Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ____другий (магістерський)_____
(рівень вищої освіти)

Бездротовий контролер для керування обчислювальним пристроєм
(тема)

Виконав: студент 2 курсу, групи СКСм-20-2

_____Комаровський В.Е._____
(прізвище, ініціали)

Спеціальність *123 _Комп'ютерна інженерія*

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

Освітня програма_____

*Спеціалізовані комп'ютерні системи*_____
(повна назва освітньої прграми)

Керівник роботи _Литвинова Є.І.__
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти другий (магістерський)

Спеціальність 123_ Комп'ютерна інженерія

(шифр і назва)

Тип програми Освітньо-професійна

(освітньо-професійна або освітньо-наукова)

Освітня програма Спеціалізовані комп'ютерні системи

(повна назва

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«__25__»__03____ 2022 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Комаровському Володимиру Едуардовичу

(прізвище, ім'я, по батькові)

1. Тема роботи (проекту)_ Бездротовий контролер для керування обчислювальним пристроєм

Wireless Controller to Control a Computing Device

затверджена наказом по університету від «_24_»__03__2022 р. № _408 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20.05.2022

3. Вихідні дані до роботи (проекту)__функція – керування обчислювальним пристроєм; середовище розробки Arduino IDE;

4. Перелік питань, що потрібно опрацювати у роботі

Вступ. Аналіз предметної галузі та постановка задачі.

Опис використаних технологій. Реалізація поставлених задач. Висновки.

Перелік посилань. Додатки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 19 слайдів

6. Консультанти розділів роботи (проекту)

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

7. Дата видачі завдання 20.01.2022

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів курсового проекту (роботи)	Термін виконання етапів роботи	Примітка
1	Отримання завдання	20.01.2022 - 21.01.2022	
2	Огляд літератури за темою роботи	22.01.2022 - 01.02.2022	
3	Розробка структури контролера	01.02.2022 - 14.02.2022	
4	Розробка програмної реалізації контролера	15.02.2021 - 03.03.2022	
5	Розробка програмної реалізації комп'ютерного додатку	04.03.2021 - 23.03.2022	
6	Тестування програмної реалізації контролера	24.03.2021 - 10.04.2022	
7	Висновки	11.04.2021 - 12.04.2022	
8	Оформлення пояснювальної записки	13.04.2021 - 08.05.2022	

Студент _____
(підпис)

Керівник роботи (проекту) _____ Литвинова Є.І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 86 с., 23 рис., 2 табл., 15 джерел.

MPU-6050, HC-05, ГІРОСКОП, ІНЕРЦІЙНИЙ ТРЕКІНГ, ARDUINO.

Метою дослідження є зменшення вартості бездротового контролера за рахунок використання модуля MPU6050 та здійснення керування за допомогою плати Arduino.

Задачами дослідження атестаційної роботи є огляд архітектури системи інерційного трекінгу, принципу дії, властивостей та структури 3-осьового гіроскопу і 3-осьового акселерометру, розробка структури та програмної реалізації бездротового контролера на базі Arduino.

Результатом атестаційного проектування є бездротовий контролер, побудований на базі MPU6050 та Bluetooth модулю HC-05, що керується за допомогою плати Arduino Nano. За допомогою цього контролеру можна керувати курсором комп’ютера.

ABSTRACT

Course project: 86 pages, 23 figures, 2 tables, 15 sources.

MPU-6050, GYROSCOPE, INERTIAL TRAKING, ARDUINO.

The purpose of this work is to reduce the cost of a wireless controller by using the MPU6050 module with the Bluetooth module HC-05 and control this controller by the Arduino board.

The objectives of this project are to study the system of inertial tracking, the principle of operation, properties and structure of a 3-axis gyroscope and 3-axis accelerometer, development of the structure and software implementation of a wireless controller based on Arduino.

The result of the work is a wireless controller based on the MPU6050 and HC-05, controlled by an Arduino Nano board. You can use this controller to control the computer cursor.

ЗМІСТ

ВСТУП

В представленій атестаційній роботі розроблено комп'ютерну систему, яка представляє собою контролер для керування комп`ютером за допомогою інерційного трекінгу. Дана тематика є актуальною, оскільки пошук нових способів керування обчислювальним пристроєм дозволяє збільшити продуктивність та гнучкість спеціалізованих систем. Особливо враховуючи стрімкий ріст популярності систем Virtual Reality (VR), Augmented Reality (AR) та обладнання житла системами Smart House.

З постійним розвитком технологій, їх покращенням та зменшенням вартості, поступово користувачі стали відмовлятись від консервативних дротових систем керування комп’ютером (миші, клавіатури, геймпаду). Люди шукають найбільш зручний спосіб взаємодії з комп’ютером. Вони поступово намагаються позбутись обмежень старих моделей контролерів на користь гнучкості, свободи у керуванні, можливості пересуватись житлом, які дають нові типи контролерів. Особливо це стосується систем, призначених для керування комп’ютером під час ігрового процесу та систем, які потребують свободи руху користувача.

Саме тому в умовах загального переосмислення підходів до контролю обчислювальними пристроями, розробки нових стандартів та розвитку таких напрямків як VR та AR виникають великі можливості для розробки нових та покращення існуючих прототипів контролерів. Саме цьому тематика атестаційної роботи є актуальною на сьогоднішній день, а рішення може набути широкого застосування.

Метою дослідження є зменшення вартості бездротового контролера за рахунок використання модуля MPU6050 та здійснення керування за допомогою плати Arduino.

Задачами дослідження атестаційної роботи є огляд архітектури системи інерційного трекінгу, принципу дії, властивостей та структури 3-осьового гіроскопу і 3-осьового акселерометру, розробка структури та програмної реалізації бездротового контролера на базі Arduino.

1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Критерії вибору обладнання

Обладнання для виконання атестаційної роботи повинно відповідати наступним вимогам:

- надійність;
- швидкість роботи;
- гнучкість – тобто, можна змінювати прошивку приладу або налаштування швидко та легко;
- економна робота – автономної роботи пристрою повинно вистачати для комфортного використання без необхідності додаткового заряджання/зміни акумуляторів;
- компактність – в результаті прилад не повинен мати великих габаритів;
- мобільність – кінцева розробка з легкістю переміщується та не займає багато місця при зберіганні.

1.2 Опис апаратно-програмних засобів використаних у проекті

Для того, аби виконати завдання атестаційної роботи необхідні наступні компоненти:

- плата Arduino Nano;
- 3-х осьовий акселерометр і 3-х осьовий гіроскоп MPU-6050;
- система автономного живлення 5 Вольт;
- Bluetooth модуль HC-05;
- блок живлення;
- резистори опором 1 КОм та 2 КОм;

- дві кнопки;
- дроти;
- макетна плата 5 см на 7 см;
- PBS-40 роз'єм з штирковим кроком у 2.54 мм типу "мама";
- стрічка з липучкою.

Arduino - це невелика плата, що має власний процесор та пам’ять. На платі розміщені контакти, до яких підключаються різні компоненти, такі як датчики, світлодіоди, сенсори та інше. Є можливість завантажити у процесор плати програму, що буде керувати підключеними до неї компонентами. Таким чином, за допомогою Arduino можна створювати різноманітні пристрої. Вибір Arduino більш детально описано у главі 2.3 Вибір модулю.

Модулі з гіроскопом та акселерометром в більшості своїй використовуються для того, щоб визначити положення тіла, на якому вони розташовуються. Саме завдяки таким модулям телефони повертають зображення одночасно з поворотом телефону, ноутбуки міцніше фіксують жорсткий диск під час падіння, спеціальні стабілізатори утримують камеру для зйомки фільмів нерухомою в дуже активних сценах та інше.

Акселерометр в найпростішому вигляді - це закріплений на пружинці важіль, встановлений у власному корпусі. При струшуванні або повороті корпусу, важіль переміщається всередині нього по інерції. І оскільки при прискоренні у відповідному напрямку важіль рухається, він неминуче натягує за собою і пружинку, коливання якої можуть бути прийняті до уваги для визначення напрямку і прискорення зміни положення всього корпусу.

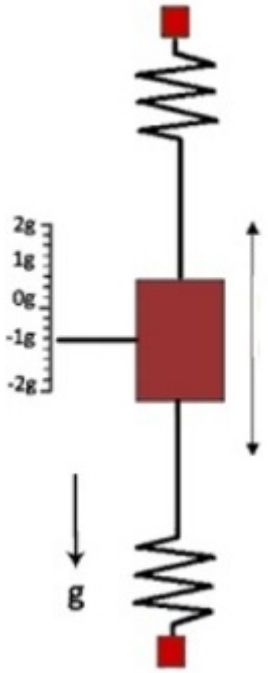

Рисунок 1.1 – Схема акселерометру у найпростішому вигляді

Так, три пружинки з важелями, встановлені вздовж трьох просторових осей, дозволяють отримати повне уявлення про напрям і значення прискорення щодо землі для того предмета, до якого вони прикріплені.

У смартфоні, наприклад, вся ця конструкція розміщена всередині маленької мікросхеми, і всі рухи «важелів» відбуваються всередині неї. Однак в результаті стає можливим визначення прискорення всього смартфона в просторі.

Гіроскоп діє зовсім інакше. Дзиґа, закріплена на осі в першій рамці, і обертається з високою швидкістю. Перша рамка має свободу обертання, будучи закріплена на осі (перпендикулярній осі вовчка) всередині другої рамки. Третя рамка несе на собі вісь обертання для другої рамки (перпендикулярну до осей обертання дзиґи і першої рамки).

У підсумку, як не повертай таку конструкцію, дзиґа буде прагнути зберегти вертикальне положення, хоча кільця і будуть повертатися.

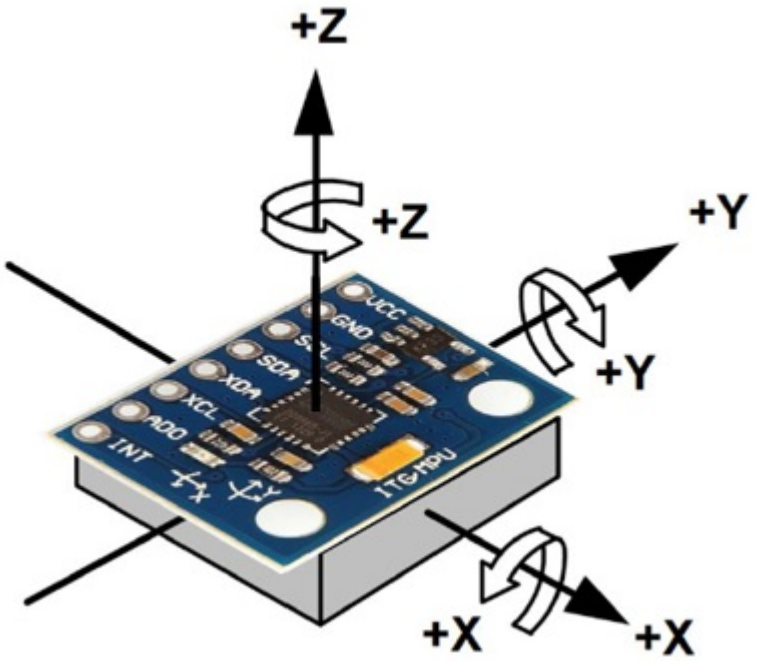

Рисунок 1.2 – Схема роботи гіроскопу

Для більш точного визначення параметрів переміщення предмета в просторі, гіроскоп зазвичай працює спільно з акселерометром, дозволяючи отримати повне уявлення про фізичний рух усього корпусу того чи іншого пристрою.

Для даної розробки вирішено використати модуль 3-х осьового акселерометра и 3-осьового гіроскопа MPU-6050 (рис. 1.3), оскільки даний модуль є достатньо доступним з огляду на його ціну та задовольняє критеріям, виставленим до нього.

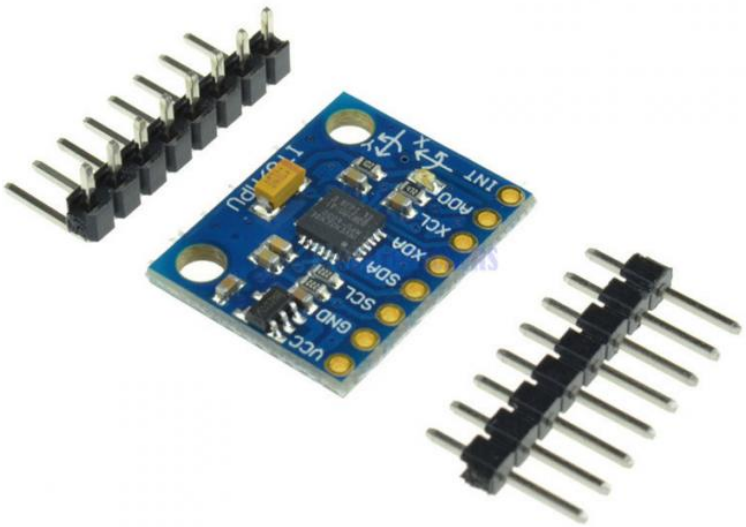

Рисунок 1.3 – Модуль MPU6050

Для того, щоб зробити бездротовий контролер, потрібно використати один з існуючих способів бездротової передачі даних. Для цього проекту буде використана технологія Bluetooth, яка є дуже розповсюдженою і яка відповідає всім запропонованим вимогам.

Контролер Arduino не підтримує бездротовий зв'язок за замовчуванням, а часто це просто необхідно. Наприклад, віддалене управління роботом на Arduino, відправка даних з метеостанції на Arduino в Інтернет або на домашній сервер, зв'язок кількох пристроїв між собою. Тут на допомогу розробникам пристроїв приходить безліч зовнішніх модулів для організації різних технологій бездротового зв'язку: модулі Wi-Fi, GSM / GPRS, IR, Bluetooth, радіо модулі для роботи в різних частотних діапазонах.

Технологія Bluetooth використовується для передачі даних між двома пристроями, які знаходяться на достатньо невеликій відстані один від одного, до того ж необов'язкова пряма видимість між ними. Вона забезпечує хорошу стійкість до широкосмугових перешкод, що дозволяє безлічі пристроїв, що знаходяться в одному місці, одночасно спілкуватися між собою, не заважаючи

один одному. Дуже широко дана технологія використовується в телефонах, планшетах, ноутбуках та в багатьох інших пристроях.

Одне з кращих рішень для організації двостороннього зв'язку через Bluetooth Arduino-пристрою з планшетом, ноутбуком або іншим Bluetooth-пристроєм - Bluetooth-модуль HC-05, який може працювати як master (здійснювати пошук Bluetooth-пристроїв і ініціювати установку зв'язку), так і як slave (ведений пристрій, той до якого підключаються).

Найчастіше HC-05 зустрічаються у вигляді двох спаяних плат. Верхня - заводська плата з мікросхемою BC417. Нижня - спеціальна плата для саморобних пристроїв, що містить найпотрібніші ніжки GPIO з кроком 2.54 мм, стабілізатор напруги і кнопку скидання. На рис. 1.4 зображені плата BC417 (зліва) та модуль HC-05 разом з підкладкою (зправа).

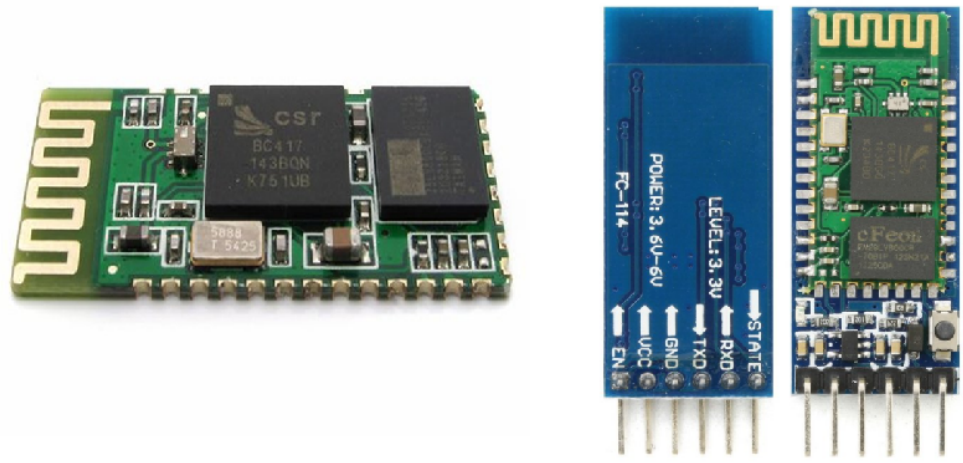

Рисунок 1.4 – Bluetooth модуль HC-05 без та з підкладкою

У модулю HC-05 є достатньо відомий аналог - HC-06. Але ці два модулі мають принципову відмінність. HC-05 відрізняється від HC-06 тим, що він може працювати в двох режимах роботи: ведений (slave) і ведучий (master). На відміну

від нього HC-06 працює тільки веденим, тобто він не здатний знаходити інші пристрої та самостійно встановлювати з ними зв'язок [10].

У даній атестаційній роботі Bluetooth буде працювати лише в slave режимі. Але сучасні бездротові пристрої, які працюють за допомогою технології Bluetooth, як правило, після включення самостійно шукають обчислювальний пристрій, з яким вони будуть взаємодіяти. Саме для можливості реалізації такої функціональності у майбутньому, був обраний модуль HC-05.

Bluetooth-модулі працюють з логічними рівнями напруги 3.3 В, тому використання п’яти-вольтової напруги може вивести їх з ладу. Але найчастіше на платі є все необхідне для узгодження рівнів напруги. Бувають ситуації, в яких між Arduino TXD - Bluetooth RXD необхідне узгодження напруги, тому використання дільника - гарантія надійної роботи.

Для створення найпростішого дільника використовуються резистори. Резистор (від латинського «resisto», що означає "пручаюся") - це пасивний елемент електричного кола, що володіє певним або змінним значенням електричного опору. На відміну від активних елементів, пасивні не мають можливості керувати потоком електронів.

Резистори відповідають за лінійне перетворення сили струму в напругу і навпаки, а також для обмеження струму і поглинання електричної енергії. Резистор є одним з найпопулярніших компонентів і використовується в більшості електронних пристроїв.

Резистори використовуються в якості:

- перетворювачів струму в напругу і навпаки;
- подільників напруги, ця властивість застосовується в вимірювальних апаратах;
- елементів для зменшення або повного видалення радіоперешкод.

Параметри, які потрібно враховувати при виборі резистора, залежать від характеру схеми, в якій він буде використаний. До основних характеристик відносяться:

- номінальний опір. Ця величина вимірюється в Ом, 1 кОм (1000 Ом), 1 МОм (1000 кОм), 1 ГОм (1000 МОм);

- максимальна потужність, що розсіюється - гранична потужність, яку здатний розсіювати елемент при довготривалому використанні. На схемах номінальну потужність розсіювання вказують тільки для потужних резисторів. Чим вище потужність, тим більше розмір деталі;

- клас точності. Визначає, на скільки фактична величина опору може відрізнятися від заявленої.

При необхідності приймають до уваги граничну робочу напругу, надмірний шум, стійкість до температури і вологи, коефіцієнт напруги. Якщо планується встановити деталь в апарат, що працює на високих і надвисоких частотах, враховують паразитну ємність і паразитну індуктивність. Ці величини повинні бути мінімальними [5].

Блок живлення — джерело живлення, призначене для забезпечення живлення електроприладу електричною енергією, відповідаючи вимогам його параметрів: напруги, струму, і т. д. шляхом перетворення енергії інших джерел живлення.

Блок живлення виконує наступні завдання:

- забезпечення передачі потужності — передача заданої потужності з найменшими втратами і дотриманням заданих характеристик на виході без шкоди для себе. Зазвичай потужність джерела живлення беруть з деяким запасом;

- стабілізація — напруга, струм та інші параметри на виході джерела живлення повинні лежати в певних межах, в залежності від його призначення, враховуючи вплив великої кількості дестабілізуючих факторів: зміни напруги на

вході, струму навантаження і т. д. Найчастіше необхідна стабілізація напруги на навантаженні, однак іноді (наприклад, для заряджання акумуляторів) необхідна стабілізація струму;

- захист — напруга, або струм навантаження у разі несправності (наприклад, короткого замикання) будь-яких кіл може перевищити допустимі межі і вивести електроприлад або саме джерело живлення з ладу. Також у багатьох випадках вимагається захист від проходження струму по небажаному шляху: наприклад, проходження струму через землю при дотику людини або стороннього предмета до струмоведучих частин;

- керування — може включати регулювання, включення / вимикання яких-небудь кіл, або джерела живлення в цілому. Може бути як безпосереднім (за допомогою органів управління на корпусі пристрою), так і дистанційним, а також програмним (забезпечення включення / вимикання, регулювання в заданий час або з настанням якихось подій);

- контроль — відображення параметрів на вході і на виході джерела живлення, включення / вимикання кіл, спрацьовування захисту.

Для живлення проекту буде використаний набір з трьох АА батарейок, розміщений у спеціальному батарейному відсіку, який допоможе полегшити процес вилучення старих та встановлення нових батарейок.

Для живлення плати Arduino необхідно, щоб джерело струму мало напругу приблизно 5В. Одна батарейка типу АА може служити джерелом струму з напругою 1.5В. Тобто для живлення плати потрібно використати три таких батарейки, підключених послідовно. При послідовному підключенні напруга буде складатися, тому три послідовно підключені батарейки дадуть напругу приблизно у 4.5В, чого буде достатньо для живлення розробленої системи.

Для того, щоб систему можна було повністю виключити, необхідно реалізувати можливість припинення подачі струму до системи. Для цього

необхідно використати перемикач, який би розривав електричне коло. Таким перемикачем була обрана кнопка для ліхтарика з фіксацією, представлена на рис. 1.5.

Рисунок 1.5 – Кнопка для ліхтарика з фіксацією

Всі компоненти мають бути розташовані на макетній платі. Але ця плата має бути не великого розміру, щоб зручно розміщуватись на зап’ясті користувача. Дослідним шляхом було вирішено, що розмір 50 мм на 70 мм буде найбільш зручним. Також ця плата повинна мати крок між пінами рівний 2,54 мм. При таких характеристиках плата буде мати 18 на 24 отвори, що достатньо для розміщення всіх компонентів контролеру. Для створення цього проекту буде використовуватись печатна макетна плата PCB-1L-BR-5X7 (рис. 1.6).

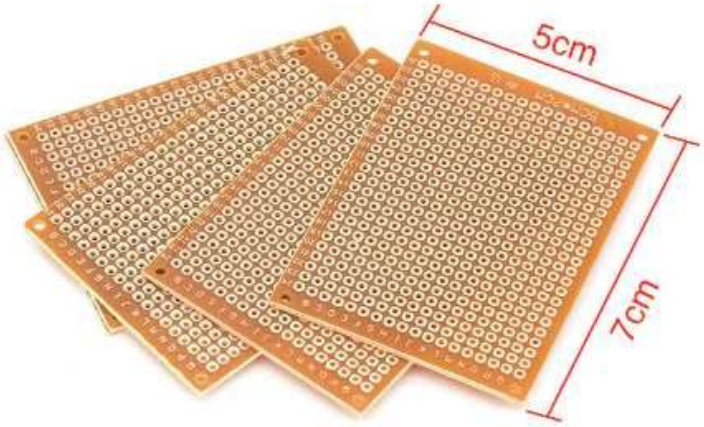

Рисунок 1.6 – Макетна плата

Для полегшення процесу монтажу та демонтажу компонентів буде використовуватись PBS-40 – штирковий роз'єм з кроком 2.54мм типу "мама" однорядний прямий на 40 пінів (рис. 1.7).

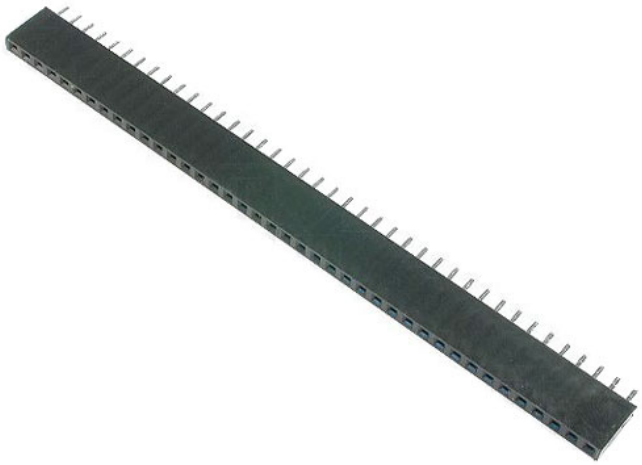

Рисунок 1.7 – Штирковий роз’єм PBS-40

Розроблений контролер має бути розміщеній на зап’ясті користувача, тому він повинен надійно фіксуватись та легко одягатись та зніматись. Для цього було вирішено використати стрічку з липучкою UGreen, яка представлена на рис. 1.8. Головними перевагами цієї стрічки є:

- вона не чіпляється до одягу;
- виконана з м’якого матеріалу, який не натирає шкіру зап’ястя під час тривалого використання контролеру;
- поверхня з’єднання розташована по всій площині стрічки, що дозволяє легко використовувати контролер всім людям, не зважаючи на розмір обхвату зап’ястя.

Рисунок 1.8 – Стрічка з липучкою UGreen

1.3 Практичне застосування

В результаті виконання атестаційної роботи було розроблено контролер для керування обчислювальним пристроєм, який може бути використаний в якості альтернативи комп’ютерної миші, джойстику або VR-контролеру. Також

розроблений пристрій матиме успіх в сфері керування телевізорами, які мають функцію Smart TV, або Android приставками. Це пов'язано з дуже вузьким спектром завдань, які повинен виконувати контролер для такої техніки: переміщення курсору та клік.

1.4 Постановка задачі

Задачами атестаційної роботи є дослідження системи інерційного трекінгу, принципу дії, властивостей та структури 3-осьового гіроскопу і 3-осьового акселерометру, представлених модулем MPU-6050, розробка структури та програмної реалізації бездротового контролера на базі Arduino, а також розробка додатку обчислювального пристрою для обробки даних контролера.

Задачі вирішуються за допомогою створення контролеру на основі вищенаведеного модуля MPU-6050 та Bluetooth модуля HC-05, а також написання функціонального програмного забезпечення для платформи Arduino та для комп’ютеру, за допомогою технології .Net Core Desktop Application. Створена система повинна відповідати всім зазначеним вимогам, працювати на власному джерелі живлення, передавати дані, необхідні для розрахунку переміщення курсору, за допомогою бездротових технологій.

Вирішення поставлених задач передбачає виконання таких дій:

- вибір необхідних компонентів;
- створення схеми макету;
- збирання макету згідно до схеми;
- розроблення програмного забезпечення для плати Arduino;
- розроблення програмного забезпечення комп’ютерного додатку.

2 ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

2.1 Вибір мови програмування

Для програмування Arduino використовується спрощена версія мови C++ з використанням бібліотек, які містять функції необхідні для роботі з платою.

C++ - універсальна мова програмування високого рівня з підтримкою декількох парадигм програмування. Зокрема: об’єктно-орієнтованої та процедурної парадигм.

При створенні C++ розробники прагнули зберегти сумісність з мовою C. Більшість програм на C правильно працюватимуть і з компілятором C++. C++ має синтаксис, заснований на синтаксисі C.

Нововведеннями C++ порівняно з C є:

- підтримка об'єктно-орієнтованого програмування через класи;
- підтримка узагальненого програмування через шаблони;
- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- простори імен;
- вбудовані функції;
- перевантаження операторів;
- перевантаження імен функцій;
- посилання і оператори управління вільно розподіленою пам'яттю.

Стандарт C++ на 1998 рік складається з двох основних частин: ядра мови і стандартної бібліотеки. Стандартна бібліотека C++ увібрала в себе бібліотеку шаблонів STL, що розроблялася одночасно із стандартом. Зараз назва STL

офіційно не вживається, проте в спільноті програмістів на C++ ця назва використовується для позначення частини стандартної бібліотеки, що містить визначення шаблонів контейнерів, ітераторів, алгоритмів і функторів.

Стандартизація визначила мову програмування C++, проте за цією назвою можуть ховатися також неповні, обмежені до стандарту варіанти мови. Спочатку мова розвивалася поза формальними рамками, спонтанно, у міру завдань, що ставилися перед нею. Розвиток мови супроводив розвиток кросс-компілятора Cfront. Нововведення в мові відбувалися в зміні номера версії кросс-компілятора. Ці номери версій кросс-компілятора розповсюджувалися і на саму мову, але стосовно теперішнього часу мову про версії мови C++ не ведуть [7].

Мова програмування пристроїв Arduino заснована на C++ і скомпонована з бібліотекою AVR Libc і дозволяє використовувати будь-які її функції. Разом з тим вона проста в освоєнні, і на даний момент Arduino - це, мабуть, найзручніший спосіб програмування пристроїв на мікроконтролерах.

Для написання програми, яка буде обробляти дані з контролеру, буде використовуватись мова програмування C#. Вона має багато вбудованих бібліотек та засобів, отримання даних через Bluetooth, їх обробки та перетворення на команди для курсора.

C# (вимовляється як "сі шарп") - сучасна об'єктно-орієнтована і типобезпечна мова програмування. C# дозволяє розробникам створювати різні типи безпечних та надійних програм, що виконуються в .NET. C# відноситься до широко відомого сімейства мов C, і буде виглядати дуже знайомим будь-кому, хто працював з C, C++, Java або JavaScript.

.NET – це безкоштовна платформа розробки з відкритим вихідним кодом для створення різних типів програм (наприклад, веб-додатків, мобільних додатків, ігор, IoT-додатків та інше). Платформа .NET підтримує кросплатформеність, тобто на цій платформі можна створювати додатки, які

будуть працювати на різних операційних системах (Windows, macOS, Linux, Android та інші) та при різних архітектурах процесорів (x64, x86, ARM32, ARM64).

.NET дозволяє використовувати спеціальні можливості платформи, такі як API операційної системи. Прикладами є Windows Forms та WPF у Windows та власні прив'язки до кожної мобільної платформи з Xamarin.

C# - це об'єктно- і компонентно-орієнтована мова програмування. C# надає мовні конструкції, які здійснюють безпосередню підтримку таку концепцію роботи. Завдяки цьому C# підходить для створення та застосування програмних компонентів. З моменту створення мова C# збагатилася функціями для підтримки нових більш жорстких робочих навантажень та сучасними рекомендаціями щодо розробки програмного забезпечення. C# - це об'єктно-орієнтована мова, тому саме розробник визначає типи та їх поведінку. Звісно, існує дуже велика кількість вже створених типів, яких достатньо для створення простих додатків (такі типи як Point, DateTime, Timer).

Мова програмування C# пропоную дуже багато функцій, які дозволяють створювати надійні та стійкі програми. Наприклад, Garbage Collector – це спеціальних механізм додатків, написаних на мові C#, який автоматично звільняє пам'ять, зайняту недоступними об'єктами, що не використовуються. Це дуже важливо для покращення працездатності програми, для того, щоб зменшити кількість споживаною пом’яті, а також позбутися витоків пам’яті. Це особливо помітно при порівнянні з мовою програмування C++, в якій уся відповідальність за резервування та звільнення пам’яті припадає на розробника.

Типи, що допускають значення null, забезпечують захист від змінних, які посилаються на видалені об'єкти. Обробка винятків надає структурований та розширюваний підхід до виявлення помилок та відновлення роботи додатків після їх обробки. Синтаксис LINQ створює загальний шаблон для роботи з

даними будь-якого джерела. Підтримка мов для асинхронних операцій забезпечує синтаксис для створення розподілених систем.

У C# діє єдина система типів. Всі типи C#, включаючи типи-примітиви, такі як int та double, успадковують від одного кореневого типу об'єкта. Всі типи використовують загальний набір операцій, а значення будь-якого типу можна зберігати, передавати та обробляти подібним чином. Більше того, C# підтримує як визначені користувачами типи-посилання, так і типи-значення. Також можна виділяти об'єкти динамічним чином та зберігати спрощені структури у стеку. Є підтримка універсальних методів та типів, що забезпечують підвищену безпеку типів та продуктивність [11].

2.2 Вибір середовища програмування

Для написання, налагодження і завантаження прошивок на плату Arduino необхідно встановити Arduino IDE (Integrated Development Environment). Порівняння Arduino IDE з іншими середами програмування наведена в таблиці 2.1.

Програмне забезпечення Arduino з відкритим кодом дозволяє легко писати код і завантажувати його на плату. Він працює в Windows, Mac OS X та Linux. Середовище написане за допомогою мови програмування Java та базується на процесорі Processing та іншому відкритому програмному забезпеченні. Це програмне забезпечення можна використовувати з будь-якою платою Arduino [8].

Таблиця 2.1 – Порівняння середовищ програмування

Назва IDE	Плюси середи	Мінуси середи
Arduino IDE	- проста у використанні; - підсвічування коду; - безкоштовна; - має вбудований програматор; - має багато корисних для розробки вбудованих функцій; - оптимізована для роботі з платами Arduino.	- hex-файли не зручно витягати з тимчасової папки; - необхідний певний bootloader, який буде розпізнавати плату підключену через USB як COM-порт; - деякі вбудовані функції мають складний інтерфейс; - підтримує не всі мікроконтролери Atmel.
Eclipse	- безкоштовна; - завдяки різноманітності плагінів розширюється додатковими функціями; - кросплотформенна; - покрокова компіляція і миттєва перевірка помилок; - зберігає і відновлює сесії.	- використовує багато системних ресурсів; - велика кількість вбудованих плагінів потребує більше часу на вивчення; - зустрічаються труднощі зі зворотною сумісністю; - складність конфігурацій; - іноді виникають проблеми з системами контролю версій.
Microsoft Visual Studio	- менше коду для написання; - інтуїтивний стиль кодування; - можливості відладки.	- складний у вивченні; - використовує багато системних ресурсів; - відсутній вбудований програматор.
Atmel Studio	- підсвічування синтаксису; - багато плагінів для полегшення процесу розробки; - підтримка зовнішніх програматорів.	- напрямок портів; - налаштовується безпосередньо регістрами мікроконтролерів; - незручний симулятор виконання коду; - потребує багато часу для вивчення інтерфейсу; - відсутня покрокова відладка коду.

Інтерфейс середовища розробки Arduino містить наступні основні елементи: текстовий редактор для написання коду, область для виведення повідомлень, текстова консоль, панель інструментів з традиційними кнопками і головне меню. Дане програмне забезпечення дозволяє комп'ютеру взаємодіяти з Arduino як для передачі даних, так і для прошивки коду в контролер.

Програми, що створюються в середовищі розробки Arduino, іноді ще називають скетчами. Скетчі пишуться в текстовому редакторі і зберігаються в файлах з розширенням .ino. Вбудований текстовий редактор має стандартні інструменти копіювання, вставки, пошуку і заміни тексту. Область повідомлень у вікні програми є, свого роду, зворотним зв'язком для користувача і інформує його про події (в тому числі і про помилки), що виникають в процесі запису або експорту написаного коду на плату. Консоль відображає у вигляді тексту потік вихідних даних середовища Arduino, включаючи всі повідомлення про помилки та ін. Відображає системну інформацію. У нижньому правому куті вікна програми показується модель поточної плати і послідовний порт (serial port), до якого вона підключена. Кнопки на панелі інструментів призначені для створення, відкриття, збереження і прошивки програм в пристрій. Окрема кнопка запускає програму SerialMonitor [9].

Для розробки програмного забезпечення для комп’ютеру також існує багато різних IDE. Безумовними лідерами серед них є Microsoft Visual Studio та JetBrains Rider, але дехто воліє використовувати інші IDE, тому що вони більш зручні для виконання спеціальних завдань (наприклад, Microsoft Visual Studio Code).

Microsoft Visual Studio (VS) має всі необхідні інструменти для розробки повноцінного додатку. Він має приємний інтерфейс, який після деякого часу

дослідження стає достатньо зручним. До того ж його можна налаштувати власноруч в залежності від потреб та вподобань користувача.

Важливим для програміста та розробки в цілому є наявність в VS засобів підвищення продуктивності (хвилясті підкреслення в місці помилки, очищення коду, IntelliSense та інше). Вони дозволяють передбачити помилки та прискорити процес написання програмного коду. Також ця IDE має вбудовані інструменті збирання та налагодження коду (debugging), налаштування графічного інтерфейсу мобільних та комп'ютерних додатків, взаємодії з базами даних різного типу.

Тривалий час Microsoft Visual Studio була безумовним лідером серед IDE для розробки комп'ютерних додатків на Windows. Але все змінилось, коли на світ з'явилась IDE від компанії JetBrains – Rider. Вона мала ряд переваг, які будуть розглянуті нижче.

Rider на відміну від VS, не сконцентрувався лише на 32-бітних процесах. Навіть якщо у Rider є процеси, доступні тільки для back-end, наприклад, SWEA (Solution-Wide Analysis), створення коду буде проходити гладко без будь-яких пауз чи збоїв. І як зазначає більшість користувачів, що працювали з Visual Studio і Rider, останній працює набагато стабільніше і швидше.

JetBrains Rider є кросплатформним, він може працювати на платформах Windows, Mac або Linux з однаковою функціональністю та стабільністю. Visual Studio працює переважно на платформі Windows. І якщо є необхідність перейти на Linux або Mac, необхідно буде набувати додаткових рішень: Visual Studio Code (для Linux) і Visual Studio для Mac. Головним недоліком є те, що версії Visual Studio для Mac і Linux мають різний функціонал і зовнішній вигляд до якого доведеться звикати. Rider, як зовні так і за своїми функціями, однаковий на всіх платформах, тому, якщо користувач вирішить перейти з Windows на Mac або

Linux, він отримає вже звичне середовище розробки і не витрачатиме дорогоцінний час на навчання роботі з IDE.

Середовище Rider включає більшість функцій популярного розширення Visual Studio для розробників .NET - ReSharper. У складі Rider є значний набір для рефакторингу, перевірки коду і контекстних дій для всіх мов і технологій, що підтримуються ним. Visual Studio також має набори для рефакторингу та перевірки помилок коду, але набагато більш обмежений, ніж ті, що надані в Rider і ReSharper.

У Rider є безліч функцій, успадкованих від платформи IntelliJ. Підтримка систем контролю версій: окрім Git та Mercurial, Rider працює з CVS та Subversion. Інтеграція VSTS доступна через спеціальний плагін, який підтримує компанія Microsoft. Rider (за допомогою DataGrip) підтримує підключення до баз даних і SQL. Користувачам Visual Studio у більшості випадків потрібно буде використовувати ODBC.

Підтримка можливостей для front-end розробки з використанням JavaScript, TypeScript, CSS, HTML, LESS, Sass тощо. доступна в Rider завдяки тому, що продукт включає у своєму складі функції спеціалізованого ПЗ для веб-розробки – JetBrains WebStorm.

У середовищі розробки Rider також є можливість інтеграції з багатьма трекерами завдань, такими як Team Foundation Server і Visual Studio Team Services. Також він підтримує JIRA Software, YouTrack та інші рішення, а також велику кількість високоякісних спеціалізованих плагінів, розроблених для IntelliJ та ReSharper, більшість з яких безкоштовні. Visual Studio теж підтримує різні плагіни, але безкоштовними є одиниці [12].

Головними недоліками IDE Rider є те, що вона не безкоштовна і працює лише з мовами програмування платформи .Net (C#, F#, VB.NET). Але ці проблеми не є суттєвими в межах цієї атестаційної роботи, тому що програмний

код буде створюватись саме на мові програмування C#, а для студентів компанія JetBrains пропонує спеціальну безкоштовну підписку.

2.3 Вибір модулю

Arduino - це апаратно-обчислювальна платформа, яка дозволяє будь-якій людині створювати різноманітні електро-механічні пристрої. Arduino складається з програмної і апаратної частини. Програмна частина включає в себе середовище розробки (програма для написання і налагодження прошивок), безліч готових і зручних бібліотек, спрощена мова програмування. Апаратна частина включає в себе велику лінійку мікроконтролерів і готових модулів для них [6].

Вибирати модель використовуваної плати в середовищі Arduino необхідно з двох причин:

- щоб задовольнити параметрам, які використовуються під час компіляції і прошивки скетчів (такі, як тактова частота, бодрейт і ін.);
- щоб визначити установки фьюз-бітів, які використовуються під час прошивки завантажувача в контролер плати.

Головними критеріями вибору модулю для виконання проекту виступають:

- плата має бути невеликого розміру, щоб не робити загальний розмір проекту занадто великим;
- плата повинна мати достатньо внутрішньої пам’яті для збереження коду програми та мати додатковий запас пам’яті, для того щоб функціонал можна було розширяти без заміни модулю.

2.3.1 Основні види модулів

Для того, аби обрати модуль, що стане основою для розробки, було проведено аналіз основних плат Arduino. Далі наведено опис найпопулярніших плат, серед яких було обрано ту, що задовольняє потребам розроблюваного пристрою.

Uno – найпопулярніша версія базової платформи Arduino USB. Uno має стандартний порт USB. Arduino Uno багато в чому схожа з Duemilanove, але має новий чіп ATMega8U2 для послідовного підключення по USB і нове, більш зручне маркування входів / виходів. Платформа може бути доповнена платами розширення (призначеними для користувача платами з різними функціями).

Nano – це компактна платформа, яка використовується як макет. Nano підключається до комп'ютера за допомогою кабелю USB Mini-B.

Mega ADK – версія плати Mega 2560 з підтримкою USB host інтерфейсу для зв'язку з телефонами на Android і іншими пристроями з USB інтерфейсом.

Mini - найменша платформа Arduino. Прекрасно працює як макетна модель, або, в проектах, де простір є критичним параметром. Платформа підключається до комп'ютера за допомогою адаптера Mini USB.

Pro - платформа, розроблена для досвідчених користувачів, може бути частиною більшого проекту. Вона може живитися від акумуляторної батареї, але в той же час вимагає додаткової збірки і компонентів.

2.3.2 Опис модулю Arduino Nano

Для виконання завдання, поставленого в даній атестаційній роботі була обрана Arduino Nano. Її невеличкий розмір (1.85 см х 4.2 см) з легкістю встановлюється в корпуси невеликого розміру, не заважаючи розміщенню інших компонентів. Пам’яті Nano достатньо для реалізації проекту, особливо якщо вона побудована на мікроконтролері ATmega328.

Незважаючи на невеликий розмір Arduino Nano не поступається характеристиками Arduino UNO, яка э класичним представником лінійки Arduino, і яка частіше за все використовується у проектах початківців. Nano за деякими параметрами навіть перевершує UNO (наприклад, наявність аналогових пінів A6 і A7). Також слід відзначити, що ця плата дуже бюджетна. Вона має найкраще співвідношення вартості до якості. Таким чином вона робить ціну розробки мінімальною.

Платформа Nano (рис. 2.1), побудована на мікроконтролері ATmega328 (Arduino Nano 3.0) або ATmega168 (Arduino Nano 2.x), має невеликі розміри і може використовуватися для виконання різноманітних задач. Вона має схожу з Arduino Duemilanove функціональність, проте відрізняється складовими частинами. Відмінність полягає у відсутності силового роз'єму постійного струму і роботі через кабель Mini-B USB. Nano розроблена і продається компанією Gravitech.

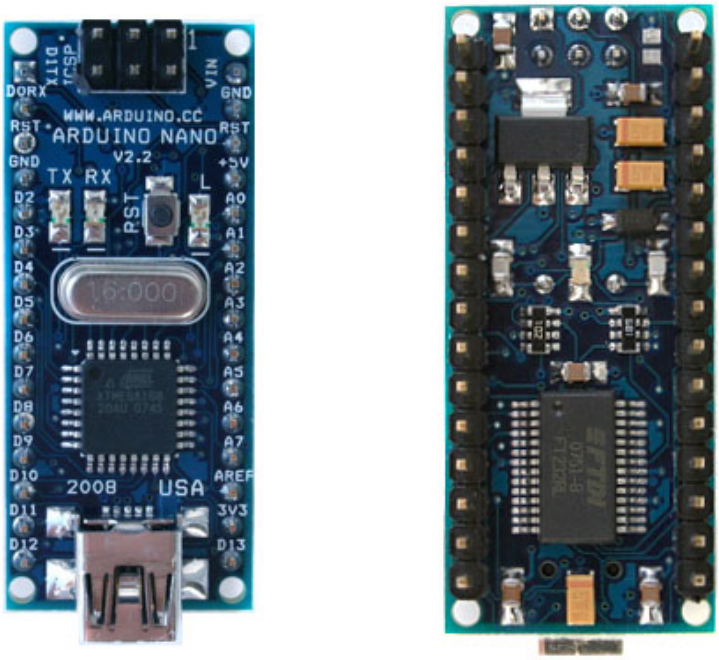

Рисунок 2.1 – Arduino Nano

Arduino Nano може отримувати живлення через підключення Mini-B USB, або від нерегульованого 6-20 В (вивід 30), або регульованого 5 В (вивід 27), зовнішнього джерела живлення. Автоматично вибирається джерело з найвищим напругою.

Мікросхема FTDI FT232RL отримує живлення, тільки якщо сама платформа має живлення від USB. Таким чином при роботі від зовнішнього джерела (не USB), буде відсутня напруга 3.3 В, що генерується мікросхемою FTDI, при цьому світлодіоди RX і TX блимають тільки при наявності сигналу високого рівня на виводах 0 і 1.

Мікроконтролер ATmega168 має 16 кБ флеш-пам'яті для зберігання коду програми, а мікроконтролер ATmega328, в свою чергу, має 32 кБ (в обох випадках 2 кБ використовується для зберігання завантажувача). ATmega168 має 1 кБ ОЗУ і 512 байт EEPROM, а ATmega328 - 2 кБ ОЗУ і 1 Кб EEPROM.

Таблиця 2.2 – Характеристики модулю

Мікроконтролер	Atmel ATmega168 або ATmega328
Робоча напруга (логічний рівень)	5 В
Вхідна напруга (рекомендована)	7-12 В
Вхідна напруга (гранична)	6-20 В
Цифрові Входи/Виходи	14 (6 з котрих можуть бути використані в якості виходів ШИМ)
Аналогові входи	8
Постійний струм через вхід/вихід	40 мА
Флеш-пам’ять	16 Кб (ATmega168) або 32 Кб (ATmega328) при цьому 2 Кб використовуються для завантажувача
ОЗУ	1 Кб (ATmega168) або 2 Кб (ATmega328)
EEPROM	512 байт (ATmega168) або1 Кб (ATmega328)
Тактова частота	16 МГц
Розміри	1.85 см x 4.2 см

Кожен з 14 цифрових виводів Nano може налаштовуватися як вхід або вихід. Виводи працюють при напрузі 5В. Кожен вивід має навантажувальний резистор (стандартно відключений) 20-50 кОм і може пропускати до 40 мА.

На платформі Nano встановлені 8 аналогових входів, кожен розрядністю 10 біт (тобто може приймати тисячі двадцять чотири різних значення). Стандартно виводи мають діапазон вимірювання до 5 В щодо землі, проте є можливість змінити верхню межу за допомогою функції analogReference ().

Контролери Arduino Nano активно використовуються в найрізноманітніших проектах. Використання мініатюрного контролера дозволяє створювати пристрої в невеликому форм-факторі, що є важливим для проектів в області автоматизації і робототехніки. Ця плата досить компактна, зручна і має всі можливості "великої UNO".

3 РЕАЛІЗАЦІЯ ПОСТАНОВЛЕНИХ ЗАДАЧ

3.1 Реалізація фізичної моделі

Схему фізичної моделі показано на рис. 3.1.

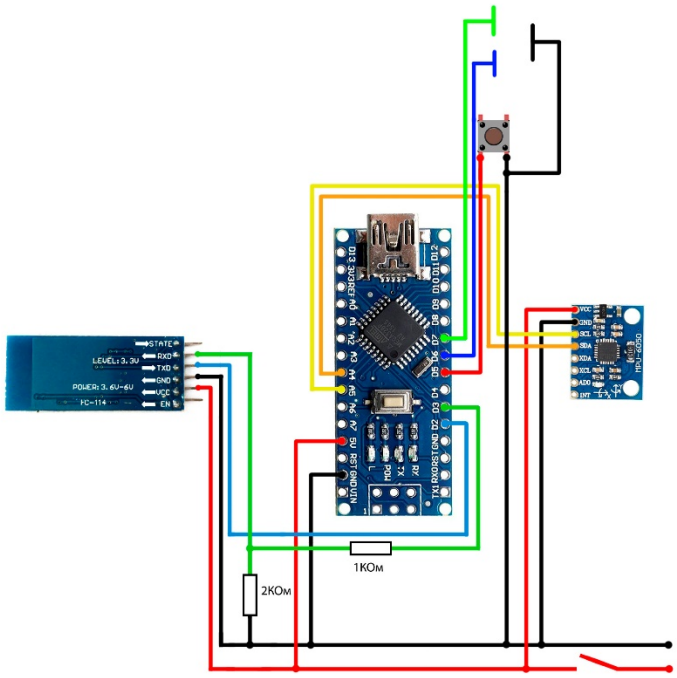

Рисунок 3.1 – Схема фізичної моделі

В даній фізичній моделі використовуються апаратні засоби, що були описані раніше, а саме:

- модуль Arduino Nano;
- резистори на 1000 та 2000 Ом;
- Bluetooth модуль HC-05;
- модуль MPU6050;
- кнопка Reset;

- кнопка-перемикач живлення;
- кнопки для моделювання кнопок миші;
- стрічка з липучкою UGreen;
- PBS-40 роз'єм з штирковий крок 2.54мм типу "мама" однорядний прямий;
- кейс для батарейок;
- макетна плата;
- з'єднуючі дроти.

На схемі також показано спосіб з’єднання, який було реалізоване на практиці. Реалізовану модель без кнопок показано на рис. 3.2.

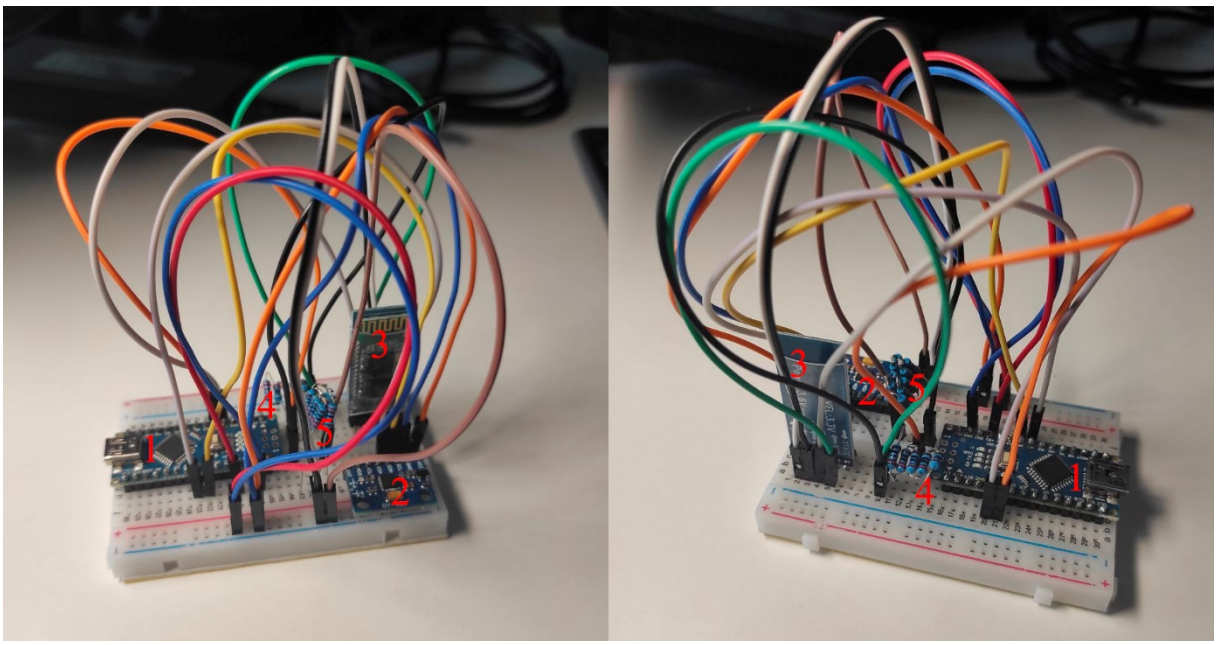

Рисунок 3.2 - Макет атестаційної роботи

На рис. 3.2 зображений макет атестаційної роботи, зібраний на базі макетної плати, на якому цифрами визначені всі елементі системи. Для збірки проекту використовувалась макетна плата через те, що вона дозволяю поєднати

компоненти системи без пайки, що дозволяє швидко робити зміни в конструкції, заміну та перевірку компонентів, а також передбачити, який розмір буде у вихідного макету.

На рисунку Arduino Nano позначена цифрою 1. Вона розміщена з краю таким чином, щоб був вільний доступ до MicroUSB роз'єму. Завдяки цьому роз'єму можна змінювати прошивку Arduino Nano. Саме з цієї причини було прийняте рішення про таке положення плати. Цифрою 2 позначений модуль MPU-6050. Він розміщений попереду плати Arduino Nano на вільному місці.

Bluetooth модуль HC-05 відмічений на рисунку 3.2 під номером 3. Він розташований на макетній платі вертикально, що допомагає трохи покращити передачу сигналу та покращити менеджмент простору. Також для підвищення стабільності роботи модулю та системи в цілому був використаний дільник напруги. Він збудований на базі двох резисторів, відмічених цифрами 4 та 5 на рисунку. Резистор під номером 4 має опір 1000 Ом, а під номером 5 - 2000 Ом [3].

Через нестачу резисторів з зазначеними показниками опору, було прийнято рішення, замінити резистори з великим опором декількома послідовно з'єднаними резисторами з меншим опором. Таким чином при створенні макету атестаційної роботи були використані резистори з номінальним опором 220 Ом. З’єднавши їх в послідовні блоки з п’яти та десяти резисторів були відтворені резистори з опором приблизно рівним 1000 Ом та 2000 Ом відповідно. Звістно, отримані резистори мають значення опору трохи вище за необхідний: 1100 Ом для першого та 2200 Ом для другого блоків (фактично, це значення трохи менше, через фабричну похибку опору використаних резисторів). Але таке відхилення від необхідного значення не є критичним і не є достатнім приводом для того, щоб не використовувати їх в рамках цього проекту.

Використання макетної плати без пайки у кінцевому вигляді апаратної частини проекту не є можливим через те, що всі дроти можуть бути розташовані

тільки над платою, що робить пристрій необґрунтовано великим та незручним. А це в свою чергу суперечить одній з встановлених вище вимог до кінцевого вигляду апаратної частини. З огляду на цей факт було вирішено зібрати проект на базі односторонній макетній платі.

Перед пайкою компонентів на макетну плату потрібно разрахувати оптимальне положення всіх компонентів та дротів на платі. Така підготовка необхідна для того, щоб дроти не накладались один на інший під час пайки. Це може сильно ускладнити процес пайки дротів та компонентів, а також їх демонтаж (при необхідності заміти зламану частину системи). Для вирішення цієї задачі була використана програма Fritzing.

Програма Fritzing має велику бібліотеку радіоелектронних компонентів. Кожен з них можна поєднувати з іншими компонентами на макетній платі або навіть без неї. Ця програма пропоную зручний інтерфейс для створення електронного макету майбутнього пристрою. Вона допомагає встановлювати компоненті на макетній платі згідно з існуючими на них отворами для контактних пінів.

Макет апаратної частини атестаційної роботи без кнопок наведений на рис. 3.3. Нажаль, у компонентній базі програми не знайшлось кнопки живлення, яка була використана у реальному проекті, тому на макеті вона замінена звичайною чотирипіновою кнопкою. Також потрібно зазначити, що представлений макет є не повним, бо на ньому не відображені дроти живлення, які відходять від кожного компоненту. Причиною цьому є те, що у програмі не має необхідного компоненту джерела живлення, а програма не дає можливість створити дріт, який одним кінцем прикріплений до компоненту, а іншим до отвору у макетній платі. До того ж велика кількість додаткових дротів може зробити макет незрозумілим.

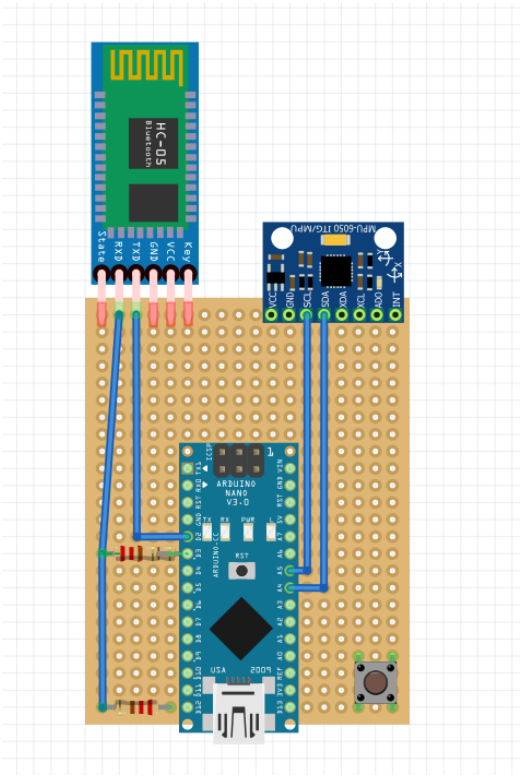

Рисунок 3.3 - Макет апаратної частини розроблений у програмі Fritzing

При створенні реального пристрою по створеному макету було прийнято рішення, припаювати компоненти не безпосередньо до макетної плати, а за допомогою додаткового прошарку з PBS-40. PBS-40 - цю однорядний прямий роз'єм штирковий крок з відстанню між пінами 2.54мм. Таким чином процес пайки стає безпечним, бо зникає можливість перегріти компоненти під час монтажу. Також така організація апаратної частини дозволить з легкістю змінювати компоненти пристрою, якщо вони вийдуть з ладу, або при покращенні пристрою шляхом використання нових компонентів.

Зовнішній вигляд макетної плати з припаяними компонентами та дротами зображений на рис. 3.4 (верхня частина) та рис. 3.5 (нижня частина).

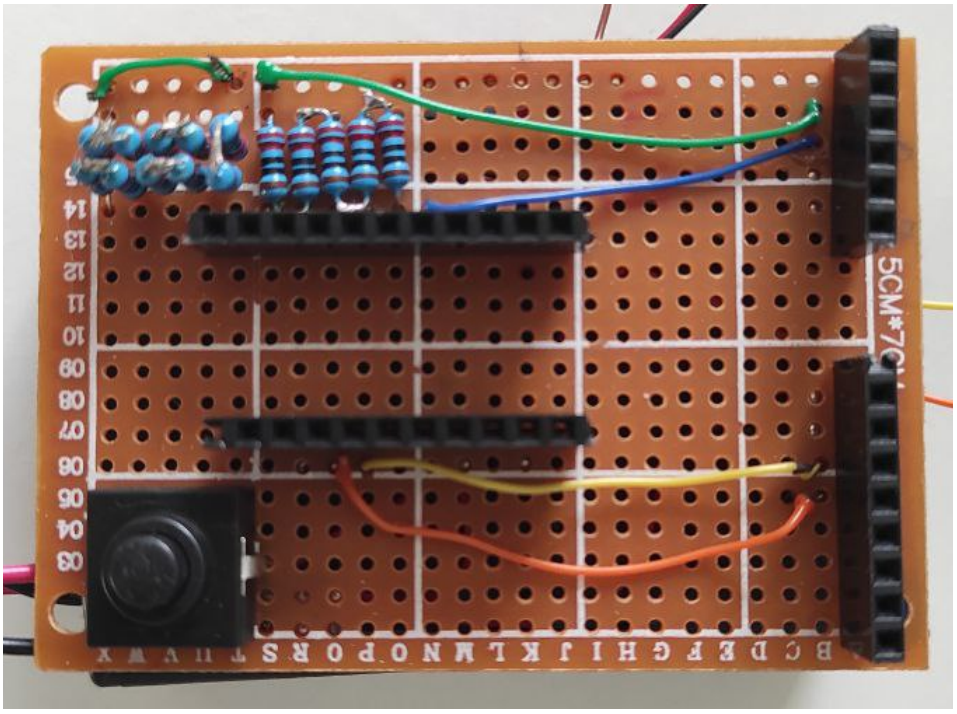

Рисунок 3.4 – Верхня частина макетної плати з припаяними компонентами

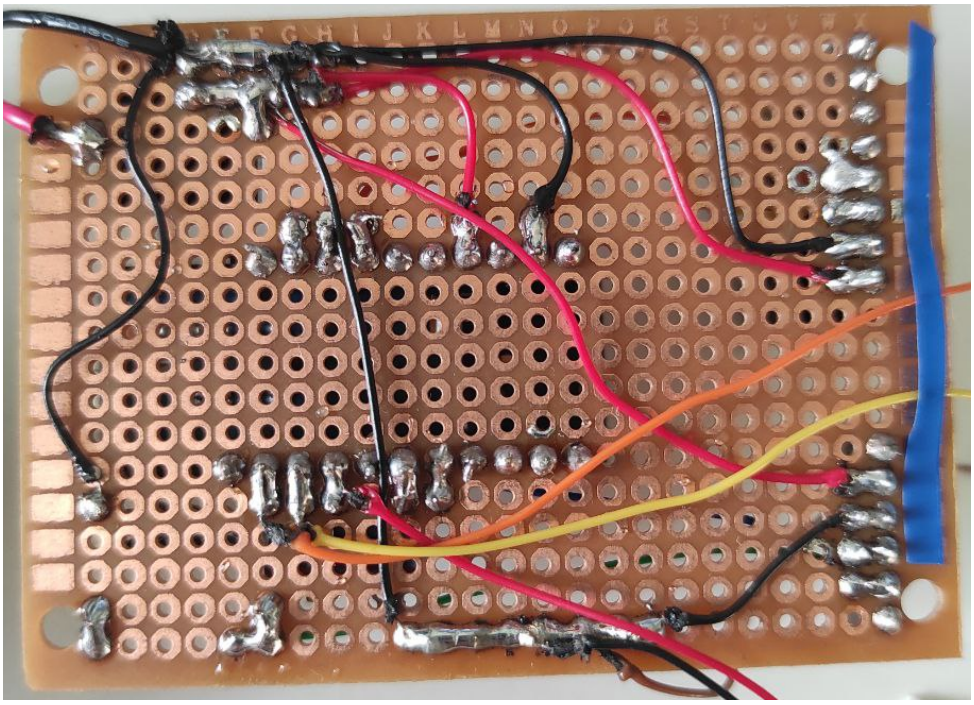

Рисунок 3.4 – Верхня частина макетної плати з припаяними компонентами

Для покращення сприйняття схеми кольори дротів для підключення Bluetooth та MPU-6050 модулів співпадають з кольорами дротів на схемі рис. 3.2. До того ж усі дроти живлення мають красний та чорний колір (5 V та GND відповідно).

Кінцевий вид зібраного пристрою наведений на рис. 3.5. Під зібраною макетною платою розташований блок живлення, а попереду плати 4 наліпки на пальці, які виконують роль двох кнопок керування, та кнопки reset.

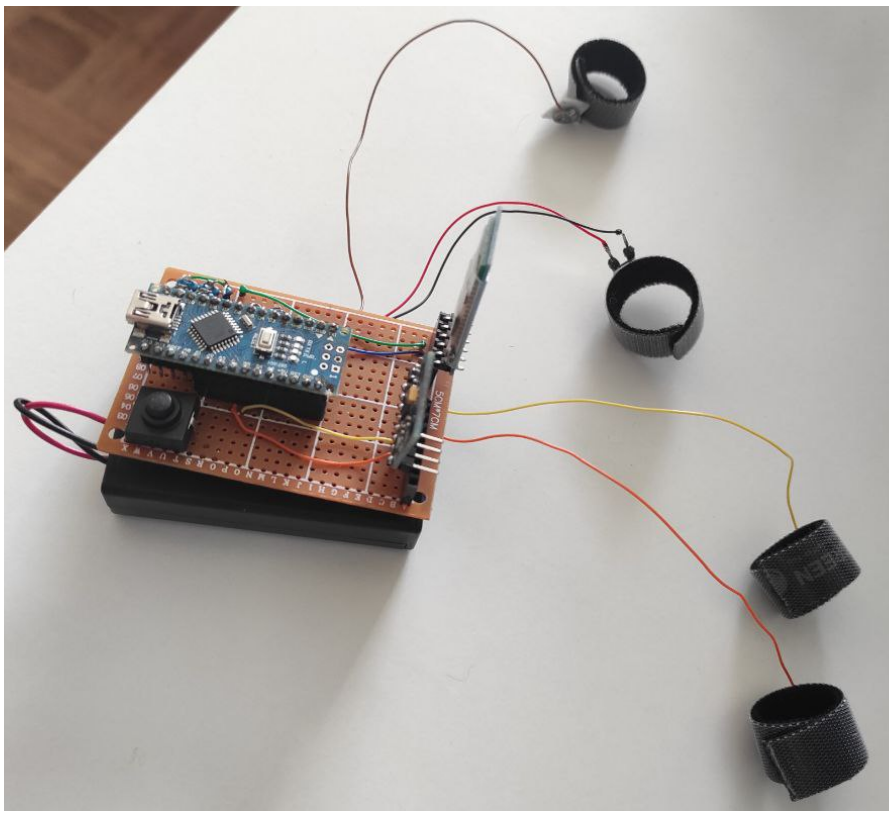

Рисунок 3.6 – Зовнішній вигляд зібраного проекту

На рис. 3.7 представлено те, як пристрій повинен розміщуватись та кріпитись на зап'ясті руки. Спеціальною наліпкою головна частина пристрою кріпиться до долоні, а кнопки наліплюються до пальців.

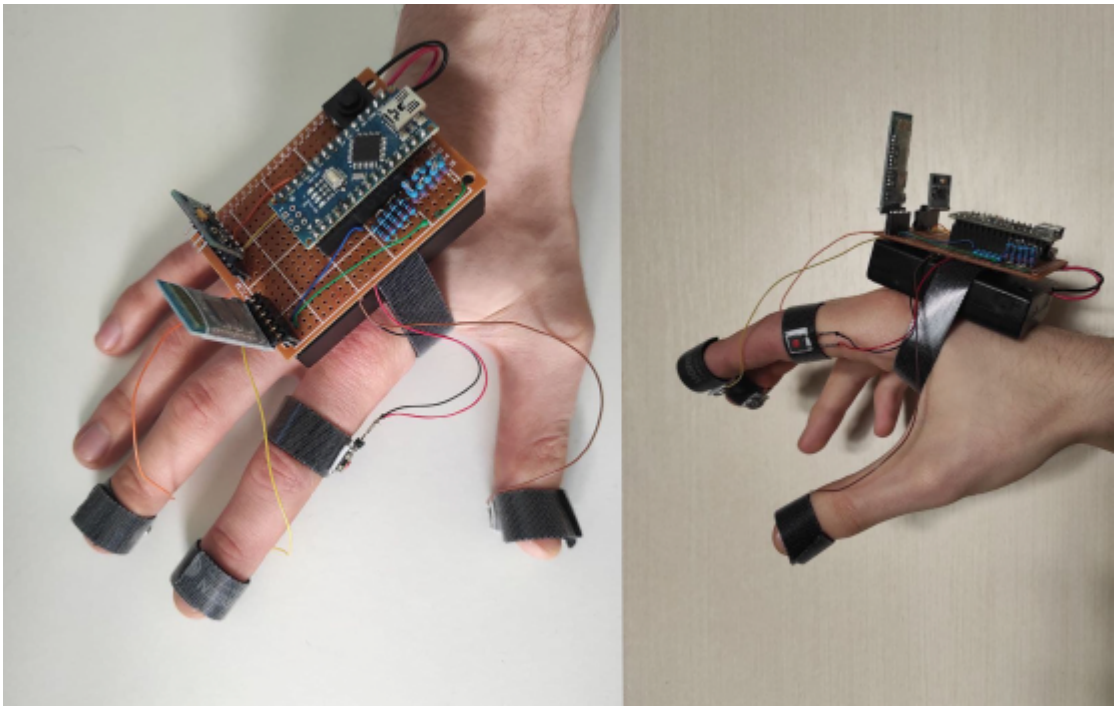

Рисунок 3.7 – Зовнішній вигляд вдягненого пристрою

3.2 Реалізація програмного коду для Arduino

Реалізацію програмного коду можна умовно поділити на 3 частини: написання алгоритму для зчитування даних з гіроскопу та акселерометру, обробки натиску кнопок та написання алгоритму для передачі даних через Bluetooth модуль на обчислювальний пристрій. Під час розробки обчислювальним пристроєм виступав персональний комп'ютер під операційною системою Windows 10, але сигнал може приймати будь-який пристрій та інтерпретувати його так, як потрібно для виконання його завдань.

Модуль MPU-6050 достатньо складний у використанні. Перш ніж почати його використовувати потрібно його налаштувати. Річ у тому, що кожен модуль MPU-6050 має деяку похибку в розрахунках, отриману як результат не досконалого виробництва. Це так би мовити плата за те, що модуль дуже дешевий та доступний. Для вирішення цієї проблеми є багато способів, але в програмній

реалізації цьєї атестаційної роботи був використаний один з найпростіших способів – розрахунок середньої похибки. Такий вибір обумовлений тим, що Arduino Nano має обмеження в пам'яті та ресурсі центрального процесору. Тим не менш цей спосіб визначає похибку з точністю, яка повністю задовольняє потреби проекту.

Цей спосіб полягає в находженні середнього значення похибки шляхом багаторазового повторення зчитування даних з гіроскопу та акселерометру у нульовій позиції та нерухомості (в такій позиції всі показники повинні дорівнювати нулю). Програмна реалізація описаного алгоритму наведена в прикладі 3.1.

```
      void calculate_IMU_error(){
      while (c < 200) {
      Wire.beginTransmission(MPU);
      Wire.write(0x3B);
      Wire.endTransmission(false);
      Wire.requestFrom(MPU, 6, true);
      AccX = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
      AccY = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
      AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
      AccErrorX = AccErrorX + ((atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ),
2))) * 180 / PI));
      AccErrorY = AccErrorY + ((atan(-1 * (AccX) / sqrt(pow((AccY), 2) +
pow((AccZ), 2))) * 180 / PI));
      c++;
      }
      AccErrorX = AccErrorX / 200;
      AccErrorY = AccErrorY / 200;
      c = 0;
      while (c < 200) {
      Wire.beginTransmission(MPU);
      Wire.write(0x43);
      Wire.endTransmission(false);
      Wire.requestFrom(MPU, 6, true);
      GyroX = Wire.read() << 8 | Wire.read();
```

```
GyroY = Wire.read() << 8 | Wire.read();
GyroZ = Wire.read() << 8 | Wire.read();
GyroErrorX = GyroErrorX + (GyroX / 131.0);
GyroErrorY = GyroErrorY + (GyroY / 131.0);
GyroErrorZ = GyroErrorZ + (GyroZ / 131.0);
c++;
}
GyroErrorX = GyroErrorX / 200;
GyroErrorY = GyroErrorY / 200;
GyroErrorZ = GyroErrorZ / 200;
Serial.print("AccErrorX: ");
Serial.println(AccErrorX);
Serial.print("AccErrorY: ");
Serial.println(AccErrorY);
Serial.print("GyroErrorX: ");
Serial.println(GyroErrorX);
Serial.print("GyroErrorY: ");
Serial.println(GyroErrorY);
Serial.print("GyroErrorZ: ");
Serial.println(GyroErrorZ);
}
```

Приклад 3.1 – Програмний код реалізації алгоритму пошуку похибки розрахунків модулю MPU6050

Наведений програмний код виконується в два етапи: спочатку для акселерометру, потім для гіроскопу. Після розрахунку похибок, вони виводяться у serial console. Цю консоль можна відкрити у Android IDE. Це дуже розповсюджений спосіб передачі логів системи у реальному часі для Arduino. Але для того, щоб побачити ці дані, плата Arduino Nano повинна бути підключена до комп'ютеру.

Для того, щоб не робити розрахунки похибки при кожному старті системи, значення отримані при перших розрахунках вносяться до коду і більше не змінюються. Це рішення дозволяє зберегти час при старті системи та позбавляє необхідності тримати систему у нерухомому стані.

Обраний модуль 3-осьового гіроскопу і 3-осьового акселерометру дає змогу отримати дані про положення систему у просторі в декількох форматах: у вигляді кватерніонів або кутів Ейлера. Обидва способи мають свої переваги та недоліки, тому робити вибір потрібно з огляду на потреби системи, в якій вони будуть використовуватись.

Кути Ейлера – кути, що описують поворот абсолютно твердого тіла у тривимірному евклідовому просторі. В загальному вигляді кути Ейлера описують положення тіла шляхом представлення трьох кутів повороту навколо кожної з осей тіла відносно початкового положення тіла, яке обертають. Початковим положенням вважається таке положення, в якому кут нахилу від усіх трьох осей обертання дорівнює нулю.

Концепція кутів Ейлера також відома, як отримання даних про положення тіла у вигляді трьох параметрів: Yaw, Pitch, Roll. Ці англійські слова характеризують обертання навколо кожної з трьох осей – x, y та z. Ця теорія значно легше до сприйняття ніж кватерніони.

Кватерніони – це запропонована у 1843 році математиком Вільямом Гамільтоном система гіперкомплексних чисел, яка утворює векторний простір розміром чотири над полем дійсних чисел.

Кватерніони надають зручне математичне позначення орієнтації простору та обертання об'єктів у цьому просторі. У порівнянні з кутами Ейлера, кватерніони дозволяють простіше комбінувати обертання, а також уникнути проблеми, пов'язаної з неможливістю повороту навколо осі, незалежно від скоєного обертання по інших осях. У порівнянні з матрицями вони мають більшу обчислювальну стійкість і можуть бути більш ефективними.

Кватерніони знайшли своє застосування у багатьох сферах: комп'ютерній графіці, робототехніці, навігації, молекулярній динаміці та інше. Кватерніони - найкращий спосіб плавно інтерполювати обертання. Проста інтерполяція

матриць обертання не працює, тому що на практиці не завжди є можливим отримати матрицю обертання в результаті. Інтерполяція кутів Ейлера не призводить до плавного обертання, що може дуже сильно нашкодити загальній роботі системи (наприклад, зробити анімацію об’єктів у комп’ютерній грі обривчастою, що дуже погано вплине на загальне враження від ігрового процесу). Таким чином, для анімації обертань, як це потрібно у комп'ютерній графіці чи робототехніці, кватерніони – це шлях до найкращого результату.

Для кращого розуміння кватерніонів, буде корисно порівняти їх з yaw, pitch, roll, що є концепцією, з якою більшість людей вже знайомі. Щоб відобразити зміну орієнтації, спочатку вказується кут повороту, який є поворотом навколо осі z. Потім додається pitch, який є обертанням навколо осі Y. І, нарешті, обертання навколо осі x. Звичайно, об’єкт системи може робити це в іншому порядку або, швидше за все, відразу за всіма осями, але кінцевим результатом все одно буде зміна орієнтації. Ключовим тут є те, що для визначення положення об’єкту в системі потрібні лише три параметри (ψ, θ, ϕ).

Тепер можна порівняти цю концепцію з кватерніоном, який вимагає використання чотирьох параметрів. Отже, кватерніон спочатку використовує вектор і вказує його в напрямку, в якому вам потрібно рухатися. Це зображено червоною стрілкою на рис 3.8 і завжди дорівнює одиниці довжини. Оскільки стрілка може вказувати куди завгодно, і це тривимірний простір, нам потрібні три параметри, щоб визначити його. Параметри напряму задаються у вигляді синусів. Коли у нас є напрямок, ми можемо виконати оберт навколо власної осі (зображено сірою стрілкою на рис.3.8), щоб перейти до остаточної орієнтації. Це мета четвертого параметра. Він визначає в градусах (або радіанах), яким повинне бути значення цього оберту.

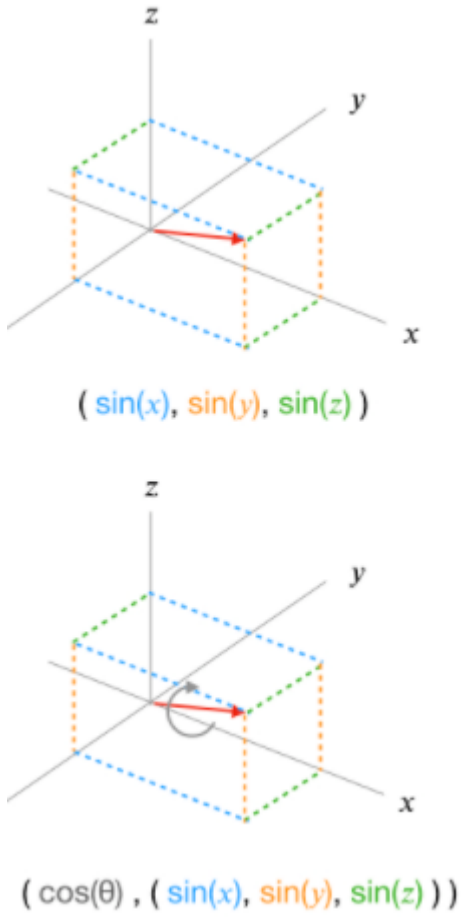

Рисунок 3.8 – Представлення кватерніонів

Для полегшення розрахунку даних модуль MPU-6050 обладнаний спеціальним процесором, який робить всі розрахунки самостійно. Цей процесор має назву Digital Motion Processor (DMP). DMP – це блок обробки, який може самостійно виконувати алгоритм розрахунку орієнтації. Наприклад, очевидні переваги використання DMP для реалізації алгоритму об'єднання датчиків у складні конструкції. По-перше, офіційно наданий invensense алгоритм розрахунку повинен бути набагато надійнішим за оригінальні алгоритми. По-друге, алгоритм обчислення положення, реалізований DMP, звільняє мікроконтролер від тиску обробки алгоритму. Натомість мікроконтролер повинен дочекатися зовнішнього переривання, згенерованого після завершення

обчислення DMP, і прочитати результат обчислення положення у зовнішньому перериванні. Така реалізація дозволить одно-кристальному мікрокомп'ютеру зберегти багато часу для вирішення інших завдань, що покращує продуктивність системи в реальному часі.

Хоча метод отримання положення тіла у просторі за допомогою методу кватерніонів і є найбільш точним, він має декілька недоліків. Наприклад, кватерніони можуть стати недійсними із-за помилки округлення чисел з плаваючою комою. Однак цієї помилки можна позбутись за допомогою ренормалізації кватерніона. Також важливим є те, що для їх обчислення потрібен значний ресурс мікропроцесора, і для їх розуміння потрібно ознайомитись з м

Головним недоліком кутів Ейлера є можливість отримання помилки в розрахунках через явище під назвою Gimbal lock. Головним ефектом цього явища є те, що кінцеве положення тіла може відрізнятися при зміні порядку осей обертання. Наприклад, нам необхідно повернути об’єкт на 45 градусів по кожній з осей обертання (X: 45°, Y: 45°, Z: 45°). Якщо спочатку повернути на 45° навколо осі X, потім навколо Y і в кінці навколо Z, то вийде результат як на лівій половині рис. 3.9. Якщо порядок буде Z-X-Y, то результат буде інший, як у правій половинці.

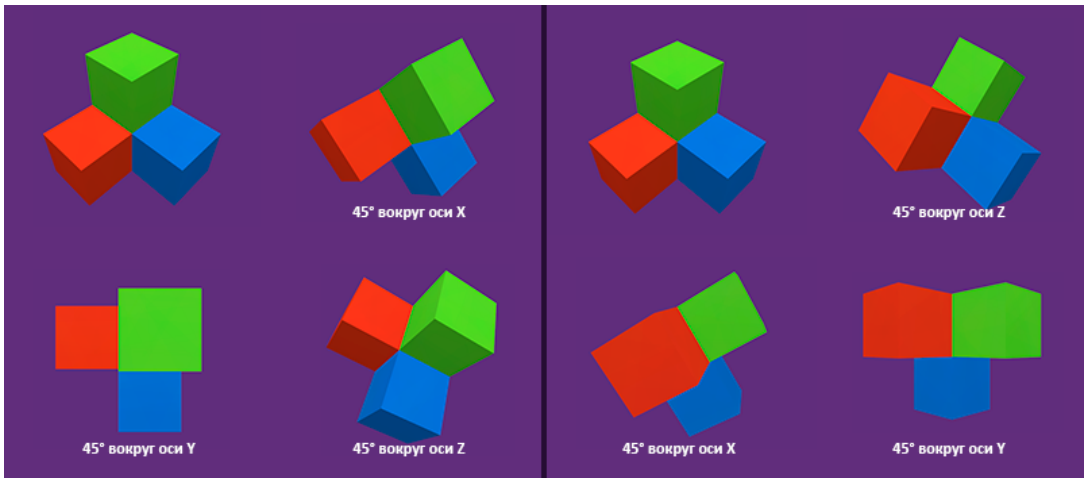

Рисунок 3.9 – Приклад ефекту Gimbal lock

Насправді вище описані не просто кути Ейлера, а кути Тейта-Брайана. Ейлерові кути мають багато варіацій, що збивають з пантелику: в деяких потрібно обертати навколо глобальних осей, в інших осі повертаються після кожного кроку, в третіх осі завжди закріплені на самому об'єкті і рухаються разом з ним. До того ж додається різний порядок поворотів. Тому при використанні кутів Ейлера треба чітко розуміти, як ці кути будуть працювати у конкретному проекті.

При реалізаціє програмного коду для Arduino кути з модулю MPU-6050 зчитуються у вигляді кутів Ейлера. Таке рішення було обґрунтоване трьома факторами, описаними нижче.

По-перше, кути Ейлера зрозуміліші і не потребують вивчення великої кількості додаткової теорії. Це робить написаний код більш зрозумілим, та полегшує процес пошуку та виправлення помилок або вдосконалення коду.

По-друге, складність розрахунків кватерніонів потребує більше часу. Розрахунок кватерніонів за допомогою DMP не має сенсу, бо процесор Arduino Nano зайнятий лише обробкою натискання кнопок та відправкою даних по Bluetooth, і це потребує дуже мало ресурсів. Тому більш потужний процесор Arduino Nano буде виконувати обробку всіх даних швидше, ніж DMP процесор MPU-6050 буде виконувати лише розрахунок положення системи у просторі. А при порівнянні дослідницьким шляхом часу, витраченого на розрахунок кватерніонів та кутів Ейлера за допомогою процесору Arduino, стало зрозуміло, що розрахунок кватерніонів потребує трохи більше часу.

По-третє, кватерніони оперують чотирьома параметрами, для того, щоб визначити кути оберту системи (три параметри) та його обертання навколо власної осі (четвертий параметр). Саме цей четвертий параметр не є істотним в рамках цього атестаційного проекту, тому що умовно систему можна

представити як промінь, що вказує на точку на екрані. А від обертання промінню навколо власної осі нічого не зміниться. Тому цей параметр є не істотним і його розрахунок лише сповільнятиме систему.

Для зчитування даних з модулю MPU-6050 використовується бібліотека Wire.h. Значення гіроскопу та акселерометру зчитаються як два слова (байти), об'єднуються в подвійне слово, представлене типом int у мові програмування C, а потім діляться на константу (спеціальне значення зазначене в документації к модулю, яке залежить від налаштувань чутливості приладу). Після цього від них віднімають значення похибки, для отримання більш точних результатів. На останньому етапі розрахунків виконуються математичні розрахунки для перетворення отриманих параметрів в кут Ейлера, представлений трьома головними параметрами: roll, pitch, yaw. Повний код розрахунків наведений у прикладі 3.2.

```
      // === Read acceleromter data === //
        Wire.beginTransmission(MPU);
        Wire.write(0x3B); // Start with register 0x3B (ACCEL_XOUT_H)
        Wire.endTransmission(false);
        Wire.requestFrom(MPU, 6, true); // Read 6 registers total, each axis
value is stored in 2 registers
        //For a range of +-2g, we need to divide the raw values by 16384,
according to the datasheet
        AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; // X-axis value
        AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; // Y-axis value
        AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; // Z-axis value
        // Calculating Roll and Pitch from the accelerometer data
        accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI)
- AccErrorX; // AccErrorX ~(0.58) See the calculate_IMU_error()custom function
for more details
        accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 /
PI) - AccErrorY; // AccErrorY ~(-1.58)

        // === Read gyroscope data === //
        previousTime = currentTime;        // Previous time is stored before the
actual time read
        currentTime = millis();            // Current time actual time read
        elapsedTime = (currentTime - previousTime) / 1000; // Divide by 1000 to
get seconds
        Wire.beginTransmission(MPU);
        Wire.write(0x43); // Gyro data first register address 0x43
        Wire.endTransmission(false);
```

```
        Wire.requestFrom(MPU, 6, true); // Read 4 registers total, each axis
value is stored in 2 registers
        GyroX = (Wire.read() << 8 | Wire.read()) / 131.0; // For a 250deg/s
range we have to divide first the raw value by 131.0, according to the datasheet
        GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
        GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;

        // Correct the outputs with the calculated error values
        GyroX = GyroX - GyroErrorX; // GyroErrorX ~(-0.56)
        GyroY = GyroY - GyroErrorY; // GyroErrorY ~(2)
        GyroZ = GyroZ - GyroErrorZ; // GyroErrorZ ~ (-0.8)

        // Currently the raw values are in degrees per seconds, deg/s, so we
need to multiply by sendonds (s) to get the angle in degrees
        gyroAngleX = gyroAngleX + GyroX * elapsedTime; // deg/s * s = deg
        gyroAngleY = gyroAngleY + GyroY * elapsedTime;
        yaw =  yaw + GyroZ * elapsedTime;

        // Complementary filter - combine acceleromter and gyro angle values
        roll = 0.96 * gyroAngleX + 0.04 * accAngleX;
        pitch = 0.96 * gyroAngleY + 0.04 * accAngleY;
```

Приклад 3.2 – Програмна реалізація роботи з модулем MPU-6050

Після отримання результатів параметрів roll, pitch, yaw вони відправляються на програмне забезпечення, встановлене на обчислювальному пристрої, за допомогою Bluetooth модулю.

Для роботи з Bluetooth модулем використовувалась бібліотека SoftwareSerial.h. Вона необхідна для того, щоб створити з'єднання з Bluetooth модулем через визначені контакти на платі Arduino Nano. Згідно зі схемою атестаційної роботи використовувались контакти D2 та D3. Для правильної роботи системи потрібно зазначити частоту роботи з'єднання. Воно повинно співпадати с частотою Bluetooth на обчислювальному пристрої. Процес ініціалізації Bluetooth модулю наведений у прикладі 3.3.

```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(2, 3);

void setup()
{
    Serial.begin(9600);
    BTserial.begin(9600);
}
```

Приклад 3.3 – Процес ініціалізації Bluetooth модулю

Процес передачі даних через модуль HC-05 виглядає дуже просто і наведений у прикладі 3.4.

```
      BTserial.println((String)roll + " " + pitch + " " + yaw + " " +
isLeftHolded + " " + isRightHolded);
```

Приклад 3.4 – Процес передачі даних по Bluetooth

Програмний код дуже схожий на код, яким виконують запис логів, які можна переглянути за допомогою Arduino IDE. Тобто формується рядок с даними, і потім відправляється через послідовний порт Bluetooth модулю. Таким чином можна сказати, що написання коду, для передачі даних не викликає великих труднощів. Але потрібно звернути особливу увагу на формат передачі даних.

На сьогоднішній день у системах інтернет сполучення (таких як IoT, online ігри, web-додатки з Front-End та Back-End частинами та інші) використовують дуже велику кількість форматів передачі даних. В залежності від специфіки задач, які представлені перед системою, DTO (Data Transfer Object) може бути представлене у таких форматах як JSON, XML, HTML та інші. На сьогоднішній день одним з найпопулярніших є формат JSON.

JSON - це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, та може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передачі структурованої інформації через мережу. Процес перетворення об'єктів в формат JSON називається серіалізація, а перетворення з JSON в об'єкт – десеріалізація.

Популярність формат JSON набув завдяки декільком важливим перевагам:

- не займає багато місця, є компактним у написанні та швидко компілюється;

- створення текстового вмісту зрозуміло людині, просто у реалізації, а читання із боку середовища розробки немає ніяких проблем. Читання може здійснюватися і людиною, оскільки нічого складного у поданні даних немає;

- структура перетворюється для читання будь-якими мовами програмування;

- майже всі мови мають необхідні бібліотеки або інші інструменти для читання даних у форматі JSON.

Але для виконання цієї атестаційної роботи було прийнято рішення не використовувати JSON формат, а передавати необхідні параметри як рядок з усіма параметрами, відокремленими пробілом. Тобто відправляється як звичайний текст.

Головною причиною такого рішення стало надлишкова інформація, яка транспортується у моделі JSON. Вже згадувалось, що формат відображення даних JSON не займає багато місця та є достатньо компактним, але все одно для правильного процесу десеріалізації необхідні деякі додаткові символи. Наприклад, у JSON записуються не тільки значення параметрів, а і їх назву. Це дозволяє безпомилково розшифрувати отриманні дані, незважаючи на послідовність параметрів на стороні відправника. До того ж, саме завдяки цьому JSON є форматом, який легко читати навіть людині.

Але кожна назва параметру – це ще кільки байтів до розміру DTO. Часом назва параметру потребує навіть більше пам’яті, ніж його значення. Особливо це помітно при передачі числових параметрів та параметрів з типом Boolean (true або false), як ,наприклад, прапорці натискання кнопок у цій атестаційній роботі. Саме через це об’єм пам’яті, необхідний для передачі даних контролеру у форматі JSON, значно перевищує об'єм пам'яті, необхідний для передачі тієї ж інформації рядком.

Для порівняння у прикладі 3.5 наведений приклад даних у форматі JSON, а у прикладі 3.6 наведені ті ж самі дані, але рядком. Різниця у розмірі між ними досягає приблизно 4 рази (83 символи для JSON та 21 – у рядку) для мінімізованого JSON, та близько 5 разів (110 символів для JSON та 21 – у рядку) для відформатованого формату JSON. У мінімізованому форматі з моделі прибираються всі пробіли та символи переносу строки, для мінімізації розміру. А відформатований JSON більш підходящий для сприйняття людиною.

```
{
   "roll":10.52,
   "pitch":12.34,
   "yaw":23.45,
   "isLeftButtonHold":1,
   "isRightButtonHold":0
}
```

Приклад 3.5 – Приклад представлення даних у форматі JSON

```
10.52 12.34 23.45 1 0
```

Приклад 3.6 – Приклад представлення даних рядком

Такі зайві витрати гарантують стабільність роботи системи: попереджають випадкові порушення формату передачі даних, не чутливих к зміні послідовності. Але для транспортуванню даних за допомогою Bluetooth таке збільшення розміру даних може призвести до зменшення швидкості роботи всієї системи. Саме через це дані відправляються як перелік параметрів звичайним текстом.

Процес зчитування даних з модулю MPU-6050 та відправлення їх на комп'ютер за допомогою технології Bluetooth є досить вагомим з точки зору витраченого часу. Особливо це помітно на фоні обробки натискання кнопок, які майже не витрачають часу роботу процесору. Через цю особливість не можна

проводити обробку кнопок та роботу з модулем MPU-6050 один за одним у функції loop.

Обробка натискання всіх кнопок системи займає приблизно 1-2 мс, що приблизно в 30 разів швидше ніж процес зчитування даних з модулю MPU-6050 та їх відправлення. Тому послідовне виконання призведе до того, що перевірка стану кнопок буде відбуватися недостатньо часто, а це призведе до втрати їх чутливості. Тобто швидкий клік може бути не зчитаним, а відгук на довге натискання потребуватиме додаткового часу для розпізнавання. Це є дуже поганою поведінкою для системи керуванням комп'ютером.

Для запобігання цьому використовується концепція програмних таймерів. Вони дозволяються викликати необхідні функції з визначеним періодом. Реалізовано це за допомогою системної функції millis(), яка повертає кількість мілісекунд, які минули з моменту старту системи, та глобальної змінної, яка зберігає час останнього виклику функції. Віднімаючи від значення, отриманого за допомогою millis(), час останнього виклику функції, можна отримати кількість мілісекунд, які минули з останнього виклику. Якщо це значення більше ніж зазначений період, тоді потрібно викликати функцію, інакше – чекати.

Такий програмний таймер був реалізований для роботи з модулем MPU-6050, тому що це найбільш довгий процес. Таймер налаштований на період в 30 мілісекунд. Функції обробки натискання кнопок викликаються кожну ітерацію функції loop, тому що вони швидко виконуються, і це забезпечує достатню чутливість кнопок. Програмна реалізація цього рішення наведена у прикладі 3.7.

```
RightClickHandler();
LeftClickHandler();
ResetButtonHandler();

if(millis() - mpuTimer > 30){
```

```
    mpuTimer = millis();
    SendMPU6050Data();
  }
```

Приклад 3.7 - Реалізація програмного таймеру

Апаратна частина атестаційної роботи обладнана трьома кнопками: одна кнопка для скидання налаштувань модулю MPU-6050 – Reset, і дві кнопки для моделювання натискання лівої та правої кнопки миші – керуючі кнопки. Для підключення кнопок використовувались цифрові піни D5-D7 плати Arduino Nano. У програмному коді для роботи з кнопками використовується бібліотека GyverButton.h. Вона обертає дуже багато низікорівневої роботи, що дозволяє не перейматися особливостями взаємодії з кнопкою безпосередньо. Наприклад, ця бібліотека дозволяє налаштовувати часові затримки, після яких натискання вважається дійсним. Це дозволяє позбутися проблем пов'язаних з брязкотом контактів та випадковими натисканнями.

Для того, щоб користуватись можливостями бібліотеки GyverButton.h, необхідно використовувати об’єкт класу GButton для роботи з кнопкою. Для створення об’єкту класу GButton використовується конструктор, в якому можна зазначити три параметри: пін, на якому працюватиме кнопка; тип кнопки (HIGH_PULL чи LOW_PULL); напрям кнопки (нормально зімкнута чи розімкнута). Але також є конструктор лише з одним параметром: піном. При використанні такого конструктору тип кнопки буде визначений як HIGH_PULL і кнопка буде нормально розімкнута. Саме цей конструктор був використаний для створення об’єктів кнопки у цій атестаційній роботі.

Крім того потрібно кожну ітерацію визивати метод tick() для кожного об’єкту кнопки. Цей метод необхідний для збирання даних про стан кнопки. Викликати цей метод потрібно періодично, тому найкращою практикою є викликання його кожну ітерацію функції loop. Зважаючи на це, виклик цього

методу для кожної кнопки виконується на початку функції loop, до виклику методів обробки натискання кнопок, щоб мати актуальну інформацію про їх стан.

При додаванні кнопок до апаратної частини системи до них додають підтягуючий або стягуючий резистор (в залежності від того, на якому рівні має працювати кнопка). Головною метою такого резистору є гарантування, що на логічному виході завжди буде один рівень сигналу (високий або низькі) у розімкнутому стані кнопки. Схемі підключення стягуючого та підтягуючого резисторів наведена на рис. 3.10 та рис. 3.11.

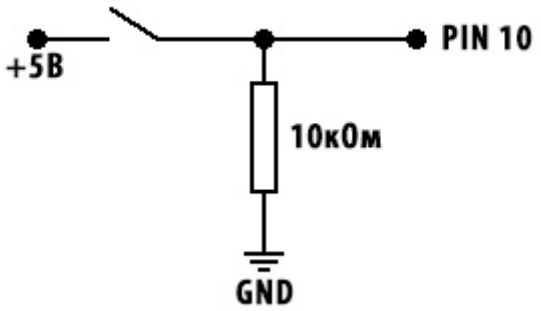

Рисунок 3.10 - Схема підключення стягуючого резистора

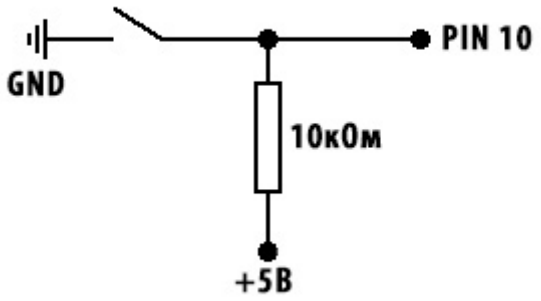

Рисунок 3.11 - Схема підключення підтягуючого резистору

Слід зазначити, що фізично стягуючі та підтягуючі резистори нічим не відрізняються, тобто один і той же резистор можна використовувати і так, і так. Відмінність полягає лише у схемі підключення [14].

При виборі резистору слід звертати увагу на його значення опору. Мінімальне значення опору резистора можна розрахувати за допомогою закону Ома для ділянки електричного кола:

$R = U / I$

Максимальний струм, який може видати цифровий пін Arduino - 40 мА (0,04 А). При напрузі 5 В, згідно із законом Ома, опір ділянки ланцюга буде дорівнювати:

$R = U / I = 5 / 0.04 = 125$ Ом

Таким чином, мінімально допустимий для використання номінал резистора — 125 Ом. Але для того, щоб резистор, що стягує/підтягує, не впливав на інші ділянки ланцюга при замиканні (ефект дільника напруги), рекомендується вибирати набагато більший номінал (1-10 кОм).

Але для того, щоб не перенавантажувати апаратну частину системи додатковими елементами, під час реалізації всіх кнопок була використана особливість мікроконтролеру. Вона полягає у тому, що у мікроконтролерів AVR є вбудовані резистори для всіх GPIO пінів. Ці резистори підключені до живлення (до виходу VCC), тобто буквально дублюють схему рис. 3.6 з підтягуючим резистором і дозволяють не використовувати зовнішній резистор. Але треба мати на увазі, що при такій реалізації кнопка обов’язково буде працювати як нормально зімкнута, тобто на цифровому вході плати буде сигнал 1, коли кнопка буде в звичайному стані, і 0 – коли кнопка буде натиснута. Для більшої зручності всі ці подробиці ховаються під реалізацією бібліотеки GyverButton.h, тому для правильного функціонування треба тільки правильно вказати тип кнопки під час ініціалізації об’єкту класу GButton.

Одна з вище наведених кнопок виступає як Reset для модулю MPU6050. Головною метою цієї кнопки є дати користувачу контролера завдати поточне положення контролеру (поточний кут нахилу системи) як початкові. Таким

чином у користувача завжди є можливість самому обрати, в якому положенні керувати курсором комп'ютеру (прямо перед екраном монітору, трохи збоку, опустивши руку до полу і т.і.), а не постійно шукати єдине положення в якому курсор буде у центрі екрану.

З точки зору модуля MPU6050, необхідно зробити скидання поточних значені регістрів з даними акселерометра та гіроскопу. Фактично, зробити reset модулю. У документації до модулю наведений спосіб того, як можна це зробити. Для цього необхідно встановити значення регістру PWR_MGMT_1 (опис регістру наведений на рис 3.12) в 0 [15].

6B	107	PWR_MGMT_1	R/W	DEVICE _RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]

Рисунок 3.12 - Опис регістру PWR_MGMT_1 модулю MPU6050

Для обробки натискання кнопки Reset створена спеціальна функція, яка викликається при кожній ітерації функції loop(). Її реалізація наведена у прикладі 3.8. Алгоритм цієї функції можна описати наступним чином: якщо кнопка B_Reset натиснута, скинь налаштування модулю MPU-6050 та глобальні змінні системи, які зберігали проміжні дані акселерометру та гіроскопу.

Для визначення поточного стану кнопки використовується функція бібліотеки GyverButton.h – isPress(). Вона повертає true, якщо кнопка натиснута, а після цього скидає її у стан «відтиснута». Це забезпечує те, що функція ресет буде викликана лише один раз за єдине натискання. Для скидання налаштувань модулю MPU-6050 потрібно вказати значення для одного з його регістрів. Для цього використовується бібліотека Wire.h. Після завершення процесу скидання налаштувань у консоль робиться запис з повідомленням «Reset». Це необхідно для покращення процесу спостерігання за процесами системи за допомогою Arduino IDE.

```
void ResetButtonHandler(){
  if(B_Reset.isPress()){
    Wire.begin();
    Wire.beginTransmission(MPU);
    Wire.write(0x6B);
    Wire.write(0x00);
    Wire.endTransmission(true);
    accAngleX = 0;
    accAngleY = 0;
    gyroAngleX = 0;
    gyroAngleY = 0;
    gyroAngleZ = 0;
    yaw =0;
    Serial.println("Reset");
    return;
  }
}
```

Приклад 3.8 - Програмна реалізація функції обробки натискання кнопки Reset

Для реалізації кнопок, необхідних для моделювання лівої та правої кнопки миші, були створені дві окремі функції. Програмна реалізація цих функцій для лівої та правої керуючих кнопок наведена у прикладі 3.9. Для контролю поточного стану кожної з кнопок були створені глобальні змінні типу bool. Причиною тому є те, що обробка натискання кнопок і відправка даних по Bluetooth знаходяться в різних функціях, і глобальні змінні - найпростіше і найбільш правильне рішення для передачі даних між цими функціями. Саме підтримкою актуального значення цих глобальних змінних фактично і займаються функції обробники натискання кнопок.

```
void RightClickHandler(){
  if(B_FingerRight.isHold()){
    isRightHolded = 1;
  }
  else{
    isRightHolded = 0;
  }
}

void LeftClickHandler(){
  if(B_FingerLeft.isHold()){
    isLeftHolded = 1;
```

```
  }
  else{
    isLeftHolded = 0;
  }
}
```

Приклад 3.9 - Програмна реалізація обробки натискання керуючих кнопок

На відміну від реалізації обробки кнопки Reset, для отримання поточного стану керуючих кнопок використовується функція isHold(). Вона повертає true, якщо кнопка натиснута вже деякий час. Цей час зазначається при створені змінної кнопки і необхідний для того, щоб можна було відокремити швидкий клік від довгого натискання. Цей час дорівнює 100 мс для керуючих кнопок.

Використання функції isPress() не є можливим, бо після її відпрацювання вона скидає стан кнопки, а це значить, що буде втрачена можливість відстежувати тривалий натиск. Це буде дуже критичним недоліком, тому що тривалий натиск лівої кнопки миші використовується як окремий тип команди. Він використовується, наприклад, для виділення деяких об’єктів на робочому столі комп’ютеру, перетягування об’єктів між директоріями, малювання в програмі Paint, тощо.

Після отримання поточного стану кнопки відбувається зміна значення глобальної змінної, яка зберігає стан поточної кнопки. Потрібно відзначити, що дія відбувається не тільки, коли кнопка натиснута, але й коли її відпустили. Інакше була б втрачена подія відтискання кнопки, що призвело би ефекту залипання кнопки після першого натискання.

3.3 Реалізація програмного коду для Desktop Application

Розробка програмного забезпечення для створення Desktop Application проводилась за допомогою технології .NET WinForms. Це технологія яка має високі показники надійності та швидкості роботи, які повністю задовольняють потреби проекту.

Для створення WinForms проекту використовувалась IDE від компанії JetBrains – Rider. При створенні нового WinForms проекту ця IDE автоматично створює декілька файлів:

- Program.cs – цей файл містить точку старту всієї програми – функцію Main(). У цій функції виконуються базові налаштування системи, а потім відкривається вікно головної форми;
- MainForm.cs – у цьому файлі знаходиться одна з двох частин класу MainForm, класу головного та єдиного вікна програми. Ця частина зберігає всю кодову базу, яка описує логіку поведінки додатку: обробка подій натискання на кнопки, процес ініціалізації компонентів системи, зберігання об'єктів сервісів додаткової обробки інформації, тощо. Саме у цьому файлі робиться більшість змін при налаштуванні вікна додатку;
- MainForm.Designer.cs – у цьому файлі зберігається друга частина класу MainForm. Як зрозуміло з назви, це частина, що відповідає за зовнішній вигляд вікна додатку. В цей файл вноситься інформація про те, які графічні компоненти створені на формі, налаштування їх зовнішнього вигляду (розмір, колір, текст, відступи, тощо), а також посилання на функції-обробники внутрішніх подій (натискання на елемент, зміна значення, виключення елементу, тощо). Зазвичай зміни у цей файл вносить IDE самостійно, коли програміст змінює інтерфейс у спеціальному графічному редакторі;
- MainForm.resx – це файл, який повністю контролюється IDE. Він зберігає метадані форми.

Створення об'єкту форми, як і для будь-якого об'єкту у .NET, починається з виклику конструктору. У конструкторі відбувається ініціалізація всіх інтерфейсних елементів за допомогою виклику методу InitializeComponent(). Цей метод знаходиться у файлі MainForm.Designer.cs і створюється системою, тому і виклик цього метода з конструктору система додає автоматично. Якщо його

прибрати, форма не буде створена, а користувач побачить помилку (у debug режимі) чи взагалі тільки екстрене завершення роботи додатку (у звичайному режимі роботи).

Не враховуючи опис налаштувань самої форми, MainForm.Designer.cs містить опис для шести елементів інтерфейсу:

- trackbar – це повзунок, який дозволяє налаштовувати чутливість контролеру. Працює так само, як і налаштування чутливості для комп'ютерної миші;
- numericUpDown – цей елемент дозволяє точним значення налаштовувати значення чутливості контролеру. Крім того він є відображенням значення trackbar-у;
- label – виконує функцію заголовку для двох попередніх елементів;
- comboBox - це елемент з випадаючим списком, в якому відображаються всі доступні послідовні порти, та в якому необхідно обрати порт, з якого буде відбуватись підключення до контролеру;
- button – кнопка, яка ініціює підключення та відключення від контролеру;
- pictureBox – велике полотно, необхідне для малювання допоміжних елементів.

Після того, як об'єкт був створений, виконується створення самої форми. Під час цього процесу викликається метод Form_Load() підписаний на подію форми Load, тобто цей метод буде викликатись лише один раз під час завантаження форми. В цьому методі зазвичай проводять всі додаткові розрахунки, необхідні для завантаження форми. В програмному коді цієї атестаційної роботи цей метод містить виклик методу RefreshSerialPorts(), який актуалізує список послідовних портів комп'ютеру.

Список послідовних портів необхідний для того, щоб обрати той порт, який зарезервований для зв'язку з контролером на базі Arduino. Для того, щоб такий

порт з'явився потрібно підключитись та провести синхронізацію з контролером. Цей процес майже такий самий як і для будь-якого Bluetooth-пристрою:

1) зайти у Bluetooth налаштування комп'ютеру;

2) оновити список доступних пристроїв;

3) знайти пристрій з ім'ям HC-05 та підключитись до нього;

4) після того, як з'явиться вікно для введення паролю, ввести пароль «1234» (пароль за замовчування для всіх модулів HC-05);

5) підтвердити з'єднання.

Програмний код методу RefreshSerialPorts() наведений у прикладі 3.10. Для отримання списку всіх доступних портів використовується статична функція класу SerialPort - GetPortNames() [4]. Вона повертає масив всіх портів у текстовому форматі. Для зручності він додатково сортується за алфавітом. Отриманий список портів заноситься до колекції елементів інтерфейсного елементу comboBox. Для того, щоб уникнути дублювання елементів, перед додаванням нових елементів усі старі видаляються. Після цього перший елемент колекції зазначається як selected (обраний): це необхідно для того, щоб comboBox завжди мав обраний елемент.

```
      private void RefreshSerialPorts()
          {
            var ports = SerialPort.GetPortNames().OrderBy(p =>
p).ToArray();
            if (!ports.Any())
            {
              MessageBox.Show("There is no serial ports to show. Pair
Arduino controller by Bluetooth");
              throw new ArgumentException("There is no serial ports to
show");
            }

            var selectedItem = cb_SerialPort.SelectedItem as string;

            cb_SerialPort.Items.Clear();
            cb_SerialPort.Items.AddRange(ports);
            cb_SerialPort.SelectedItem = selectedItem != null &&
ports.Contains(selectedItem)
              ? selectedItem
              : cb_SerialPort.Items[0];
```

```
}
```

Приклад 3.10 - Реалізація методу RefreshSerialPorts()

Можливий випадок, коли функція GetPortNames() поверне порожній масив портів. На такий випадок була створена перевірка: якщо у масиві немає елементів, показати користувачу повідомлення з описом помилки та створити системний виняток зі схожим повідомленням. Створення винятку призведе до завершення роботи додатку. Це необхідно, тому що без послідовного порту програма не зможе працювати.

Окрім виклику при завантаженні форми метод RefreshSerialPorts() викликається кожного разу, коли користувач відкриває список comboBox-у. Таке рішення дозволяє користувачу завжди бачити актуальний список портів. Для того, щоб не перезаписати вже вибраний елемент, його зчитують, перевіряють чи він не дорівнює null (жоден елемент не вибраний) та чи він знаходиться в актуальному списку портів (вибраний порт більше не обслуговується). Якщо обидві умови виконані, тоді він залишається вибраним елементом checkbox-у, інакше – його замінить перший елемент актуального списку.

Підключення до контролеру відбувається після натискання на кнопку Connect. Під час спрацьовування події OnClick викликається метод b_Connect_Click(). Його алгоритм поділяється на дві частини в залежності від стану порту та назви кнопки. Якщо текст кнопки «Connected», послідовний порт ще не підключений, тому його потрібно підключити та змінити текст кнопки на «Disconected». Якщо текст кнопки «Disconected», порт підключений і його потрібно відключити та змінити текст кнопки на «Connected». Код методу наведений у прикладі 3.11.

```
private void b_Connect_Click(object sender, EventArgs e)
    {
      const string connectText = "Connect";
      const string disconnectText = "Disconnect";
```

```
    if (b_Connect.Text.Equals(connectText))
    {
      SerialPortInit();
      b_Connect.Text = disconnectText;
    }
    else
    {
      _serialPort?.Dispose();
      b_Connect.Text = connectText;
    }
  }
```

Приклад 3.11 - Реалізація методу b_Connect_Click()

Процес підключення до контролеру описаний в методі SerialPortInit() (приклад 3.12). Для підключення до послідовного порту використовується клас з простору імен System.IO.Ports – SerialPort. Цей клас дозволяє підключитись до визначеного порту, вказати налаштування з'єднання і розпочати обмін інформацією. Під час створення об'єкту класу були вказані наступні параметри:

- PortName – назва послідовного порту, до якого необхідно підключитись;
- BaudRate – швидкість послідовної передачі даних. Необхідно, щоб цей показник співпадав зі значення, вказаним у програмній реалізації апаратної частини контролеру під час ініціалізації Bluetooth модулю;
- Parity – поле, яке зазначає необхідність використати протокол перевірки парності, і якщо так, то який саме. У реалізації Bluetooth зв'язку в рамках цього атестаційного проекту протокол перевірки парності не використовується;
- DataBits – стандартна кількість біт на один байт;
- StopBits – стандартна кількість стоп-бітів на байт;
- Handshake – зазначає, чи необхідно використовувати протокол рукостискання для передачі даних через послідовний порт, і якщо так, то який саме. У цій атестаційній роботі протокол рукостискання не використовується.

Крім описаних вище параметрів можна зазначити ще багато, але це не є необхідним. Всі вони мають значення за замовчуванням або успадковують його від налаштувань операційної системи. Якщо значення успадковані від ОС, тоді

змінювати їх дуже не рекомендується, бо з великою ймовірністю ці параметри вже налаштовані найбільш продуктивним чином. Їх зміна може призвести до зменшення швидкості роботи порту, а іноді й до нестабільної роботи.

```
private void SerialPortInit()
    {
      var portName = (string) cb_SerialPort.SelectedItem;

      var port = new SerialPort
      {
        PortName = portName,
        BaudRate = 9600,
        Parity = Parity.None,
        DataBits = 8,
        StopBits = StopBits.One,
        Handshake = Handshake.None,
      };

      try
      {
        port.Open();
      }
      catch (Exception ex)
      {
        MessageBox.Show(ex.ToString());
        return;
      }

      _readThread = new Thread(ReadPortOnDataReceived);
      _readThread.Start(port);
      _serialPort = port;
}
```

Приклад 3.12 - Реалізація методу SerialPortInit()

Після створення та налаштування об'єкту класу SerialPort викликається метод Open(), який встановлює зв'язок з контролером за допомогою технології Bluetooth. Якщо порт зайнятий іншим додатком, контролер не увімкнений або в нього якісь проблеми з модулем Bluetooth, зв'язок не буду встановлено. Причиною цьому є те, що метод Open() намагається встановити зв'язок протягом часу, зазначеного у полі ReadTimeout (воно дорівнює 500 мс за замовчуванням). Як тільки час буде вичерпано, генерується IOException виняток з повідомленням «The semaphore timeout period has expired.».

Виникнення винятку призведе до екстреного завершення роботи програми, а це не є бажаною поведінкою програми. Для запобігання цьому була використана конструкція try_catch. Вона дозволяє перехоплювати винятки, обробляти їх особливим чином та запобігати завершенню роботи програми. Для цього в блок try вносять код, який потенційно може завершитись помилкою, а в блок catch код, який обробляє помилку, яка виникла.

В якості обробки помилок використовуються лише дві команди:

1) створити допоміжне вікно для демонстрації користувачеві тексту помилки, що повинно допомогти йому зрозуміти, в чому проблема;

2) закінчити виконання поточного методу. Це допоможе запобігти виклику наступних команд, які повинні викликатись лише після успішного встановлення зв'язку.

Якщо встановлення зв'язку з контролером відбулося без помилок, створюється додатковий потік (класс Thread), який постійно перевіряє порт на наявність нових даних. Для цього клас SerialPort має спеціальний event – DataReceived. Цей event дозволяє автоматично викликати зазначений метод при надходженні нових даних, але на практиці його використання пов'язано з проблемами.

Використання event-ів достатньо зручний спосіб реагування на зовнішні події, такі як завершення обробки великого об'єму інформації. Але використання DataReceived event-у в рамках цієї атестаційної роботи призвело до значної втрати в швидкості обробки даних, отриманих з контролеру. Це сильно подіяло на швидкість переміщення курсору: він став рухатись дуже уривчасто та повільно.

Для докладного аналізу був розроблений додатковий програмний код, який дозволив отримати частоту передачі та прийому даних між контролером на базі Arduino та комп'ютерним додатком. Отримані результати підтвердили сильну

втрату у швидкості обробки даних: частота відправлення даних перевищувала частоту її обробки майже в десять разів.

Головною причиною цьому є принцип роботи event-ів: вони виконуються в тому ж потоці, що і код, який їх викликав. Тобто алгоритм постійної перевірки порту на наявність нових даних після отримання кожної нової порції даних мав зупинитись для виконання логіки всіх зареєстрованих event-ів. Це призводить до великих пауз між двома сусідніми зчитуваннями. До того ж необхідно виділяти додатковий час для роботи головного алгоритму роботи форми.

Використання окремого потоку вирішило цю проблему. Окремий потік зосереджений лише на отриманні та обробці даних від контролера. Йому не потрібно виділяти свій час для виконання алгоритмів роботи головної форми, а тому його працездатність значно вище. Алгоритм його роботи наведений у прикладі 3.13.

```
      private void ReadPortOnDataReceived(object portObject)
          {
            var serialPort = portObject as SerialPort;
            if (serialPort is null)
            {
              MessageBox.Show("PortObject must be a SerialPort class
object");
              throw new ArgumentException("PortObject must be a SerialPort
class object");
            }

            var mouseService = new MouseService();
            while (serialPort.IsOpen)
            {
              var bluetoothDto = _serialPort.ReadControllerData();
              if (bluetoothDto is null)
              {
                continue;
              }

              var coordinate = bluetoothDto.Angles.ToCoordinate((int)
nud_Distance.Value);
              mouseService.MoveAndClick(new MouseCommandModel(
                coordinate,
                bluetoothDto.LeftButtonState == 1,
                bluetoothDto.RightButtonState == 1));
              Console.WriteLine($"{bluetoothDto.LeftButtonState}");
            }
```

```
}
```

Приклад 3.13 - Реалізація алгоритму зчитування та обробки даних контролеру

Увесь алгоритм роботи окремого потоку представляє метод ReadPortOnDataReceived(). Він має один аргумент – порт, з якого буде відбуватись зчитування. Таке рішення є хорошою практикою, бо дозволяє уникнути недійсних посилань на об'єкт порту. Тобто, якщо об'єкт буде перезаписаний у головному потоці, але додатковий потік буде на нього посилатись, система автоматичного прибирання об'єктів (Garbage Collector), не зможе видалити об'єкт порту, який більше не використовуються. Така поведінка може призвести до memory leak (витоку пам'яті).

Аргумент методу є об'єкт типу object (базового типу для всіх інших типів у .Net), тому що в окремий потік можна передавати тільки об'єкти цього типу. Для зручного використання його перетворюють на об'єкт типу SerialPort за допомогою оператору as. Цей оператор дозволяє безпечно проводити перетворення: якщо перетворити об'єкт не вдалося, він поверне null. У такому випадку буде створене вікно з повідомленням, а робота окремого потоку закінчиться з помилкою завдяки створеній перевірці.

Зчитування та обробка даних виконується всередині оператору while. Він виконує зчитування доки порт відкритий. Як тільки порт закриється, блок while припинить свою роботу, алгоритм методу ReadPortOnDataReceived() буде повністю виконаний і потік успішно завершиться. Алгоритм блоку while можна поділити на дві частини: зчитування та обробку інформації.

Процес зчитування виконує метод об'єкту послідовного порту ReadControllerData(), реалізація якого наведена у прикладі 3.14. Алгоритм методу поділяється на три частини:

1) зчитати строку даних з порту (кожний рядок це окремий набір даних);

2) перевірити правильність даних;

3) сформувати та повернути об'єкт BluetoothDto, який містить всі отриманні дані у зручному вигляді.

```
      public static BluetoothDto ReadControllerData(this SerialPort port)
          {
            var data = port.ReadLine().Trim('\r');
            if (string.IsNullOrWhiteSpace(data))
            {
              return null;
            }

            //validation: is all params given
            var readParams = data.Split(' ',
StringSplitOptions.RemoveEmptyEntries);
            if (readParams.Length != 5 ||
                !float.TryParse(readParams[0].Replace('.', ','), out var
x) ||
                !float.TryParse(readParams[1].Replace('.', ','), out var
y) ||
                !float.TryParse(readParams[2].Replace('.', ','), out var
z) ||
                !int.TryParse(readParams[3], out var isLeftHolded) ||
                !int.TryParse(readParams[4], out var isRightHolded))
            {
              return null;
            }

            // invert param to prevent axis inversion
            return new BluetoothDto(new EulerAngles(-x, y, -z),
isLeftHolded, isRightHolded);
        }
```

Приклад 3.14 - Реалізація методу ReadControllerData()

Зчитування даних з порту виконується за допомогою методу ReadLine(), який зчитує першу отриману строку даних, видаляючи її з буферу обміну, щоб запобігти подвійне зчитування даних. Інколи отриманий рядок може мати спеціальний символ на кінці строки – «\r». Він не є частиною «корисних» даних, тому він видаляється за допомогою метода Trim().

Після отримання чистої строки даних, вона перевіряється на наявність символів. Якщо отриманий рядок не має символів, метод зчитування повертає null, тобто демонструє відсутність даних для зчитування. Ситуація з отриманням

рядка даних без символів трапляється дуже рідко на початку взаємодії з контролером, при перших зчитуваннях даних.

Якщо рядок даних має якісь символи, тоді вважається що дані отримані правильно і їх можна розділяти на окремі параметри. Для отримання окремих параметрів рядок потрібно розділити на підрядки, кожний з яких буде представляти значення свого параметру. Для цього використовується метод Split() класу String. Він розділяє рядок на підрядки згідно з визначеним символом дільником. У описаному вище принципу алгоритмі формування рядка даних було зазначено, що параметри у рядку, записані через пробіл, тому саме його і потрібно використовувати як символ дільник.

Щоб отримати значення параметрів, треба перетворити підрядки до числових значень. Перші три параметри мають бути типу float (числа з плаваючою комою), а всі інші параметри – типу int (цілі числа). Але на етапі перетворення підрядків до числових значень також можуть виникати непередбачувані ситуації. Тому цей процес теж був покритий перевірками.

Для урахування всіх можливих проблем були створені 6 перевірок: одна – на кількість параметрів і п'ять – на можливість перетворення підрядка на числове значення. Перевірки проводяться послідовно через логічний оператор OR, тобто якщо одна перевірка виявить проблему, всі інші виконуватись не будуть а метод негайно поверне null. Це дозволяє зберегти швидкість роботи алгоритму.

Першою перевіркою є перевірка кількості отриманих параметрів, тому що пріоритет цієї перевірки найвищий. Через принцип роботи логічного оператору OR, описаний вище, в першу чергу потрібно виконувати найпростіші перевірки. Тоді якщо вони виявлять проблему, наступні більш важкі для системи операції виконуватись не будуть. До того ж, наступні перевірки звертаються до кожного підрядка, тому невідповідність кількості підрядків очікуваному значенню призведе до помилки OutOfRangeException. Така помилка примусово завершить

роботу всього потоку по зчитуванню даних з порту, що призведе до неправильної роботи всього додатку.

Для перетворення рядка на число кожен з числових типів даних має метод Parse(). Але використовувати цей метод для перевірки даних дуже не зручно, тому що він створює виняток з помилкою кожного разу, коли не може перетворити рядок на число. А як було описано вище, така поведінка терміново завершить роботу потоку зчитування. Для запобігання цьому потрібно кожну перевірку розміщати в блоку try і робити обробку винятку в блоку catch. Це дуже багато зайвого коду, зайве навантаження на процесор (використання конструкції try_catch відносно важка операція) та зменшення читабельності коду. Тому для перевірок був використаний метод TryParse().

Метод TryParse() повертає не перетворене значення рядку, а об'єкт типу bool, який представляє результат перевірки (true – перетворення пройшло вдало, false – перетворити рядок не вдалось). Якщо результатом виконання функції є true, в out параметрі методу можна отримати перетворене значення. Інакше цей параметр буде дорівнювати значенням за замовчуванням для типу, до якого намагалися перетворити рядок. Для числових значень це значення дорівнює нулю.

Після всіх перевірок формується об'єкт класу BluetoothDto, який метод поверне як результат зчитування даних з Bluetooth порту. Перші три параметри укладаються у внутрішній об'єкт типу EulerAngles відповідно до принципу єдиної відповідальності принципів SOLID. Останні два параметри передаються як є.

Отримавши дані, потік зчитування даних перевіряє їх. Якщо вони дорівнюють null, в процесі зчитування виникла помилка, а значить потрібно переходити до наступного зчитування. Якщо дані вдалось зчитати правильно, вони проходять процес обробки, а потім відправляються до спеціального сервісу, який інтерпретує їх в рух курсору.

Процес обробки отриманих даних полягає в тому, щоб перетворити отримані кути Ейлера у координати на плоскості. Щоб вирішити це завдання був розроблений спеціальний алгоритм. Він базується на звичайній геометрії прямокутного трикутника.

Для прикладу розглянемо принцип дії алгоритму для однієї осі (одного кута). Уявімо, що контролер на базі Arduino знаходиться у точці простору під назвою A, екран монітору це площина γ, і з точки A на площину γ спрямований промінь, що перетинає її у точці B (рис. 3.13). Якщо контролер знаходиться у початковому положенні (кут α дорівнює нулю), тоді точка B буде співпадати з началом координат і точкою O. Точка O - це точка перетину перпендикуляру, опущеного з точки A до площини γ. Якщо контролер відхилиться на деякий кут від початкового положення, точка B змінить своє положення. Тоді відстань OB буде характеризувати відхилення від началу координат, тобто фактично координату B за однією віссю на площині γ.

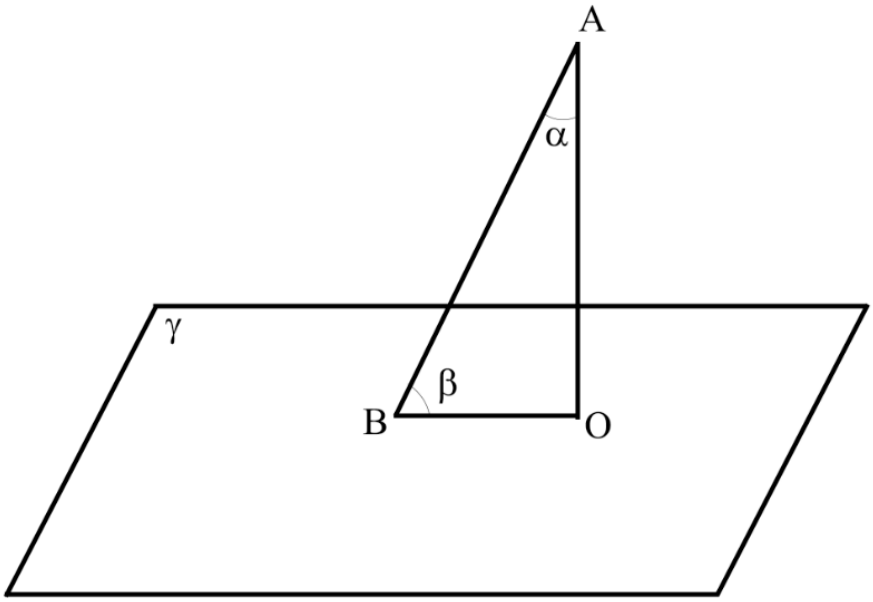

Рисунок 3.13 - Схема роботи алгоритму перетворення кута на координату

Практично це ніби стояти на місці і вказувати лазером на стіну, міняючи його положення лише змінюючи кут його нахилу (не переміщуючи у просторі). Для користувача така поведінка є дуже типовою, бо саме за таким принципом люди вказують пальцями на оточуючі об'єкти.

З точки зору алгоритму необхідно знайти сторону ВО у прямокутному трикутнику АОВ, знаючи тільки кут α. Для вирішення цієї задачі не вистачає даних про довжину сторони АО. Саме від цієї сторони залежить чутливість системи, бо чим вона більше, тим більше буде відрізок ОВ при однаковому куті α (рис. 3.14). Саме тому її довжина налаштовується користувачем як характеристика чутливості контролеру.

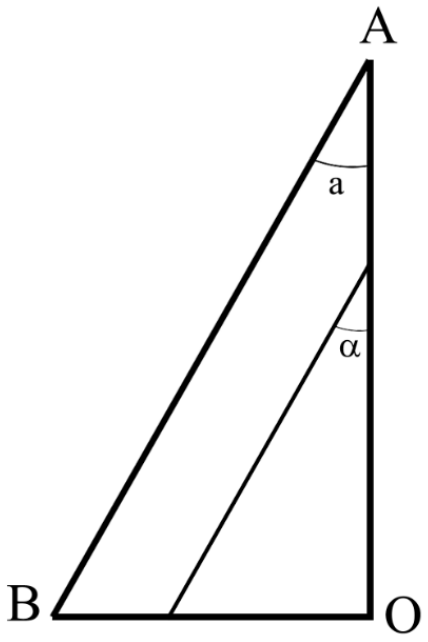

Рисунок 3.14 - Схема залежності довжини ВО від ОА

Отже, маючи кут α та сторону ОА, можна розрахувати кут β, а потім знайти сторону ВО. Кут β можна знайти виходячи з того, що сума кутів трикутника

дорівнює 180 градусів, один з кутів прямий (90 градусів), а другий кут відомий. Таким чином виходить, що кут β дорівнює:

$\beta = 180 - 90 - \alpha$

Сторону ВО можна знайти за допомогою теореми синусів:

$$\frac{AO}{\sin\beta} = \frac{BO}{\sin\alpha}$$

Описаний вище алгоритм знаходження координати курсору виходячи з куту нахилу контролера реалізований у методі ToCoordinate(), код якого наведений у прикладі 3.15.

```
public static Point ToCoordinate(this EulerAngles angles, float h)
    {
      var xProjection = GetProjection(angles.X, h);
      var yProjection = GetProjection(angles.Y, h);

      var x = xProjection * GetSign(angles.X);
      var y = yProjection * GetSign(angles.Y);
      return new Point((int) x, (int) y);
    }

    private static double GetProjection(float angel, float h)
    {
      var absAngle = Math.Abs(angel);
      var sinAlfa = Math.Sin(GetRadian(absAngle));
      var sinBetta = Math.Sin(GetRadian(90 - absAngle)); // 180 - 90 - angle
      return sinAlfa * h / sinBetta;
    }

    private static int GetSign(float angle) => angle > 0 ? 1 : -1;
    private static double GetRadian(double degrees) => Math.PI * degrees /
180.0;
```

Приклад 3.15 - Реалізація алгоритму перетворення кутів Ейлера на координату курсору

Під час реалізації представленого вище алгоритму виникали деякі нюанси, які потребували незначних вдосконалень. По-перше, потрібно враховувати знак отриманого кута. Знак відповідає за напрямок відхилення курсору відносно початку координат, що є дуже важливою інформацією. Тому було прийнято рішення розраховувати довжину відрізку ВО так, ніби кут завжди зі знаком плюс.

Тобто знак мінус завжди заміняється знаком плюс під час розрахунку довжини BO. Це логічно, тому що довжина не залежить від сторони нахилу. А потім до значення довжини додається знак. Таким чином отримується значення відхилення від початку координат.

По-друге, для розрахунку синусів в .NET C# використовується функція Math.Sin(). Вона дуже проста в використанні, але має один нюанс: вона приймає значення кутів у радіанах. Через це неможна одразу передавати значення, отримані з контролеру, бо вони представлені в градусах. Спочатку потрібно перетворити значення у градусах до радіанів. Для вирішення цієї задачі створений метод GetRadian(), код якого представлений у прикладі 3.15.

Після перетворення кутів Ейлера на координати монітору отримані дані можна використовувати для управління курсором. Для того, щоб відокремити логіку керування курсором, був створений клас під назвою MouseService. Він містить всю логіку необхідну для керування курсором та моделювання натискання кнопок миші. Головним і єдиним методом цього класу є метод MoveAndClick() (приклад 3.16). Він приймає об'єкт класу MouseCommandModel, який містить три властивості:

- MousePosition – об'єкт типу Point, який містить координату на екрані, до якої повинен переміститись курсор;
- PressLeftButton – об'єкт типу bool, який сигналізує, чи потрібно натиснути ліву кнопку миші;
- PressRightButton – об'єкт типу bool, який сигналізує, чи потрібно натиснути праву кнопку миші.

```
        public void MoveAndClick(MouseCommandModel command)
            {
              var screenSize = Screen.PrimaryScreen.Bounds.Size;
              var center = new Point(screenSize.Width / 2, screenSize.Height
/ 2);
              var x = center.X + command.MousePosition.X;
              var y = center.Y + command.MousePosition.Y;
```

```
            Cursor.Position = new Point(x, y);

            //press if released
            if (IsLeftMouseButtonPressed != command.PressLeftButton)
            {
              var mouseEvent = command.PressLeftButton ?
MouseFlags.LeftDown : MouseFlags.LeftUp;
              MakeMouseEvent(mouseEvent);
              IsLeftMouseButtonPressed = !IsLeftMouseButtonPressed;
              var action = mouseEvent == MouseFlags.LeftDown ? "Pressed" :
"Released";
              Console.Write($"Left {action}");
            }

            if (IsRightMouseButtonPressed != command.PressRightButton)
            {
              var mouseEvent = command.PressRightButton ?
MouseFlags.RightDown : MouseFlags.RightUp;
              MakeMouseEvent(mouseEvent);
              IsRightMouseButtonPressed = !IsRightMouseButtonPressed;
              var action = mouseEvent == MouseFlags.LeftDown ? "Pressed" :
"Released";
              Console.Write($"Right {action}");
            }
    }
      private static void MakeMouseEvent(MouseFlags mouseEvent)
          {
            mouse_event(mouseEvent, Cursor.Position.X, Cursor.Position.Y,
0, UIntPtr.Zero);
    }
```

Приклад 3.16 - Реалізація методу MoveAndClick()

Метод MoveAndClick() умовно можна поділити на дві частини: переміщення курсору та виконання дій над кнопками миші. Спочатку виконується саме переміщення, тому що користувач очікує, що натиск на кнопку відбудеться там, куди він щойно перемістив курсор.

Для переміщення курсору по заданих координатах використовується клас System.Windows.Forms.Cursor. Для роботи з ним не потрібно створювати об'єкт класу. Це дуже логічно, бо користувач комп'ютеру має лише один курсор і управління ним має відбуватись з одного місця. Сам клас Cursor має статичне поле (поле, до якого можна звертатись без створення об'єкту класу) Position типу Point. При зміні значення цього поля курсор автоматично змінює своє положення.

Але перед тим, як змінити координати положення курсору, потрібно провести додаткові розрахунки. Отримані шляхом перетворення отриманих з контролеру показників координати курсору ще додатково зміщуються. Причиною цьому є особливості роботи процесу скидання налаштувань модулю MPU-6050 на контролері та розрахунок координат екрану.

При скиданні показників модулю MPU-6050 всі показники нахилу контролеру скидаються в 0, тому і координата курсору буде скинута у положення (0;0) на екрані. Ця функція була розроблена для того, щоб часом скидати показники модулю MPU-6050 і запобігти накопиченню похибки даних, а також для того, щоб користувач міг автоматично переміщувати курсор до центру екрану. Це позбавить його необхідності довго водити рукою в пошуках курсору на екрані.

Однак положення (0;0) на екрані знаходиться не в центрі. Особливість екранної системи координат полягає у тому, що початок координат розміщений у верхньому лівому куті екрану, а Y координати збільшуються при зміщенні донизу, а не догори, як це працює у Декартовій системі координат. Тому при скиданні показників нахилу контролеру, курсор автоматично опиниться у верхньому лівому куті екрану, що не є очевидним.

Для того, щоб скидання модулю MPU-6050 переміщало курсор саме до центру, розраховані координати курсору додаються не до початку координат, а до центру екрану. Для цього знаходяться ширина та висота екрану в пікселях, а потім шляхом поділу навпіл знаходяться координати центру. Програмний код реалізації цього алгоритму наведений у прикладі 3.16.

Реалізація натискання клавіш миші потребує, щоб до проекту були імпортовані функції з бібліотеки User32.dll. Ці функції дозволяють програмно модулювати дії периферійних пристроїв, таких як миша, клавіатура, тощо. Для реалізації роботи цього атестаційного проекту необхідна лише функція для

керування курсором. Вона називається mouse_event(), і її реалізація наведена у прикладі 3.17.

```
      [DllImport("User32.dll")]
          private static extern void mouse_event(MouseFlags dwFlags, int
dx, int dy, int dwData, UIntPtr dwExtraInfo);

          [Flags]
          enum MouseFlags
          {
            Move = 0x0001,
            LeftDown = 0x0002,
            LeftUp = 0x0004,
            RightDown = 0x0008,
            RightUp = 0x0010,
            Absolute = 0x8000,
            MiddleDown = 0x0020,
            MiddleUp = 0x0040,
            XDown = 0x0080,
            XUp = 0x0100,
            Wheel = 0x0800,
            HWheel = 0x01000, // Horizontal wheel
    }
```

Приклад 3.17 - Імпортовані функції для керування курсором

Алгоритм керування натиском лівої та правої кнопки миші повністю співпадають. Цей алгоритм можна сформулювати наступним чином: якщо кнопку потрібно натиснути / відпустити і вона ще не натиснута / відпущена, над нею потрібно зробити відповідну дію. Для перевірки чи кнопка вже натиснута використовуються внутрішні поля класу MouseService: IsLeftMouseButtonPressed та IsRightMouseButtonPressed. Ці дві змінні типу bool зберігають поточний стан кнопок миші.

Для того, щоб зробити дію над кнопкою миші, необхідно підібрати відповідне значення MouseFlags. MouseFlags це тип-перелічення, яке містить всі можливі дії над мишею (повний список можна переглянути у прикладі 3.17). Їх багато, але в цьому проекті використовуються лише чотири: LeftDown, LeftUp, RightDown, RightUp. Вибір прапору базується на даних, отриманих з контролеру.

Після визначення прапору визивається функція mouse_event() та змінюється стан змінної, яка відповідає за стан кнопки, над якою виконана дія.

Після того як всі, описані вище, дії завершено потік зчитування даних переходить до наступної ітерації блоку while. Цей цикл буде працювати доки порт зчитування буде відкритий. Закривається він у двох випадках: коли натискають кнопку Disconect на головній формі та коли закривають програму.

Окрім кнопки підключення до контролеру інтерфейс форми містить два елементи, які допомагають користувачу налаштувати чутливість контролеру: trackBar і numericUpDown. Вони контролюють значення довжини відрізку AO (рис. 3.13). За замовчуванням значення чутливості встановлене 1300. Дослідним ляхом було встановлено, що така чутливість є найбільш зручною. Але це значення можна змінювати в проміжку між 500 та 3000.

Також був розроблений допоміжний елемент інтерфейсу, який повинен допомогти користувачеві швидше налаштувати чутливість контролеру. Під час перевірок стало зрозуміло, що оцінювати швидкість руху курсору зі звичайною стрілкою (форма курсору за замовчуванням) не зручно, бо на великій швидкості він зливається з фоном, і його важко відстежувати. Щоб зробити переміщення курсору більш явним, більша частина форми використовується як платформа для тестів. Коли на цю платформу наводиться курсор, під ним з'являється червоне коло. Воно значно більш примітне і дозволяє без зусиль стежити за швидкістю курсору. Таке рішення допомагає значно швидше звикнути до нового способу керування курсором.

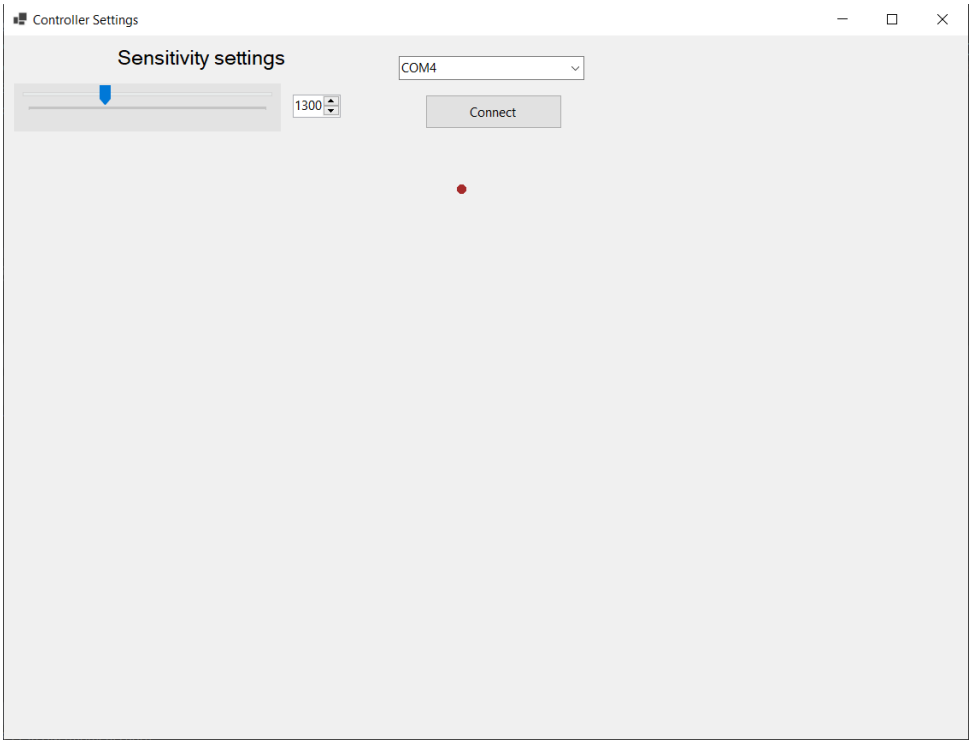

Рисунок 3.15 - Загальний вигляд інтерфейсу програми

3.4 Аналіз отриманого результату

У ході виконання даної атестаційної роботи було розроблено макет бездротового контроллеру для керування обчислювальним пристроєм, робота якого заснована на технології інерційного трекінгу, яка реалізований на базі модулю MPU6050. Також був розроблений додаток для обчислювального пристрою, який приймає сигнал контроллеру та керує курсором. Було виконані всі поставлені завдання, проведено аналіз та вибір складових комп'ютерної системи, які відповідають заявленим критеріям.

Створений макет є дуже компактним завдяки невеликим розмірам усіх компонентів системи. Компоненти були обрані згідно їх характеристик, щоб вони витримали робочі навантаження та були сумісні між собою. Завдяки невеликому розміру макет можна з легкістю розмістити у руці, транспортувати та зберігати. Єдине обмеження це необхідність міняти акумулятор за необхідністю.

Крім того, в середовищі Arduino IDE розроблено програмний код, за допомогою якого реалізовується управління системою, а саме зчитування данних з гіроскопу та акселерометру, приведення їх до зручного формату та передача їх за допомогою технології Bluetooth. Через простоту та доступність інструментів для роботи з Arduino систему дуже легко вдосконалювати. Наприклад, частоту та чутливість датчиків можна змінювати лише трохи вдосконаливши програмний код. А для використання додаткових апаратних засобів, їх треба правильно ввести в схему та змінити програмний код для правильної роботи системи. Таким чином система вийшла дуже гнучкою.

ВИСНОВКИ

У ході виконання атестаційної роботи «Бездротовий контролер для керування обчислювальним пристроєм» було розроблено контролер на основі гіроскопу та акселерометру, який передає дані за допомогою бездротових технологій. Дана розробка повністю відповідає тематиці атестаційної роботи.

Контролер працює на базі модулів MPU6050 та HC-05. Контролер здатний розраховувати параметри інерційного трекінгу та передавати їх через Bluetooth на обчислювальний пристрій. Опираючись на ці параметри програма, встановлена на цьому пристрої, може керувати курсором або виконувати інші дії.

Покращити розробку можна за допомогою додавання додаткових функцій, таких як:

- додати кнопки або інші способи передачі додаткової інформації;
- використати менші за розміром компоненти та переробити пристрій у формі браслету для більш зручного використання;
- використати багаторазовий блок живлення;
- додання функцію розпізнавання жестів, для того щоб використовувати комбінації жестів як незалежні команди;
- використати Bluetooth для об’єднання декількох таких систем та синхронізувати їх роботу.

Усі поставлені завдання атестаційної роботи було виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хофман Д. Освоение Arduino: Проектный подход к электронике, схемам и программированию / Джон Хофман., 2018. – 302 с. (дата звернення: 25.02.2021).

2. Arduino and MPU6050 Accelerometer and Gyroscope Tutorial [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/#comments. (дата звернення: 15.02.2019).

3. Урок 15. Bluetooth модуль HC-06 подключение к Arduino. Управление устройствами с телефона. [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://lesson.iarduino.ru/page/bluetooth-modul-hc-06-podklyuchenie-k-arduino-upravlenie-ustroystvami-s-telefona/. (дата звернення: 18.03.2019).

4. SerialPort Class [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/api/system.io.ports.serialport?view=dotnet-plat-ext-5.0. (дата звернення: 25.12.2021).

5. Что такое резистор [подробная статья] - для чего нужен URL: https://www.radioelementy.ru/articles/chto-takoe-rezistor/ (дата звернення: 25.02.2021).

6. Програмування Ардуіно arduino. Arduino - основи URL: https://lickeys.ru/uk/programmy/programmirovanie-arduino-arduino-arduino-osnovy-programmirovaniya/ (дата звернення: 24.05.2021).

7. Огляд і основи мови програмування C++ URL: http://www.znannya.org/?view=Cpp_basics (дата звернення: 24.12.2021).

8. Arduino Software (IDE) URL: https://www.arduino.cc/en/main/software (дата звернення: 24.05.2021).

9. Environment Начало работы с ArduinoURL: https://doc.arduino.ua/ru/guide/Environment (дата звернення: 24.05.2021).

10. Arduino и модули Bluetooth HC-05/06 // voltiq.ru URL: https://voltiq.ru/arduino-and-hc-05-hc-06/ (дата звернення: 13.11.2021).

11. Краткий обзор языка C# // docs.microsoft.com URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (дата звернення: 13.11.2021).

12. 6 причин, почему JetBrains Rider лучше чем Visual Studio // Софтлист URL: https://softlist.com.ua/articles/6-prichin-pochemu-jetbrains-rider-lyche-chem-visual-studio/ (дата звернення: 13.11.2021).

13. The MPU6050 Explained // Programming Robots URL: https://mjwhite8119.github.io/Robots/mpu6050 (дата звернення: 13.11.2021).

14. Стягивающие и подтягивающие резисторы. В чём различие // funny DIY URL: http://funnydiy.ru/1010 (дата звернення: 13.11.2021).

15. MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2 // invensense URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf (дата звернення: 13.11.2021).