Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Зибін Дмитро Вадимович,
студент групи НАС-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Програмне забезпечення для кібернетичного костюму з можливістю взаємодії з віртуальною реальністю

Спеціальність:
121 – Інженерія програмного забезпечення

Освітня програма:
Інженерія програмного забезпечення

Керівник:
Тройніна Анастасія Сергіївна,
канд. техн. наук, доцент

Одеса – 2021

ЗМІСТ

Міністерство освіти і науки України

Державний університет «Одеська політехніка»

Навчально-науковий інститут комп'ютерних систем

Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Освітня програма: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Любченко В. В.

«___»_____20__ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Зибіна Дмитра Вадимовича, група НАС-161

1. Тема роботи: Програмне забезпечення для кібернетичного костюму з можливістю взаємодії з віртуальною реальністю

Керівник роботи: Тройніна Анастасія Сергіївна, кандидат технічних наук, доцент затверджені наказом ректора від «25» жовтня 2021р. № 374

2. Зміст роботи: вимоги до програмної системи, план виконання проекту, проектування програмної системи, розробка системи, тестування та випробування програмної системи

3. Перелік ілюстративного матеріалу: Згідно до слайдів презентації

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

5. Дата видачі завдання «___» _____ 20 ___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Вимоги до програмної системи	11.07.2021	Виконав
2	План виконання проекту	20.07.2021	Виконав
3	Проектування програмної системи	10.08.2021	Виконав
4	Розробка системи	30.09.2021	Виконав
5	Тестування та випробування програмної системи	30.10.2021	Виконав
6	Визначення властивостей програмної системи	10.11.2021	Виконав

Здобувач вищої освіти _____ Д. В. Зибін

Керівник роботи _____ А. С. Тройніна

АНОТАЦІЯ

Робота присвячена програмному забезпеченню для обробки інформації зі змінюваного числа датчиків акселерометра і гіроскопа кібернетичного костюму, що використовується для отримання інформації про переміщення об'єкта з метою створення реалістичної моделі людського тіла в віртуальному просторі.

Метою роботи є зменшення мінімального часу зчитування та обробки даних датчиків при максимально можливій точності та мінімальному числі цифрових перешкод.

Програмно-апаратний продукт, виконаний на елементній базі Arduino, що дозволяє при поточному розвитку ринку мікроелектроніки створювати недорогу систему захоплення руху (Motion Capture), яка не поступається сучасним аналогам, використовуваним в великих організаціях, індустрії кіно і відеоігор, а також взаємодіяти з віртуальною реальністю, проводити експерименти в кросплатформеному середовищі розробки Unity 3D.

Ключові слова: Arduino, Motion capture, віртуальна реальність, Unity 3D.

ABSTRACT

The work is devoted to software for processing information from a variable number of accelerometer sensors and a cybersuit gyroscope, which is used to obtain information about the movement of an object in order to create a realistic model of the human body in cyberspace.

The aim is to reduce the minimum time of reading and processing of sensor data with the highest possible accuracy and the minimum number of digital interference.

Arduino-based software and hardware product that allows the current development of the microelectronics market to create an inexpensive motion capture system (Motion Capture), which is not inferior to modern counterparts used in large organizations, the film and video game industry, as well as interact with virtual reality , to conduct experiments in a cross-platform Unity 3D development environment.

Keywords: Arduino, Motion capture, virtual reality, Unity 3D.

ВСТУП

28 років тому на великий екран вийшов комп'ютерний мультфільм "Сіндбад: Покрив туману". Це один із перших масштабних проектів, що використовують технологію Motion Capture [1]. Цей прийом використовується для запису дій акторів, а потім використовується у комп'ютерній графіці. Принцип роботи системи захоплення руху з використанням спеціального обладнання наступний: на виконавця одягається костюм із спеціальними датчиками. Коли він відтворює необхідний рух, дані, отримані з цих датчиків, вводяться в комп'ютер і перетворюються там на тривимірну модель, щоб точно відтворити всі дії виконавця та на основі цього реалізувати анімацію персонажа. Ця технологія була розроблена для індустрії розваг і підходить для управління роботизованими механізмами в таких галузях, як гірничодобувна промисловість. Проблема таких костюмів – дорожнеча та складність програмної моделі [2].

З наукової точки зору роботу можна представити наступною формулою дослідження:

- актуальність: віртуальна реальність є важливим досягненням науки та техніки, завдяки якому можливі зміни у багатьох галузях людської діяльності, зокрема у масовій свідомості;
- мета дослідження: зменшення мінімального часу зчитування та обробки даних датчиків при максимально можливій точності та мінімальному числі цифрових перешкод;
- вирішувані завдання: побудова удосконаленої архітектури кібернетичного костюму; розробка алгоритмів - зчитування даних з регістрів MPU 6050, отримання готових значень датчика по осях x, y і z, розподілу отриманих від обробного центру кібернетичного костюма даних по осях x, y і z на відповідні їх адресам кінцівки 3D моделі в Unity 3D; формування вимог до програмної системи; проектування, реалізація і тестування програмної системи;
- об'єкт дослідження: архітектура кібернетичного костюму;

- предмет дослідження: процес передачі первинних даних у кібернетичному костюмі з датчиків акселерометра, гіроскопа та вигину на 3D-модель у Unity3D;
- методи дослідження: порівнювальний аналіз, експеримент;
- наукова новизна: вперше запропоновано недорогу систему для взаємодії з віртуальною реальністю з високою точністю передачі рухів людини, створену на базі загальнодоступних елементів Arduino;
- практична значимість: реалізація ідеї дозволяє використання віртуальної реальності у різних галузях діяльності людини;
- публікації: 10-а міжнародна конференція студентів і молодих учених "Сучасні Інформаційні Технології 2020" Одеса, 14-15 травня 2020 року. Блажко О.А., Зибін Д.В. Кібернетичний костюм для взаємодії з віртуальною реальністю; 11-а міжнародна конференція студентів і молодих учених "Сучасні Інформаційні Технології 2021" Одеса, 14-15 травня 2021 року. Блажко О.А., Зибін Д.В. Кібернетичний костюм для взаємодії з віртуальною реальністю; XII Konferencji Horyzonty Nauki: Forum Prac Dyplomowych 2021, Blazhko A., Zybin D., Cybernetic Suit for Interaction with Virtual Reality; X Міжнародна науково-практична конференція. Рувінська В.М., Тройніна А.С., Зибін Д.В. Кібернетичний костюм для взаємодії з віртуальною реальністю / «Інформаційні та управляючі системи і технології (ІУСТ-ОДЕСА-2021).

Мета роботи - провести дослідження у сфері кібернетичних костюмів для роботи з віртуальною реальністю, а саме, їх архітектури та алгоритмів роботи. Завдання - отримати максимально стабільну та стійку до перешкод передачу первинних даних від акселерометра, гіроскопа та датчиків вигину у 3D-модель з мінімально можливою кількістю помилок у шинах передачі даних. Для візуалізації даних, отриманих з акселерометра, гіроскопа та датчиків вигину, розташованих на костюмі, пропонується використовувати програмне середовище Unity 3D [3]. Необхідно досягти максимально можливої точності повторення рухів оператора костюма за мінімальних фінансових витрат.

1 КРИТИЧНИЙ АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Аналіз проблем користувачів

Для аналізу проблем користувачів та варіантів засобів для вирішення цих проблем використаємо Value proposition canvas.

«Шаблон ціннісної пропозиції (англ.Value proposition canvas) – єдиний цілісний документ, який є розширенням шаблону бізнес моделі, і дозволяє детально роздивитися процес створення цінності для користувача» [4].

Шаблон ціннісної пропозиції складається з 6 частин, причому 3 частини – блок опис профілю клієнта, а інші 3 частини – блок саме пропозиції продукту.

Перший блок складається з таких частин, як:

– задачі покупця або customer jobs – задачі, які користувач має намір виконати або проблеми, які потребують вирішення;

– біль покупця або pains – проблеми, з якими стикається покупець в процесі виконання своїх задач;

– вигоди для покупця або gains – цінності, які покупець бажає отримати за допомогою послуг.

Другий блок включає в себе такі частини, як:

– продукти чи послуги або products&services – це послуги та продукти, довкола яких будується ваша ціннісна пропозиція;

– знеболюючі або painrelievers – опис того, як послуги або продукти зменшать проблеми в процесі виконання задач;

– створення вигоди або gaincreators – опис того, як створюють цінності, які бажає отримати покупець, щоб бути задоволеним.

Створимо шаблон ціннісної пропозиції для користувачів системи Motion Capture.

Customerjobs:

– ціна на існуючі зразки Motion Capture досить висока;

– системи Motion Capture складні у використанні та вимагають фахівців для

роботи;

– існуючі системи Motion Capture не піддаються вільної модернізації під необхідні конкретні користувачі.

Pains:

– такі сфери, як кіберспорт, навчання на тренажерах та реабілітаційна медицина не можуть дозволити собі систему Motion Capture через її вузькоспрямованість та дорожнечу;

– такі зразки Motion Capture як Leap Motion та Kinect дозволяють працювати з віртуальним простором лише у фронтальній проекції та з невисокою точністю.

Gains:

– система Motion Capture повинна мати можливість легкої та швидкої модернізації під потрібні користувачів;

– система Motion Capture не повинна мати високу вартість;

– система Motion Capture повинна мати можливість отримання тактильного відгуку від віртуального простору;

– система Motion Capture повинна працювати в будь-якій проекції та мати високу точність зчитування положення частин тіла користувача.

Products&services:

– програмне забезпечення для кібернетичного костюму з можливістю взаємодії з віртуальною реальністю.

Painrelievers:

– програмне забезпечення дозволяє отримати систему Motion Capture на базі загальнодоступних та дешевих елементів Arduino;

– дозволяє працювати в будь-якій проекції та положенні;

– дозволяє вносити зміни щодо потреби.

Gain creators:

– програмна система дозволяє працювати з віртуальним простором без особливих труднощів (проекція, становище, відстань);

– використання даного програмного забезпечення та кібернетичного

костюма дозволяє значно знизити витрати на обладнання;

– інтерфейс віртуального простору може бути змінений на потрібний для користувача.

1.2 Аналіз аналогів

Розглядаючи існуючі аналоги, представлені в табл. 1.1, можна відзначити високу вартість готових зразків.

Leap Motion / MS Kinect v2 - це нова технологія захоплення руху для взаємодії людини з комп'ютером. Він використовує в своїй роботі інфрачервоні камери, щоб з високою точністю відстежувати рух пальців і рук людини, а також інших об'єктів. Ціна на ці пристрої невисока. Однак є ряд недоліків. Наприклад, зйомка рухів людини може вироблятися тільки у фронтальній проекції камери. Людина не може стояти боком до пристрою. Крім того, подібні пристрої в принципі не здатні передавати людині будь-які тактильні реакції з віртуального простору (див. табл. 1.1).

Таблиця 1.1 – Порівняння існуючих рішень

Аналоги	Ціна	Простота експлуатації	Безпека для оператора костюма	Обмежені рухи тіла оператора	Передача відповіді з комп'ютера оператору	Використання Arduino
Leap Motion / MS Kinect v2	~4000 грн	Низький (Потрібна людина, яка розуміє систему)	Безпечно	Працює тільки у фронтальній проекції	Неможливо	-
Motion Capture Suit from OptiTrack	~8000 грн	Високий (Потрібна інструкція із застосування)	Опромінення сигналами ІЧ - передавача	Жодного	Відсутнє	-

Продовження таблиці 1.1

Аналоги	Ціна	Простота експлуатації	Безпека для оператора костюма	Обмежені рухи тіла оператора	Передача відповіді з комп'ютера оператору	Використання Arduino
Motion Capture Suit from NANSENSE	178000 грн	Високий (Потрібна інструкція із застосування)	Опроміненння сигналами ІЧ - передавача	Жодного	Відсутнє	-
Keywish Gesture Motion Tracking glove	1376 грн	Високий (Потрібна інструкція із застосування)	Безпечно	Лише рука оператора	Можливо	+
Запропонований проект кібернетичного костюму	~4000 грн	Високий (Потрібна інструкція із застосування)	Безпечно	Жодного	Можливо	+

Костюми захоплення руху від OptiTrack [5] і NANSENSE [6] по конструкції аналогічні запропонованого проекту, за винятком того, що всі датчики в таких костюмах використовують мережу Wi-Fi для передачі даних, що негативно позначається на людях через високий випромінювання передавачів. Крім того, такі костюми призначені не для передачі відповіді з віртуального середовища людині, а тільки для збору даних, що робить їх застосовними тільки в галузі кінематографії та ігрової індустрії. Відсутність реакції віртуального світу не дозволяє повністю зануритися в змодельовану ситуацію, орієнтуватися в штучному просторі і розвинути м'язову пам'ять у оператора скафандра. Крім того, відсутність модульної

конструкції змушує користувача купувати костюм цілком, навіть якщо потрібно тільки його частину.

Рукавичка Keywish Gesture Motion Tracking [7] була розроблена командою ентузіастів. Він зчитує дані з однієї руки людини.

Пропонований проект кібернетичного костюма має аналогічну рукавичку, вбудовану в систему, яку можна використовувати окремо, якщо користувач побажає.

Більш того, у всіх перерахованих вище аналогах відсутнє так званий конструктор сценаріїв взаємодії з віртуальним середовищем. Конструктор - це програма, в якій користувач може незалежно моделювати і налаштовувати віртуальне середовище, з якої він хотів би взаємодіяти, використовуючи кібернетичний костюм або його окремі частини, а також налаштовувати кібернетичний костюм відповідно до своїх бажань.

Якщо проаналізувати все це з фізичної точки зору, можна зробити висновок, що акустичні, магнітні, оптичні і механічні системи виявлення руху сьогодні широко поширені. У акустичної системі набір акустичних датчиків вибирає звук з джерела звуку, розташованого на актора. Щоб визначити положення кожного джерела звуку в просторі, розраховується відстань між передавачем і кількома приймачами. Очевидним недоліком даної системи є вплив джерел шуму і обмеження кількості датчиків. Магнітна система досить точна і швидка, щоб виявляти простий рух об'єкта. Принцип роботи заснований на зміні положення магніту щодо детектора. Така система дорога, але у неї є багато недоліків, пов'язаних з можливістю перекриття магнітних полів, які викликані різними магнітними структурами і викликають перешкоди. При цьому техніка, яку носять актори, дуже проста, що не сковує рухи, має спеціальну відмітку і контролюється камерою. У механічних системах датчики, розташовані на тілі людини, використовуються для визначення просторового положення в певних точках. Така система досить проста в розробці, але бездротова система дуже дорога і може бути небезпечна для здоров'я при впливі ультрависокочастотної енергії від передавача. Запропонований варіант - найдешевший і перспективний.

2 ТЕОРЕТИЧНА ЧАСТИНА СИСТЕМИ

Розглянемо більш детально систему Motion Capture на апаратній базі кібернетичного костюму для взаємодії з віртуальною реальністю з точки зору функціоналу та архітектури.

2.1 Прототипи кібернетичного костюму

Перший прототип кібернетичного костюму. Кібернетичний костюм, виготовлений на базі Arduino [8] і мережі датчиків, розташованих на тілі людини, відноситься до першого типу і дозволяє позбутися від проблеми пошуку носія костюма «в об’єктиві». інших пристроїв, наприклад, таких як Leap Motion. Датчики - це акселерометри та гіроскопи, з'єднані на одній платі MPU6050 [9], і датчики вигину на основі оптрона [10].

Наступні рядки представляють структуру костюма:

– "Голова-груди" (два акселерометри та гіроскопи, розташовані на голові та грудях);

– "Рука", яка поділяється на "Руку" (акселерометр і гіроскоп, розташовані на зап'ясті, і 5 датчиків згину, прикріплених до кінцівок пальців), і "Лікоть" (датчик згинання, акселерометр та гіроскоп, розташовані на ліктьовому суглобі);

– "Нога", яка поділяється на "Стопу" (акселерометр та гіроскоп, розташовані на стопі) та "Коліно" (акселерометр, гіроскоп та згини датчиків, розташовані у колінному суглобі);

– Центральний вузол, який розташований на задній панелі в області талії і включає в себе плату Arduino Mega з мікроконтролером ATmega2560, розгалужувач екрану та комутатор шини TCA9548A I2C.

Крім розробки самого костюма, проводилася робота з візуалізації рухів людини в кібернетичному костюмі на екрані комп’ютера. Процес візуалізації проводився в середовищі програмування Unity 3D.

Для встановлення контакту між Unity 3D і Arduino був написаний спеціальний ескіз для мікроконтролера ATmega2560, який дозволяє платі Arduino відправляти дані з усіх датчиків костюма на послідовний порт. Передача даних здійснюється за допомогою кабелю USB. Мова програмування - C#. Отримання даних в Unity 3D здійснюється класом Receiver, який відкриває підключення до COM-порту, до якого підключений кібернетичний костюм, і зчитує дані, і надійшов їх подальший розподіл на інші класи 3D-моделі людини. Сама 3D-модель повторює всі повороти справжньої людини. Кожен клас, який відповідає за ту чи іншу частину тіла, оснащений умовою перевірки відповідності отриманих даних за допомогою контрольного номера. Якщо число відповідає класу, дані, отримані за допомогою математичних виразів, перетворюються на кватерніони, систему гіперкомплексних чисел, які утворюють векторний простір чотиримірності над полем дійсних чисел. Кожна вісь декартового простору має свій кватерніон [11].

Спочатку, коли кібернетичний костюм підключили до 3D-моделі, кінцівки останнього зігнуті під неприродними кутами, він був трохи схожий на відображення реальної людини. Частини тіла 3D-моделі калібрувалися поетапно, з послідовним включенням однієї частини тіла за іншою. Щоб усунути відволікаючі фактори, такі як передача даних з датчиків частин тіла, які не використовувалися під час калібрування, їх зчитування та подальша передача даних були вимкнені на рівні програмного забезпечення. В результаті за одиницю часу лише та частина моделі, з якою проводилися калібрувальні маніпуляції, перебувала в «робочому» стані. Виконуючи множення на мінус одиницю з кутами нахилу по осях x, y або z, застосування або віднімання підібраних експериментально цілих чисел, можна було мінімізувати помилкові вигини кінцівок моделі, або повністю позбутися від них [12].

Другий прототип кібернетичного костюму. Як згадувалося раніше, спочатку костюм був досить простою архітектурою з набором датчиків і мікроконтролером, що обслуговує їх. Експерименти при виконанні прискорених фізичних вправ виявили обмеження швидкості в роботі обладнання через

неправильне зчитування з датчиків значень положення частин тіла людини в просторі, що потребувало внесення змін у апаратно-програмне забезпечення.

Перший прототип костюма складався з системи з 10 акселерометрів MPU6050 та гіроскопічних датчиків, 14 датчиків вигину на основі оптронів, цифрового комутатора шини TCA9548A I2C та мікроконтролера ATmega2560. Мікроконтролер по черзі опитував усі 10 акселерометрів та гіроскопів, одночасно перетворюючи первинні дані у значення по осях x, y та z. Датчики гнучкості не тестувалися. Під час калібрування отриманих даних у середовищі Unity 3D вдалося перенести рух кінцівок людини на 3D-модель. Однак спотворення отриманих даних були помічені: кінцівки моделі зігнулися мимовільно і через деякий час їх положення не відповідали реальному положенню датчиків на костюмі. Причина була визначена перемиканням цифрової шини I2C та її неможливістю обробити одночасно 10 датчиків акселерометрів та гіроскопів, внаслідок чого з’явились неправильні дані з датчика. Було прийнято рішення усунути перемикач шляхом обробки кожної кінцівки проміжним мікроконтролером. В результаті центр збору почав складатися з чотирьох плат Arduino Nano та однієї Arduino Mega, але позбутися від перешкоди не вдалося. У бібліотеці датчика MPU6050 виявлена проблема, а саме, неправильна математична обробка первинних значень.

Третій прототип кібернетичного костюму. Третій прототип використовував іншу бібліотеку, засновану на системі переривань, яка «зупиняла» датчик до того моменту, коли збираючий мікроконтролер не запитає у нього нові значення. Конструкція фурнітури особливих змін не зазнала. Однак прототип не встиг експериментувати: проміжні контролери не змогли правильно працювати з двома датчиками одночасно. Але під час роботи була відзначена висока точність роботи з одним датчиком. Дані були стабільними. Вирішено використовувати зв'язок контролер-датчик із системою переривань. Протягом січня-лютого 2021 року нова архітектура костюма була сформована з нуля, що можна побачити на рис. 2.1: тепер кожен датчик має власний контролер обробки, встановлений разом з ним в одному корпусі. Всього таких локальних випадків сім: голова, лікті, коліна та стопи. На малюнку ці локальні огорожі розташовані в далеких частинах

архітектури і представлені як N.Slave + MPU. Префікс Slave означає, що цей пристрій є підлеглим у протоколі зв'язку RS-485 і знаходиться під контролем ведучого, а саме N.Master. Сам майстер, а саме N.Master, спілкується з Arduino Mega, яка вже є частиною збірного центру. Датчик тулуба переміщено в збірний центр. Центр збірки костюма також зазнав значних змін: кількість плат для обробки та з'єднання збільшилася вдвічі - до двох Arduino Mega та шести Arduino Nano (одна з них пов'язана з обробкою датчика тулуба). Також експериментально було встановлено оптимальну швидкість збирального центру для зв'язку з Unity 3D - 115200 бод в секунду.

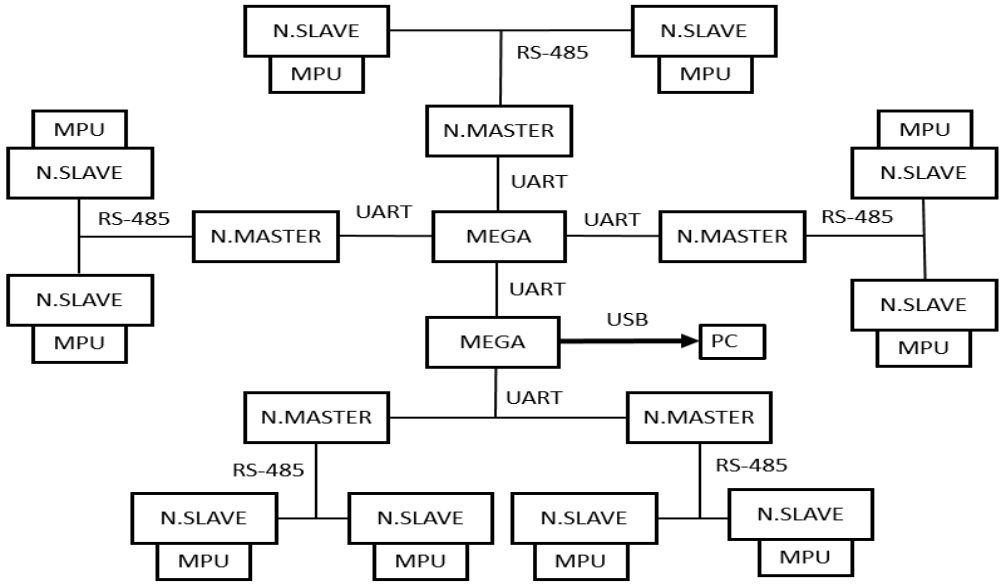

Рисунок 2.1 – Архітектура кібернетичного костюма

Результатом модернізації стало наступне:

– забезпечений легкий доступ до внутрішніх елементів корпусів;

– архітектура костюма отримала модульний дизайн та автономну роботу його частин;

– усі системи та підсистеми стали взаємозамінними та мають однаковий код, що дозволяє у разі виходу з ладу одного з контролерів швидко замінити його на новий;

– завдяки використанню високоточного коду для обробки первинних даних акселерометрів та гіроскопів вдалося позбутися від перешкод;

– було додано зовнішнє джерело живлення через збільшення споживання струму, що виключило вихід з ладу стабілізатора напруги шини USB комп’ютера;

– розподіл завдань всередині системи підвищив її стабільність.

На даний момент всередині костюму є деякі збої в передачі даних, причина яких вже встановлена (невідповідність затримок в часі в шинах RS-485) і може бути усунена [13].

2.2 Система кібернетичного костюму для передачі відповідей з віртуального середовища людині

Виходячи з початкової ідеї створення пристрою для взаємодії з віртуальним простором, кібернетичний костюм планується розділити на дві підсистеми: читання та отримання. Підсистема зчитування відповідає за сканування положення частин людського тіла в космосі та надсилання інформації до 3D-моделі в середовищі розробки Unity 3D за допомогою бібліотек для роботи с послідовним портом кібернетичного костюму та віртуальним COM-портом в Unity 3D. Приймальна підсистема повинна отримувати відповіді від різних об’єктів у імітованому віртуальному просторі в Unity 3D, включаючи вищезгадану 3D-модель, і передавати їх людині, увімкнувши датчики вібрації на кінцях пальців і нагріваючи або охолоджуючи елементи Пельтьє, які повинні розташовуватися на долонях і кінчиках пальців. Крім того, конструкція приймаючої підсистеми також передбачає наявність датчиків вібрації, розташованих на інших частинах людського тіла, тим самим формуючи певну матрицю для передачі реакцій віртуального простору [14].

2.3 Програмне забезпечення кібернетичного костюму

Як згадувалося раніше, перший прототип кібернетичного костюма складався з 10 датчиків акселерометра і гіроскопа, перемикача шини I2C і мікроконтролера

ATmega2560. Робота програмного забезпечення для цієї версії костюма складалося з таких кроків:

– створення сенсорних об'єктів MPU6050 - на цьому кроці система завантажує ту кількість сенсорних об'єктів, з якими їй згодом потрібно було працювати;

– ініціалізація об'єктів MPU6050 - запуск датчиків, відправка команд на доступ до завантаженого програмного забезпечення мікроконтролерів самих датчиків для подальшої калібрування прискорення і розташування відносно осей x, y і z;

– калібрування значень - математичний розрахунок поточного положення датчика щодо осей x, y та z та розрахунок коефіцієнтів коректної роботи з даними, отриманими від датчика;

– прямий початок підрахунку значень з датчиків.

Відповідно до структури шини I2C, дві однакові адреси пристроїв не можуть бути одночасно на лінії даних. Датчик MPU6050 може мати 68 або 69 адрес. Ця незручність обмежує використання більш ніж двох пристроїв MPU6050 на одній шині I2C одночасно. Щоб усунути цю проблему, був використаний цифровий комутатор шини I2C TCA9548A.

Мультиплексор I2C (перемикач, розширювач) TCA9548A призначений для вирішення проблеми підключення кількох I2C пристроїв з однаковими адресами або мають різну напругу логічних сигналів до одного мікроконтролера. Пристрій дозволяє повністю контролювати процес доступу ведучого до підлеглих через шину I2C.

Мультиплексор підключається через шину I2C до мікроконтролера (головного), а датчики та модулі (підпорядковані) підключаються до портів мультиплексора. Таким чином, мультиплексор виступає посередником в обміні даними між ведучим і підлеглими. Спочатку вам потрібно перейти до адреси мультиплексора (за замовчуванням 0x70) і записати номер порту, з яким ви хочете працювати, в регістр конфігурації мультиплексора. Ви можете вказати кілька

портів одночасно. Номер активного порту визначається одиницею у відповідному біті реєстру. Наприклад, 00000100 означає, що третій порт стане активним.

Далі робота виконується так само, як якщо б ведений був підключений безпосередньо до мікроконтролера. Тобто ви можете скористатися знайомими бібліотеками та звернутися до адреси ведомого пристрою (дисплей, датчик тощо).

Якщо необхідно перейти на інший пристрій, то для цього достатньо просто записати в регістр мультиплексора номер потрібного порту.

Слід зазначити, що якщо адреса будь-якого периферійного пристрою відповідає адресі мультиплексора, то необхідно змінити адресу мультиплексора за допомогою портів A1, A2, A3.

Крім того, мультиплексор може працювати як перетворювач логічного рівня незалежно від кожного каналу.

На основі принципу роботи перемикача TCA9548A мікроконтролер ATmega 2560 обробляв одночасно лише один з 10 датчиків MPU6050, а решта перебували в режимі "сну", чекаючи, коли мультиплексор отримає доступ до них для нової порції даних . Час обходу для всіх 10 датчиків зайняв близько 10 мілісекунд, що було надзвичайно високою швидкістю обробки.

Однак, як зазначалося раніше, ця версія програмного забезпечення не змогла зарекомендувати себе через низьку надійність. Під час тестування та налагодження було помічено несанкціоноване збільшення значень уздовж осі z.

Програмне забезпечення для другого прототипу костюма було доповнено чотирма мікроконтролерами посередників, а саме ATmega328 / P. ATmega328 / P є мікроконтролером сімейства AVR, як і всі інші, він має 8-розрядний процесор і дозволяє виконувати більшість інструкцій за один такт.

Кожен з чотирьох ATmega328 / P був підключений до пари датчиків MPU6050 з адресами 68 і 69 відповідно. Зв'язок здійснювався по цифровій шині I2C. Завданням ATmega328/P було перетворення отриманих даних від мікроконтролерів MPU6050 в готові значення по осях x, y і z, а потім відправка їх на центральний контролер ATmega2560. Зв'язок з ATmega2560 здійснювався через аналогову шину UART.

Універсальний асинхронний приймач-передавач (UART) - це вузол обчислювального пристрою для зв'язку з іншими цифровими пристроями, який серіалізує передані дані таким чином, щоб їх можна було передати на інший подібний пристрій по фізичній цифровій лінії. Цей метод перетворення добре стандартизований і широко використовується в інформатиці.

UART – це логічна схема, яка, з одного боку, підключена до шини обчислювального пристрою, а з іншого боку, має два або більше виводів для зовнішнього з'єднання.

Передача даних до UART відбувається один біт через регулярні інтервали. Інтервал часу визначається зазначеною швидкістю UART і вказується в швидкості передачі даних для конкретного з'єднання.

Для безпеки передачі даних на початку кожної передачі даних з ATmega328/P система додає певний персональний характер. Кожен ATmega328 / P мав унікальний символ і ніде не повторювався.

В результаті цієї схеми навантаження на центральний мікроконтролер ATmega2560 значно знизилася. Однак ця концепція не могла усунути несанкціоноване збільшення значень уздовж осі z. Як було зазначено вище, проблема була в базовій бібліотеці MPU6050.

Третій прототип використовував бібліотеку, яка використовується в основному для програмного забезпечення для легких літаків, таких як дрони. Принцип підрахунку даних у цій бібліотеці базується на системі переривань DMP.

DMP відноситься до процесора, вбудованого в MPU6050, який може безпосередньо обчислювати кватерніон і положення без необхідності додаткових математичних операцій. Використання DMP значно спрощує розробку 4-осевого коду. DMP означає цифровий процесор руху. Як випливає з назви, mpu6050 - це не просто датчик, він також містить процесор, який може самостійно виконувати алгоритм обчислення орієнтації. Наприклад, переваги використання DMP для реалізації алгоритму об'єднання датчиків у конструкції очевидні. Алгоритм обчислення положення, реалізований DMP, знімає мікроконтролер від тиску обробки алгоритму. Що потрібно зробити мікроконтролеру, це дочекатися

зовнішнього переривання, що генерується після завершення обчислення DMP, і прочитати результат обчислення положення у зовнішньому перериванні.

Однокристальний мікрокомп'ютер має багато часу на інші завдання, такі як контроль швидкості двигуна, що покращує продуктивність системи в режимі реального часу.

Кроки ініціалізації MPU6050 при використанні бібліотеки на основі переривань:

– ініціалізація інтерфейсу в I2C. Ініціалізація ліній даних SDA та SCL, підключених до MPU6050;

– скинення MPU6050, щоб усі регістри всередині MPU6050 були відновлені до значень за замовчуванням. Після скидання відновлення реєстру живлення за замовчуванням, а потім активація MPU6050;

– встановлення повного діапазону датчика гіроскопа та акселерометра;

– налаштування повного діапазону двох датчиків відповідно через регістр конфігурації гіроскопа та регістр конфігурації датчика прискорення;

– встановлення інших параметрів, вимкнення переривання, вимкнення інтерфейсу I2C та FIFO, встановлення частоти дискретизації гіроскопа та акселерометра;

– налаштування переривань, інтерфейсу I2C та FIFO;

– налаштування джерела системного годинника;

– активація датчика швидкості повороту (гіроскопа) та датчика прискорення.

Архітектура костюма з системою переривань дозволила надзвичайно точно обчислити значення датчиків по осях x, y та z. Датчик DMP управляється ATmega328 / P. Зв'язок між ATmega328 / P та посередниками контролера здійснюється через інтерфейс RS-485.

RS-485 використовується для обміну даними між кількома пристроями по двопровідній лінії зв'язку (кручена пара) у режимі напівдуплексу. Передача відбувається одночасно тільки в одному напрямку. Прийом неможливий. Для отримання даних трансивер повинен перейти в режим прийому.

З точки зору електричних характеристик і принципів зв'язку, RS-422 повністю сумісний з RS-485, але в повному дуплексному режимі. Одна вита пара завжди використовується для прийому даних, а інша – для передачі даних.

Принцип роботи з передачею даних через RS-485 схожий на протокол UART: на початку відправлення даних є адреса пристрою, з якого пристрій-посередник, або майстер, хоче прочитати дані. Крім того, приймальний пристрій, який також є підлеглим, отримує адресу і перевіряє її своєю власною. Якщо адреса правильна, підлеглий надсилає дані майстру у дробовому форматі. Через RS-485 за одиницю часу можна передати лише 8 біт інформації, а оскільки дробове значення по осях x, y і z займає в пам'яті 32 біти, система розділила дані на 4 частини і відправила їх на наступників. Після цього посередники передають готові дані на ATmega2560, де вони форматують і збирають дані з усіх датчиків і надсилають їх до Unity3D.

2.4 Математична обробка сирих даних з датчика акселерометра та гіроскопа

Калібрування гіроскопа та акселерометра є дуже важливим кроком. Ці значення гіроскопа представлені у вигляді «gyro_x_setzt», який необхідно інтегрувати, щоб отримати кут повороту відносно осі на основі кутової швидкості. Якщо «gyro_x_scallt» містить неправильну або неправильно вибрану базу даних, помилка також буде інтегрована і стане серйозною. В ідеалі, коли гіроскоп не рухається навколо осі координат, вимірюване значення має показувати нуль. На практиці досягти ідеального стану практично неможливо, тому цю помилку необхідно мінімізувати. Щоб компенсувати «дрейф», можна також за допомогою акселерометра розрахувати кут, порівняти дані з результатом роботи гіроскопа, а потім компенсувати цю помилку.

Розрахунок кута за допомогою гіроскопа mpu6050:

$$\text{gyro\_x\_scalled} = d/dt * \theta_x \quad (2.1)$$

$$\text{gyro\_y\_scalled} = d/dt * \theta_y \quad (2.2)$$

$$gyro\_x\_scalled = d/dt * \theta_z \quad (2.3)$$

Для розрахунку кута необхідно проінтегрувати змінну "gyro_x_scalled".

$$T = t_n - t_{n-1} \quad (2.4)$$

де n = {1,2,3, …} є кількістю ітерацій.

Також варто зазначити, що значення «gyro_x_scalled» залишається незмінним у кожному інтервалі циклу. Існує багато методів і методів інтеграції для компенсації цієї помилки, але ми не будемо вдаватися в подробиці.

Для досягнення дискретного інтегрування ми використовуємо метод Ейлера як один із найпопулярніших алгоритмів. Математично інтеграл Ейлера можна записати так:

$$\theta_x (t_n) = gyro\_x\_scalled * T + \theta_x (t_{n-1}) \quad (2.5)$$

Припускаємо, що початкові кути щодо осей x, y, z після калібрування дорівнюють 0, 0 і 90 градусів відповідно, так що для ітерації n=0:

$$\theta_x (t_0) = 0°, \quad (2.6)$$

$$\theta_y (t_0) = 0°, \quad (2.7)$$

$$\theta_z (t_0) = 90°, \quad (2.8)$$

Значення T (час кожної ітерації) і динаміка самого гіроскопа (швидкість зміни кута і ступінь нелінійності) істотно впливають на точність розрахунку. Чим повільніше змінюється кут і чим менший інтервал ітерації, тим точніший результат. З огляду на це, на жаль, плата Arduino дуже повільна. Кристал працює на частоті 16 МГц. Вимірювання кожні 10-20 мс стає дуже важким (оскільки процесор зайнятий не тільки обчисленням кута, а й іншими паралельними завданнями). Можна використовувати T як змінну або константу. У проекті не враховувалися

динамічні фактори, а використовувалася лише частота ітерацій з інтервалом 20 мс (0,02 с).

Оскільки калібрування gyro_x_scallt є неповним, gyro_x_scallt ніколи не буде дорівнює нулю, і тоді angle_x_gyro змінить своє значення. Щоб вирішити цю задачу, за допомогою акселерометра обчисліть кут, а отримане значення порівняйте з кутом нахилу гіроскопа. Оскільки MPU6050 розташований горизонтально, прискорення по осі z становить 1g (тобто 9.81). Ми можемо використовувати цей вектор прискорення та його проекцію на вісь y для обчислення кута між віссю x та віссю y.

Розрахований акселерометром кут знаходимо за такою формулою:

$$\theta_x = \tan^{-1}(\text{accel\_x\_scalled} / \text{sqrt}(\text{accel\_y\_scalled}^2 + \text{accel\_z\_scalled}^2)) \quad (2.9)$$

$$\theta_y = \tan^{-1}(\text{accel\_y\_scalled} / \text{sqrt}(\text{accel\_x\_scalled}^2 + \text{accel\_z\_scalled}^2)) \quad (2.10)$$

Для найкращих результатів кут нахилу гіроскопа та акселерометра поєднується з фільтром:

$$\theta_x = \text{Filter\_gain} * \theta_{x+} (1 - \text{Filter\_gain}) * \theta_x \quad (2.11)$$

Остаточно рівняння визначення кута нахилу набуває вигляду:

$$\theta_x (t_n) = \text{gyro\_x\_scalled} * T + \theta_x (t_{n-1}) \quad (2.12)$$

Основна проблема кута нахилу акселерометра: сильний шум сигналу і дуже сильна вібраційна чутливість, без цього жоден механізм не може працювати. Якщо також перемістить MPU6050 вздовж однієї з осей координат, отримане значення заважатиме обчисленню кута.

2.5 Алгоритми для вирішення задач отримання та обробки значень датчиків

На рис. 2.2 представлений алгоритм, що здійснює зчитування даних з регістрів MPU 6050. Вона складена з урахуванням особливостей протоколу для датчика і контролера.

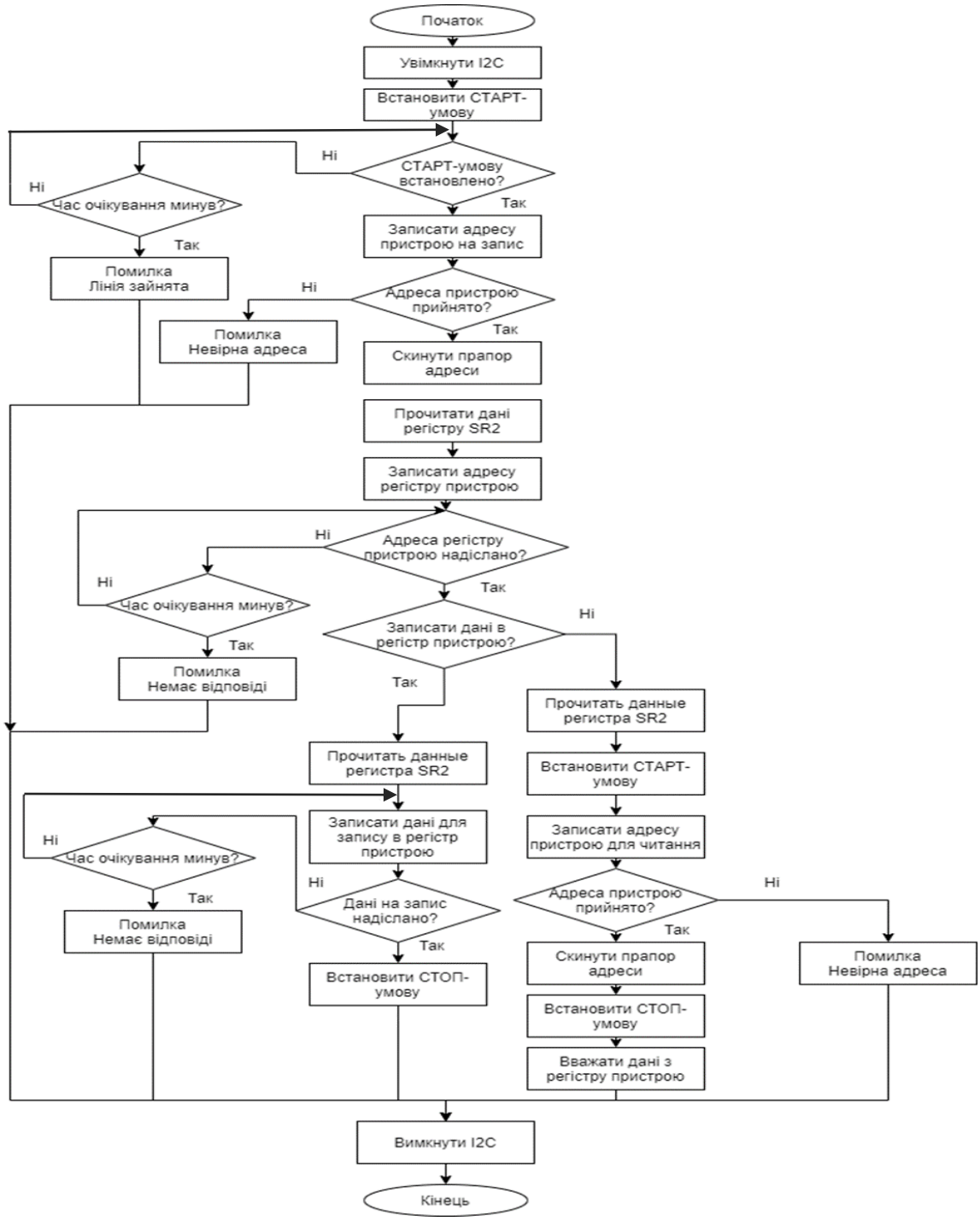

Рисунок 2.2 – Алгоритм, що здійснює зчитування даних з регістрів MPU 6050

По-перше, необхідність читання регістра SR2 там, де це зазначено в блок-схемі. Зчитування вмісту цього регістра призводить до скидання деяких бітів-прапорів в ньому. Якщо цього не зробити, модуль MPU 6050 не зможе перейти до наступної стадії роботи і станеться зависання програми. Друга особливість пов'язана із завершенням сеансу зв'язку при читанні даних з регістрів MPU 6050. Сигналом завершення обміну даних є установка стоп-умови на лінії даних.

У випадку з мікроконтролером ATMega команду на генерацію стоп-умови для нього необхідно віддати після прочитання передостаннього байта. У разі якщо приймається один байт, то після отримання підтвердження прийняття адреси від веденого пристрою. Якщо це зробити після прочитання останнього байта, мікроконтролер чекатиме прийом ще одного байта, якого не буде, і програма зависне. Після читання чи записи в регістр, MPU6050 автоматично збільшує адреса регістра на один, що дозволяє за один сеанс зв'язку зчитувати або записувати дані в кілька регістрів поспіль.

На рис. 2.3 представлена схема алгоритму отримання готових значень датчика по осях x, y і z.

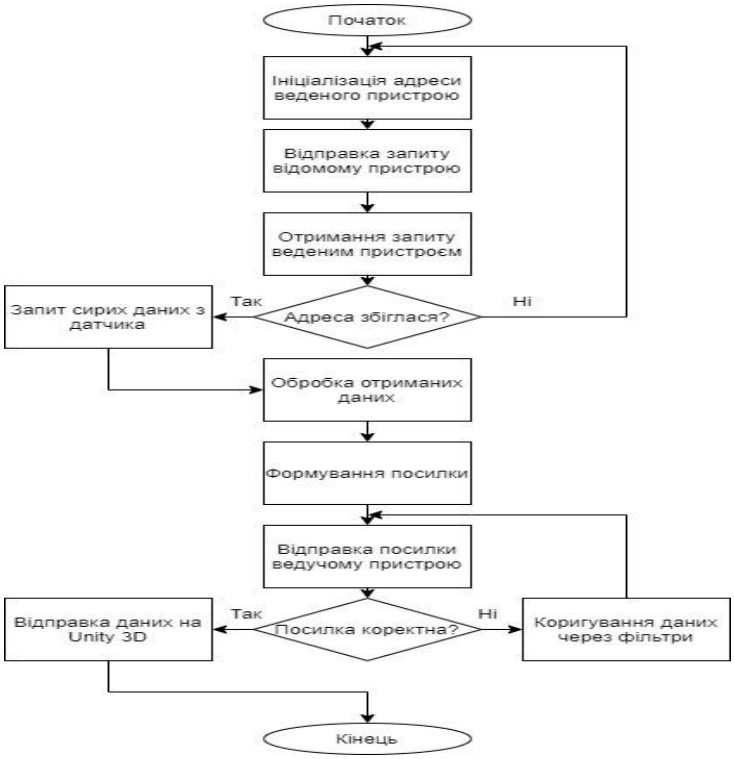

Рисунок 2.3 – Алгоритм отримання готових значень датчика по осях x, y і z

Після подачі електроживлення в кібернетичний костюм відбувається ініціалізація всіх ведених пристроїв та їх датчиків акселерометра та гіроскопа. Після ініціалізації збираючий центр, він же провідний пристрій по черзі посилає замість даних на кожен ведений пристрій. Після отримання запиту веденим пристроєм відбувається перевірка на відповідність адреси пристрою, що запитується. Якщо адреса вірна, ведений пристрій готує посилку даних по осях x, y, z і відправляє її провідному пристрою. Ведучий після отримання посилки перевіряє дані на наявність шуму та цифрових помилок. Якщо такі, дані пропускаються через фільтри і знову проходить перевірку на наявність перешкод. В іншому випадку провідний пристрій надсилає дані в Unity 3D.

На рис. 2.4 представлена схема алгоритму розподілу отриманих від обробного центру кібернетичного костюма даних по осях x, y і z на відповідні їх адресами кінцівки 3D моделі в Unity 3D.

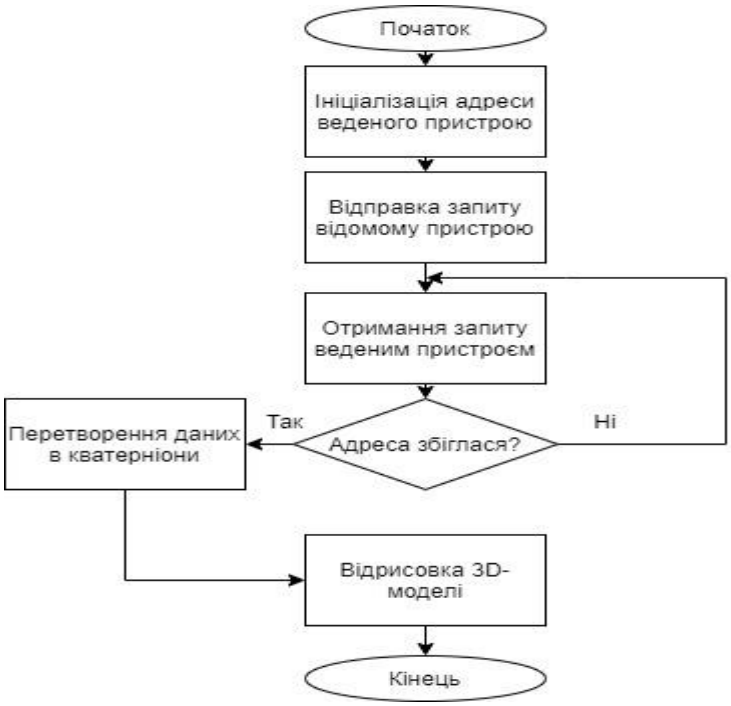

Рисунок 2.4 – Алгоритм розподілу отриманих від обробного центру кібернетичного костюма даних по осях x, y і z

Після запуску програми в Unity 3D відбувається ініціалізація та відкриття COM-порту та відправка запиту на пачку готових даних з кібернетичного костюма.

Після отримання пачки даних клас розподільник відправляє її на класи, що відповідають за кінцівки 3D-моделі. Кожен клас перевіряє адресу пачки даних зі своїм. Якщо адреса кінцівки вірна, відбувається конвертація даних по осях x, y, z в кватерніони і малювання нового положення кінцівки 3D-моделі. В іншому випадку клас ігнорує поточну пачку і чекає наступну.

2.6 Огляд апаратних ресурсів

Датчик MPU-6050 на рис. 2.5, який використовується в кібернетичному костюмі - це перший у світі інтегрований 6-осьовий пристрій відстеження руху. Він об'єднує 3-осьовий гіроскоп, 3-осьовий акселерометр та цифровий процесор руху (DMP) у невеликій упаковці 4x4x0,9 мм. Він використовує спеціальну шину датчика I2C для безпосереднього прийому вхідних даних від зовнішнього 3-осьового компаса, щоб забезпечити повний 9-осьовий вихід MotionFusion.

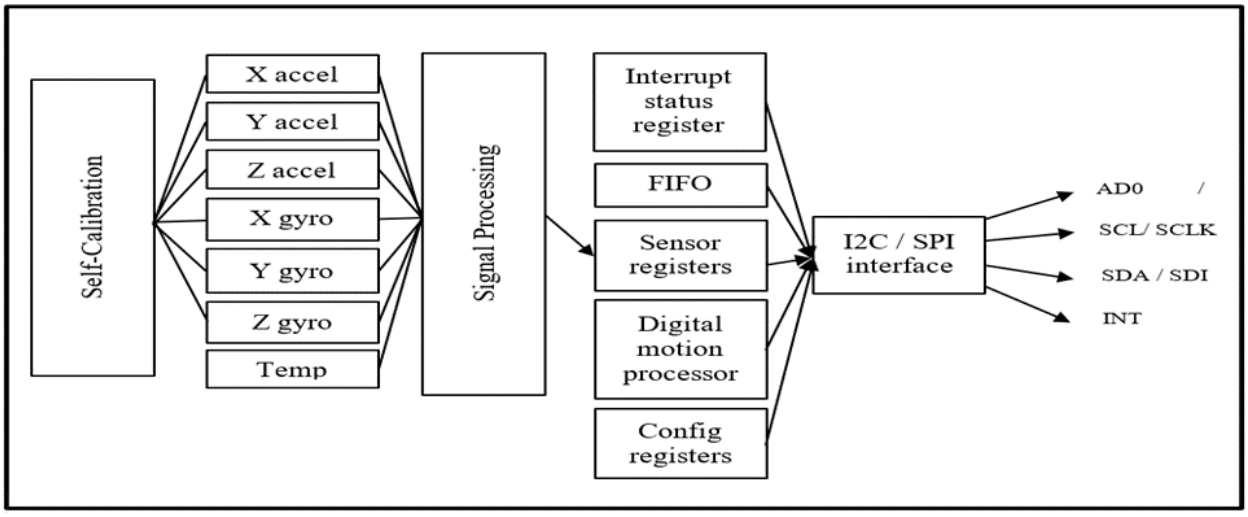

Рисунок 2.5 – Конструкція MPU-6050

MPU-6050 дозволяє виробникам усунути дорогий і складний вибір, інтеграцію та інтеграцію на рівні системи дискретних пристроїв та забезпечити найкращий рух для споживачів. MPU-6050 також призначений для неінерціальних цифрових датчиків, таких як датчик тиску на його допоміжному порту I2C.

MPU-6050 має три 16-розрядних аналого-цифрових перетворювачів (АЦП) для виведення цифрового гіроскопа та три 16-розрядних АЦП для виведення цифрового акселерометра. Вбудований 1024-байтовий FIFO дозволяє системі зчитувати дані датчика серією і переходити в режим сну, коли MPU збирає більше даних, тим самим допомагаючи зменшити споживання енергії. MPU-6050 має усі необхідні вбудовані мікросхеми обробки та датчики, необхідні для підтримки багатьох варіантів використання на основі руху, унікально підтримує використання малопотужних програм MotionInterface у портативних програмах та зменшує вимоги до обробки процесора. Завдяки вбудованому виходу MotionFusion DMP усуває MPU-6050 від інтенсивних обчислювальних вимог MotionProcessing, зводячи до мінімуму необхідність частого опитування вихідних даних датчика руху.

Для забезпечення гнучкості в живленні, MPU-6050 працює в діапазоні напруги живлення VDD від 2,375 до 3,46 В. Крім того, MPU-6050 має опорний контакт VLOGIC, який визначає логічний рівень інтерфейсу I2C.

Особливості гіроскопа, використаного в кібернетичному костюмі.

Трьохосьовий MEMS гіроскоп в MPU-6050, що використовується в кібернетичному костюмі, має широкий спектр функцій [15]:

– датчики кутової швидкості по осі X, осі Y та Z (гіроскопи) з програмованими користувачем діапазонами ± 250, ± 500, ± 1000 та ± 2000 °/с;

– зовнішній сигнал синхронізації, підключений до штиря FSYNC, підтримує синхронізацію зображення, відео та GPS;

– вбудований 16-розрядний АЦП підтримує одночасне сканування гіроскопів;

– покращене зміщення чутливості та стабільність температури зменшують потребу у калібруванні користувача;

– покращена продуктивність низькочастотного шуму;

– цифровий програмований фільтр низьких частот;

– робочий струм гіроскопа: 3,6 мА;

– струм у режимі очікування: 5 мА;

– коефіцієнт чутливості, відкалібрований на заводі;
– самоперевірка користувача.

Особливості акселерометра, використаного в кібернетичному костюмі.

Тривісний акселерометр MEMS в MPU-6050, що використовується в кібернетичному костюмі, забезпечує кілька функцій [15]:

– тривісний акселерометр з цифровим виходом та програмованими повномасштабними значеннями ± 2 г, ± 4 г, ± 8 г та ± 16 г;
– вбудований 16-розрядний АЦП дає змогу проводити вибірку акселерометра одночасно без необхідності використання зовнішнього мультиплексора;
– нормальний робочий струм акселерометра: 500 мА;
– акселерометр малої потужності: 10 мА при 1,25 Гц, 60 мА при 20 Гц, 110 мкА при 40 Гц;
– виявлення напрямку та сигналізація;
– розпізнавання дотиків;
– програмоване користувачем переривання;
– самоперевірка користувача MPU-6050.

3 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

3.1 Формування функціональних вимог

Функціональні вимоги регламентують функціонування або поведінку системи. Функціональні вимоги відповідають на запитання «що має робити система» в тих або інших ситуаціях. Функціональні вимоги встановлюють цілі, задачі та сервіси, які надаються системою замовнику, і визначають роботу розробника [16].

Функціональні вимоги записуються у вигляді варіантів використання – популярний і досить продуктивний спосіб надання вимог.

За основу для діаграми варіантів використання візьмемо критерії, які важливі для сервісної компанії замовника.

Програмне забезпечення кібернетичного костюма для взаємодії з віртуальною реальністю має забезпечувати максимально якісну математичну обробку отриманих даних від датчиків акселерометра і гіроскопа, швидке і перешкодостійке управління 3D-моделлю в віртуальному просторі і обмін інформацією з оператором кібернетичного костюма.

Крім того, програмне забезпечення повинно дозволяти швидко і при мінімальній кількості витрат вносити в себе зміни та бути кросплатформенних, а саме має підтримувати зручне для користувача взаємодія з будь-якою операційною системою.

На рис. 3.1 наведені найбільш популярні операційні системи, виходячи зі статистики компанії Net Applications за поширеністю різних операційних систем в світі за 2020 рік [17].

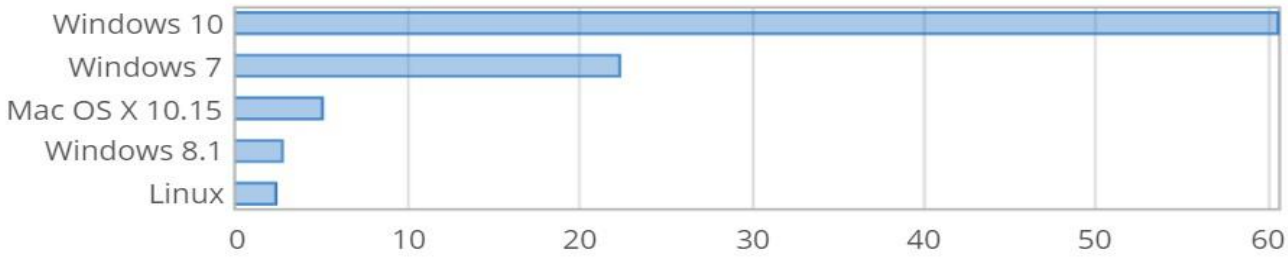

Рисунок 3.1 – Популярність операційних систем у відсотках за 2020 рік

Проаналізувавши наведену діаграму популярності операційних систем за 2020 рік (на рис. 3.1), можна зробити висновок, що найбільш поширеними є такі операційні системи як Windows 10 і Windows 7. Але виходячи з того, що Windows 10 користується найбільшим попитом, при розробці слід в першу чергу орієнтуватися на неї.

У представленому програмному забезпеченні для кібернетичного костюма повинна бути можливість обробки змінюваної кількості датчиків акселерометра, гіроскопа і вигину, стійкість передачі даних усередині системи, висока швидкість обміну інформацією між програмним забезпеченням костюма і Unity 3D, наявність математичної моделі для перетворення отриманих даних від датчиків в значення положення частини тіла оператора в тривимірній системі координат по осях x, y і z, а також можливість перетворення отриманих координат в кватерніони для частин тіла 3D-моделі.

Детально функціональні вимоги до системи наведені в табл. 3.1.

Таблиця 3.1 – Функціональні вимоги до системи

Функції	M	S	C	W	Реалізовано
Можливість обробки змінюваної кількості датчиків акселерометра, гіроскопа і вигину	+				+
Перешкодостійка передача даних усередині системи	+				+
Висока швидкість обміну інформацією між програмним забезпеченням костюма і Unity 3D	+				+
Наявність математичної моделі для перетворення отриманих даних від датчиків	+				+
Можливість перетворення отриманих координат в кватерніони	+				+
Підтримка відповідної реакції від віртуального простору				+	-

Діаграма варіантів використання системи представлена на рис. 3.2.

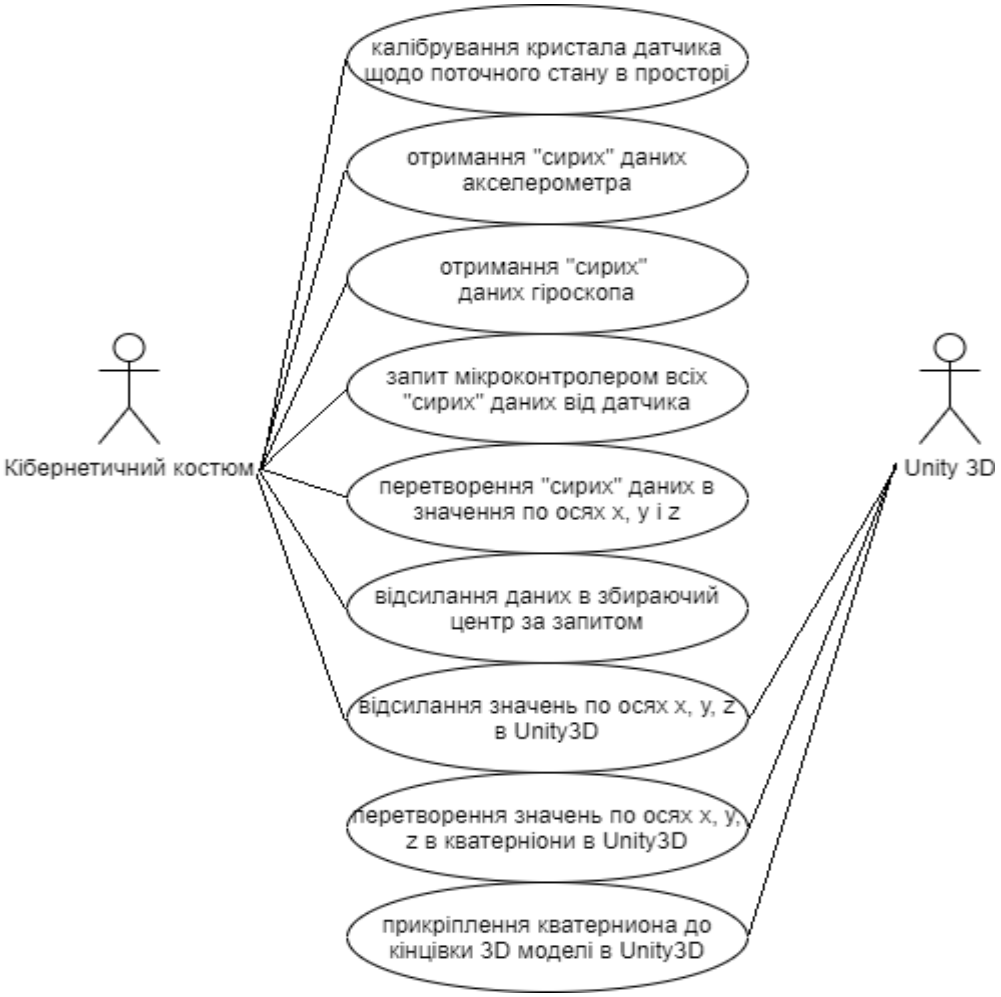

Рисунок 3.2 – Діаграма варіантів використання настільного застосування

Сценарії варіантів використання.

1. Назва прецеденту: калібрування кристала датчика щодо поточного стану в просторі.

Актори: Кібернетичний костюм.

Ініціатор: система.

Мета: відкалібрувати кристал датчика щодо поточного стану в просторі.

Передумова: наявність підключення до електромережі.

Основний успішний сценарій:

1. Кібернетичний костюм ініціалізує програмне забезпечення датчику;
2. Кібернетичний костюм отримує поточне положення датчика в просторі;

3. Кібернетичний костюм генерує змінні для коригування даних.

2. Назва прецеденту: отримання "сирих" даних акселерометра.

Актори: Кібернетичний костюм.

Ініціатор: система.

Мета: отримати "сирі" дані акселерометра.

Передумова: наявність підключення до електромережі, успішне завершення ініціалізації датчика.

Основний успішний сценарій:

1. Кібернетичний костюм очікує початок обробки датчика;
2. Кібернетичний костюм запитує в регістрах кристала датчику параметри щодо прискорення в просторі;
3. Кібернетичний костюм отримує значення прискорення по запиту.

3. Назва прецеденту: отримання "сирих" даних гіроскопа.

Актори: Кібернетичний костюм.

Ініціатор: система.

Мета: отримати "сирі" дані гіроскопа.

Передумова: наявність підключення до електромережі, успішне завершення ініціалізації датчика.

Основний успішний сценарій:

1. Кібернетичний костюм очікує початок обробки датчика;
2. Кібернетичний костюм запитує в регістрах кристала датчику параметри щодо гіроскопічного положення в просторі;
3. Кібернетичний костюм отримує значення гіроскопічного положення по запиту.

4. Назва прецеденту: запит мікроконтролером усіх "сирих" даних від датчика.

Актори: Кібернетичний костюм.

Ініціатор: система.

Мета: отримати усіх "сирі" даних від датчика.

Передумова: наявність підключення до електромережі, успішне завершення ініціалізації датчика.

Основний успішний сценарій:

1. Кібернетичний костюм очікує початок нової ітерації запиту усіх даних;
2. Кібернетичний костюм запитує в регістрах кристала датчику готові параметри щодо гіроскопічного положення та прискорення в просторі;
3. Кібернетичний костюм отримує значення гіроскопічного положення та прискорення по запиту.

5. Назва прецеденту: перетворення "сирих" даних в значення по осях x, y і z.

Актори: Кібернетичний костюм.

Ініціатор: система.

Мета: перетворити "сирі" дані в значення по осях x, y і z.

Передумова: наявність підключення до електромережі, успішне завершення ініціалізації датчика, успішне отримання усіх даних з датчика.

Основний успішний сценарій:

1. Кібернетичний костюм очікує початок нової ітерації обробки даних;
2. Кібернетичний костюм здійснює передачу "сирих" даних до алгоритму математичної обробки;
3. Кібернетичний костюм отримує значення по осях x, y і z та очікує запиту з збираючого центру.

6. Назва прецеденту: відсилання даних в збираючий центр за запитом.

Актори: Кібернетичний костюм.

Ініціатор: система.

Мета: відіслати дані в збираючий центр за запитом.

Передумова: наявність підключення до електромережі, успішне завершення ініціалізації датчика, успішне отримання усіх даних з датчика.

Основний успішний сценарій:

1. Кібернетичний костюм очікує запит з збираючого центру;
2. Кібернетичний костюм отримує запит;
3. Кібернетичний костюм перевіряє коректність запиту за адресом пристрою, отриманого із запиту;
4. Кібернетичний костюм відсилає готові значення по осях x, y і z в збираючий центр.

Альтернативний сценарій:

3.1. адреса пристрою невірна. Кібернетичний костюм переходить до пункту 1.

7. Назва прецеденту: відсилання значень по осях x, y, z в Unity 3D.

Актори: кібернетичний костюм, Unity 3D.

Ініціатор: система.

Мета: відіслати значення по осях x, y, z в Unity 3D.

Передумова: наявність підключення до електромережі, успішне завершення ініціалізації датчика, успішне отримання усіх даних з датчика.

Основний успішний сценарій:

1. Кібернетичний костюм очікує запит з Unity 3D;
2. Кібернетичний костюм отримує запит;
3. Кібернетичний костюм перевіряє коректність запиту за адресом пристрою, отриманого із запиту;
4. Кібернетичний костюм відсилає готові значення по осях x, y і z в Unity 3D.

Альтернативний сценарій:

3.1. адреса пристрою невірна. Кібернетичний костюм переходить до пункту 1.

8. Назва прецеденту: перетворення значень по осях x, y, z в кватерніони в Unity 3D.

Актори: Unity 3D.

Ініціатор: система.

Мета: перетворити значення по осях x, y, z в кватерніони в Unity 3D.

Передумова: наявність підключення до електромережі, успішне завершення ініціалізації датчика, успішне отримання усіх даних з збираючого центру.

Основний успішний сценарій:

1. Unity 3D очікує дані з збираючого центру;
2. Unity 3D отримує дані з збираючого центру;
3. Unity 3D перевіряє адресу кінцівки, для якої належать отримані із запитом дані;
4. Unity 3D здійснює передачу "сирих" даних до алгоритму математичної обробки.

Альтернативний сценарій:

3.1. адреса кінцівки невірна. Unity 3D переходить до пункту 1.

9. Назва прецеденту: прикріплення кватерніона до кінцівки 3D моделі в Unity 3D.
Актори: Unity 3D.
Ініціатор: система.
Мета: прикріпити кватерніон до кінцівки 3D моделі в Unity 3D.
Передумова: наявність підключення до електромережі, успішне завершення ініціалізації датчика, успішне отримання усіх даних з збираючого центру.
Основний успішний сценарій:

1. Unity 3D очікує дані з алгоритму математичної обробки;
2. Unity 3D отримує дані з алгоритму математичної обробки;
3. Unity 3D перевіряє коректність даних;
4. Unity 3D змінює положення кінцівки 3D моделі щодо нових значень по осях x, y, z.

Альтернативний сценарій:

3.1. дані кінцівки некоректні. Unity 3D переходить до пункту 1.

Тепер встановимо пріоритети для кожного прецеденту, використовуючи принцип MoSCow (див. табл. 3.2).

Таблиця 3.2 – Пріоритети функцій програмної системи

Функція системи	Пріоритет
Калібрування кристала датчика щодо поточного стану в просторі	M
Отримання "сирих" даних акселерометра	M
Отримання "сирих" даних гіроскопа	M
Запит мікроконтролером всіх "сирих" даних від датчика	M
Перетворення "сирих" даних в значення по осях x, y і z	M
Відсилання даних в збирає центр за запитом	M
Відсилання значень по осях x, y, z в Unity3D	M
Перетворення значень по осях x, y, z в кватерніони в Unity3D	M
Прикріплення кватерниона до кінцівки 3D моделі в Unity3D	M

Як видно з табл. 3.2, різні функції мають різні пріоритети. Пріоритет ‘M’ (must) – функція має найвищий пріоритет і повинна бути першочергово реалізована. Пріоритет ‘S’ (should) – важлива функція, але не з найвищою пріоритетністю, не має вирішального значення, але все одно є обов'язковою до виконання. Пріоритет ‘C’ (could) – бажана функція. Пріоритет ‘W’ (would) – найменш критична вимога.

3.2 Формування нефункціональних вимог

Нефункціональні вимоги регламентують внутрішні та зовнішні умови або атрибути функціонування системи. Виділяють такі основні групи нефункціональних вимог:

– зовнішні інтерфейси (найбільш важливим є інтерфейс користувача);

– атрибути якості (застосовність, надійність, продуктивність, експлуатаційна придатність);

– обмеження.

Зовнішні інтерфейси.

Програмна система має мати стриманий і простий інтерфейс користувача для настільного застосування. Система повинна підтримуватися в найбільш популярних операційних системах.

Атрибути якості.

– при створені програми повинен використовуватися інструментарій Arduino IDE, Logic 2.3.30 та Visual Studio 2015;

– система повинна працювати з Unity 3D в будь-якій операційній системі, яку підтримує Unity 3D.

Також за стандартом ISO 9126, із зазначеними у ньому характеристиками та їх атрибутами, сформульовані такі нефункціональні вимоги до програмної системи (див. табл. 3.3).

Таблиця 3.3 – Атрибути характеристик

Характеристика по ISO 9126	Властивість програмного забезпечення	Сценарії атрибутів якості
Функціональність	Здатність взаємодіяти	Система взаємодіє з Unity3D кожен раз, коли це потрібно користувачу
	Здатність взаємодіяти	Система успішно взаємодіє с Unity3D в 95% випадків

Продовження таблиці 3.3

Характеристика по ISO 9126	Властивість програмного забезпечення	Сценарії атрибутів якості
Функціональність	Безпека	Система здатна протистояти в 99% випадків потенційних загроз
	Безпека	Програмне забезпечення кібернетичного костюму не може бути змінено звичайними користувачами
Ефективність	Часові характеристики	Час реакції системи на дії користувача не довше 0.1 сек
		У разі нормального навантаження система обробляє не менше 10 кінцівок оператора костюму в сек
	Використання ресурсів	Додаток займає не більше 1 Мб в пам’яті костюма та не більше 1 Гб на пристрої користувача
		Додаток не споживає більше 10% роботи процесора і 20% оперативної пам'яті пристрою
Надійність	Відмовостійкість	Система продовжить роботу в 90% випадків після внутрішнього збою протягом 20 сек

Продовження таблиці 3.3

Характеристика по ISO 9126	Властивість програмного забезпечення	Сценарії атрибутів якості
Надійність	Відмовостійкость	Система здатна усунути 95% помилок усередині системи
	Поновлюваність після порушення в роботі	Система збереже поточний стан в 99% ситуацій і відновить його при екстреному відключенні протягом 1 хвилини
Переносимість	Адаптивність	Система дозволяє працювати в Unity 3D на будь-якій операційній системі, яку підтримує Unity 3D

Обмеження

В даний період не розробляється система відповідного відгуку від віртуального простору, але планується її створення на базі елементів Пельт'е і датчиків вібрації.

4 ПРОЕКТУВАННЯ СИСТЕМИ

4.1 Структури даних

Дані, що представлені в табл. 4.1, алгоритм використовує для відтворення положення тіла оператору костюму, знаходяться в 14-байтовому регістрі від 0x3B до 0x48.

Таблиця 4.1 – Первісні дані мікроконтролера датчика MPU6050

Назва	Тип	Точність
X_accel	float	два знаки після коми
Y_accel	float	два знаки після коми
Z_accel	float	два знаки після коми
X_gyro	float	два знаки після коми
Y_gyro	float	два знаки після коми
Z_gyro	float	два знаки після коми

Ці дані будуть оновлюватися динамічно, а частота оновлення може становити до 1000 Гц. Адреса відповідного регістру і назва даних вказані нижче. Кожен розмір даних становить 2 байта.

– 0x3B, компонент осі X акселерометра ACC_X;
– 0x3D, компонент осі Y акселерометра ACC_Y;
– 0x3F, компонент осі Z акселерометра ACC_Z;
– 0x41, поточна температура TEMP;
– 0x43, кутова швидкість обертання навколо осі X GYR_X;
– 0x45, кутова швидкість обертання навколо осі Y GYR_Y;
– 0x47, кутова швидкість обертання навколо осі Z GYR_Z.

Тривісні компоненти ACC_X, ACC_Y і ACC_Z акселерометра є 16-розрядними цілими числами зі знаком, відповідно, представляють прискорення пристрою по трьох осях. Коли береться від'ємне значення, прискорення відбувається уздовж осі координат в негативному напрямку;

Три компонента прискорення виражені в одиницях, кратних прискоренню сили тяжіння g, та діапазон прискорення, який може бути виражений, тобто збільшення може бути встановлено рівномірно, з 4 вибираними значеннями збільшення: 2g, 4g, 8g, 16g. Взявши ACC_X як приклад, якщо збільшення встановлено на 2g, це означає, що коли ACC_X приймає мінімальне значення -32768, поточне прискорення в 2 рази перевищує прискорення сили тяжіння уздовж позитивного напрямку осі X, якщо воно встановлено на 4g, воно приймає значення -32768. Це вказує на прискорення сили тяжіння в 4 рази уздовж позитивного напрямку осі X і так далі. Чим менше збільшення, тим вище точність та чим більше збільшення, тим більше діапазон, який слід встановити відповідно до конкретного застосуванням.

Компоненти кутової швидкості GYR_X, GYR_Y і GYR_Z, які обертаються навколо трьох координатних осей X, Y і Z, є 16-розрядними цілими числами зі знаком. Якщо дивитися від початку координат до напрямку осі обертання, обертання за годинниковою стрілкою відбувається, коли приймається позитивне значення, а обертання проти годинникової стрілки - коли приймається негативне значення.

Три складові кутової швидкості виражені в одиницях «градуси / секунди», і діапазон кутової швидкості, який можна виразити, тобто збільшення можна задати рівномірно, має 4 обираних збільшення: 250 градусів / секунду, 500 градусів / секунду, 1000 градусів / секунду, 2000. градуси / секунда Беручи GYR_X як приклад, якщо збільшення встановлено на 250 градусів в секунду, це означає, що коли GYR приймає позитивне максимальне значення 32768, поточна кутова швидкість складає 250 градусів в секунду за годинниковою стрілкою, якщо вона встановлена на 500 градусів в секунду, для індикації поточного значення потрібно 32768 Кутова швидкість складає 500 градусів в секунду за годинниковою стрілкою. Очевидно, що чим менше збільшення, тим вище точність і чим більше збільшення, тим більше діапазон.

Далі початкові дані надходять за запитом у мікроконтролер Arduino Nano для їх перетворення в готові значення по осі x, y і z. Після обробки вони мають вигляд, представлений у табл. 4.2.

Таблиця 4.2 – Готові дані від MPU6050 в збираючьому центрі

Назва	Тип	Точність
accel_N_x	float	два знаки після коми
accel_N_y	float	два знаки після коми
accel_N_z	float	два знаки після коми

Після обробки, мікроконтролер Arduino Nano чекає запиту від збираючого центру кібернетичного костюма і у разі його отримання відсилає їх. Збираючий центр у свою чергу є сполучною ланкою між датчиками костюма та Unity 3D. Після отримання збираючим центром готових даних від датчиків, відбувається їх відправлення в Unity 3D для малювання на їх основі положення кінцівки 3D-моделі. Дані, отримані та перероблені в Unity 3D змін не зазнають і залишаються у тому ж форматі.

4.2 MotionProcessing

Обробка руху – це процес розпізнавання, вимірювання, синтезу, аналізу та оцифрування руху об’єктів у тривимірному просторі. Технологія MotionProcessing поєднує інерційні одиниці з цифровою обробкою та програмами на основі руху, що дозволяє споживчим пристроям легко інтегрувати рух, відповідаючи ключовим вимогам щодо вартості, розміру, надійності та терміну служби акумулятора [18]:

– внутрішній механізм цифрової обробки руху (DMP) підтримує 3D -обробку руху та алгоритми розпізнавання жестів;

– MPU-60X0 збирає дані гіроскопа та акселерометра та синхронізує вибірку даних із визначеною користувачем швидкістю. Повний набір даних MPU-60X0 включає в себе дані 3-осьового гіроскопа, 3-осьового акселерометра та дані про

температуру. Розрахований вихід MPU для системного процесора може також включати дані курсу з цифрового 3-осьового магнітометра стороннього виробника;

– FIFO буферує весь набір даних і зменшує вимоги до тактової частоти системного процесора, дозволяючи процесору пакетів зчитувати дані FIFO. Після зчитування пакетних даних FIFO системний процесор може заощадити енергію, перейшовши в режим очікування з низьким енергоспоживанням, тоді як MPU збирає більше даних;

– програмоване переривання підтримує функції розпізнавання жестів, панорамування, масштабування, прокручування, виявлення дотиків та тремтіння камери.

4.3 Інтерфейс I2C

I2C – це двопровідний інтерфейс, що складається з послідовних даних (SDA) та сигналу послідовного тактового сигналу (SCL). У загальній реалізації інтерфейсу I2C підключеним пристроєм може бути головний пристрій або підлеглий пристрій [19]. Ведучий розмістить адресу ведомого на шині, а підлеглий з відповідною адресою підтвердить майстер. Під час спілкування з системним процесором MPU-60X0 завжди діє як підлеглий пристрій, тому він виконує роль головного пристрою. Лінії SDA і SCL зазвичай вимагають підтягувальних резисторів для VDD. Максимальна швидкість шини становить 400 кГц.

Адреса ведомого MPU-60X0 - b110100X, а довжина - 7 біт. Біт LSB 7-бітної адреси визначається логічним рівнем на виводі AD0. Це дозволяє підключити два MPU-60X0 до однієї шини I2C. При використанні в цій конфігурації адреса одного пристрою має бути b1101000 (контакт AD0-логіки низький), а адреса іншого-b1101001 (високий логічний висновок AD0).

4.4 Проектування архітектури системи

Предметна область – «Віртуальне середовище».

Так як програмне забезпечення представляє собою застосування для відображення рухів оператора кібернетичного костюма, то схема роботи застосування представлена у виді «отримав дані - відобразив на 3D-моделі». При отриманні пачки даних для певної кінцівки по осях x, y і z, система бере перше значення з пачки даних, яке є адресою тієї кінцівки, дані про становище в просторі якої були отримані від обробного центру кібернетичного костюма.

Далі по адресі, що була отримана з пачки даних, клас маршрутизатор розподіляє дані по осях x, y і z в той клас, якій має ту саму адресу і який відповідає за ту ж кінцівку, з якої були отримані вище зазначені дані.

В рамках розробки системи наведена та розглянута діаграмма компонентів системи. Діаграма компонентів представлена на рис. 4.1.

Як можно бачити з діаграми компонентів, система представлена чотирма частинами: MPU6050 певної кінціки кібернетичного костюму, Arduino Nano, що відповідає за обробку датчика кінцівки, збираючого центру, який з'єднує кібернетичний костюм з Unity 3D, та сам Unity 3D, в якому відбувається процес відтворення 3D-моделі оператора кібернетичного костюму.

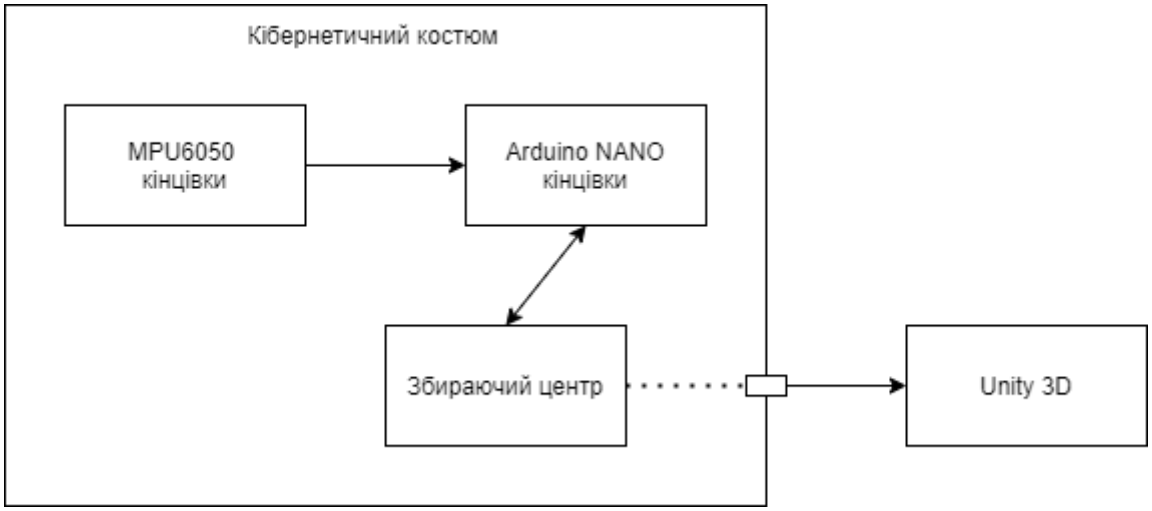

Рисунок 4.1 – Діаграма компонентів системи

Для візуалізації процесів взаємодії частин системи між собою були представлені дві діаграми активності. Діаграми активності представлені на рис. 4.2 та 4.3.

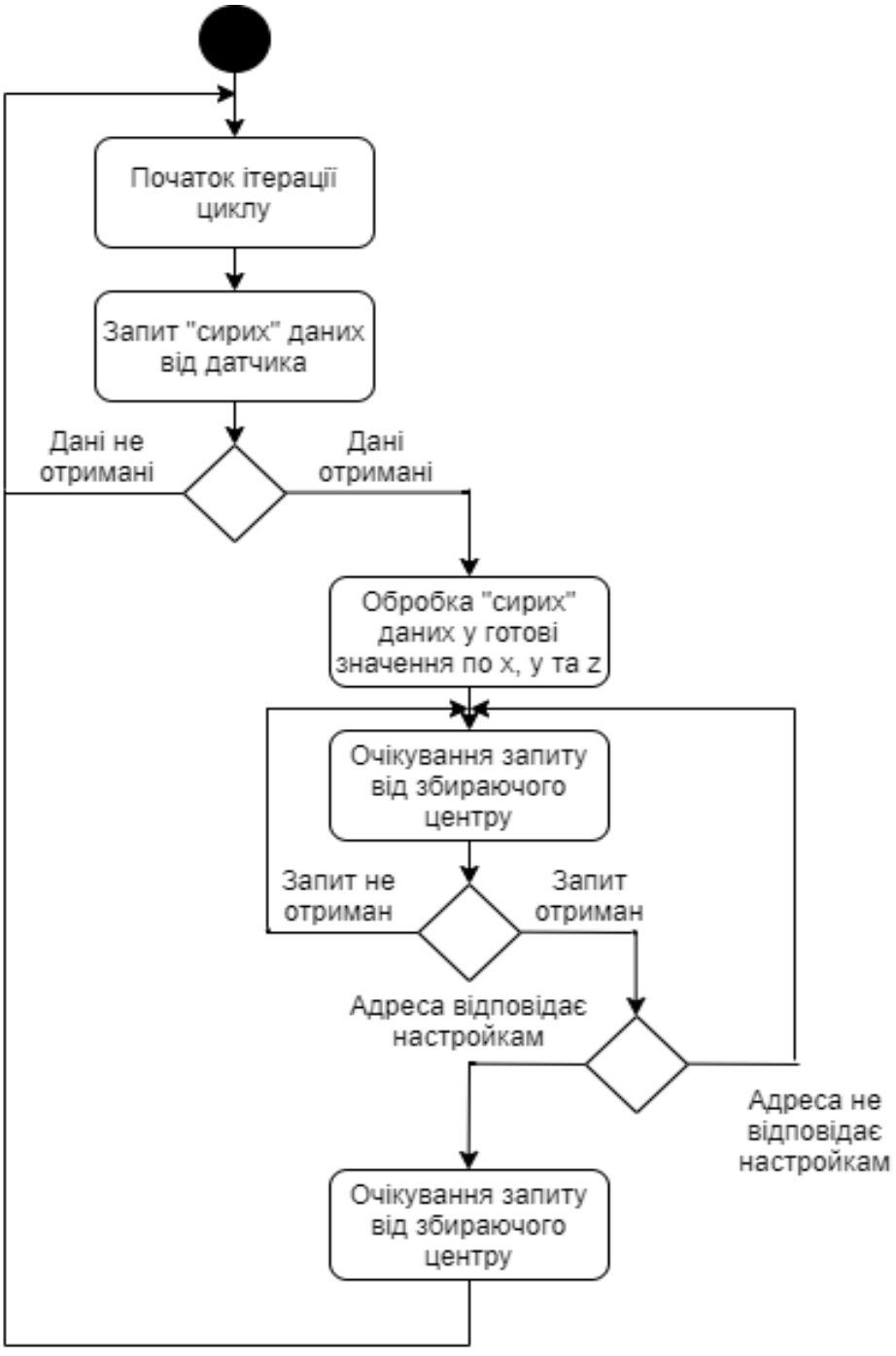

Рисунок 4.2 – Діаграма активності збираючого центру кібернетичного костюму

Діаграма активності збираючого центру кібернетичного костюму являє собою процес отримання та обробки "сирих" даних від датчика гіроскопа та акселерометра для всіх частин тіла оператора кібернетичного костюма.

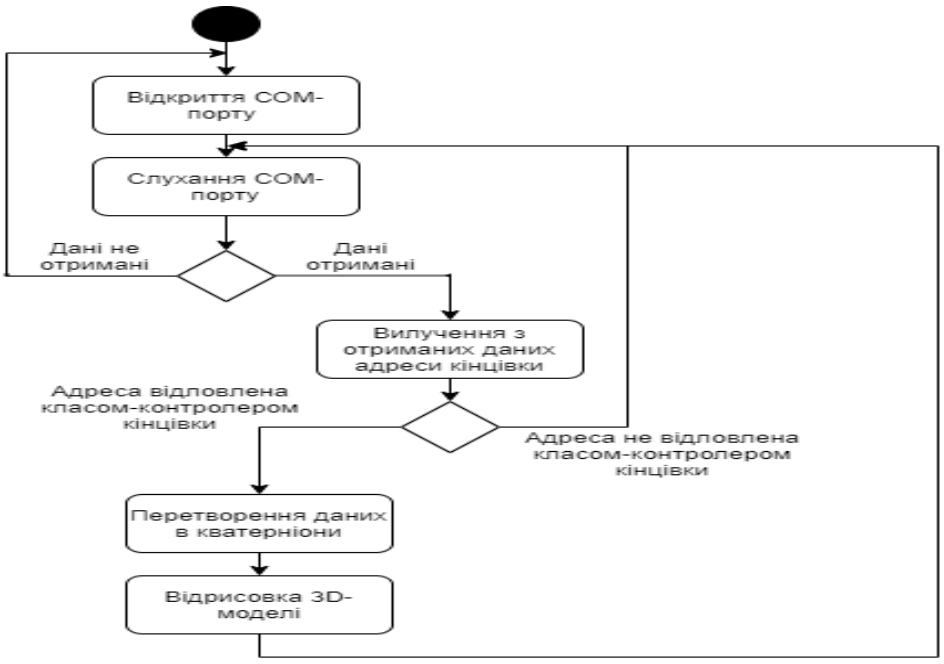

Рисунок 4.3 – Діаграма активності кібернетичного костюму та Unity 3D

Діаграма активності кібернетичного костюма Unity 3D являє собою процес передачі готових значень всіх датчиків кіберентичного костюма на скелет 3D-моделі в Unity 3D.

В рамках розробки даної системи наведена та розглянута концептуальна діаграма класів стосовно частини Unity 3D. Діаграма класів представлена на рис. 4.3.

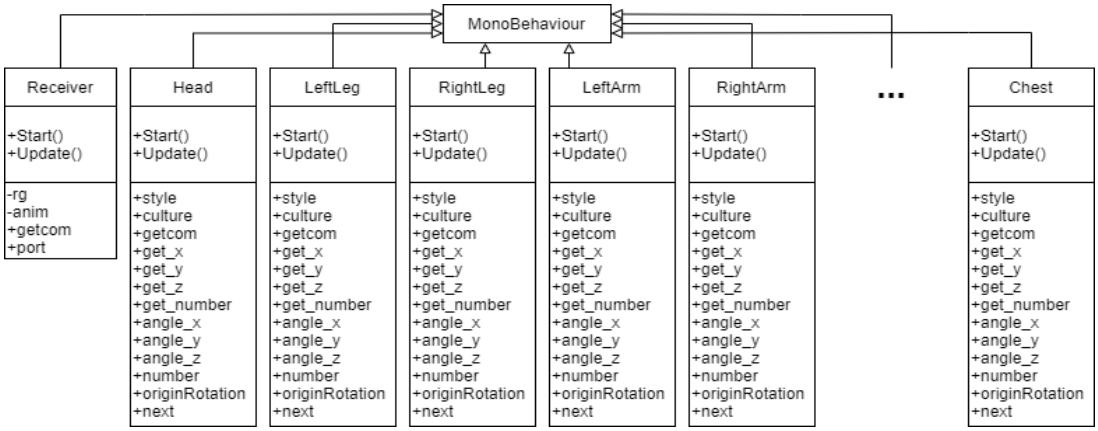

Рисунок 4.3 – Діаграма класів Unity-частини системи

MonoBehaviour - це базовий клас, з якого походить кожен сценарій Unity. Receiver - це клас-розподільник, який створює та активує COM-порт для зв'язку з

обробляючим центром кібернетичного костюму, запускає його, отримує дані від обробного центру, а після відсилає його на класи-контролери, які відповідають за кінцівки 3D-моделі. На діаграмі класів, що представлена на рис. 4.2, зображено 7 типових класів з 23 існуючих: Head, LeftLeg, RightLeg, LeftArm, RightArm, Head, Chest.

В табл. 4.1 описані методи класів для 3D-моделі.

Таблиця 4.1 – Методи класів проекту

Метод	Призначення
Start()	Метод класу Receiver, який ініціалізує і запускає зчитування даних з COM-порту
Update()	Метод класу Receiver, який отримує і оновлює дані, що прийшли з COM-порту
Start()	Метод класу Head, який ініціалізує обертання голови 3D-моделі
Update()	Метод класу Head, який виробляє вилучення даних по осях x, y і z з отриманої посилки даних, обчислює на їх основі кватерніони і передає команди на обертання голови 3D-моделі
Start()	Метод класу Chest, який ініціалізує обертання корпусу 3D-моделі
Update()	Метод класу Chest, який виробляє вилучення даних по осях x, y і z з отриманої посилки даних, обчислює на їх основі кватерніони і передає команди на обертання корпусу 3D-моделі
Start()	Метод класу LeftLeg, який ініціалізує обертання лівої ноги 3D-моделі
Update()	Метод класу LeftLeg, який виробляє вилучення даних по осях x, y і z з отриманої посилки даних, обчислює на їх основі кватерніони і передає команди на обертання лівої ноги 3D-моделі
Start()	Метод класу RightLeg, який ініціалізує обертання правої ноги 3D-моделі

Продовження таблиці 4.1

Метод	Призначення
Update()	Метод класу RightLeg, який виробляє вилучення даних по осях x, y і z з отриманої посилки даних, обчислює на їх основі кватерніони і передає команди на обертання правої ноги 3D-моделі
Start()	Метод класу RightArm, який ініціалізує обертання правої руки 3D-моделі
Update()	Метод класу RightArm, який виробляє вилучення даних по осях x, y і z з отриманої посилки даних, обчислює на їх основі кватерніони і передає команди на обертання правої руки 3D-моделі
Start()	Метод класу LeftArm, який ініціалізує обертання лівої руки 3D-моделі
Update()	Метод класу LeftArm, який виробляє вилучення даних по осях x, y і z з отриманої посилки даних, обчислює на їх основі кватерніони і передає команди на обертання лівої руки 3D-моделі

4.5 Проектування графічного інтерфейсу користувача

При проектуванні графічного інтерфейсу користувача були використані ресурси Unity 3D.

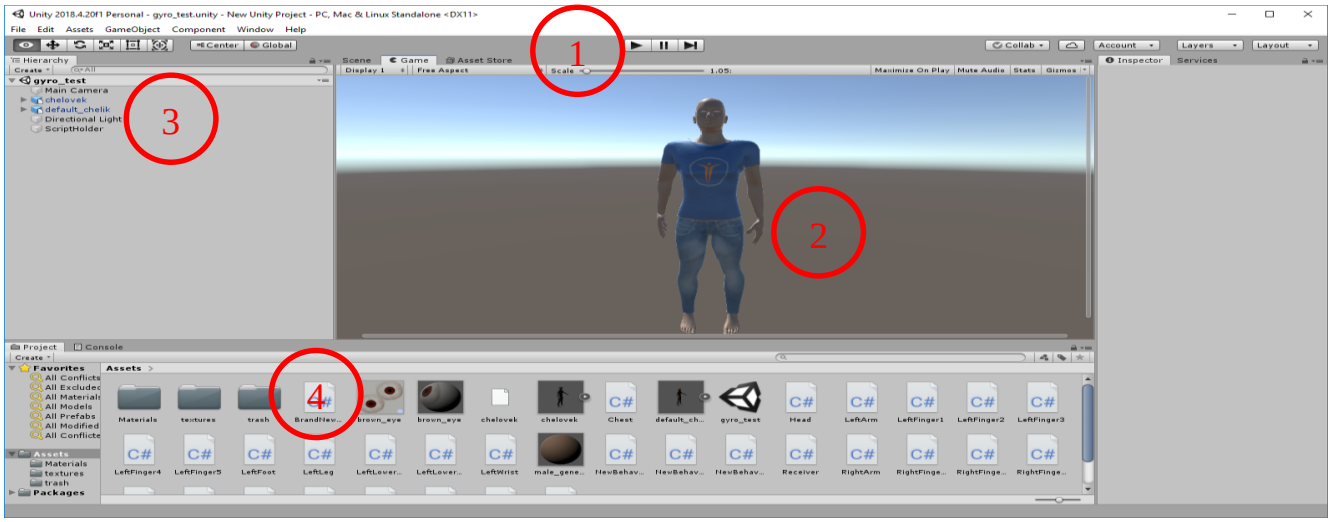

Рисунок 4.4 – 3D-модель

На рис. 4.4 зображено 3D-модель в Unity 3D. У центральній частині екрану знаходиться камера, яка дивиться на фронтальну частину 3D-моделі, на яку подаються дані з кібернетичного костюма. Проекція може бути змінена в будь-який момент

Пункти вікна:

1 – Панель управління. Дозволяє запускати та зупиняти роботу проекту, а також ставити його на тимчасову паузу.

2 – 3D моделі. Дозволяє відображати дії оператора кібернетичного костюма.

3 – Дерево частин тіла 3D моделі. Дозволяє побачити, з яких частин складається 3D модель, а також вибрати для обробки одну з частин.

4 – Панель скриптів та елементів 3D-моделі. Відображає всі елементи коду, що беруть участь у проекті, а також ресурси, необхідні для відтворення 3D-моделі, оточення та робочого процесу.

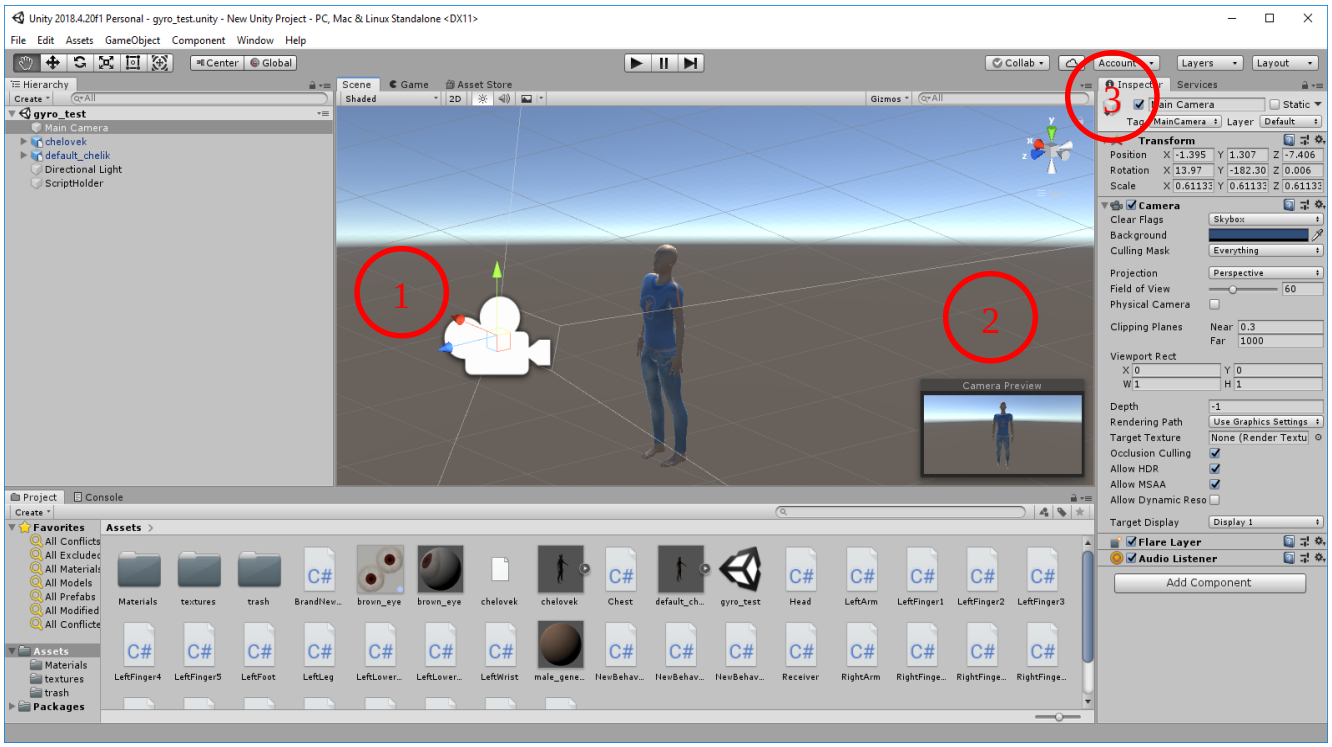

Рисунок 4.5 – Вікно коригування камери обзору на 3D-модель

На рис. 4.5 зображено вікно коригування камери обзору на 3D-модель.

Пункти вікна:

1 – Камера. Червона, зелена та синя стрілки дозволяють змінювати положення камери по осях x, y та z.

2 – Вікно попереднього перегляду. Дозволяє побачити проекцію на 3D-модель відразу після зміни положення камери.

3 – Панель параметрів камери. Дозволяє задати вручну положення камери, її поворот, нахил та видалення.

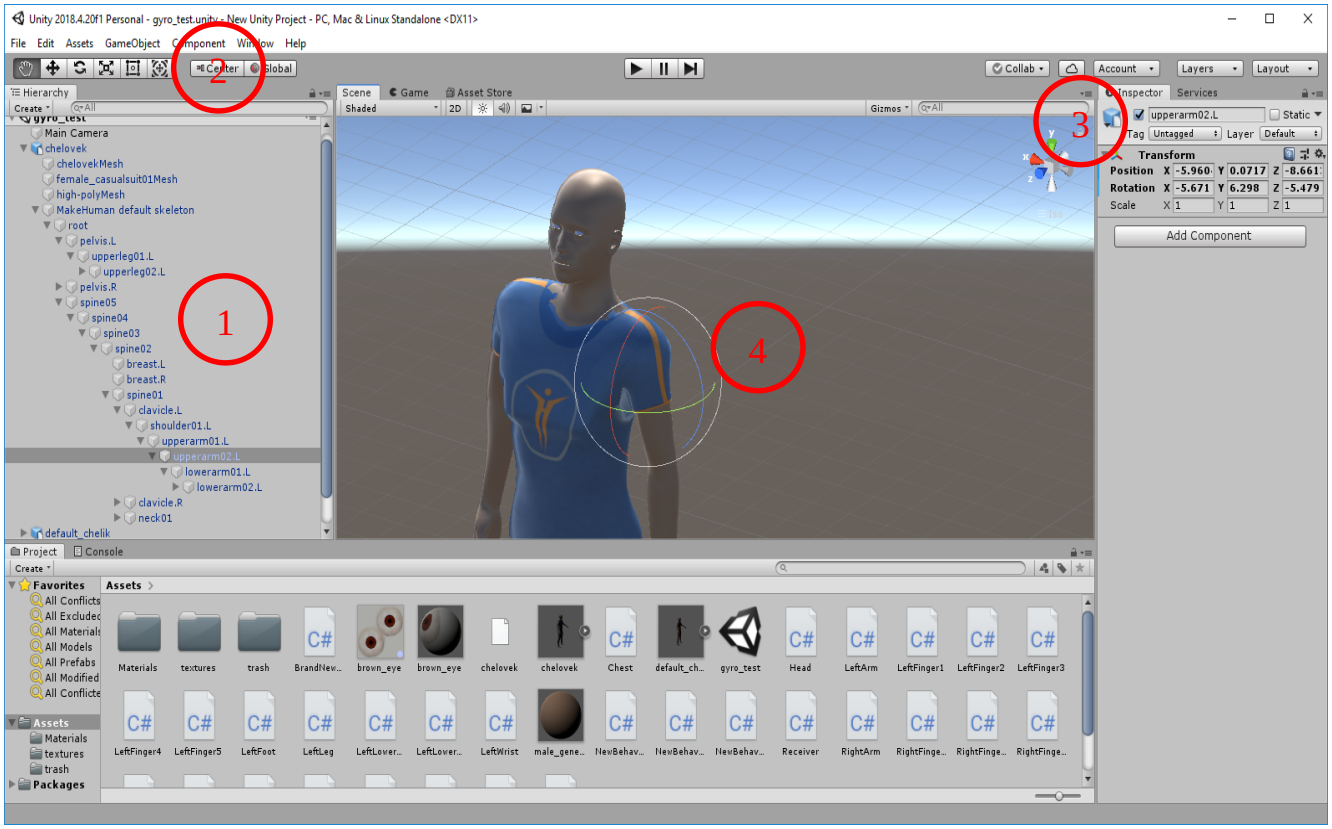

Рисунок 4.6 – Вікно коригування початкового положення частин тіла 3D-моделі

На рис. 4.6 зображено коригування початкового положення частин тіла 3D-моделі.

Пункти вікна:

1 – Дерево частин тіла 3D моделі. Дозволяє побачити, з яких частин складається 3D модель, а також вибрати для обробки одну з частин.

2 – Панель інструментів. Дозволяє вибирати режими для зміни положення частини 3D-телі моделі (переміщення, обертання).

3 – Панель параметрів частини тіла 3D-моделі. Дозволяє задати вручну положення частини тіла 3D-моделі, її поворот та нахил.

4 – Інструментарій для ручної зміни положення частини тіла 3D моделі. Червона, зелена та синя смуги дозволяють змінювати нахил частини тіла 3D-моделі по осі x, y та z відповідно.

Як можна бачити зі скріншотів інтерфейсу, Unity 3D дуже простий у використанні і дозволяє швидко вносити калібрування положення 3D моделі під потрібні вимоги.

5 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

5.1 Опис програмних технологій

Для програмної реалізації першої частини алгоритму обрана мова програмування пристроїв Ардуіно, що заснована на C / C ++ та скомпонована з бібліотекою AVR Libc та дозволяє використовувати будь-які її функції. Сі - це компільована статично типізована мова програмування загального призначення, відповідно до дизайну якої, її конструкції близько зіставляються типовим машинним інструкціям. Також "C" є універсальною мовою програмування. Вона характеризується простими виразами, сучасними процесами управління та структурами даних та великою кількістю опцій оператора. Але відсутність обмежень і спільність мови роблять її більш зручною та ефективною для багатьох завдань [20].

В якості середовища розробки для написання коду на мові Ардуіно був використаний редактор Arduino IDE. Це сучасний інструмент, що має багато зручних функцій, які полегшують процес написання коду, контролю його версій, тестування та перевірку коректності.

Для реалізації другої частини алгоритму обрана мова програмування C#, яка є сучасною об'єктно-орієнтованою та типобезопасною мовою програмування. В якості середовища розробки для написання коду на C# було використано Unity 3D - міжплатформене середовище розробки комп'ютерних ігор. Unity дозволяє створювати додатки, що працюють під більш ніж 20 різними операційними системами, що включають персональні комп'ютери, ігрові консолі, мобільні пристрої, інтернет-додатки та інші.

Основними перевагами Unity є наявність візуальної середовища розробки, міжплатформеної підтримки і модульної системи компонентів. До недоліків відносять появу складнощів при роботі з багатокомпонентними схемами і труднощі при підключенні зовнішніх бібліотек [21].

5.2 Опис програмних бібліотек

При розробці програмного забезпечення для кібернетичного костюма використовувалися бібліотеки, описані в розділах 5.2.1 – 5.2.5.

5.2.1 SoftwareSerial.h. Arduino реалізує апаратну підтримку інтерфейсу послідовного передачі даних через контакти 0 і 1 (також використовується для зв'язку з комп'ютером через USB). Апаратне забезпечення з послідовним інтерфейсом реалізовано за допомогою UART. Це дозволяє мікроконтролеру Atmega обробляти дані навіть під час обробки інших завдань.

Бібліотека SoftwareSerial дозволяє використовувати програмне забезпечення, яке реплікує функціональність UART (звідси назва «SoftwareSerial») для реалізації послідовного порту на будь-якому іншому цифровому виході Arduino. Ця бібліотека дозволяє програмно створювати кілька послідовних портів, що працюють на швидкості до 115 200 бод. Для пристроїв, які використовують інвертовані сигнали, бібліотека надає відповідні параметри, в тому числі інвертовані.

При використанні кількох послідовних портів одночасно можна отримати лише один з них [22].

На платах Arduino Mega і Mega2560 деякі контакти не підтримують переривання, які виникають при зміні рівня сигналу. Тому як виходи RX на цих платах можна використовувати лише наступні виходи: 10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8 (62), A9 (63), A10 (64)) , A11 (65), A12 (66), A13 (67), A14 (68), A15 (69).

На Arduino Leonardo деякі результати не підтримують переривання, які виникають при зміні рівня сигналу. Тому в якості виходу RX можна використовувати лише наступні контакти на цій платі: 8, 9, 10, 11, 14 (MISO), 15 (SCK), 16 (MOSI).

5.2.2 Wire.h. Ця бібліотека дозволяє вам взаємодіяти з пристроями I2C/TWI. На платах Arduino з компоновкою R3 (термостат 1.0), SDA (лінія передачі даних) і

SCL (лінія тактового сигналу) можна використовувати, щоб зробити висновки про вихід AREF. Arduino Due має два інтерфейси I2C/TWI: SDA1 і SCL1 розташовані біля виходу AREF, а додаткові дроти розташовані на контактах 20 і 21.

Існують 7-розрядні та 8-розрядні версії адресації I2C. Сьомий біт ідентифікує пристрій, а восьмий біт визначає, чи надсилаються дані чи зчитуються з пристрою. Бібліотека використовує 7 адрес для адресації. Якщо вихідний код містить 8-бітну адресацію, ви повинні відкинути молодші біти (зсунути значення на 1 біт вправо). Адреса від 0 до 127 [23].

5.2.3 SPI.h. Ця бібліотека дозволяє взаємодіяти з пристроями SPI через плату Arduino, яка діє як головний пристрій.

Послідовний периферійний інтерфейс SPI — це синхронний послідовний протокол даних, який використовується мікроконтролером для зв’язку з одним або кількома периферійними пристроями на невеликій відстані. Його також можна використовувати для зв'язку між двома мікроконтролерами.

Через з’єднання SPI завжди є головний пристрій (зазвичай мікроконтролер) для керування периферійними пристроями. Зазвичай всі пристрої мають три загальнодоступні лінії:

MISO (Master In Slave Out, Master In, Slave Out) — відома лінія пристрою, що використовується для передачі даних на головний пристрій;

MOSI (Master Out Slave In, Master Out, Slave In) - лінія, по якій ведучий пристрій передає дані від периферійного пристрою;

SCK (Serial Clock) - тактовий імпульс для синхронної передачі даних, що виводиться ведучим пристроєм;

І один рядок, окремий для кожного пристрою:

SS (Slave Select) – вихід на кожному пристрої, що дозволяє майстрові вмикати або вимикати використання певних підпорядкованих пристроїв.

Коли вибраний вихід з SS встановлений на низький логічний рівень, відомий пристрій взаємодіє з помічником. Якщо вихід SS встановлений на високий

логічний рівень, підпорядкований пристрій ігноруватиме майстра. Це дозволяє мати кілька пристроїв SPI, які використовують ті самі лінії MISO, MOSI і CLK.

Щоб написати код для нового пристрою SPI, потрібно пам’ятати наступні моменти:

Порядок передачі даних наступний: старший біт (MSB, найбільш значущий біт) чи молодший біт (LSB, найменш значущий біт) першим? Він керується другим параметром у SPISettings: MSBFIRST або LSBFIRST. Більшість мікросхем SPI використовують порядок першого MSB.

Стандарт SPI забезпечує ступінь свободи, і реалізація кожного пристрою має незначні відмінності. Це означає, що при написанні коду потрібно звернути особливу увагу на технічний опис пристрою [24].

5.2.4 Ресурси Unity 3D. Unity — це кросплатформне середовище розробки комп’ютерних ігор. За допомогою Unity ви можете створювати програми, які працюють на більш ніж 20 різних операційних системах, включаючи ПК, ігрові консолі, мобільні пристрої, Інтернет-додатки тощо. Unity була випущена в 2005 році і з тих пір продовжує розвиватися.

Основними перевагами Unity є наявність візуального середовища розробки, кросплатформна підтримка та модульна компонентна система. Недоліками є труднощі при використанні багатокомпонентних схем і труднощі при підключенні до зовнішніх бібліотек.

На Unity написані тисячі ігор, додатків і моделювання для різних платформ і типів. У той же час великі розробники та незалежні студії використовують Unity [25].

Редактор Unity має простий інтерфейс Drag & Drop, який легко налаштувати і складається з різних вікон, тому ви можете налагоджувати гру безпосередньо в редакторі. Движок підтримує дві мови сценаріїв: C#, JavaScript (змінений). Boo (діалект Python) раніше підтримувався, але був видалений у версії 5. Фізичні розрахунки виконуються за допомогою фізичного движка PhysX NVXIA.

Проекти в Unity поділяються на сцени (рівні) - єдиний файл, що містить ігровий світ і власні об'єкти, скрипти та налаштування. Сцена може містити об'єкти (моделі) і порожні ігрові об'єкти-об'єкти без моделей ("пустушки"). У свою чергу, об’єкт містить набір компонентів, з якими взаємодіє скрипт. Об’єкти також мають назву (Unity допускає два або більше об’єктів з однаковою назвою), можуть мати мітку (мітку) і рівень, на якому вона має відображатися. Тому кожен об’єкт на сцені повинен мати компонент Transform – він зберігає координати положення, обертання та розміру об’єкта по всіх трьох осях. За замовчуванням об’єкти з видимою геометрією також мають компонент візуалізації сітки, щоб зробити об’єктну модель видимою.

Зіткнення можна застосувати до об’єктів (так звані коллайдерами в Unity), яких існує кілька типів.

Unity також підтримує фізику твердого тіла та організаційну фізику, а також Rag Doll. Редактор має систему успадкування об’єктів; дочірні об’єкти повторюють будь-які зміни положення, повороту та масштабу батьківського об’єкта. Сценарій в редакторі додається до об'єкта як окремий компонент.

Unity 3D підтримує систему рівня деталізації (LOD), суть якої полягає в заміні високодетальних моделей на менш деталізовані моделі далеко від гравця, і навпаки, а також систему відсіювання оклюзії, яка по суті є об’єктом, який не потрапляє в поле зору камери, не візуалізуючи геометрію та зіткнення, тим самим зменшуючи процесор і дозволяючи оптимізувати проект. Під час компіляції проекту створюється виконуваний файл гри (.exe) (для Windows) і розташовується в окремій папці-Game Data (включаючи всі ігрові шари та спільні бібліотеки).

5.3 Інструкція з встановлення реалізованого алгоритму

Для встановлення першої частини алгоритму (програмного забезпечення для кібернетичного костюму) необхідно мати його вихідний код, драйвер USB-роз’єму CH340 та встановлену середу розробки Arduino IDE. Необхідно виконати команду компіляції та завантаження коду на апаратний носій, що знаходиться в

обробляючому центрі кібернетичного костюму. Далі, для встановлення другої частини алгоритму (програмного забезпечення для взаємодії з віртуальною реальністю) необхідно мати його вихідний код та встановлену систему контроля версій git. Необхідно виконати команду git clone для клонування репозиторію на локальний комп'ютер або отримати готовий архів та розархівувати його. Необхідний розмір вільної пам'яті на ПК: 500 МБ. Додаткових дій по налаштуванню виконувати непотрібно.

5.4 Інструкція з використання

Для встановлення програмного забезпечення для відтворення віртуального середовища необхідно мати встановлену на ПК платформу для розробки програм Unity 3D.

Старт сканування положення кінцівок оператора кібернетичного костюму для взаємодії з віртуальною реальністю відбувається автоматично одразу після підключення костюму до USB комп'ютера.

Для відтворення положення тіла оператора костюму у віртуальному середовищі потрібно вказати COM-порт, до якого приєднано кібернетичний костюм та виконати команду «Play». Після декількох секунд ініціалізації COM-порту, програмних забезпечень костюму та Unity 3D, 3D-модель почне відтворення рухів оператора костюму в реальному часі.

6 ТЕСТУВАННЯ СИСТЕМИ

6.1 Функціональне тестування

Для тестування системи будемо використовувати функціональне тестування.

Функціональне тестування (англ. Functionaltesting) – це тестування, метою якого є виявлення невідповідностей між реальною поведінкою реалізованих функцій та очікуваною поведінкою відповідно до специфікацій, складених до програми.

Функціональні тести повинні охоплювати всі реалізовані функції з урахуванням найбільш ймовірних типів помилок. Тестові сценарії, що поєднують окремі тести, орієнтовані на перевірку якості розв’язку функціональних задач.

Тестування програмного забезпечення - перевірка відповідності між реальним і очікуваним поведінкою програми, що здійснюється на кінцевому наборі тестів, обраному певним чином.

Цілі тестування:

– підвищити ймовірність того, що додаток буде працювати правильно при будь-яких обставинах;

– підвищити ймовірність того, що додаток буде відповідати всім описаним вимогам;

– надання актуальної інформації про стан продукту на даний момент.

Тест дизайн - етап процесу тестування ПО, на якому проектуються и створюються тестові випадки (тест кейси), відповідно до визначених раніше критеріями якості та цілями тестування.

6.2 Матриця відповідності вимог

Traceability matrix - Матриця відповідності вимог - двовимірна таблиця, яка містить відповідність функціональних вимог (functional requirements) ПО і підготовлених тестових сценаріїв (test cases). У табл. 6.1 представлена матриця відповідності вимог.

Таблиця 6.1 – Матриця відповідності вимог

Функціональні вимоги / Test Case	Активація системи	Калібрування кристалів датчиків	Відкриття та налаштування COM-порту	Перемальовка частини 3D-моделі у Unity 3D
Test Case #1	X			
Test Case #2		X		
Test Case #3			X	
Test Case #4				X
Test Case #5	X			
Test Case #6			X	

Тестовий випадок (Test Case) - артефакт, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації тестируемой функції або її частини.

У табл. 6.2-6.5 представлено опис декількох позитивних тест-кейсів. Позитивний тест кейс використовує тільки коректні дані і перевіряє, що додаток правильно виконало функцію, що викликається.

Таблиця 6.2 – Опис тестового сценарію – Test Case №1

Ідентифікатор	Test Case №1
Назва	Дозволяє перевірити активацію системи та її готовність до обміну даних
Передумови	1.Користувач подав енергоживлення на кібернетичний костюм 2.Користувач з'єднав кіберкостюм та Unity 3D.

Продовження таблиці 6.2

Ідентифікатор	Test Case №1
Опис кроків	1.Після подачі енергоживлення чекаємо кілька секунд для калібрування кристалів датчиків 2.В Unity 3D вказуємо номер COM-порту 3.Натискаємо кнопку "Старт".
Очікуваний результат №1	1.Якщо фізичне з'єднання фіксоване і COM-порт вказано правильно, користувач отримує повторення своїх рухів на 3D-моделі

Таблиця 6.3 – Опис тестового сценарію – Test Case №2

Ідентифікатор	Test Case №2
Назва	Дозволяє системі провести калібрування кристалів датчиків
Передумови	1.Користувач подав енергоживлення на кібернетичний костюм 2.Користувач з'єднав кіберкостюм та Unity 3D.
Опис кроків	1.Одразу після подачі енергоживлення користувач кілька секунд стоїть нерухомо по стійці смирно.
Очікуваний результат №1	1.Система обчислює коректні значення по осях x, y та z.

Таблиця 6.4 – Опис тестового сценарію – Test Case №3

Ідентифікатор	Test Case №3
Назва	Дозволяє системі зробити відкриття та налаштування COM-порту
Передумови	1.Користувач подав енергоживлення на кібернетичний костюм 2.Користувач з'єднав кіберкостюм та Unity 3D. 3.Користувач встановив драйвер CH340

Продовження таблиці 6.4

Ідентифікатор	Test Case №3
Опис кроків	1.Користувач відкриває програмне забезпечення в Unity 3D 2.Користувач вказує назву COM-порту, до якого приєднано кібернетичний костюм 3.Користувач вказує швидкість даних у бодах, кількість біт та інші параметри 4.Користувач натискає "Старт"
Очікуваний результат №1	1.Система відкриває COM-порт згідно з заданими параметрами

Таблиця 6.5 – Опис тестового сценарію – Test Case №4

Ідентифікатор	Test Case №4
Назва	Дозволяє системі здійснити перемальовування частини 3D-моделі в Unity 3D
Передумови	1.Користувач подав енергоживлення на кібернетичний костюм 2.Користувач з'єднав кіберкостюм та Unity 3D.
Опис кроків	1.Користувач вказує назву COM-порту 2.Користувач наживає кнопку «Старт» 3.Unity 3D отримує нові дані від збираючого центру кібернетичного костюма
Очікуваний результат №1	1.Кінцевість 3D-моделі змінює своє положення відповідно до положення реальної частини тіла користувача

У табл. 6.6-6.7 розглянуто два негативних тест-кейсів. Негативний тест кейс оперує як коректними, так і некоректними даними (мінімум 1 некоректний параметр) і ставить за мету перевірку виняткових ситуацій.

Таблиця 6.6 - Опис тестового сценарію - Test Case №5

Ідентифікатор	Test Case №5
Назва	Дозволяє не допустити самовільні рухи 3D-моделі за відсутності коректних даних від кібернетичного костюма
Передумови	1.Користувач подав енергоживлення на кібернетичний костюм 2.Користувач з'єднав кіберкостюм та Unity 3D.
Опис кроків	1.Після подачі енергоживлення чекаємо кілька секунд для калібрування кристалів датчиків 2.В Unity 3D вказуємо номер COM-порту 3.Натискаємо кнопку "Старт".
Очікуваний результат №1	1.Оскільки фізичне з'єднання не зафіксовано / COM-порт не вказано правильно, 3D-модель залишається у вихідному стані

Таблиця 6.7 - Опис тестового сценарію - Test Case №6

Ідентифікатор	Test Case №6
Назва	Дозволяє системі не допустити відкриття та налаштування COM-порту при неправильно заданих параметрах
Передумови	1.Користувач подав енергоживлення на кібернетичний костюм 2.Користувач з'єднав кіберкостюм та Unity 3D. 3.Користувач встановив драйвер CH340
Опис кроків	1.Користувач відкриває програмне забезпечення в Unity 3D 2.Користувач вказує неправильну назву COM-порту, до якого приєднано кібернетичний костюм 3.Користувач вказує швидкість даних у бодах, кількість біт та інші параметри 4.Користувач натискає "Старт"

Продовження таблиці 6.7

Ідентифікатор	Test Case №6
Очікуваний результат №1	1.Система не відкриває COM-порт згідно з заданими параметрами, у зв'язку з незайнятістю зазначеного COM-порту

Отже, у табл. 6.2-6.7 представлено опис декількох позитивних та неготивних тест-кейсів, що ставлять за мету перевірку коректності даних та роботу системи в виняткових ситуаціях.

6.3 Експерименти щодо вимірювання характеристик костюма та його програмного забезпечення

Для оцінки ефективності роботи прототипів кібернетичного костюму була проведена серія експериментів.

6.3.1 Апаратура для оцінки ефективності роботи системи. Для отримання готових даних та аналізу обміну даними в цифрових та аналогових шинах кібернетичного костюма використовувалися:

– Цифровий осцилограф UNI-T UTD2025C, 25МГц, з екраном на 5.7 дюймів.

Цифровий осцилограф UNIT-T UTD-2025C – це професійний прилад, який слугує для спостереження сигналу з автоматичним визначенням його параметрів. Інструмент наділений кольоровим 5,7-дюймовим рідкокристалічним дисплеєм, здатністю математичної обробки сигналу та можливістю комутації з комп'ютером. Невелика вага та компактність дозволяють використовувати пристрій не тільки стаціонарно, а й як переносний інструмент. Можливості приладу Прилад може виконувати вимірювання за 28 параметрами в автоматичному режимі, копіюючи зображення з дисплея. Передбачено також запис форми сигналу з наступним його відтворенням. Крім цього, осцилограф має функцію збереження форми сигналу та налаштувань. Об'єм пам'яті цього пристрою досягає 25000 пікселів. Смуга

пропускання приладу дорівнює 25 мегагерцям, а швидкість вибірки становить 250 мільйонів вибірок за секунду. Цифровий осцилограф UNIT-T UTD-2025C може працювати від джерела живлення з напругою від 100 до 240 вольт. Підключення до комп'ютера виконується за допомогою USB-інтерфейсу [26].

– Логічний аналізатор Saleae 8 каналів.

Логічний аналізатор Saleae є клоном найпопулярнішого логічного аналізатора Saleae Logic. Підключається до комп'ютера через USB (кабель у комплекті), дозволяє аналізувати дані, що надходять на будь-який з 8 каналів (одночасно). Дуже зручно для налагодження, реверсу інжинірингу, ознайомлення з роботою різних протоколів. Працює з рівнем сигналів 5В та нижче. Підтримує протоколи CAN, DMX-512, I2C, I2S/PCM, MANCHESTER, 1-WIRE, SPI, Simple Parallel, UNI/O, UART, Частота дискретизації до 24МГц (можна вибирати 16МГц, 12МГц, 8МГц, 4МГц, МГц, 1 МГц, 500 кГц, 250 кГц, 200 кГц, 100 кГц, 50 кГц, 25 кГц; Працює з офіційним ПЗ Logic Software, ОС: Windows XP, Vista, 7, MacOS X, Ubuntu, архітектури 32 та 64 біт [27].

– Arduino IDE.

Arduino IDE – це інтегроване середовище розробки для Windows, MacOS та Linux, розроблене на Сі та C++, призначене для створення та завантаження програм на Arduino-сумісні плати, а також на плати інших виробників.

Просте та функціональне середовище розробки для створення власного ПЗ, яким керуються численні пристрої, зібрані початківцями та досвідченими електронниками. З'єднання ПК із мікроконтролером реалізовано через інтерфейс USB. Код мовою Сі та C++ пишеться в редакторі, в якому є підсвічування команд та спеллчекер [28].

6.3.2 Оцінка ефективності першого прототипу кібернетичного костюма. Робота першого прототипу кібернетичного костюма ґрунтувалася на цифровому комутаторі шини I2C TCA9548A. 8 датчиків акселерометра і гіроскопа перебували в обробці комутатора, решта двох оброблялася безпосередньо платою Arduino Mega. Тестування ефективності проводилося за допомогою вбудованого

середовища розробки Arduino IDE. В ході експериментів швидкість обміну даними між збираючим центром кіберкостюму та Unity 3D досягала 9600 бод на секунду. Кількість даних в одну ітерацію обробки - 10 значень осі x, 10 значень осі y і 10 значень осі z, в сумі 30 значень у форматі дробового числа. Адреси кінцівок опущені.

Дані, що використовуються для обробки, мають вигляд, представлений у табл. 6.8.

Таблиця 6.8 – Готові дані для першого прототипу

Назва	Тип	Точність
accel_N_x	float	два знаки після коми
accel_N_y	float	два знаки після коми
accel_N_z	float	два знаки після коми

У процесі парсингу одержуваних даних було відзначено мимовільна інкрементація всіх 10 значень осі z. Кожні 2-3 ітерації мікроконтролера значення осі z збільшувалося на одну одиницю. Якщо взяти всі 30 значень, що видаються мікроконтролером збираючого центру, за 100%, то перешкода, що зустрічається, становить одну третину від усіх значень, тобто, округлено, 33%. Отже, коректність даних можна округлено оцінити лише 66%. Крім того, архітектура першого прототипу не передбачала збільшення числа датчиків і не дозволяла отримати швидкий доступ до апаратної архітектури костюма.

6.3.3 Оцінка ефективності другого прототипу кібернетичного костюма. Апаратна та програмна архітектура другого прототипу кібернетичного костюма ґрунтувалася на протоколі UART (Universal asynchronous receiver/transmitter) та чотирьох платах посередниках Arduino Nano, кожна з яких обробляла одну конкретну кінцівку. Два датчики лінії "Голова-Груди" знаходилися в обробці центральною платою Arduino Mega, як і в першому прототипі. Тестування

ефективності проводилося за допомогою інтегрованого середовища розробки Arduino IDE та логічного аналізатора Saleae. В ході аналізу швидкість обміну даними між збираючим центром кіберкостюму і Unity 3D варіювалася між 9600 і 115200 бод на секунду, проте найкраща швидкість експериментально була встановлена на 9600 бод на секунду. При збільшенні швидкості обміну було відзначено зависання програмного забезпечення Unity 3D, а також сильну затримку в отриманні нових пачок даних від збираючого центру. Кількість даних в одну ітерацію обробки залишилося незмінним - 10 значень осі x, 10 значень осі y і 10 значень осі z, в сумі 30 значень у форматі дробового числа. Адреси кінцівок опущені. У процесі парсингу одержуваних даних було відзначено та сама мимовільна інкрементація всіх 10 значень осі z, що і в першому прототипі. Кожні 2-3 ітерації мікроконтролера значення осі z збільшувалося на одну одиницю. Крім того, в шинах обміну даними UART між платами посередника і центральною платою стали спостерігатися спотворення і втрата даних по всіх осях всіх кінцівок. З усередненого підрахунку перешкод можна відзначити від 3 до 17 спотворень і втрат на 100 ітерацій центру, що збирає. Якщо взяти всі 30 значень, що видаються мікроконтролером збираючого центру, за 100%, то мимовільна інкрементація по осі z становить одну третину від усіх значень, тобто, округлено, 33%.

Дані, що використовуються для обробки, мають вигляд, представлений у табл. 6.9.

Таблиця 6.9 – Готові дані для другого прототипу

Назва	Тип	Точність
accel_N_x	float	два знаки після коми
accel_N_y	float	два знаки після коми
accel_N_z	float	два знаки після коми

Отже, коректність даних можна округлено оцінити в 66% по осях x і y, за умови, що перешкод та спотворень немає. Враховуючи вищевикладене спостереження, можна відзначити, що коректність значень осі x і y кожні 100

ітерацій за наявності перешкод становить від 27,4% до 32,01% від початкових 33%. Підсумовуючи, можна відзначити, що загальна точність даних другого прототипу складала в середньому 59,41%. Крім того, другий прототип зберігав конструктивні недоліки першого прототипу, а саме не передбачав збільшення числа датчиків та не дозволяла отримати швидкий доступ до елементів апаратної архітектури костюма.

6.3.4 Оцінка ефективності третього прототипу кібернетичного костюма. Як було зазначено раніше, апаратну та програмну архітектуру третього прототипу кібернетичного костюма було повністю перероблено під нову концепцію "Контролер - датчик". Ця концепція передбачає, що кожен датчик кібернетичного костюма перебуває у обробці лише одного мікроконтроллера. Як мікроконтролер використовується Arduino Nano. Дана архітектура виключає високе навантаження на мікроконтроллер та дозволяє центру, що збирає, виконувати тільки функції збирання даних від усіх мікроконтролерів і відсилання їх в Unity 3D. Крім того, в третьому прототипі стала використовуватися бібліотека MPU6050 6of від Еріка Олімана, яка знайшла своє застосування переважно в безпілотних літальних апаратах та дронах. Робота бібліотеки характеризується 99,99% точністю даних по осях x, y та z. Виходячи з застосування нового високоточного алгоритму вдалося позбутися мимовільної інкрементації значень по осі z і виключити наявність перешкоди готових даних на стороні мікроконтролера-посередника. Для збору готових даних збираючим центром використовується послідовний периферійний інтерфейс SPI (Serial Peripheral Interface), який дозволяє центральному мікроконтролеру обробляти незалежну кількість посередників.

Дані, що використовуються для обробки, мають вигляд, представлений у табл. 6.9.

Таблиця 6.9 – Готові дані для третього прототипу

Назва	Тип	Точність
accel_N_x	float	два знаки після коми

Продовження таблиці 6.9

Назва	Тип	Точність
accel_N_y	float	два знаки після коми
accel_N_z	float	два знаки після коми

У початковій версії третього прототипу в збираючому центрі відзначалися спотворення всіх значень по осях x, y і z. Для їх усунення було введено систему цифрових фільтрів. Завдання фільтрів полягає у перевірці кожного значення на різке збільшення чи зменшення числа порівняно з попереднім. Також, фільтри не допускають такі типи значень, як: -0,00, 0,00, -180.0, 180.0. Однак система фільтрів не здатна відловити такий тип значення, як "nan". Частота появи "nan" на сотню ітерацій всіх значень становить від 2 до 9 одиниць, що становить від 2% до 7%. Звідси можна дійти висновку, що загальна коректність даних становить у середньому 94,5%. Крім того, нова архітектура "Контролер - датчик" дозволяє отримати швидкий доступ до елементів апаратної архітектури костюма.

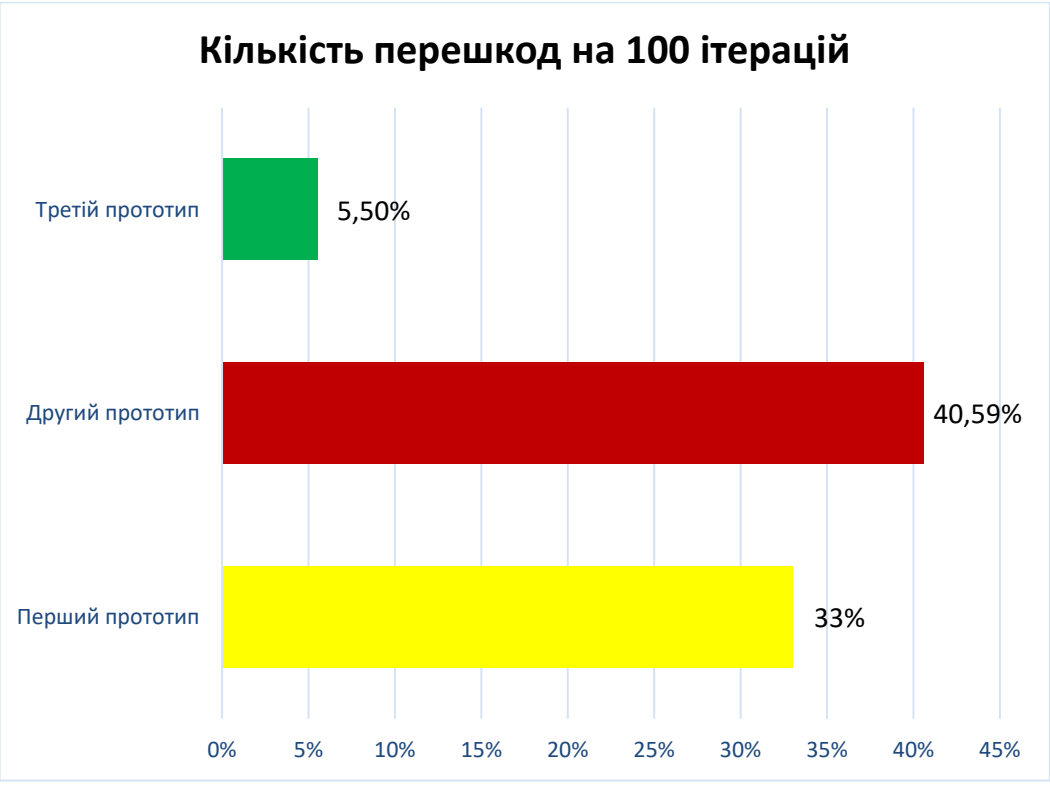

Рисунок 6.1 – Кількість перешкод на 100 ітерацій

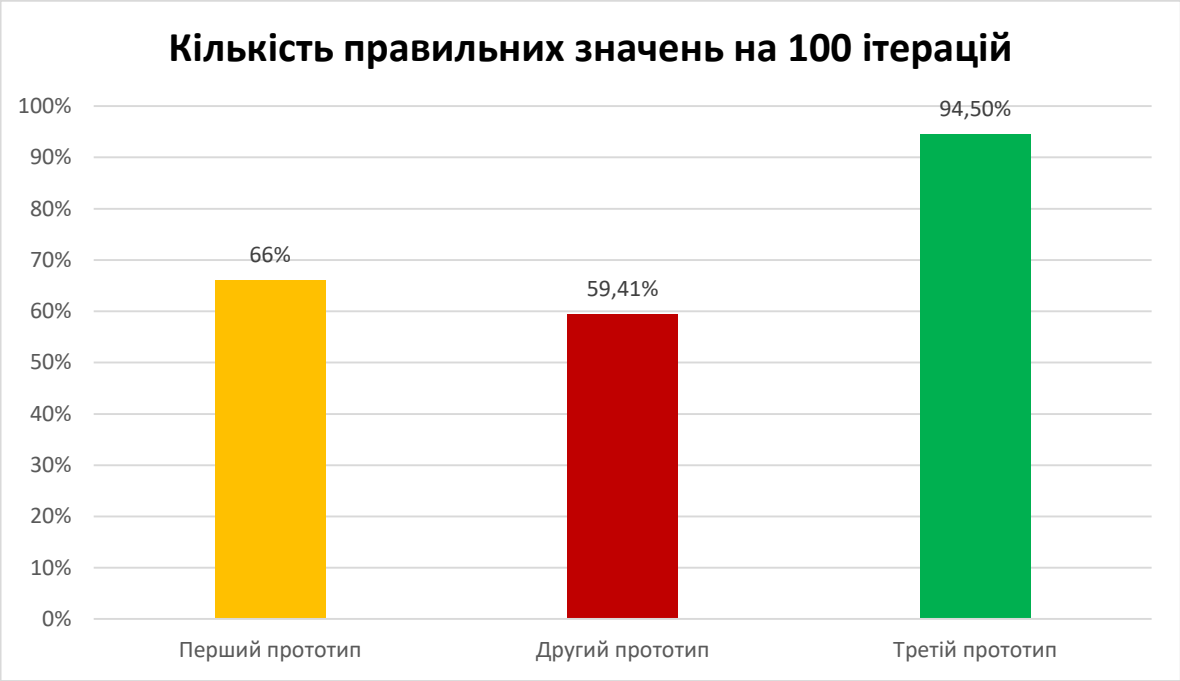

Рисунок 6.2 – Кількість правильних значень на 100 ітерацій

Як результат, модернізація кібернетичного костюма для взаємодії з віртуальною реальністю дозволила підвищити точність даних з 66% до 94,5% (див. рис. 6.1), а також знизити кількість перешкод з 40,59% до 5,5% (див. рис. 6.2).

ВИСНОВКИ

Основним завданням даної роботи є удосконалення архітектури кібернетичного костюму з можливістю взаємодії з віртуальною реальністю та розробка програмного забезпечення для нього. У ході виконання кваліфікаційної роботи було досягнуто мінімальний час зчитування та обробки даних датчиків за максимально можливої точності та мінімальної кількості цифрових перешкод за допомогою загальнодоступних елементів Arduino.

В процесі розробки системи було виконано аналіз існуючих рішень, розглянуто теоретичну частину предметної області, проаналізовано функціональні та нефункціональні вимоги, спроектовано програмну систему та виконано її реалізацію, тестування та розгортання.При конкурентному аналізі були розглянуті деякі з найбільш відомих прототипів костюмів Motion Capture, з яких видно, що придбати такий продукт для комерційної організації середнього і нижчого рівня досить проблематично.

В теоретичній частині було розглянуто програмну та апаратну архітектуру костюму, з якими працює програмне забезпечення, а також поточний процес малювання 3D-моделі в Unity 3D. За вимогами до програмної системи та аналізу існуючих аналогів системи було складено діаграму варіантів використання системи та описано прецеденти.

При проектуванні системи було створено діаграми класів для всіх проектів системи для наочної візуалізації відношень між класами. Система була протестована за допомогою складання функціональних тестів та їх ручної перевірки.

У рамках дипломного проекту було проведено експерименти щодо вимірювання характеристик костюма та його програмного забезпечення для різних прототипів. У наслідок експериментів можна зробити висновок, що модернізація кібернетичного костюма для взаємодії з віртуальною реальністю дозволила підвищити точність даних по осях x, y та z для всіх кінцівок з 66% до 94,5%, а також знизити кількість перешкод з 40,59% до 5,5%.

Робота над проектом була здійснена завдяки підтримці лабораторії Gamehub Інституту комп’ютерних систем Державного університету «Одеська політехніка» та стипендіальній підтримці фонду Черновецького.

Результати кваліфікаційної роботи були опубліковані в матеріалах і тезах:

– 10-а міжнародна конференція студентів і молодих учених "Сучасні Інформаційні Технології 2020" Одеса, 14-15 травня 2020 року. Блажко О.А., Зибін Д.В. Кібернетичний костюм для взаємодії з віртуальною реальністю;

– 11-а міжнародна конференція студентів і молодих учених "Сучасні Інформаційні Технології 2021" Одеса, 14-15 травня 2021 року. Блажко О.А., Зибін Д.В. Кібернетичний костюм для взаємодії з віртуальною реальністю;

– XII Konferencji Horyzonty Nauki: Forum Prac Dyplomowych 2021, Blazhko A., Zybin D., Cybernetic Suit for Interaction with Virtual Reality;

– X Міжнародна науково-практична конференція. Рувінська В.М., Тройніна А.С., Зибін Д.В. Кібернетичний костюм для взаємодії з віртуальною реальністю / «Інформаційні та управляючі системи і технології (ІУСТ-ОДЕСА-2021).

СПИСОК ВИКОРИСТАННОЇ ЛІТЕРАТУРИ

1. Vicon, What is Motion Capture, 2018. URL: https://www.vicon.com/about-us/what-is-motion-capture.

2. Блажко О.А., Зибін Д.В., Кібернетичний костюм для взаємодії з віртуальною реальністю / «10-а міжнародна конференція студентів і молодих учених "Сучасні Інформаційні Технології 2021» Одеса. Матеріали 10-ї Міжнародної конференціі студентів і молодих учених, 14 - 15 травня 2020 р., Одеса – с. 76 - 77.

3. M. Dealessandri, що є кращим game engine: є Unity right for you?, 2021. URL: https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine- is-unity-the-right-game-engine-for-you.

4. Класифікація потреб по Симонову П. В. [Електроний ресурс] – Режим доступу: www / URL : https://psy.wikireading.ru/14534

5. TrackLab, Optitrack motive: tracker motion capture and 6 DOF object tracking, 2021. URL: https://tracklab.com.au/products/brands/optitrack/optitrack-motive-tracker/.

6. Ian Failes, A brief look at what’s available in mobcap glove tech, 2020. URL: https://beforesandafters.com/2020/09/17/a-brief-look-at-whats-available-in-mocap-glove-tech/.

7. GitHub, Gesture-MotionTracking, 2021. URL: https://github.com/keywish/Gesture-MotionTracking.

8. Arduino, What is Arduino?, 2018. URL: https://www.arduino.cc/en/Guide/Introduction

9. D. Roetenberg, H. Luinge, P. Slycke, Xsens MVN: Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors. Xsens Motion Technologies BV, Technical Report, 2013. URL: https://www.scirp.org/(S(czeh2tfqyw2orz553k1w0r45))/reference/ReferencesPapers.aspx?ReferenceID=2263664

10. Y. Park, J. Lee, J. Bae Development of a wearable sensing glove for measuring the motion of fingers using linear potentiometers and flexible wires, IEEE Trans. Ind. Inform., volume 11 Issue 1, 2015, pp 198-206, doi: 10.1109/TII.2014.2381932.

11. Romanyuk, O. N., Vyatkin, S. I. & Antoshchuk, S. G. “3d Vector Fields Visualization Using Graphics Processing Units”. Herald of Advanced Information Technology. Publ. Science і Technical. 2019; Vol.2 No.3:p.173–182. Odesa. Ukraine. DOI: https://doi.org/10.15276/hait.03.2019.1

12. Блажко О.А., Зибін Д.В., Кібернетичний костюм для взаємодії з віртуальною реальністю / «11-а міжнародна конференція студентів і молодих учених "Сучасні Інформаційні Технології 2021» Одеса. Матеріали 11-ї Міжнародної конференціі студентів і молодих учених, 14 - 15 травня 2021 р., Одеса – с. 189 - 190.

13. Рувінська В.М., Тройніна А.С., Зибін Д.В. Кібернетичний костюм для взаємодії з віртуальною реальністю / «Інформаційні та управляючі системи і технології (ІУСТ-ОДЕСА-2021). Матеріали X Міжнародної науково-практичної конференціі, 23 - 25 вересень 2021 р., Одеса – с. 41 - 44.

14. Larshin, V. P., Lishchenko, N. V., Babiychuk, O. B. & Piteľ, Ján. “Virtual Reality and Real Measurements in Physical Technology”. Applied Aspects of Information Technology. Publ. Nauka і Tekhnika. Odessa: Ukraine. 2021; Vol.4 No.1: 24–36. DOI: https://doi.org/10.15276/aait.01.2021.2

15. MPU-60X0 Обзор, URL: http://pro-interes.com/wp-content/uploads/2019/12/MPU-6050_rus.pdf

16. Кавіцька В.С. Конспект лекцій з дисципліни «Основи програмної інженерії» для студентів напряму 6.050103 – Програмна інженерія / Кавіцька В.С., Любченко В.В. – Одеса: ОНПУ,2016. – 87 с.

17. Поширеністю різних операційних систем в світі за 2020 рік, URL: https://ain.ua/2020/09/07/top-5-os-v-mire/

18. M. Menolotto, S. Dimitrios Komaris, S. Tedesco, B. O'Flynn, Motion Capture Technology in Industrial Applications: A Systematic Review, 2020. URL: https://www.researchgate.net/publication/344477681_Motion_Capture_Technology_in_Industrial_Applications_A_Systematic_Review

19. Dr. S. Patel, P. Talati, S. Gandhi, Design of I2C Protocol, 2019. URL: https://www.researchgate.net/publication/332142672_Design_of_I2C_Protocol

20. Керниган Б.В., Ричи Д.М., Мова С, URL: https://nsu.ru/xmlui/bitstream/handle/nsu/9058/kr.pdf

21. Unity, URL: https://nsu.ru/xmlui/bitstream/handle/nsu/9058/kr.pdf

22. Библиотека SoftwareSerial, URL: https://doc.arduino.ua/ru/prog/SoftwareSerial

23. Библиотека Wire, URL: https://doc.arduino.ua/ru/prog/Wire

24. Библиотека SPI для Arduino, URL: https://radioprog.ru/post/245

25. Що таке Unity 3D, URL: http://web.spt42.ru/index.php/chto-takoe-unity-3d

26. Цифровий осцилограф UNI-T UTD2025C, URL: https://sxema.com.ua/p641208636-tsifrovoj-ostsillograf-uni.html

27. Логічний аналізатор Saleae, URL: https://arduino.ua/prod651-logicheskii-analizator-saleae-8-kanalov

28. Arduino IDE, URL: https://arduino-ide.com