Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Мікропроцесорна система з реалізацією жестового управління»

08-54.БКР.012.00.000 ПЗ

Виконав: студент 4 курсу, групи 1КІ-23мс
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма — «Комп'ютерна інженерія»
Лубко В.Р. _____
Керівник д.т.н., проф. каф. ОТ
_____ Мартинюк Т.Б.
Рецензент к.т.н., доц. каф. ПЗ
Катєльніков Д. І. _____

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

«19» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо—кваліфікаційний рівень бакалвр
Галузь знань — 12 Інформаційні технології
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н.проф. Азаров О. Д.
«21» березня 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Лубку Валентину Романовичу

1 Тема роботи: «Мікропроцесорна система з реалізацією жестового управління», керівник роботи Мартинюк Тетяна Борисівна, д.т.н., професор кафедри ОТ, затверджено наказом вищого навчального закладу від «20» березня 2025 року №97.

2 Строк подання студентом роботи 13.05.2025

3 Вихідні дані до роботи: технічний опис модуля гіроскопа MPU6050 та модулів бездротового зв'язку HC-05 та HC-06, технології розробки: мова програмування C++ для Arduino, бібліотеки для роботи з протоколом UART, модулем MPU6050 та сервоприводами, цільові платформи системи – Arduino Uno та Arduino Nano, середовище розробки – Arduino IDE.

4 Зміст розрахунково-пояснювальної записки: вступ; аналіз існуючих прототипів та визначення вимог до створюваної системи; розробка й тестування апаратно-програмної системи; висновки.

5 Перелік графічного матеріалу: електрична принципова схема пульта керування, блок-схема основного циклу програмного коду пульта керування, блок-схема основного циклу програмного коду роботизованого маніпулятора.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	д.т.н., проф. каф. ОТ Мартинюк Т.Б.	21.03.25	13.05.25
Нормоконтроль	ас.. каф. ОТ Швець С. І.	14.05.25	30.05.25

7. Дата видачі завдання 21.03.2025 р.

8. Календарний план приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання	Підпис
1	Постановка задачі роботи	21.03.25	
2	Пошук матеріалів та інформаційних джерел	21.03.25— 26.03.25	
3	Огляд та аналіз існуючих рішень	27.03.25— 30.03.25	
4	Проєктування апаратної частини системи	31.03.25—6.04.25	
5	Розробка програмної складової системи	7.04.25—19.04.25	
6	Компонування та тестування прототипу	20.04.25— 27.04.25	
7	Оформлення пояснювальної записки та ілюстративного матеріалу	28.04.25— 11.05.25	
8	Перевірка якості виконання бакалаврської кваліфікаційної роботи та усунення недоліків	13.05.25	

Студент Валентин ЛУБКО

Керівник роботи д.т.н., проф. Тетяна МАРТИНЮК

АНОТАЦІЯ

УДК 681.004

Лубко В.Р. Мікропроцесорна система з реалізацією жестового управління. Бакалаврська кваліфікаційна робота зі спеціальності 123 «Комп'ютерна інженерія», освітня програма «Комп'ютерна інженерія». Вінниця: ВНТУ, 2025. 68 с.

На укр. мові. Бібліогр.: 20 назв; рис.: 19; табл.: 1.

У бакалаврській кваліфікаційній роботі розроблено систему бездротового керування роботизованим маніпулятором на основі жестів оператора. Основу апаратної частини системи складають мікроконтролери Arduino Nano та Arduino Uno, модулі гіроскопів і акселерометрів MPU6050, Bluetooth-модулі HC-05/HC-06 та чотири сервоприводи. Сигнали жестів зчитуються за допомогою гіроскопічних датчиків, обробляються мікроконтролером та передаються на маніпулятор через Bluetooth-з'єднання.

Для передачі даних реалізовано власний протокол із контрольними сумами, що забезпечує надійність зв'язку. Приймач розпізнає та інтерпретує пакети команд, які використовуються для керування положенням сервомеханізмів. У конструкції передбачено автономне живлення, індикацію стану передачі та функцію тимчасової паузи без розриву з'єднання. Система працює в режимі реального часу, забезпечуючи точне, плавне та стабільне керування маніпулятором на відстані.

Робота демонструє можливості застосування жестового керування в галузях робототехніки, побутової автоматизації та допоміжних пристроїв для людей з обмеженими можливостями.

Ключові слова: Arduino, жестове керування, MPU6050, Bluetooth, HC-05, HC-06, сервопривід, маніпулятор, бездротовий зв'язок, прототипування.

ABSTRACT

Lubko V.R. Microprocessor System with Gesture Control Implementation. Bachelor’s qualification project in specialty 123 "Computer Engineering", educational program "Computer Engineering". Vinnytsia: VNTU, 2025. 68 p.

In Ukrainian. Bibliography: 20 titles; figures: 19; tables: 1.

This bachelor’s qualification project presents the development of a wireless gesture-controlled robotic manipulator system. The hardware part of the system is based on Arduino Nano and Arduino Uno microcontrollers, MPU6050 gyroscope and accelerometer modules, HC-05/HC-06 Bluetooth modules, and four servo motors. Gesture signals are captured by gyroscopic sensors, processed by the microcontroller, and transmitted to the manipulator via Bluetooth connection.

A custom data transmission protocol with checksum verification was implemented to ensure reliable communication. The receiver decodes and interprets command packets that determine the positions of the servomechanisms. The system includes autonomous power supply, transmission status indication, and a pause mode that allows temporary halting of data transfer without disconnecting the Bluetooth link. Operating in real time, the system enables precise, smooth, and stable remote control of the manipulator.

The project demonstrates the potential of gesture-based control systems for applications in robotics, household automation, and assistive technologies for people with disabilities.

Keywords: Arduino, gesture control, MPU6050, Bluetooth, HC-05, HC-06, servo motor, manipulator, wireless communication, prototyping.

ЗМІСТ

ВСТУП

У сучасному світі роботизовані системи відіграють дедалі важливішу роль у різних сферах діяльності, починаючи від промисловості й закінчуючи медициною та дослідницькими місіями. Завдяки своїм можливостям вони можуть виконувати завдання в умовах, небезпечних або недоступних для людини, наприклад, у зонах підвищеного радіаційного фону, на глибоководних об’єктах або навіть у відкритому космосі. Для ефективного використання таких систем необхідно забезпечити інтуїтивно зрозуміле та гнучке керування. Одним із перспективних підходів є використання жестового керування, що дозволяє передавати команди роботизованій системі за допомогою рухів руки оператора.

Жестове керування відкриває нові можливості в області дистанційного управління роботизованими системами. Воно дозволяє зменшити час навчання оператора, зробити керування більш природним та зменшити кількість допоміжних елементів керування. Такий підхід особливо корисний у випадках, коли оператор не має можливості користуватися стандартними пультами або клавіатурами, наприклад, під час мобільної роботи або в складних умовах середовища.

Актуальність даної роботи полягає в необхідності дослідження та розробки методів, які забезпечують ефективне, інтуїтивно зрозуміле дистанційне керування без використання складних або дорогих апаратних рішень.

Метою даної роботи є аналіз існуючих методів дистанційного керування роботизованими системами та розробка прототипу жестового керування за допомогою мікроконтролерів Arduino. Запропоноване рішення передбачає наявність двох незалежних мікроконтролерів: один зчитує жести оператора за допомогою акселерометрів, а другий виконує команди, керуючи роботизованим маніпулятором. Комунікація між ними реалізована за допомогою бездротового з’єднання Bluetooth.

Об'єктом дослідження є система дистанційного керування роботизованим маніпулятором, що базується на використанні жестів оператора.

Предметом дослідження є методи та технології реалізації бездротового жестового керування за допомогою акселерометрів і мікроконтролерів Arduino.

Завданням є дослідження переваг та недоліків жестового керування за допомогою розробки прототипу системи керування роботизованим маніпулятором, що реагує на жести оператора.

Методи, за допомогою яких буде досягнуто мети, наступні:

— моделювання — для побудови теоретичних моделей системи жестового керування;

— розробка програмного забезпечення — для реалізації функцій зчитування жестів, бездротової передачі даних та керування маніпулятором;

— експериментальне тестування — для перевірки роботи прототипу в реальних умовах;

— аналіз результатів — для оцінки ефективності запропонованих рішень у порівнянні з існуючими технологіями.

Новизна роботи полягає в інтеграції двох окремих Arduino-пристроїв, один з яких використовується для зчитування жестів за допомогою MPU6050, а другий для безпосереднього керування роботизованим маніпулятором. Комунікація між пристроями реалізується через модулі Bluetooth HC-05/HC-06, що дозволяє створити просту, гнучку та ефективну систему керування без потреби в складному апаратному забезпеченні.

Результати роботи були **апробовані** на Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп’ютерної інженерії, що підтверджує актуальність і значущість виконаного дослідження. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24330

1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Роль роботизованих маніпуляторів у сучасній техніці

Роботизовані маніпулятори стали невід’ємною частиною сучасної техніки завдяки своїй здатності виконувати точні, повторювані та потенційно небезпечні завдання замість людини. Це різновид виконавчих механізмів, здатних імітувати рухи людської руки або виконувати вузькоспеціалізовані дії, які надто складні або небезпечні для безпосередньої участі людини. Їхня універсальність, висока точність, повторюваність рухів і можливість інтеграції з сенсорними системами зробили такі пристрої ключовими в багатьох технологічних галузях. Вони широко застосовуються у промисловості, медицині, дослідницьких місіях, логістиці, будівництві, аграрному секторі та навіть у побуті.

Застосування роботизованих маніпуляторів у промисловості є одним із наймасштабніших і найпоширеніших. У складальних лініях автомобілебудування, на підприємствах електронної промисловості, у виробництві побутової техніки вони виконують операції, які вимагають високої швидкості, точності та надійності: складання, зварювання, фарбування, укладання деталей, сортування. Завдяки здатності працювати безперервно і менш залежно від умов навколишнього середовища, роботизовані системи дозволяють значно знизити виробничі витрати та підвищити рівень безпеки праці. Важливу роль вони відіграють у металургії, хімічному виробництві, важкому машинобудуванні та енергетиці.

У медицині роботизовані маніпулятори використовуються для забезпечення більшої точності, стабільності та контрольованості під час виконання складних хірургічних втручань. Завдяки високій точності позиціювання та мінімізації людського фактора, вони дозволяють проводити операції через малі розрізи з мінімальним ушкодженням навколишніх тканин. Це сприяє зниженню травматичності процедури, зменшенню ризику інфекційних ускладнень, скороченню тривалості післяопераційного періоду та прискоренню реабілітації пацієнтів. Сучасні системи, такі як хірургічний комплекс da Vinci, забезпечують

лікарю можливість точного управління інструментами завдяки інтуїтивно зрозумілому інтерфейсу та високоякісній візуалізації операційного поля.

Роботизовані системи також широко застосовуються в дослідницьких цілях. У космічних місіях маніпулятори встановлюються на бортах супутників, автоматичних станцій та орбітальних комплексів. Зокрема, роботизована рука Canadarm2, встановлена на Міжнародній космічній станції (МКС), використовується для технічного обслуговування зовнішніх модулів, стикування вантажних кораблів та переміщення астронавтів у відкритому космосі. У проектах планетарних досліджень, таких як Mars Rover, маніпулятори виконують надскладні завдання з забору ґрунту та аналізу зразків.

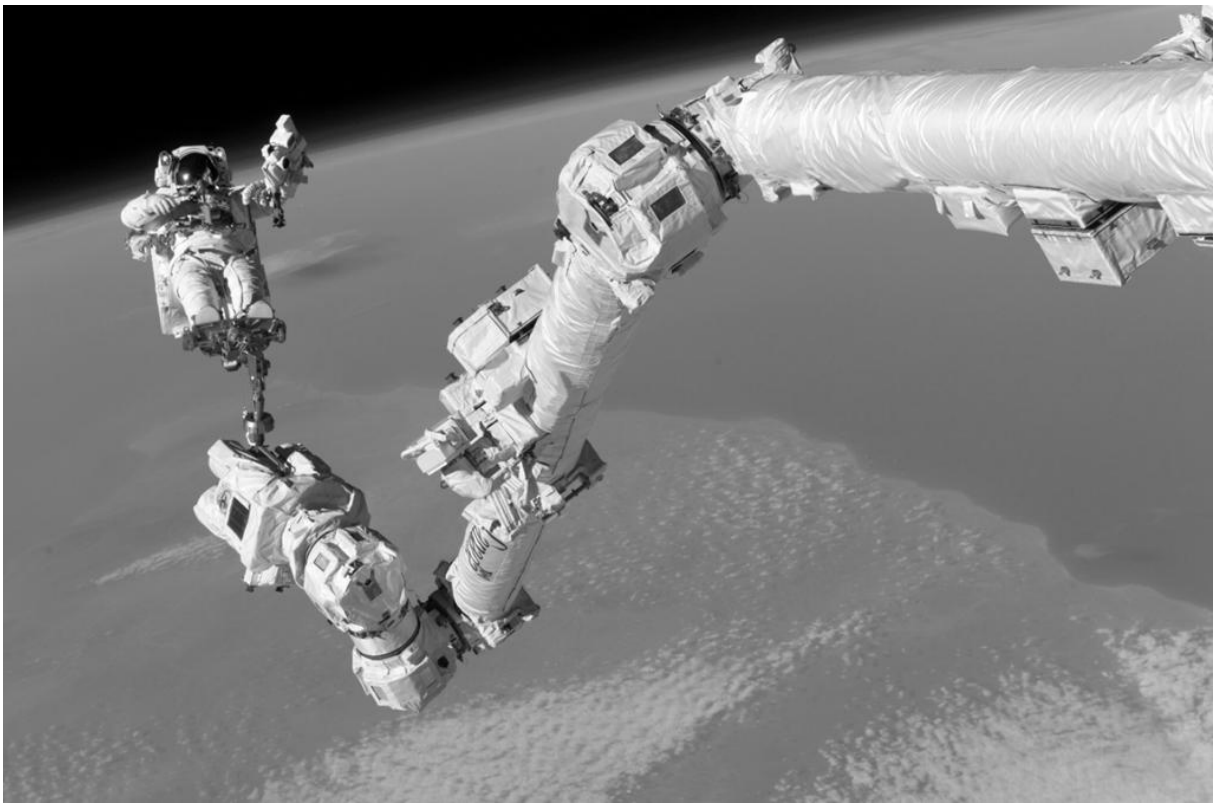

Рисунок 1.1 — Маніпулятор мобільної обслуговуючої системи Canadarm2

Завдяки розвитку штучного інтелекту, комп’ютерного зору та сенсорних технологій, маніпулятори починають відігравати важливу роль і в побутовому середовищі. Наприклад, роботи-доглядальники здатні допомагати людям

похилого віку або людям з обмеженими можливостями в повсякденних діях: подати предмет, приготувати їжу, виконати нескладні господарські завдання.

Таким чином, роботизовані маніпулятори вже зараз є ключовим елементом у розвитку автоматизованих і інтелектуальних систем. Їхня здатність адаптуватися до нових задач, взаємодіяти з навколишнім середовищем і працювати в автономному режимі робить їх важливими не лише в сьогоденні, а й у майбутніх технологічних парадигмах.

1.2 Класифікація роботизованих маніпуляторів

Роботи-маніпулятори, як технічні засоби розширення фізичних можливостей людини, мають широкий спектр конструкцій та функціональних особливостей. Залежно від завдань, які вони виконують, середовища, в якому працюють, а також вимог до точності, швидкодії й сили, маніпулятори можуть істотно відрізнятися за конструкцією та принципами дії.

Умовно можна класифікувати роботизовані маніпулятори за такими критеріями:

— за розміром;
— за потужністю;
— за типом приводу;
— за типом керування;
— за типом зворотного зв’язку.

Щодо класифікації за розміром, існують мініатюрні маніпулятори, які застосовуються в мікрохірургії або ювелірній справі, де необхідна надвисока точність. З іншого боку, у важкій промисловості використовуються великогабаритні маніпулятори, здатні переміщувати тонно-вагові об’єкти, наприклад, у виробництві автомобілів або на будівництві.

Класифікуючи за потужністю, можна побачити, що маніпулятори можуть бути як малопотужними — для роботи з тендітними або легкими об’єктами (наприклад, у лабораторній автоматизації), так і високопотужними, здатними

витримувати значні механічні навантаження. Прикладом останніх є гідравлічні маніпулятори, які активно використовуються у гірничій промисловості.

У залежності від призначення, маніпулятори можуть мати електричний, пневматичний або гідравлічний привід. Електричні забезпечують високу точність і швидкодію, пневматичні — компактність і простоту обслуговування, а гідравлічні — значну силу та витривалість.

За типом керування розрізняють ручне, напівавтоматичне та автоматичне керування. Деякі сучасні системи використовують жести, голосові команди або повністю автономне програмне керування, що значно розширює можливості застосування таких пристроїв у складних або динамічних умовах.

Також вони можуть сильно розрізнятися за типом зворотного зв’язку. Системи зворотного зв’язку дозволяють оператору або програмі контролювати поточний стан маніпулятора. Це може бути реалізовано як через прості засоби спостереження (наприклад, відеокамери), так і через складні набори датчиків, які вимірюють положення, силу, тиск, температуру тощо. Водночас у деяких випадках використання зворотного зв’язку не є обов’язковим — наприклад, при повністю автоматичному або безпосередньому керуванні.

Завдяки такій різноманітності конструктивних і функціональних характеристик, маніпулятори можуть адаптуватися до найширшого спектру завдань – від побутового використання до космічних місій.

1.3 Методи керування роботизованими системами

Ефективність функціонування роботизованого маніпулятора значною мірою залежить від методу його керування. Спосіб взаємодії між оператором або контролером і виконавчим механізмом визначає не лише точність та швидкість виконання завдань, але й сферу застосування маніпулятора.

Основні способи керування можна класифікувати наступним чином:

— ручне керування;

— жестове керування;

— голосове керування;

— автоматичне керування;

— навчання рухам методом демонстрації;

— керування з елементами штучного інтелекту.

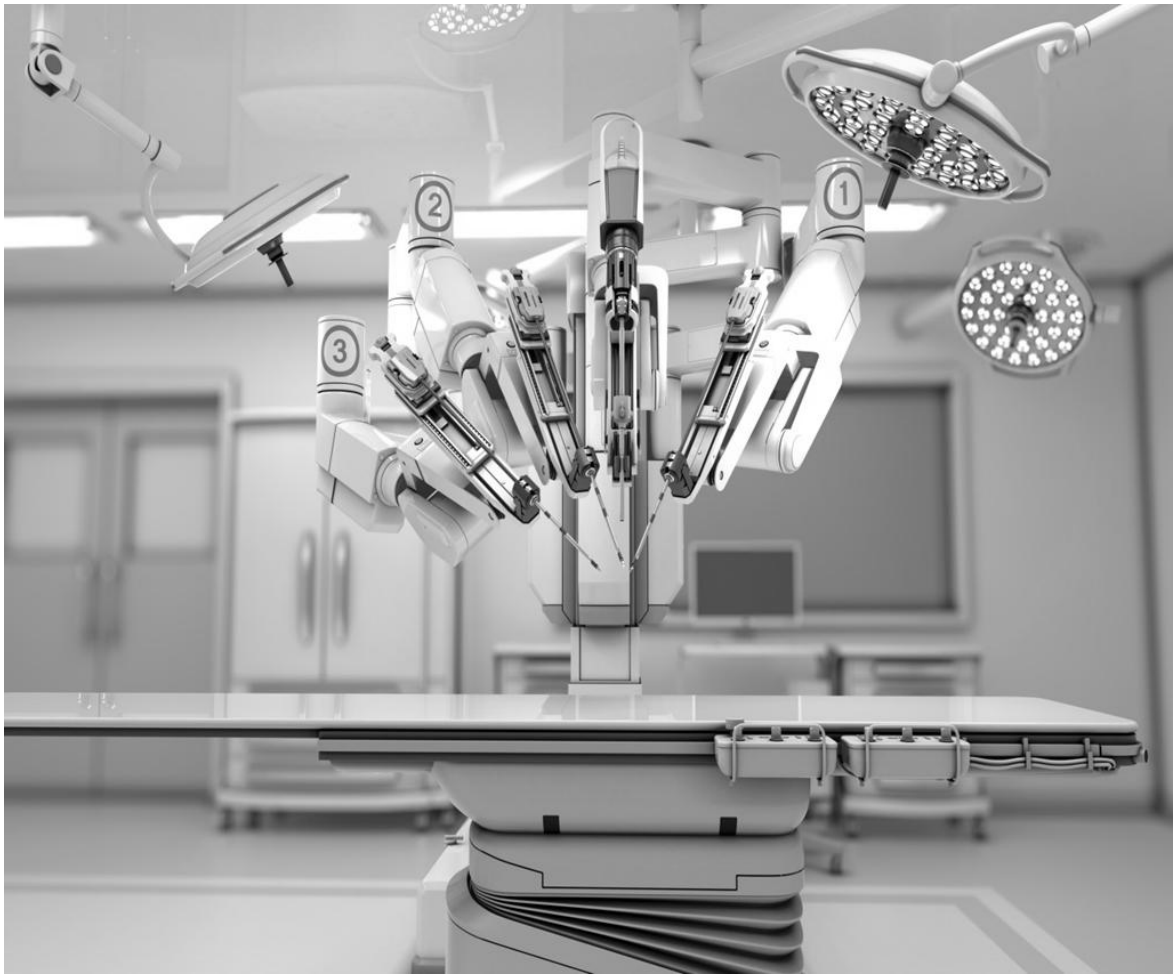

Рисунок 1.2 — Хірургічний роботизований маніпулятор компанії Da Vinci

Ручне керування передбачає пряме управління маніпулятором за допомогою інтерфейсів, таких як джойстики, кнопки, повзунки або аналогові потенціометри. Це найпростіший та найінтуїтивніший спосіб, що часто використовується у побутових пристроях або системах, де потрібен безперервний контроль з боку оператора. Його перевагами є швидке реагування та мінімальні вимоги до складності системи.

Жестове керування реалізується через інтерпретацію рухів оператора, що зчитуються за допомогою камер, інерціальних датчиків або сенсорних рукавичок. Такий підхід забезпечує природну і зручну взаємодію, знижує фізичне

навантаження на користувача та дозволяє використовувати систему навіть у тих випадках, коли традиційні засоби керування є незручними або недоступними.

Голосове керування використовує розпізнавання мови для перетворення словесних команд у дії. Цей спосіб особливо актуальний у системах допомоги людям з обмеженими можливостями або в побутових роботах, де зручність і простота інтерфейсу мають критичне значення. Розвиток технологій мовного аналізу сприяє поширенню цього методу.

Автоматичне керування базується на попередньо заданому алгоритмі, за яким маніпулятор виконує дії без участі оператора. Цей метод широко застосовується в промислових умовах, зокрема на виробничих лініях, де необхідна висока повторюваність і точність операцій при мінімальному людському втручанні.

Навчання методом демонстрації полягає у фізичному "навченні" маніпулятора: оператор у режимі запису один раз виконує всі необхідні рухи, після чого система здатна їх точно повторювати в автономному режимі. Такий підхід поєднує гнучкість ручного керування з перевагами автоматизації.

Керування з використанням штучного інтелекту є найбільш перспективним напрямком. Завдяки застосуванню комп’ютерного зору, алгоритмів машинного навчання та нейромереж, маніпулятори отримують здатність адаптувати свою поведінку до умов довкілля, самостійно планувати траєкторію руху та приймати рішення в непередбачуваних ситуаціях.

Таким чином, вибір методу керування залежить від вимог до точності, автономності, швидкодії, зручності для оператора та вартості реалізації. Часто в одній системі поєднуються кілька способів для досягнення максимального ефекту.

1.4 Жестове керування: принципи та особливості

Жестове керування є інноваційним і перспективним методом взаємодії людини з роботизованими системами. Цей підхід дозволяє операторам використовувати природні жести, що здійснюються руками, для керування

складними технічними системами, такими як роботи-маніпулятори. Системи жестового керування забезпечують безконтактне управління, що знижує фізичні обмеження і дозволяє досягати високої точності при виконанні складних завдань.

Жестове керування базується на виявленні та інтерпретації рухів або поз рук оператора, що передаються системі через спеціальні датчики чи сенсори. Для цього використовуються різноманітні технології, зокрема:

— інфрачервоні датчики, які відслідковують позицію та рухи рук у просторі, зчитуючи інформацію про переміщення;

— гіроскопи та акселерометри, які дозволяють реєструвати рухи, орієнтацію та швидкість рухів оператора;

— камери з комп'ютерним зоровим аналізом, які за допомогою відеоаналізу здатні розпізнавати жести рук і пальців, визначаючи їх точну позицію та виконувані дії.

Переваги жестового керування:

— інтуїтивність;

— безконтактність;

— висока точність і швидкість;

— адаптивність.

Інтуїтивність полягає в тому, що жести є природною формою взаємодії людини з навколишнім середовищем. Їхнє використання у керуванні не потребує додаткового навчання або вивчення складних інтерфейсів. Завдяки цьому навіть користувачі без спеціальної підготовки можуть швидко оволодіти керуванням, що особливо важливо для систем масового або побутового застосування.

Безконтактність жестового керування є його вагомою перевагою у сферах, де фізичний контакт із пристроєм є небажаним або небезпечним. Це актуально, наприклад, у стерильному медичному середовищі, роботах із токсичними речовинами або радіоактивними матеріалами, а також у публічних або автоматизованих системах, де важлива гігієна.

Висока точність і швидкість забезпечуються використанням сучасних сенсорів, здатних миттєво фіксувати положення і рухи тіла з великою

деталізацією. Це дозволяє оперативно реагувати на дії користувача, що критично при виконанні точних технічних операцій, зокрема у хірургії, мікроелектроніці чи керуванні дронами.

Адаптивність систем жестового керування означає можливість налаштування під конкретного користувача або сценарій застосування. Від простих однозначних рухів до складних комбінацій — все це може бути програмно реалізовано та змінено відповідно до потреб. Така гнучкість відкриває широкі перспективи впровадження у різних галузях — від індустрії розваг до професійної робототехніки.

Сфери застосування жестового керування:

— медицина;

— промисловість;

— віртуальна та доповнена реальність;

— космічні дослідження.

У медицині жестове керування активно застосовується в роботизованих хірургічних системах для виконання складних і точних операцій. Завдяки такому типу керування хірург може дистанційно керувати інструментами, зменшуючи навантаження на руки, а також мінімізуючи ризик тремору. Це сприяє підвищенню точності втручання, скороченню тривалості операції та зниженню ризиків для пацієнта.

У промисловості жестове керування використовується для автоматизації виробничих процесів, де критично важливі точність, швидкодія та гнучкість керування. Роботизовані маніпулятори з підтримкою жестів можуть працювати в складних умовах — наприклад, на лініях складання або в зонах підвищеної небезпеки, де безпека людини є пріоритетом.

У сферах віртуальної та доповненої реальності жестове керування дозволяє створювати природні, інтуїтивно зрозумілі інтерфейси для взаємодії з віртуальними об’єктами. Це відкриває нові можливості в ігровій індустрії, освіті, дизайні та тренажерних системах, де важливо досягти ефекту присутності та повного занурення.

У космічних дослідженнях жестове керування використовується для дистанційного контролю роботизованих систем, які виконують технічні завдання в умовах мікрогравітації або на інших планетах. Це особливо важливо в ситуаціях, де безпосередня присутність людини неможлива або небажана через високі ризики чи надмірну віддаленість об'єкта керування.

Обмеження та недоліки жестового керування:

— надійність;

— втома;

— складність інтеграції.

Проблема надійності є одним із ключових викликів для систем жестового керування. Робота таких систем значною мірою залежить від точності датчиків і стабільності середовища. Наявність шумів — як у фізичному, так і в електромагнітному сенсі — може впливати на якість зчитування жестів. Некоректна калібровка сенсорів або перебої в передачі сигналів здатні призвести до помилок у розпізнаванні, що особливо критично в системах, де точність є вирішальною.

Фізична втома оператора також є важливим фактором. Тривале використання жестів для керування вимагає постійної активності м’язів рук і передпліччя. З часом це може призвести до м’язового напруження, втоми та зниження точності виконання жестів. Подібне обмеження унеможливлює тривалу безперервну роботу без перерви, що знижує ефективність у деяких застосуваннях.

Ще одним недоліком є складність інтеграції. Для коректного функціонування системи жестового керування потрібно забезпечити узгоджену роботу апаратних і програмних компонентів, виконати точну калібровку сенсорів, а також створити алгоритми, здатні адаптуватися до особливостей жестів різних користувачів. Це вимагає досвіду, часу й ретельної налагоджувальної роботи, особливо при створенні кастомізованих або універсальних рішень.

1.5 Аналіз існуючих прототипів та комерційних рішень

У сучасному світі робототехніки спостерігається значний прогрес у розробці роботів-маніпуляторів, зокрема тих, що підтримують жестове керування. Це дозволяє операторам взаємодіяти з роботами інтуїтивно, без необхідності використання традиційних пультів чи джойстиків. Розглянемо деякі з доступних прототипів та комерційних рішень у цій галузі.

Рисунок 1.3 — Приклад трьохступеневого маніпулятора

Приклади прототипів роботів-маніпуляторів з жестовим керуванням

— дворівневий маніпулятор з керуванням за допомогою жестів руки;

— маніпулятор з трьома ступенями свободи для операцій pick-and-place;

— маніпулятор для космічних досліджень з адаптивним захоплювачем.

Перший тип прототипу — це дворівневий (двоступеневий) маніпулятор, який реагує на положення руки оператора у просторі. Рухаючи рукою вздовж координатних осей X та Y, оператор може керувати положенням кінцівки маніпулятора. Завдяки використанню сенсорів, система зчитує нахили чи

переміщення руки та транслює їх у відповідні координати, що дозволяє досягти плавного та точного керування у реальному часі.

Другий приклад — маніпулятор з трьома ступенями свободи, оптимізований для виконання типових промислових задач типу "pick-and-place", тобто захоплення і переміщення об’єктів із однієї точки в іншу. Такий пристрій зазвичай інтегрується у виробничі конвеєрні лінії й дозволяє автоматизувати рутинні операції, підвищуючи продуктивність і точність виконання завдань. Жестове керування тут може використовуватись як етап навчання або калібрування дій.

Третій прототип створено для умов космічного простору, де традиційні методи керування обмежені. Оснащений кабельним приводом і адаптивним захоплювачем, він може виконувати делікатні операції у мікрогравітаційному середовищі. Адаптивна конструкція захоплювача дозволяє працювати з об’єктами різної форми і твердості, а дистанційне керування жестами підвищує ефективність і безпеку виконання завдань за межами космічного корабля або станції.

Комерційні рішення у сфері жестового керування

— Leap Motion Controller 2;

— uHand UNO;

— Ziro UI Gloves.

Leap Motion Controller 2 є одним із найвідоміших пристроїв для відстеження рухів рук і пальців з високою точністю. Завдяки інфрачервоним камерам та спеціалізованим алгоритмам, він здатен у режимі реального часу визначати положення кожного пальця користувача у тривимірному просторі. Пристрій широко застосовується у віртуальній та доповненій реальності, дозволяючи створювати природні інтерфейси взаємодії з цифровим середовищем без використання фізичних контролерів.

uHand UNO — це відкритий біонічний роботизований захоплювач, оснащений сенсорами, що забезпечують тактильний зворотний зв’язок. Користувач може не лише керувати рухами захоплювача, але й відчувати взаємодію з об’єктами, які він тримає. Це дозволяє підвищити точність

маніпуляцій, зробити керування більш природним та реалізувати складні завдання, пов’язані з роботою з крихкими або нестандартними предметами.

Ziro UI Gloves — це розумні рукавички, які використовують сенсори для відстеження жестів і трансформації рухів руки у команди для керування роботизованими пристроями. Вони забезпечують безконтактне та інтуїтивне керування, що особливо корисно в навчальних демонстраціях, дослідженнях та експериментальній робототехніці. Завдяки зручному дизайну та простоті інтеграції, рукавички можуть застосовуватись як у професійному середовищі, так і в аматорських проєктах.

2 ПРОЄКТУВАННЯ АПАРАТНОЇ ЧАСТИНИ СИСТЕМИ

2.1 Архітектура апаратної частини системи

Апаратна частина розробленої системи жестового керування складається з двох взаємопов’язаних блоків — пульта керування, який здійснює зчитування рухів оператора та формує відповідні керуючі сигнали, і маніпулятора, що виконує дії на основі отриманих команд. Такий підхід дозволяє реалізувати систему з чітким розподілом функцій між пристроями, що в подальшому спрощує обслуговування, модифікацію та масштабування роботи.

Пульт керування — це автономний пристрій, призначений для виявлення просторових переміщень руки оператора та перетворення їх у цифрову форму. Основними сенсорними елементами є два модулі MPU6050, які поєднують у собі тривісний акселерометр і тривісний гіроскоп, дозволяючи зчитувати орієнтацію руки в просторі з високою точністю. Крім того, для реалізації додаткових функцій управління, зокрема, керування "клешнею" маніпулятора, у пульт інтегровано аналоговий двовісний джойстик із вбудованою кнопкою. Всі ці сигнали обробляються мікроконтролером Arduino Nano, який також виконує функції форматування даних перед передачею.

Передача даних між пультом та маніпулятором відбувається за допомогою модуля бездротового зв’язку Bluetooth HC-05, який забезпечує надійне з'єднання на відстані до 10 метрів у стандартних умовах. Для індикації поточного стану передачі даних у корпус пульта було інтегровано світлодіод, який змінює режим роботи залежно від активності зв’язку. Пульт живиться від літій-йонного акумулятора, що дозволяє повністю відмовитись від дротового живлення й забезпечує мобільність пристрою. Усі компоненти розміщено в компактному корпусі, розробленому спеціально для зручного носіння на руці або передпліччі оператора.

Другий блок системи — роботизований маніпулятор, який реалізовано на базі модифікованого готового набору конструктора. Центральним елементом цього блоку є контролер Arduino Uno, що виконує роль приймача та виконавця

команд. З’єднання з пультом забезпечується за допомогою Bluetooth-модуля HC-06, який приймає вхідні дані та передає їх у контролер. Керування рухами маніпулятора реалізується за допомогою чотирьох сервоприводів SF006C, які утворюють рукоподібну конструкцію з можливістю імітації рухів людської руки.

Для зручності підключення та стабілізації роботи сервоприводів використано шилд Rollarm 1.0, що спрощує з’єднання з платою Arduino Uno та дозволяє мінімізувати кількість дротових з’єднань. До складу конструкції також входять чотири потенціометри, які в стандартній версії набору призначені для ручного управління сервоприводами. У даній роботі вони не використовуються, однак залишаються функціональним резервом для майбутнього розширення системи.

Загальна архітектура апаратної частини роботи спрямована на досягнення максимальної простоти, надійності й функціональності в умовах макетування на базі загальнодоступних і недорогих апаратних компонентів.

2.2 Система зчитування жестів та сенсорні компоненти

Пульт керування являється основним об’єктом даної роботи і виконує наступні функції:

— зчитування даних про рухи оператора;

— з’єднання з маніпулятором;

— передача команд маніпулятору методом бездротового зв’язку (Bluetooth).

Центром всього пристрою є плата Arduino Nano на базі мікроконтролера ATmega328P. Вона опитує модулі гіроскопів, збираючи з них дані, обробляє їх та надсилає маніпулятору за допомогою модуля HC-05. Також вона керує індикаційним світлодіодом, сповіщуючи оператора про стан передачі даних. Плата Arduino Nano є компактною, проте повнофункціональною платформою для вбудованих застосувань. Основні технічні характеристики:

— мікроконтролер — ATmega328P;

— робоча напруга — 5 В;

— кількість цифрових входів/виходів — 14 (з них 6 можуть використовуватися як PWM-виходи);

— аналогові входи — 8;

— тактова частота — 16 МГц;

— пам’ять Flash — 32 КБ (з яких 2 КБ використовуються завантажувачем);

— ОЗП (SRAM) — 2 КБ;

— EEPROM — 1 КБ;

— інтерфейси — UART, SPI, I2C.

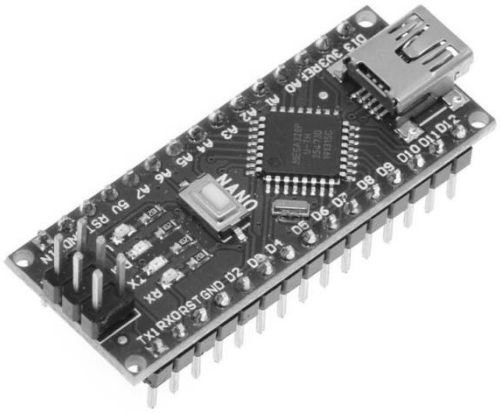

Рисунок 2.1 — Плата Arduino Nano

Для зчитування даних про рух оператора в системі використовуються два датчики MPU6050, кожен із яких є комбінацією тривісного цифрового акселерометра та тривісного цифрового гіроскопа. Завдяки цьому модуль здатний вимірювати як лінійне прискорення, так і кутову швидкість у просторі за трьома координатними осями — умовно X, Y та Z.

Цифровий акселерометр працює на основі мікромеханічних (MEMS) структур, які реагують на інерційні сили, що виникають при русі. Усередині сенсора розташовані мікроскопічні маси, прикріплені до пружин. Під час прискорення ці маси зміщуються, що змінює електричний потенціал на

відповідних конденсаторах. Ці зміни перетворюються в цифрові сигнали, які відповідають значенням прискорення вздовж кожної осі. Таким чином акселерометр дозволяє визначити, як швидко об’єкт змінює своє положення в просторі, або ж розраховувати його нахил відносно вертикалі під дією сили тяжіння.

Гіроскоп у складі MPU6050 вимірює, з якою швидкістю обертається датчик навколо кожної з трьох осей. Він реагує на зміну положення в просторі й дозволяє визначати, як саме поверталась рука оператора. Дані з гіроскопа використовуються для розрахунку кута нахилу, а також для відстеження поворотів з високою точністю.

Оскільки гіроскопи мають властивість до накопичення похибок (дрейфу) під час тривалого використання, в практиці часто використовують дані акселерометра для їх корекції. Завдяки такій комбінації, модуль MPU6050 здатен надавати досить точні та стабільні показники кутової орієнтації.

Обмін даними між мікроконтролером Arduino Nano та модулями MPU6050 здійснюється через інтерфейс I2C — стандартний цифровий двопровідний протокол зв’язку, який широко використовується для підключення периферійних пристроїв. Оскільки Arduino Nano має лише одну апаратну шину I2C, підключення двох ідентичних пристроїв можливе лише за умови, що вони мають різні адреси. MPU6050 дозволяє змінювати свою адресу з допомогою апаратного входу AD0. При подачі на цей пін високого рівня (5 В) модуль приймає альтернативну адресу. Саме цим способом було досягнуто підключення двох датчиків MPU6050 до одного мікроконтролера без конфлікту адрес — обидва датчики використовують одну і ту ж шину, але з різними адресами, що дозволяє по черзі опитувати кожен з них незалежно.

Аналоговий джойстик KY-023 являється модулем, який поєднує в собі два потенціометри та кнопку. Один з потенціометрів, що відповідає за його вісь X, використовується для керування «клешнею» - хапальним механізмом маніпулятора, який керується одним з сервоприводів. Доки джойстик знаходиться в нейтральному стані, механізм не рухається, проте зміщення джойстика в ліву

або праву сторону призведе до стиснення чи роз тиснення клешней хапального механізму в залежності від сторони зміщення. Також у джойстику використовується кнопка, яка замикає контакти якщо натиснути на сам джойстик. Вона виконує функцію перемикання режимів передачі даних, які записані в контролері і індикуються світлодіодом:

— режим керування: мікроконтролер передає команди на маніпулятор, і той, в свою чергу, виконує їх (під час цього режиму світлодіод блимає приблизно два рази на секунду);

— режим очікування: мікроконтролер не передає жодних даних на маніпулятор, проте залишається підключеним (світлодіод постійно світиться).

Режим очікування потрібен для можливості безпечного початку, паузи або закінчення роботи маніпулятора з можливістю швидкого повернення до роботи.

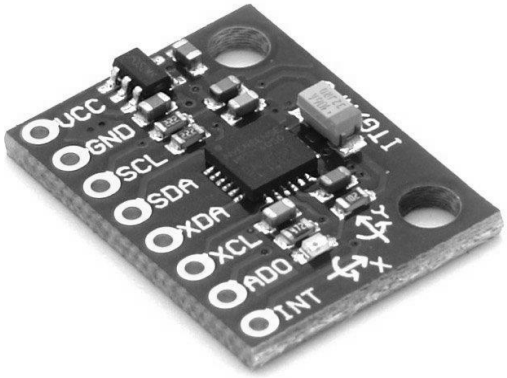

Рисунок 2.2 — Модуль гіроскоп+акселерометр MPU6050

Основні технічні характеристики MPU6050:

— робоча напруга — від 3.3 до 5 В;

— інтерфейс зв’язку — I2C (до 400 кГц);

— вбудований 16-бітний АЦП на кожен канал;

— діапазон вимірювання гіроскопа — ±250, ±500, ±1000, ±2000 °/с;

— діапазон вимірювання акселерометра — ±2g, ±4g, ±8g, ±16g;

— частота дискретизації — до 1 кГц;

— висока стабільність показників і вбудований цифровий фільтр низьких частот (DLPF);

— підтримка вбудованого буфера FIFO та можливість апаратного переривання.

Рисунок 2.3 — Модуль джойстика KY-023

Модуль HC-05 необхідний для забезпечення передачі даних за допомогою технології Bluetooth. Мікроконтролер, отримавши вхідні дані, формує пакет і передає його на модуль по інтерфейсу UART. Отримавши пакет даних, модуль самостійно передає його на інший пристрій Bluetooth, з яким утворено пару, а саме на подібний модуль на маніпуляторі. Характеристики Bluetooth модуля HC-05:

— протокол зв'язку Bluetooth Specification v2.0 + EDR;

— частота GFSK (Gaussian Frequency Shift Keying);

— потужність відправки ≤4dBm, Class 2;

— потужність прийому ≤-84dBm at 0.1% BER;

— швидкість асинхронна 2.1Mbps (Max) / 160 kbps, синхронна 1Mbps / 1Mbps;

— безпека Authentication and encryption;

— профіль Bluetooth serial port;

— живлення + 5VDC 50mA;

— робочі температури -20 ~ +75 C.

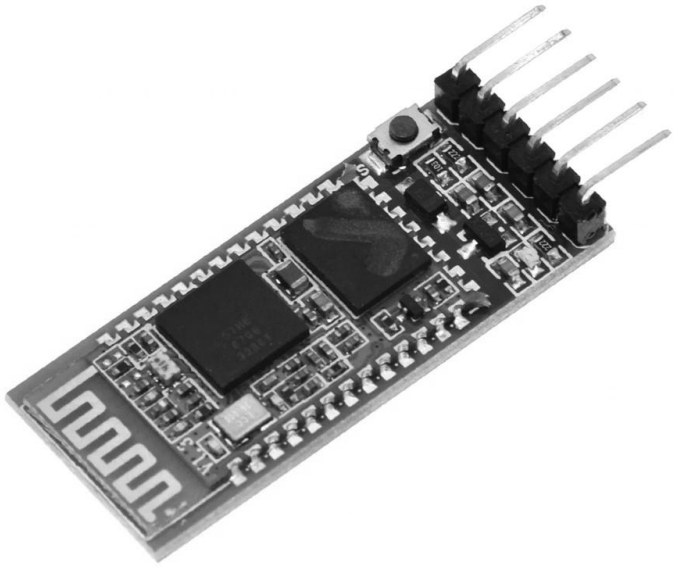

Рисунок 2.4 — Bluetooth модуль HC-05

Для автономності роботи та мобільності пульт оснащений літій-йонним акумулятором. Його номінальна напруга становить 3.7 вольт (при повному заряді вона досягає 4.2 вольт), чого цілком достатньо для роботи мікроконтролера ATmega328P, діапазон робочої напруги якого становить від 1.8 до 5.5 вольт, хоча й не на повну частоту (10 МГц замість 16 при напрузі в діапазоні від 2.7 до 4.5 вольт).

2.3 Роботизований маніпулятор: конструктивні особливості

Роботизований маніпулятор є виконавчим пристроєм системи та слугує для повторення рухів, що здійснює оператор за допомогою пульта. Основу конструкції становить готовий набір для складання навчального маніпулятора, який було модифіковано для підтримки керування через бездротове з’єднання.

Рисунок 2.5 — Роботизований маніпулятор на базі Arduino Uno

Центральним компонентом маніпулятора виступає мікроконтролер Arduino Uno на базі чипа ATmega328P, який приймає сигнали від пульта, обробляє їх та формує відповідні команди для сервоприводів. Основні характеристики Arduino Uno:

— мікроконтролер — ATmega328P;

— тактова частота — 16 МГц;

— оперативна пам’ять (SRAM) — 2 КБ;

— пам’ять програм (Flash) — 32 КБ (з яких 0.5 КБ зайнято завантажувачем);

— EEPROM — 1 КБ;

— кількість цифрових входів/виходів — 14 (з них 6 можуть використовуватися як виходи PWM);

— кількість аналогових входів — 6;

— робоча напруга — 5 В;

— напруга живлення (рекомендована) — від 7 до 12 В;

— максимальний вихідний струм з одного піну — 40 мА.

Для прийому даних з пульта використовується модуль Bluetooth HC-06, що з’єднаний з мікроконтролером через інтерфейс UART. На відміну від HC-05, модуль HC-06 працює лише в режимі приймача (slave), що відповідає вимогам даної системи: ініціалізація з’єднання завжди здійснюється з боку пульта.

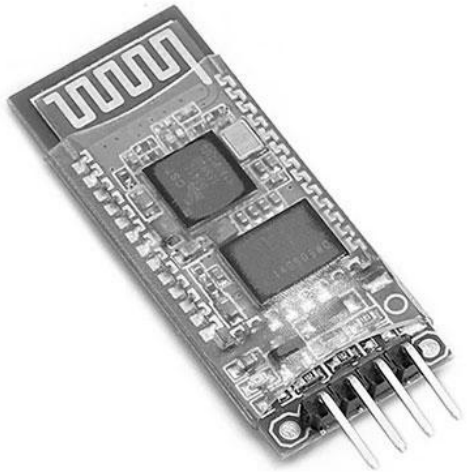

Рисунок 2.6 — Bluetooth модуль HC-06

До Arduino Uno підключені чотири сервоприводи типу SF006C, які відповідають за обертання та згинання секцій маніпулятора:

— перший сервопривод відповідає за обертання всієї конструкції навколо вертикальної осі;

— другий та третій — за згинання “плеча” та “ліктя” маніпулятора;

— четвертий сервопривод керує “клешнею” — хапальним механізмом, що відкривається і закривається відповідно до положення джойстика на пульті.

Рисунок 2.7 — Сервопривод SF006C

Для зручності підключення сервоприводів до мікроконтролера використовується спеціальний шилд Rollarm 1.0, який виконує функцію розгалужувача та дозволяє підключити декілька сервоприводів без потреби додаткових з’єднань.

Конструктивно маніпулятор зібраний з пластикових елементів, що імітують руку з суглобами. У базовому наборі також наявні чотири потенціометри, призначені для прямого аналогового керування сервоприводами. Проте в межах цієї роботи вони не використовуються, оскільки замість них реалізовано дистанційне керування на основі даних з гіроскопів.

Живлення роботизованого маніпулятора реалізується за допомогою зовнішнього джерела постійного струму з номінальною напругою 5 В. Такий підхід дозволяє забезпечити стабільну та надійну роботу всіх компонентів системи, зокрема мікроконтролера Arduino Uno, Bluetooth-модуля HC-06 і чотирьох сервоприводів, що входять до складу виконавчого механізму. З технічної точки зору, плата розширення Rollarm 1.0 передбачає можливість живлення від батареї з номінальною напругою 9 В, що є зручним варіантом у випадках автономного використання системи. Проте на практиці такий спосіб

живлення виявився малоефективним: типові батареї формату «Крона» або їхні аналоги не здатні забезпечити необхідний рівень струму, особливо в умовах одночасного навантаження кількох енергоспоживаючих компонентів. За результатами тестування було встановлено, що при живленні від батареї відбувається суттєве зниження стабільності роботи системи: серво-механізми працюють ривками або взагалі не активуються, а модуль Bluetooth може втрачати з’єднання. З огляду на це, було прийнято рішення відмовитися від 9-вольтових батарей на користь стабілізованого джерела живлення з напругою 5 В, яке здатне постачати струм у межах від 1 до 2 А, що є достатнім для коректної роботи всієї апаратної частини системи.

Таким чином, роботизований маніпулятор являє собою компактну, але функціональну платформу, яка здатна точно повторювати рухи оператора в режимі реального часу, забезпечуючи ефективне жестове дистанційне керування.

2.4 Канали та засоби передачі даних

Між пультом та маніпулятором передача даних відбувається за технологією Bluetooth. Це дозволяє керувати роботом не маючи безпосереднього фізичного контакту, і при цьому тримати досить велику швидкість передачі даних в радіусі кількох метрів. Передача даних відбувається кількома кроками:

— ініціалізація підключення;
— формування пакету даних;
— передача його між модулями;
— дешифрація пакету.

При увімкненні (подачі живлення) на пульт його модуль HC-05, попередньо налаштований в режим master, намагається з'єднатися з модулем HC-06, що знаходиться на маніпуляторі. Якщо з'єднання вдалося, пульт стає готовим до передач даних, а маніпулятор – до їх прийому.

Зчитавши і обробивши показники з гіроскопів мікроконтролер шифрує дані в масив восьмибітних беззнакових цілочисельних значень. Наприклад, масив можна зашифрувати наступним чином:

— перший або два перші елементи мають статичне значення, наприклад, 86 і 76. Таким чином тут позначається початок пакету;

— наступні два елементи складають одне шістнадцятибітне число, таким чином передається значення вісі гіроскопа, що керує першим сервоприводом (на приймачі воно буде дешифруватися назад з двох восьмибітних в одне шістнадцятибітне для подальших обчислень);

— наступні чотири елементи, подібно до двох попередніх, також передають два шістнадцятибітні значення двох вісей гіроскопів, що керують другим та третім сервоприводом відповідно;

— наступний елемент містить значення кута для сервопривода, що керує хапальним механізмом (оскільки він керується джойстиком і за своєю природою не потребує такої великої точності у виставленні кута, як інші, було вирішено передати це значення у восьмибітноу розмірі);

— останні два елементи містять у собі так звану контрольну суму — вона обчислюється передавачем при формуванні пакету і приймачем при його отриманні — таким чином, якщо контрольна сума пакету збігається з тією, яка розрахована приймачем, приймач розуміє, що дані прийшли в цілісності, і їх можна безпечно обробляти.

Сформувавши пакет, мікроконтролер пульта передає його за допомогою UART на модуль HC-05, який, в свою чергу, передає отримані дані на модуль HC-06 на приймачі. Приймач отримує дані, подібно до пульта, по інтерфейсу UART, записує їх у власний подібний масив і перевіряє його. Якщо перші два елементи статичні і збігаються з очікуваними, а останні два співпадають з обчисленою контрольною сумою, починається дешифрація даних – процес, зворотний до того, що виконує пульт. Після дешифрації дані стають повністю готовими до подальшої обробки.

Таблиця 2.1 — Вміст пакета даних, що передає пульт керування

Номер байту	0	1	2	3	4	5	6	7	8	9	10
Призначення	Заголовок		Серв. 1		Серв. 2		Серв. 3		Серв. 4	Контр. сума	
Зразок конвертованого значення	86	76	117	48	109	96	175	200	120	170	204
Зразок реального значення	86	76	30000		28000		45000		120	43724	

3 РОЗРОБКА ПРОГРАМНОЇ СКЛАДОВОЇ СИСТЕМИ

3.1 Середовище розробки та попередня конфігурація

Перед безпосереднім завантаженням програмного коду на мікроконтролери та тестуванням системи необхідно провести попередню конфігурацію окремих апаратних компонентів, зокрема Bluetooth-модуля HC-05, який відповідає за бездротову передачу даних від пульта до маніпулятора. Без правильного налаштування цієї складової коректна робота системи буде неможливою.

Технологія Bluetooth передбачає наявність двох ролей у з’єднанні: Master (ведучий) та Slave (підлеглий). Пристрій у ролі Master ініціює з’єднання, тоді як пристрій у ролі Slave лише відповідає на запити. У нашій системі ці ролі розподіляються наступним чином:

— HC-05, встановлений у пульті керування, виконує роль Master, оскільки саме пульт відповідає за ініціалізацію з’єднання і початок передачі команд;

— HC-06, підключений до маніпулятора, виконує роль Slave. Даний модуль апаратно не підтримує можливість працювати в ролі Master, тому він лише приймає запити від ведучого пристрою.

Щоб HC-05 працював у необхідному режимі, його потрібно перевести в режим AT-команд (режим конфігурації), де він сприймає текстові команди по UART. Для цього перед подачею живлення на модуль потрібно встановити високий логічний рівень на пін EN або KEY (в залежності від модифікації плати), що переводить модуль у AT-режим. У цьому режимі частота блимання індикаторного світлодіода змінюється (близько одного разу на секунду замість звичайного швидкого миготіння), що слугує візуальним підтвердженням переходу в режим конфігурації.

Для обміну командами з модулем HC-05 зручно використовувати послідовний монітор Arduino IDE, оскільки плата Arduino Nano вже має вбудований USB-UART перетворювач. Важливо, щоб швидкість передачі даних

(baud rate) була встановлена на 38400 бод, оскільки саме ця швидкість зазвичай є стандартною в АТ-режимі для більшості модулів HC-05.

Після встановлення з’єднання з модулем та перевірки відповіді на команду «AT» (відповідь «OK») виконуються наступні команди налаштування:

— AT+ROLE=1 — встановлює роль Master;

— AT+BIND=xxxx,xx,xxxxxx — задає MAC-адресу пристрою Slave (тобто модуля HC-06 на маніпуляторі), до якого модуль має автоматично під’єднуватися. Зверніть увагу, що адреса має бути у форматі, який підтримує прошивка модуля (зазвичай з комами);

— AT+CMODE=0 — обмежує з’єднання лише пристроєм із прописаною MAC-адресою. Це підвищує стабільність з’єднання, оскільки модуль не реагуватиме на інші пристрої поблизу.

Цей набір команд дозволяє модулю HC-05 автоматично ініціювати з’єднання з конкретним Slave-модулем при кожному увімкненні. Таким чином, пульт не потребує жодних додаткових налаштувань у процесі роботи, а сам зв’язок встановлюється за кілька секунд після увімкнення живлення обох пристроїв.

Усі ці налаштування зберігаються у енергонезалежній пам’яті модуля HC-05, тому достатньо провести конфігурацію лише один раз. У разі потреби ці параметри можна змінити в майбутньому, знову перевівши модуль у АТ-режим і повторивши процедуру.

Налаштування модуля HC-06 не є обов’язковим, оскільки він не підтримує АТ-режим під час роботи у ролі Slave без спеціального прошивання. Проте він автоматично приймає запити від ведучого пристрою (HC-05), якщо останній має правильну MAC-адресу та знаходиться в зоні досяжності.

Таким чином, правильно проведене попереднє налаштування модулів забезпечує надійний канал зв’язку між пультом керування та маніпулятором, що є критично важливим для стабільної та безпечної роботи системи загалом.

3.2 Програмне забезпечення модуля передавача

Програмний код для передавача, як і для приймача, був написаний в середовищі Arduino IDE. Це один з накращих виборів середовища для роботи з платами Arduino, оскільки воно створене для його використання.

Для роботи програми необхідно задіяти дві сторонні бібліотеки:

— MPU6050.h — ця бібліотека містить в собі вже налаштовані команди та функції для повноцінного керування датчиками MPU6050, що спрощує процес написання коду безпосередньо під потребу;

— SoftwareSerial.h — в даному випадку ця бібліотека використовується для використання програмного UART, оскільки апаратний використовується для дебагу та налаштувань, програмний з’єднується з модулем Bluetooth.

Оскільки в даній роботі використовуються два модулі гіроскопів, потрібно створити два об’єкти класу MPU6050, вписавши в його конструкторі i2c адресу відповідного модуля:

```
MPU6050 mpuHand(0x69);
MPU6050 mpuShoulder(0x68);
```

Також необхідно створити об’єкт класу SoftwareSerial, в його конструкторі вписуються номери пінів, які виконуватимуть функцію RX (зчитування) та TX (передача):

```
SoftwareSerial BTSerial(2, 3);
```

Вся програма використовує для роботи дві глобальні змінні:

— dataToSend – масив з одинадцяти восьмибітних беззнакових цілочисельних змінних (він містить у собі пакет даних, який пізніше надсилатиметься приймачу);

— clawAngle – також восьмибітна беззнакова цілочисельна змінна, яку змінює положення джойстика (вона відповідає за керування хапальним механізмом, а саме кутом, на який буде повернутий його сервопривод);

— sending – змінна типу bool, що визначає, в якому режимі перебуває зараз передавач – передавання даних чи очікування.

Функція startButton() повертає значення типу uint8 і виконує функцію антидребезгу кнопки джойстика. Порівнюючи поточний та попередній стан кнопки, він визначає саме момент її натискання. Це дозволяє уникнути багаторазового спрацювання її функцій в один момент натискання.

```
uint8_t startButton() {
  bool currentRawState = digitalRead(CTRL_BTN);
  if (currentRawState != prevRawState) {
    lastDebounceTime = millis();
    prevRawState = currentRawState;
  }
  if ((millis() - lastDebounceTime) > DEBOUNCE_DELAY) {
    if (prevDebouncedState == HIGH && currentRawState == LOW) {
      pressRegistered = true;
    } else {
      pressRegistered = false;
    }
    prevDebouncedState = currentRawState;
  } else {
    pressRegistered = false; // всередині дребезгу
  }
  return pressRegistered ? 1 : 0;
}
```

Функція SetClawAngle() виконує зчитування даних з однієї вісі джойстика і відповідно змінює значення змінної clawAngle, що дозволяє керувати хапальним механізмом.

```
void SetClawAngle(){
  int potValue = analogRead(POT_PIN);
    if (potValue > 520) {
      clawAngle+=5;
    } else if (potValue < 500) {
      clawAngle-=5;
    }
    clawAngle = constrain(clawAngle, 10, 170);
}
```

Функція CRC_check() виконує найпростішу перевірку контрольної суми – додавання значень усіх елементів масиву значень, з яких складається пакет. Попри простоту, його все ще можна назвати надійним.

```
uint16_t CRC_check(uint8_t *mesagge, uint16_t length) {
  uint16_t crc_out = 0;
  for (int var = 0; var < (length - 2); ++var) {
    crc_out += mesagge[var];
  }
  return crc_out;
}
```

Наступна функція, SetDataToSend(), виконує збір даних до пакету, який надсилатиметься приймачу. Тут виконуються не лише дії запису значень до масиву dataToSend, а й розподіли шістандцятибітних чисел на два восьмибітних, а також обчислення контрольної суми.

```
void SetDataToSend(uint16_t g1p, uint16_t g1r,uint16_t g2r) {
  dataToSend[2] = (g1p >> 8) & 0xFF;
  dataToSend[3] = g1p & 0xFF;
  dataToSend[4] = (g1r >> 8) & 0xFF;
  dataToSend[5] = g1r & 0xFF;
  dataToSend[6] = clawAngle;
  dataToSend[7] = (g2r >> 8) & 0xFF;
  dataToSend[8] = g2r & 0xFF;
  uint16_t crc = CRC_check(dataToSend, 11);
  dataToSend[9] = (crc >> 8) & 0xFF;
  dataToSend[10] = crc & 0xFF;
}
```

Для збереження та обчислення даних, зібраних з датчиків, використовується клас Angles. Він не має конструктора, тому що всі дані, які йому необхідні, він збирає постійно і записує в свої шістнадцятибітні змінні x, y та z, які зберігають значення певного кута нахилу даного датчика. Отримує їх він за допомогою простого методу GetAcceleration().

```
void GetAcceleration(int16_t gx, int16_t gy, int16_t gz){
    x = gx;
    y = gy;
```

```
    z = gz;
}
```

Інші методи цього класу це обробники окремо кожної вісі: GetX(), GetY() та GetZ(). Вони влаштовані майже ідентично і виконують перетворення значень нахилу з діапазону -9000 – 9000, в якому їх видає бібліотека MPU6050.h, в простіший для надсилання 0 – 65535.

```
uint16_t GetX() {
    int16_t ax = x;
    ax = constrain(ax, -16384, 16384);
    float angle = ax / 16384.0;
    if (angle < 0)
      angle = 90 - degrees(acos(angle));
    else
      angle = degrees(acos(-angle)) - 90;
    return map(angle * 100, -9000, 9000, 0, 65535);
   }
```

Функція setup(), як і loop(), є базовою для мікроконтролерів Arduino. Дана функція виконується один раз на початку роботи пристрою, тому тут зазвичай виконується ініціалізація всіх необхідних компонентів.

```
void setup() {
  Wire.begin();
  Serial.begin(9600);
  BTSerial.begin(9600);
  mpuHand.initialize();
  mpuShoulder.initialize();
  pinMode(POT_PIN, INPUT);
  pinMode(CTRL_BTN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, HIGH);
}
```

Функція loop() є головною в коді, оскільки відображає основний цикл роботи всієї програми. Більшість дій викликаються у ній, а саме:

— створення об’єктів класу Angles - hand і shoulder;

— зчитування даних з гіроскопів і запис їх до відповідних об’єктів;

— виконання функцій SetClawAngle() та SetDataToSend();

— обробка події натискання кнопки джойстика з штучною затримкою після передачі першого пакету;

— керування індикаційним світлодіодом та передача даних в залежності від значення змінної sending.

```
void loop() {
  Angles hand, shoulder;
hand.GetAcceleration(mpuHand.getAccelerationX(),mpuHand.getAccelerationY(
),mpuHand.getAccelerationZ());
shoulder.GetAcceleration(mpuShoulder.getAccelerationX(),mpuShoulder.getAcc
elerationY(),mpuShoulder.getAccelerationZ());
  SetClawAngle();
  SetDataToSend(hand.GetX(),hand.GetZ(),shoulder.GetY());
  if(startButton()){
    sending = !sending;
    if(sending){
    BTSerial.write(dataToSend,11);
    delay(1000);
    }
  }
  if(sending){
      digitalWrite(LED_PIN, !digitalRead(LED_PIN));
    BTSerial.write(dataToSend,11);
  }else{
      digitalWrite(LED_PIN, HIGH);
  }
  delay(100);
}
```

Блок-схема, що зображує роботу основного циклу пульта керування, зображена в графічній частині на рисунку В.2.

Таким чином виконується зчитування даних про рухи оператора та передача їх до маніпулятора.

3.3 Програмне забезпечення модуля приймача

Програмний код приймача також було написано в середовищі Arduino IDE, що дозволяє швидко реалізовувати проєкти з використанням Arduino-сумісних

плат. В ньому описано зчитування даних з датчиків по шині i2c, конвертація їх в пакет даних, надсилання даних та перемикання режимів.

Для реалізації функціоналу в роботі задіяна стороння бібліотека Servo.h – стандартна бібліотека для керування сервоприводами за допомогою PWM-сигналу.

```
#include <Servo.h>
```

У програмі оголошуються чотири об’єкти класу Servo, які відповідають за чотири приводи маніпулятора:

```
Servo Servo_0, Servo_1, Servo_2, Servo_3;
```

Дані, що надходять від передавача, зберігаються в масиві buffer з одинадцяти байтів. Це пакет повідомлення, який буде оброблено після проходження перевірки контрольної суми та заголовка.

```
uint8_t buffer[11];
```

Для обробки даних про положення елементів маніпулятора використовується клас Angles, який містить початкові дані про повороти сервоприводів, а також виконує більшість основних функцій.

Метод GetAngles() виконує розшифрування отриманого пакету даних:

— з двох байтів формується значення g1pitch, g1roll, g2roll;

— додатково зчитується значення claw — положення хапального механізму;

— значення кутів масштабуються через map() у потрібний діапазон (від 10 до 170), зручний для керування сервоприводами. Оскільки для керування сервоприводом бібліотекою Servo.h необхідно використовувати значення кута (від 0 до 180 градусів), діапазон від 10 до 170 був обраний з метою запобігання пошкодження сервоприводів.

```
void GetAngles(uint8_t *message) {
```

```
    g1pitch = Rotate(((uint16_t)message[2] << 8) | message[3]);
    g1roll = ((uint16_t)message[4] << 8) | message[5];
    claw = message[6];
    g2roll = ((uint16_t)message[7] << 8) | message[8];
    g1roll = map(g1roll, 0, 65535, 10, 170);
    g2roll = map(g2roll, 0, 32000, 10, 170);
  }
```

Метод Rotate() виконує накопичувальне обертання першого сервопривода лише у разі достатньо великої зміни кута. Це було зроблено для більшої зручності керування:

```
uint8_t Rotate(uint16_t angleGet){
  int16_t angle = map(angleGet, 0, 65535, -100, 100);
  static int16_t prevAngle = angle;
  if (abs((int)angle - (int)prevAngle) > 20){
     rotate += angle/10;
     rotate = constrain(rotate, 10, 170);
    return angle;
  }else{
    angle = prevAngle;
    return prevAngle;
  }
 }
```

Метод MoveServo() подає вже обчислені кути на відповідні сервоприводи:

— Servo_0 — поворот основи;

— Servo_1 — нахил "руки";

— Servo_2 — рух "плеча";

— Servo_3 — хапальний механізм (клешня).

```
void MoveServo() {
    Servo_0.write(rotate);
    Servo_1.write(g1roll);
    Servo_2.write(g2roll);
    Servo_3.write(claw);
  }
```

У функції setup() виконується ініціалізація більшості компонентів програми:

— запуск порту Serial для обміну даних від модуля Bluetooth через UART;

— приєднання сервоприводів до відповідних пінів;

— ініціалізація масиву buffer нулями.

Функція CRC_check() працює ідентично до такої ж в передавачі з метою перевірки отриманих даних.

Функція checking_message() перевіряє, чи пакет починається з значень 86 і 76 — це своєрідний заголовок, який символізує про початок пакету. Проте пакет також варто перевіряти з кінця, для чого виконується обчислення контрольної суми і порівняння її з останніми двома елементами масиву. Якщо всі перевірки пройдено, дані вважаються коректними і допускаються до подальшої обробки.

```
bool checking_message(uint8_t *message) {
   if ((message[0] == 86) && (message[1] == 76)) {
      uint16_t crc_received = ((uint16_t)message[9] << 8) | message[10];
      uint16_t crc_calculated = CRC_check(message, 11);
      return (crc_received == crc_calculated);
   }
   return false;
}
```

Цикл loop() виконує безперервне очікування байтів по UART:

— кожен новий байт додається до кінця буфера, при цьому увесь масив зміщується, видаляючи перший елемент;

— якщо буфер містить коректне повідомлення, його дані розбираються, і відповідні серви отримують команди;

— фінальна затримка delay(1) забезпечує мінімальну паузу між ітераціями, запобігаючи перевантаженню процесора.

```
void loop() {
  if (Serial.available()) {
    uint8_t c = Serial.read();
    for (int i = 0; i < 10; i++) {
       buffer[i] = buffer[i + 1];
    }
    buffer[10] = c;
    if (checking_message(buffer)) {
```

```
        angles.GetAngles(buffer);
        angles.MoveServo();
    }
}
  delay(1);
}
```

Блок-схема, що зображує роботу основного циклу роботизованого маніпулятора, зображена в графічній частині на рисунку В.3.

Таким чином, дана програма реалізує приймання пакетів даних, перевірку їхньої цілісності та передавання відповідних команд на сервоприводи маніпулятора, забезпечуючи повноцінне бездротове керування за допомогою жестів оператора.

4 КОМПОНУВАННЯ ТА ТЕСТУВАННЯ ПРОТОТИПУ

4.1 Дослідження компонентної бази

Для побудови функціонального прототипу системи жестового керування роботизованим маніпулятором було здійснено підбір апаратних компонентів, які б поєднували простоту використання, низьку вартість та технічну сумісність із засобами бездротового зв’язку. Основним елементом обрано двоступеневий маніпулятор, реалізований на базі мікроконтролера Arduino Uno, який, в свою чергу, є однією з найпоширеніших платформ для початкової розробки електронних систем. Його архітектура дозволяє швидко інтегрувати різноманітні модулі та датчики, що значно спрощує процес прототипування.

Вибір саме двоступеневого маніпулятора обумовлений необхідністю зменшити складність системи керування, зокрема кількість ступенів свободи, які потребують окремого контролю. Обмеження кількості керуючих сигналів до чотирьох значно знижує обчислювальне навантаження на систему обробки жестів та забезпечує інтуїтивно зрозуміле управління для користувача. Крім того, таке рішення дозволяє сфокусуватись на реалізації основної функціональності — стабільного та точного бездротового управління на основі вхідних даних із гіроскопів.

З огляду на доступність готових механічних рішень, для прискорення етапу розробки було вирішено використати готовий конструктор SunFounder Robotic Arm, який широко застосовується в навчальних та дослідницьких цілях. Проте, незважаючи на модульність конструкції, він не передбачає штатного бездротового інтерфейсу, що зумовило необхідність подальших апаратних модифікацій.

4.2 Модифікація конструкції маніпулятора

Оскільки обраний маніпулятор не має вбудованої підтримки бездротового зв’язку, першочерговою задачею стала інтеграція відповідного каналу передачі даних. Для цього було обрано Bluetooth-модуль HC-06, який забезпечує надійний та енергоефективний спосіб бездротової комунікації з мікроконтролером. Цей

модуль є сумісним із Arduino Uno за напругою живлення, проте має іншу логічну напругу — 3.3 В.

З огляду на те, що Arduino Uno працює на логічному рівні 5 В, пряме з’єднання передавальної лінії Arduino з приймальною лінією модуля HC-06 може призвести до пошкодження останнього через перевищення допустимого рівня логічного сигналу. З метою забезпечення електричної безпеки модуля було реалізовано простий перетворювач на основі резистивного подільника напруги. Оптимальне співвідношення резисторів (1 кОм та 2 кОм) дозволяє знизити напругу з 5 В до приблизно 3.3 В, що відповідає рівню логічної одиниці для HC-06.

Усі інші зміни стосувалися виключно програмної реалізації. Оскільки початково маніпулятор був орієнтований на локальне керування за допомогою аналогових сигналів з потенціометрів, його програмну логіку було адаптовано для взаємодії з зовнішнім джерелом команд. Зокрема, функціональність зчитування з аналогових входів була замінена на прийом та обробку структурованих пакетів даних, що надходять через UART-інтерфейс від бездротового передавача. Це забезпечило повну інтеграцію маніпулятора до системи жестового керування, дозволивши йому реагувати на рухи користувача в режимі реального часу.

4.3 Пристрій для жестового управління

Процес створення пульта керування є ключовим етапом у реалізації системи жестового управління, оскільки саме цей пристрій безпосередньо зчитує жести оператора та перетворює їх у команди для виконавчих механізмів. Перед початком розробки було визначено основні функціональні вимоги до пристрою, які включають:

— розпізнавання жестів користувача в реальному часі;

— обробку інформації з сенсорів та генерацію відповідних команд для керування чотирма сервоприводами;

— реалізацію бездротової передачі даних до виконавчого пристрою;

— забезпечення автономності роботи за допомогою вбудованого акумулятора;

— наявність можливості тимчасово вимикати передачу даних без втрати зв’язку чи перезавантаження системи.

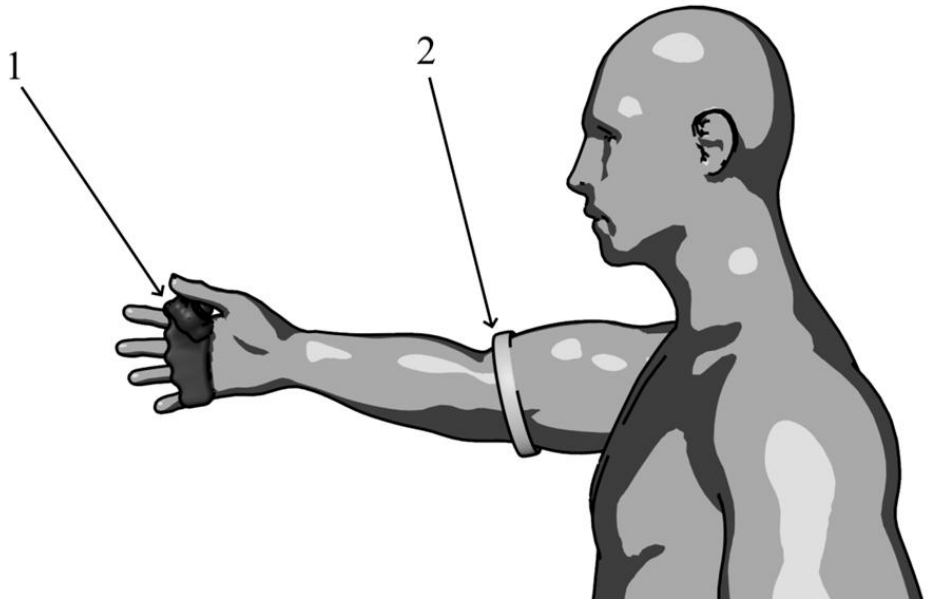

Рисунок 4.1 Розміщення гіроскопів на тілі оператора

Одним із найпростіших та найефективніших способів розпізнавання жестів є використання інерціальних сенсорів, зокрема гіроскопів та акселерометрів. У даній роботі було використано два комбіновані модулі MPU6050, які здатні вимірювати як лінійне прискорення, так і кутову швидкість. Модулі розміщуються способом, показаним на рисунку 4.1:

1) перший модуль знаходиться всередині самого пульта;

2) другий модуль закріплюється вище ліктя оператора, як показано на рисунку.

Такий підхід дозволяє реалізувати базову інтерпретацію жестів за допомогою лише двох сенсорів, значно спрощуючи конструкцію пульта.

Зчитування, обробку та формування керуючих сигналів виконує плата Arduino Nano, яка обрана завдяки своїм компактним розмірам, низькому енергоспоживанню та достатній кількості портів введення/виведення.

Arduino Nano також відповідає за формування та відправку структурованих пакетів даних через Bluetooth-з’єднання. Для реалізації останнього використовується модуль HC-05, який забезпечує стабільний бездротовий канал зв’язку на відстані до 10 метрів у межах прямої видимості.

Для забезпечення автономної роботи в конструкцію пульта вбудовано літій-йонний акумулятор. Це джерело живлення є оптимальним для подібних застосувань завдяки можливості багаторазової зарядки та сумісності з живленням більшості Arduino-плат. Активація живлення здійснюється за допомогою окремої кнопки, розміщеної на корпусі пристрою.

Особливу увагу під час розробки було приділено реалізації режиму тимчасового призупинення передачі даних. У програмному забезпеченні пульта передбачено, що під час встановлення з’єднання з маніпулятором автоматично ініціалізуються початкові значення кутів сервоприводів. Така поведінка, закладена безпосередньо в програмному коді пульта, призводить до повернення маніпулятора у вихідне положення при кожному перепідключенні, що є небажаним у реальних умовах експлуатації. Щоб уникнути повного вимкнення пульта під час коротких перерв у керуванні, було реалізовано можливість тимчасового припинення передачі даних без розриву Bluetooth-з’єднання. Для перемикання між активним та пасивним режимами використовується кнопка, інтегрована безпосередньо в джойстик пульта керування. Це рішення забезпечує зручність користування та дозволяє зберігати поточний стан системи. Для візуального інформування оператора про режим роботи пульта застосовується індикаторний світлодіод: у режимі передачі даних він блимає, тоді як у режимі паузи — світиться постійно.

Для виготовлення корпуса була задіяна технологія 3D друку, завдяки чому корпус вийшов компактний і зручний. Після розміщення всіх вищеописаних елементів всередині нього було отримано повноцінний пульт керування. Його елементи показані на рисунку 4.2:

1) джойстик для керування хапальним механізмом і функцією натискання для перемикання між режимами передачі даних;

2) індикаційний світлодіод, використовується для сповіщення оператора про те, в якому режимі передачі даних знаходиться пульт;

3) перемикач для увімкнення та вимкнення пульта керування;

4) провідники для зв’язку з датчиком, що кріпиться вище ліктя.

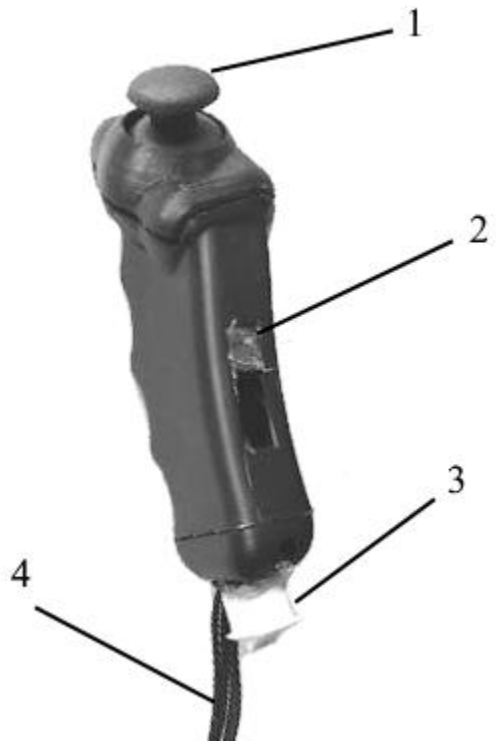

Рисунок 4.2 — Готовий пульт керування

Для повноцінного керування необхідно не лише тримати пульт, а й закріпити на собі другий датчик гіроскоп за допомогою гнучкого пластикового тримача, як показано на рисунку 4.3.

4.4 Тестування та базове управління системою

Перед початком роботи необхідно подати живлення на маніпулятор через один з доступних входів для живлення. Для кращої та стабільнішої подачі живлення в даному випадку використовується телефонний блок живлення і кабель типу USB Type B. Після цього маніпулятор виставить всі сервоприводи в початковий стан. На його Bluetooth модулі в цей момент блиматиме червоний світлодіод, сповіщуючи про спробу встановити з’єднання з пультом.

Рисунок 4.3 — Спосіб кріплення датчика гіроскопа на операторі

Наступним кроком необхідно увімкнути пульт за допомогою перемикача в його нижній частині. На пульті засвітиться зелений світлодіод, сповіщуючи про увімкнення в режимі без передачі даних. В цей момент потрібно почекати, доки червоний світлодіод на Bluetooth модулі маніпулятора не припинить блимати і не почне постійно світитися – це свідчитиме про успішне підключення.

Далі потрібно прийняти нейтральну позу керування (рисунок 4.4) — виставити руку, на якій встановлений пульт, вперед приблизно на кут 45º відносно розслабленого стану, і те ж зробити ліктем. Кисть виставити так, щоб вона була перпендикулярно до землі.

Тепер, після одиничного натискання на кнопку перемикання режимів (світлодіод на пульті повинен почати блимати), розпочинається безпосередньо процес керування.

Керування хапальним механізмом («клешнею») не являється складним — для того, щоб скласти «клешню», потрібно змістити джойстик в напрямку вперед (від себе), для того, щоб відпустити – назад (на себе).

Для керування поворотом маніпулятора вліво-вправо використовується нахил пульта у відповідну сторону відносно тіла оператора. Чим більший нахил,

тим швидше буде відбуватися поворот. При цьому він не буде повертатись, доки пульт знаходиться у нейтральному положенні відносно цієї вісі обертання (рисунок 4.5).

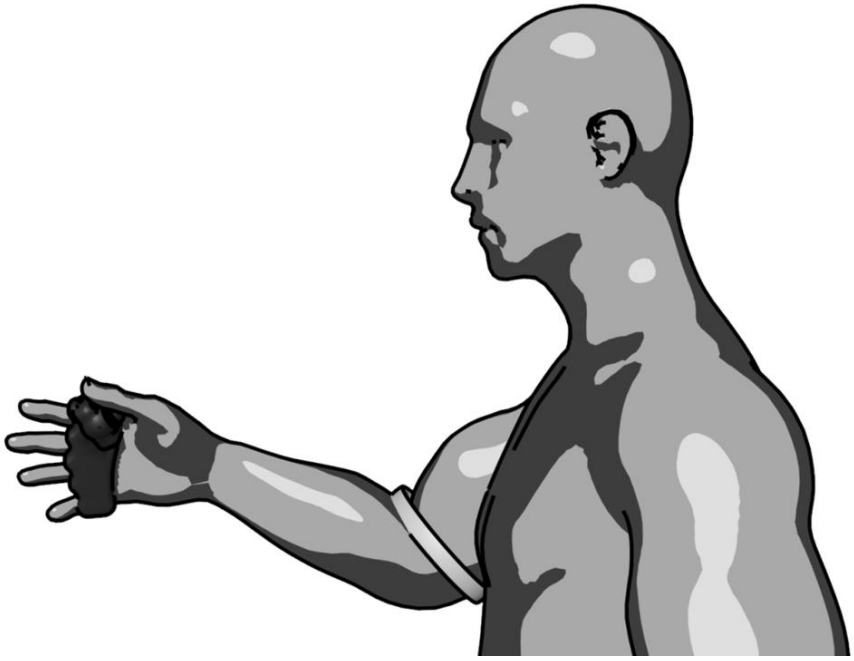

Рисунок 4.4 — Нейтральна поза для керування

Керування нахилом першого ступеня маніпулятора — плеча — відбувається за допомогою зміни кута нахилу гіроскопа на передпліччі по вісі, перпендикулярній землі та йде в сторону відносно точки зору оператора (рисунок 4.6), тобто рухаючи плечем вперед або назад. Маніпулятор буде повторювати заданий кут, попередньо перетворивши його у власний діапазон.

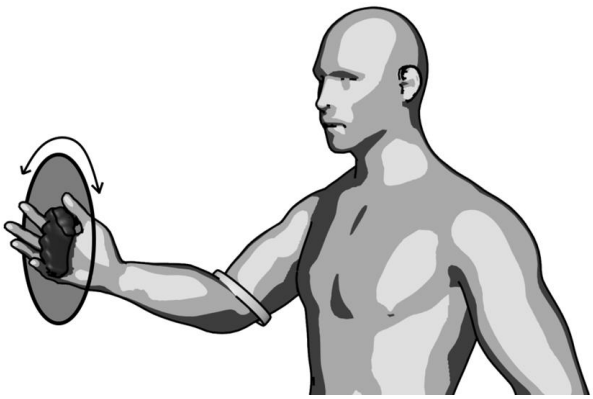

Рисунок 4.5 — Вісь для керування поворотом вліво-вправо маніпулятора

Нахил другого ступеня маніпулятора, який можна назвати його ліктем, керується аналогічно до керування плечем, з відмінністю лише в тому, що там для цього використовується вже нахил пульта по тій самій вісі (рисунок 4.7). І так само аналогічно до керування плечем, лікоть маніпулятора буде повторювати кут нахилу пульта, але в своєму діапазоні.

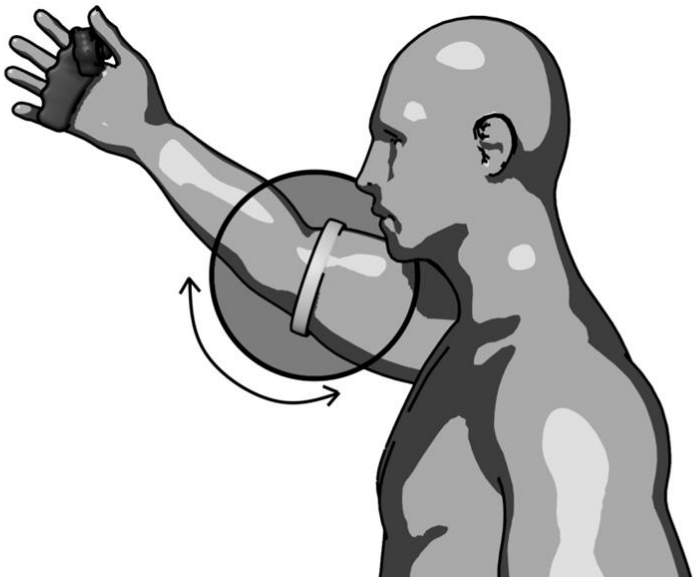

Рисунок 4.6 — Вісь для керування нахилом плеча маніпулятора

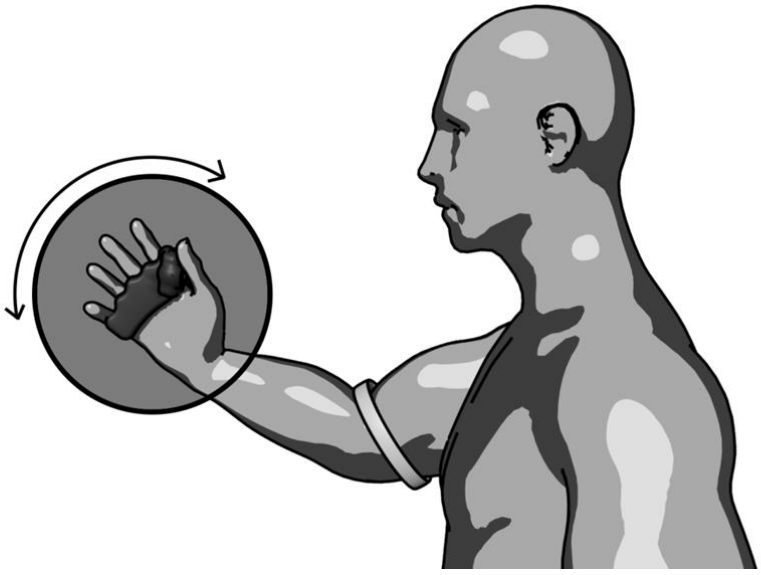

Рисунок 4.7 — Вісь для керування нахилом ліктя маніпулятора

Для завершення роботи з маніпулятором достатньо лише вимкнути його, натиснувши на перемикач знизу. Втративши зв’язок, маніпулятор залишиться в останньому набутому стані доти, доки з’єднання з пультом не буде відновлено або доки не перезапустити його, наприклад, вимкнути і увімкнути подачу живлення.

Таким чином, виконуючи нахили руки по різних вісях, виконується жестове керування роботизованим маніпулятором.

ВИСНОВОК

У процесі виконання бакалаврської кваліфікаційної роботи було розроблено повнофункціональну систему бездротового керування роботизованим маніпулятором на основі інтерпретації жестів оператора. Робота охопила всі етапи створення подібної системи — від підбору апаратного забезпечення та макетування пристроїв до програмної реалізації алгоритмів обробки даних, організації зв’язку та передачі команд між пристроями.

Для реалізації функції розпізнавання рухів було застосовано два цифрових сенсори MPU6050, які поєднують у собі гіроскоп та акселерометр. Вони дозволяють зчитувати положення руки оператора в просторі з високою точністю. Отримані дані передаються на мікроконтролер Arduino Nano, який обробляє інформацію та формує пакети для передавання. Передача даних між пультом керування та маніпулятором здійснюється через бездротовий Bluetooth-зв’язок з використанням модулів HC-05 та HC-06. У кожному пакеті реалізована проста перевірка на помилки завдяки додаванню контрольної суми, що суттєво підвищує надійність передачі даних навіть в умовах перешкод.

На приймальній стороні, Arduino Uno приймає дані, виконує верифікацію пакетів і, у разі коректності, перетворює їх на конкретні значення кутів для сервоприводів маніпулятора. За допомогою бібліотеки Servo.h реалізовано плавне й стабільне керування виконавчими механізмами. Було враховано також захист від занадто різких або некоректних змін положення, що дозволило уникнути потенційних механічних пошкоджень.

У результаті розробки:

— створено канал стабільного двостороннього обміну даними між мікроконтролерами;

— забезпечено точне та надійне керування чотирма незалежними сервоприводами;

— реалізовано можливість тимчасового зупинення передачі даних без втрати з’єднання;

— досягнуто інтуїтивного керування маніпулятором через зміну положення руки в просторі;

— впроваджено оптимальну структуру пакету даних для передачі координат та перевірки достовірності.

Система показала хорошу чутливість, стабільність роботи в реальному часі та придатність для використання в умовах лабораторних досліджень або демонстраційних стендів. Отриманий результат може стати базою для подальшого розширення функціоналу, наприклад, через додавання додаткових ступенів свободи, інтеграцію з машинним зором або використання мережевих протоколів зв’язку. Також можливе застосування цієї розробки в навчальних проєктах, а також у галузі реабілітаційних технологій — для створення допоміжних засобів керування пристроями людьми з порушеннями моторики.

Таким чином, поставлені задачі було виконано в повному обсязі, а результати роботи підтвердили ефективність і доцільність обраного підходу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Акселерометри MPU6050: технічна документація [Електронний ресурс] / InvenSense. – Режим доступу: https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/

2. Модулі HC-05/HC-06 для Bluetooth-зв’язку: огляд і підключення [Електронний ресурс]. – Режим доступу: https://lastminuteengineers.com/hc05-bluetooth-module-arduino-tutorial/

3. Arduino: офіційна документація [Електронний ресурс]. – Режим доступу: https://www.arduino.cc/en/Guide/Introduction

4. Gesture Based Robot Control Using Hand Movements / Patel K., Desai J. // *International Journal of Engineering Research & Technology*, 2021. – Vol. 10, Issue 5. – Режим доступу: https://www.ijert.org/gesture-based-robot-control-using-hand-movements

5. Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп’ютерної інженерії: матеріали конференції [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24330

6. Система хірургічної робототехніки da Vinci [Електронний ресурс]. – Режим доступу: https://www.intuitive.com/en-us/products-and-services/da-vinci

7. Canadarm: космічний маніпулятор [Електронний ресурс]. – Режим доступу: https://www.asc-csa.gc.ca/eng/canadarm/default.asp

8. Gesture Recognition Technology for Robotic Arm Control [Електронний ресурс] / IEEE Xplore. – Режим доступу: https://ieeexplore.ieee.org/document/8792637

9. Leap Motion Controller 2: офіційний сайт [Електронний ресурс]. – Режим доступу: https://www.ultraleap.com/product/leap-motion-controller-2

10. Інтерфейс I2C [Електронний ресурс]. – Режим доступу: https://docs.arduino.cc/learn/communication/wire/

11. UART [Електронний ресурс]. – Режим доступу: https://docs.arduino.cc/learn/communication/uart/

12. Протокол Bluetooth [Електронний ресурс]. – Режим доступу: https://www.bluetooth.com/

13. Модуль MPU6050 [Електронний ресурс]. – Режим доступу: https://www.alldatasheet.com/datasheet-pdf/view/1132807/TDK/MPU-6050.html

14. Цифровий гіроскоп [Електронний ресурс]. – Режим доступу: https://modelistam.com.ua/ua/giroskopy-radioupravlyaemyh-modelyah-a-43/

15. SunFounder Robot Rollarm [Електронний ресурс]. – Режим доступу: https://docs.sunfounder.com/projects/rollarm/en/latest/

16. Модуль джойстика KY-023 [Електронний ресурс]. – Режим доступу: https://naylampmechatronics.com/img/cms/Datasheets/000036%20-%20datasheet%20KY-023-Joy-IT.pdf

17. Arduino UNO [Електронний ресурс]. – Режим доступу: https://docs.arduino.cc/hardware/uno-rev3/

18. Arduino Nano [Електронний ресурс]. – Режим доступу: https://docs.arduino.cc/hardware/nano/

19. Arduino IDE [Електронний ресурс]. – Режим доступу: https://www.arduino.cc/en/software/

20. ATmega328P [Електронний ресурс]. – Режим доступу: https://www.alldatasheet.com/datasheet-pdf/view/241077/ATMEL/ATMEGA328P.html

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н.. Азаров О.Д.
« 21 » березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи
«Мікропроцесорна система з реалізацією жестового управління»
08-54.БКР.012.00.000 ТЗ

Науковий керівник: проф., д.т.н.
_____ Мартинюк Т.Б.
Студент групи 1КІ-23мс
_____ Лубко В.Р.

ВНТУ 2025

1 Підставою для виконання бакалаврської кваліфікаційної роботи (БКР)

Наказ про затвердження тема БКР.

2 Мета БКР і призначення розробки

2.1 Мета роботи — розробка прототипу системи, що використовує керування роботизованим маніпулятором за допомогою жестового управління.

2.2 Призначення розробки — система жестового управління призначена надати більш інтуїтивний спосіб дистанційного керування роботизованим маніпулятором.

3 Вихідні дані для виконання БКР

3.1 Проведення аналізу існуючих рішень у сферах керування роботизованими маніпуляторами та зчитування даних про жести людини;

3.2 Підбір компонентів для розробки системи.

3.3 Модифікація роботизованого маніпулятора.

3.4 Розробка апаратної та програмної складових для системи керування.

3.5 Налаштування бездротового зв’язку між системою керування та роботизованим маніпулятором.

3.6 Проведення тестування та валідації системи в реальних умовах.

4 Вимоги до виконання БКР

4.1 Використання модулів MPU6050, HC-05 і HC-06 для збору даних про жести оператора та передачу їх на маніпулятор.

4.2 Розробка програмного забезпечення на мові C++ для зчитування та обробки даних про жести оператора.

4.3 Забезпечення дистанційного бездротового керування роботизованим маніпулятором.

4.4 Проведення тестування системи для забезпечення її надійності та стабільності в умовах експлуатації.

Розробка системи бездротового керування роботизованим маніпулятором на основі жестів оператора, що включає використання двох датчиків MPU6050, мікроконтролера Arduino Nano для збору та обробки даних, модуля Bluetooth HC-05 для передавання інформації, а також приймального блоку на базі Arduino Uno з сервоприводами та модулем HC-06 для виконання команд у реальному часі.

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати наведено в Таблиці А.1.

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, рецензію, анотації до БКР українською та іноземною мовами, довідка про відповідність оформлення БКР діючим вимогам.

7 Порядок контролю виконання та захисту БКР:

Виконання етапів графічної документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 «Комп’ютерна інженерія» (освітня програма «Комп’ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіт СУЯ ВНТУ-03.02.02-П.001.01:21».

Таблиця А.1 – Етапи БКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз літературних джерел. Попередня розробка основних розділів	21.03.2025	26.03.2025	Вступ
2	Огляд та аналіз існуючих рішень в жестовому керуванні	27.03.2025	30.03.2025	Розділ 1
3	Розробка апаратної складової системи, модифікація роботизованого маніпулятора	31.03.2025	6.04.2025	Розділ 2
4	Розробка програмної складової системи	7.04.2025	19.05.2025	Розділ 3
5	Тестування робочого прототипу системи	20.04.2024	27.04.2024	Розділ 4
6	Пояснювальна записка	28.04.2024	13.05.2024	Презентація

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Мікропроцесорна система з реалізацією жестового управління»

Тип роботи: _____ бакалаврська кваліфікаційна робота _____
Підрозділ _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism ___2___%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)
____Крупельницький Л.В., доц. каф ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ___Захарченко С.М.____
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ _____Мартинюк Т. Б._____
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____Лубко В. Р._____
(підпис) (прізвище, ініціали)

ДОДАТОК В
Графічна частина

МІКРОПРОЦЕСОРНА СИСТЕМА З РЕАЛІЗАЦІЄЮ ЖЕСТОВОГО УПРАВЛІННЯ

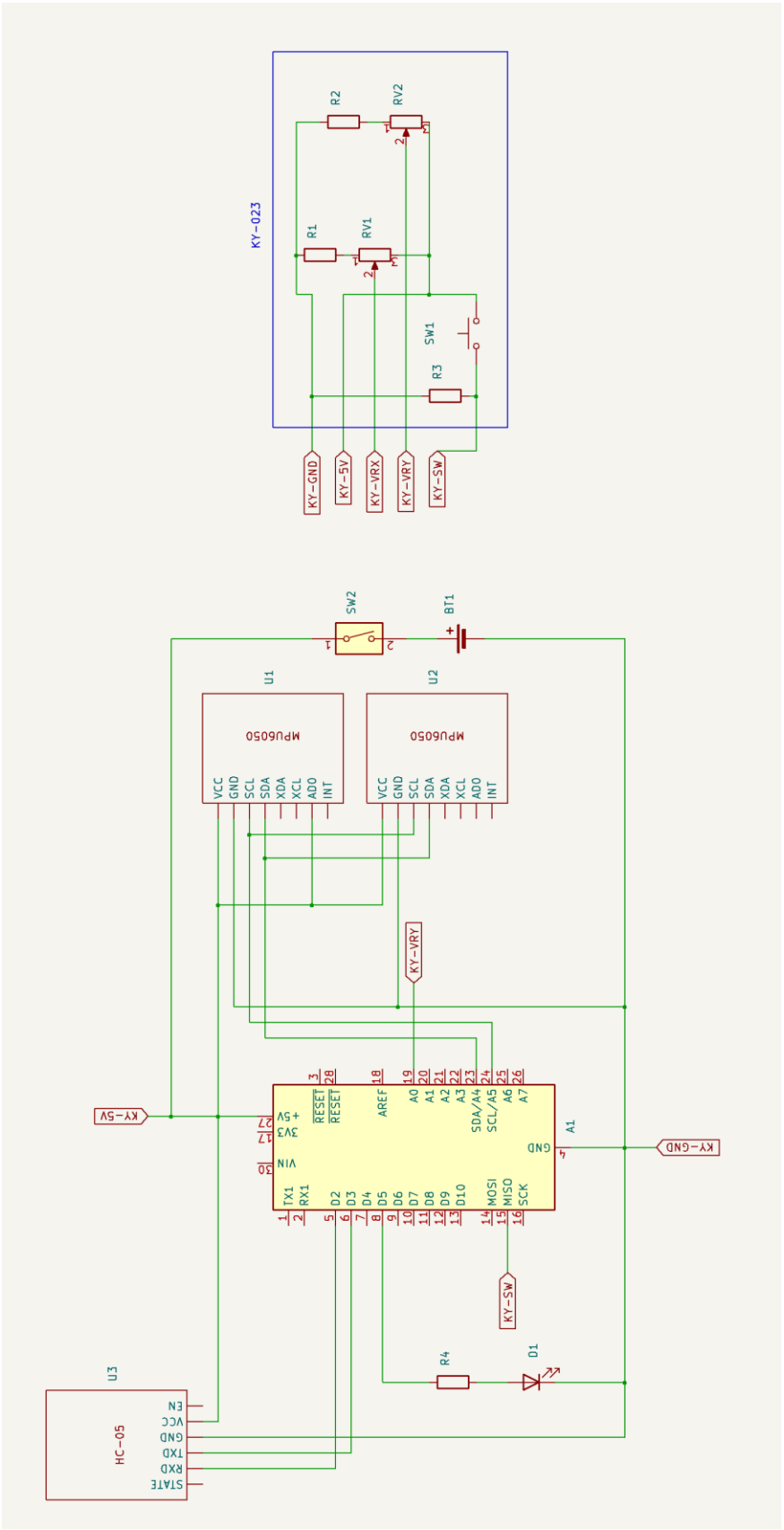

Рисунок В.1 — Електрична принципова схема пульта керування

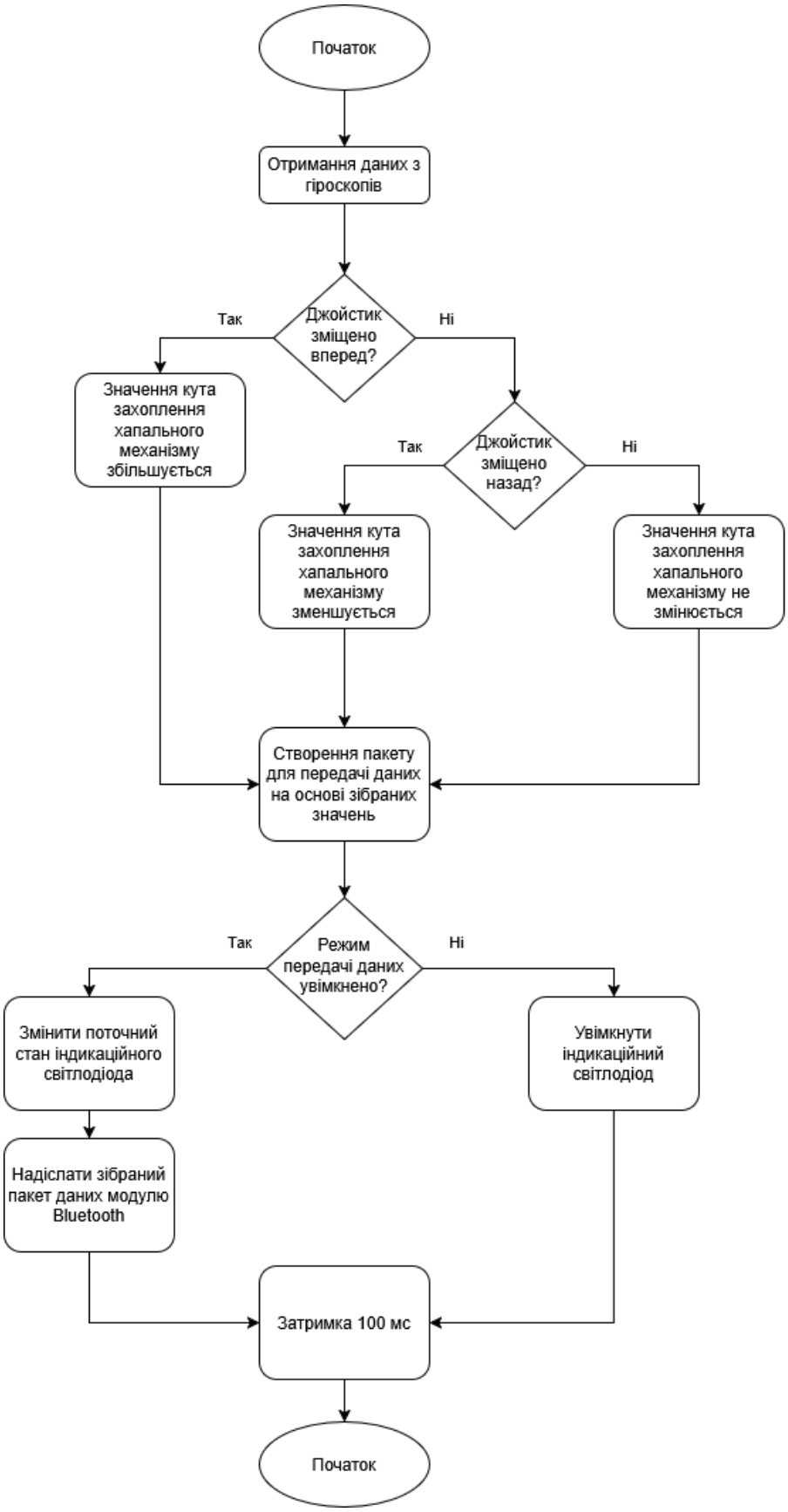

Рисунок В.2 — Блок-схема роботи основного циклу роботи програмного коду пульта керування

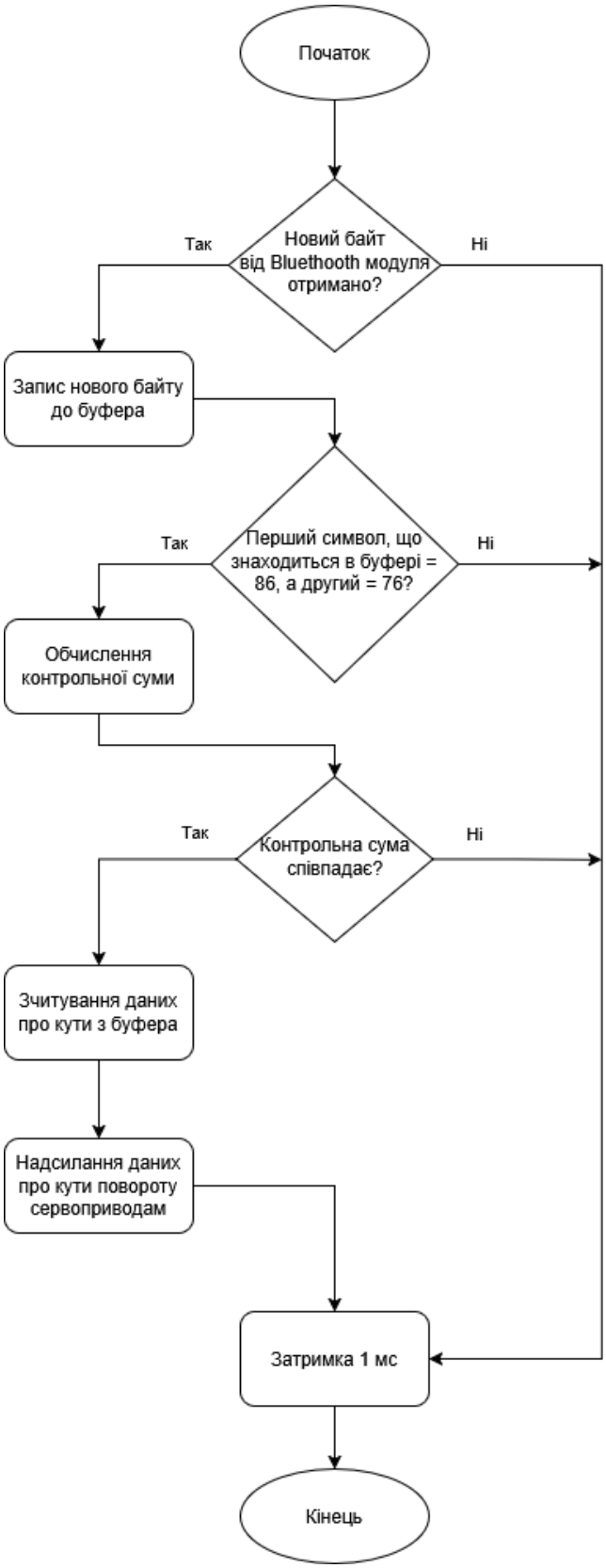

Рисунок В.3 — Блок-схема роботи основного циклу роботи програмного коду роботизованого маніпулятора

ДОДАТОК Г

Лістинг коду пульта керування

```
#include <MPU6050.h>
#include <SoftwareSerial.h>
#define POT_PIN A0
#define CTRL_BTN 12
#define LED_PIN 5
#define DEBOUNCE_DELAY 100

MPU6050 mpuHand(0x69);
MPU6050 mpuShoulder(0x68);
SoftwareSerial BTSerial(2, 3);

uint8_t dataToSend[11] = { 86, 76, 0,0, 0,0, 0, 0,0, 0,0 };
uint8_t clawAngle;
bool sending = false;
bool prevRawState = HIGH;
bool prevDebouncedState = HIGH;
uint32_t lastDebounceTime = 0;
bool pressRegistered = false;

uint8_t startButton() {
  bool currentRawState = digitalRead(CTRL_BTN);
  if (currentRawState != prevRawState) {
    lastDebounceTime = millis();
    prevRawState = currentRawState;
  }
  if ((millis() - lastDebounceTime) > DEBOUNCE_DELAY) {
    if (prevDebouncedState == HIGH && currentRawState == LOW) {
      pressRegistered = true;
    } else {
      pressRegistered = false;
    }
    prevDebouncedState = currentRawState;
  } else {
    pressRegistered = false; // всередині дребезгу
  }
  return pressRegistered ? 1 : 0;
}


void SetClawAngle(){
```

```
  int potValue = analogRead(POT_PIN);
    if (potValue > 520) {
      clawAngle+=5;
    } else if (potValue < 500) {
      clawAngle-=5;
    }
    clawAngle = constrain(clawAngle, 10, 170);
}

void setup() {
  Wire.begin();
  Serial.begin(9600);
  BTSerial.begin(9600);
  mpuHand.initialize();
  mpuShoulder.initialize();
  pinMode(POT_PIN, INPUT);
  pinMode(CTRL_BTN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, HIGH);
}

uint16_t CRC_check(uint8_t *mesagge, uint16_t length) {
  uint16_t crc_out = 0;
  for (int var = 0; var < (length - 2); ++var) {
    crc_out += mesagge[var];
  }
  return crc_out;
}

void SetDataToSend(uint16_t g1p, uint16_t g1r,uint16_t g2r) {
  dataToSend[2] = (g1p >> 8) & 0xFF;
  dataToSend[3] = g1p & 0xFF;
  dataToSend[4] = (g1r >> 8) & 0xFF;
  dataToSend[5] = g1r & 0xFF;
  dataToSend[6] = clawAngle;
  dataToSend[7] = (g2r >> 8) & 0xFF;
  dataToSend[8] = g2r & 0xFF;

  uint16_t crc = CRC_check(dataToSend, 11);
  dataToSend[9] = (crc >> 8) & 0xFF;
  dataToSend[10] = crc & 0xFF;
}
```

```
class Angles {
  public:
    int16_t x, y, z;

    void GetAcceleration(int16_t gx, int16_t gy, int16_t gz){
      x = gx;
      y = gy;
      z = gz;
    }

    uint16_t GetX() {
      int16_t ax = x;
      ax = constrain(ax, -16384, 16384);
      float angle = ax / 16384.0;

      if (angle < 0)
        angle = 90 - degrees(acos(angle));
      else
        angle = degrees(acos(-angle)) - 90;

      return map(angle * 100, -9000, 9000, 0, 65535);
    }

    uint16_t GetY() {
      int16_t ay = y;
      ay = constrain(ay, -16384, 16384);
      float angle = ay / 16384.0;

      if (angle < 0)
        angle = 90 - degrees(acos(angle));
      else
        angle = degrees(acos(-angle)) - 90;
      return map(angle * 100, -9000, 9000, 0, 65535);
    }

    uint16_t GetZ() {
      int16_t az = z;
      az = constrain(az, -16384, 16384);
      float angle = az / 16384.0;

      if (angle < 0)
        angle = 90 - degrees(acos(angle));
      else
        angle = degrees(acos(-angle)) - 90;
```

```
      return map(angle * 100, -9000, 9000, 0, 65535);
    }
};

void PrintData(){
  for(int i = 0;i < 11;i++){
     Serial.print(dataToSend[i]);
     if(dataToSend[i] >= 100){
       Serial.print(" ");
     }else if(dataToSend[i] >= 10){
       Serial.print("  ");
     }else{
       Serial.print("   ");
     }
    }
     Serial.println();
}

void loop() {
  Angles hand, shoulder;

hand.GetAcceleration(mpuHand.getAccelerationX(),mpuHand.getAccelerationY(),mpu
Hand.getAccelerationZ());
     shoulder.GetAcceleration(mpuShoulder.getAccelerationX(),mpuShoulder.getAcc
elerationY(),mpuShoulder.getAccelerationZ());
     SetClawAngle();
     SetDataToSend(hand.GetX(),hand.GetZ(),shoulder.GetY());
     if(startButton()){
       sending = !sending;
       if(sending){
       BTSerial.write(dataToSend,11);
       delay(1000);
       }
     }
     if(sending){
        digitalWrite(LED_PIN, !digitalRead(LED_PIN));
       BTSerial.write(dataToSend,11);
     }else{
        digitalWrite(LED_PIN, HIGH);
     }
     delay(100);
}
```

ДОДАТОК Д

Лістинг коду роботизованого маніпулятора

```
#include <Servo.h>
Servo Servo_0, Servo_1, Servo_2, Servo_3;
uint8_t buffer[11];

class Angles {
 private:
  uint16_t g1pitch, g1roll, g2roll;
  uint8_t claw, rotate = 90;
 public:
  void GetAngles(uint8_t *message) {
    g1pitch = Rotate(((uint16_t)message[2] << 8) | message[3]);
    g1roll = ((uint16_t)message[4] << 8) | message[5];
    claw = message[6];
    g2roll = ((uint16_t)message[7] << 8) | message[8];
    g1roll = map(g1roll, 0, 65535, 10, 170);
    g2roll = map(g2roll, 0, 32000, 10, 170);
  }
 uint8_t Rotate(uint16_t angleGet){
  int16_t angle = map(angleGet, 0, 65535, -100, 100);
  static int16_t prevAngle = angle;
  if (abs((int)angle - (int)prevAngle) > 20){
     rotate += angle/10;
     rotate = constrain(rotate, 10, 170);
    return angle;
  }else{
    angle = prevAngle;
    return prevAngle;
  }
 }
  void PrintAngles() {
    Serial.print("Angles: ");
    Serial.print(g1pitch); Serial.print(" ");
    Serial.print(g1roll); Serial.print(" ");
    Serial.print(claw); Serial.print(" ");
    Serial.print(g2roll); Serial.println();
  }
  void MoveServo() {
    Servo_0.write(rotate);
    Servo_1.write(g1roll);
    Servo_2.write(g2roll);
```

```
      Servo_3.write(claw);
    }
};
Angles angles;
void setup() {
  Serial.begin(9600);
  Servo_0.attach(4);
  Servo_1.attach(6);
  Servo_2.attach(5);
  Servo_3.attach(7);
  Serial.println("Bluetooth ready!");
  for (int i = 0; i < 11; i++) {
    buffer[i] = 0;
  }
}
uint16_t CRC_check(uint8_t *mesagge, uint16_t length) {
  uint16_t crc_out = 0;
  for (int var = 0; var < (length - 2); ++var) {
    crc_out += mesagge[var];
  }
  return crc_out;
}
bool checking_message(uint8_t *message) {
    if ((message[0] == 86) && (message[1] == 76)) {
        uint16_t crc_received = ((uint16_t)message[9] << 8) | message[10];
        uint16_t crc_calculated = CRC_check(message, 11);
        return (crc_received == crc_calculated);
    }
    return false;
}
void loop() {
  if (Serial.available()) {
    uint8_t c = Serial.read();

    for (int i = 0; i < 10; i++) {
        buffer[i] = buffer[i + 1];
    }
    buffer[10] = c;
    if (checking_message(buffer)) {
        angles.GetAngles(buffer);
        angles.MoveServo();
    }
}
  delay(1);
```