Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель системи трекінгу кінцівок людини за допомогою сенсорів

(тема)

Виконав:

студент II курсу, групи СПм-21-2

Дашков Д.Є.
(прізвище, ініціали)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма Системне програмування
(повна назва освітньої програми)

Керівник: доц. Ляшенко О.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

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

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Дашкову Дмитру Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Модель системі трекінгу кінцівок людини за допомогою сенсорів

затверджена наказом по університету від " 03 " квітня 2023 р. № 318 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 17 травня 2023 р.

3. Вхідні дані до роботи Створити модель для відстеження руху кінцівок людини.

4. Перелік питань, що потрібно опрацювати у роботі

1. Розробка моделі;
2. реалізація програмного забезпечення;
3. створення прототипу пристрою;

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Загальна схема моделі, Сингнал потенциомітра при руху пальця, Загальна схема, Схема підключення мікроконтролера, 3Д модель пристрою, Фото прототипу, Графік роботи фільтра Калмана, Визначення arctan, Проекція на осі, Приклад роботи ADC, Загальна структура дескрипторів, Скріншот інтерфейсу підключення, Інтерфейс додатку, Візуалізація додатку, Осцилограма сигналу, Графіх даних з компьютеру; Слайд-презентація – 11 слайдів; Схема плати пристрою

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	03.04.23-11.04.23	
2	Формування вимог системи	12.04.23-20.04.23	
3	Розробка та налагодження системи	21.04.23-26.04.23	
4	Тестування	27.04.23-30.04.23	
5	Оформлення матеріалів кваліфікаційної роботи	1.05.23-05.05.23	
6	Подання кваліфікаційної роботи керівникові та	06.05.23-10.05.23	
	її попередній захист		
7	Подання роботи на рецензування	10.05.23-16.05.23	

Дата видачі завдання 03 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доцент Ляшенко О.С

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 70 с., 18 рис., 3 дод., 22 джерел.

MEMS, ЕЛЕКТРОМЕХАНІЧНИЙ ДАТЧИК, ТРЕКІНГ ОБ’ЄКТІВ, USB, OpenGL, GLUT.

Метою кваліфікаційної роботи є розробка моделі актуальної проблеми захвату рухів кінцівок людини за допомогою механічних та електромеханічних датчиків.

У ході виконання кваліфікаційної роботи було проведено аналіз існуючих методів трекінгу об’єктів. Крім того, розроблена електрична схема та програмне забезпечення. На основі яких був створений прототип пристрою для стеженням рухів кінцівок людини.

ABSTRACT

Master’s thesis: 70 pages, 18 figures, 3 appendices, 22 sources.

MEMS, ELECTROMECHANICAL SENSOR, OBJECT TRACKING, USB, OpenGL, GLUT.

The purpose of the qualification work is to develop a model of the actual problem of capturing the movements of human limbs using mechanical and electromechanical sensors.

In the course of the qualification work, an analysis of existing object tracking methods was carried out. In addition, the electrical circuit and software were developed. On the basis of which, a prototype of a device for tracking the movements of human limbs was created.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ADC – пристрій, що перетворює вхідний аналоговий сигнал в дискретний код (англ., Analog-to-digital converter)

ARM – 32-бітна RISC архітектура процесорів, яку розробила компанія ARM Limited (англ., Advanced RISC Machine)

CV – теорія та технологія створення машин, які можуть проводити виявляння, відстежування та визначення об'єктів (англ., Computer vision)

EMG – метод дослідження біоелектричних потенціалів (англ., Electromyography)

FDM – моделювання методом наплавлення (англ., Fused Deposition Modeling)

GCC – набір компіляторів для різних мов програмування, розроблений в рамках проекту GNU (англ., GNU Compiler Collection)

GLUT – бібліотека, яка надає просту міжплатформенну структуру для створення вікон і керування ними(англ., OpenGL Utility Toolkit)

GPIO – інтерфейс для зв'язку між компонентами комп'ютерної системи (англ., General-Purpose Input/Output)

GPL – універсальна загальнодоступна ліцензія GNU (англ., GNU General Public License)

HAL - апаратний рівень абстракції (англ., Hardware Abstraction Layer)

HID – стандарт підключення через універсальну послідовну шину USB (англ., human interface device)

IMU - інерційний вимірювальний пристрій (англ., Inertial measurement unit)

LDO – найменування класу лінійних стабілізаторів (англ., Low Drop-Out)

MEMS – мікро-електромеханічні системи

PID – пропорційно-інтегрально-диференціювальний (ПІД) регулятор, пристрій у керуючому контурі зі зворотним зв'язком (англ., Proportional–integral–derivative controller)

STM – європейська мікроелектронна компанія (англ., STMicroelectronics)

USB – послідовний інтерфейс для підключення периферійних пристроїв до обчислювальної техніки (англ., Universal Serial Bus)

ВСТУП

Пристрої відстеження людських кінцівок останнім часом стають все більш популярними, особливо в сферах спорту, фізіотерапії та віртуальної реальності. У цих пристроях використовується комбінація датчиків і програмного забезпечення для відстеження рухів кінцівок і забезпечення даних для аналізу продуктивності в реальному часі.

Розробка пристроїв відстеження людських кінцівок була зумовлена прогресом у сенсорних технологіях і алгоритмах машинного навчання. У цих пристроях зазвичай використовується комбінація інерційних датчиків, таких як акселерометри та гіроскопи, і оптичних датчиків, таких як камери та датчики глибини, щоб фіксувати рух кінцівок. Потім дані з цих датчиків обробляються складними алгоритмами для точного відстеження рухів кінцівок у тривимірному просторі.

Однією з ключових переваг пристроїв відстеження людських кінцівок є їхня здатність надавати зворотний зв’язок щодо продуктивності в реальному часі. Спортсмени та пацієнти, які проходять фізіотерапію, можуть використовувати дані з цих пристроїв, щоб відстежувати свій прогрес і відповідно коригувати тренування [1]. У середовищах віртуальної реальності пристрої відстеження людських кінцівок можуть забезпечити більший ефект занурення, дозволяючи користувачам взаємодіяти з віртуальним світом за допомогою власних рухів тіла.

Загалом пристрої відстеження людських кінцівок можуть революціонізувати спосіб моніторингу та покращити роботу людини. Оскільки сенсорні технології та алгоритми машинного навчання продовжують розвиватися, ці пристрої ставатимуть ще точнішими та кориснішими, відкриваючи нові можливості для спорту, розваг та медицини.

У даній роботі представлено пристрій для відстеження рухів кінцівок людини, до складу якого входять плати з мікропроцесором та інерційними мікромеханічними датчиками. Для фіксації рухів пристрій використовує

результат акселерометра і гіроскопа, за їх значеннями розраховується траєкторія руху: кут повороту і прискорення. Дані зчитуються мікроконтролером, після чого фільтруються та обробляються мікроконтролером, а потім передаються на комп’ютер для подальшого аналізу та відображення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальності обраної теми

Актуальність трекінгу руху кінцівок людини полягає в здатності точно відстежувати та аналізувати рухи різних частин тіла під час різних дій. Це важливо в багатьох сферах, включаючи спортивну науку [2, 3], реабілітацію [4] та віртуальну реальність.

У спортивній науці захоплення руху можна використовувати для відстеження рухів спортсменів під час тренувань і змагань. Це може надати тренерам і інструкторам цінну інформацію про техніку, продуктивність і запобігання травмам [5]. Дані захоплення руху також можна використовувати для аналізу моделей рухів спортсменів і визначення факторів, які сприяють досягненню успіхів.

У реабілітації захоплення руху можна використовувати для відстеження прогресу пацієнтів, коли вони одужують після травм або операцій [6]. Це може допомогти фізіотерапевтам розробити ефективні програми вправ і контролювати ефективність методів лікування. Захоплення руху також може використовуватися для розробки протезів і допоміжних пристроїв, адаптованих до конкретних потреб пацієнта [7].

У віртуальній реальності захоплення руху можна використовувати для створення реалістичних аватарів [8] і покращення ефекту занурення для користувачів [9]. Дані захоплення руху можна використовувати для анімації віртуальних персонажів і створення природних рухів. Це можна використовувати в широкому діапазоні додатків, включаючи ігри, навчання та стимуляційні вправи.

Загалом актуальність фіксації руху людських кінцівок полягає в здатності точно відстежувати й аналізувати дані про рухи, що може надати цінну інформацію про продуктивність і поведінку людини. Покращуючи наше розуміння того, як рухається людське тіло, захоплення руху може призвести до

прогресу в спортивній науці, медицині та віртуальній реальності, а також покращити способи взаємодії люди з різними електронними пристроями.

1.2 Існуючі аналоги

Існують різні підходи до фіксації руху людських кінцівок, зокрема оптичне фіксування руху за допомогою комп’ютерного зору (CV), інерційні вимірювальні датчиків (IMU) та електроміографія (EMG). Кожен із цих методів має переваги та недоліки, і вибір техніки залежить від конкретного застосування та необхідної точності та прецизійності.

Оптичні системи захоплення руху використовують камери для відстеження положення кінцівок людини за допомогою алгоритмів комп’ютерного зору. Додатково ця технологія може використовувати маркери, прикріплених до кінцівок об’єкта, для більшої стабільності [10]. Ці системи є високоточними та можуть фіксувати детальні дані про рух, але зазвичай вони дорогі та потребують контрольованого середовища з мінімальними перешкодами.

IMU – це невеликі датчики, які прикріплюються до кінцівок і вимірюють прискорення, кутову швидкість і напруженість магнітного поля. Ці датчики можна використовувати в широкому діапазоні середовищ і відносно недорогі, але вони можуть страждати від дрейфу та шуму, що може вплинути на точність вимірювань.

EMG підхід вимірює електричну активність у м’язах кінцівок і може використовуватися для визначення руху кінцівки [11]. Цей метод є неінвазійним і може використовуватися в різних ситуаціях, але він може мати перешкоди та може не надати такої точної чи детальної інформації про рух кінцівок, як оптичні підходи чи підходи на основі IMU.

Виходячи з цього на даний момент основним конкуруючими методами захоплення руху є гіроскопічні датчики та відеокамера з алгоритмами комп’ютерного зору. Порівняння обох технологій покаже явні переваги в

можливих сферах їх використання.

Основними перевагами CV можна виділити наступне:

Техніка комп’ютерного зору є більш гнучкою, ніж гіроскопи, щодо типів рухів, які вони можуть виявляти [12]. Наприклад, вони можуть виявляти рухи в будь-якому напрямку, повороти та зміни масштабу чи перспективи. Однак методи комп’ютерного зору можуть бути дорогими з точки зору обчислень і вимагати значної обчислювальної потужності.

На них також можуть впливати зміни в умовах освітлення або оклюзії в сцені. Методи комп’ютерного зору також можуть бути більш сприйнятливими до шуму та помилок, ніж датчики гіроскопа, особливо якщо камера не стабілізована або якщо на сцені є рухомі об’єкти.

До переваг IMU датчиків можна віднести:

Гіроскопічні датчики безпосередньо вимірюють кутову швидкість і можуть використовуватися для виявлення змін в орієнтації та обертанні об’єкта.

Датчики гіроскопа, як правило, є більш точними, ніж методи комп’ютерного зору для виявлення обертання, і менш чутливі до шуму чи помилок. Однак датчики гіроскопа є менш гнучкими, ніж методи комп’ютерного зору, оскільки вони розроблені спеціально для виявлення обертання та не можуть виявляти інші типи руху.

На датчики гіроскопа також може впливати дрейф з часом, що може призвести до помилок у вимірюванні обертання. Це можна виправити, поєднавши дані гіроскопа з інформацією від інших датчиків, таких як акселерометри або магнітометри [13].

Таким чином, методи комп’ютерного зору можуть бути більш гнучкими та можуть виявляти ширший діапазон руху, але також вони більш дорогі як у обчислювальному так і у фінансовому плані. Датчики гіроскопа більш точні для виявлення обертання, але менш гнучкі та схильні до дрейфу з часом. Вибір методу залежить від конкретних вимог програми та наявних ресурсів.

1.3 Доцільності вдосконалення існуючих рішень

Незалежно від використовуваного методу, проблема фіксації руху кінцівок людини часто пов’язана з необхідністю обробки великих обсягів даних, роботи з шумом і невизначеністю та точного моделювання динаміки руху кінцівок. Ці виклики вимагають складних алгоритмів і методів обробки сигналів, таких як фільтри Калмана, додаткові фільтри та нейронні мережі, щоб точно оцінити рухи кінцівок і надати корисну інформацію про рухи людини. Вдосконалення цих методів дозволить підвищити точності та надійності зібраних даних. Чим точніші та надійніші дані, тим краще рішення може бути прийняте на основі цих даних.

Ще одна причина — зробити їх доступнішими та дешевшими. Зараз багато систем захоплення руху є дорогими та потребують спеціальної підготовки для використання. Зробивши захоплення руху доступнішим і дешевшим, більше людей отримають користь від цієї технології.

Інша причина вдосконалення систем захоплення руху полягає в тому, щоб зробити їх більш універсальними та придатними для різних застосувань. Розробляючи більш універсальні системи захоплення руху, ми можемо використовувати ту саму технологію в широкому діапазоні застосувань, від спортивних тренувань до моделювання віртуальної реальності.

Нарешті, вдосконалення систем захоплення руху може допомогти нам краще зрозуміти, як працює та рухається людське тіло. Збираючи точні та достовірні дані про рух людини, ми можемо отримати уявлення про основні механізми людського тіла. Це може призвести до прогресу в таких галузях, як біомеханіка та робототехніка, і, зрештою, покращити наше розуміння фізіології та поведінки людини.

1.4 Постанова задачі

Проблема фіксації руху кінцівок людини включає в себе завдання точного відстеження та аналізу рухів різних частин тіла під час різних дій. Задача даної роботи складається з розробки моделі та прототитпу пристрою для точного захвату певної кінцівки люди, в даному випадку кисті та пальців руки, за допомогою метода мікроелектромеханічних датчиків (MEMS), які відносяться IMU сенсорів та більш простих механічних датиків.

2 АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ ВИРІШЕННЯ

2.1 Апаратна складова

Для вирішення поставленої задачі потрібен центральний обчислювальний модуль, до якого ставляться наступні вимоги [14]:

Наявність інтерфейсу для підключення до сенсору;

Наявність інтерфейсу, за допомогою якого можливо передати дані до комп’ютеру;

Ядро з можливість обчислення математичних формул з плаваючу точкою;

Невелике енергоспоживання;

Для вирішення поставлених вимог обраний мікроконтролер STM32F401 фірми STM. Мікроконтролери STM32 – це сімейство 32-розрядних мікроконтролерів на основі ARM Cortex-M, розроблених STMicroelectronics. Вони широко використовуються в різних вбудованих системах, починаючи від споживчої електроніки і закінчуючи промисловою автоматизацією.

До їх ключових особливостей належать:

Архітектура: мікроконтролери STM32 базуються на архітектурі процесора ARM Cortex-M, яка забезпечує потужну та ефективну платформу для вбудованих програм. Вони пропонують ряд ядер Cortex-M, включаючи Cortex-M0, Cortex-M3, Cortex-M4, Cortex-M7 і Cortex-M33, кожне з різними рівнями продуктивності та можливостей.

Інтеграція периферійних пристроїв: мікроконтролери STM32 постачаються з широким набором інтегрованих периферійних пристроїв, таких як контакти GPIO (введення/виведення загального призначення), UART (універсальний асинхронний приймач/передавач), SPI (послідовний периферійний інтерфейс), I2C (інтегрована схема), USB (універсальна послідовна шина), АЦП (Аналогово-цифровий перетворювач), ЦАП (Цифро-аналоговий перетворювач), таймери тощо. Ці інтегровані периферійні пристрої забезпечують безперебійне взаємодія із зовнішніми пристроями та полегшують

реалізацію функцій роботу з зовнішніми датчиками.

2.2 Використані технологій

2.2.1 Мова розробки для апаратної частини

Основною мовою написання софту для мікроконтролерів є C. Вона має відносно простий синтаксис, а також підтримує конструкції низькорівневого програмування, такі як вказівники та бітові маніпуляції, які необхідні для роботи з апаратним забезпеченням мікроконтролера [15]. На додаток до своїх можливостей низького рівня, C підтримує високорівневі конструкції програмування, такі як функції, структури та масиви, які полегшують написання складних програм. Це важливо для додатків мікроконтролерів, де код може швидко стати складним, наприклад у системах керування або програмах збору даних. Таким чином, C – це універсальна та ефективна мова програмування, яка добре підходить для роботи з мікроконтролерами. Його низькорівневі можливості та підтримка конструкцій програмування високого рівня роблять його ідеальним для додатків вбудованих систем.

2.2.2 Мова розробки додатку

C++ – це потужна мова програмування, яка широко використовується для розробки настільних програм. Це розширення мови програмування C і надає додаткові функції та можливості. Ось деякі ключові характеристики та переваги C++ у контексті настільних програм: Ефективність і продуктивність: C++ дозволяє маніпулювати пам'яттю на низькому рівні та забезпечує прямий контроль над апаратними ресурсами. Це робить його ідеальним для створення високопродуктивних настільних додатків, які потребують ефективного керування пам’яттю та швидкості обчислення. Об’єктно-орієнтоване програмування: C++ підтримує парадигму об’єктно-орієнтованого

програмування (ООП), що дозволяє розробникам структурувати свій код за допомогою класів і об’єктів. ООП полегшує модульність, повторне використання коду та сприяє чистому та організованому дизайну програмного забезпечення. Стандартна бібліотека шаблонів (STL): C++ надає потужну стандартну бібліотеку шаблонів (STL), яка пропонує колекцію повторно використовуваних алгоритмів і структур даних. STL надає готові до використання контейнери, такі як вектори, списки, карти та алгоритми для ефективного сортування, пошуку та обробки даних. Кросплатформна розробка: C++ дуже портативний і може використовуватися для розробки кросплатформних настільних програм. Використовуючи незалежні від платформи бібліотеки та фреймворки, такі як Qt або wxWidgets, розробники можуть писати код, який можна скомпілювати та запускати в різних операційних системах, включаючи Windows, macOS і Linux.

2.2.3 Бібліотека HAL

У контексті мікроконтролерів STM32 HAL означає (Hardware Abstraction Layer) апаратний рівень абстракції. HAL – це програмний рівень, наданий STMicroelectronics, який абстрагує низькорівневі деталі апаратного забезпечення та забезпечує уніфікований інтерфейс для взаємодії з апаратною периферією мікроконтролерів STM32.

HAL спрощує процес розробки, надаючи узгоджений набір API і функцій для керування та налаштування периферійних пристроїв STM32, таких як GPIO, таймери, UART, SPI, I2C, ADC тощо. Він приховує складність базових апаратних регістрів і конфігурацій, дозволяючи розробникам зосередитися на логіці програми, а не на складних деталях внутрішньої частини мікроконтролера.

Абстракція: HAL абстрагує апаратні деталі низького рівня, такі як адреси реєстрів і конфігурації, надаючи інтерфейс вищого рівня, який легше зрозуміти та використовувати. Він забезпечує узгоджений API для різних сімейств

мікроконтролерів STM32, що полегшує перенесення коду між пристроями.

Портативність: HAL сприяє переносимості шляхом інкапсуляції апаратного коду. Розробники можуть писати свою логіку додатків за допомогою HAL API, і той самий код можна повторно використовувати на різних мікроконтролерах STM32 без серйозних змін.

Стандартизація: HAL відповідає стандартизованій специфікації API, визначеній STMicroelectronics. Це забезпечує послідовність і сумісність між різними мікроконтролерами STM32, полегшуючи розробникам роботу з різними пристроями сімейства STM32.

Загалом, HAL відіграє вирішальну роль в екосистемі STM32, надаючи стандартизований рівень абстракції для взаємодії з апаратною периферією мікроконтролера. Це спрощує процес розробки та покращує переносимість коду.

2.2.4 Аналогово-цифрове перетворення

АЦП (аналогово-цифровий перетворювач) – це процес перетворення аналогового сигналу в цифрове представлення. У мікроконтролерах та інших вбудованих системах перетворення АЦП зазвичай використовується для перетворення аналогових рівнів напруги в цифрові значення, які можуть бути оброблені та проаналізовані цифровою системою.

Процес перетворення АЦП включає наступні кроки:

Вибірка: АЦП періодично виконує вибірку аналогового вхідного сигналу. Вхідний сигнал утримується та вимірюється протягом певного часу, відомого як час дискретизації. Цей процес вибірки фіксує миттєве значення аналогового сигналу в кожній точці вибірки.

Квантування: дискретизована аналогова напруга потім квантується в дискретне цифрове значення. Квантування передбачає поділ діапазону вхідної напруги на кінцеву кількість рівнів або кроків. Кількість рівнів визначає роздільну здатність АЦП, зазвичай виражену в бітах. Наприклад, 8-розрядний

АЦП може представляти аналоговий сигнал із 2^8 (256) дискретними рівнями.

Перетворення: квантоване аналогове значення перетворюється на двійкове цифрове значення. Це досягається шляхом порівняння квантованого рівня напруги з еталонною напругою та створення цифрового представлення, пропорційного вхідній напрузі.

Роздільна здатність: Роздільна здатність АЦП відноситься до кількості бітів, які використовуються для представлення цифрового виходу. АЦП з вищою роздільною здатністю можуть забезпечити більш точне та точне представлення вхідного аналогового сигналу.

Вихід: перетворене цифрове значення зазвичай зберігається в регістрі або місці пам’яті для подальшої обробки. Його можна використовувати для розрахунків, алгоритмів керування або відображати користувачеві.

Процес перетворення АЦП може запускатися різними механізмами, такими як програмна ініціація або апаратні тригери. Деякі АЦП можуть виконувати безперервні перетворення, тоді як інші вимагають явних тригерів для кожного перетворення.

Важливо зазначити, що точність і продуктивність перетворень АЦП залежать від таких факторів, як роздільна здатність АЦП, частота дискретизації, якість обробки аналогового сигналу (наприклад, фільтри згладжування) і опорна напруга.

2.2.5 Протокол I2C

I2C (Inter-Integrated Circuit) – це послідовний протокол зв’язку, який дозволяє кільком пристроям спілкуватися один з одним за допомогою двопровідного інтерфейсу сигнальних ліній: SDA (Serial Data) і SCL (Serial Clock). SDA – це двонаправлена лінія даних, яка використовується для передачі та отримання даних між пристроями, тоді як SCL – це однонаправлена лінія синхронізації, яка використовується для синхронізації передачі даних між пристроями.

Зв'язок по шині I2C ініціюється головним пристроєм, який надсилає умову запуску, яка сигналізує про початок транзакції. Потім головний пристрій надсилає адресу цільового підлеглого пристрою, а потім біт читання/запису, який вказує, чи є транзакція операцією читання чи запису.

Після того як цільовий пристрій підтвердить свою адресу, головний пристрій може почати передачу або прийом даних. Кожен байт даних підтверджується приймаючим пристроєм, і транзакція завершується умовою зупинки, надісланою головним пристроєм.

2.2.6 Протокол USB HID

USB HID (Human Interface Device) – це протокол, який дозволяє USB-пристроям, таким як клавіатури, миші, ігрові контролери та інші пристрої введення, спілкуватися з головним комп’ютером. Пристрої HID розроблені за принципом plug-and-play, тобто для них не потрібно встановлювати додаткові драйвери на головному комп’ютері. Пристрої HID обмінюються даними з головним комп’ютером за допомогою стандартних кінцевих точок USB, які є логічними каналами для передачі даних між пристроєм і хостом. Пристрої HID зазвичай використовують кінцеві точки переривань, які дозволяють пристрою надсилати дані на хост, щойно вони стають доступними. Це важливо для пристроїв введення, таких як клавіатури та миші, яким потрібно надсилати дані швидко та в режимі реального часу. Пристрої HID використовують стандартизований формат звіту для зв’язку з хостом. Звіт – це блок даних, який містить інформацію про стан пристрою або вхідні дані. Наприклад, клавіатура може надіслати звіт із зазначенням того, які клавіші натиснуті в даний момент. Формат звіту визначається сторінкою використання HID та ідентифікатором використання, які вказують тип пристрою та формат даних. Пристрої HID часто використовуються у вбудованих системах та інших програмах, де потрібні пристрої введення. Вони особливо корисні для взаємодії з мікроконтролерами та іншим апаратним забезпеченням низького рівня, оскільки не потребують

встановлення додаткових драйверів чи програмного забезпечення на головному комп’ютері. Пристрої HID також можна налаштувати для підтримки певних типів введення або форматів даних, що робить їх гнучким і універсальним рішенням для широкого діапазону програм. Таким чином, USB HID – це протокол, який дозволяє USB-пристроям обмінюватися даними з головним комп’ютером за допомогою стандартизованого формату звіту. Пристрої HID розроблені за принципом «підключай і працюй» і не потребують встановлення додаткових драйверів на хості. Вони часто використовуються у вбудованих системах та інших програмах, де потрібні пристрої введення, і можуть бути налаштовані для підтримки певних типів введення або форматів даних.

2.2.7 Бібліотека OpenGL

OpenGL – це API (Application Programming Interface) візуалізації графіки, який використовується для створення високоякісної двовимірної та тривимірної комп’ютерної графіки. Він надає потужний набір функцій і інструментів для створення та керування графікою на різноманітних платформах і операційних системах.

Однією з ключових особливостей OpenGL є його здатність використовувати апаратне прискорення для відтворення графіки. Це означає, що він може використовувати переваги спеціалізованих графічних процесорів (GPU), наявних у сучасних комп’ютерах, для швидкого й ефективного виконання складних графічних операцій. Це забезпечує високоякісну візуалізацію складних 3D-сцен у режимі реального часу, що важливо для таких програм, як відеоігри та віртуальна реальність. А також інший плюс цієї бібліотеки - розширюваність: OpenGL розроблений таким чином, щоб бути розширюваним, дозволяючи постачальникам апаратного забезпечення надавати додаткові функції та розширення, окрім основної специфікації. Це дає розробникам доступ до розширених можливостей візуалізації, специфічних для певного графічного обладнання.

Загалом, OpenGL – це потужний і універсальний API для візуалізації графіки, необхідний для створення високоякісної 2D і 3D комп'ютерної графіки. Його здатність користуватися перевагами апаратного прискорення та підтримка розширених графічних функцій робить його необхідним інструментом для широкого спектру додатків, від відеоігор до наукової візуалізації.

2.2.8 Бібліотека GLUT

GLUT (OpenGL Utility Toolkit) – це бібліотека, яка надає просту міжплатформенну структуру для створення вікон і керування ними, обробки введених користувачем даних і базової взаємодії з OpenGL. Він був розроблений, щоб полегшити написання програм, які використовують OpenGL для візуалізації графіки та обробки взаємодії користувача.

GLUT пропонує набір функцій і зворотних викликів, які обробляють керування вікнами, введення користувача (клавіатура, миша) і базову обробку подій. Він абстрагує специфічні для платформи деталі та забезпечує узгоджений інтерфейс для різних операційних систем, що полегшує написання програм OpenGL, які є портативними.

Деякі ключові функції та функції GLUT включають:

Керування вікнами: GLUT спрощує створення вікон і керування ними. Він надає функції для створення вікон, встановлення властивостей вікон (таких як розмір, положення та заголовок), керування зміною розміру та закриття вікон та керування кількома вікнами.

Обробка введення: GLUT дозволяє обробляти введення користувача, наприклад, події клавіатури та миші. Він забезпечує зворотні виклики для отримання подій введення, включаючи натискання та відпускання клавіш клавіатури, рухи миші та натискання кнопок.

Функції таймера: GLUT надає функції таймера для планування періодичних оновлень або анімації. Це дозволяє встановити функцію

зворотного виклику таймера, яка викликається через регулярні проміжки часу, надаючи вам контроль над часом оновлення вашої програми.

Керування контекстом OpenGL: GLUT спрощує створення контекстів OpenGL і керування ними. Він надає функції для ініціалізації контексту OpenGL, встановлення режиму відображення та обробки подвійної буферизації.

Допоміжні функції: GLUT містить різні допоміжні функції, такі як функції для малювання основних геометричних фігур, завантаження зображень і обробки простих 2D і 3D трансформацій.

Підтримка між платформами: GLUT розроблений як кросплатформенний і працює на різних операційних системах, включаючи Windows, macOS і Linux. Він абстрагує специфічні для платформи деталі та забезпечує послідовний API для керування вікнами та обробки введення.

Важливо зазначити, що GLUT є старішою бібліотекою, і її замінили більш сучасні альтернативи, такі як GLFW (Graphics Library Framework) і SDL (Simple DirectMedia Layer). Ці новіші бібліотеки надають більше можливостей, кращу продуктивність і покращену підтримку між платформами. Однак GLUT все ще широко використовується в освітніх цілях і для швидкого створення прототипів простих програм OpenGL.

2.2.9 Бібліотека GLFW

GLFW (Graphics Library Framework) – це бібліотека C++, яка зазвичай використовується для створення вікон і керування ними, обробки введених даних користувачами та керування контекстами OpenGL. Він надає простий і крос-платформний API для створення та керування вікнами, обробки вводів з клавіатури, миші та джойстика, а також доступу до контекстів OpenGL і OpenGL ES.

GLFW спрощує процес створення крос-платформних програм, які працюють з віконими додатками, обробляють введені користувачем дані та використовують OpenGL для візуалізації графіки. GLFW широко

використовується в розробці додатків комп’ютерної графіки, ігрових двигунів та інтерактивного моделювання.

Головна причина застосування цієй бібліотеки у тому що GLFW надає єдиний інтерфейс для обробки введення з клавіатури, миші та джойстика. Він підтримує отримання стану окремих клавіш, кнопок миші та осей джойстиків, а також реєстрацію зворотних викликів для отримання подій введення. А також тому що GLFW розроблений як портативний і працює на різних платформах, включаючи Windows, macOS, Linux і різні Unix-подібні системи. Він абстрагує специфічні для платформи API і забезпечує узгоджений інтерфейс для керування вікнами та обробки введення.

2.2.10 Фотополімерний 3Д друк

Фотополімерні 3D-принтери, також відомі як полімерні 3D-принтери, використовують рідкі фотополімерні смоли, які твердіють під дією світла. Ось деякі переваги та недоліки фотополімерних 3D-принтерів:

Переваги

Відбитки з високою роздільною здатністю: фотополімерні 3D-принтери можуть створювати високодеталізовані відбитки з гладкими поверхнями та складним дизайном. Вони чудово створюють складні геометрії та дрібні деталі, які можуть бути складними для інших технологій 3D-друку.

Широкий вибір матеріалів: існує широкий вибір фотополімерних смол для різних застосувань. Ці смоли можуть мати різні властивості, такі як гнучкість, міцність, прозорість або біосумісність. Ця універсальність дозволяє створювати ряд функціональних прототипів і продуктів кінцевого використання.

Гладка поверхня: фотополімерні відбитки часто мають гладку поверхню безпосередньо з принтера, що усуває потребу в подальшій обробці в багатьох випадках. Це особливо корисно для естетичних або функціональних частин, які потребують високого рівня деталізації та якості поверхні.

Опорні конструкції: фотополімерні 3D-принтери зазвичай використовують опорні конструкції для створення виступів і складних геометрій. Ці опори легко знімаються, а опори на основі смоли часто не залишають видимих слідів або пошкоджень на готовому відбитку або майже не залишають їх.

Недоліки:

Обмежений об’єм збірки: фотополімерні 3D-принтери зазвичай мають менші об’єми виготовлення порівняно з іншими технологіями 3D-друку, такими як принтери моделювання наплавлення (FDM). Це обмежує розмір об’єктів, які можна надрукувати як одну частину, вимагаючи, щоб більші об’єкти друкували кількома секціями та збирали пізніше.

Вищі витрати на матеріали: фотополімерні смоли можуть бути дорожчими порівняно з нитками, які використовуються в принтерах FDM. Вартість смоли може збільшитися, особливо для великих або складних відбитків. Крім того, деякі смоли мають обмежений термін придатності, і їх, можливо, потрібно буде замінити, якщо вони не будуть використані протягом певного періоду часу.

Довший час друку: фотополімерний 3D-друк зазвичай займає більше часу порівняно з деякими іншими технологіями. Процес затвердіння для кожного шару може зайняти багато часу, що призведе до довшого друку складних або великих моделей. Це може вплинути на продуктивність і час виконання проектів.

Вимоги до постобробки: хоча фотополімерні відбитки мають гладку поверхню, вони часто вимагають додаткових етапів постобробки. Це може включати видалення опор, очищення надлишків смоли та затвердіння відбитків під ультрафіолетовим світлом для досягнення їх остаточних механічних властивостей. Постобробка може додати додатковий час і зусилля для загального робочого процесу.

2.3 Методологічне підгрунтя для рішення поставленої задачі

Фільтр Калмана – це математичний алгоритм, який використовується для оцінки стану системи на основі серії шумових вимірювань. Він зазвичай використовується в системах відстеження руху для оцінки положення та швидкості об’єкта на основі вимірювань таких датчиків, як акселерометри, гіроскопи та магнітометри. У системі відстеження руху фільтр Калмана отримує вимірювання від датчиків і використовує їх для оцінки стану системи, яка включає положення, швидкість та інші відповідні змінні. Фільтр також моделює невизначеності вимірювань і динаміку системи та використовує цю інформацію для підвищення точності оцінок. Фільтр Калмана працює за допомогою моделі переходу стану та моделі вимірювання для оцінки стану системи. Модель переходу стану описує, як стан системи змінюється з часом, тоді як модель вимірювання пов’язує вимірювання зі станом системи. Потім фільтр використовує ці моделі для прогнозування стану системи на кожному кроці часу та порівнює прогнози з фактичними вимірюваннями, щоб оновити оцінку. Однією з ключових переваг фільтра Калмана є його здатність справлятися з шумовими вимірюваннями та невизначеністю в динаміці системи. Це робиться шляхом поєднання вимірювань із прогнозованою оцінкою стану та їх зважування на основі відповідної невизначеності. Це дозволяє фільтру підвищити точність оцінок навіть за наявності шумних даних. У системах відстеження руху фільтр Калмана можна використовувати для оцінки положення, швидкості та орієнтації об’єкта на основі даних датчиків, таких як акселерометри, гіроскопи та магнітометри. Поєднуючи ці вимірювання, фільтр може забезпечити більш точні та надійні оцінки руху об’єкта.

Основна ідея фільтра Калмана полягає в рекурсивному оновленні оцінки стану системи на основі двох джерел інформації: вимірювань від датчиків і прогнозів з математичної моделі системи. Фільтр підтримує два ключових компоненти: оцінку стану та коваріаційну матрицю. Оцінка стану – це вектор, який представляє найкраще припущення про поточний стан системи на основі

всієї доступної інформації до поточного часу. Коваріаційна матриця є мірою невизначеності або помилки в оцінці стану. Фільтр Калмана працює в два етапи: етап прогнозування та етап оновлення. На етапі прогнозування фільтр використовує математичну модель для прогнозування стану системи на наступному часовому етапі на основі поточної оцінки стану. Цей прогноз потім використовується для обчислення коваріаційної матриці для прогнозованого стану. На етапі оновлення фільтр використовує фактичні дані вимірювання, щоб виправити прогнозовану оцінку стану та коваріаційну матрицю. Фільтр Калмана обчислює приріст, який є ваговим коефіцієнтом, який визначає, наскільки довіряти прогнозованому оцінюванню стану порівняно з вимірюванням. Потім вимірювання використовується для оновлення оцінки стану та коваріаційної матриці, які потім використовуються на наступному етапі прогнозування. Фільтр Калмана призначений для обробки шумових або неповних вимірювань, і він здатний включати нові вимірювання, коли вони стають доступними, постійно уточнюючи свою оцінку стану системи. Використовуючи як математичну модель, так і фактичні вимірювання, фільтр Калмана може забезпечити точнішу та надійнішу оцінку стану системи, ніж будь-яке з цих джерел окремо.

3 МОДЕЛЬ ТА ЇЇ РЕАЛІЗАЦІЯ

3.1 Опис моделі

Концептуальна модель відстеження людських кінцівок за допомогою датчиків MEMS (мікроелектромеханічних систем) та механічних ставить перед собою групу технічних питань, які власне і будуть описом моделі у цілому.

Розташування датчиків: датчики, такі як гіроскопи та акселерометри, потенциометри стратегічно розміщені та націлені на різні сегментах тіла, щоб фіксувати рух і орієнтацію кожної кінцівки.

Збір даних датчиків: MEMS датчики постійно вимірюють кутову швидкість і прискорення кісті людини, а також зміні резистори фіксують зміну положення пальців. Ці вимірювання фіксуються датчиками та надсилаються в блок обробки для подальшого аналізу.

Об’єднання датчиків: Дані від кількох датчиків об’єднуються, щоб отримати повне представлення руху та орієнтації кінцівки. Алгоритми об’єднання датчиків, такі як фільтри Калмана або додаткові фільтри для MEMS сенсорів, і медіанні фільтри для механічних, використовуються для об’єднання даних і отримання більш точної оцінки положення та руху кінцівки.

Калібрування та ініціалізація: перед початком відстеження датчики можуть пройти процес калібрування, щоб забезпечити точні вимірювання. Це передбачає визначення зсуву датчика, компенсацію похибок датчика та встановлення початкових опорних положень і орієнтації.

Обробка в реальному часі: обробка даних датчика та алгоритми відстеження руху реалізовані на процесорі, такому як мікроконтролер. Можливості обробки в реальному часі мають вирішальне значення для захоплення та інтерпретації рухів кінцівок майже в режимі реального часу для таких програм, як віртуальна реальність, спортивний аналіз або реабілітація.

Візуалізація вихідних даних: відстежувані дані кінцівок можна візуалізувати різними способами залежно від програми. Це може включати

відображення положення кінцівки, орієнтації, кутів суглобів або відображення відстежуваних даних на віртуальному аватарі чи графічному представленні.

Інтеграція з конкретними програмами: відстежувані дані кінцівок можна інтегрувати в різні програми залежно від конкретного випадку використання. Це може включати керування віртуальними персонажами чи аватарами в іграх чи анімації, надання зворотного зв’язку для аналізу спортивних результатів чи реабілітаційних вправ або керування пристроями тактильного зворотного зв’язку для захоплюючих вражень.

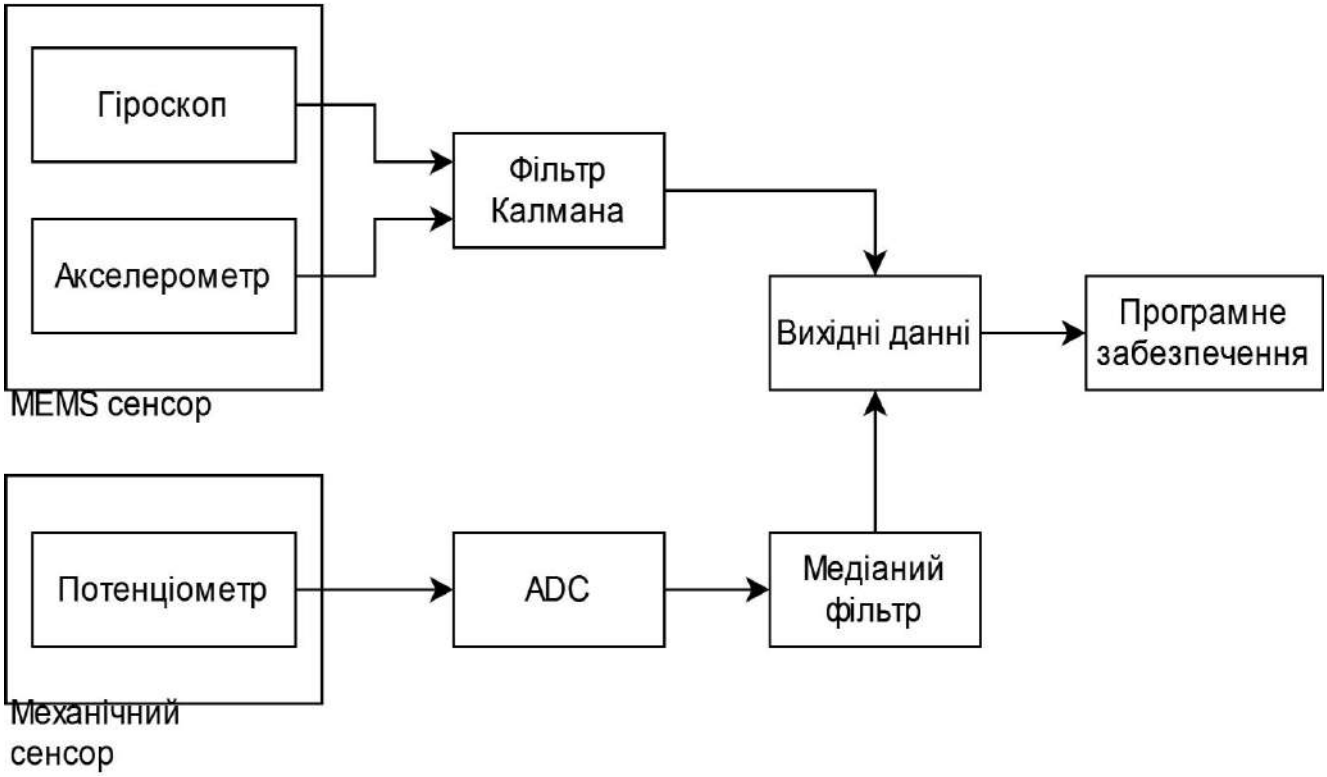

Рисунок 3.1 – Загальна схема моделі

Важливо зазначити, що конкретна реалізація та алгоритми, які використовуються для відстеження людських кінцівок за допомогою датчиків, можуть відрізнятися залежно від бажаного рівня точності, вимог програми та доступних ресурсів.

3.2 Реалізація апаратної частини

В якості обчислювального модуля, було обрано мікроконтролер STM32F401CCU6, котрий призначений для використання в широкому спектрі застосувань, включаючи промислову автоматизацію, побутову електроніку та пристрої Інтернету речей (IoT). Він має тактову частоту до 84 МГц, до 512 КБ флеш-пам'яті та до 96 КБ SRAM. Він також включає низку периферійних пристроїв, включаючи кілька послідовних інтерфейсів, таймери та аналого-цифрові перетворювачі. Основною перевагою цього мікроконтролера є модуль з плаваючою комою, який прискорює виконання математичних операцій. Тобто він здатний покрити усі поставленні вимоги.

Іншою важливою частиною системи є MEMS датчик, який буде передавати данні про рух. Для цієї цілі використано MPU-6050 – це 6-осьовий датчик, який зручно поєднує гіроскоп і акселерометр в одному чіпі. MPU-6050 зазвичай використовується в таких додатках, як робототехніка, дрони та системи захоплення руху, де потрібне точне вимірювання орієнтації та руху. Він також широко використовується в проектах любителів через його низьку вартість і простоту використання. Хоча MPU-6050 є популярним і широко використовуваним IMU, він має деякі обмеження. Одним із основних обмежень є відсутність магнітометра, необхідного для точної оцінки орієнтації за наявності магнітних перешкод. Але його головна особливість це можливість використовувати одне фізичне підключення та збирати дані через одну шину I2C.

Щоб встановити зв’язок між двома пристроями за допомогою I2C, кожен пристрій повинен мати можливість підтягнути лінії SDA та SCL до логічного високого рівня (тобто 1) і звільнити їх, щоб їх можна було підтягнути до логічного низького рівня (тобто 0). Однак, якщо жоден із пристроїв не підтягує лінії SDA та SCL до високого рівня, тоді лінії залишаться плаваючими, і зв’язок не працюватиме. Ось чому підтягуючі резистори необхідні для зв’язку I2C. Підтягувальні резистори підключаються між лініями SDA і SCL і напругою

живлення (Vcc) системи. Вони забезпечують шлях для підтягування ліній до логічного високого рівня, коли жоден із пристроїв їх активно не керує. Значення підтягуючого резистора визначає швидкість, з якою лінії повертаються до логічного високого рівня, коли вони не активні. Якщо значення занадто низьке, лінії можуть не досягти логічного високого рівня вчасно, що призведе до помилок зв'язку. З іншого боку, якщо значення занадто високе, то лінії можуть не мати достатнього струму, щоб з часом досягти логічного низького рівня, що призведе до повільного зв'язку. У зв'язку з цим було обрано класичне значення резисторів R1, R2 4.7 кілоом.

Потенциомітри R_POT0 – R_POT3 використовуються для отримання даних о положені пальців людини. Їх центральний вивід підключен до пінів мікроконтролерру з аналогово-цифровим перетворювачем, який цифрує аналоговий сигнал та передає його числеве значення.

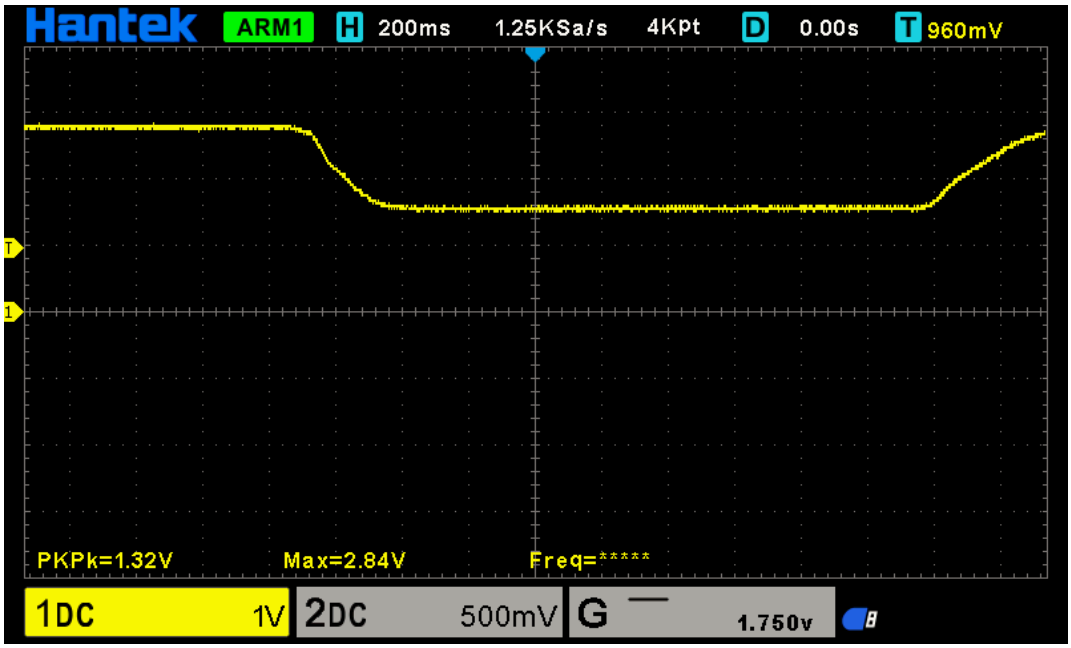

Рисунок 3.2 – Сингнал потенциомітр при руху пальця

Для передачі оброблених даних для їхньої подальшої обробки та візуалізації використана USB шина з HID інтерфейсом для більшої сумістності між цілоьвими платформами відключення.

Крім передачі даних, USB дозволяє пристрію отримувати живлення від порту при підключенні до комп'ютера, саме через це була введена вимога на невелике енергоспоживання, щоб вкластися у межі стандартів протоколу в 0.5A.

Для перетвореня 5 вольт USB порту до необхідних 3.3 вольту для роботи датчику та мікроконтролеру був використаний простий лінійний перетворувач напруги. Потреби в живленні данной схеми не є дуже великими: 100-200mA для STM32F401 та не більше 10mA для MPU-6050, тому якісь більш серьозні перетворювачи по типи імпульсних тут не потрібні.

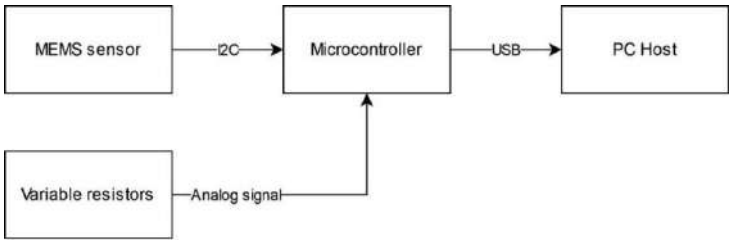

Рисунок 3.3 – Загальна схема

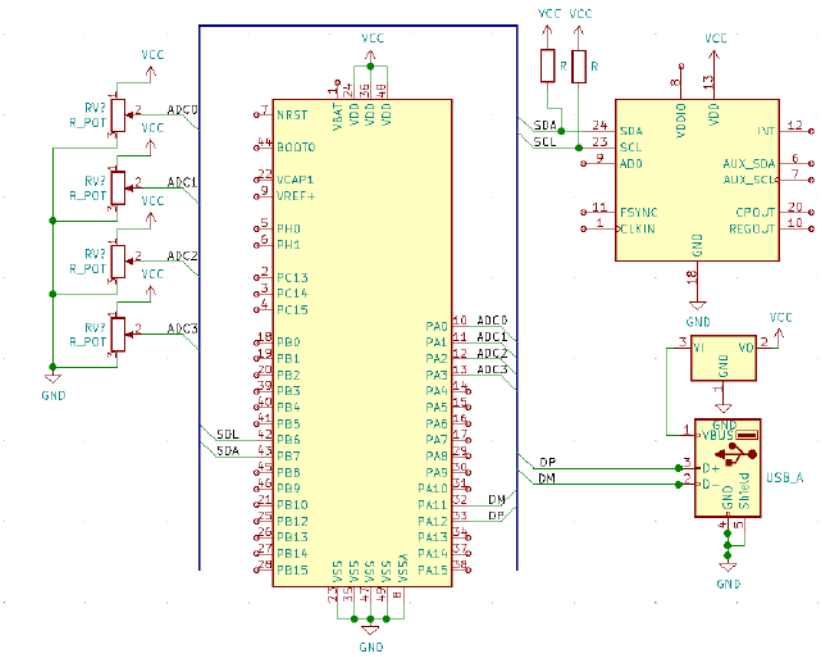

Рисунок 3.4 – Схема підключення мікроконтролера

Також крім електричної схеми була спроектована 3Д модель прістрою з механізмом трекінку пальців через механічні даичики – потенціометри, а також з кріпліннням для MEMS сенсору для захвату рухів цілої кісті людини. Додатково модель має дуговий виріз під контур кісті та отвори під два ремні, що забезпечую комфорт і важливіше міцне та стійке до різких рухів креління.

За результатами розробки була надрукована фізична модель на 3Д принтері з використанням фотополімеру, через що незначно знизилась міцність девайсу, але було досягнено надзвичайної точності вихідних деталей відносно оригінального чертежу у 0.05мм. Це забезпечило відноносну простоту зборки всієї конструкції за рахут гарного прилягання деталей, а також плавний хід роботи усіх механізмів, що дуже важливо, та не додає додаткової нагрузки для людини при роботі з пристроєм.

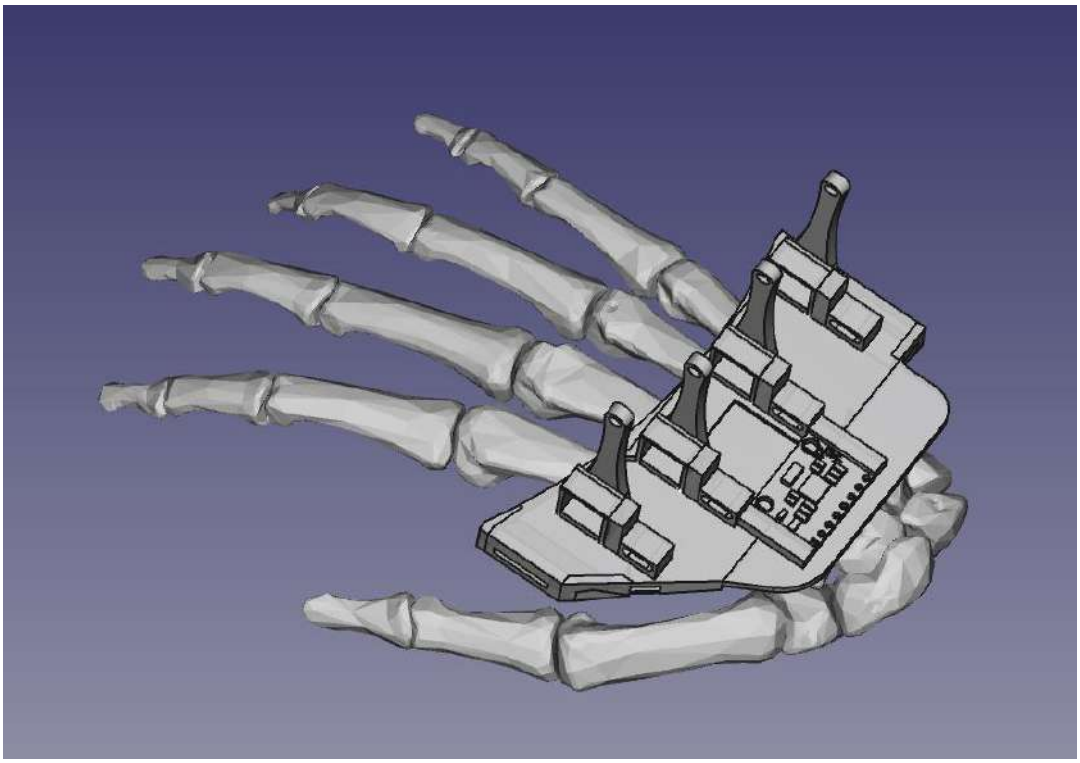

Рисунок 3.5 – 3Д модель пристрою

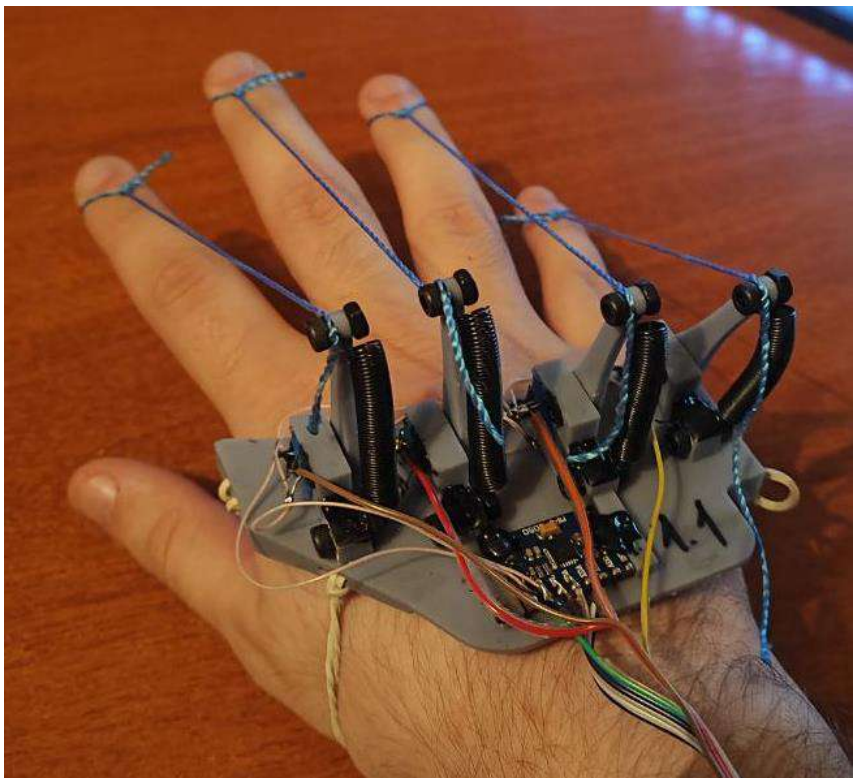

Рисунок 3.6 – Фото прототипу

3.3 Програмне забезпечення пристрою

3.3.1 Робота з MEMS сенсором

Найголовнішою частиною пристрою є мікро-електроміханічний датчик, який працює з шиною I2C. Для забезпечення роботи з сенсором MPU6050 потрібно спершу ініціалізувати I2C шина за допомогою HAL бібліотеки. А також імплементувати функцію для ініціалізації власне самого сенсора по цій шині.

Лістинг 3.1 – Ініціалізація I2C

```
I2C_HandleTypeDef hi2c1;
void MX_I2C1_Init(void)
{
  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 400000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
```

```
  hi2c1.Init.OwnAddress1 = 0x10;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0x11;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }
}
```

Лістинг 3.2 – Ініціалізація сенсору

```
uint8_t MPU6050_Init(I2C_HandleTypeDef *I2Cx)
{
    uint8_t check;
    uint8_t Data;
    HAL_I2C_Mem_Read(I2Cx, MPU6050_ADDR, WHO_AM_I_REG, 1, &check, 1,
i2c_timeout);

    if (check == 104) // 0x68 will be returned by the sensor if everything
goes well
    {
        Data = 0;
        HAL_I2C_Mem_Write(I2Cx, MPU6050_ADDR, PWR_MGMT_1_REG, 1, &Data, 1,
i2c_timeout);
        Data = 0x07;
        HAL_I2C_Mem_Write(I2Cx, MPU6050_ADDR, SMPLRT_DIV_REG, 1, &Data, 1,
i2c_timeout);
        Data = 0x00;
        HAL_I2C_Mem_Write(I2Cx, MPU6050_ADDR, ACCEL_CONFIG_REG, 1, &Data,
1, i2c_timeout);
        Data = 0x00;
        HAL_I2C_Mem_Write(I2Cx, MPU6050_ADDR, GYRO_CONFIG_REG, 1, &Data,
1, i2c_timeout);
        return 0;
    }
    return 1;
}
```

Після ініціалізаціїї для забеспечення нормальної роботи сенсору його необхідно калібрувати, це зменшить початковий дрифт значень, що приведе до більш точних значень. Калібруваня відбувається через запис значень до регістрів сенсору, необхідні значення вираховуються за допомогою PID сенсору[34], котрий знаходить значення з мінімальною похибкою.

Лістинг 3.3 – Калібруваня сенсору

```
void CalibrateAccel(uint8_t Loops )
```

```
{
      float kP = 0.3;
      float kI = 20;
      float x;
      x = (100 - map(Loops, 1, 5, 20, 0)) * .01;
      kP *= x;
      kI *= x;
      PID( 0x3B, kP, kI,  Loops);
}
```

Після початкової настоїки та калібрування можна отримати доступ до даних сенсору, що відбувається через читання регістрів через функцію HAL_I2C_Mem_Read бібліотеки HAL та збереження результатів до масиву у власну пам’ять для обробки значень.

Лістинг 3.4 – Отримання даних з сенсору

```
void MPU6050_Read_Gyro(I2C_HandleTypeDef *I2Cx, MPU6050_t *DataStruct)
{
    uint8_t Rec_Data[6];
    HAL_I2C_Mem_Read(I2Cx, MPU6050_ADDR, GYRO_XOUT_H_REG, 1, Rec_Data, 6,
i2c_timeout);

    DataStruct->Gyro_X_RAW = (int16_t)(Rec_Data[0] << 8 | Rec_Data[1]);
    DataStruct->Gyro_Y_RAW = (int16_t)(Rec_Data[2] << 8 | Rec_Data[3]);
    DataStruct->Gyro_Z_RAW = (int16_t)(Rec_Data[4] << 8 | Rec_Data[5]);
    DataStruct->Gx = DataStruct->Gyro_X_RAW / 131.0;
    DataStruct->Gy = DataStruct->Gyro_Y_RAW / 131.0;
    DataStruct->Gz = DataStruct->Gyro_Z_RAW / 131.0;
}
```

3.3.2 Обробка даних з MEMS сенсору

MEMS сенсор повертає сирі дані гіроскопу та акселерометру, які майже не можливо використовувати для роботи та аналізу, тому вони потребують обробки, в даному випадку цим займається фільтр Калмана.

Основна ідея фільтра Калмана полягає в рекурсивному оновленні оцінки стану системи на основі двох джерел інформації: вимірювань від датчиків і прогнозів з математичної моделі системи. Фільтр підтримує два ключових компоненти: оцінку стану та коваріаційну матрицю [16]. Оцінка стану — це вектор, який представляє найкраще припущення про поточний стан системи на основі всієї доступної інформації до поточного часу. Коваріаційна матриця є

мірою невизначеності або помилки в оцінці стану. Фільтр Калмана працює в два етапи: етап прогнозування та етап оновлення. На етапі прогнозування фільтр використовує математичну модель для прогнозування стану системи на наступному часовому етапі на основі поточної оцінки стану. Цей прогноз потім використовується для обчислення коваріаційної матриці для прогнозованого стану. На етапі оновлення фільтр використовує фактичні дані вимірювання, щоб виправити прогнозовану оцінку стану та коваріаційну матрицю. Фільтр Калмана обчислює приріст Калмана, який є ваговим коефіцієнтом, який визначає, наскільки довіряти прогнозованому оцінюванню стану порівняно з вимірюванням. Потім вимірювання використовується для оновлення оцінки стану та коваріаційної матриці, які потім використовуються на наступному етапі прогнозування. Фільтр Калмана призначений для обробки шумових або неповних вимірювань і може включати нові вимірювання, щойно вони стануть доступними, постійно уточнюючи свою оцінку стану системи[17]. Використовуючи як математичну модель, так і фактичні вимірювання, фільтр Калмана може забезпечити точнішу та надійнішу оцінку стану системи, ніж будь-яке з цих джерел окремо.

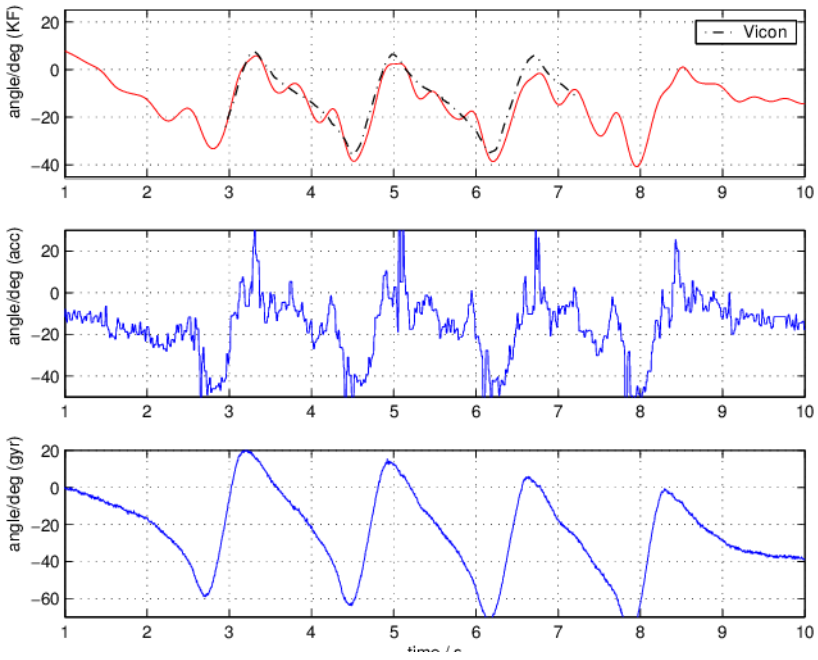

Рисунок 3.7 – Графік роботи фільтра Калмана

Головний недолік у тому, що метод розрахунку малозастосовний для розрахунку кута повороту по осі Z, оскільки для його розрахунку знадобляться проекції accel_angle_x і accel_angle_y, а в даному випадку вони майже дорівнюють нулю [18]:

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases}$$

Рисунок 3.8 – Визначення arctan

Тому розрахунок цього кута неможливий або буде виконано з великою похибкою. Розрахунок потрібного кута повинен проводитися тільки за значенням гіроскопа [19].

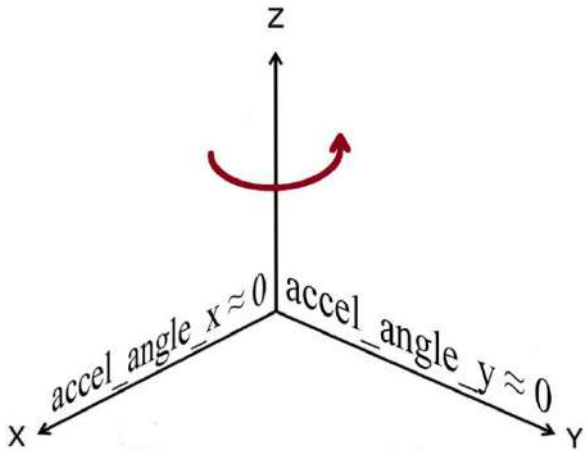

Рисунок 3.9 – Проекція на осі

Крім того, для отримання більш точного кута повороту по осі Z альтернативою є використання допоміжного модуля магнітометра – цифрового компаса. Модуль магнітометра може надавати необроблені дані магнітного

поля, які є вимірюваннями в напрямках X, Y та Z. Шляхом обчислення та віднімання кута магнітного відхилення від кута нахилу можна отримати кут повороту осі Z.

З іншого боку, фільтр Калмана здатний працювати зі складнішими системами та може включати кілька датчиків та інших джерел інформації для підвищення точності оцінки [20]. Однак це вимагає більшої обчислювальної потужності і його реалізація може бути складнішою.

Лістинг 3.5 – Реалізація фільтра Калмана

```
          double dt = (double)(HAL_GetTick() - timer) / 1000;
          timer = HAL_GetTick();
          double roll;
          double roll_sqrt = sqrt(
              DataStruct->Accel_X_RAW * DataStruct->Accel_X_RAW + DataStruct-
>Accel_Z_RAW * DataStruct->Accel_Z_RAW);
          if (roll_sqrt != 0.0)
          {
              roll = atan(DataStruct->Accel_Y_RAW / roll_sqrt) * RAD_TO_DEG;
          }
          else
          {
              roll = 0.0;
          }
          double    pitch    =    atan2(-DataStruct->Accel_X_RAW,    DataStruct-
>Accel_Z_RAW) * RAD_TO_DEG;
          if ((pitch < -90 && DataStruct->KalmanAngleY > 90) || (pitch > 90 &&
DataStruct->KalmanAngleY < -90))
          {
              KalmanY.angle = pitch;
              DataStruct->KalmanAngleY = pitch;
          }
          else
          {
              DataStruct->KalmanAngleY    =    Kalman_getAngle(&KalmanY,    pitch,
DataStruct->Gy, dt);
          }
          if (fabs(DataStruct->KalmanAngleY) > 90)
              DataStruct->Gx = -DataStruct->Gx;
          DataStruct->KalmanAngleX = Kalman_getAngle(&KalmanX, roll, DataStruct-
>Gx, dt);
```

3.3.3 Робота з механічними сенсорами

Робота з механічними датчиками полягає у зчитуванні аналогово сигналу з центрального виводу резисторів за допомоги ADC.

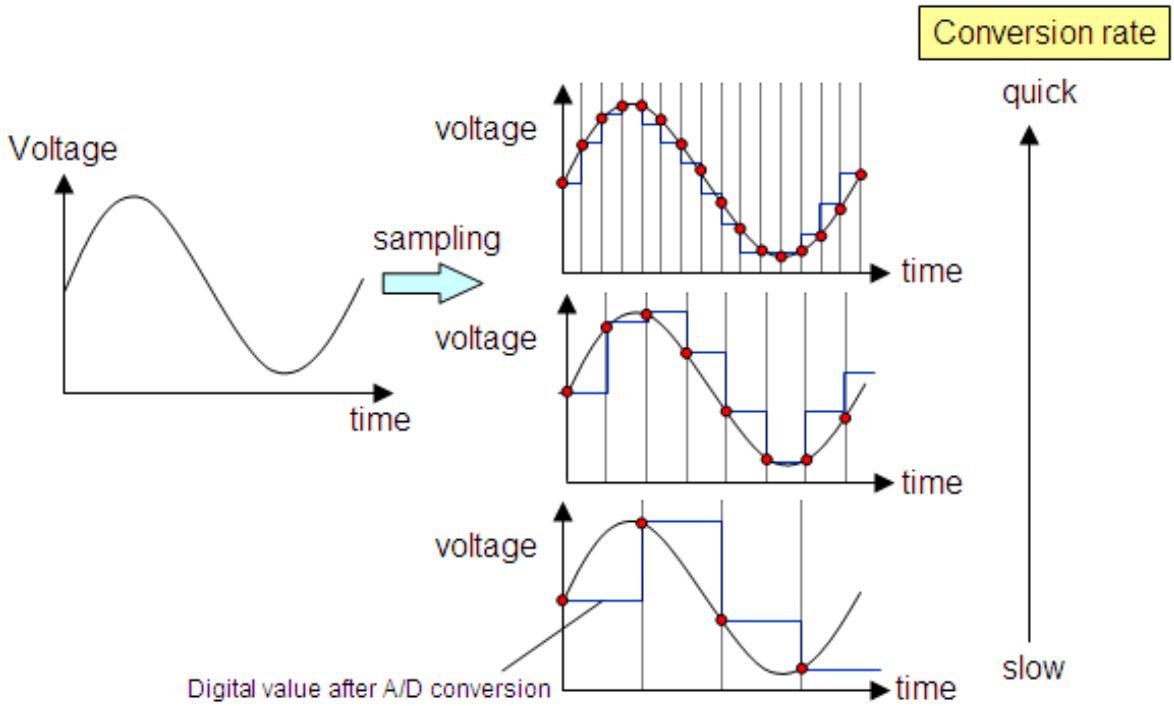

Рисунок 3.10 – Приклад роботи ADC

Тобто уся робота зводиться до ініціалізації ADC перефірії мікроконіролера, запуску перетворень та отриманню значень за допомогою функції HAL_ADC_GetValue().

Лістинг 3.6 – Ініціалізація ADC

```
ADC_HandleTypeDef hadc;
void ADC_Init(void)
{
  ADC_ChannelConfTypeDef sConfig;

  // Initialize ADC peripheral
  hadc.Instance = ADC1;
  hadc.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc.Init.ContinuousConvMode = DISABLE;
  hadc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc.Init.NbrOfConversion = 1;
  HAL_ADC_Init(&hadc);

  // Configure ADC channel
  sConfig.Channel = ADC_CHANNEL_0;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_239CYCLES_5;
  HAL_ADC_ConfigChannel(&hadc, &sConfig);
}
```

Отримані значення достатньо отфільтрувати медіанним фільтром для зменшення похибки роботи ADC. Медіанний фільтр – це тип фільтра обробки цифрового сигналу, який зазвичай використовується для зменшення шуму або відхилень у сигналі. Він працює шляхом заміни кожної точки даних середнім значенням підмножини сусідніх точок даних [21]. У медіанному фільтрі ковзне вікно певного розміру рухається через вхідний сигнал. Для кожної позиції вікна збираються точки даних у вікні та обчислюється середнє значення. Медіана – це середнє значення у відсортованому списку чисел, де половина чисел є більшими, а половина – меншими. Замінюючи кожну точку даних середнім значенням, медіанний фільтр ефективно усуває викиди або шум, який може бути присутнім у вихідному сигналі. Це особливо ефективно для зменшення імпульсного шуму, який складається з раптових і різких змін значень сигналу. Ключова перевага медіанного фільтра полягає в тому, що він зберігає краї та деталі сигналу, одночасно ефективно зменшуючи шум. На відміну від інших фільтрів, таких як середній або ковзаючий середній, медіанний фільтр не розмиває та не спотворює різкі переходи в сигналі. Однак медіанний фільтр також має деякі обмеження. Це може призвести до невеликої затримки вихідного сигналу через час, необхідний для збору та обчислення середнього значення. Крім того, він може бути не настільки ефективним у зменшенні певних типів шуму, які не нагадують викиди або імпульсивні збурення. Але у даннії схемі такі випадки будуть результатом поломки пристрою і їх не потрібно опрацьовувати окремо.

Лістинг 3.7 – Реалізація медіанного фільтра

```
float median_filter(float* data, int length) {

    // Create a temporary copy of the data
    float temp[length];
    for (int i = 0; i < length; i++) {
        temp[i] = data[i];
    }

    // Sort the temporary copy in ascending order
    for (int i = 0; i < length - 1; i++) {
        for (int j = i + 1; j < length; j++) {
```

```
            if (temp[j] < temp[i]) {
                float tempVal = temp[i];
                temp[i] = temp[j];
                temp[j] = tempVal;
            }
        }
    }

    // Find the median value
    float median;
    if (length % 2 == 0) {
        median = (temp[length / 2 - 1] + temp[length / 2]) / 2.0;
    } else {
        median = temp[length / 2];
    }
    return median;
}
```

3.2.4 Робота з USB

Протокол USB (універсальна послідовна шина) є широко використовуваним стандартом для підключення пристроїв до комп'ютерної системи. Він забезпечує простий і універсальний інтерфейс для зв'язку та передачі даних між пристроями та хостами. Протокол USB складається з різних рівнів і специфікацій, які визначають, як взаємодіють пристрої та хости.

USB працює в архітектурі головний-підлеглий, де хост (зазвичай комп'ютер) діє як головний і контролює зв'язок з одним або кількома пристроями USB. Хост ініціює зв'язок за допомогою дескрипторів, надсилаючи на пристрої запити на керування, групові запити, запити на переривання або ізохронну передачу, які відповідають відповідним чином.

Дескриптори USB – це структури даних, які надають інформацію про пристрій USB та його можливості. Вони є важливою частиною нумерації USB-пристроїв і зв'язку. Дескриптори USB визначені в специфікації USB і використовуються хостом для розуміння характеристик підключеного пристрою USB.

Дескриптор пристрою: надає загальну інформацію про пристрій USB, наприклад версію USB, клас пристрою, підклас, протокол, постачальника та ідентифікатори продукту, кількість конфігурацій тощо. Дескриптор пристрою отримує хост під час нумерації пристроїв. Дескриптор конфігурації: описує

різні конфігурації, що підтримуються пристроєм USB. Кожна конфігурація може мати один або кілька інтерфейсів і кінцевих точок. Дескриптор конфігурації надає інформацію про вимоги до живлення, максимальне енергоспоживання та інші деталі конфігурації. Дескриптор інтерфейсу: він визначає інтерфейс у конфігурації та вказує кількість кінцевих точок, клас інтерфейсу, підклас, протокол та інші деталі, що стосуються інтерфейсу [22].

Дескриптор кінцевої точки: описує характеристики кінцевої точки, такі як її адреса, тип (контрольний, груповий, переривання чи ізохронний), напрямок (вхідний або вихідний), максимальний розмір пакету та інтервал опитування для кінцевих точок переривання.

Дескриптор рядка: він надає зрозумілу людині рядкову інформацію, наприклад виробника, назву продукту, серійний номер та іншу інформацію, що стосується певної мови. Дескриптори рядків є необов’язковими та можуть використовуватися для надання локалізованої інформації про пристрій. Дескриптори USB зазвичай зберігаються у мікропрограмі пристрою USB і витягуються хостом під час процесу нумерації. Хост використовує інформацію, надану дескрипторами, щоб визначити, як спілкуватися з пристроєм і завантажити відповідний драйвер пристрою.

Вивчаючи дескриптори USB, хост може визначити клас пристрою та його можливості, розподілити ресурси та встановити необхідні канали зв’язку. Дескриптори USB відіграють важливу роль у забезпеченні належного зв’язку та сумісності між USB-пристроями та хостами.

У даному проекті пристрою надан клас USB девайсу як HID, дана специфікація класу, дозволяє забезпечити стандартизований інтерфейс для людського введення, що робить їх легко впізнаваними та придатними для використання в різних операційних системах. Тобто операційна система розпізнає цей пристрій як базовий девайс вводу інформації та не потребує додаткових драйверів, зі специфікою Plug-and-Play і може автоматично розпізнавати та налаштовувати пристрої HID.

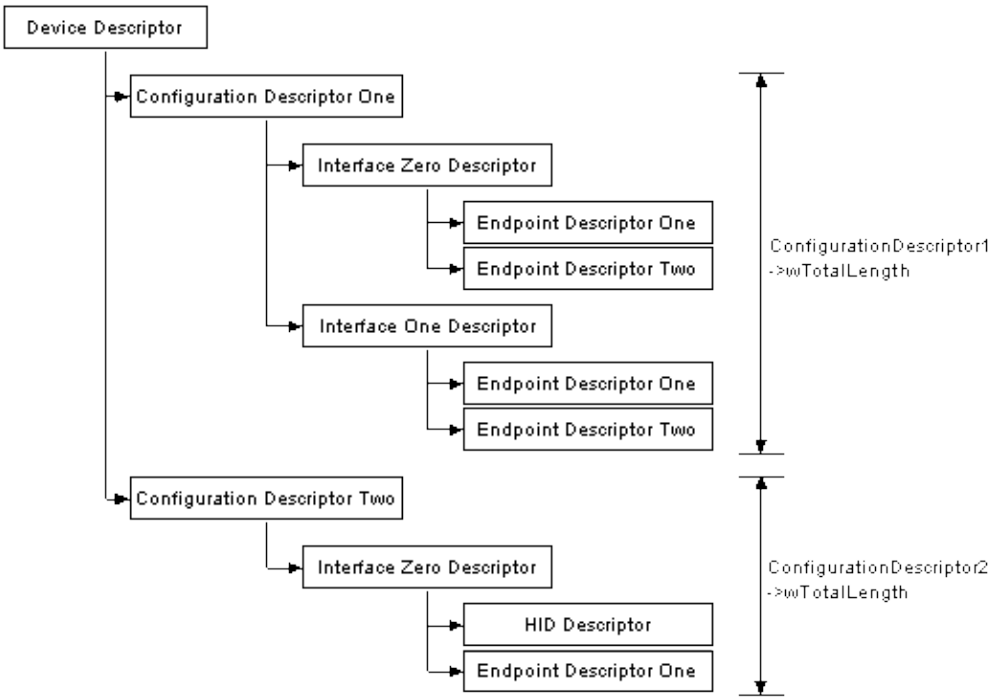

Рисунок 3.11 – Загальна структура дескрипторів

Власне для старту роботи з USB необхідно створити дескриптор та правильно описати специфікацію пристрою, щоб хост міг його опізнати. У даному випадку необхідно виділити як мінімум 8 осей для передачі інформації о положенні пальців руки та кісті.

Лістинг 3.8 – HID дескриптор

```
const uint8_t HID_ReportDescriptor[] = {
  // Usage Page (Generic Desktop)
  0x05, 0x01,
  // Usage (Joystick)
  0x09, 0x04,
  // Collection (Application)
  0xA1, 0x01,
  // Axes
  // 8 Axes (X, Y, Z, Rx, Ry, Rz, Slider, Dial)
  0x05, 0x01,          // Usage Page (Generic Desktop)
  0x09, 0x01,          // Usage (Pointer)
  0xA1, 0x00,          // Collection (Physical)
  0x15, 0x00,          // Logical Minimum (0)
  0x26, 0xFF, 0x00,    // Logical Maximum (255)
  0x75, 0x08,          // Report Size (8 bits)
  0x95, 0x08,          // Report Count (8)
  0x09, 0x30,          // Usage (X)
```

```
  0x09, 0x31,          // Usage (Y)
  0x09, 0x32,          // Usage (Z)
  0x09, 0x33,          // Usage (Rx)
  0x09, 0x34,          // Usage (Ry)
  0x09, 0x35,          // Usage (Rz)
  0x09, 0x36,          // Usage (Slider)
  0x09, 0x37,          // Usage (Dial)
  0x81, 0x02,          // Input (Data, Variable, Absolute)
  0xC0,                // End Collection (Physical)
  // Buttons
  // 8 Buttons
  0x05, 0x09,          // Usage Page (Button)
  0x19, 0x01,          // Usage Minimum (Button 1)
  0x29, 0x08,          // Usage Maximum (Button 8)
  0x15, 0x00,          // Logical Minimum (0)
  0x25, 0x01,          // Logical Maximum (1)
  0x75, 0x01,          // Report Size (1 bit)
  0x95, 0x08,          // Report Count (8)
  0x81, 0x02,          // Input (Data, Variable, Absolute)
  0xC0                 // End Collection (Application)
};
```

Описану структуру необхідно передати до HAL бібліотеки USB перефірії USBD_RegisterClass(&hUsbDeviceFS, & HID_ReportDescriptor), таким чином мікроконтролер сам передасть інформацію для роботи з ним і операційна система почне з ним працювати одразу після підключення.

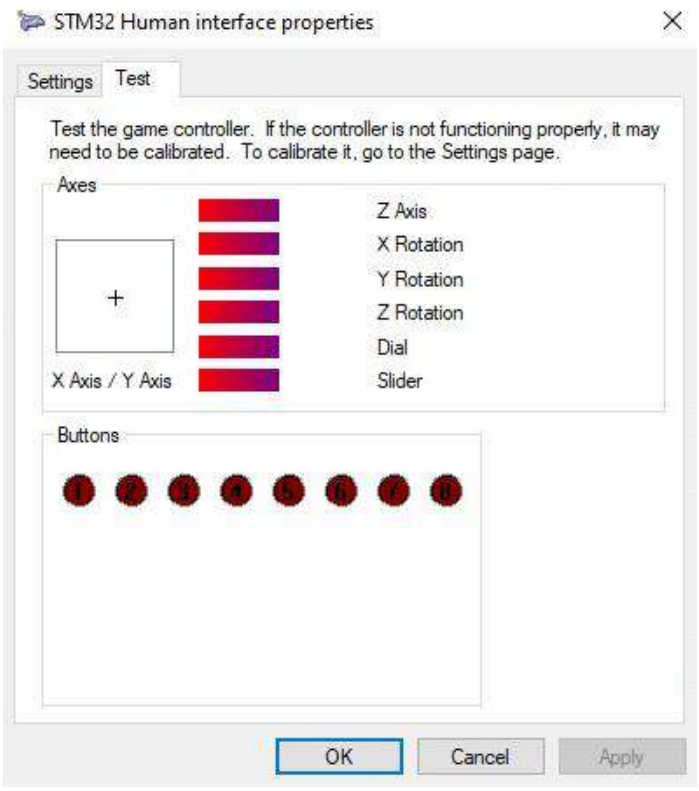

Рисунок 3.12 – Скріншот інтерфейсу підключеня

Після завершення ініціалізації можна почати передавати актуальну інформацію, для цього необхідно створити структуру відповідну описану у дескрипторі.

Лістинг 3.9 – Структура даних HID дескриптор

```
typedef struct USB_REPORT {
	int8_t x;
	int8_t y;
	int8_t z;
	int8_t rx;
	int8_t ry;
	int8_t rz;
	int8_t sl;
	int8_t dl;
	uint8_t buttons;
} USB_REPORT_X;
```

Заповнивши поля структури даними, залишається тільки її передати за допомогою функції MX_USB_Send_Report((uint8_t*)&usb_data, sizeof(USB_REPORT_X)) та комп’ютер сам обновить дані.

4 ВІДОБРАЖЕНЯ І АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Підключення до комп’ютеру

Для роботи з пристроєм візуалізації та збору інформації було розроблено просте програмне забезпечення. Воно сканує систему та знаходить HID пристрої, які підлючені до компьютера. Після програма шукає цільовий пристрій по імені та підключається до нього. Головне призначення цього забеспечення - це відображення данних з пристрою через невеликі проміжки часу для їч подальшого аналізу та порівняння. Додатково программа може візуалізувати поточні данні у окремому вікні.

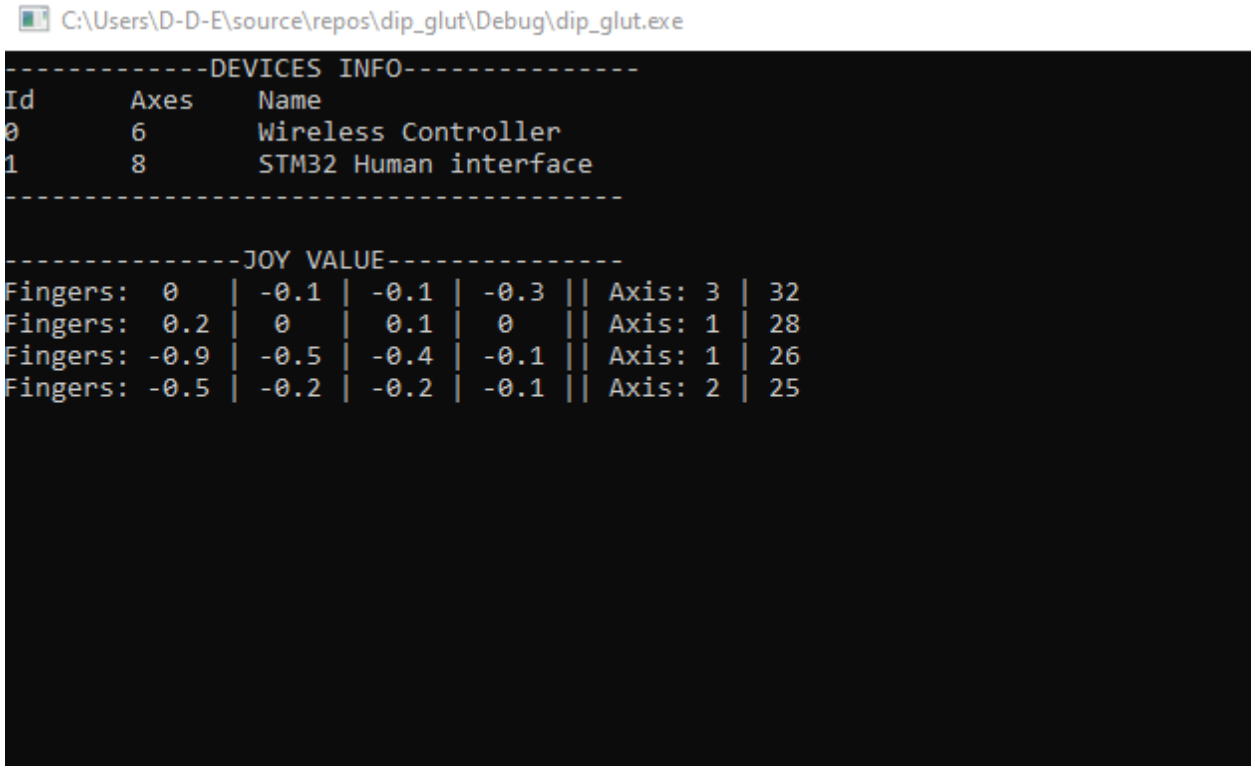

Рисунок 4.1 – Інтерфейс додатку

Рисунок 4.2 – Візуалізація додатку

4.2 Аналіз роботи системи

По закінченню основних робіт по створенню моделі, необхідно

4.2.1 Аналіз механічних датчиків

Для аналізу роботи системи з механічними датчиками для трекінгу руху пальців, був використан осцилограф для знаття реального аналогового сигналу та порівняння з отриманими результатами з компьютеру. Це дає змогу розглянути роботу системи з механічними сенсорамии у цілому.

На обох зображеннях дані згибання вказівного пальця до входу в пристрій та те що потрапляє до компьютера.

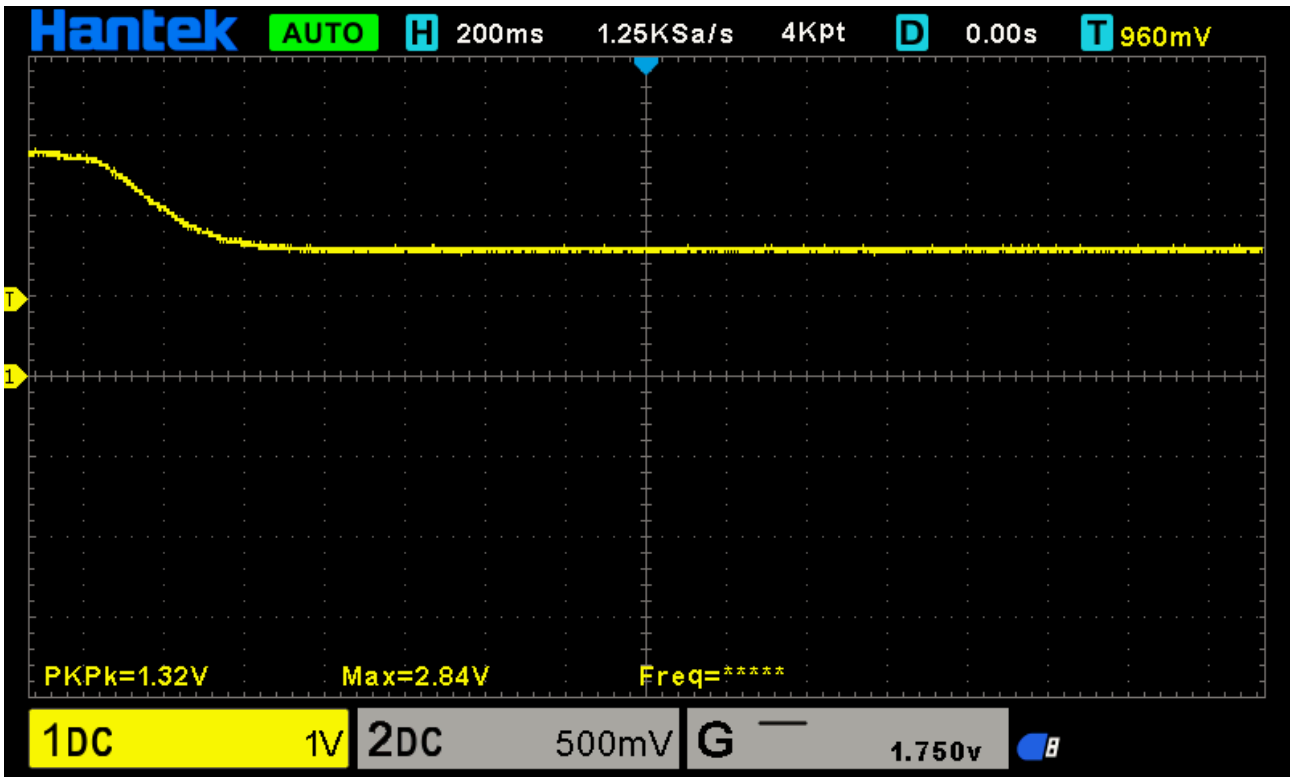

Рисунок 4.3 – Осцилограма сигналу

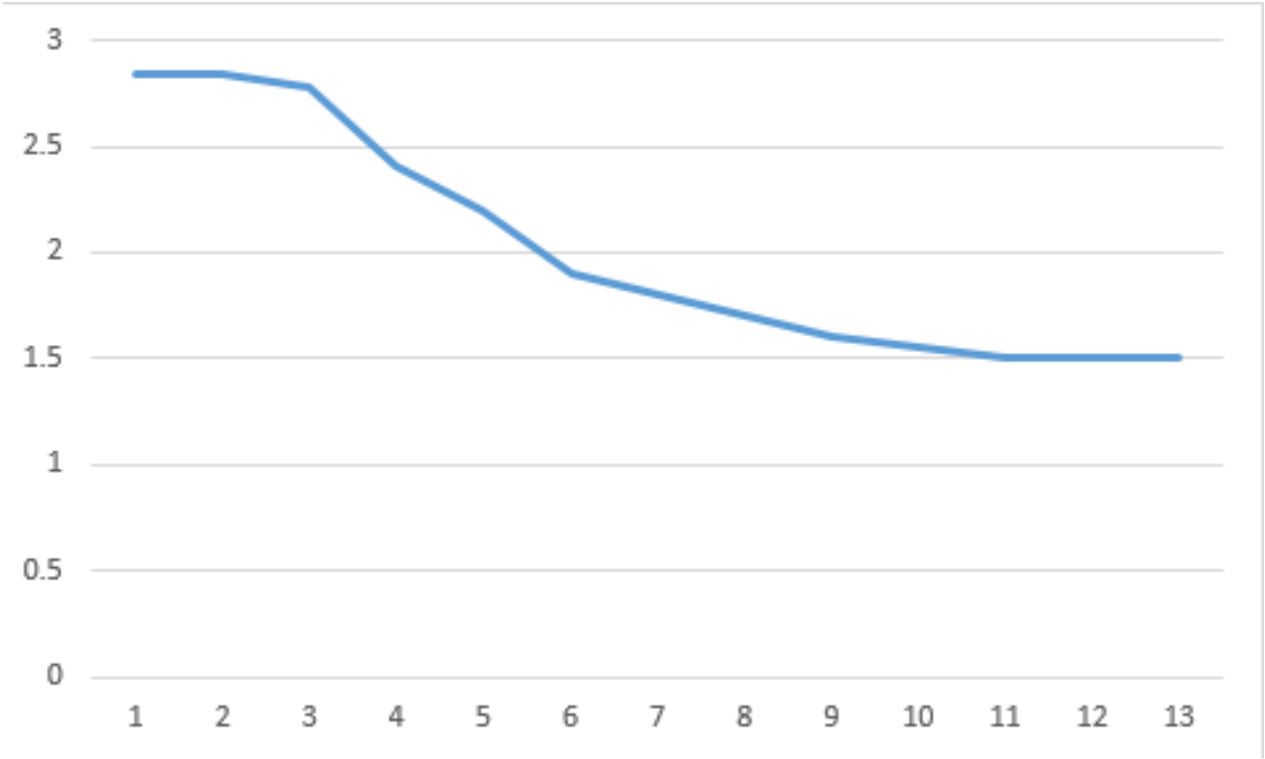

Рисунок 4.4 – Графіх даних з компьютеру

Як наглядно можна побачити отриманий сигнал є досить точним у порівнянні з тим що є насправді. Найголовніша різниця, сигнал з компьютеру не плавний та більш різкий, це можна пояснити, тим що осцилограф має швидший АЦП та дискретизує більше точок сигналу за рівний проміжок часу. Для вирішення данної проблеми можно спробувати застосувати декілька рішень:

- збільшити тактову частоту ядра та перефірії;
- застосувати другий АЦП у режимі чергування;
- застосувати зовнішній АЦП;

Але загальному вигляді це не є велика проблема у роботі з пристроєм на базовому рівні, наприклад з у відео іграх. Прискорення роботи АЦП може знадобитися лише у потребі більш точних данних наприклад у медицинській сфері.

4.2.1 Аналіз MEMS датчику

Гіроскопічний сенсор є цифровим, тому нажаль тетсування його роботи є меньш точнішою, через те що вся робота по перетворенню сиглалу знаходиться всередині самого чіпа. Для аналізу роботи системи у цілому можна створити графік умовно сирих даних гіроскопа та акселірометра і порівняти з даними після роботи фільтру Калмана. Так можна побачити роботу двох сенсорів у парі, які компенсують шуму один одного та створюють на виході чистий сигнал. З часом значення гіроскопа почнуть дрейфувати через шум і зсув у даних датчика. Але цей процес дуже повільний і протікає майже не помітно. Фільтр виявить цей дрейф і повільно надасть більшої ваги оцінці акселерометра, що дасть точнішу оцінку орієнтації об’єкта. Окрім фільтра з цим явищем можно боротися через пере калібровку датчику, але це спричинить остановку передачі даних на час обновлення.

Крім того, сумісний підхід показує хороший результат при ударах датчиків або інших позаштатних різких відхиленнь.

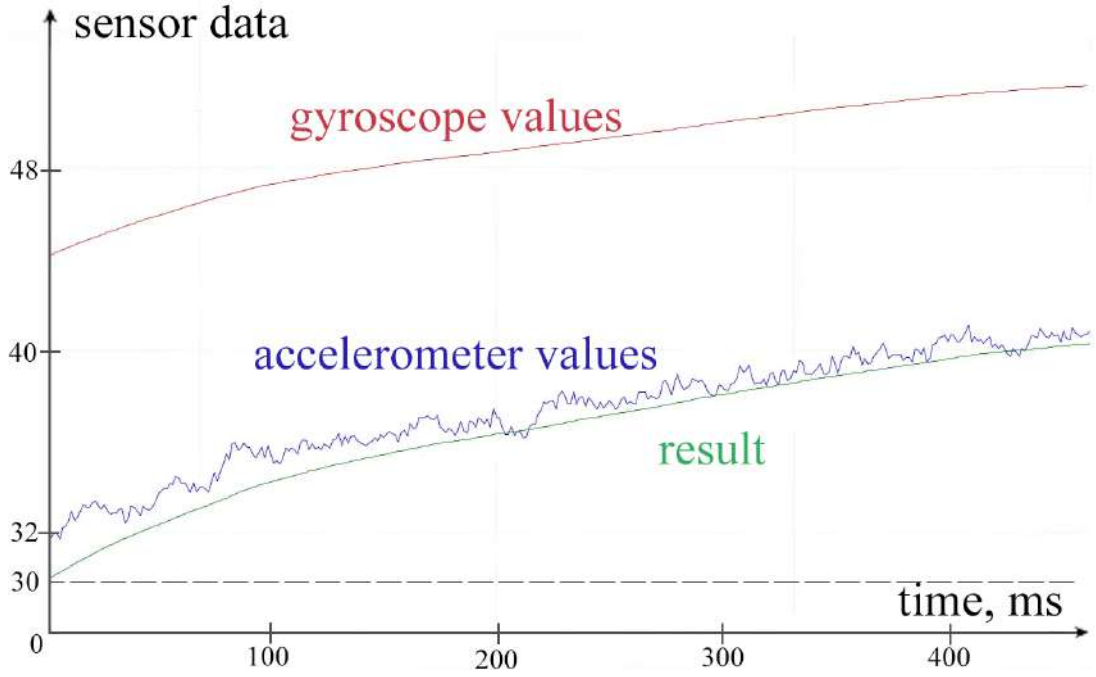

Рисунок 4.5 – Графіх даних з компьютеру

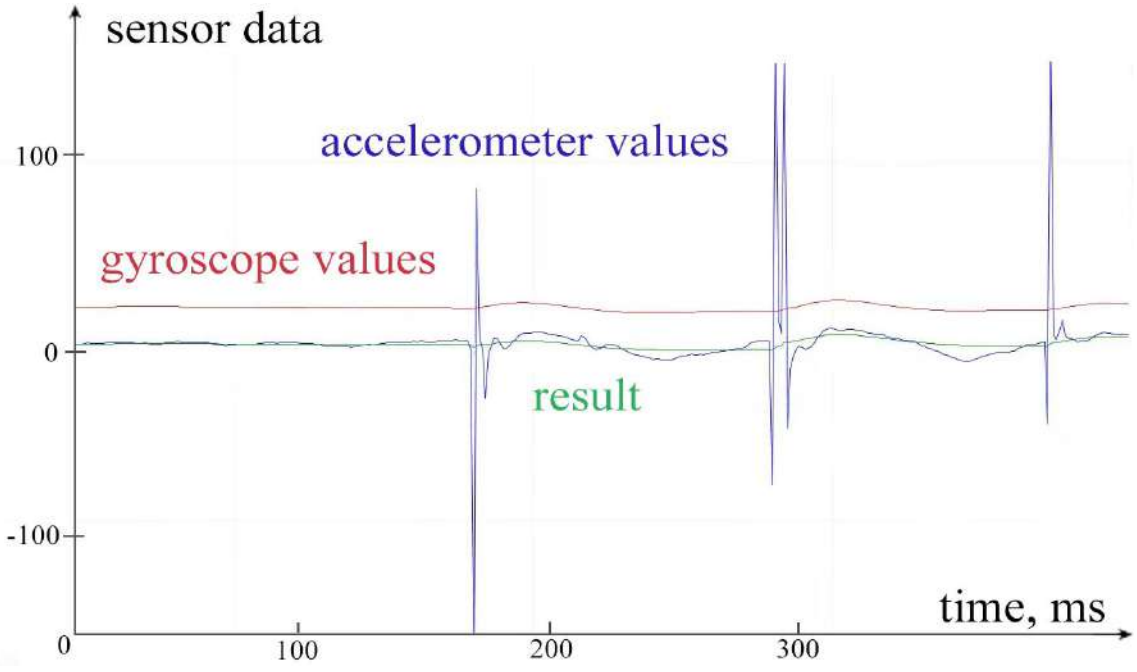

Рисунок 4.6 – Графіх даних з компьютеру

ВИСНОВКИ

Гіроскопи зазвичай використовуються в системах відстеження руху людини для вимірювання орієнтації та кутової швидкості сегментів тіла під час різних видів діяльності, наприклад аналізу ходи, моніторингу спортивних результатів і реабілітації. Вони працюють за принципом кутового моменту, що означає, що вони можуть виявляти зміни кутової швидкості навколо певної осі. У відстеженні руху людини гіроскопи зазвичай використовуються в поєднанні з іншими датчиками, такими як акселерометри та магнітометри, щоб отримати більш повну картину руху тіла. Поєднуючи вимірювання від гіроскопів з вимірюваннями від інших датчиків, можна отримати більш точне та повне уявлення про рух тіла. Однією з головних проблем у використанні гіроскопів для відстеження руху людини є необхідність усунення шуму та дрейфу вимірювань. Щоб пом’якшити ці проблеми, можна використовувати різні методи калібрування та фільтрації, включаючи оцінку зсуву, об’єднання датчиків і фільтрацію Калмана. Іншою проблемою у використанні цього методу для відстеження руху людини є необхідність забезпечити точне розміщення та вирівнювання датчиків. Гіроскопи повинні бути встановлені в певній орієнтації відносно вимірюваного сегмента тіла, і будь-яке зміщення може призвести до помилок у вимірюваннях.

Для задач обробки “сирих” даних для трекінгу фільтр Калмана є ефективним інструментом, котрий забезпечує засоби для точної оцінки стану системи на основі шумових вимірювань. Він особливо корисний для систем відстеження руху, які включають кілька датчиків, таких як гіроскопи, акселерометри та магнітометри, оскільки вони можуть ефективно поєднувати вимірювання від цих датчиків для отримання більш точного представлення руху системи. Його можна використовувати для широкого спектру застосувань у відстеженні руху, включаючи аналіз ходи, моніторинг спортивних результатів і реабілітацію. Використовуючи фільтр Калмана для оцінки положення, швидкості та прискорення системи, можна отримати уявлення про механізми

руху людини, виявити аномалії чи недоліки та розробити цілеспрямовані втручання для покращення продуктивності або усунення травм. Однак ефективність фільтру в системах відстеження руху залежить від ретельного налаштування параметрів і вибору моделі. Параметри фільтра Калмана повинні бути обрані таким чином, щоб збалансувати точність і чутливість, а використовувана модель повинна точно відображати динаміку відстежуваної системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1. Bächlin M., Tröster G. Swimming performance and technique evaluation with wearable acceleration sensors / M. Bächlin, G. Tröster // Pervasive and Mobile Computing. 2012. 8(1). P. 68-81.

- 2. Fuss F.K. Instrumentation of sports equipment: Routledge Handbook of Sports Technology and Engineering / F.K. Fuss // Routledge. 2013. P. 71-86.

- 3. Real-time swimmers' feedback based on smart infrared (SSIR) optical wireless sensor / Hagem R.M., Thiel D.V., O'Keefe S., Fickenscher T. // Electronics Letters. 2013. 49(5). P. 340-341.

- 4. Niźnikowski T., Sadowski J., Starosta W. Coordination Abilities in Physical Education, Sports and Rehabilitation. // Józef Piłsudski University of Physical Education, Warsaw. Faculty of Physical Education and Sport, 2016. 323 p.

- 5. Investigation of sensor-based quantitative model for badminton skill analysis and assessment / Shan C.Z., Sen S.L., Fai Y.C., Ming E.S.L. // Journal of TEKNOLOGI. 2015. 72(2).

- 6. Senanayake C., Senanayake S. Human assisted tools for gait analysis and intelligent gait phase detection / C. Senanayake, S. Senanayake // Innovative Technologies in Intelligent Systems and Industrial Applications, CITISIA. 2009. P. 230-235.

- 7. FreeWalker: A smart insole for longitudinal gait analysis / Wang B., Rajput K.S., Tam W.K., Tung A.K., Yang Z. // Engineering in Medicine and Biology Society (EMBC). 37th Annual International Conference of the IEEE. 2009. P. 3723-3726.

- 8. Design and implementation of an integrated performance monitoring tool for swimming to extract stroke information at real time / Chakravorti N., Le Sage T., Slawson S. E., Conway P.P., West A.A. // IEEE Transactions on Human-Machine Systems. 2013. 43(2). P. 199-213.

- 9. A mobile motion capture system based on inertial sensors and smart shoes / Jung P.G., Oh S., Lim G., Kong K. // Journal of Dynamic Systems,

Measurement, and Control. 2014. 136(1): 011002.

- 10. Azcueta J.P.V., Libatique N.C., Tangonan G.L. In situ sports performance analysis system using inertial measurement units, high-fps video camera, and the Android platform / J.P.V. Azcueta, N.C. Libatique, G.L. Tangonan // In Proceedings of the 2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control Environment and Management. 2014. P. 12-16.

- 11. Trajectory generation for myoelectrically controlled lower limb active knee exoskeleton / Kundu A.S., Mazumder O., Chattaraj R., Bhaumik S., kumar Lenka P. // Contemporary Computing (IC3), Seventh International Conference on. 2014. P. 230-235.

- 12. Borzikov V.V., Rukina N.N., Vorobyova O.V. Human Motion Video Analysis in Clinical Practice (Review) // Соврем. технол. Мед, 2015, No4. URL: cyberleninka.ru/article/n/human-motion-video-analysis-in-clinical-practice-review.

- 13. Comparing nape vs. T4 placement for a mobile Wireless Gait Analysis sensor using the Dynamic Gait Index test / Nukala B., Shibuya N., Rodriguez A., Tsay J., Nguyen T., Zupancic S., Lie D.Y. // Eighth International Conference on Mobile Computing and Ubiquitous Networking (ICMU). 2015. P. 68-69.

- 14. Chan Y.J., Huanga J.-W. Multiple-point vibration testing with microelectromechanical accelerometers and micro-controller unit // Mechatronics, Volume 44, June 2017. – P. 84-93

- 15. Керниган Б., Ритчи Д. Язык программирования С Пер. с англ., 3-е изд., испр. - СПб.: "Невский Диалект", 2001. - 352 с

- 16. Einicke, G.A. (2012). Smoothing, Filtering and Prediction: Estimating the Past, Present and Future. Rijeka, Croatia: Intech. ISBN 978-953-307-752-9.

- 17. A Dynamic Positioning System Based on Kalman Filtering and Optimal Control, J.G. Balchen, N.A. Jenssen, S. Saelid, E. Mathisen, MODELING, IDENTIFICATION AND CONTROL, 1980, VOL. 1, No.3

- 18. Буданов А.С., Егунов В.А. Использование углов Эйлера в инерциальных навигационных системах // Инженерный вестник Дона, 2021,

No7. URL: ivdon.ru/ru/magazine/archive/n7y2021/7072.

- 19 D. Dashkov O. Liashenko Motion capture with MEMS sensors // Advanced Information Systems 2023

- 20. Kalman R.E. A new approach to linear filtering and prediction problems / R.E. Kalman // Journal of basic Engineering. 1960. 82(1). P. 35-45.

- 21. Arias-Castro, Ery; Donoho, David L. (June 2009). "Does median filtering truly preserve edges better than linear filtering?". Annals of Statistics. 37 (3): 1172–2009. arXiv:math/0612422. Bibcode:2006math.....12422A. doi:10.1214/08-AOS604. MR 2509071. Zbl 1160.62086.

- 22 USB Implementers' Forum / Device Class Definition for Human Interface Devices (HID) Firmware Specification 1996-2001 URL: https://www.usb.org/sites/default/files/documents/hid1_11.pdf ДОДАТОК А