**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет імені Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра комп'ютерної інженерії**

ДОПУЩЕНО ДО ЗАХИСТУ
В. о. завідувача кафедри,
канд. техн. наук, доцент
_____ Я. М. Крайник
« __ » _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

# «АЛГОРИТМИ МАШИННОГО НАВЧАННЯ ДЛЯ РОЗПІЗНАВАННЯ РУКОПИСУ HID-ПРИСТРОЯМИ»

Спеціальність 123 Комп'ютерна інженерія

123 – КР.1 – 405.21810518

**Студент:** _____ С. В. Овчар
*підпис*
«___» _____ 202___ р.

**Керівник:** старший викладач _____ В. В. Старченко
*підпис*
«___» _____ 202___ р.

Миколаїв – 2022

ЗАВДАННЯ

на виконання бакалаврської роботи

**НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!!**

ЗАРЕЗЕРВОВАНА Сторінка 1

ця сторінка після друку буде замінена

ЗАВДАННЯ

на виконання бакалаврської роботи

**НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!!**

ЗАРЕЗЕРВОВАНА Сторінка 2

ця сторінка після друку буде замінена

АНОТАЦІЯ

1 сторінка !!!!

**НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!**

ЗАРЕЗЕРВОВАНА Сторінка 1

ця сторінка після друку буде замінена

ABSTRACT

1 сторінка !!!!

**НЕ ВИДАЛЯТИ цю СТОРІНКУ з файлу !!!!!!!!!!!!!!!!**

ЗАРЕЗЕРВОВАНА Сторінка 2

ця сторінка після друку буде замінена

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

**ЛКМ** – ліва клавіша миші

**МН** – машинне навчання

**МРТ** – магнітно-резонансна томографія

**ПКМ** – права клавіша миші

**BLE** – Bluetooth Low Energy

**DMP** – Цифровий процесор руху (англ. Digital Motion Processor)

**FIFO** – Структура даних "Черга" (англ. First In First Out)

**HID** – Інтерфейс взаємодії з людиною (англ. Human Interface Device)

**IMU** – Інерційний вимірювальний пристрій (англ. Inertial Measurement Unit)

**IoT** – Інтернет речей (англ. Internet of Things)

**LQE** – Лінійно-квадратичне оцінювання (англ. Linear Quadratic Estimation)

**MEMS** – Мікроелектромеханічні системи (англ. Microelectromechanical systems)

**SDK** – Набір засобів розробки (англ. Software Development Kit)

**SiP** – Система в єдиному корпусі (англ. System in Package)

**USB** – Універсальна послідовна шина (англ. Universal Serial Bus)

# ВСТУП

Зростання потужності та зменшення габаритів електронних пристроїв – зміни, що стали головним трендом для електронних пристроїв XXI ст., дає змогу розробляти принципово нові види взаємодії людини з комп’ютером, знаходити способи виконання повсякденних задач новими, інноваційними шляхами.

Актуальність розробки подібних пристроїв у сучасну еру технологічного розвитку полягає в практичній необмеженості можливих способів використання мікроконтролерів для покращення життя людини. Незважаючи на малий розмір як самих обчислювальних пристроїв, так і сенсорів чи датчиків, що розташовані на тілі людини, отримання даних та комунікація як із внутрішньою системою датчиків, так і з навколишнім світом, відбувається безперервно. Одним із перспективних напрямів використання подібних пристроїв може бути розпізнавання рукопису для більш зручного набору тексту. Звичний форм-фактор клавіатур комп’ютерів, який, незважаючи на свою звичність та достатню для сучасного користувача швидкість набору тексту, негативно впливає на необхідність постійного перебування поруч із пристроєм для введення тексту, також займаючи щонайменше одну із кінцівок людини для безпосереднього процесу набору тексту. Використання сенсорів, розташованих на руці користувача, дасть змогу не тільки вводити текст. Аналіз досліджень попередніх років показує, що пристрої схожого форм-фактору також можуть бути використані під час медичної реабілітації [1].

**Мета БР:** Розробити програмно-апаратний комплекс бездротової клавіатури, що обладнаний необхідним набором датчиків для фіксування положення долоні користувача у просторі, на базі алгоритмів машинного навчання для розпізнавання жестів та подальшого виведення за допомогою інтерфейсу Bluetooth HID.

**Завдання БР** згідно з Індивідуальним планом та темою БР:

- виконати аналіз існуючих методів інтерактивної взаємодії людини з комп’ютером;
- дослідити особливості роботи наявних пристроїв HID-пристроїв;
- з наявної елементної бази обґрунтувати вибір оптимальних компонентів для реалізації HID-пристрою, що відповідає вимогам ергономіки;
- розробити принципову схему HID-пристрою на макетній платі;
- виготовити макетний зразок HID-пристрою, оптимізований для задачі розпізнавання жестів людини;
- розробити програмне забезпечення мікроконтролера HID-пристрою для первинної обробки даних сенсорів;
- розробити питання з охорони праці та безпеки життєдіяльності.

**Об’єкт дослідження**: Методи інтерактивної взаємодії людини з комп’ютером за допомогою HID-пристроїв.

**Предмет дослідження**: Ергономічний HID-пристрій, оптимізований для задачі розпізнавання жестів людини на базі мікропроцесору i.MX RT1160 та бібліотеки машинного навчання TensorFlow.

**Практичне значення** БР полягає у створенні принципово нового способу взаємодії людини з комп’ютером. Розроблений пристрій значно полегшуватиме використання комп’ютера людьми з обмеженими можливостями.

**Апробація**: Всеукраїнська науково-практична конференції «Могилянські читання–2021» (м. Миколаїв, 8 листопада 2021 р.). Основні результати роботи було опубліковано у збірнику тез конференції [2]. Прототип пристрою було представлено на конкурсі студентських проєктів «Startup BSNU 2020»[3].

# РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ ТА ПАТЕНТНОЇ ІНФОРМАЦІЇ ЩОДО ІСНУЮЧИХ ПРИСТРОЇВ УПРАВЛІННЯ ПК ЗА ДОПОМОГОЮ ЖЕСТІВ

## 1.1 Аналіз можливостей бібліотеки TensorFlow

Бібліотека TensorFlow – одна із найбільш поширених бібліотек для машинного навчання, доступна на станом на сьогоднішній день. Багато компаній та корпорацій усього світу впроваджують використання алгоритмів машинного навчання для покращення власних продуктів. Відомі способи використання бібліотеки TensorFlow для класифікації зображень будівель, покращення стійкості мережевих елементів, модерації онлайн-чатів, вивчення мов, створення шарів абстракції у мікроконтролерах і навіть ідентифікації об’єктів на скануваннях МРТ головного мозку [4].

Бібліотека TensorFlow Lite Micro, в свою чергу, створена для запуску моделей машинного навчання на мікроконтролерах та інших пристроях, що не обладнані великими об’ємами пам’яті для зберігання програмного коду моделей. Базове ядро бібліотеки для мікроконтролерів займає менш ніж 16 кБ пам’яті та може запускати примітивні моделі машинного навчання [5].

Впровадження алгоритмів машинного навчання у програмний код мікроконтролерів може покращити функціонування багатьох пристроїв, які вже зараз використовуються людьми – від IoT-пристроїв до звичайної персональної електроніки – без залежності від дорогого та потужного обладнання, на якому, зазвичай, запускаються стандартні програми з використанням МН. Це також значно підвищує рівень безпеки, тому що дані не відправляються у сторонні сервіси для подальшого аналізу – увесь цикл аналізу та обробки вхідних даних відбувається на стороні мікроконтролера.

Для запуску моделі TensorFlow необхідна 32-бітна апаратна платформа. Тестування алгоритмів командою розробників відбувається в основному на процесорах на базі ядра серії Arm Cortex-M, але дозволяється запуск на інших архітектурах, зокрема ESP32. Перелік плат розробки, які дозволяють

запуск моделей машинного навчання, постійно оновлюється. Станом на сьогодні функціонал перевірено розробниками на наступних платформах:

- Arduino Nano 33 BLE Sense;
- SparkFun Edge;
- STM32F746 Discovery kit;
- Adafruit EdgeBadge;
- Adafruit TensorFlow Lite for Microcontrollers Kit;
- Adafruit Circuit Playground Bluefruit;
- Espressif ESP32-DevKitC;
- Espressif ESP-EYE;
- Wio Terminal: ATSAMD51;
- Himax WE-I Plus EVB Endpoint AI Development Board;
- Synopsys DesignWare ARC EM Software Development Platform;
- Sony Spresense.

Бібліотека TensorFlow активно підтримується командою розробників та має велику кількість доступних для перегляду та запуску прикладів програмного коду, з яких можна починати розробку. Посилання на репозиторій з прикладами програмного коду наведено у Додатку А.

### 1.2 Аналіз обчислювальної здатності Teensy 4.0

Звичні та популярні серед розробників плати розробки Arduino не здатні виконувати задачі розпізнавання через малу потужність. Новітні мікроконтролери серії ESP32, хоча і обладнані мікроконтролерами з двома повноцінними ядрами більшої потужності, мають низький рівень надійності та захищеності пам’яті від несанкціонованого зчитування. Для розробки даного пристрою буде доцільним використання такого мікроконтролера, що матиме відносно невеликий розмір та більшу потужність і поєднанні з надійністю та безпечністю для використання

Підвищення швидкості проведення обчислень було вирішено за допомогою використання більш потужної плати розробки – Teensy 4.0.

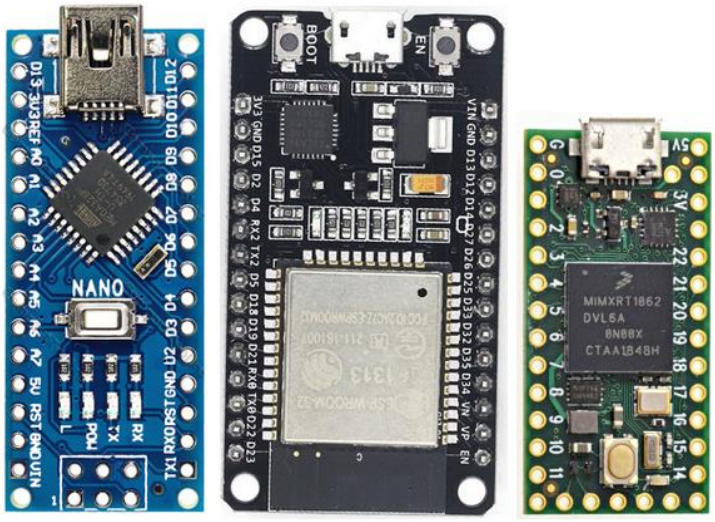


Рисунок 1.1 – З лівої сторони на праву - Arduino Nano, ESP32 devkit v1 та Teensy 4.0

Як бачимо, плата розробки Teensy 4.0 має найменший розмір серед трьох досліджуваних варіантів, та обладнана процесором NXP iMXRT1062 ARM Cortex-M7 з тактовою частотою 600 МГц, що є одним із найбільш потужних рішень для Embedded-систем, доступним сьогодні [6]:

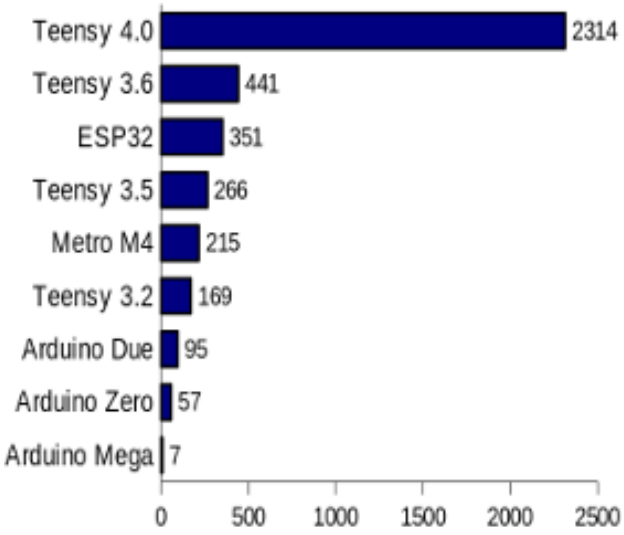


Рисунок 1.2 – Порівняння потужності мікроконтролерів з використанням CoreMark

Teensy 4.0 обладнана кросоверним мікропроцесором i.MX RT1160 – новим процесором i.MX сімейства RT, яке містить передові технології NXP –

впровадження високопродуктивного ядра Arm Cortex-M7, що працює на частоті до 600 МГц і енергоефективного ядра Cortex-M4 з частотою до 240 МГц [7]. Процесор i.MX RT1160 має 1 МБ вбудованої оперативної пам’яті загалом, включаючи 768 КБ RAM, яка може бути налаштована як TCM (512 КБ RAM спільно з M7 TCM і 256 КБ оперативної пам’яті спільно з M4 TCM) або ОЗУ загального призначення на чіпі. i.MX RT1160 інтегрує розширений модуль керування живленням регуляторами DCDC і LDO, що спрощує послідовність живлення. i.MX RT1160 також забезпечує різноманітні інтерфейси пам’яті, включаючи SDRAM, RAW NAND FLASH, NOR FLASH, SD/eMMC, Quad/Octal SPI, Hyper RAM/Flash та широкий спектр інших інтерфейсів для підключення периферійних пристроїв, таких як WLAN, Bluetooth™, GPS, дисплеї та датчики камери. i.MX RT1160 також має багаті аудіо та відео функції, включаючи MIPI CSI/DSI, PK-дисплей, графічний прискорювач, камеру інтерфейс, SPDIF та аудіо інтерфейс I2S.

### 1.3 Аналіз загальних принципів побудови Wearable HID-пристроїв

Технології розпізнавання рукопису як альтернативи традиційним засобам вводу беруть початок із кінця XX століття. Такі пристрої, як Pencept Penpad (див. рис. 1.3), хоч і презентували принципово нові способи введення даних, на жаль, не мали суттєвих переваг у порівнянні зі звичайними персональними комп’ютерами, і навіть, мали дещо більші розміри. Відсутність переваг таких альтернативних методів і стала причиною зупинки активної їх розробки.

Сучасні тенденції зменшення розмірів та підвищення потужності мікроконтролерів надають можливість розробки новітніх пристроїв, самі концепції яких нещодавно могли здаватися фантастикою. Так, концепція wearable-девайсів, яка виникла у другій половині XX ст. з появою електронних годинників-калькуляторів Hewlett-Packard, на даний момент еволюціонувала у різноманітні розумні годинники та фітнес-трекери, що

широко використовуються користувачами та мають високий попит завдяки функціональності та доступності.


Рисунок 1.3 – Термінал з рукописним вводом Pencept Penpad

Розробка новітніх методів взаємодії людини з комп’ютером за останні десятиліття значно активізувалась. Велика кількість науковців та розробників аналізують доступні на даний момент алгоритми розпізнавання жестів людини для використання їх як методів введення даних [8,9]. Увага приділяється також і пристроям бездротового керування, які, на відміну від звичних пультів дистанційного управління, реагують на команди, викликані певними жестами – рухами пристрою у просторі [10]. Проте розробці саме бездротових альтернатив клавіатурам та комп’ютерним мишам приділяється менше уваги, про що свідчить відсутність поширення пристроїв подібного форм-фактору.

Одним із небагатьох пристроїв, призначених для бездротового управління комп’ютерами, є Tap Strap 2 (див. рис. 1.4) – комбінація клавіатури та миші, розроблена компанією Tap Systems, Inc.

Пристрій обладнаний чотирма акселерометрами та оптичним сенсором комп’ютерної миші[11], що дозволяє використовувати його, залишаючи руку вільною.

Бездротова клавіатура MPBoard, конструкція якої може стати прикладом для подальшого аналізу, створена на основі плати розробки ESP32

Devkit V1, та використовувала дані, отримані з п’яти датчиків MPU6050, розташованих на кожному із пальців руки (див. рис. 1.5).

Клавіатуру MPBoard побудовано на основі мікроконтролера ESP32, що має вбудовані інструменти взаємодії за допомогою Bluetooth.

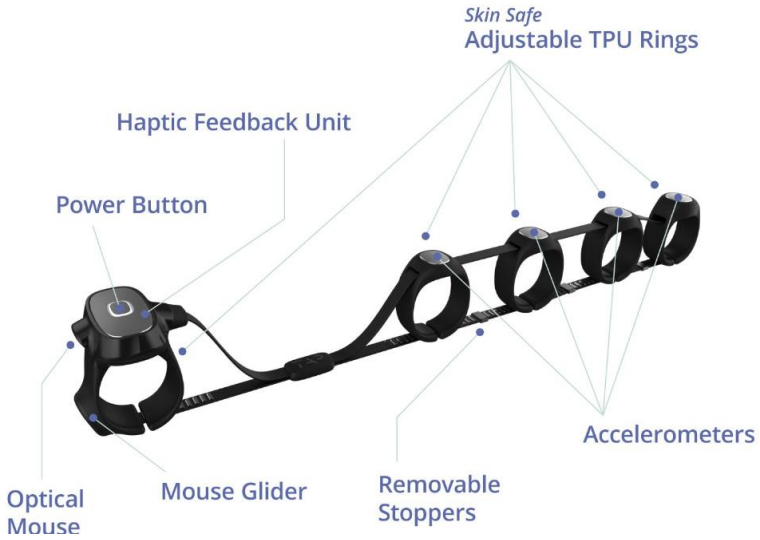


Рисунок 1.4 – Tap Strap 2

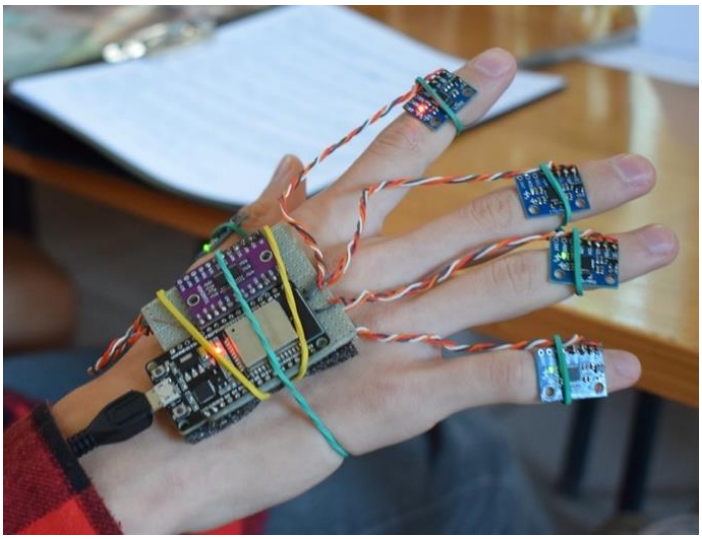

Рисунок 1.5 – Бездротова клавіатура MPBoard

Мікроконтролер, встановлений у MPBoard, отримував дані із гіроскопів MPU6050, розташованих на кожному із пальців руки. Детальну принципову схему з’єднання елементів наведено у Додатку Б. Дані, отримані з датчиків, проходили етап фільтрації методом рухомого середнього для

очищення від високочастотного шуму, та на основі аналізу відфільтрованих даних приймалось рішення про передачу інформації на комп’ютер.

Попри відносну простоту конструкції пристрою – мікроконтролер ESP32, I2C – мультиплексор TCA9548A та 5 модулів гіроскопів GY-521 на основі MPU6050, великий об’єм даних кожного опитування датчиків (30 чисел типу float, по 6 з кожного опитаного модуля акселерометра та гіроскопа) став причиною вимушеного завершення цього етапу розробки через складність детального аналізу даних стандартним апаратним забезпеченням – як 8-бітними мікроконтролерами AVR, які є основою більшості плат розробки Arduino, так і мікроконтролерами ESP32 [2].

Маючи на увазі проблеми попередньої версії прототипу пристрою, описані вище, було сформовано основні концепції, яких слід дотримуватись під час розробки нового wearable-пристрою, для бездротового управління комп’ютерами за допомогою алгоритмів розпізнавання жестів:

– мікроконтролер більшої потужності для підвищення можливостей обчислювання;

– простота принципової схеми, що дасть змогу зібрати пристрій на макетній платі без необхідності очікування друку плат промислової якості;

– зменшений об’єм даних кожного опитування сенсорів для підвищення швидкості роботи алгоритмів машинного навчання у подальшому;

– компактність та надання повної свободи рухів;

– повноцінна заміна традиційних методів введення даних.

Для проектування принципової схеми пристрою доцільно використати відкриту програму EasyEDA. Це програмне забезпечення надає розробникам можливість проектувати схеми електронних пристроїв у веб-браузері без необхідності встановлення будь-яких програм на персональний комп’ютер [12]. Власний досвід проектування принципових схем прототипів пристроїв для слідкуванням за положенням руки у просторі показує, що даний

комплекс програм повністю задовольняє потреби проектувальника, а розвинена спільнота користувачів, що додають до бібліотеки створені власноруч компоненти та елементи схем, значно прискорює процес розробки.

Прототип пристрою повинен містити чотири датчики Холла для реагування на рухи пальців руки, та модуль гіроскопа та акселерометра MPU-9250, необхідний для розпізнавання рухів руки у просторі. Розроблена принципова схема може бути перетворена на схему друкованої плати, проте виготовлення друкованих плат та їх доставка від заводу-виробника до замовника може займати достатньо великий проміжок часу [13], впродовж якого подальший процес розробки зупиняється до моменту отримання друкованих плат. Тому доцільним вважається зібрати прототип на макетній платі, у разі необхідності вносити зміни до принципової схеми, а друковану плату проектувати вже на фінальному етапі розробки.

У випадку з даним прототипом пристрою це ніяк не впливає на функціонал, тому можна вважати доцільним використання макетної плати замість повноцінної друкованої плати.

### 1.4 Аналіз алгоритмів фільтрації даних з датчиків

Дані, отримані з гіроскопа, містять 9 значень – зчитані дані з трьох осей акселерометра, гіроскопа та магнетометра. Достатнім для подальшого аналізу положення руки буде обчислення на основі отриманих даних кутів повороту гіроскопа у просторі.

Визначити кути повороту можна різними способами. Одними із найбільш популярних є кватерніони та кути Ейлера, і через простоту останніх доцільно обрати саме їх як основний спосіб описання положення долоні у просторі.

Кути Ейлера представляють поворот тіла як результат послідовних поворотів навколо трьох осей, що пов’язані з тілом. Порядок поворотів при цьому маж значення, оскільки зміна порядку виконання поворотів дає новий результат. Використання кутів Ейлера популярне через простоту та

наочність, проте у деяких стандартних задачах управління орієнтацією описання поворотів за трьома осями є недостатнім [14,15]:

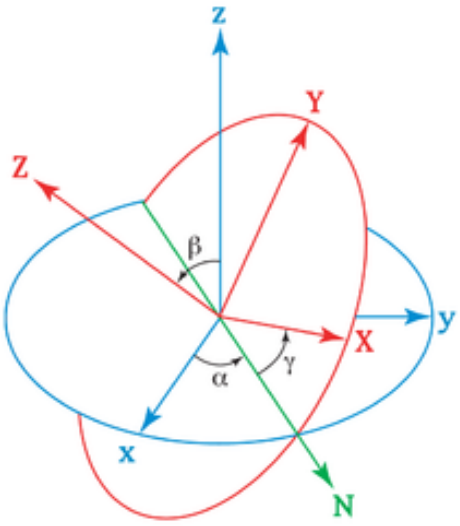


Рисунок 1.6 – Кути Ейлера

Дані, отримані з гіроскопа, описаним вище чином перетворюються у три кути повороту, разом із бінарними значеннями, отриманими з датчиків Холла, в подальшому будуть відправлені на комп’ютер. Проте показання датчика містять в собі високочастотний шум, який може погіршити якість аналізу (див. рис. 1.7).

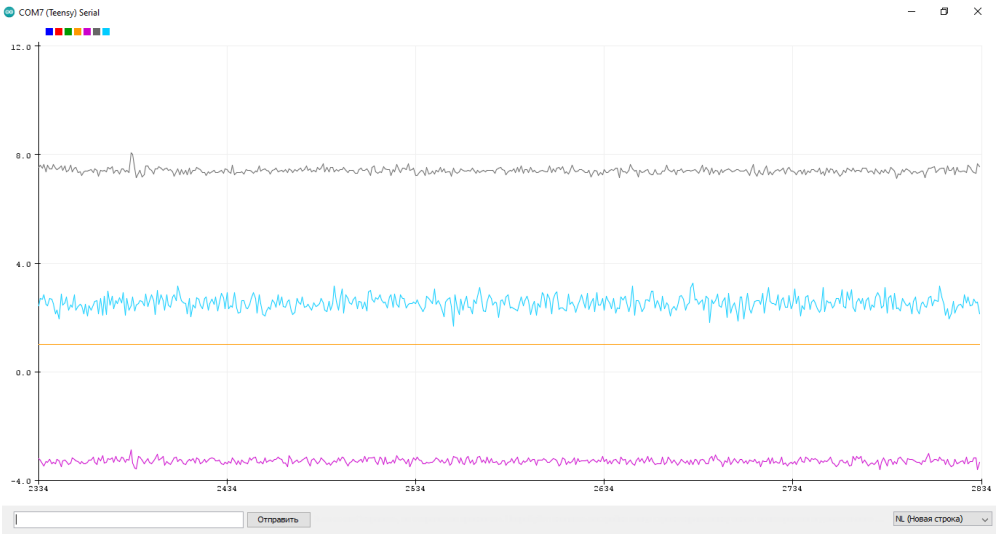


Рисунок 1.7 – Дані до застосування фільтрації

Для покращення роботи алгоритмів розпізнавання буде доцільним застосування до отриманого набору даних алгоритм фільтрації, що матиме

достатню швидкість реакції, невеликий час фільтрації та об’єм пам’яті, необхідний для збереження змінних для фільтра. Попередня обробка вхідних даних в подальшому полегшить процес тренування нейронних мереж.

Цифрові (програмні) фільтри дозволяють відфільтрувати різноманітні шуми. Одними із найбільш популярних фільтрів, що застосовуються в електроніці, є фільтр Калмана та так зване «рухоме середнє».

Фільтр Калмана – це фільтр, основний принцип якого полягає у розумінні фізики самого явища шуму. Алгоритм Калмана (LQE), досліджує серію вимірювань, що спостерігаються протягом тривалого часу, включаючи статистичний шум та інші неточності, і дає оцінки невідомих змінних, які, як правило, більш точні, ніж ті, що ґрунтуються на одному вимірюванні [16].

Фільтр Калмана ефективно справляється з невизначеністю через шумові дані датчика і, певною мірою, із випадковими зовнішніми факторами.

Фільтрація за допомогою рухомого середнього — це обчислення для аналізу точок даних шляхом створення серії середніх різних підмножин повного набору даних. Варіації включають: прості, накопичувальні або зважені форми фільтрів на основі рухомого середнього [17].

Для виконання задачі фільтрації даних з гіроскопа можна обрати алгоритм експоненціального рухомого середнього. Алгоритм експоненціального рухомого середнього, на відміну від класичного алгоритму фільтрації методом рухомого середнього, набагато більш оптимальний у плані реалізації.

Замість збереження декількох останніх вимірювань, експоненціальний фільтр тримає у пам’яті лише одне попереднє значення, до якого вже було застосовано фільтрацію. Нове отримане значення, надходячи до функції фільтрації, віднімається від збереженого, та отримана різниця множиться на певний коефіцієнт – число з рухомою комою зі значенням від 0 до 1. Добуток різниці значень та коефіцієнта додається до збереженого у пам’яті значення. Результат фільтрації наведено на рисунку 1.8

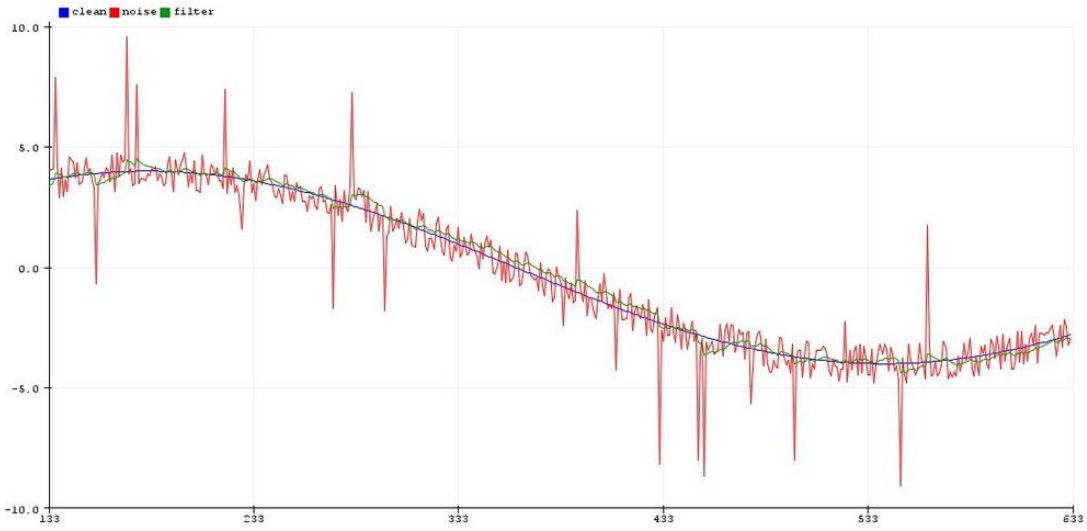


Рисунок 1.8 – Результат застосування фільтрації до вхідних даних

Як бачимо, випадкові високочастотні шуми датчика зникають, і дані мають вигляд, придатний для подальшого аналізу.

### 1.5 Висновки до розділу 1

Порівняння наявних на споживчому ринку пристроїв для розпізнавання жестів, а також огляд розробленого прототипу пристрою MPBoard вказав на можливі шляхи спрощення конструкції пристрою та зменшення об’єму даних для аналізу, замінивши надлишкові модулі акселерометрів та гіроскопів на датчики Холла, що надаватимуть необхідні дані про положення пальців руки користувача, зменшуючи при цьому кількість даних та дозволивши аналізувати отримані з датчиків дані без використання нейронної мережі, залишивши для обчислень за допомогою нейронних мереж лише дані, отримані з одного датчика MPU-9250. Аналіз бібліотеки для реалізації алгоритмів машинного навчання для архітектури мікроконтролерів TensorFlow підтвердив рішення про доцільність використання саме обраної бібліотеки для реалізації моделей розпізнавання жестів. Порівняльний аналіз алгоритмів фільтрації вхідних даних з датчиків виявив, що найбільш дієвим та простим у реалізації попередньої обробки даних з акселерометра та гіроскопа є алгоритм експоненціального ковзного середнього.

# РОЗДІЛ 2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ HID-ПРИСТРОЮ

Маючи на увазі результати аналізу стеку технологій, що найчастіше використовуються у розробці програмного та апаратного забезпечення для вбудованих систем, було розпочато розробку прототипу пристрою, що буде здатний точно визначати положення та рухи руки користувача, надаючи можливість розпізнавання жестів на основі отриманих з датчиків даних.

## 2.1 Датчики Холла для визначення позиції пальця руки

У попередньому розділі було визначено, що для визначення позиції пальців руки користувача доцільно використати датчики ефекту Холла. Таке рішення, на відміну від використання IMU-сенсорів спрощує конструкцію пристрою та значно зменшує навантаження на мікроконтролер.

Датчики Холла – це пристрої, які вимірюють наявність та силу магнітного поля поруч із ними. Названий цей тип датчиків на честь американського фізика Едвіна Холла, який у 1879 році опублікував статтю, в якій було описано появу різниці потенціалів, яку сучасна наука називає напругою Холла, на електричному провіднику, яка є поперечною до електричного струму в провіднику та до прикладеного магнітного поля, перпендикулярного до струму (див. рис. 2.1) [18].

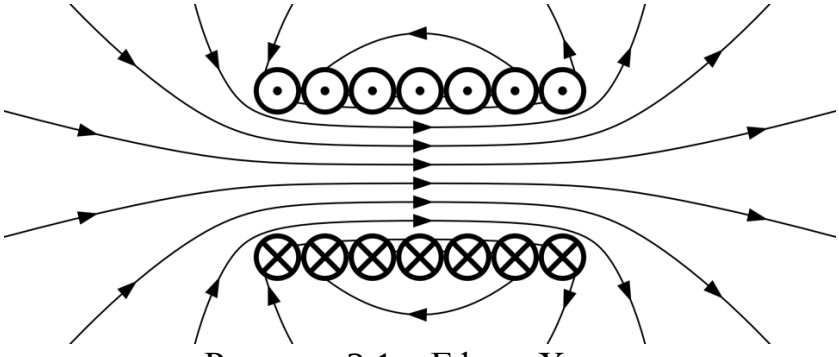

Рисунок 2.1 – Ефект Холла

Таким чином, використовуючи зв’язок магнітного поля та різниці потенціалів на кінцях провідника та вимірюючи значення напруги Холла,

можна визначити, чи присутнє поруч із провідником магнітне поле, та отримати приблизне значення сили магнітного поля, що діє на провідник.

На сьогоднішній день для створення провідників, що використовуються у датчиках Холла, використовуються наступні матеріали [19]:

- арсенід галію (GaAs);
- арсенід індію(InAs);
- фосфід індію (InP);
- антимонід індію (InSb);
- графен.

Використання цих матеріалів дає найбільшу чутливість до визначення рухливості електронів [19].

Існують два типи датчиків Холла – біполярні та уніполярні. Особливість біполярних датчиків Холла полягає у використанні позитивного магнітного поля (південного полюсу магніту) для підвищення напруги на виводах, та негативного магнітного поля (північного полюсу) для зменшення напруги. Уніполярні датчики потребують лише взаємодії з позитивним магнітним полем, реагуючи на відстань від датчика до магнітного поля [20].

Виробники сучасних датчиків Холла також надають можливість отримання даних з датчика у цифровому та аналоговому вигляді. Аналогові датчики Холла змінюють напругу в залежності від сили магнітного поля, що діє на провідник, у той час цифрові датчики Холла виводять бінарний результат – «0» або «1» в залежності від наявності та сили магнітного поля, яке діє на датчик.

Хоча потужність процесора, встановленого на платі Teensy 4.0, є достатньою для виконання маніпуляцій з числами з плаваючою точкою, необхідність застосування додаткової фільтрації до отриманого сигналу з кожного із датчиків ускладнює алгоритм програми для мікроконтролера. Тому доцільно використати цифровий датчик Холла, дані з якого не

потребуватимуть обробки та будуть придатні для аналізу одразу після їх отримання. Одним із найбільш розповсюджених цифрових датчиків Холла, які можна використати у вбудованих системах з малою напругою, є датчик A3144, який і було використано у проєкті.

Датчик A3144 виготовляється у різноманітних корпусах, одним із яких є корпус TO92 (див. рис. 2.2), що дозволяє використання A3144 на макетних платах та при навісній пайці.

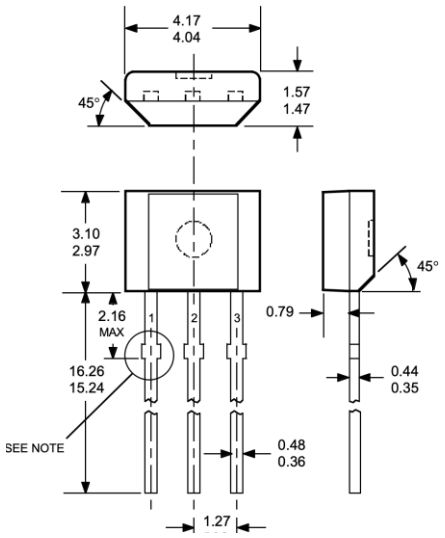


Рисунок 2.2 – Розміри датчика A3144 у корпусіTO92

Використання транзистора перед третім виводом датчика (див. рис. 2.3) дозволяє отримати вихідне значення у бінарному форматі.

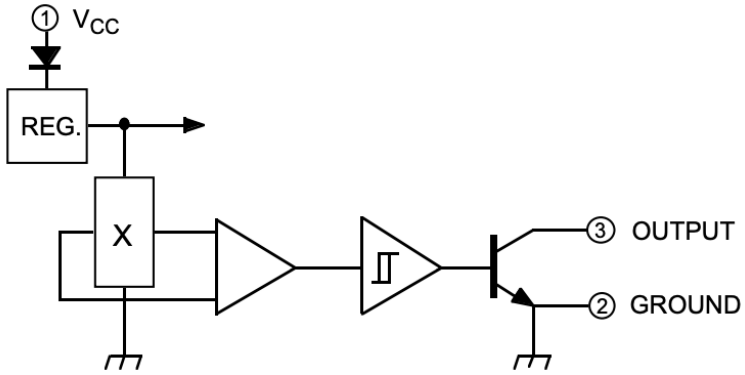


Рисунок 2.3 – Функціональна блокова діаграма датчика A3144

Стандартна схема підключення датчика до будь-якого пристрою передбачає використання pull-up резистора між виводами VCC та OUTPUT

(див. рис. 2.4), який встановлює позитивне значення виводу за умови закритого транзистора на виводі датчика. Коли транзистор відкрито, наявність резистора запобігає виникненню короткого замикання на ділянці схеми. Відсутність опору між заземленням ділянки схеми та виводом датчика дозволяє встановити низький логічний рівень.

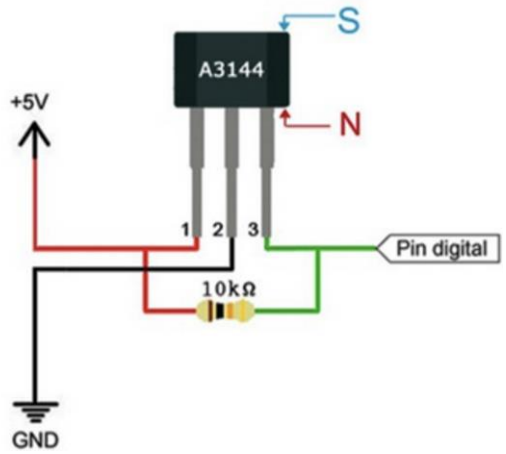


Рисунок 2.4 – Схема підключення датчика A3144

Використовуючи схему підключення, вказану на рисунку 2.4, можна під’єднати до загальної схеми пристрою чотири датчики ефекту Холла, розташовані на кожному із пальців руки, що дасть змогу визначати їх положення.

### 2.2 IMU-сенсор для слідкування за рухами руки

Слідкування за рухами руки, окрім необхідності опитування датчиків Холла на пальцях руки, також потребує наявності датчика, опитування якого даватиме інформацію про положення у просторі, напрям та швидкість руху. Зазначені властивості мають IMU-сенсори, що являють собою комбінацію акселерометра та гіроскопа. В деяких випадках до комбінації акселерометра та гіроскопа додається магнітометр, здатний визначати абсолютне положення у просторі відносно магнітного поля Землі [1].

Акселерометр вимірює проекцію повного прискорення – рівнодіючу сил негравітаційної природи, яка діє на інертну масу, обчислюючи величину прискорення в залежності від величини цієї маси. Акселерометр входить до

складу інерційних навігаційних систем, які інтегрують отримані виміри, визначаючи інерційну швидкість та координати носія.

Сучасна конструкція акселерометрів (див. рис. 2.5) створена на основі MEMS, що складаються з інертної маси та датчиків. Рух інертної маси в заданому напрямку та зміна відстані від маси до датчиків дає можливість визначити величину прискорення в цьому напрямку. Акселерометр, розміщений вертикально, також вимірює гравітаційне прискорення землі (приблизно 9.8м/с$^2$).

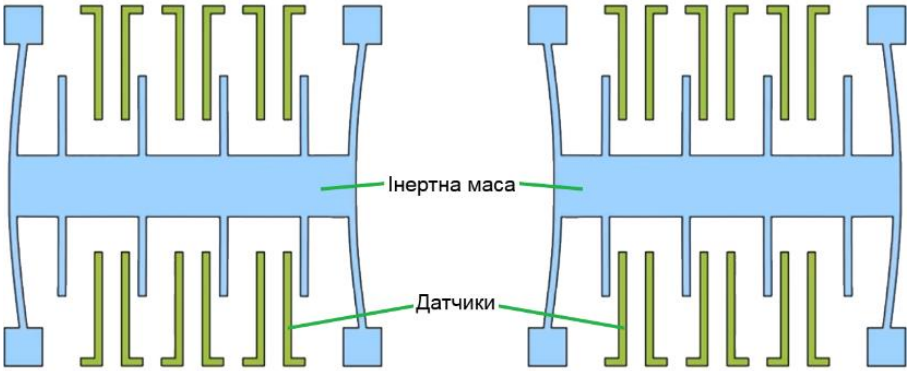


Рисунок 2.5 – Конструкція сучасного MEMS-акселерометра

Розташування трьох акселерометрів в перпендикулярних площинах, можемо отримати дані про прискорення, що діють на сенсор, відносно трьох осей, і, порівнявши отримані значення, визначити кути нахилу сенсора.

IMU-сенсор, який використано в проєкті – MPU9250 – це друге покоління 9-осьового модуля обробки руху для смартфонів, планшетів, вбудованих датчиків та інших пристроїв [21]. MPU-9250 поставляється в стандартному для індустрії корпусі QFN 3x3x1 мм, що дозволяє інженерам швидко розробити проєкти з використанням датчика.

Пристрій MPU-9250 споживає 9,3 мкА енергії і має на 44% менший розмір, ніж пристрій першого покоління компанії. Ефективність шуму гіроскопа в 3 рази краща, а повний діапазон компаса більш ніж у 4 рази кращий, ніж у конкурентних пропозицій [21].

MPU-9250 — це SiP, яка поєднує в собі два модулі: MPU-6500, який містить 3-осьовий гіроскоп, 3-осьовий акселерометр і вбудований DMP,

здатний обробляти. складні алгоритми MotionFusion; і AK8963, 3-осьовий цифровий компас. MPU-9250 підтримує перевірений на ринку MotionFusion від InvenSense. Єдина конструкція може підтримувати MPU-9250 або MPU-6500, надаючи гнучкість для підтримки будь-якого пристрою в різних продуктах.

MPU-9250 включає в себе MotionFusion і мікропрограму для калібрування під час виконання, що дозволяє виробникам побутової електроніки комерціалізувати економічно ефективну функціональність на основі руху.

На основі сенсора MPU-9250, завдяки вказаним вище перевагам, створено велику кількість модулів для макетування (див. рис. 2.6), які дають змогу обладнати будь-який прототип пристрою IMU-сенсором.


Рисунок 2.6 – Модуль GY-9250 на основі сенсора MPU-9250

Комунікація з датчиком відбувається за допомогою інтерфейсу I2C (Inter-Integrated Circuit Bus), перша специфікація якого була опублікована у 1992 році, і, завдяки простоті реалізації, шини I2C стали одними із найбільш популярних та використовуються у більшості вбудованих систем.

Інтерфейс пристроїв I2C є однією із переваг цієї шини та складається лише із двох виводів – SCL (Serial Clock Line) та SDA (Serial Data Line). SCL призначений для зв’язку з лінією синхронізації, а SDA – для зв’язку з лінією

передачі послідовних даних. Обидві лінії під’єднані до напруги живлення через pull-up резистори (див. рис. 2.7).

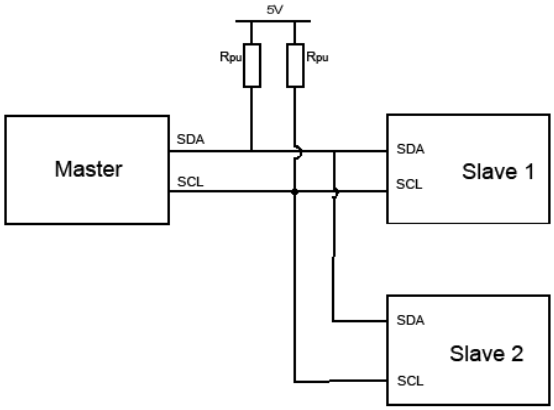


Рисунок 2.7 – Шина I2C

Кожен slave-пристрій повинен мати адресу, за якою до нього можна буде звернутись. Найчастіше використовується 7-бітна адресація, тобто кількість пристроїв для під’єднання обмежена і становить 128 пристроїв на одну шину.

Цикл передачі даних складається з генерації сигналу про старт передачі, вказання адреси пристрою, якому адресоване звернення, інформації про метод комунікації та самих даних, зчитування кожного байту з котрих обов’язково завершується бітом підтвердження зчитування, та сигналу про завершення передачі даних (див. рис. 2.8).

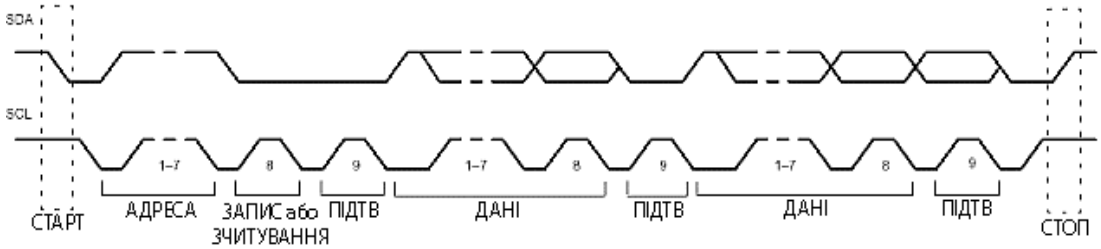


Рисунок 2.8 – Цикл передачі даних шиною I2C

Після генерації стану «СТАРТ» master-пристрій відсилає перший байт, що тримає в собі 7 біт адреси та вказівку, зчитувати дані чи передавати. Після цього slave-пристрій, що має вказану адресу, повинен відправити біт підтвердження, після чого починається процес передачі інформації. Коли всі

байти було успішно передано, або передача певного байта не була завершена генерацією біту підтвердження, master-пристрій завершує передачу та звільняє лінію.

Враховуючи описані вище характеристики та особливості сенсора MPU-9250, а також простоту комунікації з датчиком з використанням шини I2C, до прототипу пристрою було додано модуль IMU-сенсора GY-9250.

### 2.3 Розробка ергономічного прототипу пристрою

В результаті аналізу загальних принципів побудови HID-пристроїв, описаного у попередньому розділі, для проєктування принципової схеми було обрано середовище розробки EasyEDA (див. рис. 2.9) – веб-набір інструментів EDA, який дозволяє інженерам-технікам проєктувати, моделювати, ділитися – публічно та приватно – та обговорювати схеми, моделювання та друковані плати. Інші функції включають створення переліку матеріалів, файлів Gerber, а також файли вибору й розміщення та документальних виходів у форматах PDF, PNG та SVG.

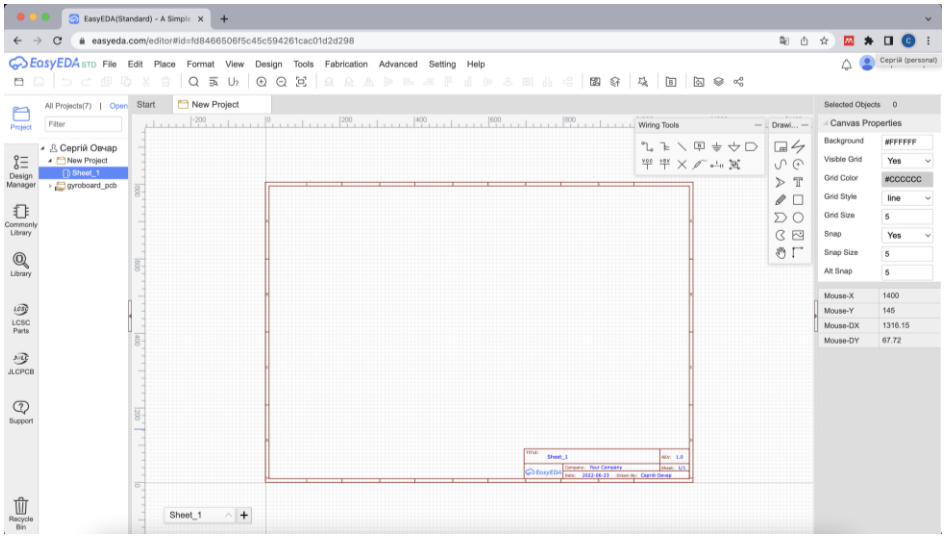


Рисунок 2.9 – Середовище проєктування EasyEDA

EasyEDA дозволяє створювати та редагувати принципові схеми, SPICE моделювання змішаних аналогових і цифрових схем, а також створювати та редагувати макет друкованих плат і, за бажанням, виготовлення друкованих плат [22].

Зареєстровані користувачі можуть безкоштовно завантажувати файли Gerber з інструменту, також EasyEDA пропонує послугу виготовлення друкованих плат. Ця служба також може приймати введення файлів Gerber від сторонніх інструментів.

Конструкція пристрою передбачає наявність чотирьох цифрових датчиків Холла A3144, та один модуль IMU-сенсора на базі SiP MPU9250. Враховуючи наявність резисторів підтягування для шини I2C на самому модулі GY-9250, а також використання лише одного пристрою на шині, буде достатньо використання модуля без необхідності додаткових резисторів підтягування на I2C-шині.

На платі розробки Teensy 4.0 є доступ до двох шин I2C (див. рис. 2.10), та кожен із виводів може бути використаний у якості GPIO-виводу. Для забезпечення компактності макету пристрою було вирішено використати шину I2C0 (виводи 18 та 19 – SDA0 та SCL0 відповідно), та виводи 20-23 для підключення датчиків Холла на кожному із пальців.

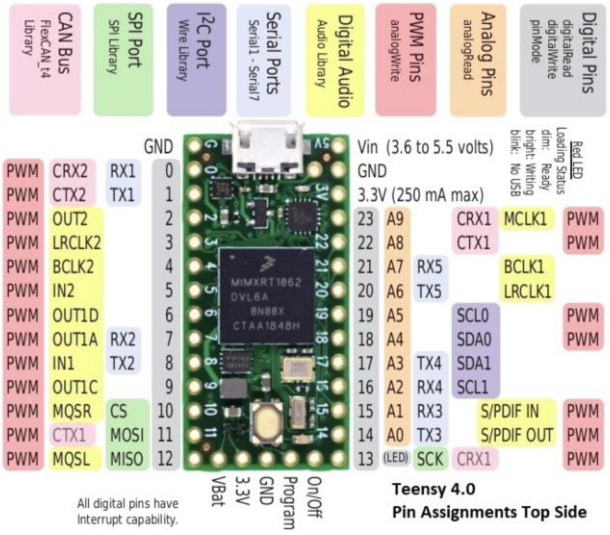


Рисунок 2.10 – Доступні інтерфейси на платі розробки Teensy 4.0

Датчики A3144 не розраховані на використання із напругою живлення 3.3 вольти, тому було вирішено організувати живлення плати від USB та використати для живлення датчиків Холла вивід Vin, на якому доступна напруга живлення плати без перетворювачів та стабілізації.

Згідно з описаними вище критеріями, було розроблено принципову схему пристрою (див. рис. 2.11), що містить необхідні елементи, правильно пов’язані між собою. На основі даної схеми можна розпочинати розробку ергономічного прототипу пристрою:

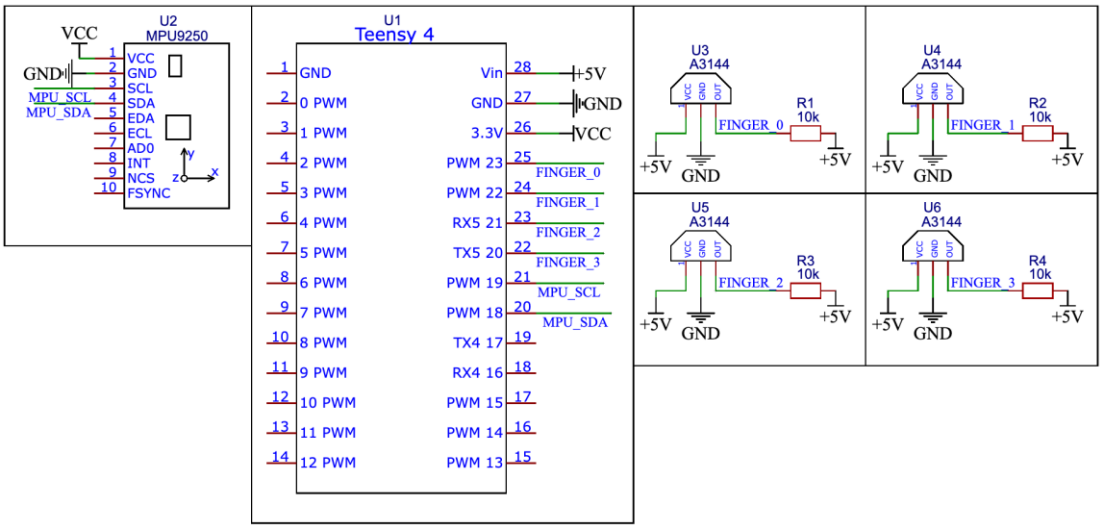


Рисунок 2.11 – Принципова схема пристрою

Для зручного розміщення датчиків Холла на пальцях руки необхідно розробити корпус, в якому буде розміщено датчики та магніти. Корпус не має перешкоджати вільним рухам. Розміщені в корпусі елементи мають бути статично закріплені.

Для розробки даного елементу було використано середовище розробки Fusion 360 – безкоштовне програмне забезпечення від компанії Autodesk. Це програмне забезпечення дозволяє створювати тривимірні моделі, модифікувати відкриті моделі, завантажені користувачами ресурсу, а також створювати задачі для ЧПУ-установок на основі створених моделей.

Для прототипу ергономічного HID-пристрою було створено дві тривимірні моделі – корпус для магніту діаметром 3 мм, та корпус для датчика Холла A3144 (див. рис.2.12). Дані моделі у подальшому були роздруковані на 3D-принтері з використанням полілактидної мононитки. Чотири копії розробленої моделі корпусу в подальшому було встановлено на прототип.

На основі принципової схеми було розроблено прототип пристрою, частини якого були закриті у корпус (див. рис. 2.12). Остаточний вигляд конструкції пристрою наведено на рисунку 2.13.

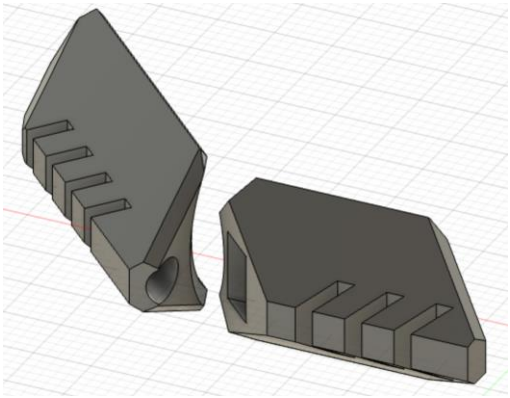

Рисунок 2.12 – Корпус для магніту та датчика Холла

Рисунок 2.13 – Прототип ергономічного HID-пристрою

## 2.4 Висновки до розділу 2

Робота, описана в даному розділі, була зосереджена на створенні прототипу ергономічного HID-пристрою. Результатом роботи став готовий прототип пристрою, який містить чотири датчики Холла на кожному із пальців руки, та IMU-сенсор для отримання даних про рухи та положення руки користувача у просторі. Розроблений прототип відповідає критеріям, визначеним у Розділі 1, та придатний для впровадження алгоритмів машинного навчання для розпізнавання жестів.

# РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ПРИСТРОЮ ДЛЯ РОЗПІЗНАВАННЯ ЖЕСТІВ

Створений прототип пристрою має достатній для організації розпізнавання жестів функціонал. Для впровадження алгоритмів машинного навчання необхідно розробити програмний код для мікроконтролера, за допомогою якого буде зібрано набір даних для тренування нейронної мережі. Отримані набори даних повинні бути проаналізовані програмою, яка перетворить вхідний датасет у готову для впровадження модель нейронної мережі. Компактна модель нейронної мережі, у свою чергу, за допомогою бібліотек TensorFlow Lite Micro для фреймворку Arduino надаватиме можливість взаємодії з комп’ютером в режимі реального часу з використанням управління жестами.

## 3.1 Програмний код для збору тренувальних даних

Програми для мікроконтролера, які будуть використані під час навчання та безпосереднього використання нейронної мережі, матимуть єдине джерело вхідних даних – отримана з сенсора MPU9250 інформація про кути повороту руки та прискорення, що діє на неї. Тому буде доцільним поєднати вказаний функціонал в єдину кодову базу, надавши спільний доступ до комунікації з датчиками обом підтипам програм.

Реалізація такого функціоналу доступна з використанням директив препроцесора мови C++. Директива #define дозволяє встановити символьний опис значенню, яке на етапі компіляції буде замінене на константне. Додаткове використання умовного оператора #ifdef дає змогу на етапі компіляції визначити ділянки коду, які не будуть використані у програмі. Таким чином, можна у одному проєкті створити дві програми, виконання яких буде залежати від встановлених директивами #define значень.

Програма для збору та відправки тренувальних даних буде виконана, якщо в проєкті встановлено прапорець #define TRAINING_MODE. Цей

режим роботи програми передбачатиме збір та попередню обробку даних, які за допомогою бібліотеки Serial буде відправлено для подальшого аналізу.

Для отримання та обробки даних з датчиків було вирішено використати відкриту бібліотеку MPU9250_DMP від SparkFun [23]. Ця бібліотека використовується для організації спілкування з датчиком, а також використання вбудованого модуля DMP, що дозволяє обчислити кути Ейлера для модуля у даний момент часу, застосувавши алгоритми фільтрації до отриманих значень.

Ініціалізація та комунікація з датчиком відбувається за допомогою стандартної бібліотеки Wire. Із частотою, заданою в коді програми завдяки функціям setSampleRate та dmpBegin, до FIFO-буферу датчика будуть надходити дані з результатами обчислень. Дані про кути повороту датчика також надходять до буферу. Зберігаючи інформацію про попереднє значення кутів повороту, а також вимірюючи час між вимірами, можна отримати приблизне значення кутової швидкості – відношення зміни кута повороту руки користувача до відрізку часу, за який надійшли нові дані:

$$\omega = \frac{\Delta\varphi}{\Delta t}, \tag{3.1}$$

де $\Delta\varphi$ – різниця між попереднім та поточним значенням кута повороту;

$\Delta t$ – час надходження нових даних з датчика.

Дані, отримані з датчика, будуть використані як під час тренування нейронної мережі, так і під час використання пристрою за призначенням. Тому ділянка програмного коду із викликами функцій опитування сенсорів та обчислення поточного положення руки в просторі була додана як загальна частина програмного коду, яка буде виконуватись кожну ітерацію головного циклу. Наступні дії, що пов’язані з форматуванням та відправкою даних для подальшого аналізу, було обгорнуто у директиву #ifdef TRAINING_MODE, яка виконуватиметься лише за умови наявності в коді програми директиви

#define TRAINING_MODE. У всіх інших випадках компіляція наступної ділянки коду не відбуватиметься.

Для того, щоб програма отримала інструкцію щодо необхідності запису інформації, кожен рядок, що містить результати обчислень та передається на комп’ютер, повинна починатися з унікального символу.

Дані про положення пальців руки подаються для подальшого обчислення у вигляді символів: «.» та «|» в залежності від поточного положення. Для запобігання виникненню помилок під час збору даних, що можуть бути викликані схожістю чисел «0» та «1» на дані акселерометра, було вирішено використати саме такі символи, що будуть однозначно трактуватися під час розробки програмного забезпечення.

Дані, отримані з MPU9250 – прискорення, яке вимірює акселерометр, та кути повороту відносно вертикальної, поперечної та повздовжньої осей (осей крену, рискання та тангажу) – після проведення обчислень також додаються до рядка з інформацією.

Завершується рядок числом, яке позначає кількість мікросекунд, що пройшла між вимірами.

Встановивши значення частоти оновлення датчика у 60 Гц, отримуємо рядки із результатами вимірювань (див. рис. 3.1), які досить легко відрізнити програмно та провести необхідні обчислення.

```
>|,|,|,|,0.9838,0.1668,0.0501, 0.0249,-0.2808, 0.8369,15207
>|,|,|,|,0.9838,0.1666,0.0499, 0.0200,-0.2930, 0.8384,14424
>|,|,|,|,0.9838,0.1665,0.0501,-0.0215,-0.2974, 0.8389,15207
>|,|,|,|,0.9838,0.1665,0.0505,-0.0205,-0.2930, 0.8516,15206
>|,|,|,|,0.9838,0.1664,0.0508,-0.0068,-0.2866, 0.8359,14426
```

Рисунок 3.1 – Результат роботи програми для опитування датчиків

Отримавши остаточний формат передавання даних, кількість та значення отриманих символів, а також організувавши їх передавання

послідовним портом з використанням стандартної бібліотеки Serial, можна перейти до етапу збору датасетів для тренування нейронної мережі.

### 3.2 Збір наборів даних для тренування нейронної мережі

Для реалізації збору даних з датчиків, та підготовки цих даних до подальшого тренування нейронних мереж, доцільно створити застосунок мовою програмування Python – інтерпретованою мовою високого рівня з динамічною типізацією. Завдяки простоті реалізації програм мова програмування Python розповсюджена серед веб-розробників, розробників ігор та спеціалістів із Data Science [24].

Алгоритм дій програми для отримання наборів даних наступний:

1. з’єднання з мікроконтролером для отримання вимірів;
2. обчислення кутової швидкості руху руки користувача на основі отриманих даних;
3. реєстрація жесту за умови різкого підвищення кутової швидкості;
4. закінчення жесту за умови зниження швидкості руху руки;
5. візуалізація отриманого жесту для перегляду та збереження;
6. зв’язування жесту із літерою, яку він означає;
7. збереження жесту до бази даних для тренування.

Базовий графічний інтерфейс побудовано з використанням стандартної бібліотеки Python – Tkinter. Ця бібліотека є частиною SDK Python, не потребує встановлення додаткових модулів, та надає розробникам можливість створення графічних інтерфейсів для будь-яких програм.

Отримання доступу до послідовного порту мікроконтролера здійснюється за допомогою бібліотеки pyserial. Вона надає можливість перегляду доступних портів для підключення, з’єднання з обраним портом, та отримання даних з нього.

Отримані з послідовного порту мікроконтролера дані надалі використовуються для візуалізації жестів. У вікні програми розташовано

зображення, яке змінюється в залежності від положення кожного із пальців руки, в реальному часі відображаючи це на зображенні (див. рис. 3.2).

Активація режиму запису жестів відбувається, коли користувач згинає усі пальці, окрім вказівного.

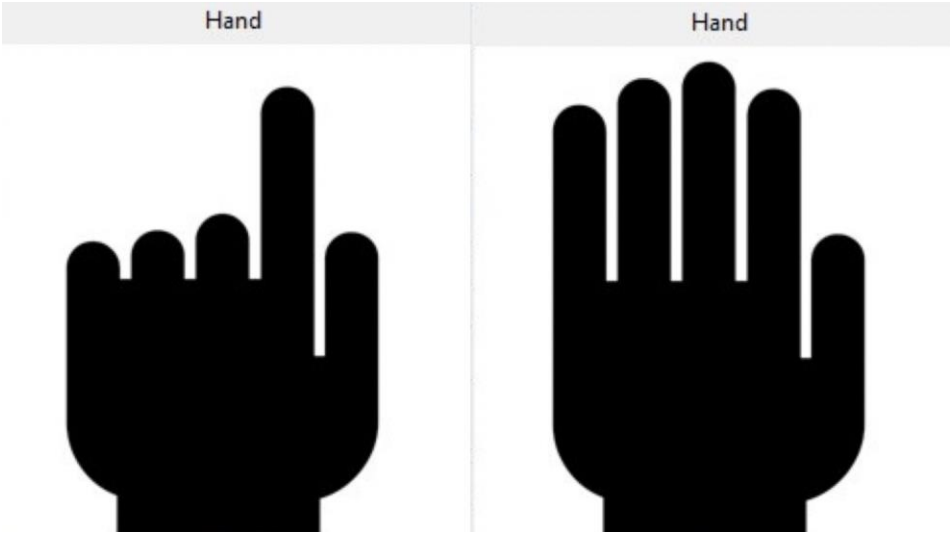


Рисунок 3.2 – Візуалізація руки користувача, з лівої сторони на праву – режим запису жестів, відсутність запланованих операцій

Поруч із візуалізацією руки користувача знаходиться перелік доступних для тренування символів (див. рис. 3.3). Поруч із самим символом вказується кількість зареєстрованих жестів.

| | Count |
|---|---|
| Glyph 'a' | 87 |
| Glyph 'b' | 0 |
| Glyph 'c' | 0 |
| Glyph 'd' | 0 |

Рисунок 3.3 – Перелік доступних для тренування символів

Коли режим запису жестів активовано (див. рис. 3.2), програма розпочинає аналіз отриманих з мікроконтролера даних. Коли кутова швидкість, обчислена за формулою (3.1), перевищує порогову швидкість для початку запису жесту, отримані з датчиків дані зберігаються та з’являються на графіку (див. рис. 3.4), який візуалізує виконання жесту. Завершення

запису жесту відбувається, коли обчислена кутова швидкість стає меншою за встановлений мінімум. При завершенні запису графік з його відображенням масштабується до розміру вікна та додається до загального переліку жестів, зареєстрованих для обраного символу (див. рис. 3.5).

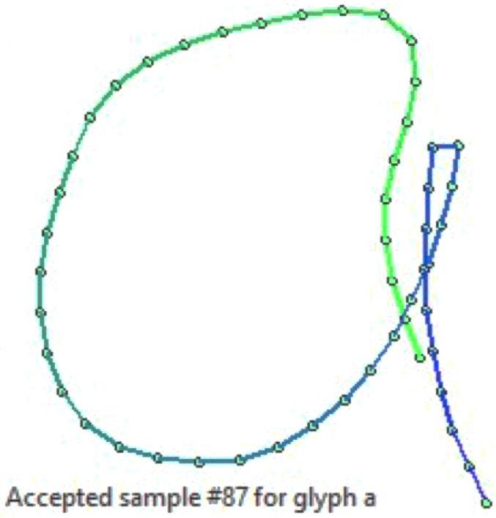


Рисунок 3.4 -Вікно візуалізації жестів

Збережені жести можна переглянути натисненням ЛКМ, або видалити натисненням ПКМ по мініатюрі з відображенням жесту у загальному переліку.

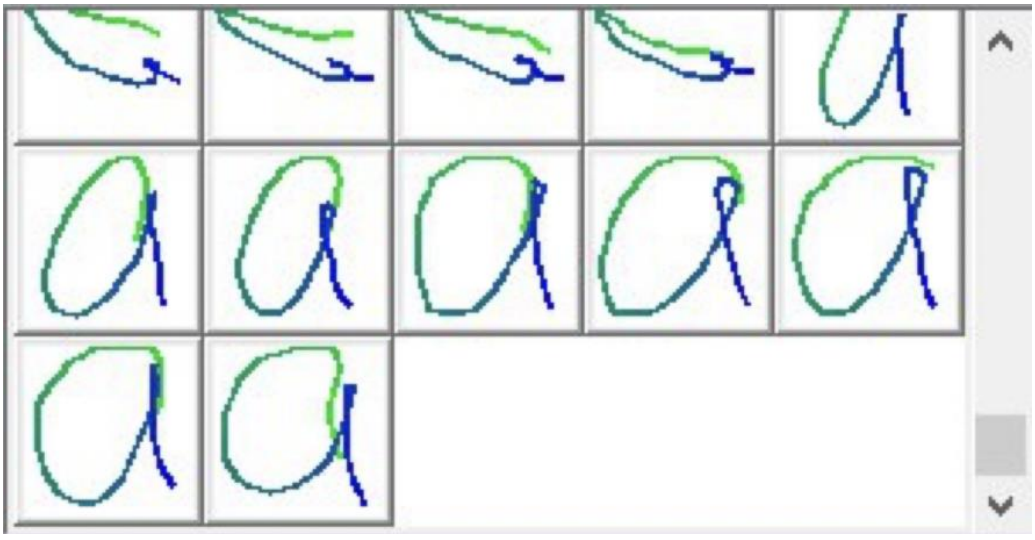

Рисунок 3.5 – Загальний перелік зареєстрованих жестів

Зареєстровані жести зберігаються у локальній базі даних. Застосунок обладнаний можливістю експорту та імпорту баз даних для жестів, таким чином дозволяючи одночасну роботу над проєктом декількох розробників.

Фінальний вигляд застосунку зображений на рисунку 3.6. Розроблений застосунок надає зручний графічний інтерфейс для збереження набору даних для тренування нейронної мережі.

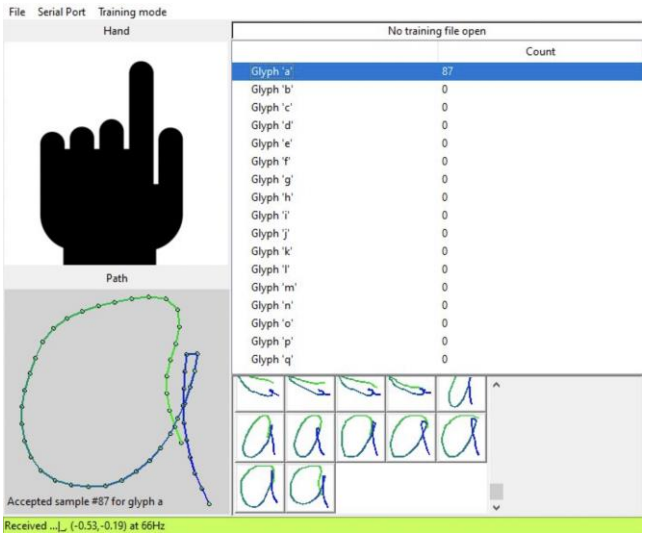


Рисунок 3.6 – Вікно застосунку для збереження набору жестів

Використовуючи даний застосунок, було підготовано набір із понад 6000 жестів, на основі яких було проведено тренування нейронної мережі.

### 3.3 Розробка нейронної мережі для розпізнавання жестів

Аналіз можливостей використання бібліотеки TensorFlow Lite Micro для мікроконтролерів у Розділі 1 показав, що на сьогоднішній день дана бібліотека має всі необхідні інструменти для впровадження алгоритмів машинного навчання у мікроконтролерах. Доступна бібліотека для мови програмування Python дала змогу використати структури даних для опису жестів, що використовувались в застосунку для реєстрації жестів, безпосередньо у процесі тренування нейронної мережі на основі згенерованого датасету.

Незважаючи на те, що бібліотека TensorFlow має низький рівень абстракції, тісна інтеграція з екосистемою Keras зробила вагомий внесок в

популяризацію машинного навчання, запропонувавши вищий рівень абстракції для взаємодії з нейронними мережами. Станом на кінець 2021 року, Keras мав більше одного мільйона унікальних користувачів, та був найпопулярнішим API для машинного навчання [25].

Для задачі розпізнавання жестів було використано послідовну модель нейронної мережі – модель Sequential у Keras. Послідовна модель підходить для простого стеку шарів, де кожен шар має рівно один вхідний тензор і один вихідний тензор. Ці моделі краще підходять для аналіу послідовних даних, таких як текстові речення, часові ряди та інші дані дискретної послідовності, тоді як згорткові нейронні мережі краще підходять для обробки просторових даних.

Важливим елементом, який слід пам’ятати про моделі послідовності, є те, що дані більше не є незалежними та однаково розподіленими вибірками, а залежать один від одного завдяки їхньому послідовному порядку. Для розпізнавання мовлення, розпізнавання голосу, передбачення часових рядів та обробки природної мови послідовні моделі користуються особливою популярністю [26].

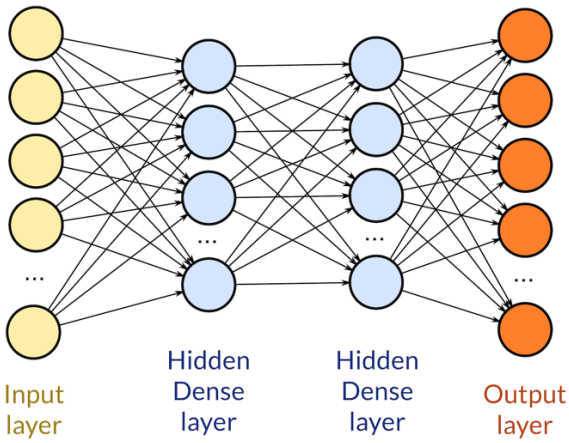


Рисунок 3.7 – Структура послідовної моделі нейронної мережі

Моделювання послідовності — це процес створення послідовності значень із набору вхідних значень. Ці вхідні значення можуть бути даними часових рядів, які показують, як певна величина змінюється з часом.

Модель, створена для розпізнавання жестів у цьому випадку, використовує послідовність повнозв’язних шарів (див. рис. 3.8), глибоко пов'язаних з попереднім шаром, що означає, що нейрони шару пов'язані з кожним нейроном його попереднього шару.

Нейрон щільного шару в моделі отримує вихід від кожного нейрона свого попереднього шару, де нейрони щільного шару здійснюють множення матриці на вектор.

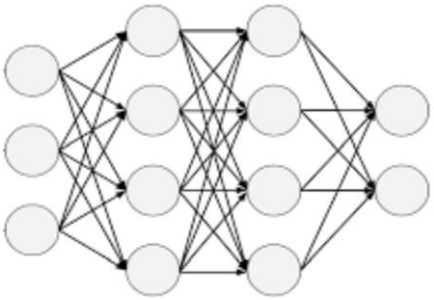

Рисунок 3.8 – Повнозв’язні шари нейронної мережі

Множення матриці на вектор – це процедура, де вектор-рядок вихідних значень з попередніх шарів дорівнює вектору-стовпцю щільного шару. Загальне правило множення матриці на вектор полягає в тому, що вектор-рядок повинен мати стільки стовпців, як і вектор-стовпець:

$$Ax = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}, \quad (3.2)$$

де A – це матриця розміру M × N;

x — (1 × N) матриця.

Значення під матрицею є навченими параметрами попередніх шарів і також можуть бути оновлені шляхом зворотного поширення. Вихід із щільного шару буде N-вимірним вектором. Таким чином, в основному щільний шар використовується для зміни розміру векторів за допомогою кожного нейрона.

У створеній нейронній мережі використано чотири повнозв’язних шари із функціями активації relu, та п’ятий шар – із softmax функцією активації для вихідних значень (див. рис. 3.9). Діаграма шарів нейронної мережі наведена в Додатку В.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 100)               0
_________________________________________________________________
dense (Dense)                (None, 32)                3232
_________________________________________________________________
dense_1 (Dense)              (None, 100)               3300
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_4 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_5 (Dense)              (None, 26)                2626
=================================================================
Total params: 39,458
Trainable params: 39,458
Non-trainable params: 0
```

Рисунок 3.9 – Параметри моделі

Дана модель має зареєстровані жести на літери латинського алфавіту. Її використання дозволяє друкувати текст за допомогою розпізнавання жестів.

Загалом тренування нейронної мережі проводилось впродовж 500 епох. Обчислена точність такої мережі наближається до 100% (див. рис. 3.10).

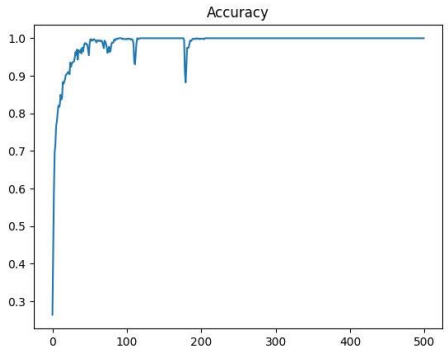


Рисунок 3.10 – Точність нейронної мережі

Така точність для нейронної мережі є достатньою, проте з рисунка 3.10 можемо побачити, що для тренування достатньо всього 200 епох.

## 3.4 Використання моделі машинного навчання у програмі мікроконтролера

Після тренування нейронної мережі отримано готовий для інтеграції файл model.h, імпортування якого та запуск за допомогою стандартного інтерпретатора бібліотеки Tensorflow для фреймворку Arduino.

Враховуючи знаходження двох програм у одній кодовій базі, розробка частини програми для взаємодії з комп’ютером як HID не потребує написання додаткового коду для опитування сенсорів. Функції взаємодії з датчиками були створені на етапі створення ПЗ для тренування моделі.

Отримавши вхідні дані, нейронна мережа визначає найбільш вірогідний символ для друку, або його відсутність. Якщо жест знайдено – він повинен бути надрукований. Для впровадження функціоналу клавіатури було використано стандартну бібліотеку Keyboard для Arduino. Ця бібліотека дозволяє використання стандартних для Arduino функцій write, print, println та інших [27].

Для створення незалежного від периферійних пристроїв девайсу було додатково передбачено функціонал комп’ютерної миші, яка працює на основі отриманих з акселерометра даних. Як і клавіатура, емуляція миші створена на основі стандартної бібліотеки Arduino Mouse [28].

Вибір між режимами введення організовано на основі комбінацій пальців. Якщо користувач згинає усі пальці, окрім вказівного, розпочинається процес розпізнавання введеного жесту, якщо користувач згинає усі пальці, окрім вказівного та середнього, пристрій переходить у режим користування мишею.

## 3.5 Висновки до розділу 3

Концепт прототипу пристрою, розробка якого була однією із цілей дипломної роботи, довів свою працездатність. Алгоритми машинного навчання, розроблені для розпізнавання жестів, виявились достатньо точними для реалізації поставлених задач. Кількість епох тренування

розробленої нейронної мережі можна скоротити вдвічі без втрати точності. Обчислювальні потужності Teensy 4.0, як і було передбачено під час аналізу, дозволяють використовувати алгоритми машинного навчання. Прототип пристрою відповідає вимогам щодо функціональності та ергономічності, описаним в Розділі 1, та може бути використаний у майбутніх наукових дослідженнях.

# ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра було розроблено програмно-апаратний комплекс для розпізнавання жестів із використанням алгоритмів машинного навчання. Обчислювальна потужність апаратної бази HID-пристрою (плати розробки Teensy 4.0) виявилася достатньою для виконання поставлених задач. IMU-сенсор MPU9250, використаний у проєкті, завдяки вбудованому модулю DMP здатен точно вимірювати власне положення.

Програмне забезпечення для форматування даних з датчиків та виведення їх для подальшого аналізу здатне виконувати поставлену задачу з частотою 2 кГц, проте для покращення ефективності частоту забору даних було зменшено до 60 Гц. Розпізнавання жестів працює із точністю 98.86%.

Основними перевагами розробленого пристрою є його ергономічність та багатофункціональність. Не заважаючи вільним рухам руки, пристрій може одночасно виконувати дії, що здатні замінити клавіатуру та комп’ютерну мишу стандартного форм-фактору.

Головним недоліком пристрою є обмежений набір символів для введення, оновлення якого можливе лише за умови повторного програмування пристрою. Подальший розвиток проєкту буде сфокусований на розширенні доступного словника та можливості додавання нових символів без необхідності повторного програмування пристрою. Покращення пристрою можливе за рахунок встановлення більш потужної мікросхеми цифрового процесору рухів та аналогових датчиків ефекту Холла.

Результати роботи можуть зацікавити потенційних інвесторів та розпочати активний етап розробки пристрою для виходу на споживчий ринок.

В процесі виконання роботи було виконано спеціальну частину з охорони праці, в якій було проаналізовано нормативно-правові акти щодо

охорони праці та безпеки життєдіяльності та визначено вимоги для безпечного виконання паяльних робіт.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Seneviratne S. et al. A Survey of Wearable Devices and Challenges // IEEE Communications Surveys and Tutorials. Institute of Electrical and Electronics Engineers Inc., 2017. Вид. 19, № 4. С. 2573–2620.
2. Овчар С.В., Старченко В.В. Аналіз апаратного забезпечення для реалізації пристроїв з функцією розпізнавання рукопису. // Збірник тез XXIV Всеукраїнської науково-методичної конференції «Могилянські читання – 2021». Миколаїв: Вид-во ЧНУ ім. Петра Могили, 2021. С. 69–71.
3. Petro Mohyla TV: Startup BSNU 2020 - Чорноморський національний університет імені Петра Могили - Чорноморський національний університет імені Петра Могили [Електронний ресурс]. URL: https://chmnu.edu.ua/petro-mohyla-tv-startup-bsnu-2020/ (дата звернення: 24.06.2022).
4. Case Studies and Mentions | TensorFlow [Електронний ресурс]. URL: https://www.tensorflow.org/about/case-studies (дата звернення: 24.06.2022).
5. TensorFlow Lite for Microcontrollers [Електронний ресурс]. URL: https://www.tensorflow.org/lite/microcontrollers (дата звернення: 24.06.2022).
6. PJRC. Teensy® 4.0 [Електронний ресурс]. URL: https://www.pjrc.com/store/teensy40.html (дата звернення: 24.06.2022).
7. NXP Semiconductors. i.MX RT1160 Crossover Processors Data Sheet for Consumer Products. 2021.
8. US9069386B2 - Gesture recognition device, method, program, and computer-readable medium upon which program is stored - Google Patents [Електронний ресурс]. URL: https://patents.google.com/patent/US9069386B2/en?q=gesture+recognition+device&oq=gesture+recognition+device (дата звернення: 24.06.2022).

9. US9921659B2 - Gesture recognition for device input - Google Patents [Електронний ресурс]. URL: https://patents.google.com/patent/US9921659B2/en?q=gesture+recognition+device+gyroscope&oq=gesture+recognition+device+gyroscope (дата звернення: 24.06.2022).
10. KR101079270B1 - Three dimensional input device - Google Patents [Електронний ресурс]. URL: https://patents.google.com/patent/KR101079270B1/en?q=gesture+recognition+device+gyroscope&oq=gesture+recognition+device+gyroscope&page=1 (дата звернення: 24.06.2022).
11. How Tap Works - Tap [Електронний ресурс]. URL: https://www.tapwithus.com/how-tap-works/ (дата звернення: 24.06.2022).
12. EasyEDA Pro Tutorial [Електронний ресурс]. URL: https://prodocs.easyeda.com/en/faq/editor/index.html (дата звернення: 24.06.2022).
13. Shipping & Delivery - JLCPCB: Help & Support [Електронний ресурс]. URL: https://support.jlcpcb.com/category/20-shipping-delivery (дата звернення: 24.06.2022).
14. Єжов С.М., Макарець М.В., Романенко О.В. Класична Механіка. К.: ВПЦ “Київський університет,” 2008. 480 с.
15. Арнольд В.И. Геометрия комплексных чисел, кватернионов и спинов. М.: МЦНМО, 2002. 40 с.
16. Kalman R.E. A New Approach to Linear Filtering and Prediction Problems. 1960.
17. Chou Y. Statistical analysis, with business and economic applications. Holt, Rinehart and Winston, 1975. 894 с.
18. Hall E.H. On a New Action of the Magnet on Electric Currents // American Journal of Mathematics. JSTOR, 1879. Вид. 2, № 3. С. 287.

19. Petruk O. et al. Sensitivity and offset voltage testing in the hall-effect sensors made of graphene // Advances in Intelligent Systems and Computing. Springer Verlag, 2014. Вид. 267. С. 631–640.
20. Ramsden Ed. Hall-effect sensors : theory and applications. Elsevier/Newnes, 2006. С. 250.
21. InvenSense Inc. MPU-9250 Product Specification [Електронний ресурс].
22. Cloud-based Electronic-Design Tools Gain Traction - IEEE Spectrum [Електронний ресурс]. URL: https://spectrum.ieee.org/cloudbased-electronicdesign-tools-gain-traction (дата звернення: 24.06.2022).
23. SparkFun. SparkFun_MPU-9250-DMP_Arduino_Library: Arduino library for the MPU-9250 enabling its digital motion process (DMP) features. [Електронний ресурс]. URL: https://github.com/sparkfun/SparkFun_MPU-9250-DMP_Arduino_Library (дата звернення: 24.06.2022).
24. Yin D.P. et al. Python for Data Analysis Data Wrangling with Pandas, NumPy, and IPython // Transplantation. 2001. Вид. 71, № 10. С. 1385–1389.
25. Why choose Keras? [Електронний ресурс]. URL: https://keras.io/why_keras/#keras-has-broad-adoption-in-the-industry-and-the-research-community (дата звернення: 24.06.2022).
26. A Tutorial on Sequential Machine Learning [Електронний ресурс]. URL: https://analyticsindiamag.com/a-tutorial-on-sequential-machine-learning/ (дата звернення: 24.06.2022).
27. Keyboard - Arduino Reference [Електронний ресурс]. URL: https://www.arduino.cc/reference/en/language/functions/usb/keyboard/ (дата звернення: 24.06.2022).
28. Mouse - Arduino Reference [Електронний ресурс]. URL: https://www.arduino.cc/reference/en/language/functions/usb/mouse/ (дата звернення: 24.06.2022).

# ДОДАТОК А
# ПОСИЛАННЯ НА ОНЛАЙН-РЕПОЗИТОРІЙ БІБЛІОТЕКИ TENSORFLOW LITE MICRO

https://github.com/tensorflow/tflite-micro

# ДОДАТОК Б

# КОД ПРОГРАМИ ДЛЯ МІКРОКОНТРОЛЕРА

```
#include <math.h>
#include <Wire.h>
#include <TensorFlowLite.h>
#include "tensorflow/lite/micro/all_ops_resolver.h"
#include "model.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"
#include "SparkFunMPU9250-DMP.h"
#define DEBUG
#define GET_MACRO(_1, _2, NAME, ...) NAME
byte me;
#ifdef DEBUG
#define formatted_debug(x, y) Serial.print(x, y)
#define noformat_debug(x) Serial.print(x)
#define formatted_debug_endline(x, y) Serial.println(x, y)
#define noformat_debug_endline(x) Serial.println(x)
#define debug_raw(x) Serial.write(x)
#else
#define formatted_debug(x, y)
#define noformat_debug(x)
#define formatted_debug_endline(x, y)
#define noformat_debug_endline(x)
#define debug_raw(x)
#endif
#define printDebug(...) GET_MACRO(__VA_ARGS__, formatted_debug, noformat_debug)(__VA_ARGS__)
#define printlnDebug(...) GET_MACRO(__VA_ARGS__, formatted_debug_endline, noformat_debug_endline)(__VA_ARGS__)
#define sanity(x) printlnDebug("Sanity " #x)
MPU9250_DMP mpu9250;
#define TRAINING_MODE
// #define HID_MODE
// #define PRINT_ANGULAR_VELOCITY
//#define PRINT_GESTURE_INFO
//#define PRINT_TF_INFO
namespace
{
  tflite::ErrorReporter *error_reporter = nullptr;
  const tflite::Model *model = nullptr;
  tflite::MicroInterpreter *interpreter = nullptr;
  TfLiteTensor *input = nullptr;
```

```
  TfLiteTensor *output = nullptr;
  int inference_count = 0;
  constexpr int kTensorArenaSize = 4 * 1024;
  uint8_t tensor_arena[kTensorArenaSize];
}
unsigned long debounce_finger = 100;
const byte finger_switch_pins[] = {20, 21, 22, 23};
#define NO_HAND_SIGN 0
#define MOUSE_HAND_SIGN 1
#define KEYBOARD_HAND_SIGN 10
bool last_finger_poitions[4];
byte hand_sign = NO_HAND_SIGN;
unsigned long last_stable_timestamp_finger[4] = {0};
float last_bearing[3];
float yaw;
float pitch;
const float start_velocity_threshold = 5. / 1000000.; // rad/microsecond
const float end_velocity_threshold = 1.5 / 1000000.;  // rad/microsecond
const float mouse_velocity_threshold = 0.01 / 1000000.; // rad/microsecond
const int history_length = 5;
float angular_velocity_window[history_length];
#define MOUSE_SCALE 500.
#define GESTURE_DEBOUNCE_LOCKOUT 200
//#define GESTURE_DEBOUNCE_LOCKOUT 0
bool is_drawing_glyph;
elapsedMillis time_since_last_gesture;
unsigned long last_timestamp;
elapsedMillis time_since_last_debug_command;
#define COMMAND_DEBOUNCE 10000
#define GESTURE_CONE_ANGLE 2.0 / 3.0 * PI
float gesture_bearing_zero[2];
#define MAX_GESTURE_LENGTH 100
float gesture_buffer[100][2]; // 0.0-1.0, 0.5, 0.5 is the starting position
float gesture_buffer_yaw_min;
float gesture_buffer_yaw_max;
float gesture_buffer_pitch_min;
float gesture_buffer_pitch_max;
float processed_gesture[50][2];
unsigned int gesture_buffer_length;
bool is_frozen = false;
elapsedMillis time_since_freeze;
#define FREEZE_TIME 2000
void setup()
{
  Serial.begin(115200);
```

```
  delay(5000);
#ifdef TRAINING_MODE
  printlnDebug("Training mode");
#endif
#ifdef HID_MODE
  printlnDebug("HID mode");
#endif
  Wire.begin();
  if (mpu9250.begin() != INV_SUCCESS)
  {
    while (1)
    {
      Serial.println("Unable to begin MPU9250 comunication");
      Serial.println("Check connection and try again.");
      Serial.println();
      delay(5000);
    }
  }
  mpu9250.setSampleRate(60);
  mpu9250.dmpBegin(DMP_FEATURE_6X_LP_QUAT | DMP_FEATURE_GYRO_CAL, 60);
  for (int i = 0; i < 4; i++)
  {
    pinMode(finger_switch_pins[i], INPUT);
  }
  static tflite::MicroErrorReporter micro_error_reporter;
  error_reporter = &micro_error_reporter;
  model = tflite::GetModel(modelBin);
  if (model->version() != TFLITE_SCHEMA_VERSION)
  {
    error_reporter->Report(
        "Model provided is schema version %d not equal "
        "to supported version %d.",
        model->version(), TFLITE_SCHEMA_VERSION);
    return;
  }
  static tflite::AllOpsResolver resolver;
  static tflite::MicroInterpreter static_interpreter(
      model, resolver, tensor_arena, kTensorArenaSize, error_reporter);
  interpreter = &static_interpreter;
  TfLiteStatus allocate_status = interpreter->AllocateTensors();
  if (allocate_status != kTfLiteOk)
  {
    error_reporter->Report("AllocateTensors() failed");
    return;
  }
```

```
  input = interpreter->input(0);
  output = interpreter->output(0);
  printDebug("Input tensor size: ");
  printDebug(input->dims->size);
  printDebug(" Shape: (");
  for (int i = 0; i < 3; i++)
  {
    printDebug(input->dims->data[i]);
    if (i < 2)
      printDebug(", ");
  }
  printlnDebug(')');
  printDebug("Output tensor size: ");
  printDebug(output->dims->size);
  printDebug(" Shape: (");
  for (int i = 0; i < 3; i++)
  {
    printDebug(input->dims->data[i]);
    if (i < 2)
      printDebug(", ");
  }
  printlnDebug(')');
  inference_count = 0;
  printDebug("Gesture cone angle: ");
  printlnDebug(GESTURE_CONE_ANGLE, 5);
  last_timestamp = micros();
}
void loop()
{
  unsigned long timestamp = micros();
  float sampleRate = timestamp - last_timestamp;
  while (Serial.available())
  {
    int incoming = Serial.read();
    Serial.write(incoming);
    time_since_last_debug_command = 0;
  }
  if (mpu9250.fifoAvailable())
  {
    if (mpu9250.update() == INV_SUCCESS && mpu9250.dmpUpdateFifo() == INV_SUCCESS)
    {
      yaw = mpu9250.calcQuat(mpu9250.qw) * -1;
      pitch = mpu9250.calcQuat(mpu9250.qx);
      float theta = 0;
      theta = asin(norm(last_bearing[0], last_bearing[1], yaw, pitch));
```

```
    float angular_velocity = theta / sampleRate;
    float angular_velocity_avg = angular_velocity;
    for (int i = history_length - 2; i >= 0; i--)
    {
      angular_velocity_avg += angular_velocity_window[i];
      angular_velocity_window[i + 1] = angular_velocity_window[i];
    }
    angular_velocity_window[0] = angular_velocity;
    angular_velocity_avg /= history_length;
    for (int i = 0; i < 4; i++)
    {
      bool finger_position = digitalRead(finger_switch_pins[i]);
      if (finger_position == last_finger_poitions[i])
      {
        last_stable_timestamp_finger[i] = millis();
      }
      else if (millis() - last_stable_timestamp_finger[i] >= debounce_finger)
      {
        last_finger_poitions[i] = finger_position;
        last_stable_timestamp_finger[i] = millis();
      }
    }
    bool too_fast_to_draw = false;
    if (time_since_last_gesture < GESTURE_DEBOUNCE_LOCKOUT)
      too_fast_to_draw = false;
    else if ((is_drawing_glyph && angular_velocity_avg >= end_velocity_threshold) || (!is_drawing_glyph && angular_velocity >=
start_velocity_threshold))
    {
      too_fast_to_draw = true;
    }
    if (last_finger_poitions[0] && last_finger_poitions[1] && last_finger_poitions[2] && !last_finger_poitions[3])
    {
      printlnDebug("Keyboard mode");
      hand_sign = KEYBOARD_HAND_SIGN;
    }
    else if (last_finger_poitions[0] && last_finger_poitions[1] && !last_finger_poitions[2] && !last_finger_poitions[3])
    {
      printlnDebug("Mouse mode");
      hand_sign = MOUSE_HAND_SIGN;
    }
    else
    {
      printlnDebug("No mode");
      hand_sign = NO_HAND_SIGN;
    }
```

```
if (too_fast_to_draw)
{
  if (hand_sign == KEYBOARD_HAND_SIGN)
  {
    if (!is_drawing_glyph)
    {
      gesture_buffer_length = 0;
      printDebug("Started gesturing after ");
      printDebug(time_since_last_gesture);
      printlnDebug(" ms");
    }
    is_drawing_glyph = true;
  }
}
else
{
  if (is_drawing_glyph)
    printlnDebug("Stopped gesturing");
  is_drawing_glyph = false;
}
char winner_glyph = 0x00;
float top_score = 0.;
if (is_drawing_glyph)
{
  if (gesture_buffer_length < MAX_GESTURE_LENGTH)
  {
    if (gesture_buffer_length == 0)
    {
      gesture_bearing_zero[0] = yaw;
      gesture_bearing_zero[1] = pitch;
      gesture_buffer[0][0] = gesture_buffer_yaw_min = gesture_buffer_yaw_max = 0.5;
      gesture_buffer[0][1] = gesture_buffer_pitch_min = gesture_buffer_pitch_max = 0.5;
    }
    else
    {
      float yaw_processed = wrapped_delta(gesture_bearing_zero[0], yaw);
      float pitch_processed = wrapped_delta(gesture_bearing_zero[1], pitch);
      yaw_processed = constrain(yaw_processed, -0.5 * GESTURE_CONE_ANGLE, 0.5 * GESTURE_CONE_ANGLE);
      pitch_processed = constrain(pitch_processed, -0.5 * GESTURE_CONE_ANGLE, 0.5 * GESTURE_CONE_ANGLE);
      yaw_processed /= GESTURE_CONE_ANGLE;
      pitch_processed /= GESTURE_CONE_ANGLE;
      yaw_processed += 0.5;
      pitch_processed += 0.5;
      gesture_buffer_yaw_min = min(gesture_buffer_yaw_min, yaw_processed);
      gesture_buffer_yaw_max = max(gesture_buffer_yaw_max, yaw_processed);
```

```
      gesture_buffer_pitch_min = min(gesture_buffer_pitch_min, pitch_processed);
      gesture_buffer_pitch_max = max(gesture_buffer_pitch_max, pitch_processed);
      gesture_buffer[gesture_buffer_length][0] = yaw_processed;
      gesture_buffer[gesture_buffer_length][1] = pitch_processed;
    }
    gesture_buffer_length++;
  }
}
else if (gesture_buffer_length > 0)
{
  time_since_last_gesture = 0;
  process_bearings(gesture_buffer, gesture_buffer_length,
                   gesture_buffer_yaw_min, gesture_buffer_yaw_max, gesture_buffer_pitch_min, gesture_buffer_pitch_max,
                   processed_gesture, 50);
  for (int i = 0; i < 100; i++)
  {
    input->data.f[i] = processed_gesture[i / 2][i % 2];
  }
  TfLiteStatus invoke_status = interpreter->Invoke();
  if (invoke_status != kTfLiteOk)
  {
    error_reporter->Report("Failed to invoke\n");
  }
  else
  {
    for (int i = 0; i <= charCount; i++)
    {
      float score = output->data.f[i];
      if (score > top_score)
      {
        winner_glyph = charMap[i];
        top_score = score;
      }
    }
    for (int i = 0; i < 50; i++)
    {
      if (winner_glyph >= ' ')
        printDebug(winner_glyph);
      else
      {
        printDebug("0x");
        printDebug(winner_glyph, HEX);
        printDebug(' ');
      }
    }
```

```
        printlnDebug();
      }
      gesture_buffer_length = 0;
    } // gesture_buffer_length > 0
    if (time_since_last_debug_command >= COMMAND_DEBOUNCE)
    {
#ifdef TRAINING_MODE
      char packet_outgoing[100] = {0};
      packet_outgoing[0] = '>';
      if (last_finger_poitions[0])
        packet_outgoing[1] = '.';
      else
        packet_outgoing[1] = '|';
      packet_outgoing[2] = ',';
      if (last_finger_poitions[1])
        packet_outgoing[3] = '.';
      else
        packet_outgoing[3] = '|';
      packet_outgoing[4] = ',';
      if (last_finger_poitions[2])
        packet_outgoing[5] = '.';
      else
        packet_outgoing[5] = '|';
      packet_outgoing[6] = ',';
      if (last_finger_poitions[3])
        packet_outgoing[7] = '.';
      else
        packet_outgoing[7] = '|';
      packet_outgoing[8] = ',';
      dtostrf(yaw, 6, 4, &packet_outgoing[strlen(packet_outgoing)]);
      packet_outgoing[strlen(packet_outgoing)] = ',';
      dtostrf(pitch, 6, 4, &packet_outgoing[strlen(packet_outgoing)]);
      packet_outgoing[strlen(packet_outgoing)] = ',';
      dtostrf(mpu9250.calcQuat(mpu9250.qy), 6, 4, &packet_outgoing[strlen(packet_outgoing)]);
      packet_outgoing[strlen(packet_outgoing)] = ',';
      dtostrf(mpu9250.calcAccel(mpu9250.ax), 7, 4, &packet_outgoing[strlen(packet_outgoing)]);
      packet_outgoing[strlen(packet_outgoing)] = ',';
      dtostrf(mpu9250.calcAccel(mpu9250.ay), 7, 4, &packet_outgoing[strlen(packet_outgoing)]);
      packet_outgoing[strlen(packet_outgoing)] = ',';
      dtostrf(mpu9250.calcAccel(mpu9250.az), 7, 4, &packet_outgoing[strlen(packet_outgoing)]);
      packet_outgoing[strlen(packet_outgoing)] = ',';
      itoa(sampleRate, &packet_outgoing[strlen(packet_outgoing)], 10);
      packet_outgoing[strlen(packet_outgoing)] = '\n';
      Serial.println(packet_outgoing);
#endif // TRAINING_MODE
```

```
#ifdef HID_MODE
      byte packetLength = 0;
      if (winner_glyph == 0x00)
      {
       if (hand_sign == MOUSE_HAND_SIGN && angular_velocity_avg >= mouse_velocity_threshold)
       {
         int xStop = wrapped_delta(last_bearing[0], yaw) * MOUSE_SCALE;
         int yStop = wrapped_delta(last_bearing[1], pitch) * MOUSE_SCALE;
         if (xStop != 0 || yStop != 0)
         {
           printDebug("Moving mouse ");
           printDebug(xStop);
           printDebug(" units right and ");
           printDebug(yStop);
           printlnDebug(" units up");
           byte buf[] = {0xfd, 0x05, 0x02, 0x00, char(xStop), char(yStop), 0x00};
           for (int i = 0; i < sizeof(buf) / sizeof(buf[0]); ++i)
           {
             Serial.print(buf[i]);
           }
           Serial.println();
         }
       }
      }
      else
      {
        Serial.print(winner_glyph);
        printDebug("Typing ");
        printlnDebug(char(winner_glyph));
      }
#endif // ifdef HID_MODE
      if (is_frozen)
      {
        printlnDebug("Back to normal");
        is_frozen = false;
      }
    }
    last_timestamp = timestamp;
    last_bearing[0] = yaw;
    last_bearing[1] = pitch;
  }
 }
}
float wrapped_delta(float old_value, float new_value)
{
```

```
  float delta = old_value - new_value;
  if (delta > PI)
    return delta - (2. * PI);
  else if (delta < -PI)
    return delta + (2. * PI);
  else
    return delta;
}
float norm(float oldX, float oldY, float newX, float newY)
{
  float xDelta = wrapped_delta(oldX, newX);
  float yDelta = wrapped_delta(oldY, newY);
  return sqrt(xDelta * xDelta + yDelta * yDelta);
}
void process_bearings(float input[][2], unsigned int input_length, float input_yaw_min, float input_yaw_max, float input_pitch_min,
float input_pitch_max, float output[][2], unsigned int output_length)
{
  unsigned long benchmark = micros();
  float buffer_working_1[MAX_GESTURE_LENGTH][2] = {0};
  unsigned int pos_buffer_1 = 0;
  buffer_working_1[0][0] = input[0][0];
  buffer_working_1[0][1] = input[0][1];
  pos_buffer_1 = 1;
  for (int i = 1; i < input_length; i++)
  {
    float yaw_delta = fabs(input[i][0] - input[i - 1][0]);
    float pitch_delta = fabs(input[i][1] - input[i - 1][1]);
    if (yaw_delta > 0.000001 || pitch_delta > 0.000001)
    {
      buffer_working_1[pos_buffer_1][0] = input[i][0];
      buffer_working_1[pos_buffer_1][1] = input[i][1];
      pos_buffer_1++;
    }
  }
  float stop_trim_length = norm(input_yaw_min, input_pitch_min, input_yaw_max, input_pitch_max) / 10.;
  int leading_point_stripping_end = 0;
  int trailing_point_stripping_start = pos_buffer_1 - 1;
  for (int i = 1; i < pos_buffer_1; i++)
  {
    float distance = norm(buffer_working_1[i][0], buffer_working_1[i][1], buffer_working_1[0][0], buffer_working_1[0][1]);
    if (distance >= stop_trim_length)
    {
      break;
    }
    else
```

```
  {
    leading_point_stripping_end++;
  }
}
for (int i = pos_buffer_1 - 2; i > 0; i--)
{
  float distance = norm(buffer_working_1[pos_buffer_1 - 1][0], buffer_working_1[pos_buffer_1 - 1][1],
                        buffer_working_1[i][0], buffer_working_1[i][1]);
  if (distance > stop_trim_length)
  {
    break;
  }
  else
  {
    trailing_point_stripping_start--;
  }
}
float buffer_working_2[MAX_GESTURE_LENGTH][2] = {0};
unsigned int pos_buffer_2 = 0;
float segment_cumulative_lengths[MAX_GESTURE_LENGTH] = {0};
float curve_length = 0;
for (int i = 0; i < pos_buffer_1; i++)
{
  if (i == 0 || i == pos_buffer_1 - 1 || (i > leading_point_stripping_end && i < trailing_point_stripping_start))
  {
    buffer_working_2[pos_buffer_2][0] = buffer_working_1[i][0];
    buffer_working_2[pos_buffer_2][1] = buffer_working_1[i][1];
    if (pos_buffer_2 == 0)
      segment_cumulative_lengths[0] = 0;
    else
    {
      float segment_length = norm(buffer_working_2[pos_buffer_2 - 1][0], buffer_working_2[pos_buffer_2 - 1][1],
                                  buffer_working_2[pos_buffer_2][0], buffer_working_2[pos_buffer_2][1]);
      segment_cumulative_lengths[pos_buffer_2] = segment_cumulative_lengths[pos_buffer_2 - 1] + segment_length;
      curve_length += segment_length;
    }
    pos_buffer_2++;
  }
}
float target_segment_length = curve_length / (output_length - 1);
int first_longer_sample = 0;
float high_point[2];
float low_point[2];
float long_dimensions_length = max(input_yaw_max - input_yaw_min, input_pitch_max - input_pitch_min);
output[0][0] = map(buffer_working_2[0][0] - input_yaw_min, 0., long_dimensions_length, 0., 1.);
```

```
  output[0][1] = map(buffer_working_2[0][1] - input_pitch_min, 0., long_dimensions_length, 0., 1.);
  for (int i = 1; i < output_length; i++)
  {
    float targetLength = i * target_segment_length;
    if (segment_cumulative_lengths[first_longer_sample] > targetLength)
    {}
    else
    {
      while (segment_cumulative_lengths[first_longer_sample] < targetLength &&
fabs(segment_cumulative_lengths[first_longer_sample] - targetLength) > 0.00001)
      {
        first_longer_sample++;
        if (first_longer_sample >= pos_buffer_2) {
          return;
        }
        low_point[0] = buffer_working_2[first_longer_sample - 1][0];
        low_point[1] = buffer_working_2[first_longer_sample - 1][1];
        high_point[0] = buffer_working_2[first_longer_sample][0];
        high_point[1] = buffer_working_2[first_longer_sample][1];
      }
    }
    float position_along_segment =
        (targetLength - segment_cumulative_lengths[first_longer_sample - 1]) /
        (segment_cumulative_lengths[first_longer_sample] - segment_cumulative_lengths[first_longer_sample - 1]);
    float standardized_yaw = low_point[0] + position_along_segment * (high_point[0] - low_point[0]);
    float standardized_pitch = low_point[1] + position_along_segment * (high_point[1] - low_point[1]);
    standardized_yaw -= input_yaw_min;
    standardized_pitch -= input_pitch_min;
    standardized_yaw = map(standardized_yaw, 0., long_dimensions_length, 0., 1.);
    standardized_pitch = map(standardized_pitch, 0., long_dimensions_length, 0., 1.);
    output[i][0] = standardized_yaw;
    output[i][1] = standardized_pitch;
  }
}
```

# ДОДАТОК В
# ДІАГРАМА НЕЙРОННОЇ МЕРЕЖІ