y); //first-order filtering
}
/////////////////////////////////////////////////////////////////
////////////////////////////KalmanFilter/////////////////////
void Kalman_Filter(double angle_m, double gyro_m)
{
  angle += (gyro_m - q_bias) * dt;          //prior estimate
  angle_err = angle_m - angle;

  Pdot[0] = Q_angle - P[0][1] - P[1][0];    //The differential of the
covariance of the prior estimate error
  Pdot[1] = - P[1][1];
  Pdot[2] = - P[1][1];
  Pdot[3] = Q_gyro;
    P[0][0] += Pdot[0] * dt;    //The integral of the covariance
differential of the prior estimate error
```

```
  P[0][1] += Pdot[1] * dt;
  P[1][0] += Pdot[2] * dt;
  P[1][1] += Pdot[3] * dt;
    //Intermediate variables in matrix multiplication
  PCt_0 = C_0 * P[0][0];
  PCt_1 = C_0 * P[1][0];
  //denominator
  E = R_angle + C_0 * PCt_0;
  //gain value
  K_0 = PCt_0 / E;
  K_1 = PCt_1 / E;
    t_0 = PCt_0;  //Intermediate variables in matrix multiplication
  t_1 = C_0 * P[0][1];
    P[0][0] -= K_0 * t_0;    //Posterior estimation error covariance
  P[0][1] -= K_0 * t_1;
  P[1][0] -= K_1 * t_0;
  P[1][1] -= K_1 * t_1;
    q_bias += K_1 * angle_err;    //Posterior estimate
  angle_speed = gyro_m - q_bias;   //The differential of the output value
gives the optimal angular velocity
  angle += K_0 * angle_err; ////Posterior estimation; get the optimal
angle
}
//////////////////////first-order filter/////////////////
void Yiorderfilter(float angle_m, float gyro_m)
{
  angleY_one = K1 * angle_m + (1 - K1) * (angleY_one + gyro_m * dt);
}
///////////////////angle PD////////////////////
void PD()
{
  PD_pwm = kp * (angle + angle0) + kd * angle_speed; //PD angle loop
control
}
///////////////////speed PI////////////////////
void speedpiout()
{
  float speeds = (pulseleft + pulseright) * 1.0;      //speed  pulse value
  pulseright = pulseleft = 0;      //clear
  speeds_filterold *= 0.7;         //first-order complementary filtering
  speeds_filter = speeds_filterold + speeds * 0.3;
  speeds_filterold = speeds_filter;
  positions += speeds_filter;
  positions += front;             //Forward control fusion
  positions += back;              //backward control fusion
  positions = constrain(positions, -3550,3550);    //Anti-integral
saturation
  PI_pwm = ki_speed * (setp0 - positions) + kp_speed * (setp0 -
speeds_filter);      //speed loop control PI
}
///////////////////speed PI////////////////////
////////////////////////////turning//////////////////////////////////
void turnspin()
{
  int flag = 0;      //
  float turnspeed = 0;
  float rotationratio = 0;

  if (left == 1 || right == 1)
```

```
  {
    if (flag == 0)                                //judge the speed before
rotate, to increase the flexibility
    {
      turnspeed = ( pulseright + pulseleft);
//current speed ; express in pulse
      flag=1;
    }
    if (turnspeed < 0)                                    //speed absolute
value
    {
      turnspeed = -turnspeed;
    }
    if(left==1||right==1)         //if press left key or right key
    {
     turnmax=3;          //max turning value
     turnmin=-3;         //min turning value
    }
    rotationratio = 5 / turnspeed;          //speed setting value
    if (rotationratio < 0.5)
    {
      rotationratio = 0.5;
    }
    if (rotationratio > 5)
    {
      rotationratio = 5;
    }
  }
  else
  {
    rotationratio = 0.5;
    flag = 0;
    turnspeed = 0;
  }
  if (left ==1)//plus according to direction parameter
  {
    turnout += rotationratio;
  }
  else if (right == 1 )//plus according to direction parameter
  {
    turnout -= rotationratio;
  }
  else turnout = 0;
  if (turnout > turnmax)   turnout = turnmax;//max value of amplitude
  if (turnout < turnmin)   turnout = turnmin;//min value of amplitude
  Turn_pwm = -turnout * kp_turn - Gyro_z * kd_turn;//turning PD algorithm
control
}
/////////////////////////////turning//////////////////////////////////
/////////////////////////////PWM end value////////////////////////////////
void anglePWM()
{
  pwm2=-PD_pwm - PI_pwm + Turn_pwm;           //assign the end value of
PWM to motor
  pwm1=-PD_pwm - PI_pwm - Turn_pwm;

  if(pwm1>255)             //limit PWM value not greater than255
  {
    pwm1=255;
```

```
  }
  if(pwm1<-255)
  {
    pwm1=-255;
  }
  if(pwm2>255)
  {
    pwm2=255;
  }
  if(pwm2<-255)
  {
    pwm2=-255;
  }
  if(angle>45 || angle<-45)      //if tilt angle is greater than 45°,
motor will stop
  {
    pwm1=pwm2=0;
  }
   if(pwm2>=0)          //determine the motor steering and speed by
negative and positive of PWM
  {
    digitalWrite(left_L1,LOW);
    digitalWrite(left_L2,HIGH);
    analogWrite(PWM_L,pwm2);
  }
  else
  {
    digitalWrite(left_L1,HIGH);
    digitalWrite(left_L2,LOW);
    analogWrite(PWM_L,-pwm2);
  }

  if(pwm1>=0)
  {
    digitalWrite(right_R1,LOW);
    digitalWrite(right_R2,HIGH);
    analogWrite(PWM_R,pwm1);
  }
  else
  {
    digitalWrite(right_R1,HIGH);
    digitalWrite(right_R2,LOW);
    analogWrite(PWM_R,-pwm1);
  }
}
```

Додаток 3

УДК 622.733-52

КЕРУВАННЯ РУХОМ ПЕРЕВЕРНУТОГО МАЯТНИКА: РОЗРОБКА УСТАНОВКИ, ІДЕНТИФІКАЦІЯ СИСТЕМИ ТА СИНТЕЗ ОПТИМАЛЬНОГО РЕГУЛЯТОРА ЇЇ РУХУ

Ю. О. РОМАСЕВИЧ, доктор технічних наук, професор
В. С. ЛОВЕЙКІН, доктор технічних наук, завідувач кафедри
О. Ю. ЗАРІВНИЙ, аспірант
О. Г. ОЛЕКСІЙКО, студент
Національний університет біоресурсів і природокористування України
E-mail: romasevichyuriy@ukr.net

https://doi.org/10.31548/dopovidi2022.01.016

Анотація. *У роботі розвинений підхід, який дає змогу виконати синтез оптимальних регуляторів динамічних систем. Він полягає у ідентифікації фізичної моделі динамічної системи (у розглядуваному випадку типу „перевернутий маятник"). Це дало підстави для постановки задачі синтезу регулятора. Такий підхід не вимагає математичної моделі системи у вигляді системи диференціальних рівнянь, що є його перевагою. Однак, для того, щоб скористатись цією перевагою необхідно на етапі ідентифікації системи виконувати оцінку її якості. Такі розрахунки показали обґрунтованість розробленого підходу. Синтез оптимального регулятора було проведено на основі відомої методології, яка передбачає зведення вихідної задачі до задачі безумовної оптимізації функції зі складною топологією. Для цього було використано модифікований метод рою часточок. Експериментальна валідація результатів регулювання показала практично повне досягнення мети регулювання – стабілізації системи із наявністю незначних залишкових коливань фазових координат системи.*

Ключові слова: *керування, маятник, оптимізація, регулятор, ідентифікація*

Актуальність. Значна кількість сучасних пристроїв, які використовуються у різних сферах виробництва, характеризуються досить складною динамікою (у ряді випадків із нелінійними ефектами). До таких пристроїв належать літальні апарати (зокрема, квадрокоптери), маніпуляційні системи роботів, засоби індивідуального пересування (квадрокоптери, гіроборди) тощо. Синтез оптимальних систем керування для таких систем ускладнюються тим, що більшість (а у найгіршому випадку – всі) параметри, які входять у рівняння руху динамічних систем, невідомі. Тому адекватні математичні моделі таких систем, на основі яких виконується оптимізація керування їхнім рухом, є недоступними розробнику. Таким чином, з'являється важливий клас задач ідентифікації динамічних систем. Розв'язання задачі

ідентифікації дає підґрунтя для подальших розрахунків оптимальних регуляторів руху систем. З практичної точки зору такі регулятори дають змогу найбільш ефективно використовувати наявні ресурси (заряд акумулятора, ресурс електродвигунів та інших виконавчих пристроїв, надійність маніпуляційних систем роботів) при досягненні мети керування (наприклад, стабілізації польоту квадрокоптера).

Аналіз останніх досліджень та публікацій. Одна з тенденцій у сучасній техніці полягає у глибокому поєднанні у одному виробі елементів різної природи. Йдеться про механічні, електричні, електронні та інші компоненти, які, будучи з'єднаними інформаційними потоками, утворюють мехатронну систему (пристрій). Саме такі пристрої можна побачити на вулицях мегаполісів (гіроскутери), при моніторингу різноманітних військових та сільськогосподарських об'єктів (квадрокоптери), у виробничих цехах підприємств машинобудування (промислові роботи) тощо. Зазначимо, що значний клас таких систем описується нелінійними рівняннями, які, до того ж, повинні містити чисельні значення окремих параметрів. У багатьох випадках точне визначення таких параметрів є неможливим. І тому досить активно йдуть пошуки інших шляхів, які б дозволили виконати математичний опис руху динамічних систем. Ці дослідження для розв'язання задач керування у англомовній літературі відомі під назвою „model free control" [1].

Крім того, досить широке розповсюдження градієнтних оптимізаційних методів може стримувати прогрес у частині розв'язку широкого класу задач керування (наприклад, коли оператором керування виступає нейронна мережа [2]). У цьому випадку увагу варто звернути на інші методи, які позбавлені недоліків градієнтних, зокрема, на метод рою часточок [3] та його модифікації [4-6]. Саме такі методи (за виключенням одного випадку) будуть використані у дані роботі.

Мета. Мета роботи полягає у розробці підходу щодо синтезу оптимального регулятора руху динамічної системи типу „перевернутий маятник". Для досягнення поставленої мети необхідно вирішити такі завдання: 1) розробити алгоритмічну частину роботи лабораторної установки типу „перевернутий маятник"; 2) виконати ідентифікацію динамічної системи „перевернутий маятник" та провести оцінку її якості; 3) розрахувати оптимальний регулятор руху динамічної системи; 4) провести експерименти та встановити якість роботи оптимального регулятора руху системи.

Методи. Перший етапом у проведенні наукового дослідження є

розробка лабораторної установки типу „перевернутий маятник". Окремі деталі установки були спроектовані з використанням САПР Solidworks. Кріплення електричного двигуна, підшипників, енкодера, а також жорсткої муфти та вставку було надруковано на 3D-принтері. Зовнішній вигляд установки зображено на рис. 1.

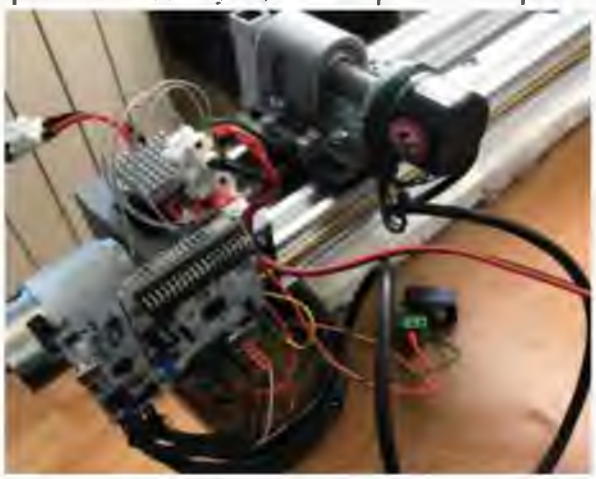


Рис. 1. Зовнішній вигляд лабораторної установки

Основні технічні характеристики установки наведено у табл. 1. Для керування рухом установки та збором даних була використана плата Nucleo F446RE, яка складається із потужного мікроконтролера на 32-бітній архітектурі з високою швидкодією та розвинутою периферією. Розробка коду виконувалась в спеціалізованому середовищі програмування STM32 Cube IDE. Для того, щоб синтезувати регулятор руху динамічної системи „перевернутий маятник" необхідно мати її математичну модель.

1. Основні технічні характеристики установки типу „перевернутий маятник"

Найменування характеристики	Одиниця вимірювання	Величина
Довжина направляючих	см	50
Робочий хід візка		41,7
Довжина маятника від осі обертання		49
Крок ходового гвинта	мм	8
Маса візка	г	100
Маса маятника		60
Максимальні оберти двигуна	об/с	31
Максимальна швидкість переміщення візка	см/с	25
Максимальне осьове зусилля, що діє на візок	Н	50
Момент інерції маятника	кг·мм^2	50

Реальна конструкція установки має значну кількість параметрів, які необхідно відобразити у математичній моделі. Для того, щоб оцінити величини цих та інших параметрів необхідно проводити додаткові експериментальні дослідження. Однак, у даній роботі застосовано інший підхід. Він ґрунтується на властивостях штучних нейронних мереж, які досить часто застосовуються для виконання всякого роду прогнозів. Для того, щоб навчити штучну нейронну мережу було зібрано набір експериментальних даних.

Набір даних для навчання можна представити у такому вигляді:

$$\begin{aligned} &X_i \rightarrow Y_i; \\ &X_i = \left[U_i, \alpha_i, \dot{\alpha}_i, x_i, \dot{x}_i\right]^T; \\ &Y_i = \left[\alpha_{i+1}, \dot{\alpha}_{i+1}, x_{i+1}, \dot{x}_{i+1}\right]^T, \end{aligned} \quad (1)$$

де X_i та Y_i – вектор входу та вектор виходу штучної нейронної мережі для i-того навчального фрейма ($i \in \overline{(1,\ I_{train})}$); I_{train} – довжина даних для навчання I_{train}=1510. Тривалість між окремими вимірами становила 0,01 с. Значення x та α отримували за допомогою фільтра Савіцького-Голея [7], а значення $\dot{\alpha}$ та $\dot{x}$ – за допомогою диференціального фільтра Савіцького-Голея. Для навчання була використана штучна нейронна мережа прямого поширення з одним прихованим шаром і п'ятьма нейронами в ньому. Для навчання використано алгоритм Adam [8]. У результаті його застосування отримано чисельні значення ваг та біасів штучної нейронної мережі, які дають змогу отримати чисельні значення, які близькі до вектора Y_i, при подачі на вхід мережі вектора X_i.

Для того, щоб оцінити якість ідентифікації динамічної системи використаємо підхід, який застосований на аналізі графічних (рис. 2) та числових характеристик.

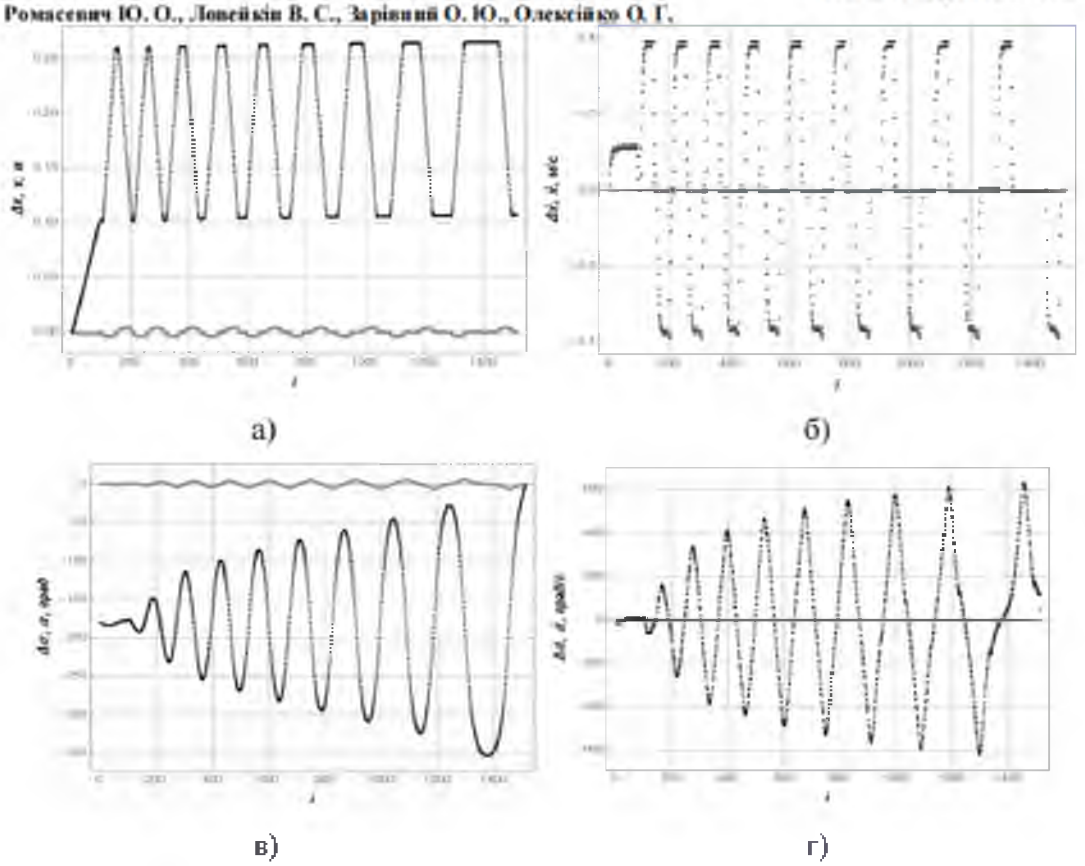


Рис. 2. Графіки тестових даних та похибки передбачення:
а) функція xi; б) функція ẋi; в) функція αi; г) функція α̇i

На рис. 3 чорні точки представляють дані тестування, а сірі відображають абсолютну похибку прогнозу штучної нейронної мережі (відхилення передбачуваного значення від тестового позначається як Δ). Числовими значеннями, які відображають якість прогнозу протягом руху системи, є коефіцієнти варіації, які обчислюються за формулами:

$$E_r = \sqrt{I_{test}^{-1}\sum_i^{I_{test}}\left(r_i^{test}-r_i^{pred}\right)^2}\, I_{test}\left(\sum_i^{I_{test}} r_i^{test}\right)^{-1} 100,\ \ r\in(\alpha,\ \dot{\alpha},\ x,\ \dot{x}), \quad (2)$$

де I_{test} – розмірність тестової вибірки (I_{test}=1515). Розрахунки відповідних величин дають наступні результати: E_α=1,58%, $E_{\dot\alpha}$=0,22%, E_x=1,45%, $E_{\dot x}$=1,79%. З рис. 3 та значень (2) випливає те, що штучна нейронна мережа може бути використана для прогнозу динаміки руху динамічної системи.

Результати. Маючи штучну нейронну мережу, можемо записати постановочну частину задачі синтезу оптимального регулятора:

$$\begin{cases} x_{\frac{t_p}{\Delta t}} \approx 0,\ x_{\frac{t_p}{\Delta t}} - x_{\frac{t_p}{\Delta t}-1} \approx 0; \\ \alpha_{\frac{t_p}{\Delta t}} \approx 0,\ \alpha_{\frac{t_p}{\Delta t}} - \alpha_{\frac{t_p}{\Delta t}-1} \approx 0; \\ \max(x_i) - \min(x_i) \le l_x; \\ \delta_x \delta_1 \sum_{i=1}^{\frac{t_p}{\Delta t}} |x_i| + \delta_\alpha \delta_2 \sum_{i=1}^{\frac{t_p}{\Delta t}} \alpha_i + \delta_U (1-\delta_1-\delta_2) \sum_{i=1}^{\frac{t_p}{\Delta t}} U_i \to \min; \\ U_{unt_i} = A_1 x_i + A_2 \dot{x}_i + A_3 \alpha_i + A_4 \dot{\alpha}_i; \\ U_i = \begin{cases} A_1 x_i + A_2 \dot{x}_i + A_3 \alpha_i + A_4 \dot{\alpha}_i, & якщо -U_{max} \le A_1 x_i + A_2 \dot{x}_i + A_3 \alpha_i + A_4 \dot{\alpha}_i \le U_{max}; \\ -U_{max}, & якщо -U_{max} > A_1 x_i + A_2 \dot{x}_i + A_3 \alpha_i + A_4 \dot{\alpha}_i; \\ U_{max}, & якщо U_{max} < A_1 x_i + A_2 \dot{x}_i + A_3 \alpha_i + A_4 \dot{\alpha}_i; \end{cases} \\ \mathrm{X}_{i+1} = F(\mathrm{X}_i, U_i),\ \mathrm{X}_i = [x_i, \dot{x}_i, \alpha_i, \dot{\alpha}_i]^{\mathrm{T}}, \end{cases} \quad (3)$$

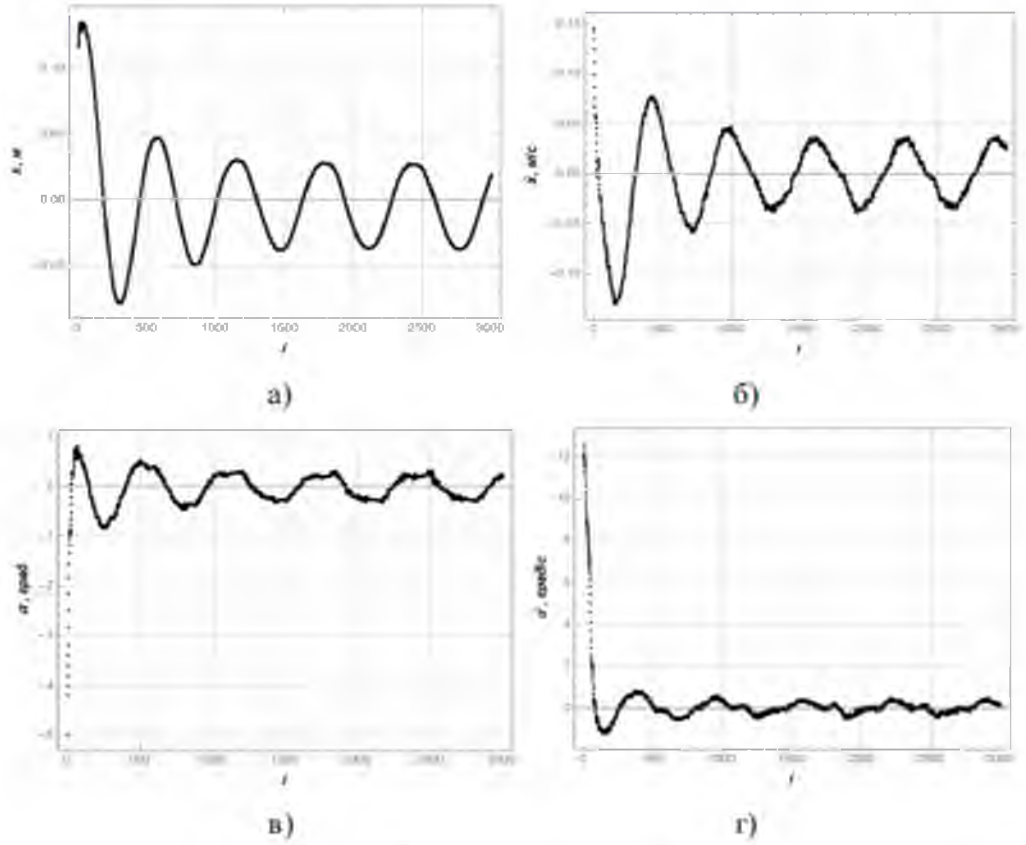
де t_p – тривалість регулювання (виходу системи у окіл уставки – нестійкого положення рівноваги маятника); Δt – крок дискретизації (0,01 с); l_x – гранично допустиме положення візка, яке обґрунтовано виходячи з можливостей переміщенні візка по напрямним установки (l_x=0,2 м); δ_1, δ_2 – вагові коефіцієнти, які показують важливість мінімізації суми абсолютних похибок лінійного та кутового положення ланок установки відповідно (δ_1=0,2; δ_2=0,3); δ_x, δ_α, δ_U – коефіцієнти, які зводять розмірності окремих компонентів критерію до безрозмірного вигляду; $A_1, \ldots, A_4$ – коефіцієнти регулятора, які необхідно визначити; U_{max} – гранично допустиме значення напруги живлення приводу візка, яке представлено у відсотках скважності ШІМ (U_{max}=90 %); F – оператор, який відповідає роботі навченої штучної нейронної мережі (предиктора), що отримана у попередніх розрахунках.

Для знаходження розв'язку задачі (3) її було зведено до задачі безумовної оптимізації узагальненого критерію як це зроблено у роботі [6]. Для визначення його мінімуму використано модифікований метод рою часточок ME-D-PSO [6]. Вкажемо їхні значення: A_1=–71,95, A_2=175,13, A_3=4,9206, A_4=–0,8184. Для того, щоб оцінити якість роботи синтезованого оптимального регулятора було проведено експериментальні дослідження. Їхня мета полягала у дослідженні впливу незалежних факторів (табл. 2) на якість роботи регулятора, тобто стабілізації руху системи.

2. План проведення експериментів

Незалежні фактори	Рівні незалежних факторів					
Початкове положення візка, м	0,1		0		-0,1	
Початкове положення маятника, град	-5	5	-5	5	-5	5
Номер експерименту	1	2	3	4	5	6

Результати експеримента №1 наведено у графічному вигляді на рис. 4. Тут видно, що протягом перших 500 циклів керування відбувається вихід системи у окіл уставки і коливання біля уставки. Це означає, що ідеальна стабілізація руху системи даним регулятором не забезпечується.



а) б) в) г)

Рис. 3. Графіки функцій, що отримані для першого експерименту: а) функція xi; б) функція ẋi; в) функція αi; г) функція α̇i

Однак, амплітуди залишкових коливань візка та маятникової ланки є незначними. У табл. 3 наведено їхні чисельні значення на п'ятому циклі залишкових коливань динамічної системи.

3. Амплітуди фазових координат на п'ятому циклі коливань

Фазова координата	Номер експерименту					
	1	2	3	4	5	6
Положення візка, м	0,037	0,084	0,064	0,041	0,063	0,037
Швидкість візка, м/с	0,039	0,042	0,068	0,044	0,056	0,046
Положення маятникової ланки, град	0,311	0,679	0,614	0,335	0,532	0,376
Швидкість маятникової ланки, град/с	0,495	0,532	0,754	0,513	0,638	0,581

Аналіз даних, які наведено у табл. 3, показує, що не існує єдиної закономірності впливу рівнів незалежних факторів на амплітуди залишкових коливань фазових координат динамічної системи. Найгірший випадок у роботі регулятора (у сенсі показників з табл. 3) відповідає 2 експерименту. Загалом відхилення положення візка від нульового значення (за виключенням експерименту 2) знаходиться на рівні 63,0…37,0% від початкового відхилення. Аналогічний показник для кута маятникової ланки знаходиться у діапазоні 6,2…13,6%, що набагато краще. Це може бути зумовлене тим, що окремі компоненти з формули регулятора (3) мають більший вплив на рух динамічної системи за рахунок того, що вони формують більші величини напруги живлення приводу візка.

Висновки і перспективи подальших досліджень:

1. для дослідження динаміки руху малоприводної системи типу „перевернутий маятник" розроблено алгоритмічну частину системи керування, яка включає функції керування електродвигуном приводу візка, зчитування даних енкодерів положення візка і маятника, а також відправку пакетів отриманих даних на ПК;

2. на основі зібраних масивів експериментальних даних проведено ідентифікацію моделі лабораторної установки типу „перевернутий маятник" для чого використано технології штучних нейронних мереж. Проведено оцінку якості прогнозу, який дає навчена штучна нейронна мережа (відносні похибки прогнозу для різних фазових координат знаходяться в межах 0,22…1,79%) та встановлено, що її доцільно використовувати як предиктор (математичну модель) динаміки системи „перевернутий маятник";

3. виконано постановку задачі синтезу оптимального регулятора стабілізації системи „перевернутий маятник" та за допомогою модифікованого методу рою часточок знайдено її розв'язок (значення коефіцієнтів регулятора);

4. з метою дослідження якості роботи синтезованого регулятора проведено 6 експериментів. Встановлено, що регулятор успішно виконує переведення систем у окіл нестійкого положення рівноваги із наявністю залишкових коливань фазових координат початкового відхилення (наприклад, амплітуда коливань маятникової ланки складає 6,2…13,6% відносно її початкового відхилення);

5. перспективи подальших досліджень у даному напрямку полягають у вдосконалення наступних методик: 1) збору експериментальних даних; 2) вибору структури штучної нейронної мережі та методу її навчання (з урахуванням критерію навчання); 3) постановки задачі синтезу оптимального регулятора руху динамічної системи та методів її розв'язання; 4) технічних засобів практичної реалізації оптимального регулятора та досконалості алгоритмічної частини.

Список використаних джерел:

1. Fliess M., Join C. Model-free control. 2013 URL: https://arxiv.org/pdf/1305.7085.pdf (дата звернення 11.01.2022 р.)

2. Zhong Y., Huang X., Meng P., Li F. PSO-RBF neural network PID control algorithm of electric gas pressure regulator. *Abstract and Applied Analysis*. 2014. Vol. 2014. Article ID 731368.

3. Kennedy J., Eberhart R.C. Particle swarm optimization. *Proceedings of the 1995 IEEE International Conference on Neural Networks*. 1995. Vol. 4 P. 1942-1948.

4. Yudong Z., Shuihua W., Genlin J. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. *Mathematical Problems in Engineering*. 2015. Article ID 931256.

5. Romasevych Yu., Loveikin V. A Novel Multi-Epoch Particle Swarm Optimization Technique. *Cybernetics and Information Technologies*. 2018. Vol. 18 (3). P. 62–74.

6. Romasevych Y., Loveikin V., Makarets V. Optimal constrained tuning of PI-controllers via a new PSO-based technique. *International journal of swarm intelligence research*. 2020. P. 87-105.

7. Savitzky A., Golay M.J.E. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*. 1964. P. 1627-1639.

8. Kingma D.P., Ba J. Adam: A Method for Stochastic Optimization. *Cornell University Library*. 2014. URL: https://arxiv.org/abs/1412.6980 (дата звернення 11.01.2022)

References:

1. Fliess M., Join C. Model-free control. Available at:: https://arxiv.org/pdf/1305.7085.pdf

2. Zhong Y., Huang X., Meng P., Li F. (2014) PSO-RBF neural network PID control algorithm of electric gas pressure regulator. Abstract and Applied Analysis, Article ID 731368. doi: 10.1155/2014/731368

3. Kennedy J., Eberhart R.C. (1995) Particle swarm optimization. Proceedings of the 1995 IEEE International Conference on Neural Networks, 1942-1948. doi: 10.1109/ICNN.1995.488968

4. Yudong Z., Shuihua W., Genlin J. (2015) A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. Mathematical Problems in Engineering, Article ID 931256. doi: 10.1155/2015/931256

5. Romasevych Yu., Loveikin V. (2018) A Novel Multi-Epoch Particle Swarm Optimization Technique. Cybernetics and Information Technologies, 18 (3), 62–74. doi: 10.2478/cait-2018-0039

6. Romasevych Y., Loveikin V., Makarets V. (2020) Optimal constrained tuning of PI-controllers via a new PSO-based technique. International journal of swarm intelligence research, 11(4), 87-105. doi: 10.4018/IJSIR.2020100104

7. Savitzky A., Golay M.J.E. (1964) Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry, 1627-1639. doi: 10.1021/ac60214a047

8. Kingma D.P., Ba J. Adam: A (2014). Method for Stochastic Optimization. Cornell University Library. Available at: https://arxiv.org/abs/1412.6980

MOTION CONTROL OF THE INVERTED PENDULUM: DEVELOPMENT OF THE INSTALLATION, IDENTIFICATION OF THE SYSTEM AND SYNTHESIS OF THE OPTIMAL MOTION CONTROLLER

Yu. O. Romasevych, V. S. Loveikin, O. Yu. Zarivny, A. G. Oleksiyiko

Abstract. *In the work, an approach has been developed that allows of synthesizing optimal controllers of dynamic systems. It consists in the identification of a physical model of a dynamical system (in the case under consideration of the "inverted pendulum" type). This provides the ground for the controller synthesis problem statement. This approach does not require a mathematical model of the system in the form of a system of differential equations, which is its advantage. However, in order to use the advantage, it is necessary to estimate the quality of the system identification. Such calculations showed the validity of the developed approach. The synthesis of the optimal controller was carried out on the basis of the well-known methodology, which presupposes the reduction of the original problem to the problem of unconstrained optimization of a function with a complex topology. For this, a modified particle swarm optimization method has been used. Experimental validation of the control results has shown in practice the complete achievement of the control goal – stabilization of the system with the presence of minor residual oscillations of the phase coordinates of the system.*

Keywords: *control, pendulum, optimization, controller, identification*

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ІНСТИТУТ МЕХАНІКИ ТА АВТОМАТИКИ АПВ НААН
ДЕРЖАВНИЙ БІОТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

ЗБІРНИК
ТЕЗ ДОПОВІДЕЙ

IX Міжнародної науково-технічної конференції з нагоди 115-ї річниці від дня народження
доктора технічних наук, професора,
члена-кореспондента ВАСГНІЛ,
віцепрезидента УАСГН
КРАМАРОВА
Володимира Савовича
(1906-1987)

«КРАМАРОВСЬКІ ЧИТАННЯ»

24-25 лютого 2022 року
м. Київ

Таблиця 2.

План повнофакторних експериментів при дослідженні оптимального режиму руху стрічкового конвеєра

Параметр	Значення					
Режим пуску конвеєра	Холостий			Завантажений		
Тривалість наростання частоти напруги живлення, с	1	3	5	1	3	5
Номер експерименту	1	2	3	4	5	6

Кожен експеримент із серії усіх повнофакторних експериментів виконувався з п'ятикратною повторюваністю.

Результати обробки експериментальних даних будуть використані для розробки рекомендацій стосовно синтезу систем оптимального керування рухом стрічковими конвеєрами.

НЕСТІЙКІ ЗАСОБИ ІНДИВІДУАЛЬНОГО ПЕРЕМІЩЕННЯ: ГІРОБОРД І ГІРОСКУТЕР

О. Г. ОЛЕКСІЙКО, магістр
Ю. О. РОМАСЕВИЧ, д.т.н., проф.
Національний університет біоресурсів і природокористування України

В сучасному світі багато людей знайомі з двоколісним самокатом, який має назву «сёгвей», проте далеко ще не всі звикли до його міні-версії. Саме тому перед більшістю людей постає питання: як працює гіроскутер без ручки? Здивування людей цілком зрозуміло, оскільки досить важко уявити, що мініатюрна платформа для ніг з двома колесами з двох боків може мати досконаліше керування, ніж солідний «Segway» із зручною високою ручкою. Проте, це дійсно так: за допомогою хитрих стабілізаторів-гіроскопів виробникам вдалося створити механізм, який може самобалансуватися та піддається керуванню лише за рахунок нахилів користувача.

Вже декілька років гіроборди застосовують чимало прихильників, оскільки вони стали одним із найзручніших способів переміщення в межах міських кварталів. Прогрес в цьому напрямку не стоїть на місті, вони стали більш потужними, що в свою чергу спонукало їх стати невід'ємною частиною життя людей, а саме, вони перемістились з категорії "для відпочинку" в категорію "для роботи". Універсальність і мобільність – ось головні переваги гіробордів.

Що ж таке гіроборд? Гіроборд – це індивідуальний міні-транспортний засіб, корпус якого складається з горизонтальної платформи з рухомими складовими частинами. Платформа функціонує на базі коліс, оснащених двигунами. Керування відбувається за рахунок поворотів і нахилів тіла, райдер (людина, яка переміщується на гіроскутері) нахиляється і частина платформи приймає необхідне для руху положення.

Більшість гіроскопів виробники завжди забезпечують комплектом плат, що керують даним пристроєм. Ці пристрої відповідають за самобалансування гіроскопа, фіксують кут нахилу, дозволяють легко і без проблем переміщуватись по дорожньому покриттю.

Принцип дії гіроскутера такий: коли система включається, вона автоматично виконує балансування, та визначає центр ваги і приймає оптимальне для людини положення, нахилом вперед або назад райдер задає напрям руху, для виконання повороту – змішує центр ваги (за допомогою натискання ногою) на ту сторону конструкції, куди треба виконати поворот. Датчики гіроскутера постійно виконують збір інформації, фіксуючи усі зміни, що відбуваються під час руху. Функціональність усіх механізмів, а також власне кочення коліс забезпечують двигуни, підключені до акумуляторної батареї.

Візуально гіроборд є пластиковим горизонтальним бордом з двома яскраво вираженими підставками для ніг і колесами, розташованими по краях конструкції, але на одній осі. На колесах знаходяться вакуумні шини.

Конструкція пристрою є доволі непростою:

- усередині кожного колеса розташовуються двигуни, що живляться від акумулятора;
- по периметру корпусу налічується велика кількість гіроскопічних датчиків;
- управляє системою процесор, що виконує збір і обробку інформації;
- між підставками для сходинок розташований міцний металевий елемент високої вантажопідйомності.

Додатково гіроборди можуть комплектуватися динаміками, світлодіодним підсвічуванням, дисплеєм, кнопками керування. У комплекті постачання засобу переміщення обов'язково йде зарядний пристрій.

Додаток І

Національний університет біоресурсів і природокористування України

Факультет конструювання та дизайну

ЗБІРНИК НАУКОВИХ ПРАЦЬ

«Вісник студентів факультету конструювання та дизайну Національного університету біоресурсів і природокористування України»

Випуск 10

Київ-2022

УДК 656.18

КІЛЬКІСНИЙ АНАЛІЗ НАУКОВО-ТЕХНІЧНИХ ДОКУМЕНТІВ У ГАЛУЗІ РОЗРОБКИ СЕГВЕЇВ

Студент – Олексійко О.Г.

Науковий керівник – д.т.н., проф. Ромпевич Ю.О.

Сучасний світ сповнений новітніми технологіями. На сьогоднішній день досить популярними альтернативними видами переміщення вважаються гіроборд, електробайк, самокати та різні електро- платформи, за допомогою яких людина може перемішуватися на певну дистанцію досить швидко та комфортно. Сегвей – електроскутер, якому не потрібне ні кермо, ні тальма. Це унікальний пристрій який при нахилі корпуса їздця вперед, починає котитися вперед, і чим більший нахил, тим більшу він розвиває швидкість (максимальна – 20 км/год). На нашу думку, він є досить цікавою, мобільною та зручною альтернативою для переміщення містом без зайвого дискомфорту.

Дані, які наведені вище (рис. 1), були взяті з ресурсів Всесвітньої організації інтелектуальної власності (ВОІВ) [1] (спеціалізоване агентство в ООН).

Рисунок 1 – Діаграма, яка показує кількість патентів, що отримані у світі за тематикою розробки сегвеїв

На діаграмі (рис. 1) спостерігаємо досить різку динаміку в 2016 та 2017 роках, в погім незначний спад в 2018 році. З 2019 по 2021 рік спад в цьому напрямку продовжився, але однак це було зв'язано з пандемією COVID-19. Однак в 2021 р. світова економіка почала відновлюватися, що сприяло зростанню зацікавленості науковців до даної теми. Науковці в даному напрямку постійно експериментують та удосконалюють уже створені вироби. Тому так званою „клавішою" стало оснащення сигвея/гіроборда: усередині кожного колеса розташовуються двигуни, що живлять від літій-іонного акумулятора; по периметру корпусу налічується велика кількість гіроскопічних датчиків; управляє системою процесор, що виконує збір і обробку інформації.

Висновок: завдяки даним, які пропонує ВОІВ, було виявлено наскільки актуальний напрямок сфери розробки сигвеїв/гіробордів в світі та яке загальне оснащення використовуються в створенні цих засобів переміщення.

Список використаних джерел:

1. https://www.wipo.int/portal/en/index.html

УДК 669.14.018.25:620.18:539.374

ЗРАЗОК ДЛЯ ДОСЛІДЖЕННЯ ПЕРЕТВОРЕНЬ В ВИЛИВКАХ

Студент – Данілов А.В.

Науковий керівник – д.т.н., проф. Афтанділянц Є.Г.

Фазові перетворення у виливках визначаються, як правило методами термічного аналізу, при якому реєструється залежність зміни параметрів зразка від температури. Отримана залежність дозволяє визначати термокінетичні параметри фазових перетворень при нагріванні та охолодженні сталі.