**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»

УДК 004.056.5

«До захисту допущено»

Завідувач кафедри

_____ *Сергій СТІРЕНКО*
(підпис) (ім'я, прізвище)

"____" _____ 2022 р.

# Магістерська дисертація

зі спеціальності: *123. Комп'ютерна інженерія*
(код та назва напряму підготовки або спеціальності)

Спеціалізація: *123. Комп'ютерні системи та мережі*

на тему: Система обробки данних для Smart City на базі нейронної мережі

Виконав: студент _II_ курсу, групи ІО-01мн
(шифр групи)

Копійка Антон Олександрович
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доцент, к. т. н. Ткаченко В. В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2022 року

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність *123. Комп'ютерна інженерія*
(код і назва)

Спеціалізація *123. Комп'ютерні системи та мережі*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
*Сергій СТІРЕНКО*
(підпис) (ініціали, прізвище)
« » _____ 2022 р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**
Копійці Антону Олександровичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Система обробки данних для Smart City на базі нейронної мережі
Науковий керівник дисертації доцент, к. т. н. Ткаченко В. В.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 26» квітня 2022 р. № НС/88/2022

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження процес збору та обробки інформації про якість дорожнього покриття, який ускладнений необхідністю збирання, оброблення та класифікації великої кількості параметрів в режимі часу наближеному до реального.

4. Предмет дослідження способи та підходи до збору показників лінійного прискорення та їх подальшого використання для класифікації якості дорожнього покриття за допомогою засобів машинного навчання та використання згорткових нейронних мереж для роботи з часовими рядами.

5. Перелік завдань, які потрібно розробити: аналіз предметної області, аналіз програмних та апаратних засобів, проектування архітектури проекту, розробка

апаратної частини, розробка моделі нейронної мережі, розробка прототипу системи.

6. Консультанти розділів дисертації:

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормоконтроль | Кулаков Ю.О., професор | | |

7. Дата видачі завдання

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Строк виконання етапів дисертації | Примітка |
|---|---|---|---|
| 1. | Затвердження теми роботи | 15.01.2022 | |
| 2. | Вивчення та аналіз завдання | 15.01.2022-15.03.2022 | |
| 3. | Розробка архітектури та загальної структури систем | 15.03.2022-25.03.2022 | |
| 4. | Розробка структур окремих підсистем | 25.03.2022-05.04.2022 | |
| 5. | Програмна реалізація системи | 05.04.2022-15.04.2022 | |
| 6. | Оформлення пояснювальної записки | 15.04.2022-20.05.2022 | |
| 7. | Передзахист | 02.06.2022 | |
| 8. | Захист | 14.06.2022 | |

Студент _____ Копійка А. О.

(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Ткаченко В. В.

(підпис) (ініціали, прізвище)

# РЕФЕРАТ

**на магістерську дисертацію**

виконану на тему: Система обробки данних для  Smart City на базі нейронної мережі

студентом: Копійкою Антоном Олександровичем

Дисертація складається із вступу та 4 розділів. Сумарний обсяг роботи – 87 аркушів, містить 24 ілюстрацій та 2 таблиці. В ході роботи було використано 35 джерела.

**Актуальність.** Підтримка якості доріг надзвичайно складна задача оскільки неможливо одночасно контролювати тисячі кілометрів дорожнього полотна, котре може бути пошкоджене під впливом тисячі факторів серед яких, як і природні (різкі перепади температур, опади, зсуви ґрунту, зміни клімату), так і техногенні (перевантаження, знос, недотримання технічних стандартів при будівництві доріг, структурні руйнування, тощо).

Враховуючи величезну протяжність доріг наразі немає єдиної системи яка б могла виконувати моніторинг якості дороги і при цьому не потребувала б великих матеріальних та людських затрат, які б можна було витратити на власне ремонт.

**Мета і завдання дослідження.** Метою магістерської роботи є запропонувати систему, котра б могла фіксувати ями на дорогах, без участі людського ресурсу при обмеженому фінансуванні. В якості рішенням проблеми можна представити вбудовану в автомобіль систему для визначення якості дорожнього покриття на базі нейронної мережі натренованої на показниках лінійного прискорення акселерометра.

Для реалізації визначеної мети були сформульовані наступні завдання**:**

- аналіз існуючих рішень в суміжній області та пошук систем, котрі можуть виступати аналогами для запропонованої;
- вибір технологій, котрі варто використовувати для створення системи;

- проектування та реалізація програмного забезпечення, моделювання нейронної мережі;

**Об’єкт дослідження** – процес збору та обробки інформації про якість дорожнього покриття, який ускладнений необхідністю збирання, оброблення та класифікації великої кількості параметрів в режимі часу наближеному до реального.

**Предмет дослідження** – способи та підходи до збору показників лінійного прискорення та їх подальшого використання для класифікації якості дорожнього покриття за допомогою засобів машинного навчання та використання згорткових нейронних мереж для роботи з часовими рядами.

**Методи досліджень.** В рамках вирішення поставлених задач було використано теоретичні методи досліджень: аналіз та класифікація інформаційної інфраструктури, абстрагування від апаратної та програмної частини задля формування вимог до магістерської дисертації, пояснення обраного стеку технологій та конкретних програмних продуктів, синтез загального способу побудови інфраструктури на різних її рівнях та узагальнення отриманих результатів задля їх візуалізації – а також емпіричний метод експерименту для визначення якості отриманого програмного забезпечення.

**Наукова новизна одержаних результатів.** Запропоновано метод класифікації стану дорожнього покриття з використанням машинного навчання на базі показників лінійного прискорення з використанням згорткових нейронних мереж на відміну від загальноприйнятої регресійної моделі при роботі з часовими рядами. Запропоновано концепт системи для автоматизованого збору та аналізу показників лінійного прискорення під час руху автотранспорту для моніторингу стану доріг з можливістю його інтеграції в масштабну систему Інтернету речей.

**Практична значимість результатів дослідження.** Отримана за результатами роботи система для збору показників лінійного прискорення та класифікатор якості дорожнього покриття, на базі цих данних, може використовуватися для оперативного виявлення дефектів дорожнього поеріття та

подальшого планування ремонтних робіт комунальним службами, а також вибору оптимального маршруту руху для звичайних автомобілістів з врахуванням ямковості дороги.

**Публкікації.** Результати дисертації було апробовано такими публікаціями:

1. Kopiika A., Piskun R., Tkachenko V., Klymenko I. ROAD MONITORING SYSTEM BASED ON IOT TECHNOLOGY FOR SMARTCITY / Information, Computing and Intelligent systems, No. 1 (2020) – Kyiv, 2020. P. 60-67.

**Ключові слова** Інтернет речей, SmartCity, виявлення вибоїн, акселерометр, SPI, STM32, нейронні мережі, часові ряди, CNN.

ЗМІСТ

ВСТУП ..... 10
РОЗДІЛ 1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ..... 12
1.1 Огляд способів збору даних для фіксування якості дорожнього покриття ... 12
1.1.1 Метод паспортизації дорожнього покриття ..... 12
1.1.2 Мобільний застосунок для детекції ям на дорогах ..... 14
1.1.3 Вбудовані в авто системи ..... 15
1.2 Огляд способів обробки данних для пошуку ям на дорогах ..... 16
1.2.1 Алгоритми пошуку ям на дорогах ..... 16
1.3 Постановка завдання ..... 21
Висновки до розділу 1 ..... 23
РОЗДІЛ 2 ПІДГОТОВКА ДО РОЗРОБКИ ПРОЕКТУ ..... 24
2.1 Огляд апаратного забезпечення, використаного в проекті ..... 24
2.1.1 Вибір засобів для отримання показників лінійного прискорення ..... 24
2.1.2Порівняння MEMS акселерометрів ..... 28
2.1.3 Вибір технології для керування периферійними пристроями ..... 33
2.2 Огляд програмного забезпечення, використаного в ході реалізації проекту 43
2.2.1 Огляд систем програмування та відлагодженя роботи мікроконтролерів 44
2.3 Використання мови С для програмування периферійних пристроїв ..... 53
Висновки до розділу 2 ..... 58
РОЗДІЛ 3 РОЗРОБКА ТА ОБГРУНТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ ..... 59
3.1 Визначення структури системи ..... 59
3.2 Опис роботи системи як складової частини проекту Smart City ..... 62
Висновки до розділу 3 ..... 65
РОЗДІЛ 4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ..... 66
4.1 Розробка пристрою для збору данних ..... 66
4.2 Розробка нейронної мережі для класифікації якості дорожнього покриття .. 73

4.3 Розробка програмного забезпечення для відображення роботи системи ..... 76
4.3 Розгляд роботи інтерфейсу програми ..... 79
Висновки до розділу 4 ..... 82
ВИСНОВКИ ..... 83
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 84

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

**Інтернет речей** — концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані давачі, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами в автоматичному режимі, за допомогою використання стандартних протоколів зв'язку.

**Розумне місто** (англ. Smart City) **—** це ефективна інтеграція фізичних, цифрових і людських систем в штучному середовищі заради сталого, благополучного і всебічного майбутнього для громадян.

**Часовий ряд —** це ряд точок даних, проіндексованих (або перелічених, або відкладених на графіку) в хронологічному порядку. Найчастіше часовий ряд є послідовністю, взятою на рівновіддалених точках в часі, які йдуть одна за одною.

**Згорткові нейронні мережі** (англ. convolutional neural network, CNN, ConvNet) — це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень.

**SPI** (англ. Serial Peripheral Interface) **—** фактичний послідовний синхронний повнодуплексний стандарт передачі даних, для забезпечення простого сполучення мікроконтролерів та периферії.

# ВСТУП

Однією з вічних проблем людства залишається покращення транспортної інфраструктури. Увесь цивілізований світ вже давно має розвинену дорожньо-транспортну мережу і єдине, що їм залишається, це підтримувати її в належному стані та запобігати її руйнуванню терміновими або плановими ремонтними роботами.

Вчасне виявлення руйнування дорожнього покриття є надзвичайно важливим для комунальних структур всіх країн світу. Щороку мільйони витрачаються на «залатування» ям та тріщин на дорогах і чим раніше такі ділянки виявлено, тим швидше їх можна локалізувати і дешевше відремонтувати.

Також, варто зазначити, що рух по нерівних дорогах дуже шкідливо впливає на технічний стан транспорту та збільшує кількість витраченого палива аніж на якісному дорожньому покриттю. Неможливо залишити без уваги ДТП, які виникають в зв’язку з низькою якістю дорожнього покриття і нерідко призводять до травмування людей і навіть смертельних аварій.

Підтримка якості доріг надзвичайно складна задача оскільки неможливо одночасно контролювати тисячі кілометрів дорожнього полотна, котре може бути пошкоджене під впливом тисячі факторів серед яких, як і природні (різкі перепади температур, опади, зсуви ґрунту, зміни клімату), так і техногенні (перевантаження, знос, недотримання технічних стандартів при будівництві доріг, структурні руйнування, тощо).

Враховуючи величезну протяжність доріг наразі немає єдиної системи яка б могла виконувати моніторинг якості дороги і при цьому не потребувала б великих матеріальних та людських затрат, які б можна було витратити на власне ремонт.

Вже нікого не здивуєш концепцією «Інтернету речей». Суть цього процесу побудована на обміні інформацією між елементами інфраструктури та предметами людського побуту за допомогою вбудованих датчиків для забезпечення людської безпеки або для покращення якості життя.

Виходячи з вище вказаного, існує необхідність створити систему котра б могла фіксувати ями на дорогах, без участі людського ресурсу при обмеженому фінансуванні. В якості рішенням проблеми можна представити вбудовану в автомобіль систему для визначення якості дорожнього покриття на базі нейронної мережі натренованої на показниках лінійного прискорення акселерометра.

Для реалізації визначеної **мети** були сформульовані наступні **завдання:**

- аналіз існуючих рішень в суміжній області та пошук систем, котрі можуть виступати аналогами для запропонованої;
- вибір технологій, котрі варто використовувати для створення системи;
- проектування та реалізація програмного забезпечення, моделювання нейронної мережі;

# РОЗДІЛ 1
# ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

## 1.1 Огляд способів збору даних для фіксування якості дорожнього покриття

Перед тим, як самому обрати власний спосіб для виявлення нерівностей на дорозі, важливо спочатку розглянути аналогічні системи, які також намагалися вирішити це питання, а точніше, на які показники та характеристики спиралися попередники для визначення якості дорожнього покриття.

### 1.1.1 Метод паспортизації дорожнього покриття

*Мобільні дорожні лабораторії*

Найбільш технологічним методом для визначення якості дорожнього покриття є застосування дорожніх лабораторій – це мобільний центр збору інформації, пов’язаної з дорожнім полотном та узбіччям. Такі автомобілі мають широкий спектр застосування. На прикладі продемонстрованому в [2] в залежності від поставлених цілей може бути розміщено велику кількість різних датчиків:

- відеокамери;
- GPS трекери;
- тривимірні лазерні сканери;
- датчики для аналізу ґрунту;
- пристрій позиціонування.

Такі системи мають широку сферу застосування. Завдяки такій лабораторії можна отримати:

-дослідження поперечного перерізу шарів асфальту;

-визначення осьового відхилення;

-знаходження колій, тріщин та ям;

-перевірка якості нанесеної розмітки;

-виміри загальної рівності дороги.

Більша частина аналізу здійснюється вручну по завершенню поїздки. Данні, зібрані з усіх цих датчиків та відзняті відеоматеріали аналізуються спеціалістами та вручну фіксуються всі ями, пошкодження та проблемні ділянки доріг та узбіч. Весь цей процес є ресурсо- та часозатратним і потребує спеціально навчений персонал для обробки отриманих результатів. Такий метод жодним чином не задовольняє можливості застосування водіями без додаткових навичок і потребує величезних затрат, оскільки це платна послуга, котра вимагає спеціалізований транспорт з дорогим обладнанням, тому він повністю не відповідає поставленим вимогам для роботи задачам.

*Автоматичні методи фіксації ям на дорогах з використанням відеоматеріалів*

Наступним методом для визначення пошкоджень на дорогах є їх фіксація з відеозаписів, який представлений в [3]. В основі пошуку ям цим способом стоїть принцип методу активних контурів [4]. Його суть м полягає в пошуку контурів об’єкта по певній точці, яка точно розташована в межах об’єкту. Метод базується на тому, що для визначення ям використовується дисперсія інтенсивності в її внутрішній області [3]. Окрім цього, як варіант, запропоновано метод знаходження вибоїн з використанням штучного інтелекту на фотографіях [5].

У даного методу є декілька суттєвих недоліків:

- дуже часто «заплатки» на дорогах відрізняються кольором, тому також фіксуються, як ями, що різко підвищує рівень похибки;

- для даного методу важливим є однакове розміщення відеокамери для всіх автомобілів, що абсолютно не можливо, враховуючи конструктивні особливості кожної моделі;
- відео дає високу ймовірність похибки, в зв'язку з перспективним спотворенням зйомки (яма може бути помітна лише частково, або перекрита тінню, проблемним є виявлення в нічний час).

Опираючись на представлені недоліки, цей спосіб також не повністю покриває умови поставленої задачі і має високий рівень похибки, тому також відкидається.

### 1.1.2 Мобільний застосунок для детекції ям на дорогах

Іншим рішенням поставленої задачі є мобільний сервіс для виявлення ям на дорогах. Українська компанія створила мобільний додаток для Android та iOS. Програма автоматично обновлює інформацію стосовно якості доріг і кольором відображає їх на карті. Погані ділянки відмічаються червоним, а кращі – зеленим.

Необхідні данні розробники збирають не самі, а за допомогою користувачів їхнього застосунку. Щоб приєднатися до системи моніторингу достатньо встановити цей застосунок на свій смартфон, зареєструвати свій обліковий запис, дозволити доступ до данних GPS та вбудованого акселерометра і помістити телефон над бардачком свого авто. Зараз кожен телефон за допомогою акселерометра може визначити силу з якою він був зрушений з місця. Таким чином розробники отримали тисячі рухомих автолабораторій, котрі збирають для них данні в обмін на вже зібрані іншими користувачами. Вся отримана інформація обробляється з використанням їх авторських математичних алгоритмів. В результаті всього цього процесу і опираючись на данні геолокації отримані з GPS

датчика, додаток відображає якість дороги на карті і допомагає обрати маршрут з кращим дорожнім покриттям. [6,7]

З огляду на зазначене вище, це, можливо найекономніший спосіб, оскільки він не потребує жодних зайвих затрат як зі сторони розробника, так і зі сторони звичайного власника авто. Проте є декілька суттєвих недоліків в данному підході. Перш за все, це велика енергозатратність при роботі даного застосунку через постійну активну роботу дачика GPS. Як результат, ви можете виявити, що ваш телефон цілком розрядився після довгої поїздки і не може бути використаним за прямим призначенням. Також, телефон має лежати в певному статичному положенні протягом всього руху, оскільки при пересуванні телефону будуть отримані неправильні показники, що позбавляє можливість користування телефоном для своїх потреб, в якості навігатора або для інших цілей, що додає зайві незручності кінцевому користувачу.

### 1.1.3 Вбудовані в авто системи

Важливість рішення поставленої в роботі задачі доказує те, що великі автомобільні компанії на зразок Volvo, Ford та Jaguar займаються роботою в суміжній сфері і самі ведуть розробки систем для пошуку ям на дорогах. Загалом кожна з цих компаній пропонує схожі концепції для детекції дефектів дорожнього покриття. Згідно з інформацією з відкритих джерел, їх система зможе визначати ями, вибоїни та каналізаційні люки, за допомогою сенсорів, розміщених під їхніми автівками. Таким чином ці датчики зможуть комунікувати з бортовим комп’ютером машини і у випадку виявлення проблемних ділянок, знижувати швидкість, або змінювати конфігурацію підвіски задля підвищення комфорту та покращення руху машини по нерівній дорозі.

Окрім того, інформація про виявлення проблем дорожнього покриття будуть надіслані на хмарні сховища з координатами місця пошкодження. Завдяки цьому, наступні машини цих брендів будуть завчасно отримувати повідомлення про ускладнення руху при потраплянні на цю ділянку дороги. Така система легко вписується в парадигму Інтернету речей, де машини зможуть обмінюватися інформацією про виявлені ями, за допомогою спільної бази данних зібраної кожною з них окремо.

На даний момент автомобілі можуть фіксувати ями лише на короткій дистанції від машини, проте планується встановити стереоскопічні цифрові камери, котрі могли б знаходити ями на відстані, якої б вистачило для того, щоб встигнути завчасно попередити, знизити швидкість, або зупинити авто в критичних випадках. [8,9,10]

## 1.2 Огляд способів обробки данних для пошуку ям на дорогах

В ході огляду відкритих джерел, було знайдено декілька методів виявлення проблемних ділянок на дорозі за допомогою датчика лінійного прискорення. Серед усіх знайдених матеріалів, розглянуті реальні кейси, які представлені в закордонних статтях і були реалізовані на практиці в проектах на подібну тему. Перед початком власної розробки важливо розглянути існуючі підходи для підбору оптимального саме для запланованої системи.

### 1.2.1 Алгоритми виявлення ям на дорогах

Оскільки даною технологією займаються всесвітньо відомі компанії, то знайти саме їх алгоритми у відкритому доступі неможливо. Проте, в ході пошуку в

Інтернеті було знайдено кілька закордонних статей, які демонструють власні алгоритми пошуку ям на дорозі.

*Z –THRESH*

Першим методом для пошуку ям на дорозі, який приходить на думку є Z – THRESH. Сутність цього методу полягає в моніторингу показників датчика лінійного прискорення по вертикальній осі Z (рис. 1.1). В дослідженнях проведених в статті [16] було проведені заміри лінійного прискорення при русі машини по дорогах різної якості. Проаналізувавши отримані данні, було встановлено пороговий показник прискорення, перевищення якого вказує на потрапляння в яму. Для цього алгоритму варто зауважити, що положення осей акселерометра відповідає звичному і не потребує переорієнтування. В представленому експерименті акселерометр знаходився в горизонтальному положенні, що забезпечує відсутність необхідності додаткових розрахунків проекцій прискорень.

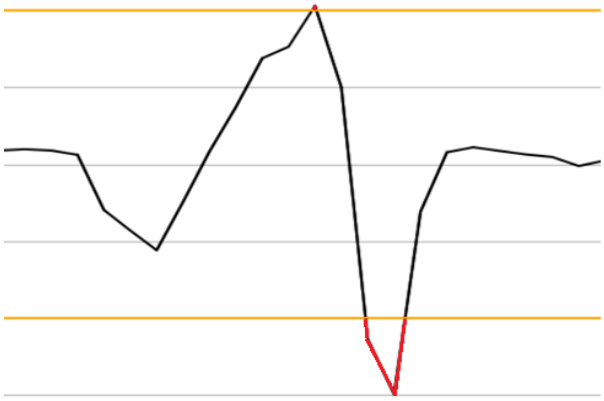

Рис. 1.1. Приклад класифікованої ями за допомогою алгоритму Z –THRESH [16]

*Z – DIFF*

Цей метод також базується на аналізі показників вертикального прискорення, проте цей трохи складніший за попередній. Для аналізу йому потрібен той же набір

даних, як і попередньому алгоритму, але якщо для Z –THRESH відслідковує досягнення граничного значення лінійного прискорення відносно стану спокою, то Z-DIFF опирається на різкі зміни показників в порівнянні з попереднім вимірюванням (рис. 1.2). Цей алгоритм фіксує швидкі зміни вертикального прискорення і визначає такі ділянки як вірогідні ями. Приклад графіка прискорення пердставлений на рисунку нижче.

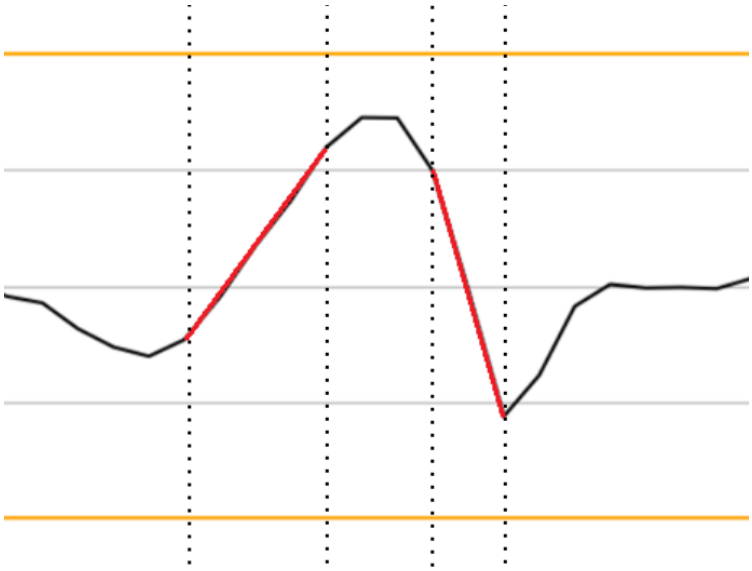

Рис. 1.2. . Приклад класифікованої ями за допомогою алгоритму Z –DIFF [16]

*G-ZERO*

Метод з назвою *G-ZERO* своєю назвою чітко ілюструє головний його принцип. Він був виявлений в ході аналізу данних, отриманих за показниками трьохосьового акселерометра протягом двох попередніх дослідів. Під час розгляду показників прискорення під час потрапляння в яму на один момент акселерометр фіксує прискорення на всіх осях рівне нулю (рис1.3).

Емпіричний аналіз, представлений в статті [16] підштовхує на два припущення:

1. такі результати означають, що автомобіль при потряплянні в яму на дорозі потрапляє в стан невагомості на певні долі секунди;
2. для фіксаціїї ями на дорогах цим методом, можна використовувати значення прискорення всіх трьох осей, без урахування розташування акселерометра в автомобілі, та необхідності передобробки отриманих показників.

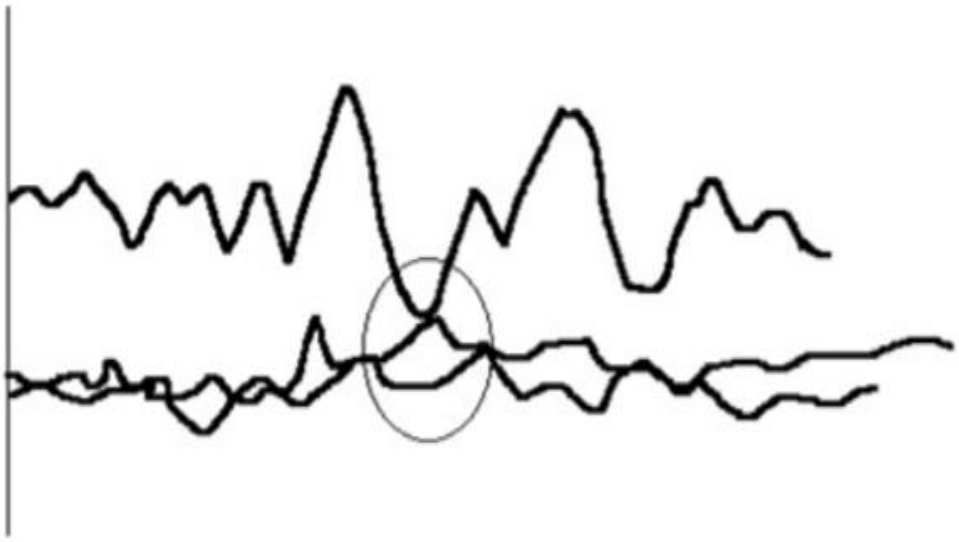

Рис. 1.3. Ймовірна яма згідно з алгоритмом G –ZERO [16]

*Алгоритм Pothole Patrol*

В статті [17] представлена система визначення ям на дорогах під назвою P² (Pothole Patrol). Алгоритм роботи даної системи є більш комплексним, так як ця система вміє класифікувати різні види дорожніх нерівностей, а саме:

-рівна дорога;

-ями;

- дорожні компенсатори;

-дорожні «рельси»;

-лежачі поліцейські;

-каналізаційні люки.

Класифікація базується на показниках, отриманих з осей X та Z акселерометра а також швидкості автомобіля, для покращення точності алгоритмів (рис. 1.4). В ході розробки були проведені тестування на різних ділянках доріг та різних покриттях в Бостоні та в його передмісті.

В ході досліджень приймали участь працівники таксі з різною манерою керування на різних автомобілях різного року випуску. Кожен автомобіль був обладнаний комп'ютером Soekris 4801 з ОС Linux з підключеним до нього трьохосьовим акселерометром. Цей датчик розміщений в кожному автомобілі в однаковому горизонтальному положенні. В ході тестування протягом декількох тижнів було перевірено близько 2500 км дорожнього покриття. Дана система показала дуже гарні результати – лише 2% похибки при виявленні ям.

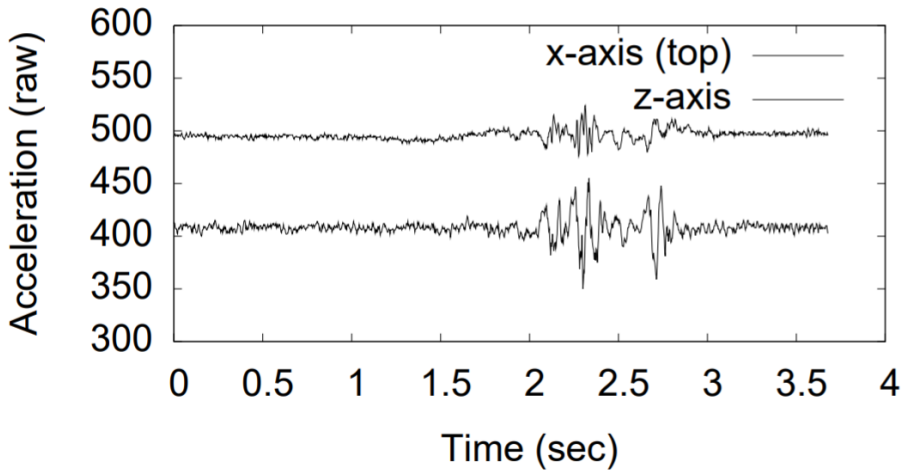


Рис. 1.4. Графік прискорення на ділянці з ямою для алгоритму P² [17]

*Використання машинного навчання для знаходження ям*

Останній з розглянутих методів був представлений в статті [18]. В ній розглядається варіант виявлення ям, за допомогою вбудованого акселерометра

використовуючи вбудований акселерометр телефона. Основною відмінністю даного методу від всіх попередніх є використання машинного навчання для класифікації. В якості класифікатора в даній роботі був використаний наївний байєсовий класифікатор. Натренована нейронна мережа на показниках прискорення по осі Z. Датасет скаладється з коротких відрізків замірів на різних ділянках дороги.

Загальний принцип полягає в тому, що під час руху показники акселерометра розбиваються на невеликі частини і перевіряються за допомогою класифікатора. В результаті ми отримуємо набір ділянок, кожна з яких відповідає або рівній дорозі або дорозі з ямами. Натренована нейронна мережа показала точність на рівні 82,857%.

## 1.3 Постановка завдання

В рамках магістерської дисертації, згідно з обраною темою, необхідно розглянути усі існуючі апаратні технології в предметній області, задля вибору найкращого апаратного забезпечення, а також виконати порівняння технологічних рішень. Опираючись на обрані технології важливо обрати програмне забезпечення, для реалізації програмної частини а також оптимальне середовище розробки.

Подальшим кроком є розробити загальний алгоритм для виявлення ям на дорогах. На етапі розробки алгоритму реалізувати методи для зчитування та обробки даних з дачиків, та збереження в хмарних сховищах, обробки результату. Побудувати апаратну модель пристрою.

Завершальним етапом є реалізація системи детекції ям за допомогою датчика лінійного прискорення з такими функціями: отримання показників з датчика, фільтрація шумів, відправка даних в хмарні сховища із їх подальшою обробкою.

Створити та натренувати нейронну мережу для класифікації ям. Результати отриманої нейронної мережі необхідно протестувати.

Також, однією з поставлених цілей є розробка методичних матеріалів для набору лабораторних робіт на базі окремих частин диплому.

## Висновки до розділу 1

1. В ході огляду існуючих рішень було виявлено, тематика є актуальною, взвязку з користю, яку вона може принести, як для дорожніх служб, так і покращити якість руху простих автомобілістів. Над вирішенням питань в цій предметній області займаються державні структури, автомобільні концерни та науковці по всьому світу.
2. Розглянуті різні методи вирішення задачі виявлення ям на дорогах. Визначено їх переваги та недоліки. Представлено варіанти вже існуючих алгоритмів для пошуку ям на дорогах.
3. Під час огляду літературних джерел встановлено, що головним і найбільш доцільним підходом для вирішення поставленої задачі є використання акселерометра. Це пояснюється його простотою, низькою вартістю, наглядністю результату та високою відмовостійкістю.
4. За результатами огляду літератури були поставлені задачі магістерської дисертації, а саме реалізація програмного і апаратного забезпечення, а також нейронної мережі для класифікації якості дорожнього покриття, як частини системи Інтернету речей.

# РОЗДІЛ 2
# ПІДГОТОВКА ДО РОЗРОБКИ ПРОЕКТУ

В першому розділі було розглянуто кілька абсолютно різних підходів до вирішення поставленої задачі, тому виникає необхідність вибору технологій, які було б доцільно використовувати саме в цьому проекті, а саме правильно обрати програмні та апаратні засоби, методи обробки інформації, мову програмування та середовище розробки). В звязку з цим твердженням необхідно провести огляд існуючих технологій.

## 2.1 Огляд апаратного забезпечення, використаного в проекті

На сьогоднішній день ринок пропонує широкий спектр цифрових датчиків, котрі мають широке застосування в безлічі проектів в залежності від тих чи інших потреб. В цьому підрозділі мною буде розглянуто необхідне апаратне забезпечення і обрано саме те, яке необхідно для подальшого використання в роботі.

### 2.1.1 Вибір засобів для отримання показників лінійного прискорення

Головним засобом для отримання значення лінійного прискорення є використання акселерометрів. Всього акселерометри бувають трьох видів: одно-, двох- та трьох осьові. Різниця полягає лише в тому в скількох площинах вимірюється значення лінійного прискорення. Найбільш універсальними є трьохосьові, оскільки вони можуть покрити усі задачі в не залежності від потреб, що спричиняє їх поширеність. [19]

Дані прилади, в залежності від принципу роботи поділяються на :

- п’єзоелектричні;

- поверхнево інтегральні;
- об’ємні інтегральні.

*П’єзоелектричний датчик*

Конструктивно, п’єзоелектричний датчик являє з себе стержень, який спричиняє тиск на п’єзокристал. Фізичний вплив на датчик призводить до вібрації стержня, що спричиняє до вироблення електричного струму, сила якого пропорційно відповідає прискоренню, яке діє на даний датчик. У даного виду датчиків досить високий рівень похибки, пов'язаний з залежністю показників від температури, вологості та тиску, проте головним недоліком є неможливість їх випуску в промислових масштабах через відмінність кожного своїми параметрами конфігурації.

*Об’ємні інтегральні датчики*

Головною відмінністю об’ємних інтегральних від п’єзоелектричних датчиків є їх конструкція і принцип роботи. Вони складаються з двох сплавлених пластин кремнію. На цій пластині розміщені тонкі смужки з приєднаною до кінців інерційною масою. До одного боку рамки приєднано вагомий важок. Такі пластини обладнані кількома імплементованими п’єзорезисторами, які утворюють напівміст. Рухаючись з прискоренням, маса починає рухатися в сторони і згинати балки, тим самим викликаючи деформацію п’єзорезисторів. Завдяки такій конструкції, електронна схема обробки сигналів, розміщена за межами кристалу генерує електронний сигнал в датчику напругою від 50-100 мВ. Акселерометри з об'ємною конструкцією мають ряд недоліків. Головна проблема – це складність у виробництві, так як об’ємні елементи на платі дуже складно з’єднувати з плоскою інтегральною схемою. Також, одним з недоліків є великі розміри датчика. При

зменшенні розмірів кристала підвищується його механічна міцність, що знижує його вартість. Тільки для розміщення чутливого елемнта на об’ємному датчику потрібна достатньо велика площа плати. А враховуючи ще і те, що окрім датчику потрібно також розмістити на платі ще і схему для формування сигналу, то це призведе до збільшення зайнятої площі на платі збільшиться принаймні в два рази. [20]

*Поверхнево інтегральний датчик*

Якщо розглядати всі існуючі датчики, то оптимальним з точки зору ціна/якість є поверхнево інтегральний датчик. Датчики такого типу в загальному носять назву MEMS (Мікроелектромеханічна система). Загальний принцип роботи даного типу датчиків полягає на розрахунку зсуву інерційної маси, відносно стану спокою та перетворення цього значення в відповідний електричний сигнал. Поміж всіх існуючих варіантів ємнісний спосіб відображення є найбільш точним, тому він є найпоширенішим серед усіх. Принцип роботи датчику такого типу наступна – датчик складається з двох частин. Більша частина такого датчика це схема формування сигналу, а друга частина це крихітний датчик прискорення, який розміщений посередині кристалу, так як зображено на рис.2.1. На кристалі розміщенна велика кількість конденсаторів наповнених повітряним діелектриком з обкладками, зробленими зі смужок полікремнієвої плівки. Нерухомими обкладинками цього конденсатора є стрижні, які розміщені на 1 мкм вище поверхні кристала в повітрі на стовпчиках-анкерах з полікремнію, з’єднаних з кристалом на молекулярному рівні. При русі стрижнів між обкладинками змінюються конфігурація конденсатора, що в свою чергу призводить до зміни показників напруги. Отримані показники напруги перетворюються в цифрові значення, які пропорційні показникам прискорення. [21]

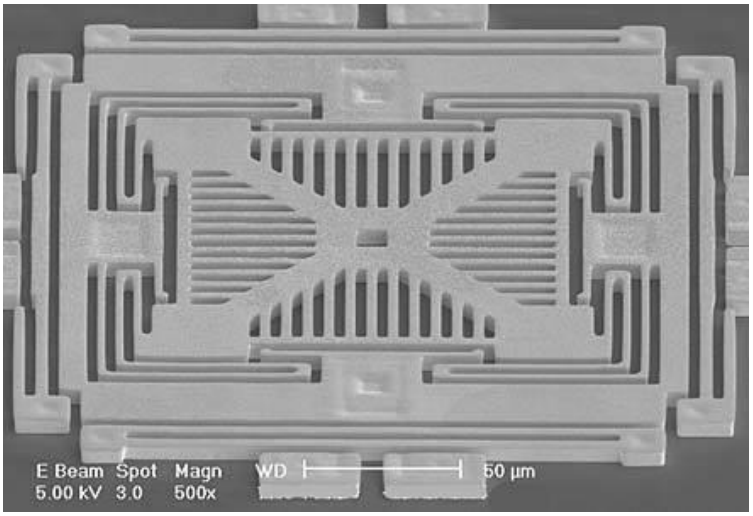


Рис. 2.1 Внутрішня структура MEMS акселерометра

Таблиця 2.1

Порівняння типів датчиків лінійного прискорення

| Тип | Точність | Ціна | Спеціалізація | Особливості |
|---|---|---|---|---|
| П’єзоелектричні | Низька | Низька | Удари, вібрації | Показники приблизні. Висока чутливість до температури та тиску. |
| Поверхнево інтегральні | Середня | Середня | Нахили, вібрації, інерційні сили | Низький рівень шуму. Великий розмір. Складно налаштовується. |
| Об’ємні інтегральні | Висока | Висока | Нахили, вібрації, інерційні сили | Малий розмір, завершена конструкція. |

Головними перевагами даного типу акселерометрів є простота в виробництві і малі розміри. Це означає, що і ціна на такі датчики є низькою в порівнянні з аналогами і вони широко представлені на ринку мікроелектроніки. Також варто зазначити, високу точність показників датчиків данного типу, достатні для проектів будь якої складності. За результатами даного підрозділу представлено порівняння оглянутих технологій на табл.2.1.

З огляду на інформацію, висвітлену вище, вирішено використовувати поверхнево інтегральний датчик для вирішення поставлених задач.

### 2.1.2 Порівняння MEMS акселерометрів

Обравши необхідний засіб для отримання показників лінійного прискорення також потрібно провести порівняти існуючих поверхневих інтегральних датчиків та обрати серед них модель, яка б задовольняла необхідні потреби.

При виборі конкретного датчика було розглянуто датчики компанії STMicroelectronics. Дана компанія заслужено займає перше місце серед виробників мікроелектроніки усвіті. Контролери та датчики виготовлені цією компанією користуються попитом для розробки у проектів будь якої складності. Це пояснюється високою якістю пристроїв і якісною офіційною документацією для роботи з ними. Усі акселерометри, виготовлені даною компанією мають схожу функціональну структуру (рис.2.2). Показники напруги отримані аналого-цифровим перетворювачем перетворюються на цифровий сигнал і видаються датчиком у вигляді 16 бітів. Після сумування 8 старших і 8 молодших бітів данні проходять через відновлюючий фільтр і стають доступними для отримання за допомогою інтерфейсів передачі даних, таких як I2C або SPI.

Для розгляду було обрано три моделі акселерометрів, а саме LSM9DS1, LSM303DLHC та LIS3DSH.

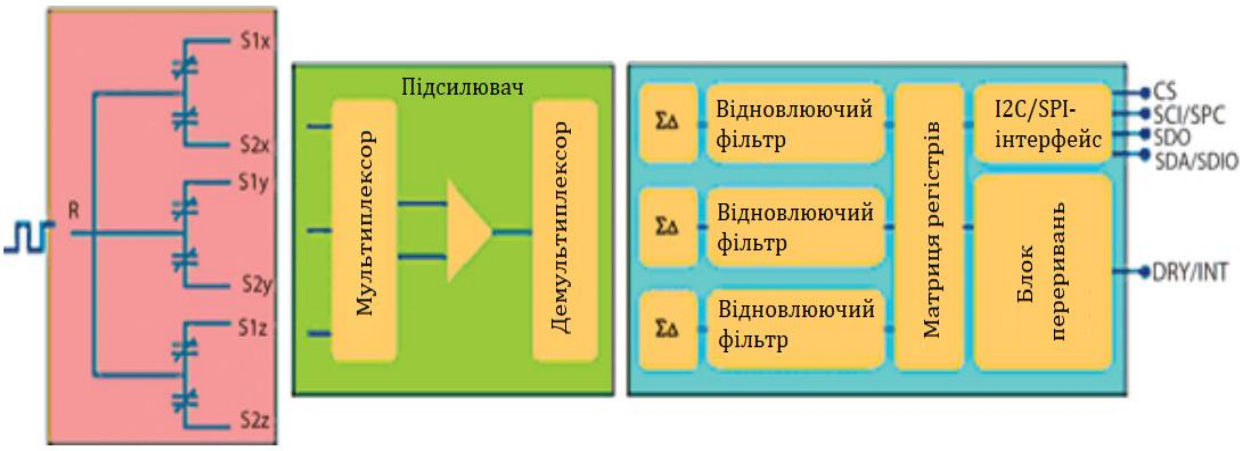


Рис. 2.2 Функціональна схема акселерометра STMicroelectronics [22]

*LSM303DLHC*

Як зазначено в офіційній документації, датчик LSM303DLHC [22] це компактний акселерометр та магнітометр. Головна різниця між даним датчиком, та аналогами – це об'єднанання разом двох датчиків на одному кристалі. В додаток до акселерометра, даний датчик оснащений магнітометром – пристроєм для орієнтації в просторі. Акселерометр, який розміщений в цьому датчику підтримує визначення показників лінійного прискорення для трьох осей. Даний акселерометр можна програмно налаштовувати на різні діапазони граничних показників від ±2g до ±16g, де g=9.8 м/с$^2$ - прискорення вільного падіння. Обмін данними між датчиком відбувається за допомогою інтерфейсу I2C (Inter-Integrated Circuit) по шині. Шина представляє з себе з дві двонапрямлені лінії – послідовну лінію даних та послідовну лінію тактування, приєднані до живлення, та керовані за допомогою відкритого колектора. Дана шина підтримує два режими роботи: стандартний (100кГц) і швидкий (400кГц). Робочою для датчика є напруга в діапазоні від 2.16В до 3.6В. Його розміри всього 3*5*1мм.

*LIS3DSH*

Наступним варіантом для порівняння розглядається MEMS акселерометр LIS3DSH. На відміну від попереднього датчику, спеціалізація цієї моделі поширюється лише на визначення лінійного прискорення. Так як і попередній, він також підтримує вимірювання одночасно в трьох проекціях і працює для визначення прискорення в рамках +-2-16g, але дану модель можна налаштувати на проміжні значення в цьому діапазоні, для економії заряду. Головними його перевагами є низька енергозатратність і компактність на кристалі. Гранична допустима напруга для роботи даного датчика складає всього 1.71 В, а загальний розмір складає всього 3*3*1мм, тобто він майже вдвічі менший в об'ємі за попередній. Окрім цього даний акселерометр показує дуже високий рівень стабільності – він може витримати перевантаження до 10000g. [23]

Для обміну даними можна використовувати один з двох підтримуваних інтерфейсів – вже раніше згаданий I2C, а окрім нього послідовний периферійний інтерфейс SPI. Даний інтерфейс створений для двосторонньої комунікації між контроллером і периферією. SPI це синхронний інтерфейс. Це означає, що обмін данними відбувається на тій же тактовій частоті, що і у мікроконтролера. Цей інтерфейс побудований на спілкуванні одного ведучого елементу (контролер), і одиного або декілька ведених елементів (датчики, периферійні пристрої). Комунікація відбувається по 4 лініям зв'язку, як це представлено на рис.2.3:

- Master Out Slave In — вихід мікроконтролера – вхід периферії. Він виконує функції передачі данних з ведучого до веденого;
- Master In Slave Out — вхід мікроконтролера – вихід периферії. Він виконує функції для передачі даних від веденого до ведучого;
- Serial Clock — послідовний тактовий сигнал, для тактування від мікроконтролера до перефирії;

- Chip Select — вибір елемента, з яким мікроконтролер обмінюється данними.

Алгоритм роботи інтерфейсу базується на обміні данних між зсувними регістрами контролера та периферійних пристроїв. Для вибору необхідного адресата подається сигнал по піну CS. А потім за допомогою зсуву регістрів вправо останні 8 біт регістру веденого переносяться в перші 8 біт ведучого та в зворотньому напрямку через шини MISO та MOSI, так як зображено на рис.2.3. Частота при якій відбувається комунікація рівна тактовій частоті ведучого пристрою. [24]

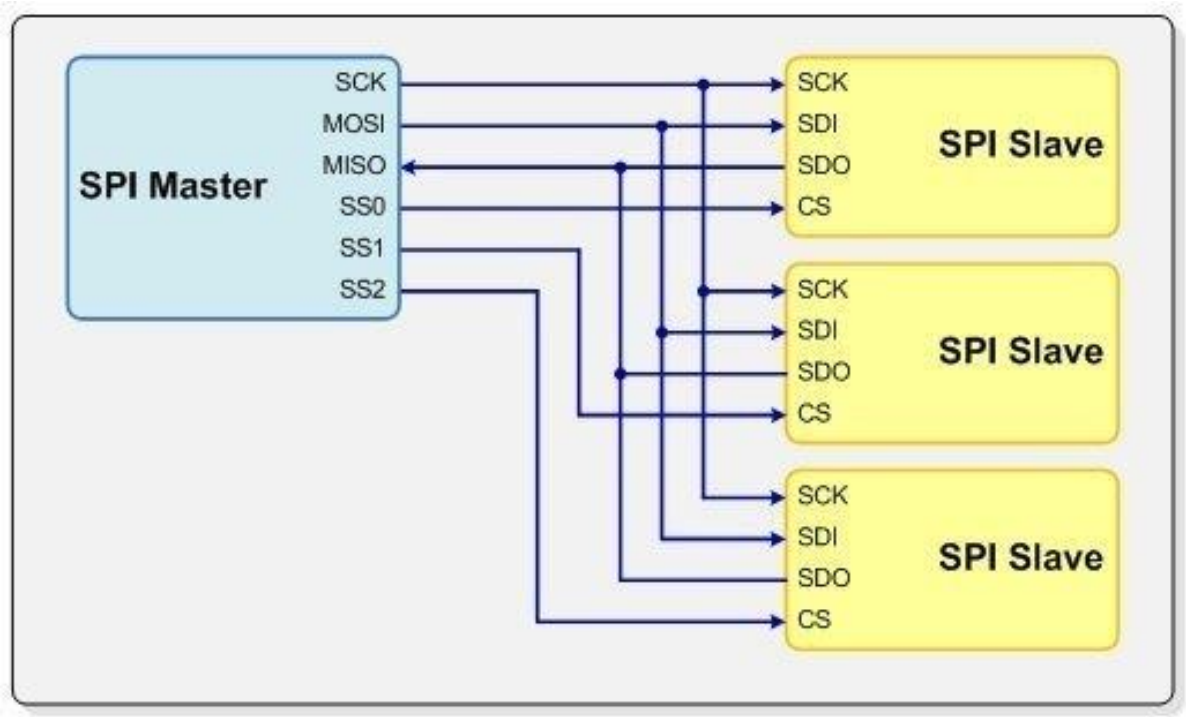


Рис. 2.3 Структурна схема комунікації в SPI [24]

Головною перевагою цього інтерфейсу є можливість комунікації контролера одразу з багатьма периферійними пристроями, що дуже зручним для систем, обладнаних великою кількістю датчиків.

*LSM9DS1*

Останнім серед розглянутих було обрано датчик LSM9DS1. Серед усіх представлених він має найбільший функціонал. Його особливістю є те, що на одному чіпі розміром 3*3.5*1 було розміщено одразу три різних датчика: акселерометр, гіроскоп та магнітометр. Такий вид периферійних пристроїв має назву інерціальний модуль. Така багатофункціональність пристрою при таких розмірах вражає, проте для забезпечення такої різноманітності можливостей, розробникам довелося пожертвувати певними характеристиками, в порівнянні з іншими представниками порівняння. Перш за все, підвищилась його енергозатратність. Необхідна напруга для роботи має варіюватися в межах від 1.9В до 3.6В. Також відсутня можливість налаштування границь вимірювання. Це призводить до зниження точності вимірювання проміжних результатів. Незмінним залишається лише підтримка інтерфейсів SPI та I2C. [25]

Таблиця 2.2

Порівняння акселерометрів компанії STM32

| Назва | Функціонал | Вольтаж | Способи передачі даних | Межі вимірювання | Чутливість |
|---|---|---|---|---|---|
| LSM303DLHC | Акселерометр, магнітометр | 2.16-3.6В | I2C | ±2g/±8g/±16g | 0.09g |
| LIS3DSH | Акселерометр | 1.71-3.6В | SPI та I2C | ±2g/±4g/±6g /±8g/±16g | 0.06g |
| LSM9DS1 | Акселерометр, магнітометр, гіроскоп | 1.9-3.6В | SPI та I2C | ±2g/±4g /±8g/±16g | 0.061g |

З огляду на інформацію, представлену вище, та посилаючись на данні з табл.2.2, найкращим вибором для вирішення поставлених задач є використання акселерометра LIS3DSH, оскільки його можливостей достатньо для покриття всіх вимог, а також він є дуже енергоекономним. Акселерометр LSM9DS1 має гарні показники з огляду на доступний функціонал, проте в представленій роботі відсутня необхідність в додаткових датчиках, тому значно правильнішим є використання спеціалізованого пристрою.

### 2.1.3 Вибір технології для керування периферійними пристроями

Важливу роль для точності фінальних результатів займає достовірність отриманих данних з периферійних пристроїв. Для цього надважливо використати такий контролер, який би повністю відповідав поставленим вимогам, а саме швидкодія, низька енергозатратність та висока швидкість розробки для данного апаратного забезпечення. В цьому підрозділі буде проведено огляд відомих технологій обробки даних з периферійних пристроїв.

*Arduino*

Перша пропозиція, яка приходить на думку при пощуці недорогих та протих плат для керування периферійними пристроями це однопроцесорні цифрові плати Arduino.

Використання Arduino найчастіше є ідеальним для початківців, або навіть для першого проекту людини без навичок в комп’ютерній мікроелектроніці. Плати Arduino дуже легкі для програмування. Це зумовлено тим, що програмування для даних пристроїв представляє з себе набір послідовних команд контролеру розміщеному на платі і не потребує жодних передналаштувань і конфігурації.

Спеціально для цього розробники цих контролерів створили власну мову програмування Arduino, а також середовище Arduino IDE для розробки на ній. [11]

В звязку зі своєю простотою, контролери Arduino зарекомендували себе важливою частиною багатьох проектів, починаючи від дрібних студентських проектів до елементів Інтернету речей. Загалом плата складається з мікроконтролера Atmel AVR з завантажувачем та набору пінів для приєднання додаткових периферійних пристроїв. Плата Arduino працює під напругою від 3.3В до 5В з тактовою частотою 8 або 16МГц. На платі розміщено багато як аналогових так і цифрових входів та виходів. Компанія розробник пропонує різні комплектації плат. Їх характеристики варіюються в залежності від ціни і реальної потреби.

Мовою програмування для Arduino являється C++ з домішками спеціалізованих бібліотек для керуванням вводу/виводу з контактів плати. Написані програми легко прошиваються на контролер, приєднаний до комп’ютера за умови використання середовище розробки Arduino IDE.

Проекти, створені з використанням Arduino легкі для реалізації, оскільки для додавання окремих деталей достатньо приєднати плату до макетної дошки і не витрачати час на зпаювання проводки. [12]

Основними перевагами Arduino є:

- низька вартість;
- середовище розробки працює майже на всіх операційних системах;
- низький рівень входження в розробку;
- програмне забезпечення з відкритим кодом і можливістю написання власних бібліотек.

*Raspberry Pi*

Raspberry Pi – це міні комп’ютер, всі елементи якого приєднані до однієї плати розміром 8,5*5,5 см. Область його застосування варіюється в залежності від того, які периферійні пристрої додатково приєднані до плати. Розробники цифрових пристроїв застосовують Raspberry Pi в якості аудіосистем, метеостанцій та навіть персональних комп’ютерів. До цього міні-комп’ютера можна підключити будь які пристрої, (мишку, клавіатуру, монітор, тощо). Raspberry Pi складається з процесора частотою 700 МГц та відеокарти VideoCore IV а також 512 мегабайтів оперативної пам'яті. Процесор вбудований в Raspberry Pi - це 4-ядерний ARM Cortex-A53 - 32-бітний RISC процесор, який широко розповсюджений для використання в портативних пристроях. Роль внутрішнього жорсткого накопичувача в даному комп’ютері виконує SD карта. На цей комп’ютер можна встановити навіть дистрибутив Linux. Офіційною мовою розробки програмного забезпечення для даної інфраструктури є Python. [13]

Головною перевагою цієї архітектури є її енергоефективність, що і є її головною перевагою для розробників, перед якими стоїть задача розробки систем з малим навантаженням.

*Мікроконтролери STM*

На сьогоднішній день мікроконтролери STM32 вважаються основним рішенням для побудови систем обробки сигналів. Це пояснюється тим, що на контролерах розміщені спеціалізовані цифрові сигнальні процесори ARM Cortex. Термін цифровий сигнальний процесор, або Digital Signal Processor - це мікропроцесор, який спеціалізується на цифровій обробці сигналів, саме тому така

архітектура поширена для вирішення задач збору і обробки показників в мікропроцесорних системах.

Найпоширенішими задачами цифрової обробки сигналів є:

- пошук сигналів;
- фільтрація;
- перетворення Фур’є.

Головною перевагою цифрових сигнальних процесорів на відміну від інших є:

-операції множення з накопиченням протягом одного машинного циклу;

-векторно-конвеєрна обробка, завдяки генераторам адресних послідовностей

- циклічне виконання набору певних команд на апаратному рівні;

-можливість одночасного доступу до декількох комірок пам’яті. [14]

Окрім цього великим плюсом мікроконтролерів STM є невисокі витрати енергії. Під час роботи в режимі «Dynamic RUN» він працює на частоті 84 МГц при показниках струму, які не будуть перевищувати 12 мА. [15]

В додаток до цього, в даних мікросхемах на програмному рівні реалізована робота з пам’яттю. Також присутні високорівневі API для програмування контролера на мовах C та C++, тому вони добре програмуються без врахування структурою і топологією плати, оскільки відсутня потреба в посиланні директив на пряму до елементів архітектури МК.

Під час огляду літератури, було прийнято рішення, що для вирішення поставлених задач необхідно обрати мікроконтролер котрий би займався збором даних отриманих з акселерометра. В даній роботі було вирішено використати один

з контролерів компанії STMicroelectronics. Перш за все, попередньо обраний а кселерометр належить даній компанії, а отже він має легко конфігуруватися з платою. По-друге, в лінійці контролерів STM є велика кількість плат різної потужності, розмірів та комплектації, що надає широкий вибір і є можливість обрати саме такий, який буде найкраще підходити для наших задач. По-третє, програмування мікроконтролерів STMicroelectronics реалізується на мові програмування високого рівня, що підвищує швидкість розробки. Також варто зазначити, що програми написані для однієї моделі могуть бути переналаштовані на іншу плату. Для цього потрібно лише зробити правки без конфігурації без зміни вихідного коду. В роботі буде представлено огляд сімейства мікроконтролерів STM32F.

*Огляд мікроконтролерів серії STM32F*

Мікроконтролери серії STM32 F4 вважаються флагманськими в лінійці контролерів STM32. Головна їх відмінність від інших, це використання міропроцесора ARM Cortex-M4. В порівнянні зі старішими версіями вони значно виграють у попередників у розрахунковій потужності. Ще одним значими плюсом є використання методу виготовлення мікроконтролерів ST Microelectronics ART Accelerator, який значно підвищує показники продуктивності мікроконтролерів побудованих на базі мікропроцесорів Cortex-M. Ця методика дозволяє отримати високі показники продуктивності - 225 DMIPS / 606 CoreMark, а також можливість роботи з флеш-пам'яттю на частоті 180 МГц. Окрім цього, в контролери даної лінійки вживлений модуль для операцій з числами з плаваючою комою, що значно збільшує спектр можливого застосування цього апартного забезпечення. Також в даних мікросхемах використовується методика динамічного споживання електроенергії, що значно знижує показники енергоспоживання. Цей метод

допомагає отримати показники витрати живлення складають 140 мкА/МГц у STM32F401 і 238 мкА/МГц у STM32F42x/43x на максимальних частотах.

Мікроконтролери серії STM32 F4 об'єднують в собі можливість роботи в реальному часі і високу продуктивність сигнальних процесорів при обробці даних і реалізують клас сигнальних мікроконтролерів. Серія представлена з п'яти підкласів контролерів. Усі представники серії STM32 F4 сумісні між собою з точки зору програмного коду та взаємодії з периферійними пристроями. [26]

Загальні характеристики контролерів серії:

- процесор ARM Cortex-M4 CPU;
- Виконання DSP-інструкцій;
- Шинна матриця AHB;
- Діапазон напруг 1,8…3,6В;
- RC-генератори з частотою 16МГц та 32кГц;
- Зовнішній генератор переривань від 4МГц до 26МГц;
- Пристрій відладки SWD/JTAG;
- 12-бітний аналогово-цифровий перетворювач;
- Сімнадцять вбудованих шістнадцяти та тридцяти двох розрядних таймерів;
- Вбудований генератор псевдовипадкових чисел;
- Підтримка інтерфейсів комунікації: I2C, I2S, USART, SPI;
- USB 2.0;
- Контролер SDIO;
- Робочий температурний діапазон від -40 до 105°C.

Як бачите, контролери цієї серії мають дуже багато спільних рис. Повний список можна знайти на рис.2.4. В серії представлено 5 лінійок пристроїв. Відрізняються лінійки між собою лише деякими характеристиками і додатковим функціоналом.

Головні відмінності лінійок серії:

- STM32F401– 84 МГц CPU/105 DMIPS. Головною відмінністю лінійки від інших є найменший розмір та енергоспоживання від інших, з чого слідує і найменша потужність;
- STM32F4x5 – 168 МГц CPU/210 DMIPS. Вбудовано додатково 1 МБайт флеш пам’яті з можливістю до сполучення та шифрування;
- STM32F4x7 – 168 МГц CPU/210 DMIPS. Окрім додаткового мегабайту флеш пам’яті реалізований інтерфейс Ethernet MAC;
- STM32F427/437 – 168 МГц CPU/210 DMIPS. Містить 2 мегабайти флеш пам’яті. По функціоналу вміщає в себе можливості STM32F405/415 та STM32F407/417;
- STM32F4x9 – 180 МГц CPU/225 DMIPS. Містить 2 мегабайти двухбанкової флеш пам’яті з інтерфейсом SDRAM. Обладнаний дисплеєм TFT LCD, та новітньою технологією Chrom-ART, яка дозволяє значно збільшити потужність при меншому енергоспоживанні в порівнянні з мікроконтролерами лінійки STM32F4x7/F4x5.

Великою перевагою контролерів STM32 F4 є розмішення на платі великої кількості портів вводу/виводу GPIO (General purpose input-output). Під назвою порти загального призначення мається на увазі, що поведінка портів програмується від поставленої задачі. Такі порти не вхідними або вихідними для жодного вузла системи. В початковому стані вони не відповідають за буть який функціонал. Вони необхідні для приєднання додаткових периферійних пристроїв на платі. В контролерах STM32 використовуються 16-ти розрядні порти. Вони повністю

незалежні між собою і кожен може відповідати своїй власній задачі. Кожен порт може бути конфігурований на такі режими роботи:

- Input floating - вхід без підтягуючого резистора;
- Input pull-down - вхід з підтягуючим резистором, приєднаним до землі;
- Input pull-up - вхід з підтягуючим резистором, приєднаним до живлення;
- Analog - аналоговий вхід;
- Output push-pull - активний вихід. При негативному логічному вираженні не видає напругу, а при позитивному – подається напруга, рівна живленню мікроконтролера;
- Output open-drain - вихід з відкритим стоком. Аналогічний за своїм функціоналом виходу з відкритим коллектором;
- Alternate function open-drain – альтернативне виведення в режимі відкритого стоку.
- Alternate function push-pull - альтернативна виведення в активному режимі;

Для безпеки і попередженню помилок при налаштувані конфігураційних параметрів використовується спеціальна система захисту. Для її активізації потрібно послідовно виконати спеціальні операції над регістрами конфігурації. Розблокування даних регістрів можливе лише після скидання конфігурації порту.

Кожен представник серії STM32 F4 обладнаний трьома аналого-цифровими та двома цифро-аналоговими перетворювачювами. АЦП показують високу швидкість перетворення ( блиько 2,4 МСемпла в одиночному та 7,2 МСемпла – в потрійному режимі). Всього АЦП має 24 аналогових каналів.

Вбудовані ЦАП мають роздільну здатність в 12 біт. Перетворення проходять в 8/12-бітовому форматі. Також є можливість вирівнювати вихідні дані по правому або лівому краю. Оскільки ЦАП містить одразу 2 канали, то можна сформувати стереосигнал.

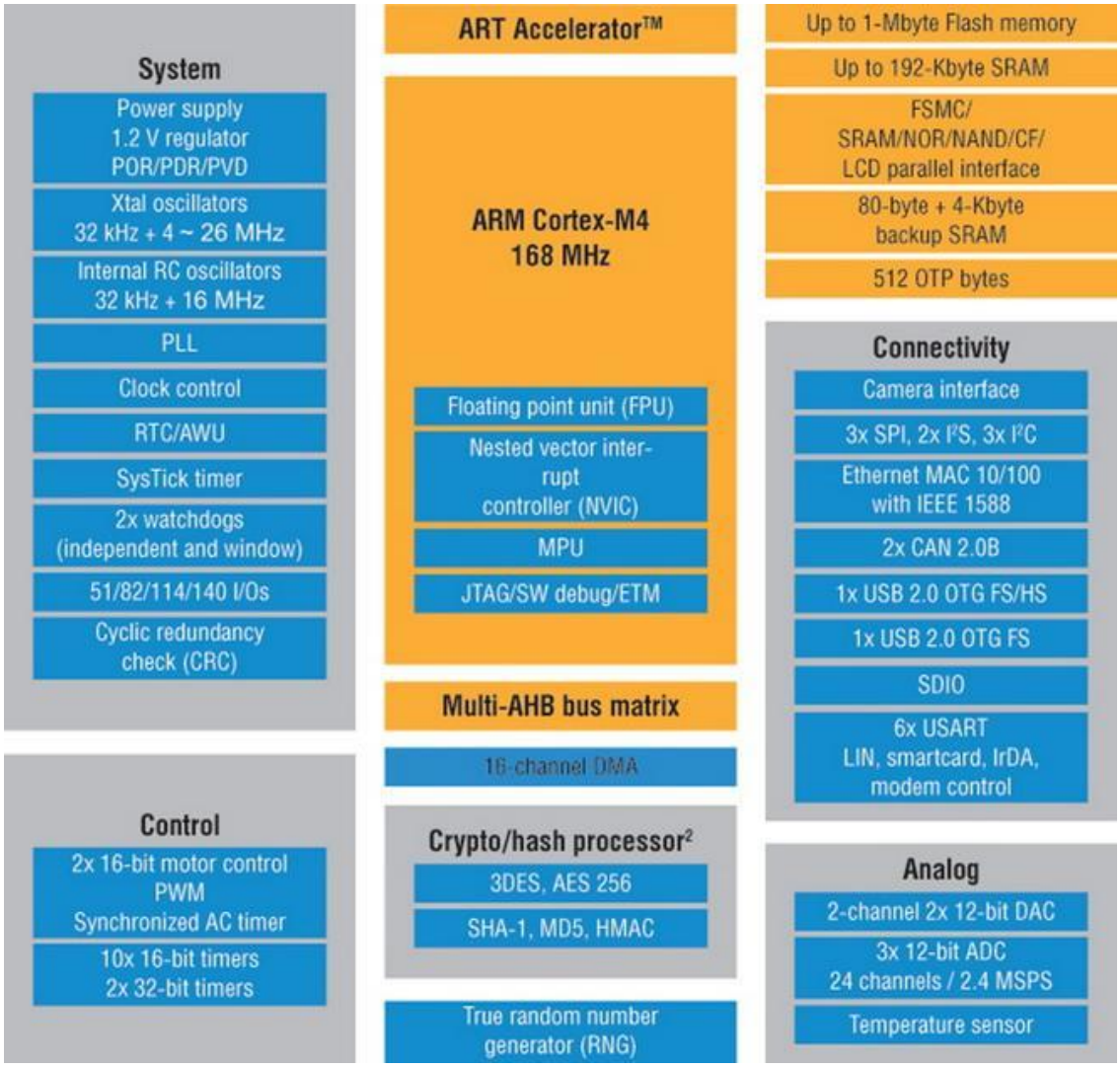


Рис. 2.4 Опис характеристик мікроконтролерів STM32F4 [26]

Окрім цього варто уваги потребують засоби відлагодження програм. Для прошиття програм на контролер він має в наявності 2 способи відлагодження: за допомогою чьотирьохпровідного JTAG-інтерфейсу, або з використанням двопровідного SWD. Виходи даних цих інтерфейсів мультиплексовані між собою, отже розробник може сам визначитись, яким інтерфейсом йому потрібно користуватись при програмуванні мікроконтролера. [26]

Після детального розгляду всієї лінійки мікроконтролерів оптимальним варіантом для вирішення поставлених задач було обрано контролер STM32F407VG.

Поміж представлених він є оптимальним з точки зору ціна/якість. Він оснащений вбудованою налагоджуючую платую з програматором-налагоджувачем ST-Link. Для програмування плати потрібно тільки підключити плату до комп'ютера з відповідним середовищем розробки з використанням спеціального кабелю і все. Окрім цього в плату вже вмонтована різноманітна периферія, а саме: акселерометр, microUSB порт, мікрофон та аудіокодек з роз'ємом для навушників. Так як і всі контролери STM32F4x7 в ньому розміщено 32-бітномт ядро RISC ARM®Cortex®-M4 з робочою частотою до 168 МГц. Ядро Cortex-M4 містить в собі арифметичний блок, для підтримки інструкцій для роботи з плаваючою точкою. Окрім цього він реалізує набір інструкцій DSP і блок захисту пам'яті (MPU), що значно підвищує рівень безпеки програми [27]. Повний опис апаратних можливостей цього контролера зображено на рис.2.5.

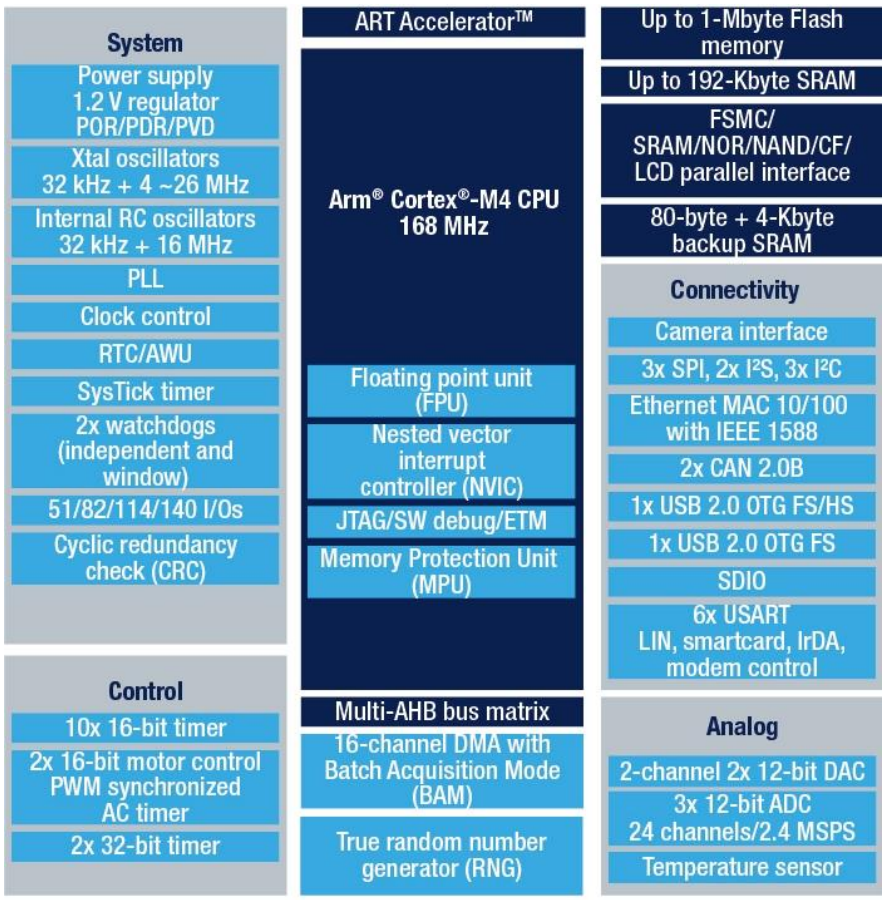


Рис. 2.5 Структура мікроконтролера STM32F407 [27]

Однією з переваг данного контролера при виборі є підтримка інтерфейсу Ethernet, який можна було б використовувати для обміну даними. Проте вирішальну роль зіграло те, що на цій платі вбудований акселерометр LIS3DSH, який вже було заплановано використовувати в розробці системи. Це дає можливість уникнути проблем зі сполученням датчика і мікроконтролера, оскільки це вже було зроблено розробником за нас. Таким чином економляться додаткові порти входів/виходів на платі, котрі можна буде використати при подальшій розробці.

Отже, в цьому підрозділі мною було розглянуто та обрано ті апаратні засоби, які будуть застосовані в ході виконання роботи. Функціонал даного забезпеченя повністю відповідає потребам, які були поставлені в ході постановки задачі.

## 2.2 Огляд програмного забезпечення, використаного в ході реалізації проекту

Обравши всі необхідні технічні засоби для проекту, потрібно провести огляд і обрати мову програмування, систему проектування та середовище розробки. Окрім того, необхідно провести огляд методів взаємодії периферійних пристроїв, а також інтерфейси обміну інформації між апаратними засобами. Окрім того потрібно провести огляд засобів для розробки та тренування нейронної мережі. Жоден проект на початку планування не обходиться без вибору таких надважливих елементів як:

- Мова розробки;
- Середовище розробки у відповідності до обраної мови програмування;
- Система проектування;
- Необхідні фреймворки, бібліотеки та ОС;

### 2.2.1 Огляд систем програмування та відлагодженя роботи мікроконтролерів

Для розробки проектів будь якої складності, які включають в себе використання мікроконтролерів необхідне обрати таке середовище розробки, яке змогло б задовільнити потреби в зручному написанні коду та подальшому прошиттю та відлагодженню його роботи на контролері. Для таких задач ідеально підходить середовище розробки Keil uVision. Даний продукт включає в себе набір спеціальних профільних утиліт для забезпечення потреб під час створення програмного забезпечення саме для мікроконтролерів.

Keil uVision надає розробнику широкий спектр можливостей, поєднаних в інтуїтивно зрозумілому інтерфейсі (рис.2.6). Основний набір функцій розміщений в контекстному меню, а рідко вживані функції сховані в додаткових меню для спрощення інтерфейсу, проте в кожному проекті можна використати потенціал усіх модулів об’єднаних в одній програмі.

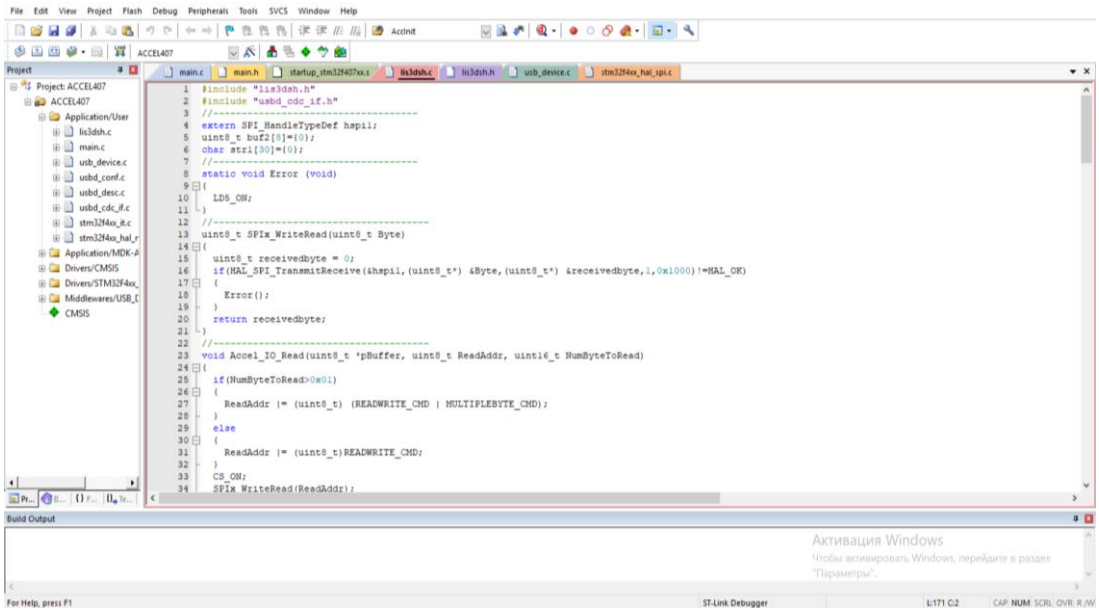


Рис. 2.6 Графічний інтерфейс Keil uVision

Головними елементами Keil uVision є:

1. База даних мікроконтролерів, в якій зберігається уся актуальна детальна інформація про всі доступні до програмування пристрої з допомогою цього програмного забепечення. В цій базі знаходяться конфігураційні налаштування а також посилання на офіційні джерела інформації з детальним технічним описом. Завдяки цьому сховищу даних середовище може автоматично налаштувати всі можливості мікроконтролера при включенні його до проекту.

2. Менеджер проектів – файлова система, необхідна для структурування та об'єднання програмних модулів та файлів в групи для забезпечення ієрархічності та цілісності проектів. Це надає можливість краще орієнтуватися поміж великого набору файлів, та контролювати порядок робочої директорії.

3. Засоби асемблювання, компіляції та збору проекту для створення виконуваного модуля програми. В цьому середовищі розробки доступна автоматична генерація асемблерних зв'язків, яка не перезбирає проект кожного разу з нуля, обробляє лише відредаговані файли, або такі файли, які залежать від змінених після останньої збірки. Також у Keil uVision присутня функція оптимізації вихідного коду програми, що дозволяє досягти оптимального використання потужностей мікроконтролера завдяки багаторазовій перекомпіляції програмного коду. Вбудований компілятор працює з кодом на мові С для сімейств контролерів ARM та інших.

4. Вбудований редактор, який допомагає спростити роботу з текстовими файлами. Перш за все це зумовлене наявністю кольорового підсвічення синтаксису, що покращує зручність читання файлів. Також для більшої зручності реалізовано можливість використання багатовіконного інтерфейсу

та налаштування візуального оформлення (кольори, шрифти, тощо). Також доступна функція редагування коду в процесі відлагодження програми.

5. Відлагоджувач-симулятор, який відповідає за налаштування скомпільованої програми на віртуальній моделі мікропроцесора перед її прошиттям на реальний аналог. Під час відлагодження віртуально моделюється робота ядра контролера, а також периферійних засобів, серед яких таймери, контролери переривань та порти вводу/виводу.

6. Додаткові утиліти, необхідні для спрощення повторюючихся задач. Серед таких функцій можна перерахувати:

- Source Browser – пошук функцій в системі;
- Find in Files – пошук символів в програмному коді;
- Tools Menu – меню стороннії утиліт;
- PC-Lint – аналізатор сирцевого тексту програми, який в реальному часі позначає ймовірні помилкові ділянки коду;
- Flash tool – утиліта, яка займається програмуванням та аналізом Flash-пам’яті мікроконтролерів. [28]

Окрім цього в даному середовищі розробки зручно реалізований процес відлагодження та компіляції програм. Keil uVision надає можливість відлагодження виконання програмного коду по рядках та перегляду вмісту внутрішніх регістрів в кожен момент часу, при під’єднанні плати до комп’ютера з ввімкненим середовищем. Це значно полегшує процес пошуку багів, тестування та налаштування мікроконтролера та перевірки коректності роботи на всіх етапах виконання програми.

*STM32CubeMX*

При програмуванні систем в які включаються мікроконтролерів найпершим, а іноді і найскладнішим є приведення плати в робочий стан. І найпершим для цього

є налаштування конфігурації та встановити відповідні значення в програмуючі регістри. До того моменту, коли ви зможете запустити будь який код на платі потрібно програмно вказати робочу тактову частоту та активувати роботу інтерфейсів, визначити режим роботи портів вводу/виводу та виконати набір інших налаштувань. І ця процедура може категорично відрізнятися від одного до іншого мікроконтролера різних виробників, а в деяких випадках і різних моделей одного виробника. Задля того, щоб розробник виконував свої прямі обов'язки, а не витрачав час на виконання рутинних інтрукцій по налаштуванню контролера, а також для повної впевненості в правильному налаштуванні обраного контролера компанія-розробник контролерів STM32 STMicroelectronics створила власне програмне забезпечення, яке спеціалізується на налаштуванні пристроїв власного виробництва під назвою STM32CubeMX.

STM32CubeMX це інструмент для автоматичного конфігурування пристроїв STM32 з графічним інтерфейсом, а також для генерації вихідного коду мовою C для ядра Arm Cortex-M або дерева пристрою Linux на ядрі Arm Cortex. Принцип роботи даної програми проста і інтуїтивно зрозумілий вже після першого її використання. В STM32CubeMX вбудовано база даних, яка містить в собі актуальні алгоритми налаштування для всіх мікроконтролерів STM32. Послідовність роботи з цим застосунком така. Перш за все обирається мікропроцесор, який ви плануєте використовувати у відповідності до бажаних периферійних пристроїв. Вибір мікроконтролера відбувається серед усіх існуючих пристроїв компанії STMicroelectronics. В цьому списку можна переглянути деталі та характеристики, а у разі необхідності і посилання на офіційну документацію з більш глибоким розглядом контролера. Після підтвердження вибору на вашу машину будуть завантажені драйвери для роботи саме з даною апаратурою. З цього моменту з'являється можливість користувацького налаштування. Все налаштування відбувається в графічному інтерфейсі. В робочій зоні програми буде відображена схема всіх входів/виходів програми, які можна налаштовувати на різні режими в

інтерактивному вигляді. Зміни можна одразу чітко побачити. Елементи можуть змінювати колір в залежності від налаштування, що можна чітко побачити на рис.2.7.


Рис. 2.7 Робоче поле програми STM32CubeMX

Окрім налаштування портів вводу виводу доступні також функції тактування всієї системи та периферії, налаштування вбудованого програмного забезпечення, серед якого: дерево тактування та калькулятор енергозатрат, а окрім цього застосунки для налаштування стеків проміжного програмного забезпечення, як інтерфейси Ethernet, USB та TCP/IP. Усі ці задачі та багато інших функцій можна виконати завдяки графічному інтерфейсу. В результаті цього, після завершення налаштування можна провести генерацію коду на мові С який відтворить на платі обрану конфігурацію з відповідними характеристиками та ініціалізованим ядром Arm Cortex, готовим до використання в середовищі розробки.

Головні особливості програми:

-Засоби для вибору мікроконтролера в графічному інтерфейсі;

-Зручний графічний інтерфейс користувача;

-Налаштування портів вводу/виводу з автоматичним вирішенням конфліктів;

-Режими налаштування програмного забезпечення та периферії з динамічною валідацією параметрів ядра;

-Дерево тактування з динамічним переконфігуруванням (рис. 2.8);

-Схеми прогнозування результатів енергоспоживання;

-Генерація коду ініціалізації проекту, включно з компіляторами для плат STM32;

-Створення дерева пристроїв Linux для ядра Arm Cortex-A;

-Автономне програмне забезпечення, що працює майже на всіх відомих операційних системах або з допомогою плагіну Eclipse. [29]

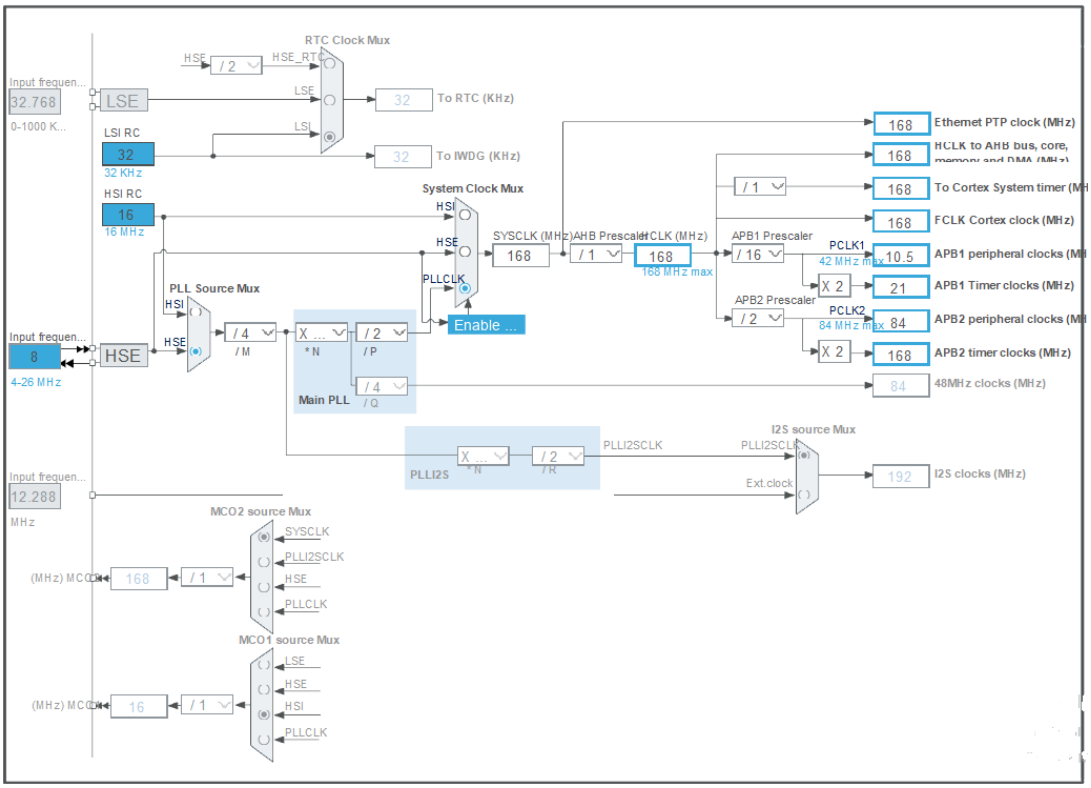


Рис. 2.8 Приклад дерева тактування в STM32CubeMX

*STM32CubeF4 FirmWare*

Для кожної серії мікроконтролерів компанія STMicroelectronics створює набір бібліотек для роботи саме з цією апаратурою. Лінійка STM32F4x7xx не виключення. Для цієї серії було створено STM32Cube FirmWare F4. Саме в цій структурі представлені всі необхідні для використання бібліотеки, в залежності від апаратного забезпечення мікросхеми з початковою назвою STM32F4.

Архітектура цього набору має три рівня реалізації, в залежності від рівня абстракції як це зображено на рис.2.9.

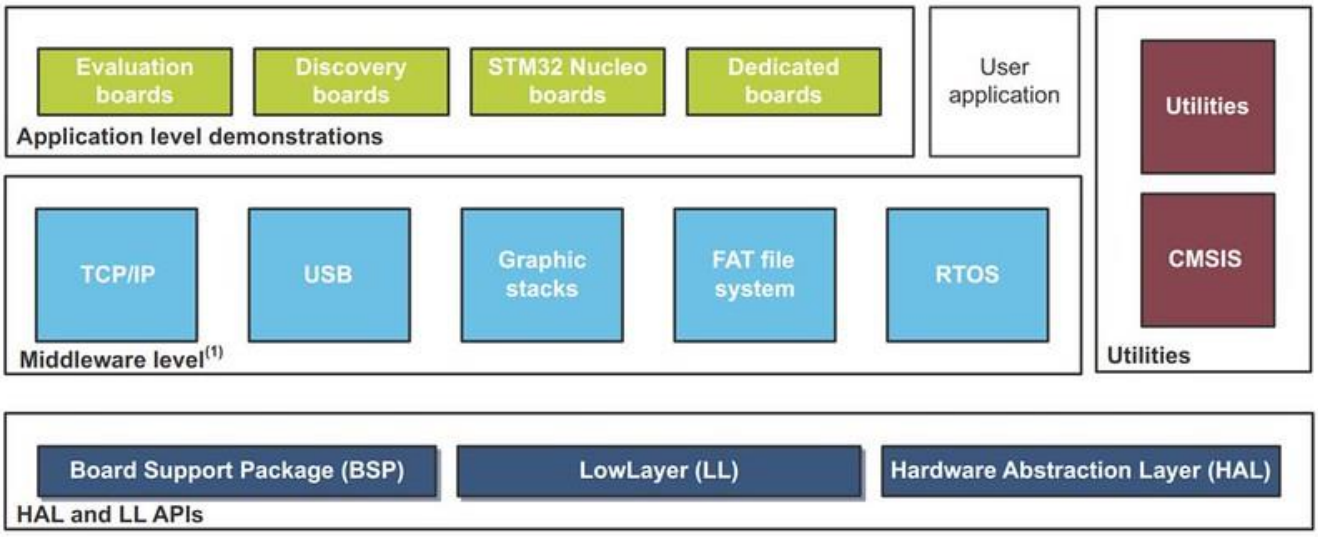


Рис. 2.9 Архітектура STM32CubeF4_FirmWare [30]

*Нульовий рівень*

Найнижчим рівнем абстракції є нульовий. На цьому рівні розміщений Пакет Підтримки Платформи, низький рівень, і рівень Апаратних абстракцій. В Пакеті Підтримки Платформи реалізовані програмні інтерфейси для апаратних компонентів на платах, такі як драйвери для акселерометра, аудіокодека, портів вводу/виводу або інших периферіних пристроїв. Також цьому пакеті реалізовані

драйвери для роботи з будь якими зовнішніми пристроями, навіть тими, які не мають відношення до компанії STMicroelectronics. Окрім цього в пакеті реалізований драйвер BSP, для роботи з пристроями, які розміщені на самій платі. В пакеті BSP наявна велика кількість простих інтерфейсів для роботи з периферією вмонтованою в плату. За допомогою BSP можна виконати безліч операцій однією командою. Наприклад, можна запалити LED лампочку за допомогою простої команди BSP_LED_On ().

Рівень Апаратних Абстракцій (HAL) містить в собі драйвери низького рівня і методів апаратних інтерфейсів, для взаємодії елементами вищих рівнів (бібліотеки, додатки, стеки). Драйвери рівня HAL це універсальний інтрументарій для керування низькими рівнями мікроконтролера на програмному рівні. HAL складається з багатофункціональних загальних, або розширених інтерфейсів API.

HAL необхідний при програмуванні контролерів, коли на нього передаються високорівневі команди, а їх виконання відбувається верхнім шаром додатка, за допомогою проміжного рівня HAL. При такій архітектурі інтерфейсів верхній рівень програми ніяким чином не прив'язаний до конкретного мікроконтролера, так як не надсилає команди напряму, а лише відправляє вказівки бібліотеці драйверів. Використання такої архітектури значно абстрактизує низькорівневі задачі на різних контролерах і надає можливість перевикористання програмного коду в багатьох проектах і спрощує перехід з одного пристрою з архітектурою STM32 на інший.

Рівень драйверів HAL надають набір універсальних API, які спрощують роботу з мікроконтролером для розробника програми. Наприклад, в бібліотеці HAL присутній широкий вибір методів для роботи з інтерфейсами комунікації. В ній містяться інтерфейси API для ініціалізації і конфігурування мікроконтролера, керування обміном інформацією методом опитування, переривання або через DMA, а також управління помилками зв'язку.

Інтерфейси драйверів HAL поділяються на дві категорії: інтерфейси загального призначення та розширені API. Загальні (generic) API містять в собі спільні для всіх пристроїв серії STM32 функції та інтерфейси. Розширені (extension) API в свою чергу, складаються зі специфічних елементів та функцій, для реалізації специфічного функціоналу для додаткових пристроїв. Драйвери бібліотеки HAL функціонально-орієнтовані. Вони не направлені на реалізацію роботи внутрішніх периферійних пристроїв. Так наприклад, API таймера поділяється на декілька категорій, в залежності від тої функціональності, як вбудована у внутрішньому пристрої таймера, тобто: базовий таймер, режим захоплення, або широтно-імпульсні модуляції (PWM).

Вихідний код бібліотеки драйверів HAL розроблений у відповідності до Strict ANSI-C. Це значить, що цей код являється платформо незалежним і зрозумілим для всіх середовищ розробки. Весь вихідний код перевірений за допомогою інструменту статистичного аналізу CodeSonarTM. Весь код добре документований, інтуїтивно зрозумілий ісумісний до стандарту MISRA-C 2004.

Окрім цього драйвери HAL підвищують рівень надійності для вбудованого програмного коду, оскільки в бібліотеках HAL реалізований механізм виявлення помилок під час роботи (run-time failure detection). Під час роботи HAL перевіряє отримані результати власних функцій. Окрім цього, динамічний пошук помилок під час виконання коду, пришвидшує час розробки і процесу відлагодження.

Останнім на цьому рівні лишився розгляд низького рівня (LL). На цьому шарі реалізовані API прикладного рівня на рівні регістрів. Це добре для оптимізації, жертвуючи портативністю. Використання функцій цього рівня потребують від розробника глибокого розуміння того, як влаштований мікропроцесор та периферійні пристроїв на платі. Драйвери LL можна назвати експертно-орієнтованим шаром, який знаходиться на найближчому рівні до апаратного

забезпечення. На відміну від HAL, LL драйвери не націлені на периферійні пристрої, для яких важлива складна конфігурація або стеки верхнього рівня.

*Перший рівень*

На першому рівні абстракції знаходяться компоненти проміжного рівня або Middleware components. На цьому рівні розміщені бібліотеки для комунікаційних систем та інтерфейсів. Представниками цього рівня є бібліотеки які охоплюють такі технології як FreeRTOS, FatFs, USB Host, STemWin, LibJPEG, LwIP і PolarSSL. Взаємодія між елементами цього рівня здійснюються напряму, шляхом виклику API. Робота з елементами нульового рівня здійснюється за допомогою зворотніх викликів.

*Другий рівень*

Цей рівень містить в собі один шар, на якому реалізовано графічне відображення в реальному часі першого рівня та нульового рівня абстракції і додатків, які застосовують периферійні пристрої для функцій підтримуваних відлагоджувальним набором. [30]

## 2.3 Використання мови С для програмування периферійних пристроїв

Програмні та апаратні засоби, які використовуються в проекті передбачають використання мови С як мови розробки програмного забезпечення системи. Мова С є однією з найчастіше використовуваних мов програмування драверів та програмного забезпечення для мікроконтролерів. На сьогоднішній день, у зв’язку з

появою нових мов програмування високого рівня С втрачає свої позиції в багатьох сферах, однак вона все ще залишається незмінною для системного програмування. Застосування мови С досить широке. Не зважаючи на малу мовну базу і урізаний функціонал, у порівнянні з іншими, дана мова може бути застосована для програмування як низькорівневого так і високорівневого програмного забезпечення і є універсальною, на відміну від більш спеціалізованих мов. Мова С розроблялася як мова системного програмування. В планах, дана мова мала використовуватися при створенні UNIX компіляторів. Як було сказано раніше, мова С не відрізняється великою функціональністю, проте її простота значить, що компілятори цієї мови пишуться легше ніж для інших, тому дана мова підтримується на безлічі платформ. Також С володіє високим рівнем переносимості, не зважаючи на те, першочергово вона є низькорівневою, тому програми написані на мові С легко компілюються на машинах з різними архітектурами.

Основною задачею мови програмування С було облегшити написання програм великого обсягу при зменшенні кількості невимушених помилок порівняно з асемблером. Мова С підтримує основні принципи функціонального програмування, але парадигма програмування на С полягає в тому, щоб максимально уникати додаткових витрат пам’яті і потужностей машини, порівняно з мовами високого рівня.

Особливості мови С:

- Невелика мовна база;
- Багато функцій винесено в стандартні бібліотеки;
-Підтримка парадигми функціонального програмування;
-Система типів;
-Передобробка для прискорення роботи при виконанні однотипних операцій;
-Засоби прямого доступу до пам’яті;

-Передача значень а не посилань в якості параметрів функції;

-Передача параметрів функції за допомогою вказівників;

-Вказівники на функції та статичні зміни;

-Області видимості імен;

-Структури та об’єднання – збірні типи даних створені користувачем, котрими можна маніпулювати. [31]

Проте є і речі, які відсутні в С на відміну від більшості мов програмування, такі як:

- Вкладені функції;
- Множинне повернення значення з однієї функції;
- Співпрограми;
- Автоматизоване управління пам’яттю;
- Засоби об’єктно-орієнтованого програмування;
- Елементи функціонального програмування.

Вказані недоліки так чи інакше можна вичеркнути, так як багато відсутніх речей частково, або повністю замінюються вбудованими засобами або додаються завдяки стороннім бібліотекам. Варто зазначити, що наявні деякі компілятори, в яких включені розширені можливості мови, такі, як вкладені функції в компіляторі GCC. В різних компіляторах додано різний додатковий функціонал, такий як використання співпрограм, багатопоточне програмування, вбудований збирач сміття, мережеві функції. Також представлена методика, яка дає можливість реалізувати механізм ООП на мові С, що побудований на фактичному поліморфізмі вказівників в мові С та підтримці вказівників на функції.

Мова С користується попитом серед розробників, не в останню чергу тому, що вона зручна для написання компіляторів для різноманітних платформ, і в зв’язку зі своєю наближеністю до мов низького рівня надає наглядність послідовності виконання програми на машині. Окрім цього мова С славиться своєю оптимізацією.

За рівнем швидкодії код написаний на С програє лише коду, написаному на Асемблері, так як той напряму відправляє команди до процесора. В звязку з тим, що розвиток компіляторів та напівпровідників рухається надзвичайно швидко, то написання програм на Асемблерів знецінюються і їх виграш над згенерованим кодом компілятора мови С мізерний. А якщо враховувати складність написання коду на Асемблері, то можна сказати, що мова С значно виграє в часі повного циклу розробки. На сьогоднішній день мова С продовжує утримувати статус одної з найбільш продуктивних мов високого рівня.

За структурою коду, програми на мові С зазвичай написані в одному стилі. Проект на мові С представляє з себе набір файлів, які згодом будуть зкомпільовані в об’єктні файли. Потім утворені файли об’єднуються поміж собою а також з файлами зовнішніх бібліотек в один виконуваний файл, або бібліотеку. Для розуміння компілятора про взаємозв’язки файлів між собою а також бібліотеками потрібно явно вказувати опис прототипів функцій, зовнішніх змінних та необхідних типів даних у кожному документі. Для таких задач в С вирішено виносити такі дані в окремі файли, які називають заголовковими. Для підключення вмісту заголовкового файлу в основний код програми потрібно додати директиву #include на початок файлу. Завдяки цьому можна організовувати систему модулів, де в якості модуля може виступати:

- файл з вихідним кодом, для якого створено інтерфейс, у вигляді header файлу;
- об'єктна бібліотека з заголовковими файлами;
- інтерфейсна бібліотека – набір заголовкових файлів;
- статична бібліотека або модулі з її заголовковими файлами;
- динамічна бібліотека або модулі з її заголовковими файлами.

На етапі передобробки директива #include вказує вставити текст іншого файлу на місце де вона викликана. Після рекурсивного виконання цієї директиви

весь код з всіх модулів збирається в одному виконуваному файлі. Повторний виклик директиви вважається зайвим, так як це може призвести до неочікуваних помилок під час компіляції. [32]

Мова програмування С часто використовується в розробці програм прикладного рівня, у вбудованих системах, кросплатформних програм, і для написання високопродуктивних програм із високою стійкістю до помилок. Програми написані на С можна виконати на безлічі різних пристроїв з різними операційнии системами. І контролери компанії STMicroelectronics не виключення. Зсилаючись на все вищевказане мова С обрана як оптимальна для опису апаратного забезпечення для плат STM32.

## Висновки до розділу 2

В цьому розділі магістрської дисертації було зроблено детальний аналіз всіх елементів системи, необхідних для її реалізації, а саме:

1. Розгляд мікроконтролерів сімейства STM32, та вибір необхідних периферійних пристроїв;
2. Розгляд програмного забезпечення з для опису конфігурації та програмування обраного апаратного забезпечення (мова програмування, середовище розробки та методи налагодження та конфігурування програми на платі);
3. Прийнято рішення обрати мікроконтролер STM32F407vg в якості апаратного забезпечення, в зв’язку тим, що в ньому вже вмонтований цифровий акселерометр. Також мікроконтролери STM32 показують високі показники ефективності при низьких енерговитратах. Розробку вирішено виконувати в середовищі Keil uVision мовою C, в зв’язку із легкістю та наглядністю роботи та відлагодження написаних програм на мікроконтролері.
4. Зібрано всю достатньо інформації для реалізації системи.

# РОЗДІЛ 3
# РОЗРОБКА ТА ОБГРУНТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

## 3.1 Визначення структури системи

В результаті огляду технологій та методів для побудови системи було зібрано всю необхідну інформацію для виконання задач поставлених на цю магістрську дисертацію.

Головна ціль цього проекту є реалізація програмного і апаратного забезпечення, в якості частини системи Інтернету речей, яке можна в подальшому інтегрувати в більш масштабний проект з мінімальними потребами в переналаштуванні. Причиною такого рішення слугує різке зростання популярності створення систем SmartCity. Найчастіше, такі системи являються компіляцією багатьох проектів різних напрямків з різними датчиками, отже побудова системи, яку легко інтегрувати в щось масштабніше приможливості самостійної роботи є оптимальним рішенням. Вибір методів та технічного забезпечення пояснюється в тому числі і вище вказаними фактами.

Структурна схема апаратної частини проекту представлена на рисунку 3.1. В першу чергу варто зазначит, що розроблена апаратна частина системи буде базуватися на мікроконтролері STM32, а саме STM32F407vg. Це пояснюється тим, що техніка цієї компанії по праву займає лідерскі позиції по поширеності серед мікроелектроніки в усьому світі. Мікроконтролери, датчики та периферійні пристрої компанії STM можна знайти майже в кожному проекті пов'язаному з Інтернетом речей та апаратурою в цілому. Окрім того, як вже вказувалось вище, великим плюсом є легкість в налаштуванні та переконфігуруванні коду для цих мікроконтролерів. Код написаний для цієї архітектури сумісний з багатьма контроллерами навіть при їх великій відмінності одне від одного. Не останнє місце

в виборі займало зручне середовище розробки, створене компанією розробником STMicroelectronics, що значно економить чам на реалізацію проектів і мінімізує їх рутинну складову.

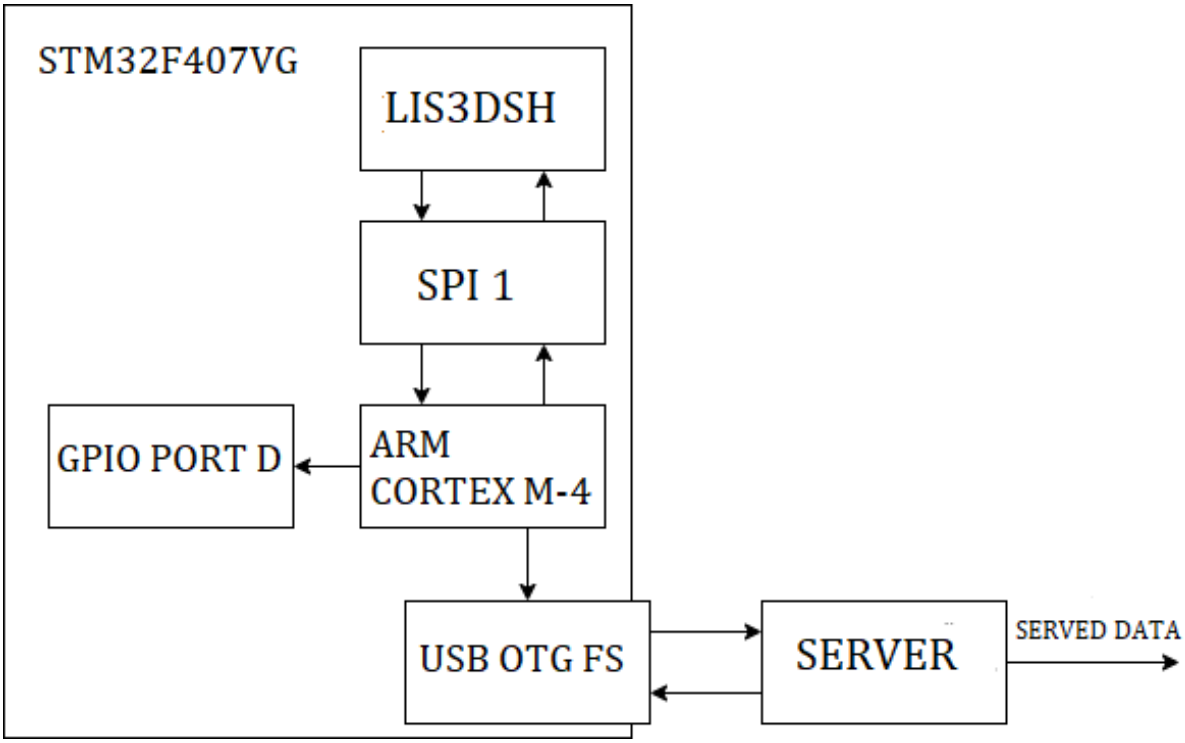


Рис. 3.1 Апаратна архітектура пристрою

Проте у відкритих джерелах наразі мало інформації про розробки пристроїв та інформаційних систем на їх базі, які можна було б інтегрувати в масштабний проект. Тому робота над проектом розділена на етапи. Перш за все необхідно написати драйвери для мікроконтролера для збору данних з акселерометра. Окрім прямих задач, дана частина роботи розглядається як елемент для ознайомлення новачків з роботою з периферійними пристроями на базі мікроконтролерів STM.

Вибір плати STM32F407vg пояснюється тим, що в неї вживлені всі необхідні елементи, котрі визначені для рішення поставленої задачі. Вона відзначається низькою енергозатратністю, при наявності гарних показників продуктивності.

Також цей контролер містить в собі декілька інтерфейсів для обміну данних, на ньому розміщено велику кількість портів вводу/виводу загального користування. Але за відсутності цієї плати її легко замінити іншою платою STM32 при умові перепризначення портів, налаштування тактування та перепрошитті самих драйверів мікроконтролера.

Для цього проекту використано датчик прискорення LIS3DSH. Цей вибір зумовлений тим, що він вбудований в плату, котру було визначено раніше, але це не є найбільшою перевагою. Перш за все він вузьконаправлений, тому не витрачає потужності на ті речі, котрі не потрібні в самому проекті, що надзвичайно важливо для мікропроцесорних пристроїв. Він відрізняється високою чутливістю і легко конфігуруються на різні діапазони граничних значень в залежності від потреби. Також важливим є те, що цей датчик комутуватується з системою завдяки інтерфейсу SPI, який є оптимальним для систем Інтернету речей в звязку з можливістю обміну інформації між багатьма периферійними пристроями.

Для спілкування з датчиком було обрано використання інтерфейсу SPI, як одного з найпоширеніших, для отримання даних з периферії. Цей вибір аргументується спрощенням роботи з данними, в звязку зі швидким синхронним зв’язком, та можливістю одночасного послідовного зв'язку між декількома датчиками і одним мікроконтролером. Для виводу інформації було використано порт micro USB вбудований в платі STM32. Це спростить збір данних для створення датасету. Данні будуть считані комп’ютером за допомогою COM-порту.

Весь етап розробки драйверів для апаратної частини було виконано в в середовищі розробки Keil uVision мовою програмування C. Цей вибір аргументовано зручністю роботи та внутрішньою оптимізацією зкомпільованого асемблерного коду, що гарантує високу швидкодію і мінімальні затримки при роботі драйверів. Конфігурування контролеру виконано за допомогою програми

STM32CubeMx, яка дає можливість автоматичного налаштування портів, бібліотек та інтерфейсів плати з генерацією подальшого коду.

Наступним етапом є збір датасету та написання нейронної мережі для класифікації якості дорожнього покриття. На цьому етапі було зібрано інформацію з послідовного COM порту і збережено у вигляді файлу csv. Далі на базі отриманого файлу, інформація була поділена на тестову та тренувальну вибірку. Потім було створено нейронну мережу для визначення ям на дорогах на базі отриманого датасету. Для розробки використовувалась мова програмування Python та бібліотека keras.

Кінцевим елементом слугує карта, на якій відображаються якість дорожнього покриття на ділянках на яких було виконано проїзд автомобілем з пристроєм на базі класифікації натренованої нейронної мережі. Окрім того для наглядності створена графічна програма, для зчитування інформації з COM-порту, виводу її в зручному графічному вигляді. Дана програма реалізована на мові програмування Python з виконристанням бібліотеки для роботи COM-портом pySerial та бібліотекою для графічного представлення даних mathplotlib.

### 3.2 Опис роботи системи як складової частини проекту Smart City

В цьому підрозділі магістерської дисертації буде описана уся бізнес логіка та опис запланованої роботи системи. Вся система поділяється на дві частини: пристрій та сервер. Наразі відсутня можливість повної автономності пристрою для фіксації ям, тому в проекті замість віддаленого сервера було використано звичайний ноутбук. Для зв’язку між частинами програми використовується обмін данними по USB to microUSB шнуру, приєднаному до портів USB на комп’ютері та micro USB на платі.

Для початку, при запуску системи важливо провести перевірку її працездатності перш ніж прийти в робочий стан. Для цього проводиться опитування всіх елементів системи. Перш за все, відбувається перевірка керуючих бітів та робочий стан відповідних портів з периферійними пристроями. У випадку, якщо результат перевірки позитивний, відбувається налаштування периферійних пристроїв, ініціалізується шина SPI та  перехід на наступний етап перевірки. Далі відбувається перевірка зв’язку між сервером та пристроєм для збору показників лінійного прискорення, шляхом опитування. У випадку не критичних помилок відбувається рестарт, а при критичних помилках накшталт несправності робочих портів пристрою робота системи припиняється. На пристрої запалюється червоний ліхтарик, який сиволізує несправність і необхідність перевірки роботи системи.

Для правильності роботи системи, надзвичайно важливе правильне положення пристрою. Він має бути прикріплений до авто так, щоб датчик прискорення був розміщений паралельно до дорожньої смуги. Для регулювання правильного положеня потрібно подивитися, чи не горить жоден з ліхтариків біля датчику. У випадку, якщо один з них, або декілька палають, то це означає, що пристрій потрібно відхилити в протилежну сторону від ліхтарика. Правильним вважається положення, коли усі чотири ліхтарики вимкнені. Тоді показники отримані з датчика можна вважати релевантними.

Після завершення всіх перевірок можна починати збір даних з датчика лінійного прискорення. Отримані з акселерометра показники приходять на контролер по шині SPI і відправляються на порт micro USB в режимі послідовного COM порту.

При отриманні, данні аналізуються за допомогою класифікатора натренованого на зібраних раніше даних. Далі визначається положення за допомогою GPS та встановлюється відповідна позначка на Google карті. Позначки бувають трьох типів: хороша дорога – зеленого кольору, незначні пошкодження –

жовтого кольору, та значні пошкодження – червоного кольору. Для візуального представлення отриманих показників з пристрою було також створено програму Pothole Monitor з графічним інтерфейсом. При її запуску, на моніторі з’являється вікно з графіком отриманих значень. В даному вікні можна змінити USB порт, який читається. Для роботи програми необхідно, щоб пристрій був приєднаний до комп’ютера USB шнуром. Після вибору необхідного порту і та натискання кнопки “Start” данні починають зчитуватися з USB порту і відображатися у вигляді графіку у робочому вікні програми.

Дані на COM порт приходять у шістнацядковому форматі. Після цього вони перетворюються в десяткові і переводяться в одиниці вимірювання g, рівні 9.8 м/с$^2$. Після цього данні зберігаються у вигляді кадрів послідовних вимірів акселерометра. Далі отримані кадри класифікуються натренованою нейронною мережею на предмет наявності ям та відображати цю інформацію в графічному вигляді на ряду з динамічним графіком зміни вертикального прискорення, який є головним параметром для класифікації. Окрім того на робочому тлі програми потрібно розмістити кнопку “Pause”, для зупинки читання послідовного порту. Таким чином можна буде надіслати команду до пристрою для повної зупинки роботи пристрою, і зупинити графічне відображення.

Після створення даного набору програмного забезпечення задачі та цілі на дану роботу можна буде вважати виконаними.

## Висновки до розділу 3

В третьому розділі магістерської дисертації було виконано планування та підготовчу роботу для виконання поставлених практичних задач, а саме:

1. Обґрунтовано вибір використання тих чи інших програмних та апаратних засобів в розробці;
2. Запропоновано архітектуру системи та її загальний принцип її роботи;
3. Визначено способи обробки та інтерпритації отриманих данних;
4. Представлено опис роботи системи;
5. Обрано функції до імплементації, необхідні в ракмках створення проекту, та методи їх реалізації.

З огляду на усе вище сказане, можна стверджувати, що завершено підготовчий етап проекту. Інформації, отриманої в ході трьох розділів цілком достатньо, для того, щоб починати реальну розробку системи.

# РОЗДІЛ 4
# РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для початку потрібно чітко вказати, що розробка проекту повністю поділяється на окремі етапи:

1. Розробку пристрою для збору данних
2. Збір датасету та створення нейронної мережі для виявлення ям
3. Створення серверу для візуалізації роботи системи
4. Створення зв’язку між пристроєм та сервером
5. Реалізація додаткового програмного забезпечення, для перевірки роботи системи та її візуалізації

## 4.1 Розробка пристрою для збору данних

Розпочати розробку системи вирішено з  побудови пристрою для збору данних про лінійне прискорення. А саме створення програмного забезпечення для налагодження мікроконтролера STM32F407vg, з вбудованим в нього акселерометром LIS3DSH та портом micro USB.

### Налаштування плати STM32F407vg

Спочатку необхідно налаштувати контролер на правильну роботу, тобто підключити необхідні інтерфейси та бібліотеки, встановити робочі порти в правильні режими роботи, провести тактування плати та багато іншого. Для реалізаціїї цих задач скористаємось описаним раніше програмним забезпеченням STM32CubeMX.

Відкривши програму, спочатку обирається робоча директорія, місце де буде збережено весь вміст проекту на диску та переходимо до наступного налаштування. Далі потрібно обрати мікроконтролер, котрий планується використовувати. В цьому випадку це STM32F407VGTx. Вікно для вибору контролера зображене на рис.4.1.

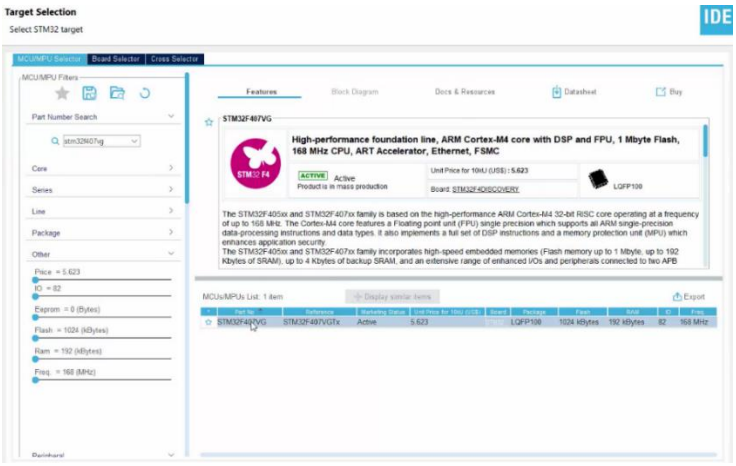


Рис. 4.1 Вікно вибору мікроконтролера

Наступним кроком є визначення назви проекту та мови програмування для цього проекту. Після цього знову натискаємо на кнопку Далі, і після цього нарешті відкривається головне вікно програми, яке зображене на рис.4.2.

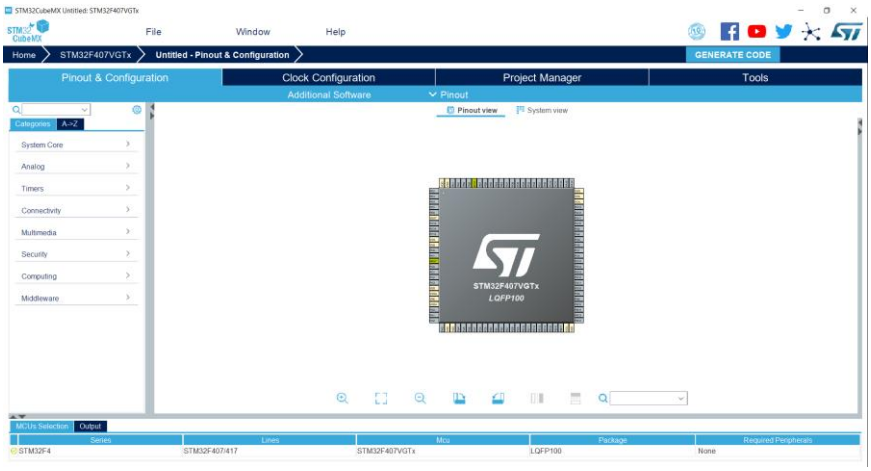


Рис. 4.2 Головне вікно програми STM32CubeMX

Наступним кроком є налаштування плати для роботи з акселерометром та портом micro USB. Спочатку налаштуємо тактову частоту мікроконтролера. У розділі налаштувань SystemCore/RCC переводимо High Speed Clock на режим Crystal Ceramic Resonator. Після цього переходимо на вкладку Clock Configuration і виставляємо налаштування у відповідності з рис. 4.3.

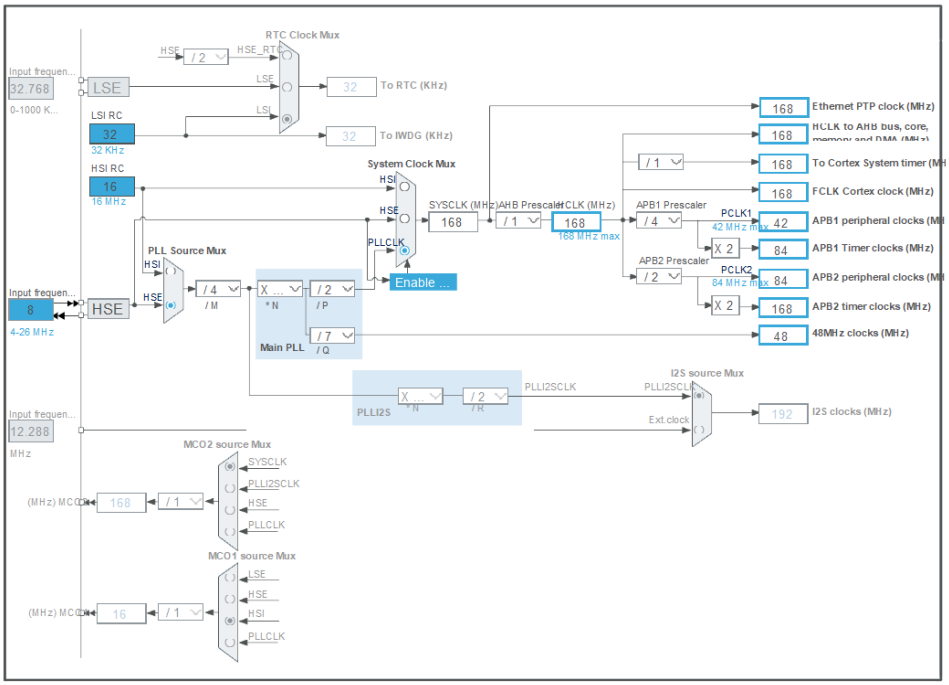


Рис. 4.3 Дерево тактування мікроконтролера

Після цього потрібно налаштувати інтерфейси комунікації, котрі влануються використовувати, тобто SPI та USB. В розділі Pinout & Configuration у вкладці Connectivity необхідно конфігурувати інтерфейси SPI та USB_OTG_FS. Інтерфейс SPI в даному пристрої відповідатиме за зчитування мікроконтролером даних з акселерометра. Налаштуємо його на режим Full-Duplex Master, з дільником Prescaler зі значенням 16 для зниження частоти считування данних на рівень, допустимий датчиком лінійного прискорення. Наступним налаштуємо інтерфейс

USB. Визначимо його режим як Device_Only. Далі в розділі Middleware в для USB встановлюємо Class For FS IP в режим Communication Device Class (Віртуальний COM порт). Після всіх цих дій можна вважати, що налаштування комунікаційних інтерфейсів завершено.

Далі нам залишилось лише активувати порти вводу/виводу. Для розуміння, які саме порти варто активувати переглянемо офіційну документацію мікроконтролера [33]. На схемі розміщеній в документації (рис.4.4) чітко вказані необхідні нам порти для взаємодії з акселерометром, а саме PA5-PA7, PE0, PE1 і PE3. Окрім того, активуємо відповідні порти для LED ліхтарів, вмонтованих в плату. Як результат, можемо побачити схему ввімкнених портів на рис. 4.5.

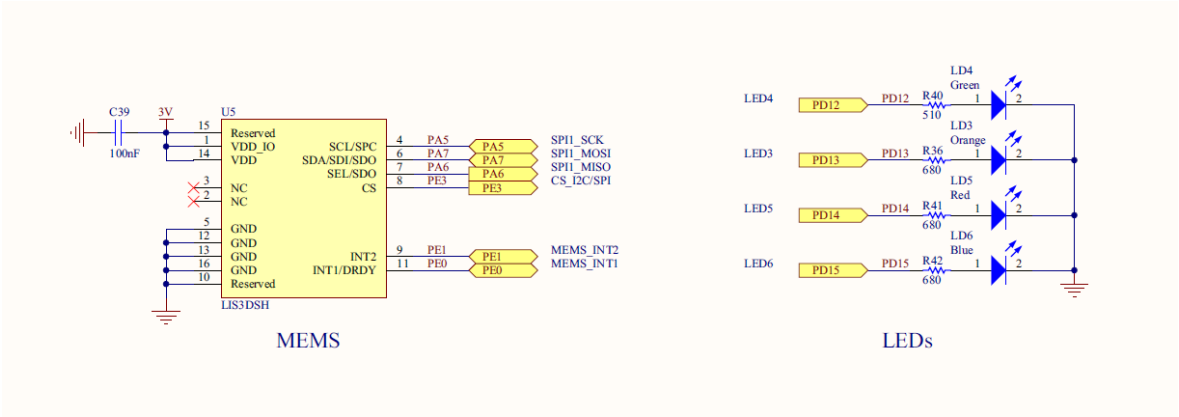


Рис. 4.4 Схема портів вводу/виводу для датчика LIS3DSH та LED ліхтарів [33]

Наразі конфігурування завершено і можна переходити до написання коду. Залишилося лише зайти в розділ Project Manager, обрати MDK-ARM як генератор коду і натиснути на кнопку Generate Code. Після цього буде автоматично створено проект в Keil uVision з уже згенерованим кодом визначених конфігурацій.

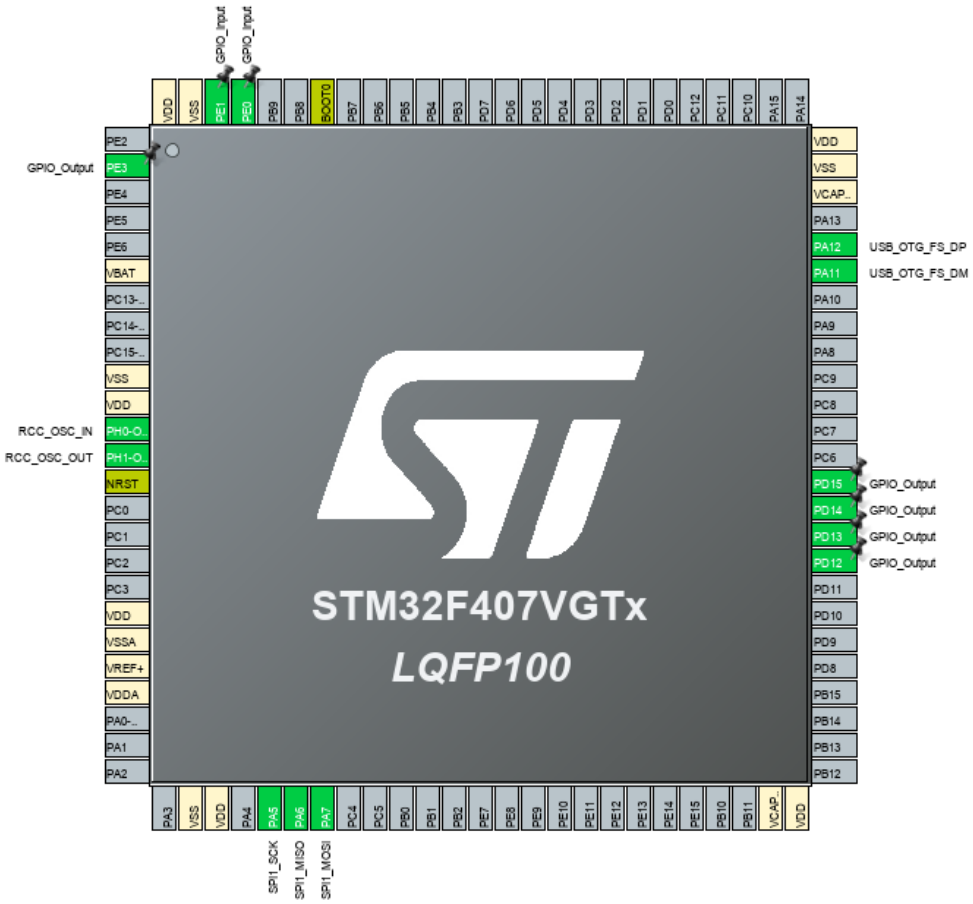


Рис. 4.5 Схема підключених портів вводу/виводу

## Опис програмного забезпечення для роботи пристрою

Після генерації початкових налаштувань мікроконтролера в нашому розпорядженні стають такі згенеровані модулі, як:

- main.c –виконавчий файл проекту, який представляє з себе робочий блок програми. В ньому описана функція main(), яка запускається на виконання при старті проекту. Також в цьому файлі розміщені функції для ініціалізації загальних портів вводу виводу, інтерфейсу SPI, налаштувань системного тактування та функції контролю помилок;
- stm32f4xx_it.c – файл, який відповідає за налаштування переривань;

- usbd_cdc_if.c – файл, відповідальний за роботу з COM портом для USB, в якому містяться методи пересилки даних;
- пакет Drivers – папка з необхідними драйверами стандартних бібліотек CMSIS та HAL.

Завдяки тому, що в проекті вже згенеровані методи для пересилки данних через послідовний порт, то одразу можна створити функцію для обміну данними. В файлі usbd_cdc_if.c вже описаний такий метод і він цілком задовольняє наші потреби, отже можна використовувати його без внесення змін.

Весь функціонал пов’язаний з операціями над акселерометром винесемо в окремий файл lis3dsh.c а всі необхідні для нього константи в заголовковий файл lis3dsh.h. Код програми представлено в додатку 4. В заголовковому файлі будуть зберігатися усі константи, які необхідні для роботи з датчиком а також прототипи функцій, які будуть використані в інших файлах. Модуль lis3dsh.c складається з методів для роботи з інтерфейсом SPI, налаштування та зчитування данних з датчика прискорення, а також обміну данних між акселерометром та контролером, серед яких:

- SPIx_WriteRead(uint8_t Byte) – функція читання та запису по шині SPI;
- Accel_IO_Read – функція читання обраної кількості байтів з адреси данних в буфер по шині SPI;
- Accel_IO_Write – функція запису обраної кількості байтів на адресу данних з буфера по шині SPI;
- Accel_ReadID(void) – смоук функція для перевірки, чи до того датчика ми доступилися;
- AccInit(uint16_t InitStruct) – функція ініціалізації контрольних регістрів акселерометра;
- Accel_GetXYZ(int16_t* pData) – метод для читання даних з датчика;

- Accel_ReadAcc(void) – головна функція функція пакету. Вона розбиває отримані данні на показники прискорення на 3 осях, та визначає правильність положення пристрою. Кінцевим результатом її роботи є відправка показників лінійного прискорення по осі Z на послідовний COM порт та запалення сигнальних ліхтарів при неправильному положенні пристрою;
- Accel_Ini(void) – функція ініціалізації датчика. Вона надсилає конфігураційні дані на контрольні регістри.

На цьому етапі вже все готово для повноцінної роботи з датчиком. Все що залишилося, це підключити дані файли до main.c, та виконати ініціалізацію датчика в коді виконавчого файлу, а також додати метод Accel_ReadAcc() в незкінченний цикл методу main. Після цього вже можна завантажувати виконавчий код на мікрокотролер. Для завантаження потрібно з'єднати плату з комп'ютером за допомогою AWG кабелю, провести компілювання проекту, а по закінченню, натиснути на кнопку Start Debugging Session в середовищі розробки. Після цього, зкомпільований код програми завантажиться на мікроконтролер.

На цей момент вже можна побачити перші результати (рис. 4.6) роботи пристрою, а саме переглянути положення пристрою в просторі. Це необхідно для правильного розміщення пристрою в робочому процесі. Для наглядності підкладемо ручку під платою в різних положеннях, які б змінювали оптимальне розміщення пристрою і подивимось на отримані сигнали. В положенні А пристрій розміщений оптимально, а отже всі ліхтарі вимкнені не запалюється. В положеннях Б і В пристрій, а отже і датчик розміщений під нахилом відносно осі абсцис, а в положеннях Г і Д відносно осі ординат. Це сигналізує, що при відхиленні пристрою від осей абсциси та ординати точність показників горизонтального прискорення падає. Тому, задля збереження правильності результатів, при неправильному положенні пристрою вмикається ліхтарик, який сигналізує в яку сторону відбувається відхилення і не вимикається аж до повного вирівнювання.

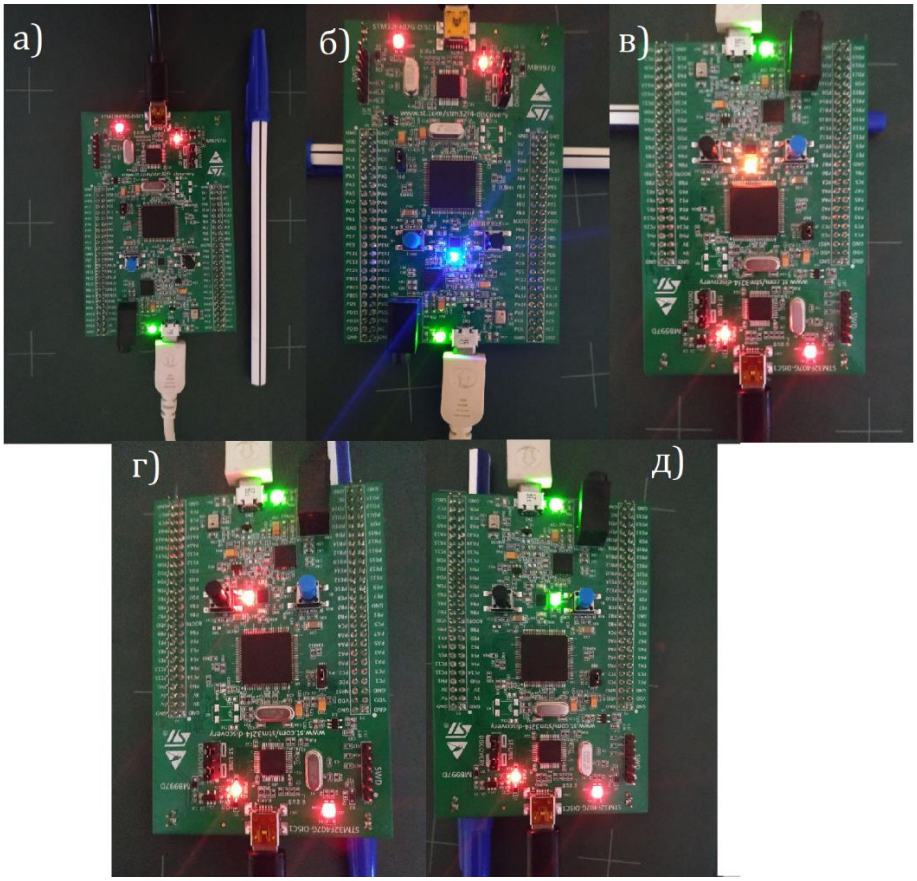


Рис. 4.6 Калібрування правильного положення пристрою

## 4.2 Розробка нейронної мережі для класифікації якості дорожнього покриття

Першим етапом перед розробкою та тренуванням нейронної мережі важливо зібрати достатній датасет. В цій роботі було зібрано власний датасет. Для цього було виконано пробний заїзд по ділянкам дороги різної якості. В якості датасету розглядається csv файл з трьома параметрами, кожен з яких представляє показник лінійного прискорення в одній із осей. Загальний розмір тренувального датасету скаладає 16000 записів. Всього в датасеті показники прискорення класифікуються по рівню якості на 4 види:

1. Рівна дорога

2. Бордюр, лежачий поліцейський, залізничний переїз
3. Дрібні ями
4. Великі ями (глибше 2 см, більше 20 см в діаметрі)

Данні збиралися в реальному часі і класифікувати якість дороги доводилось вручну. Загалом такі набори інформаціїї які збираються послідовно називають часовими рядами. На сьогоднішній день, аналіз та прогнозування часових рядів є досить поширеною темою в сферах фінансів, медицини та інших. Довгий час, основним рішенням в побудові таких мереж було використання рекурентних нейронних мереж RNN, проте в цій роботі мною було використано згорткову нейронну мережу CNN. Такий підхід є більш сучасним і не настільки складним, як часові згорткові мережі TCN, які потребують одразу дві натреновані моделі.

Згорткові нейронні мережі — це один з видів сучасних штучних нейронних мереж, котрі на одному або декількох шарах використовують операцію математичної згортки замість множення матриці.[34] Найчастіше такі нейронні мережі використовуються для обробки  зображень, але ми використаємо її саме для нашого набору данних.

Перш за все це пояснюється тим, що неможливо визначити якість дорожнього покриття саме по одному зафіксованому показнику. В нашій роботі для цього буде розглянуто набір послідовних показників прискорення, що перетворює наш набір векторів в колекцію матриць (кадрів). Саме це і призводить до того, що даний вид нейронних мереж підходить для цієї задачі оптимально.

Модель нейронної мережі складається з такого набору шарів:

1. Згортковий шар Conv2D розмірністю 16 з функцією активації ReLu
2. Шар Dropout для пониження ймовірності перетренування
3. Згортковий шар Conv2D розмірністю 32 з функцією активації ReLu
4. Ще один шар Dropout для пониження ймовірності перетренування

5. Шар Flatten, для пом’якшення входу данних
6. Шар Dense розмірністю 64 з функцією активації ReLu
7. Третій шар Dropout для пониження ймовірності перетренування мережі
8. Шар Dense на 4 виходи рівних кількості варіантів відповідей, з функцією активації softmax

Нейронна мережа тренувалася протягом 50 епох з оптимайзером Adam. Для розробки було використано бібліотеку keras з набору бібліотек мови Python. В результаті було отримано нейронну мережу, яка класифікує якість дорожнього покриття з точністю 85,2%. Графік показників точності отриманих в залежності від кількості епох продемонстровано на рис. 4.7.

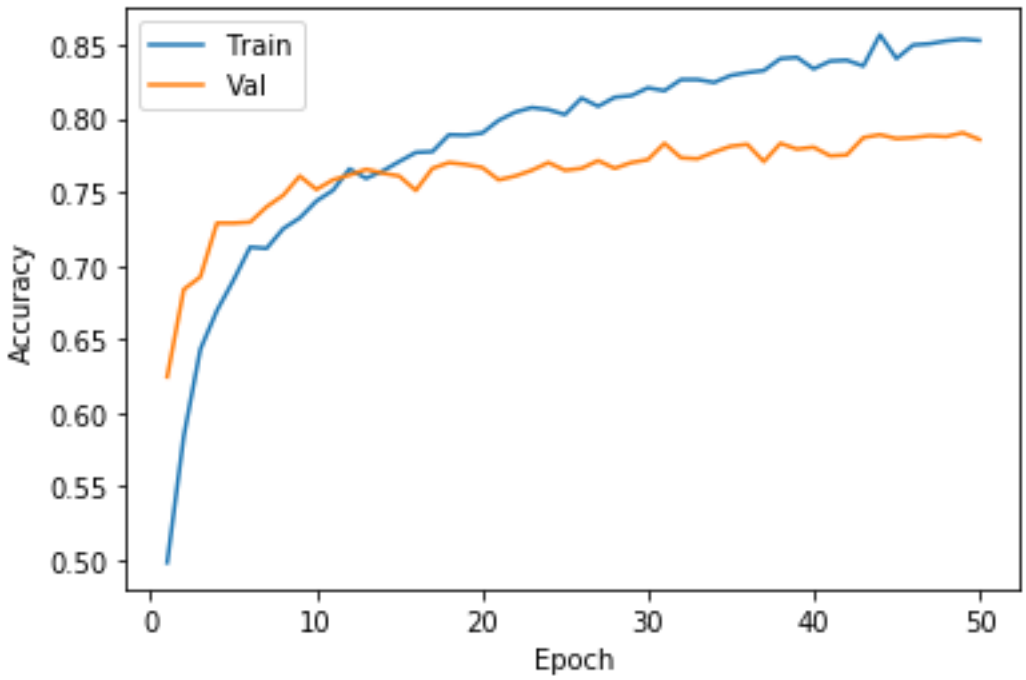


Рис.4.7. Графік точності нейронної мережі в залежності від кількості епох тренування

Підвищити точність роботи можливо при збільшенні розміру датасету. Таким чином натренована мережа матиме більшу точність.

## 4.3 Розробка програмного забезпечення для відображення роботи системи

На жаль, на сьогоднішній день немає можливості довести проект до логічного кінця в повному розумінні цього слова, оскільки для повного завершення пристрою необхідно додати GPS датчик та спосіб комунікації пристрою з сервером без прямого фізичного зв’язку. Ціллю цієї роботи є запропонувати концепт, а не завершений продукт, тому ми можемо робити абстрактні спрощення. Як вже було вказано раніше, пристрій спілкується з сервером за допомогою COM порту. Задля фіксаціїї положення спрощенням було постановлено отримувати данні GPS з робочої машини, котра є сервером для нашої системи. Тепер, коли систему можна вважати цілісною, то є можливість описати її принцип роботи.

Одразу після встановлення пристрою в правильному положенні він починає надсилати пакети данних показників трьохосьового акселерометра. Вони надходять до сервера і проходять етап обробки. На цьому етапі відбувається приведення данних до стандартних одиниць вимірювання і формуються кадри данних відповідного розміру, для проходження цих данних через класифікатор раніше описаної нейронної мережі. Один кадр данних представляє з себе виміри протягом 2.5 секунд. Після встановлення класифікатором якості дорожнього покриття на сервері визначається положення датчика та встановлюється відмітка спеціального кольору на Google Карті. Приклад вигляду карти з зафіксованими результатами якості (рис. 4.8.). Доступ до цієї карти зможе мати будь який користувач з усіх видів приладів, які надають можливість доступу до сервісів Google. Ці данні будуть мати лише оглядовчий характер для користувачів, без можливості редагування. Таким чином водії зможуть обирати маршрут з огляду на якість дорожнього покриття, а комунальні служби – пристрій для відстежування проблемних ділянок, які варто ремонтувати.

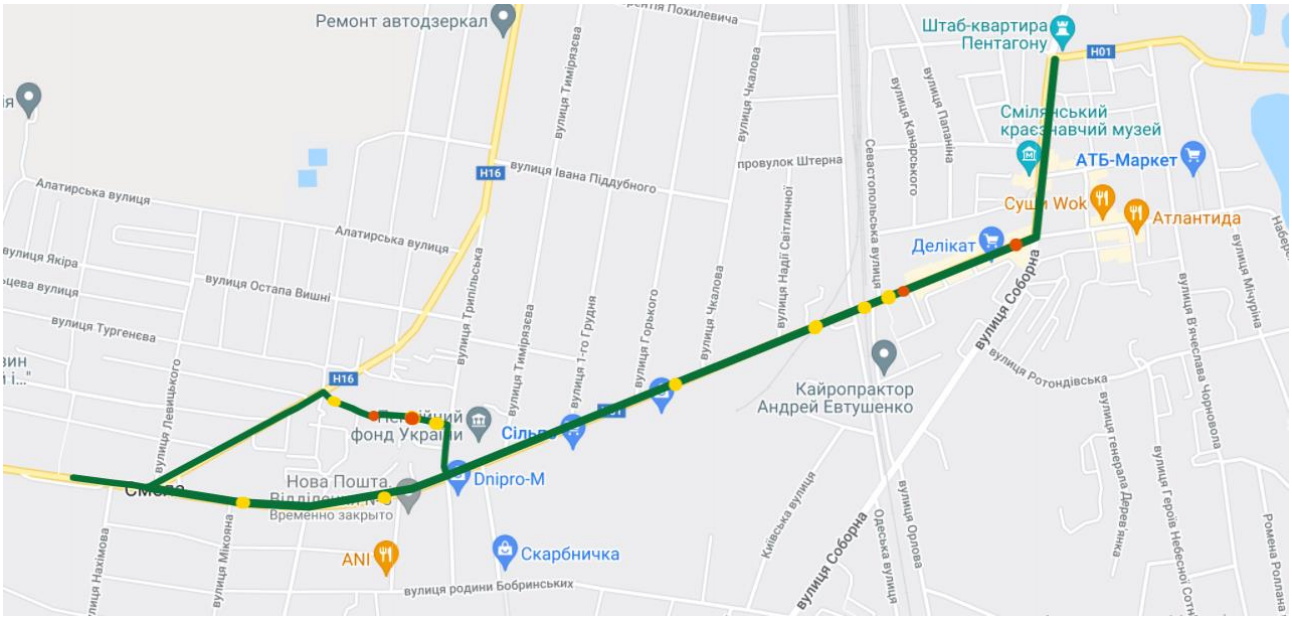


Рис.4.8. Вигляд відображення якості дорожнього покриття на карті

Окрім того, для більш візуального відображення роботи системи також було створено програму з графічним інтерфейсом, для більш детального перегляду роботи пристрою. Загальний вигляд програми представляє з себе поле для вибору COM порту, поля з відображенням шкал показників прискорення в різних проекціях, та графік показників прискорення по осі Z.

Оскільки важливо отримати не лише сухі цифри, та незрозумілі графіки без їх розтлумачення, то необхідно включити класифікатор, як частину програми. Алгоритм роботи не змінний. Данні з обраного порту, до якого підключений пристрій збираються, обробляються і формуються в кадри для виявлення неякісних ділянок дороги, а окрім того демонструють данні в реальному часі в різних відображеннях.

Опираючись на потрібний функціонал побудуємо графічну програму. Опишемо методи які необхідні для роботи програми. Загалом реалізовані такі функції:

- __init__- конструктор ініціалізації внутрішнього стану програми;
- zAxisRead –метод зчитування дануих з COM порту;
- draw – метод відбудови графіка та відображення якості дорожнього покриття;
- onButtonStartClicked – метод обробки події натискання кнопки “Open”;
- onButtonPauseClicked – метод обробки події натискання кнопки “Close”;
- onRadioButtonsClicked – метод обробки події натискання на елемент порту зі списку портів;
- handleClose - метод обробки події натискання на кнопку закриття програми;
- getSerialPorts – метод пошуку активних портів у комп’ютері;
- mainLoop – метод відображення вікна програми та елементів розміщених на ньому. Вхідна точка виклику програми.

Дана програма написана з використанням бібліотеки графічного інтерфейсу PyQt на мові програмування Python. Для зручного налаштування візуального представлення графічного інтерфейсу було використано програму Qt Designer. На рис. 4.9 можна побачити робочий інтерфейс програми після повного налаштування.

Для того щоб налштувати візуальне відображення Qt Designer володіє великим набором опцій для генерації файлу дизайну для стартового налаштування графічного інтерфейсу. Після підбору необхідних віджетів та їх налаштування залишається лише зберегти файл конфігурації та імпортувати його з сирцевого файлу проекту.

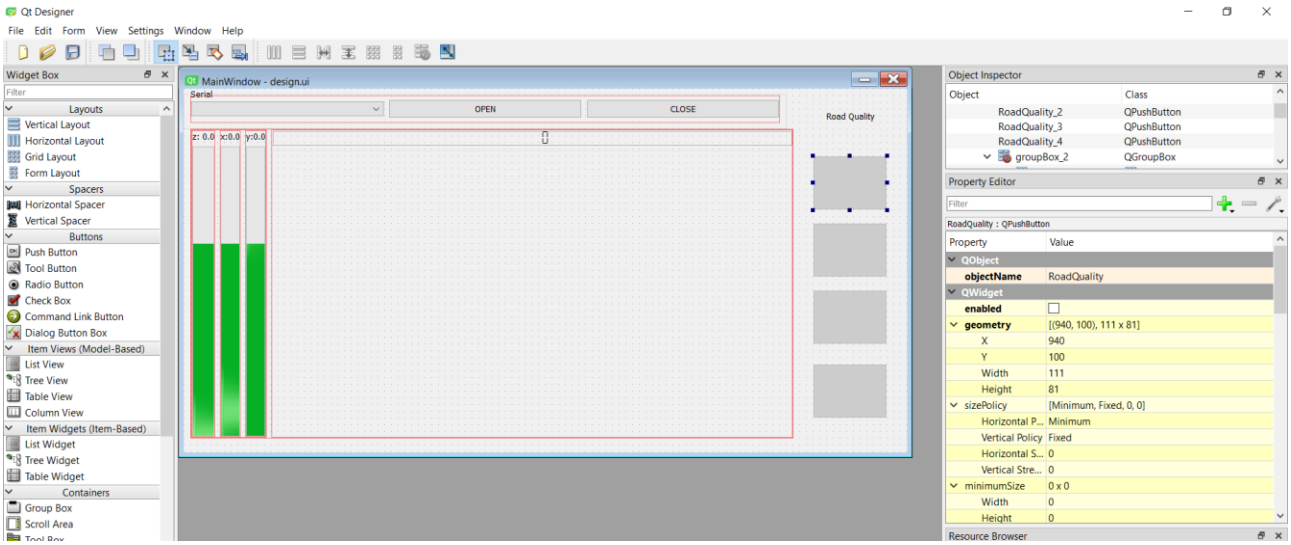


Рис. 4.9 Вигляд інтерфейсу програми в програмі Qt Designer

Для того щоб з консольної перетворити програму на зручну для користувача, залишилося зібрати її в один виконуваний файл. Для цього використано утиліту pyinstaller. Скористаємося консольною командою *pyinstaller -F -w –i "icon.ico" main.py*. Прапорці в цій команді відповідають за:

- –F – очищення папки dist від зайвого сміття по завершенню генерації exe файлу;
- -w – директива блокувати консольне вікно, яке з'являлося б під час запуску програми;
- -i – вказівник на адресу зображення іконки програми;
- main.py - файл, який потрібно перетворити в виконуваний.

## 4.3 Розгляд роботи інтерфейсу програми

Розглянемо інтерфейс програми для пошуку ям на дорозі. В результаті ми отримуємо програму для графічного відображення якості дорожнього покриття, загальний вигляд якої представлено на рис. 4.10.

Програма складається з вспливаючого вікна вибору порту з якого зчитуються данні, трьох шкал, на яких відображаються показники лінійного прискорення на різних осях координат. По центру робочого тла розміщено динамічний графік показників прискорення по осі Z та цифрові показники зверху над ним. Головним плюсом даного графіку є те, що він автопідбирає оптимальний масштаб графіку в залежності від максимального і мінімального показнику в даний проміжок часу.

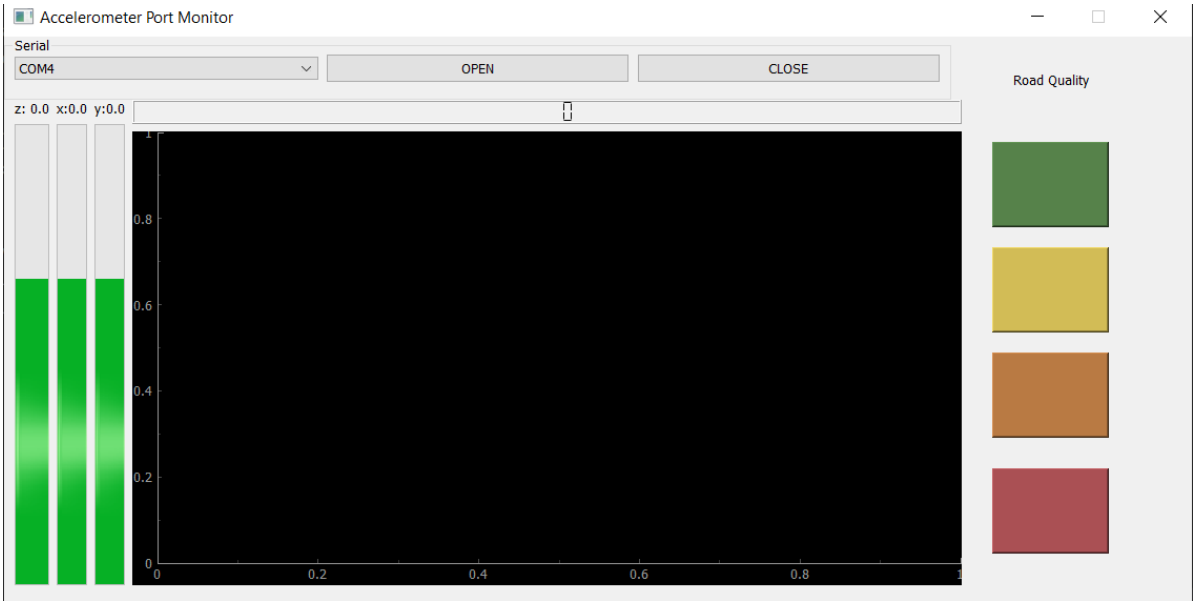


Рис. 4.10 Вигляд програми Accelerometer Port Monitor

Найбільш цікавим для нас є лінійка справа. Правий стовбець робочого тла відповідає за класифікацію якості дорожнього покриття. Справа розміщено чотири індикатора якості дорожнього покриття:

- зелений – рівна дорога;
- жовтий – лежачий поліцейський, залізничний переїзд, вибоїна;
- оранжевий – невелика яма, бруківка;
- червоний – аварійна ділянка, велика яма.

Програма збирає данні, розділяє їх на пакети необхідної довжини і надсилає їх на класифікатор для визначення якості дорожнього покриття. Після визначення на графічному інтерфейсі відображається відповідний індикатор і починається збір нового пакету данних. Такий процес буде продовжуватись до тих пір, поки на головному ерані не буде натиснуто кнопку “Close”. Приклад роботи програми зображено на рис. 4.11.

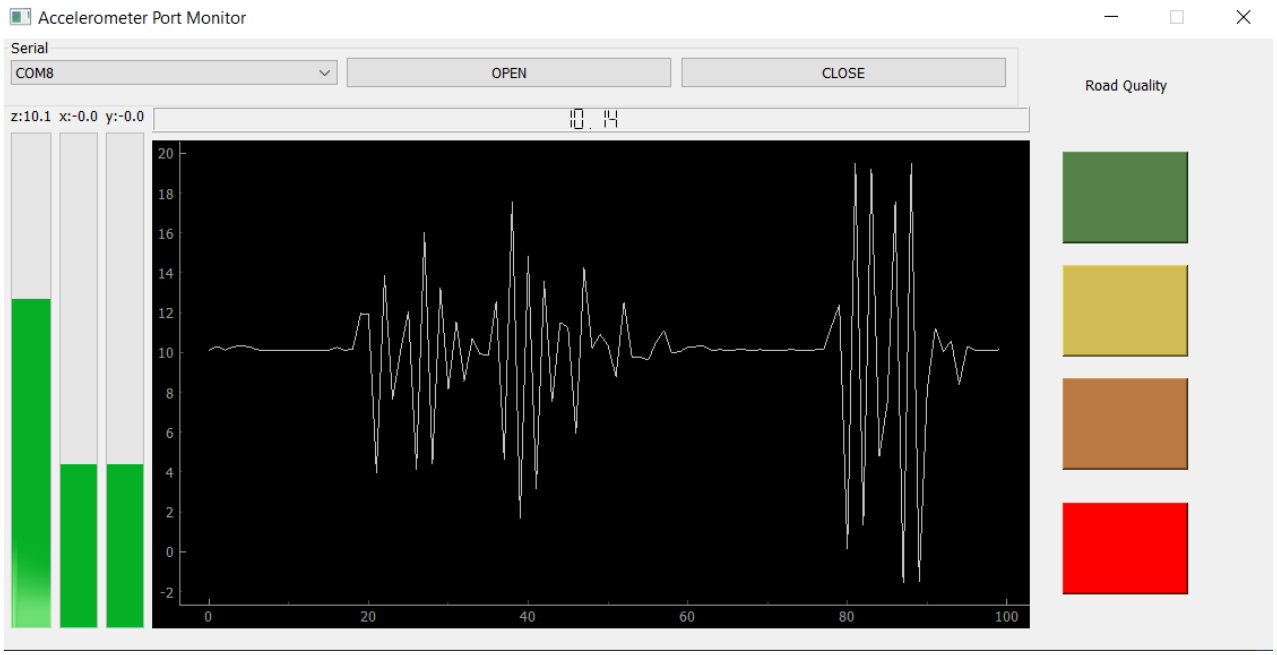


Рис. 4.11 приклад роботи графічного інтерфейсу

## Висновки до розділу 4

В четвертому розділі було виконано такі етапи роботи, як:

1. Реалізація програмного забезпечення для пристрою збору данних лінійного прискорення;
2. Розробка моделі нейронної мережі та її тренування на зібраному датасеті;
3. Розробка інтерфейсів для обробки та візуального відображення результатів роботи системи;
4. Опис методів, створених в ході реалізації системи;
5. Опис роботи візуальної частини роботи системи;
6. Тестування отриманих результатів;

Виходячи з отриманих результатів, можна стверджувати, що запропонований підхід має право на існування і показує гарні результати в порівнянні з аналогами. Говорячи про створений пристрій для збору інформації можна відмітити його наднизьке енергоспоживання, яке значно нижче в порівнянні з системами, котрі збирають данні за допомогою мобільних пристроїв. Точність роботи досить висока навіть на сьогоднішній день, а при збільшенні бази отриманих данних можна буде розширити тренувальний датасет, що підвищить в свою чергу точність роботи аналізатора. З недоліків можна виділити важливість правильного розміщення пристрою для правильних результатів роботи, але це легко вирішується завдяки вмонтуванню пристрою в автомобіль, що позбавить можливість змін його координатних осей.

# ВИСНОВКИ

Результатом магістерської дисертації є архітектурне рішення для створення системи класифікації якості дорожнього покриття з використанням нейронної мережі за показаннями лінійного прискорення, як частини системи Smart City. Також було реалізовано абстрактний прототип системи. Він був створенй опираючись на інформацію, отриману в ході огляду подібних розробок, їх основних переваг та недоліків.

Суть роботи системи – обробка показників лінійного прискорення, отриманих з пристрою, за допомогою нейронної мережі для класифікації стану дорожнього покриття та відображення отриманих результатів у зручному вигляді для кінцевого користувача.

Головна ціль проекту – розробка архітектурного рішення системи-класифікатора якості дорожнього покриття на базі нейронної мережі, як частини системи Інтернету речей, який можна було б інтегрувати в існуючу систему розумного міста в найкоротші сроки.

На сьогоднішній день було розроблено прототип системи. Він складається складається з пристрою для збору показників лінійного прискорення та серверу для обробки інформації та класифікації якості дорожнього покриття на базі данних, отриманих із пристрою. Класифікація відбувається за допомогою згорткової нейронної мережі. Таке апаратне забезпечення в подальшому може бути використано як частина більшого проекту, або на базі цього проекту можна розширювати подальший функціонал. Якість роботи нейронної мережі можна покращити шляхом збільшення розміру датасету.

В ході виконання магістерської дисертації, мною були отримані та закріплені на практиці знання про роботу з архітектурою мікроконтролерів STM, методи обробки сигналів та розробок згорткових нейронних мереж.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Причины образования различных деформаций дорожного полотна и методы контроля за ними. [Електронний ресурс]: https://ceiis.mos.ru/presscenter/news/detail/7577271.html (дата звернення 07.03.2022)
2. Передвижная лаборатория НПО «Регион» | НПО «Регион» [Електронний ресурс]: https://nporegion.ru/laboratorii/ (дата звернення 07.03.2022)
3. Алгоритмы детектирования разметки и дефектов дорожного покрытия [Електронний ресурс]: https://istina.msu.ru/publications/article/3717828/ (дата звернення 07.03.2022)
4. M. Kass, A. Witkin and D. Terzopoulos “Snakes: Active Contour Models” International Journal of Computer Vision, vol. 1, pp. 321-331, 1987 (дата звернення 07.03.2022)
5. Специальный ИИ помогает быстро ремонтировать дороги - 1Informer | новости, гаджеты, технологии [Електронний ресурс]: https://1informer.com/technologii/spetsialnyj-ii-pomogaet-bystro-remontirovat-dorogi-14569 (дата звернення 07.03.2022)
6. UaRoads [Електронний ресурс]: https://uaroads.com/ua/index (дата звернення 07.03.2022)
7. Украинские программисты запустили сервис, собирающий информацию о ямах на дорогах | AIN.UA [Електронний ресурс]: https://ain.ua/2014/05/15/ukrainskie-programmisty-zapustili-servis-sobirayushhij-informaciyu-o-yamax-na-dorogax-dannye-peredadut-v-ukravtodor/ (дата звернення 07.03.2022)
8. Ford предупредит водителя о ямах на дороге [Електронний ресурс]: https://chudo.tech/2017/02/22/ford-predupredit-voditelya-o-yamah-na-doroge/ (дата звернення 12.04.2022)

9. Volvo расскажет другой Volvo о гололеде [Електронний ресурс]: https://chudo.tech/2016/11/25/volvo-rasskazhet-drugoj-volvo-o-gololede/ (дата звернення 12.04.2022)
10. Jaguar и Land Rover научаться обнаруживать и избегать ямы на дорогах [Електронний ресурс]: https://adt.by/jaguar-i-land-rover-nauchatsya-obnaruzhivat-i-izbegat-yamy-na-dorogax/ (дата звернення 12.04.2022)
11. Arduino - Introduction [Електронний ресурс]: https://www.arduino.cc/en/Guide/Introduction (дата звернення 12.04.2022)
12. Что такое Arduino: первые шаги в освоении электроники / Амперка [Електронний ресурс]: https://amperka.ru/page/what-is-arduino (дата звернення 07.05.2022)
13. RPi Hub - eLinux.org [Електронний ресурс]: elinux.org/RPi_Hub (дата звернення 07.05.2022)
14. FAQ: ЦСП – цифровой сигнальный процессор [Електронний ресурс]: http://fpga.in.ua/dsp/dsp-theory/faq-csp-cifrovoj-signalnyj-processor.html (дата звернення 07.05.2022)
15. Для интенсивных вычислений: STM32F401 с ультранизким динамическим потреблением [Електронний ресурс]: https://www.compel.ru/lib/56926 (дата звернення 07.05.2022)
16. B. Lanjewar, Jyoti Khedkar, Rahul Sagar, Rasika Pawar, Kunal Gosavi. Survey of Road Bump and Intensity Detection algorithms using Smartphone Sensors. IJCSIT, Vol. 6, 2015. – [Електронний ресурс]: http://www.ijcsit.com/docs/Volume6/vol6issue06/ijcsit2015060659.pdf (дата звернення 07.05.2022)
17. Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, Hari Balakrishnan. The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. MobiSys, 2008. – [Електронний ресурс]:

https://www.cs.uic.edu/~jakob/papers/p2-mobisys08.pdf (дата звернення 07.05.2022)

18. Marius Hoffmann, Michael Mock, Michael May. Road-quality classification and bump detection with bycicle-mounted smartphones. – [Електронний ресурс]: http://ceur-ws.org/Vol-1088/paper7.pdf (дата звернення 11.05.2022)
19. Акселерометр. Виды и типы. Работа и применение. Особенности [Електронний ресурс]: https://electrosam.ru/glavnaja/slabotochnye-seti/oborudovanie/akselerometr/(дата звернення 11.05.2022)
20. Интегральные акселерометры [Електронний ресурс]: http://www.compitech.ru/html.cgi/arhiv/02_01/stat_66.htm (дата звернення 11.05.2022)
21. МЭМС-датчики движения от STMicroelectronics: акселерометры и гироскопы [Електронний ресурс]: https://russianelectronics.ru/mems-datchiki-dvizheniya-ot-stmicroelectronics-akselerometry-i-giroskopy/(дата звернення 11.05.2022)
22. LSM303DLHC - STMicroelectronics [Електронний ресурс]: https://www.st.com/resource/en/datasheet/DM00027543.pdf(дата звернення 11.05.2022)
23. LIS3DSH - STMicroelectronics [Електронний ресурс]: https://www.st.com/resource/en/datasheet/lis3dsh.pdf(дата звернення 11.05.2022)
24. Интерфейс SPI [Електронний ресурс]: http://s-engineer.ru/interfejs-spi/ (дата звернення 11.05.2022)
25. LSM9DS1 - STMicroelectronics [Електронний ресурс]: https://www.st.com/resource/en/datasheet/lsm9ds1.pdf (дата звернення 11.05.2022)
26. Обзор микроконтроллеров семейства STM32F4 [Електронний ресурс]: http://fpga.in.ua/dsp/dsp-theory/obzor-mikrokontrollerov-semejstva-stm32f4.html (дата звернення 11.05.2022)

27. STM32F407VG - STMicroelectronics [Електронний ресурс]: https://www.st.com/en/microcontrollers-microprocessors/stm32f407vg.html (дата звернення 14.05.2022)
28. Keil uVision - среда разработки программного обеспечения для микроконтроллеров [Електронний ресурс]: https://cxem.net/software/keil.php (дата звернення 14.05.2022)
29. STM32CubeMX – STMicroelectronics [Електронний ресурс]: https://www.st.com/en/development-tools/stm32cubemx.html (дата звернення 14.05.2022)
30. Description of STM32F4 HAL and LL drivers [Електронний ресурс]: https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf (дата звернення 14.05.2022)
31. Guidelines for Choosing a Computer Language: Support for the Visionary Organization [Електронний ресурс]: http://archive.adaic.com/docs/reports/lawlis/k.htm (дата звернення 17.05.2022)
32. Керниган Б., Ритчи Д. Язык программирования Си — 2-е изд. — М.: Вильямс, 2007. — С. 304. — ISBN 0-13-110362-8. (дата звернення 17.05.2022)
33. UM1472 User Manual [Електронний ресурс]: https://www.element14.com/community/docs/DOC-50074/l/stmicroelectronics-user-manual-of-stm32f4discovery-stm32f4-high-performance-discovery-board (дата звернення 17.05.2022)
34. Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016). Deep Learning. MIT Press. p. 32 (дата звернення 17.05.20220)
35. Копійка А. IoT система з використанням акселерометра для SmartCity [Електронний ресурс]: https://ela.kpi.ua/bitstream/123456789/34513/1/Kopiika_balakavr.pdf (дата звернення 17.01.20220)