МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри,
д-р техн. наук, проф.
_____ І. М. Журавська
« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

«Система моніторингу позиціювання об'єкта на базі одноплатного комп'ютера Raspberry Pi»

Спеціальність «Комп'ютерна інженерія»
123 – КМР.1 – 605.21710512

Студент _____*М. В. Ковальчук*
підпис
«__» _____ 202__ р.

Керівник д-р техн. наук, проф. __*І. М. Журавська*
підпис
«__» _____ 202__ р.

Миколаїв – 2023

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

ЗАТВЕРДЖУЮ

Зав. кафедри _____ І. М. Журавська

«_____» _____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної магістерської роботи

Видано студенту групи 605 факультету комп'ютерних наук

Ковальчук Микита Вадимович
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
Система моніторингу позиціювання об'єкта на базі одноплатного комп'ютеру Raspberry Pi

Затверджена наказом по ЧНУ ім. Петра Могили від 03.11.2022 № 201.
2. Строк представлення кваліфікаційної роботи «____» _____ 202__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні Розроблений прототип системи моніторингу позиціювання об'єкта на базі одноплатного комп'ютеру Raspberry Pi

4. Перелік питань, що підлягають розробці Аналіз рішень, що існують для систем моніторингу позиціювання об'єкта, визначення особливостей, переваг та недоліків; підбір елементної бази приладу; обґрунтування вибору компонентів, необхідних для реалізації системи; розроблення функціональної схеми; обрахування параметрів режимів роботи елементів схеми системи моніторингу позиціювання об'єкта; виконання розробки алгоритму функціонування системи моніторингу позиціювання об'єкта; імплементація розробленого алгоритму в апаратно-програмний модуль для моніторингу позиціювання об'єкта у просторі

5. Перелік графічних матеріалів

Макетна схема апаратно-програмного комплексу, слайди презентації_____

6. Завдання до спеціальної частини

Проведення оцінки умов праці фахівців підприємства ЧНУ ім. Петра Могили, а саме аналіз виробничого приміщення, оцінка природнього освітлення. Визначення рівня електричної та пожежної безпеки підприємства_____

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Д-р біол. наук, професор Григор'єва Л. І.	Кафедра екології Медичного інституту ЧНУ ім. Петра Могили	Спеціальна частина з охорони праці

Керівник роботи завідувач кафедри комп'ютерної інженерії, д-р техн. наук, проф. Журавська Ірина Миколаївна

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання. Здобувач вищої освіти

_____Ковальчук Микита Вадимович_____

(прізвище, ім'я, по батькові студента)

_____

(підпис)

Дата видачі завдання «____» _____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної магістерської роботи

Тема: Система моніторингу позиціювання об'єкта на базі одноплатного комп'ютера Raspberry Pi

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	**10.09.2022**	**24.09.2022**	**Виконано**
2.	Огляд літератури за темою роботи	**25.09.2022**	**08.10.2022**	**Виконано**
3.	Складання календарного плану КМР	**09.10.2022**	**23.10.2022**	**Виконано**
4.	Аналіз предметної області	**24.10.2022**	**07.11.2022**	**Виконано**
5.	Розробка проєктних рішень	**08.11.2022**	**22.11.2022**	**Виконано**
6.	Моделювання та конструювання АПЗ	**23.11.2022**	**08.01.2023**	**Виконано**
7.	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування, розробка керівництва користувача	**09.01.2023**	**20.01.2023**	**Виконано**
8.	Відгук керівника КМР	**21.01.2023**	**01.02.2023**	**Виконано**
9.	Оформлення КМР та презентації	**02.02.2023**	**13.02.2023**	**Виконано**
10.	Попередній захист	**15.02.2023**	**15.02.2023**	**Проведено**
11.	Рецензування	**15.02.2023**	**16.02.2023**	**Виконано**
12.	Завершення оформлення КМР та презентації	**16.02.2023**	**24.02.2023**	**Виконано**
13.	Захист кваліфікаційної роботи	**27.02.2023**	**27.02.2023**	**Захищено**

Розробив студент Ковальчук Микита Вадимович_____
(*прізвище, ім'я, по батькові)* *(підпис)*
«____» _____2022 р.

Керівник роботи завідувач кафедри комп'ютерної інженерії, д-р техн. наук, проф. Журавська Ірина Миколаївна
(*посада, прізвище, ім'я, по батькові)*

(підпис)

«____» _____ 2023 р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи
«Система моніторингу позиціювання об’єкта у просторі
на базі одноплатного комп’ютера Raspberry Pi»
Студент 605 гр.: Ковальчук Микита Вадимович
Керівник: д-р техн. наук, проф. Журавська І. М.

Системи позиціонування застосовуються в різних сферах, включаючи навігацію, стеження, охорону здоров’я, туризм, виробництво та особисту безпеку тощо. Більшість сучасних мобільних пристроїв мають тріади акселерометрів, гіроскопів та магнітометрів, часто на додаток до них ставиться і датчик атмосферного тиску. Цей набір датчиків може допомогти виявити різке погіршення фізичного стану власника мобільного пристрою, проте далеко не всі смартфони здатні отримати дані про власне позиціонування у просторі, та ще менше з них можуть робити це в автономному режимі довгий час.

Актуальність магістерської роботи полягає в тому, що розроблений комплекс може використовуватися в найрізноманітніших сферах від автомобільної індустрії до охорони здоров’я.

Мета магістерської роботи: розробити та оптимізувати програмно-апаратний комплекс моніторингу позиціювання об’єкта а просторі.

Об’єкт дослідження магістерської роботи: процес отримання інформації щодо позиціювання апаратного забезпечення у просторі.

Предмет дослідження магістерської роботи: методи та засоби отримання інформації щодо позиціювання апаратного забезпечення у просторі.

Практичне значення отриманих результатів: використання розробленого апаратно-програмного комплексу для моніторингу позиціювання об’єкта у просторі у системах навігації, стеженні, охороні здоров’я, туризмі, виробництві, особистій безпеці тощо.

Робота пройшла **апробацію** під час XXV Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 07–11 листопада 2022 р.).

Публікації. Основні положення магістерської роботи опубліковані у збірнику матеріалів XXV Всеукраїнської науково-практичної конференції «Могилянські читання–2022»

Пояснювальна записка магістерської роботи складається зі вступу, чотирьох розділів, висновків, переліку джерел посилання, додатків та спеціальної частини з охорони праці.

У вступі визначається актуальність теми, наведені задачі, які заплановано вирішити для досягнення поставленої мети. У першому розділі проводиться аналіз сфери та огляд існуючих аналогів систем моніторингу позиціювання. У другому розділі проведено огляд та підбір елементної бази для розробки апаратної частини кваліфікаційної роботи. У третьому розділі розроблено програмне забезпечення (ПЗ) для програмно-апаратного модулю.

Розробка ПЗ здійснена в Thonny, програмна частина створювалась на мові програмування CircuitPython. В четвертому розділі розроблено прототип системи моніторингу позиціювання, а також протестовано розроблений прототип у різних умовах. У висновках наводяться підсумки проведеної роботи та основні переваги розробленої системи перед аналогами.

У спеціальній частині з охорони праці та безпеки у надзвичайних ситуаціях проаналізовано систему заходів і засобів по запобіганню впливу на людину несприятливих факторів, які супроводжують роботу працівника ІТ-сфери. Виконано аналіз освітлення та мікрокліматичних умов на робочому місці, управління цивільним захистом на підприємстві у разі виникнення пожежі.

Сторінок – 68 (без додатків). Рисунків – 24. Таблиць – 4. Джерел посилання – 26. Додатків – 2.

Ключові слова*: контекстно-залежні системи, Raspberry Pi Pico, CircuitPython, мікроелектромеханічна система, молодший значущий біт.*

ABSTRACT

of the Master’s Thesis
"System for monitoring the positioning of an object in space
based on a Raspberry Pi single-board computer"
Student: Kovalchuk Mykyta
Supervisor: Dr. Sc. (Eng.), Professor Zhuravska I. M.

Positioning systems are used in various fields, including navigation, tracking, healthcare, tourism, manufacturing, and personal security, etc. Most modern mobile devices have triads of accelerometers, gyroscopes and magnetometers, often in addition to them and an atmospheric pressure sensor. This set of sensors can help detect a sharp deterioration in the physical condition of the owner of the mobile device, but not all smartphones are able to receive data about their own positioning in space, and even fewer of them can do it in an autonomous mode for a long time.

The relevance of the master's thesis lies in the fact that the developed complex can be used in a wide variety of areas, from the automotive industry to health care.

The goal of the master's work: to develop and optimize a software and hardware complex for monitoring the positioning of an object in space.

The object of research of the master's thesis: the process of obtaining information about the positioning of hardware in space.

The subject of research of the master's work: methods and means of obtaining information on the positioning of hardware in space.

The scientific novelty consists in the implementation of a software-hardware complex for monitoring the positioning of an object in space, which is characterized by the simplicity of the software part, the minimal use of modules, which will reduce the costs of building the complex.

The practical significance of the obtained results: after the analysis and summing up of the analyzed material, it is possible to understand the methods and principles of monitoring the positioning of the object in space, how to use them in the development of the hardware and software complex.

The work was approved during the XXV All-Ukrainian Scientific and Practical Conference "Mohyla Readings" (Mykolaiv, November 7-11, 2022).

Publications. The main provisions of the master's thesis were published in the collection of materials of the XXV All-Ukrainian Scientific and Practical Conference "Mohyla Readings-2022"

The explanatory note of the master's thesis consists of an introduction, four sections, conclusions, a list of reference sources, appendices and a special part on labor protection.

In the introduction, the relevance of the topic is determined, the tasks that are planned to be solved in order to achieve the set goal are given. In the first section, an analysis of the field and an overview of existing analogues of positioning monitoring systems is carried out. In the second section, an overview and selection of the element base for the development of the hardware part of the qualification

work was carried out. In the third section, the software (software) for the software-hardware module is developed. Software development was carried out in Thonny, the software part was created in the CircuitPython programming language. In the fourth chapter, a prototype of the positioning monitoring system was developed, and the developed prototype was tested in different conditions. The conclusions summarize the work carried out and the main advantages of the developed system over analogues.

In the special part of the health and safety emergency, a system of measures and means to prevent adverse factors that accompany the work of an IT worker is analyzed. Analysis of lighting and microclimatic conditions at the workplace, management of civil protection at the enterprise in case of fire was performed.

Pages – 68 (without appendices). Figures – 24. Tables – 4. References – 26. Appendices – 2.

Keywords*: context-dependent systems, Raspberry Pi Pico, CircuitPython, microelectromechanical systems, least significant bit.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ADL – Activities of daily living
ENBW – Equivalent noise bandwidth
GPS – Global Positioning System
IoT – Internet of Things
LTE – Long-Term Evolution
LSB – Least Significant Bit
MEMS – Microelectromechanical systems
MLM – Machine Learning Method
SE – Sensitivity
SMPS – Switched-mode power supply
SP – Specificity
TBM – Threshold-based method
Wi-Fi – Wireless Fidelity

АЦП – аналого-цифровий перетворювач
АЧХ – амплітудно-частотна характеристика
КЗС – контекстно-залежні системи
ФНЧ – фільтр низьких частот
ЦАП – цифро-аналоговий перетворювач

ВСТУП

Останні розробки в технологіях мобільного позиціонування та зростаючі вимоги повсюдного обчислення значно сприяли створенню складних програм і послуг позиціонування. Інформація про місцезнаходження є ключовим елементом діяльності, що орієнтована на людину, допомагає ефективно візуалізувати складне середовище та забезпечити репрезентативні координати для локалізації, відстеження й навігації. Використання систем позиціонування актуальне для застосування у різних сферах, а саме автомобільної індустрії, системах навігації, стеженні, охороні здоров’я, туризмі, виробництві, особистій безпеці тощо.

Більшість сучасних мобільних пристроїв мають тріади акселерометрів, гіроскопів та магнітометрів, часто на додаток до них ставиться і датчик атмосферного тиску [2]. Цей набір датчиків може допомогти виявити різке погіршення фізичного стану власника мобільного пристрою, проте далеко не всі смартфони здатні отримати дані про власне позиціонування у просторі та ще менше з них можуть робити це в автономному режимі довгий час.

Точність має вирішальне значення для належного функціонування системи позиціонування. На практиці поки що не існує єдиної техніки позиціонування, яка б підходила для різних ситуацій.

Мета магістерської роботи: розробити та дослідити програмно-апаратний комплекс моніторингу позиціювання об’єкта у просторі.

Об’єкт дослідження магістерської роботи: процес отримання інформації щодо позиціювання пристроїв у просторі.

Предмет дослідження магістерської роботи: методи та засоби отримання інформації щодо позиціювання пристроїв у просторі.

Для досягнення поставленої мети необхідно вирішити такі **завдання:**

– проаналізувати технології та методи виявлення зміни положення об’єктів у просторі;

– розглянути існуючи програмно-апаратні комплекси моніторингу позиціювання об’єкта;

– проаналізувати можливі апаратні платформи для реалізації прототипу;

– розробити прототип системи моніторингу позиціювання об’єкта у просторі;

– протестувати розроблений прототип системи;

– розробити рекомендації щодо впровадження результатів роботи.

Практичне значення отриманих результатів: використання розробленого апаратно-програмного комплексу для моніторингу позиціювання об’єкта у просторі у системах навігації, стеженні, охороні здоров’я, туризмі, виробництві, особистій безпеці тощо.

Робота пройшла **апробацію** під час XXV Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 07–11 листопада 2022 р.).

Публікації. Основні положення магістерської роботи опубліковані у збірнику матеріалів XXV Всеукраїнської науково-практичної конференції «Могилянські читання–2022» [26].

1 АНАЛІТИЧНА ЧАСТИНА. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ СИСТЕМИ, ЩО РОЗРОБЛЯЄТЬСЯ. ФОРМУВАННЯ ВИМОГ ДО АПАРАТНО-ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Сьогодні системи моніторингу позиціонування об’єктів все частіше використовуються у нашому житті. Визначення «позиціонування» зазвичай використовується для вираження здатності локалізувати фізичне положення об’єкта в попередньо визначеному просторі. Зростання повсюдних обчислень і контекстно-залежної інформації сформувало напрям до більш досконалих систем позиціонування. Крім того, останні розробки в бездротових технологіях дали початок новим парадигмам методів позиціонування, які можна підтримувати в різних ситуаціях. Термін «позиціонування» має те саме поняття, що й «локалізація»; однак це часто співвідноситься з характеристиками реального часу.

Системи позиціонування застосовуються в різних сферах, включаючи навігацію, стеження, охорону здоров’я, туризм, виробництво та особисту безпеку [18]. Схожі системи можна зустріти як в дронах, що можуть летіти за заданим маршрутом, так і в медичних приладах, що можуть виявити падіння людини та проінформувати родичів або медичний персонал.

1.1 Методи та підходи визначення позиціонування

Відстеження положення є поєднанням апаратних засобів [5] і програмного забезпечення, яке дозволяє визначити точного положення об'єкта в просторі.

Серед методів та підходів для вирішення поставленої задачі можна виділити наступні групи:

- акустичні;
- радіочастотні;
- магнітні;
- оптичні;

– інерційні;
– гібридні.

Кожний підхід має власні особливості, тому далі буде детально розглянуто кожну групу методів.

1.1.1 Акустичні методи

Акустичні прилади стеження використовують ультразвукові (високочастотні) звукові хвилі для вимірювання положення та орієнтації цільового об'єкта. Для визначення положення об'єкта вимірюється час прольоту звукової хвилі від передавача до приймачів або різниця фаз синусоїдальної звукової хвилі при прийомо-передачі. Компанія Intersense розробляє пристрої відстеження позиції на основі ультразвуку.

Акустичні трекери зазвичай мають меншу швидкість оновлення, викликану низькою швидкістю звуку в повітрі. Також проблемою є те, що швидкість звуку в повітрі залежить від багатьох факторів зовнішнього середовища, таких як температура, вологість, барометричний тиск тощо.

1.1.2 Радіочастотні методи

Методів заснованих на радіочастотах безліч. Багато в чому за принципами визначення становища вони схожі на акустичні методи відстеження (відмінність лише в природі хвилі). Найбільш перспективними на даний момент є методи Ultra-Wide Band (UWB), але навіть у кращих рішеннях на основі UWB точність досягає лише до декількох сантиметрів (DW1000 від DecaWave, Dart від Zebra Technologies, Series 7000 від Ubisense та інші). Можливо, у майбутньому стартапам на зразок Pozyx або IndoTraq вдасться досягти субміліметрової точності. Однак поки що UWB рішення для відстеження позиції не мають достатньої точності.

1.1.3 Магнітні методи

Магнітний трекінг заснований на вимірюванні інтенсивності магнітного поля у різних напрямках. Як правило, у таких системах є базова станція, яка генерує змінне або постійне магнітне поле. Оскільки сила магнітного поля зменшується зі збільшенням відстані між точкою вимірювання та базовою станцією, можна визначити місце розташування пристрою. Якщо точка вимірювання обертається, розподіл магнітного поля змінюється за різними осями, що дозволяє визначити орієнтацію. Найбільш відомими продуктами на основі магнітного трекінгу є контролер Razer Hydra та система STEM від компанії Sixense.

Точність даного методу може бути достатня висока в контрольованих умовах (у специфікаціях Hydra говориться про 1 мм позиційної точності і 1 градусі точності орієнтації), однак магнітне відстеження піддається перешкодам від струмопровідних матеріалів поблизу випромінювача або датчика, від магнітних полів, що створюються іншими матеріалами у просторі відстеження.

1.1.4 Оптичні методи

Оптичні методи являють собою сукупність алгоритмів комп'ютерного зору і пристроїв, що відстежують, в ролі яких виступають камери видимого або інфрачервоного діапазону, стерео-камери і камери глибини.

Залежно від вибору системи відліку виділяють два підходи для відстеження положення (рис. 1.1).

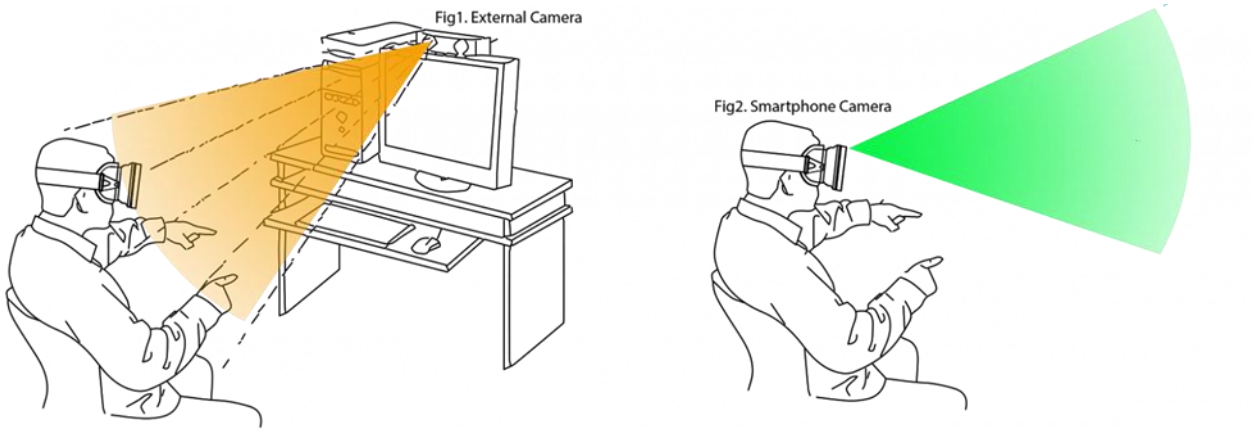

Рисунок 1.1 – Два підходи оптичного методу позиціонування

Підхід «outside-in» має на увазі присутність нерухомого зовнішнього спостерігача (камери), який визначає положення об'єкта, що рухається, за характерними точками. Використовується в Oculus Rift (Constrellation), PSVR, OSVR та безлічі Motion Capture систем.

Підхід «inside-out» передбачає наявність на об'єкті оптичного сенсора, що рухається, завдяки якому можливо відстежувати рух щодо нерухомих точок в навколишньому просторі. Використовується Microsoft Hololens, Project Tango (SLAM), SteamVR Lighthouse (гібридний варіант, тому є базові станції).

1.2 Носимі пристрої

Одними з багатьох систем моніторингу позиціювання об’єкта можна виділити пристрої виявлення падіння. Носимі пристрої можна визначити як мініатюрні електронні пристрої на основі датчиків, які носій носить під одягом, з ним або поверх нього. Переважна більшість носимих детекторів падіння мають форму акселерометрів. Деякі з них також містять інші датчики, такі як гіроскопи, для отримання інформації про положення пацієнта.

Активно досліджується використання застосунків на основі акселерометрів і гіроскопів для оцінки ходи та балансу, оцінки ризику падіння та моніторингу рухливості. Ця тенденція посилилася протягом

останніх років через доступність дешевих вбудованих датчиків, що входять до складу смартфонів.

1.2.1 Акселерометр, прикріплений до тіла

Дані про прискорення збираються під час падінь за допомогою незалежних трьох-осьових акселерометрів, прикріплених до різних частин тіла.

Застосовувані методи включають машину опорних векторів, оптимізацію роєм частинок перегрупування, розподіл гауссового кластерного знання, багатошаровий персептрон, наївний баєсів класифікатор, дерево рішень, ZeroR і OneR. Згідно досліджень [14], багатошаровий персептрон є хорошим варіантом під наглядом, хоча не існує стандартизованої методики, яка б була широко прийнята науковою спільнотою.

Ширші категорії типових подій падіння – це вперед, назад і вбік. Що стосується положення акселерометра, місце розташування на талії здається оптимальним для виявлення падіння. Прикріплені на талії акселерометри розташовані поблизу центру ваги тіла, надаючи надійну інформацію про рухи тіла суб’єкта.

Як і в попередній категорії, заявлена продуктивність дуже висока, але справа в тому, що ці пристрої мало використовуються в повсякденній практиці та не мають значного промислового впровадження детекторів падіння через значну кількість помилкових тривог, що призводить до невідповідних оповіщення. Заявлені результати дійсні для лабораторних середовищ з обмеженими даними або в обмежених умовах, але в реальних сценаріях є багато неконтрольованих факторів, які призводять до різкої втрати продуктивності. Тому дослідження повинні включати більш тривалі тести та включати показники, ближчі до реального використання, наприклад, кількість помилкових тривог на день, що може дати цільовим користувачам більш реалістичне уявлення про справжню продуктивність системи.

1.2.2 Вбудований в смартфон акселерометр

Сучасні смартфони оснащені багатим набором вбудованих датчиків, таких як акселерометр, цифровий компас, гіроскоп, GPS, мікрофон і камера. Зараз кілька дослідників користуються цим фактом для розробки детекторів падіння на основі смартфонів.

Алгоритми низької складності, засновані на порогових значеннях, використовуються в більшості рішень, і лише деякі йдуть далі та застосовують стратегії машинного навчання. Вони використовують опорні векторні машини, розріджену мультиноміальну логістичну регресію, наївну баєсівську систему, K-найближчі сусіди, дерева рішень і багаторівневі нейронні мережі.

Що стосується положення телефону, то талія залишається кращою частиною тіла, хоча спостерігається тенденція до стегна, що збігається з розташуванням кишені.

Деякі з досліджень призвели до реальних програм для виявлення падінь, які доступні для завантаження в Google Play. Таким чином, у цьому сховищі застосунків було проведено пошук із ключовими словами «детектор падіння» або «виявлення падіння». Загалом було отримано 9 застосунків, з них 7 для літніх людей. Наводячи деякі статистичні дані, 3 з них повідомили про від 1000 до 5000 завантажень, тоді як решта мали менше 500. Хоча ці цифри вказують на певний рівень інтересу, вони все ще далекі від кількості потенційних користувачів. Зосереджуючись на рейтингу програми, було виявлено, що в середньому лише 6 людей висловили свою думку про них. Це ознака того, що люди, які користуються цими програмами, здається, не в захваті від них.

1.3 Виклики

Конструкція систем моніторингу позиціювання стикається з деякими серйозними викликами.

1.3.1 Продуктивність у реальних умовах

Системи моніторингу позиціювання повинні бути максимально точними та надійними у будь-яких умовах. Надійна система моніторингу позиціювання повинна демонструвати як високу чутливість, так і специфічність. Це іноді досягається в експериментальних середовищах, але при застосуванні до реальної ситуації швидкість виявлення знижується.

Ці пристрої розроблені та перевірені в контрольованих умовах, наприклад, вони використовують дані про позиціювання, змодельовані в різних умовах роботи, наприклад дощ, спека або сильний холод через відсутність стандартизованої процедури або загальнодоступної бази даних для порівняння.

1.3.2 Зручність використання

Складність пристрою створює серйозну проблему для людей не знайомих з електронікою. Щоб подолати цю проблему, важливе значення має спосіб функціонування системи.

Система має активуватися та працювати автоматично, без втручання користувача.

1.3.3 Простота

Системи моніторингу позиціювання мають на борту багато акселерометрів та датчиків, що іноді не дозволяє системі буди гнучкою до встановлення на різні платформи.

Майбутні системи моніторингу позиціювання не повинні обмежувати розміщення пристрою однією платформою. Система має бути гнучкою для встановлення як на медичне обладнання, так і на авто або дрон.

1.4 Проблеми

Існують деякі проблеми, які можуть перешкоджати продуктивності системи. З них можна виокремити: обмеження платформи на яку встановлюється система, різка зміна погодних умов.

1.4.1 Обмеження платформи, на яку встановлюється система

Тенденція до детекторів на основі смартфонів створює деякі проблеми. По-перше, смартфони не є пристроями, спочатку призначеними для моніторингу позиціювання або будь-якої іншої важливої для безпеки програми.

Можуть виникнути труднощі з операціями в реальному часі, архітектурою датчиків, стабільністю частоти дискретизації акселерометра, особливостями операційної системи тощо. Цю особливість слід враховувати в реальному сценарії.

По-друге, смартфони не можна перевантажувати безперервними датчиками, які погіршують продуктивність телефону, наприклад, виснажуючи заряд батареї. Важливо керувати циклом сну сенсорних компонентів, щоб компенсувати кількість споживаного заряду батареї. Тим не менш, час автономної роботи смартфона завжди низький, що може перешкодити його прийняттю.

По-третє, потрібні прості у використанні смартфони. Потенційний ринок застосунків для людей з низькими технічними навичками вплине на розвиток адаптованих пристроїв. Тим не менш, детектори падіння навряд чи досягнуть у найближчому майбутньому надійності та стабільності, досягнутої іншими допоміжними технологіями, такими як пристрої для натискання на допомогу.

1.4.2 Різка зміна погодних умов

Якщо розглядати систему позиціонування, що буде встановлена на дрон або інший літаючий апарат, то велику увагу слід приділити роботі під час різних погодних умов. Оскільки такі системи можуть використовуватися для коригування польоту в автоматичному режимі, різкі пориви вітру можуть призвести до різкої зміни показників датчиків, що в свою чергу може дати занадто велике коригування і пристрій може змінити свій напрямок або взагалі впасти.

1.5 Аналогічні рішення

На ринку існує чимала кількість приладів, в яких вбудована функція моніторингу положення у просторі, але більшість з них має високу ціну або працює тільки в окремих країнах.

1.5.1 DJI Mavic 3 Fly More Combo

Mavic 3 Fly More Combo – флагманський дрон від DJI, що обладнаний двома камерами (рис. 1.2). Дрон отримує своє місцезнаходження за допомогою супутників GPS, тому він знає, де він знаходиться на земній кулі, і може ідеально зависати на місці, навіть при вітрі. Для польоту у приміщенні, щоб тримати дрон стабільно, є спрямовані вниз датчики візуального позиціонування працюють, щоб утримувати його на місці.

Рисунок 1.2 – Квадрокоптер DJI Mavic 3 Fly More Combo

Також дрон має функцію Advanced Return to Home, що призначена для забезпечення безпечного повернення квадрокоптера DJI Mavic 3 до точки зльоту. Він визначає кілька оптимальних варіантів повернення на базу і працює навіть коли загубився сигнал.

Всеспрямована система виявлення перешкод ActiveTrack 5.0 відповідає за безпечний політ без зіткнень. Нові алгоритми відео датчиків роблять процес повернення плавним і стабільним, дрон заздалегідь будує оптимальний маршрут. Датчики відстежують перешкоди в усіх напрямках та працюють швидше, ніж у попередній версії квадрокоптера.

QuickTransfer передає інформацію швидше за 80 Мбайт на секунду. Автоматично визначає, який варіант передачі даних буде найшвидшим. Працює на вбудованому модулі Wi-Fi 6 (802.11 ax).

1.5.2 Galaxy On the Go Mobile System

На рис. 1.3 можна побачити прилад для виявлення падіння від компанії Galaxy Medical Alert Systems.

Рисунок 1.3 – Galaxy On the Go Mobile System with GPS & Fall Detection

Дане рішення базується на трьохосьовому акселерометру з власними розробленими алгоритмами виявлення падіння. В наявності GPS, WiFi-модулі дані з яких оновлюються кожні 30 секунд після натискання кнопки Help, а також LTE-модуль для зв’язку з медичним закладом. Ресурс девайсу – до 96 годин постійної роботи. Працює виключно на території Канади з щомісячною передплатою від 50$.

Розробник зауважує, що даний пристрій не виявить 100% усіх падінь. Завжди потрібно натискати кнопку на девайсі, щоб отримати допомогу якомога швидше. Прилад був розроблений та рекомендований виробником для носіння на зовнішній частині одягу для правильного використання системи виявлення падіння.

1.5.3 Apple Watch SE 2

Apple Watch SE 2 оснащено функцією виявлення падіння. Цю функцію потрібно налаштувати за допомогою програми Watch на iPhone.

Після ввімкнення Apple Watch SE 2 може виявити сильне падіння та автоматично повідомити служби екстреної допомоги для подальшої допомоги. Якщо падіння сталося, годинник подасть звуковий сигнал і відобразить сповіщення. Якщо людина продовжує рухатись, годинник не

буде автоматично викликати екстрену службу і чекатиме вашої відповіді. Однак, якщо людина не рухається більше хвилини, годинник починає 30-секундний відлік, вібруючи та лунаючи звуковий сигнал, який поступово стає голоснішим. Після закінчення зворотного відліку він автоматично сповістить екстрені служби та повідомить поточне місцезнаходження.

На рис. 1.4 продемонстровано зовнішній вигляд Apple Watch SE 2.

Рисунок 1.4 – Смарт-годинник Apple Watch SE 2

Вартість даного пристрою від 300$. З мінусів можна виокремити, що для роботи смарт годинника його потрібно активувати тільки за допомогою iPhone з операційною системою від iOS 12.

1.6 Вимоги до апаратно-програмного забезпечення

Після аналізу доступної інформації, було сформовані певні вимоги до майбутнього апаратно-програмного забезпечення:

– невеликий розмір для зручного встановлення;

– простота використання;

– велика точність моніторингу позиціювання;

– розмір програмної частини не повинен перевищувати ліміт платформи;

– низька ціна у порівнянні з аналогами.

Висновки до розділу 1

У розділі 1 було досліджено та проаналізовано матеріал щодо різних систем, технологій та методів моніторингу позиціювання об’єкта. Також було розглянуто методи моніторингу позиціювання об’єкта: акустичні, радіочастотні, магнітні, оптичні, інерційні, гібридні.

Серед проблем у існуючих системах позиціонування було виділено обмеження платформи, на яку буде становлено систему. Через обмеження платформи можуть виникнути труднощі з операціями в реальному часі, архітектурою датчиків, стабільністю частоти дискретизації акселерометра, особливостями операційної системи тощо.

Були проаналізовані аналоги, та сформовані вимоги до майбутнього апаратно-програмного забезпечення. Апаратно-програмний комплекс повинен бути: невеликого розміру для зручного носіння, простим у використанні, точним при розрахунках позиціювання об’єкта, розмір програмної частини не повинен перевищувати ліміт платформи, дешевшим у порівнянні з аналогами.

2 АНАЛІЗ ПАРАМЕТРІВ MEMS-АКСЕЛЕРОМЕТРІВ, ГІРОСКОПІВ ТА МАГНІТОМЕТРІВ

Мікроелектромеханічна система (англ. Microelectromechanical systems, MEMS) – це загальний термін для широкого діапазону мікроконструкцій, методів і механізмів, які передбачають реалізацію рухомих механічних частин у мікроскопічному масштабі.

MEMS використовуються в різних датчиках, приводах, генераторах, джерелах енергії, біохімічних і біомедичних системах та осциляторах. Деякі приклади застосування MEMS у проєктуванні інженерних виробів включають:

– датчики, такі як MEMS акселерометри, мікрофони, гіроскопи, датчики тиску, датчики нахилу та інші типи резонансних датчиків;

– приводи, такі як MEMS перемикачі, мікронасоси, мікроважелі та мікрозахвати;

– генератори та джерела енергії, такі як MEMS збирачі вібраційної енергії, паливні елементи і радіоізотопні генератори енергії;

– біохімічні та біомедичні системи, такі як MEMS біосенсори, лабораторії на чіпах, повітряні мікрофлюїдні датчики та датчики частинок;

– MEMS осцилятори для точного вимірювання часу та керування частотою.

Сьогодні кожен носить на собі MEMS пристрої у вигляді смартфонів, розумних годинників і фітнес-трекерів.

2.1 MEMS-гіроскоп

Розглянемо датчик MPU-9250, який складається з трьох незалежних одновісних вібраційних датчиків кутової швидкості (MEMS гіроскопів), які реагують на обертання навколо X-, Y-, Z-осей. Дві підвішені маси чинять коливання по протилежним осям.

З появою кутової швидкості ефект Коріоліса викликає зміну напрямку вібрації, що фіксується ємнісним датчиком:

$$\overrightarrow{F_K} = -2m[\vec{w}x\overrightarrow{v_r}]. \quad (2.1)$$

Вимірювана диференціальна ємнісна складова пропорційна куту переміщення. Отриманий сигнал посилюється, демодулюється і фільтрується, даючи в результаті напругу, пропорційне кутовій швидкості обертання.

Цей сигнал оцифровується за допомогою вбудованого в плату 16 бітного АЦП. Швидкість оцифрування може програмно змінюватись від 3,9 до 8000 вибірок в секунду, а фільтри низьких частот (ФНЧ), що задаються користувачем, надають широкий діапазон можливих частот зрізу. ФНЧ потрібен, у тому числі, щоб прибирати вібрації від моторів (як правило, вище 20–25 Гц).

2.2 Триосьовий MEMS акселерометр

Триосьовий MEMS акселерометр використовує для кожної осі окрему пробну масу, яка зміщується у разі прискорення вздовж даної осі (фіксуються ємнісними датчиками). Архітектура MPU-9250 знижує схильність до температурного дрейфу і варіацій електропараметрів. При розташуванні пристрою на плоскій поверхні акселерометр виміряє 0 g по X- та Y-осях і +1 g по Z-осі.

Масштабний коефіцієнт – відношення зміни вихідного сигналу до зміни вихідного сигналу, що вимірюється. Він калібрується на заводі і не залежить від напруги живлення. Кожен сенсор забезпечений індивідуальним сигма-дельта аналого-цифровим перетворювачем (АЦП). Він складається з модулятора і цифрового фільтра низьких частот, вихідний цифровий сигнал якого має діапазон вимірювань.

2.3 Триосьовий MEMS магнітометр

Заснований на високоточній технології ефекту Холла. Включає магнітні сенсори, що визначають напруженість магнітного поля землі по осях, схему управління, ланцюг посилення сигналу і обчислювальну схему для обробки сигналів з кожного датчика.

Кожен АЦП має роздільну здатність 16 біт, діапазон вимірювань ± 4800uT. Для вимірювання слабких магнітних полів застосовують або одиницю в системі СІ мікротесла (мкТл), або гаус (Гс): 1 = 100μ.

2.4 Least Significant Bit та його розрахунок

Визначення «позиціонування» зазвичай використовується для вираження здатності локалізувати фізичне положення об’єкта в попередньо визначеному просторі. Чим менші відхилення здатен розпізнати той чи інший сенсор, тип біль точним він вважається. Одиниця молодшого розряду використовується для вказівки точності або роздільної здатності приладів з цифровим індикатором.

Припустимо, що обраний акселерометр зараз працює в діапазоні вимірів Fs = ±2g, тобто повний розмах можливих значень буде 2 * Fs = 4g. Відповідні їм значення напруги оцифровуються 16-бітним АЦП, який може розбити весь інтервал максимально на 2^{16} = 65536 ступенів. Мінімальний інкремент, який можна засікти, – це одна сходинка:

$$LSB = 2 * FS/65536. \quad (2.2)$$

Тут треба пам'ятати, що рахунок ведеться з нуля, так що насправді максимально вимірюване значення буде:

$$2 * FS_{true} = (2^{16} - 1) * LSB = 65535 * LSB = 2 * FS - LSB. \quad (2.3)$$

Тобто чим більше біт у цифровому слові АЦП або ЦАП тим менше буде розбіжність. При цьому чутливість (іноді називається масштабним коефіцієнтом) датчика на конкретному діапазоні визначатиметься як співвідношення електричного вихідного сигналу та механічного впливу. Традиційно вказується для частоти сигналу 100 Гц і температури T = +25°C.

Для MPU-9250 чутливість становить 2^{16} / (2 * Fs) ступенів на кожні g або ∘ s(LSB / г, LSB / (∘ s)), для іншого IMU, BMI088 від Bosch Sensortec, чутливість гіроскопа обчислюється так само, а для акселерометра використовується ($2^{16} – 2^4$) / (2 *Fs) ступенів на кожне g.

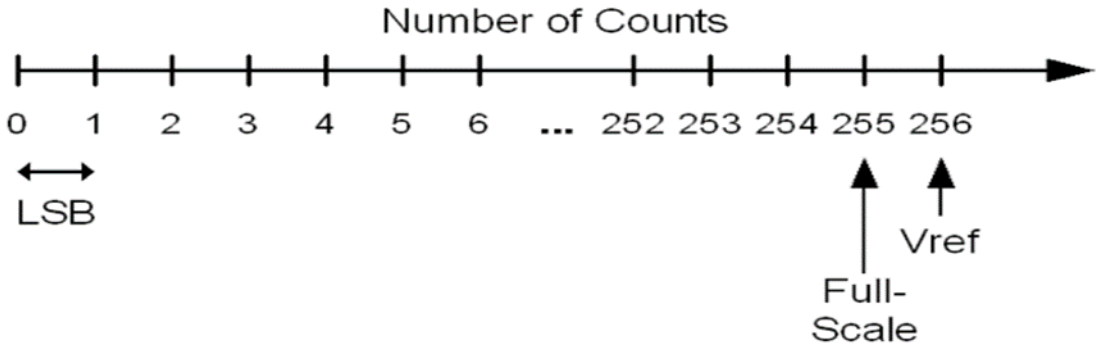

Рисунок 2.1 – Масштабний коефіцієнт

Специфікації гіроскопів та акселерометрів наведені у табл. 2.1–2.2.

Таблиця 2.1 – Специфікації гіроскопів

Параметри	**Умови**	**Мін. знач.**	**Тип**	**Макс. знач**	**Од. виміру**
Повномасштабний діапазон	FS_SEL=0		±250		∘s
	FS_SEL=1		±500		
	FS_SEL=2		±1000		
	FS_SEL=3		±2000		
Довжина слова аналогового та цифрового перетворення гіроскопа			16		біти
Масштабний коефіцієнт чутливості	FS_SEL=0		131		LSB/(0 c)
	FS_SEL=1		65,5		
	FS_SEL=2		32,8		

Параметри	Умови	Мін. знач.	Тип	Макс. знач	Од. виміру
	FS_SEL=3		16,4		
Толерантність коефіцієнта шкали чутливості	25°C		±3		%
Зміна масштабного коефіцієнта чутливості залежно від температури	–40°C до +85°C		±4		%
Нелінійність	Найкраще підходить пряма лінія; 25°C		±0,1		%
Чутливість між осями			±2		%
Початковий вихідний допуск нульової ставки	25°C		±5		∘s
Зміна температури з нульовою швидкістю	–40°C до +85°C		±30		∘s
Загальний середньоквадратичний шум	DLPFCFG = 2 (92 Гц)		0,1		∘s–rms
Спектральна щільність швидкості шуму			0,01		$(\circ s)/\sqrt{\text{Гц}}$
Механічні частоти гіроскопа		25	27	29	кГц
Відгук фільтра низьких частот	Програмований діапазон	5		250	Гц
Час запуску гіроскопа	З режиму сну		35		мс
Швидкість вихідних даних	Програмований, нормальний режим	4		8000	Гц

Таблиця 2.2 – Специфікації акселерометрів

Параметри	Умови	Мін. знач.	Тип	Макс. знач.	Од. виміру
Повномасштабний діапазон	AFS_SEL=0		±2		г
	AFS_SEL=1		±4		
	AFS_SEL=2		±8		
	AFS_SEL=3		±16		

Параметри	Умови	Мін. знач.	Тип	Макс. знач.	Од. виміру
Довжина слова аналогового та цифрового перетворення	Вихідний формат доповнення двох		16		біти
Масштабний коефіцієнт чутливості	AFS_SEL=0		16,384		LSB / г
	AFS_SEL=1		8,192		
	AFS_SEL=2		4,096		
	AFS_SEL=3		2,048		
Початковий доступ	Компонент – рівень		±3		%
Зміна чутливості проти температури	–40°C до +85°C AFS_SEL=0 Компонент – рівень		±0,026		%
Нелінійність	Найкраще підходить пряма лінія		±0,5		%
Чутливість між осями			±2		%
Початковий допуск калібрування нульової перегрузки	Компонент – рівень, X,Y		±60		мг
	Компонент – рівень, Z		±80		
Зміна нульового рівня G проти температури	–40°C до +85°C		±1,5		мг
Загальний середньоквадратичний шум	DLPFCFG = 2 (94 Гц)			8	мг-rms
Спектральна щільність потужності шуму	Режим низького рівня шуму		300		$(\circ s)/\sqrt{\text{Гц}}$
Відгук фільтра низьких частот	Програмований діапазон	5		260	Гц
Збільшення функції інформації			4		мг/LSB
Час запуску акселерометра	З режиму сну		20		мс
	З холодного старту, 1ms Vdd спад		30		
Швидкість вихідних даних	Низька потужність (з робочим	0,24		500	Гц

Параметри	Умови	Мін. знач.	Тип	Макс. знач.	Од. виміру
	циклом)				
	Робочий цикл		±15		%
	Низький рівень шуму (активний)	4		4000	Гц

Значення Fs у попередніх розрахунках було взято з таблиць специфікацій гіроскопів та акселерометрів.

2.5 Роздільна здатність

Мінімальна величина, яку достовірно бачить датчик, вкрай важлива при спробі дотримати балансу між ціною і продуктивністю. Так само як і сенсор з малою роздільною здатністю в певних областях може мати достатню точність. На жаль, LSB визначає лише теоретичне мінімально-розрізнене значення за умови, що можна використовувати всі 16 біт АЦП. Це роздільна здатність у цифровому світі.

У аналоговому кількість ефективних біт буде менше. Джерела шуму можна загалом розбити на електронний шум схеми, що перетворює рух на сигнал напруги (джонсонівський тепловий шум, дробовий шум, рожевий *1/f* фліккер-шум тощо), та тепловий механічний (броуновський, обумовлений наявністю дрібних рухомих частин) від самого детектора. Характеристики останнього залежатимуть від резонансної частоти механічної частини системи f_0 – власної частоти коливань сенсора:

$$\omega_0 = 2\pi/f_0. \tag{2.4}$$

Рівні шуму можна визначати кількома способами. Можна розглядати їх у часовій або частотній області (після перетворення Фур'є). У першому випадку беруть залишковий шум як середньоквадратичне значення сигналів від нерухомого датчика за деякий проміжок часу:

$$x_{RMS} = \sigma_X = \sqrt{\frac{\sum(x_i - \overline{x})^2}{n-1}}. \quad (2.5)$$

Прискорення або кутові швидкості обертання менше рівня широкосмугового шуму будуть невиразні. Середньоквадратичне значення змінної напруги або струму (часто називається діючим або ефективним) дорівнює величині постійного сигналу, дія якого здійснить таку ж роботу в активному (резистивному) навантаженні за період. Найбільш ефективним є підхід при оцінці широкосмугового шуму, де домінує «білий шум».

Для білого шуму відношення амплітуди (миттєвого пікового значення) до середньоквадратчного з ймовірністю 99,9 % становить:

$$N_{PP}/N_{RMS} = 6.6. \quad (2.6)$$

Називається таке відношення хрест-фактором (рис. 2.3). Можна вибрати ймовірність 95,5 % – хрест-фактор дорівнюватиме 4.

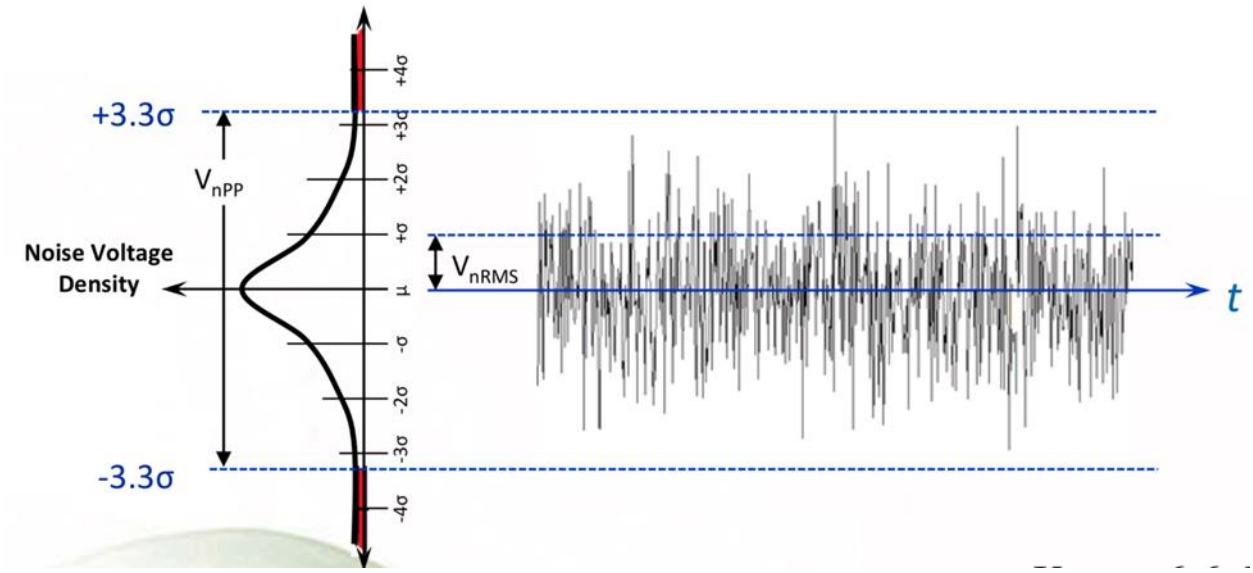

Рисунок 2.3 – Хрест-факторне відношення

Насправді ж сигнали шуму не такі ідеальні і можуть видавати піки, що збільшують хрест-фактор до 10 разів. У деяких специфікаціях можна знайти N_{PP} або сам множник.

У вузькій низькочастотній смузі 0,1–10 Гц основну роль грає фліккер-шум «*1/f*», для оцінки якого використовують значення розмаху шумового сигналу (peak-to-peak).

2.6 Спектральна щільність

Іноді сигнал зручніше розглядати у частотній області, де його опис називається спектром (залежність амплітуди та фази від частоти). Одна з можливих характеристик шуму в специфікаціях зветься спектральна щільність потужності шуму, спектральна щільність шуму, щільність потужності шуму, або просто щільність шуму) Описує розподіл потужності шуму діапазоном частот. Незалежно від представлення електричного сигналу через струм або напругу миттєву потужність, що розсіюється на навантаженні, можна нормувати (R = 1 Ом) і виразити її як:

$$p(t) = v^2\,(t)/R = i^2(t)R = x^2(t). \qquad (2.7)$$

Середня потужність, що розсіюється сигналом протягом проміжку часу (–T/2, T/2):

$$P_x^T = \frac{1}{T}\int_{-T/2}^{T/2} x^2(t)dt. \qquad (2.8)$$

Потужність – швидкість надходження енергії. Через енергію визначаються детерміновані та неперіодичні сигнали. Періодичні та випадкові сигнали виражаються через потужність, оскільки вони не обмежені за часом і, відповідно, енергії, при цьому в будь-який момент часу їхня середня потужність відмінна від нуля:

$$P_x = lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2} x^2(t)dt. \quad (2.9)$$

Можна згадати, що довільний періодичний сигнал виражається через комбінацію нескінченного числа гармонік із зростаючими частотами:

$$x(\lambda) = \frac{a_0}{2} + \sum_{n-1}^{\infty}(a_n cosn\lambda + b_n sinn\lambda), \quad (2.10)$$

що після подання косинуса та синуса в експоненційній формі:

$$cos\lambda = \frac{e^{i\lambda}+e^{-i\lambda}}{2}, sin\lambda = \frac{e^{i\lambda}-e^{-i\lambda}}{2i}, \quad (2.11)$$

і заміни $\lambda = \omega t = 2\pi f_0 t = \frac{2\pi t}{T_0}$ можна записати у вигляді

$$x(t) = \frac{a_0}{2} + \frac{1}{2}\sum_{n-1}^{\infty}\left[(a_n - ib_n)e^{in\omega t} + (a_n + ib_n)e^{-in\omega t}\right] = \sum_{n-1}^{\infty} c_n e^{in\omega t}, \quad (2.12)$$

де комплексні коефіцієнти (спектральні компоненти) ряду Фур'є для $x(t)$,

$$c_n = \frac{1}{T_0}\int_{-T_0/2}^{T_0/2} e^{-in\omega t}, dt = \begin{cases}\frac{1}{2}(a_n - ib_n), n > 0\\ \frac{a_0}{2}, n = 0\\ \frac{1}{2}(a_n + ib_n), n < 0\end{cases} \quad (2.13)$$

В загальному випадку ці коефіцієнти є такими:

$$c_n = |c_n|e^{i\theta_n}, \quad (2.14)$$

$$|c_n| = \frac{1}{2}\sqrt{a_n^2 + b_n^2}, \theta_n = arctan\left(\frac{b_n}{a_n}\right), b_0 = 0, c_0 = \frac{a_0}{2}. \quad (2.15)$$

Амплітудним та фазовим спектром називають графіки залежності $|c_n|$ та θ_n від частоти. Спектральна щільність потужності PSD(f) періодичного сигналу x(t) дає розподіл потужності сигналу за діапазоном частот:

$$PSD(f) = \sum_{n=-\infty}^{\infty} |c_n|^2 \delta(f - nf_0), \quad (2.16)$$

і має розмірність $[W/\text{Гц}] = [x^2/\text{Гц}]$. Середня нормована потужність дійсного сигналу складає:

$$P_x = \int_{-\infty}^{\infty} PSD(f), df. \quad (2.17)$$

Неперіодичні випадкові сигнали (зокрема шум) можна описати як періодичні в граничному сенсі. Якщо T_0 прагне нескінченності, послідовність імпульсів перетворюється на окремий імпульс x(t), число спектральних ліній прагне нескінченності, графік спектра перетворюється на гладкий спектр частот X(f). Для цього граничного випадку можна визначити пару інтегральних перетворень Фур'є

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft}, dt \quad (2.18)$$

та

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{i2\pi ft}, df, \quad (2.19)$$

де X(f) – Фур'є-образ.

Спектральна щільність потужності випадкового сигналу визначається через межу

$$PSD(f) = lim_{T\to\infty} \frac{1}{T} |X_T(f)|^2. \quad (2.20)$$

і описує розподіл потужності сигналу в діапазоні частот.

Оскільки припускаємо, що середнє для білого шуму датчиків у нерухомому стані дорівнює нулю ($\overline{x} = \overline{x^2} = 0$), то квадрат середньоквадратичного значення дорівнює дисперсії і є повною потужністю в нормованому навантаженні:

$$N_{RMS}^2 = \sigma_x^2 = P_x = \int_0^\infty PSD(f)\,, df = \int_0^{BW} PSD(f), df = PSD(BW - 0). \quad (2.21)$$

Дивимося у специфікації – там під ім'ям спектральної щільності вказаний квадратний корінь із неї з відповідною розмірністю $\left[\circ\, s/\sqrt{\text{Гц}}\right]$ або $\left[\text{мкг}\sqrt{\text{Гц}}\right]$. Тобто значення RMS шуму без зазначення смуги частот, де він враховувався.

На виході MEMS датчика отримуємо сигнали різної частоти. Передбачається, що заздалегідь маємо деяке уявлення про процеси, що вимірюються. Наприклад, щодо вектора прискорення дрону шумом є вібрації апарату. Відокремити їх від корисного сигналу можна за допомогою фільтра низьких частот, який обріже всі частоти вище вказаної (наприклад, 200 Гц).

MPU-9250 надає можливість налаштувати частоту зрізу фільтра низьких частот за допомогою параметра Digital Low Pass Filter Configuration (DLPFCFG). Реєстраційна карта для гіроскопа та акселерометра зображена на рис. 2.4.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
00	0	SELF_TEST_X_GYRO	R/W	xg_st_data [7:0]							
01	1	SELF_TEST_Y_GYRO	R/W	yg_st_data [7:0]							
02	2	SELF_TEST_Z_GYRO	R/W	zg_st_data [7:0]							
0D	13	SELF_TEST_X_ACCEL	R/W	XA_ST_DATA [7:0]							
0E	14	SELF_TEST_Y_ACCEL	R/W	YA_ST_DATA [7:0]							
0F	15	SELF_TEST_Z_ACCEL	R/W	ZA_ST_DATA [7:0]							
13	19	XG_OFFSET_H	R/W	X_OFFS_USR [15:8]							
14	20	XG_OFFSET_L	R/W	X_OFFS_USR [7:0]							
15	21	YG_OFFSET_H	R/W	Y_OFFS_USR [15:8]							
16	22	YG_OFFSET_L	R/W	Y_OFFS_USR [7:0]							
17	23	ZG_OFFSET_H	R/W	Z_OFFS_USR [15:8]							
18	24	ZG_OFFSET_L	R/W	Z_OFFS_USR [7:0]							
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	FIFO_MODE	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	XGYRO_Cten	YGYRO_Cten	ZGYRO_Cten	GYRO_FS_SEL [1:0]		-	FCHOICE_B[1:0]	
1C	28	ACCEL_CONFIG	R/W	ax_st_en	ay_st_en	az_st_en	ACCEL_FS_SEL[1:0]		-		
1D	29	ACCEL_CONFIG 2	R/W	-				ACCEL_FCHOICE_B	A_DLPF_CFG		

Рисунок 2.4 – Реєстраційна карта для гіроскопа та акселерометра

У цьому датчику забезпечується фільтрація показань не тільки акселерометрів, гіроскопів, але і температурного датчика. Для кожного існує в загалом від 7 до 10 режимів, що характеризуються такими поняттями, як смуга пропускання в Гц, затримка в мс, частота дискретизації (частота дискретизації, Fs) в кГц. У таблиці режимів фільтра акселерометра (рис. 2.5) додана колонка «Щільність шуму» в мкг$/rt$ Гц $=$ мкг$/\sqrt{\text{Гц}}$.

BIT	NAME	FUNCTION
[7:6]	Reserved	
[5:4]	Reserved	
[3]	accel_fchoice_b	Used to bypass DLPF as shown in table 2 below. NOTE: This register contains accel_fchoice_b (the inverted version of accel_fchoice as described in the table below).
[2:0]	A_DLPFCFG	Accelerometer low pass filter setting as shown in table 2 below.

Accelerometer Data Rates and Bandwidths (Normal Mode)

ACCEL_FCHOICE	A_DLPF_CFG	Output				
		3dB BW (Hz)	Rate (kHz)	Filter Block	Delay (ms)	Noise Density (µg/rtHz)
0	x	1,046	4	Dec1	0.503	300
1	0	218.1	1	DLPF	1.88	300
1	1	218.1	1	DLPF	1.88	300
1	2	99	1	DLPF	2.88	300
1	3	44.8	1	DLPF	4.88	300
1	4	21.2	1	DLPF	8.87	300
1	5	10.2	1	DLPF	16.83	300
1	6	5.05	1	DLPF	32.48	300
1	7	420	1	Dec2	1.38	300

Рисунок 2.5 – Режими фільтра акселерометра

Частота дискретизації – це кількість взятих за секунду точок постійного по часу сигналу при його дискретизації АЦП, вимірюється у Гц:

$$F_s = \frac{1}{\delta t}. \quad (2.22)$$

Для того, щоб у виборі виявилося значення, наближене до пікової амплітуди сигналу, важливо брати частоту дискретизації мінімум у 10 разів більше частоти корисного сигналу. MPU-9250 пропонує три варіанти: Fs = 32 кГц, 8 кГц, 1 кГц. Але це абсолютно не означає, що сигнал на виході акселерометра або гіроскопа з'являється з тим же періодом.

Якщо взяти саме дрони, тут все зводиться до боротьби за зниження енергоспоживання, підвищення швидкості обчислень і зниження шуму вихідних даних. Можна знизити частоту оновлення даних на виході,

дозволивши внутрішнім алгоритмам інтегрувати інформацію протягом іншого періоду часу. Середньоквадратичний шум знижується, але також звузиться і смуга пропускання (датчик зможе задіяти лише ті процеси, частота яких буде менше 50 % швидкості оновлення даних).

Для Fs = 32 кГц частота Найквіста буде 16 кГц. При цьому корисний сигнал вряд чи вийде за смугою Fa = 20 Гц (мало хто може змінити напрямок руху частіше 20 разів у секунду). У цьому, частота дискретизації значно перевищує частоту, необхідну для збереження інформації, що міститься в смузі Fa (40 Гц, в 400 разів вище). Полоса між частотами Fa і Fs-Fa не містить ніякої корисної інформації. Можна зменшити частоту дискретизації (на рис. 2.6 це зроблено з коефіцієнтом M). Цей процес і називається децимацією.

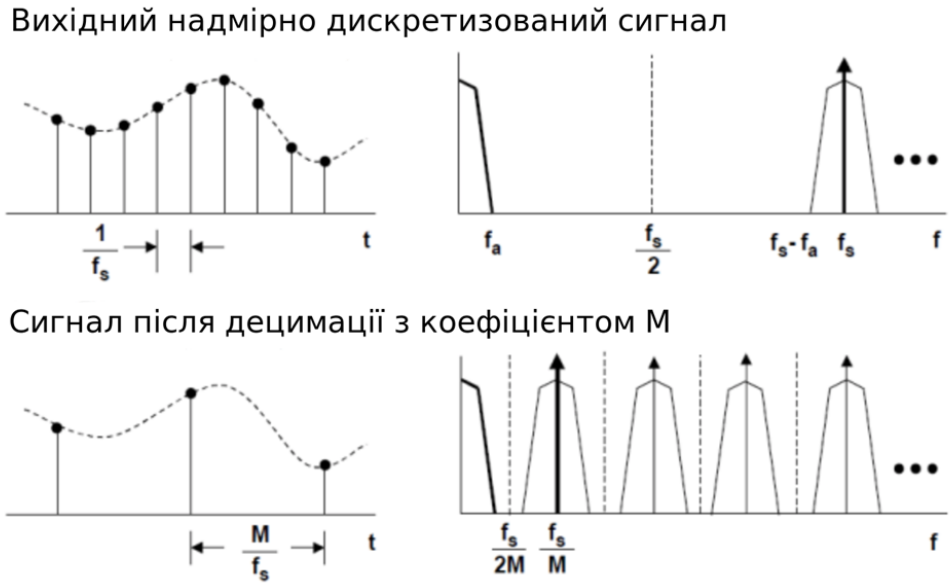

Рисунок 2.6 – Сигнал до та після децимації [7]

Відповідно до специфікації на MPU-9250, акселерометри забезпечені сигма-дельта АЦП. Схеми з його основі споживають мінімальну потужність. Слід зазначити, що смуга пропускання у даних перетворювачів дуже вузька, не перевищує звукового діапазону, але для штатного квадрокоптера більше і не потрібно. Складаються вони з двох блоків: $\sum\Delta$-модулятора і цифрового децимуючого фільтра низьких частот.

Якщо вихідний сигнал не містить частот, що перевищують частоту Найквіста децимованого сигналу, форма спектра отриманого (децимованого) сигналу збігається з низькочастотною частиною спектра вихідного сигналу. Частота дискретизації, що відповідає новій послідовності відліків, у N разів нижче, ніж частота дискретизації вихідного сигналу.

Якщо вихідний сигнал містить частоти, що перевищують частоту Найквіста децимованого сигналу, то при децимації буде місце аліасинг (накладення спектрів).

Таким чином, для збереження спектра необхідно до децимації видалити вихідний сигнал частоти, що перевищують частоту Найквіста децимованого сигналу.

2.7 Частотний відгук

Діапазон частот, у якому датчик виявляє рух та видає дійсний вихідний сигнал. У деяких специфікаціях наводиться частотна характеристика датчика – залежність електричного вихідного сигналу акселерометра від зовнішніх механічних впливів із фіксованою амплітудою, але різними частотами. У межах смуги пропускання нерівномірність частотної характеристики вбирається у заданої. У разі застосування цифрового фільтра низьких частот вибір смуги пропускання дозволяє змінювати частоту зрізу, неминуче впливаючи на швидкість відгуку датчика на зміни положення в просторі. Частота зрізу має бути менше половини швидкості оцифровки, званої також частотою Найквіста.

Для акселерометрів MPU-9250 межі смуги пропускання визначаються так, щоб усередині діапазону спектральна щільність сигналу відрізнялася від пікової (на частоті 0 Гц) не більше ніж на –3 дБ. Цей рівень приблизно відповідають падінню до половини спектральної густини (або 70.7 % від пікової спектральної амплітуди). Для енергетичних величин (потужність,

енергія, щільність енергії), пропорційних квадратам силових величин поля, виражене в децибелах відношення:

$$D_P = 10lg\frac{P_2}{P_1}. \quad (2.23)$$

Підсумок: сигнали, що пройшли через ФНЧ, менш зашумлені, у них найкраща роздільна здатність, але при цьому менша смуга пропускання. Тому зазначення дозволу специфікації без прив'язки до смуги пропускання сенсу немає.

Оцінити роздільну здатність для кожної смуги пропускання можна за піковим шумом:

$$N_{pk-pk} = TotalN_{RMS} \cdot CrestFactor = TotalN_{RMS} \cdot 4. \quad (2.24)$$

Середньоквадратична величина шуму на виході пов'язана із зазначеною у специфікації спектральною щільністю (а точніше, коренем з неї) і еквівалентною шумовою смугою пропускання (ENBW) – смуга пропускання еквівалентної системи, що має прямокутну АЧХ і однакові з вихідною системою значення на нульовій частоті та дисперсію на виході, при впливі на входи систем білого шуму):

$$N_{RMS} = PSD\sqrt{ENBW}. \quad (2.25)$$

А шумова смуга пропускання пов'язана з 3 дБ смужкою коефіцієнтами, що відповідають порядку низькочастотного фільтра:

$$ENBW = 1{,}57 \cdot f_{3\text{дБ}} \text{ для 1-го порядку,} \quad (2.26)$$

$$ENBW = 1{,}11 \cdot f_{3\text{дБ}} \text{ для 2-го порядку,} \quad (2.27)$$

$$ENBW = 1{,}05 \cdot f_{3дБ} \text{ для 3-го порядку,} \quad (2.28)$$

$$ENBW = 1{,}025 \cdot f_{3дБ} \text{ для 4-го порядку.} \quad (2.29)$$

Отримане середньоквадратичне значення враховує вклад білого шуму (ні шуму квантування, ні механічного шуму немає). Наприклад, для акселерометра розрахункове значення для BW=99 Гц, PSD=300 мкг/$\sqrt{\text{Гц}}$ виходить N_{RNS} = 4 мг. При цьому в специфікації окремо зазначений повний середньоквадратичний шум (табл. 2.3).

З необхідності зберігати у внутрішньому буфері змінні для розподілу фільтром сигналу різні частоти.

Таблиця 2.3 – Порівняння MPU-9250 та BMI088

MPU-9250			**BMI088**		
Гіроскоп					
$N_{RMS}^{Total} = 0{,}1^{\circ}/s\ (BW = 92\text{Гц})$			$N_{RNS} = 0{,}1^{\circ}/s\ (BW = 47\text{Гц})$		
$PSD = 0{,}01^{\circ}/s/\sqrt{\text{Гц}}$			$PSD = 0{,}014^{\circ}/s/\sqrt{\text{Гц}}$		
BW, Гц	N_{RNS},°/s – rms	N_{PP},°/s	BW, Гц	N_{RNS},°/s	N_{PP},°/s
			523	0,41	1,6
250	0,2	0,8	230	0,27	1,1
184	0,17	0,69	116	0,19	0,76
92	0,12	0,49	64	0,14	0,57
41	0,08	0,32	47	0,12	0,49
20	0,06	0,23	32	0,1	0,4
10	0,04	0,16	23	0,09	0,34
5	0,03	0,11	12	0,06	0,25

Акселерометр					
$N_{RMS}^{Total} = 8\text{мг}(BW = 99\text{Гц})$			$PSD_{XY} = 160\ \text{мкг}/\sqrt{\text{Гц}}$		
$PSD = 300\ \text{мкг}/\sqrt{\text{Гц}}$			$PSD_Z = 190\ \text{мкг}/\sqrt{\text{Гц}}$		
BW, Гц	N_{RNS}, мг	N_{PP}, мг	BW, Гц	N_{RNS}, мг	N_{PP}, мг
218,1	5,6	22	280	3,4	14
99	3,8	15	145	2,4	10
44,8	2,5	10	80	1,8	7
21,2	1,7	7	40	1,3	5

MPU-9250			**BMI088**		
10,2	1,2	4,9	20	0,9	4
5,05	0,9	3,4	10	0,6	2,6
420	7,8	31	5	0,5	1,8
1046	12,3	49			

Загалом, чим нижче частота обрізання фільтра, тим менше шуму сигналу. Але треба зауважити, що водночас виростає й затримка. З іншого боку, можна пропустити корисний сигнал (табл. 2.4).

Таблиця 2.4 – Порівняння MPU-9250 та BMI088 [8]

MPU-9250		**BMI088**	
Гіроскоп, 16 біт			
Діапазон (Fs), °/s	Роздільна здатність, біт (BW = 92Гц)	Діапазон (Fs), °/s	Роздільна здатність, біт (BW = 64Гц)
		±125	8
±250	9	±250	9
±500	10	±500	10
±1000	11	±1000	11
±2000	12	±2000	12
Акселерометр			
Діапазон (Fs), г	Роздільна здатність, біт N_{PP}=32мг °/s	Діапазон (Fs), г	Роздільна здатність (по X,Y), біт N_{PP} = 14мг
±2	6	±3	8
±4	7	±6	9
±8	8	±12	10
±16	9	±24	11

Як можна побачити, модуль MPU-9250 має більшу роздільну здатність ніж BMI088 за багатьма параметрами. Але це все одно но виключає обробки шуму на виході.

Висновки до розділу 2

У розділі 2 було розглянуто математичну модель роботи MEMS акселерометрів, гіроскопів та магнітометрів, MEMS акселерометри використовують для кожної осі окрему пробну масу, яка зміщується у разі прискорення вздовж даної осі.

MEMS гіроскоп складається з трьох незалежних одновісних вібраційних датчиків кутової швидкості, які реагують на обертання навколо X-, Y-, Z- осей. Дві підвішені маси чинять коливання по протилежним осям.

MEMS магнітометр заснований на високоточній технології ефекту Холла. Включає магнітні сенсори, що визначають напруженість магнітного поля землі по осях, схему управління, ланцюг посилення сигналу і обчислювальну схему для обробки сигналів з кожного датчика.

В результаті розділу 2 було визначено, що середньоквадратична величина шуму на виході пов'язана із зазначеною у специфікації спектральною щільністю, а точніше, коренем з неї, і еквівалентною шумовою смугою пропускання. Отримані дані знадобляться при роботі з модулем та його налаштуванні.

3 КОНСТРУЮВАННЯ АПАРАТНО-ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Raspberry Pi Pico

Raspberry Pi Pico – це недорога, високопродуктивна плата мікроконтролера з гнучкими цифровими інтерфейсами, побудована на кремнієвому процесорі, розробленому Raspberry Pi. Це перша мікроконтролерна плата Raspberry Pi, розроблена спеціально для фізичних обчислень.

Мікроконтролери – інший тип пристроїв, ніж одноплатні комп’ютери (як-от Raspberry Pi 4 і попередні покоління Pi), вони не працюють під управлінням операційної системи, і зазвичай вони запрограмовані на виконання лише одного завдання, хоча це завдання може бути досить складним. Вони ідеально підходять для експериментів із апаратним забезпеченням і використання їх як мозок нестандартних пристроїв, машин і винаходів.

Raspberry Pi Pico є платою із мікроконтролером RP2040.

Треба пам'ятати, що це не комп'ютер, на якому запущена доросла ОС типу Linux, а саме мікроконтролер і тому цілі застосування Pico відрізняються від тієї ж Raspberry Pi Zero та інших старших продуктів сімейства Raspberry Pi.

Технічні характеристики Raspberry Pi Pico:

– два ядра Arm Cortex-M0+ @ 133 МГц;

– 264 кбайт пам'яті (284 кбайт якщо вимкнути XIP кешування та використовувати пам'ять USB);

– 2 Мбайт флеш-пам'ять із XIP кешуванням. У RP2040 немає вбудованої флеш-пам'яті, тому чіп розпаяний на платі. RP2040 має підтримку до 16 Мбайт зовнішньої флеш-пам'яті;

– DMA контролер;

– 4 × 12-розрядних аналогових входу (на Pico доступно для користувача 3 з них);
– 2 × UART;
– 2 × SPI;
– 2 × I2C;
– 16 × PWM каналів;
– вбудований сенсор температури;
– усього 30 GPIO пінів (3,3 Вольта);
– MicroUSB B порт з USB 1.1 контролером та підтримкою хоста;
– 2 × PIO блоки для власних інтерфейсів;
– 2 × PLL (один для USB, другий для решти);
– підтримка UF2 для завантаження бінарників;
– підтримка SWD для завантаження та налагодження;
– підтримка режимів сну та зниженої частоти для зниження споживання.

Плата має зручний розмір 21 мм × 51 мм. Також має отвори для монтування, чим може похвалитися не кожна схожа плата. Можна припаяти піни для використання з макеткою або запаяти весь модуль поверхневим монтажем на іншу плату.

На Raspberry Pi Pico (рис. 3.1) стоїть знижуючий перетворювач на 3,3 Вольта. Це не просто лінійний перетворювач, які часто зустрічаються на недорогих аналогічних платах, а імпульсний стабілізатор напруги (SMPS) на Richtek RT6150B. Завдяки цьому вхідне харчування плати може бути в межах 1,8 – 5,5 Вольт.

Флеш-пам'ять W25Q16JV, хоч, і стоїть зовнішня, але перепаювати її буде не таким вже й тривіальним завданням, тому що чіп у корпусі USON-8. Є кнопка BOOTSEL та світлодіод на GPIO25. Також виведено SWD для налагодження.

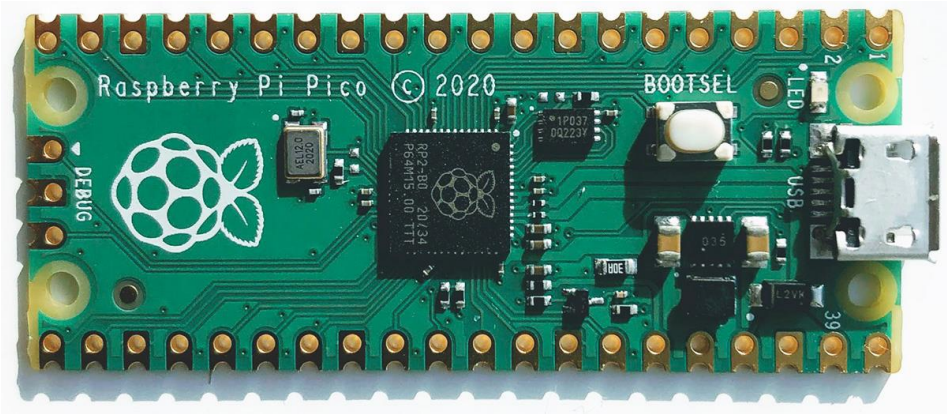

Рисунок 3.1 – Raspberry Pi Pico

На Raspberry Pi Pico виведено майже всі вхідні піни (26 з 30). На рис. 3.2 наведено розпінування плати.

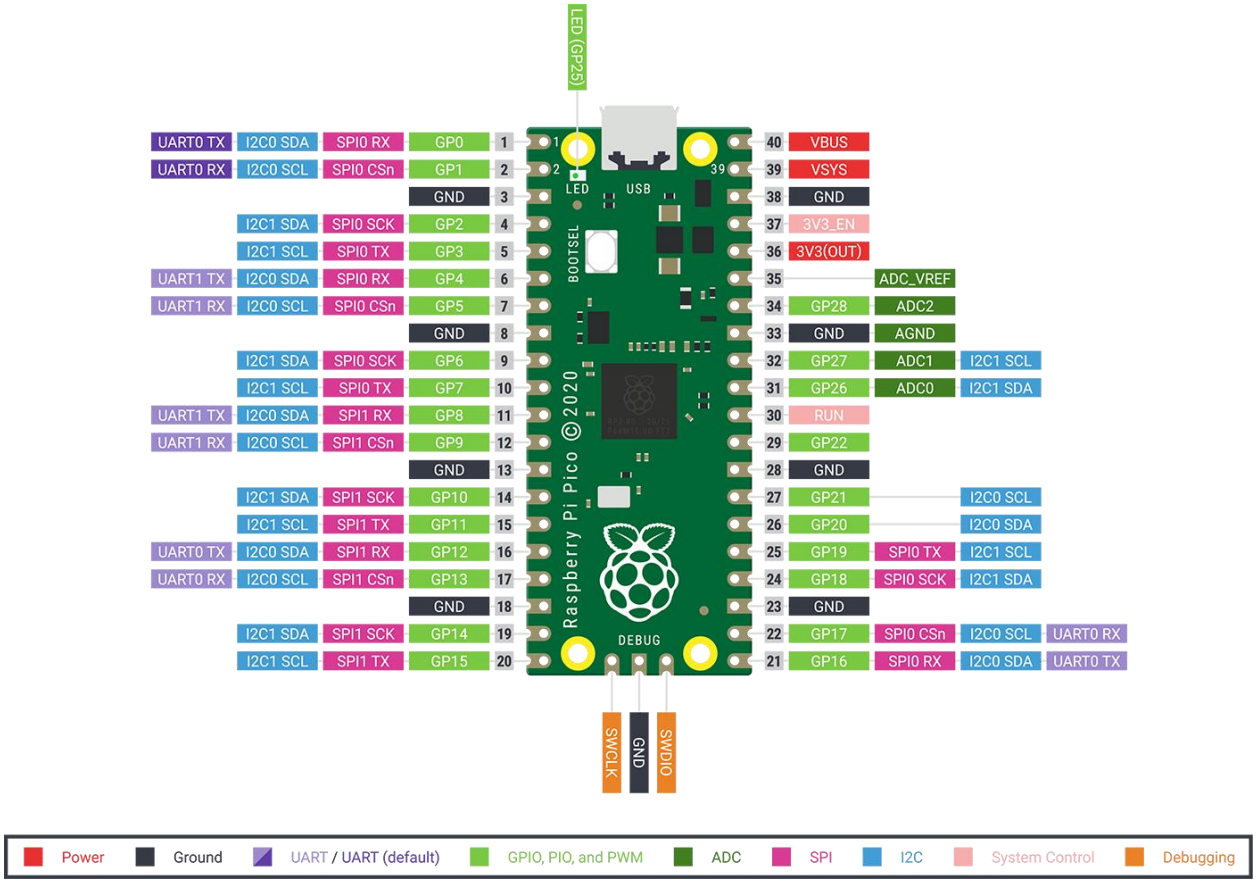

Рисунок 3.2 – Розпінування плати Raspberry Pi Pico

Деякі піни задіяні для внутрішнього застосування:

– GPIO23: вихід для контролю за енергозбереженням SMPS. Можна регулювати пульсації за рахунок зміни ККД перетворювача;

– GPIO24: вхід для VBUS sense (1 якщо VBUS по MicroUSB підключено);

– GPIO25: вихід на світлодіод, розташований на платі;

– GPIO29: аналоговий вхід для вимірювання VSYS/3.

Сам USB-порт додатково виведений на точки TP1, TP2 та TP3 внизу плати.

3.2 Розробка забезпечення на Raspberry Pi Pico

На даний момент офіційно пропонуються наступні варіанти для розробки під мікроконтролер RP2040:

– C/C++ з використанням пропонованого Pico SDK;

– CircuitPython для Pico;

– MicroPython для Pico.

Мікроконтролер RP2040 має вбудований завантажувач, що підтримує UF2 (розробка Microsoft) для завантаження бінарників. Це являє собою зовнішній USB-накопичувач, на який можна просто скопіювати бінарник.

При роботі з новою платою у флеш-пам'яті нічого не буде і UF2 активується автоматично при підключенні по USB. Коли програма записана на флеш-пам'яті, режим UF2 можна активувати утримуванням кнопки BOOTSEL при подачі живлення по USB. З'явиться накопичувач RPI-RP2, який можна використовувати для копіювання бінарних файлів UF2.

Для налагодження можна використовувати SWD. Якщо немає відповідного налагоджувача, то можна використовувати ще одну плату Pico з прошитим відладчиком.

Як більш простий варіант Pico може виводити дані стандартного виводу на UART або прикидатися USB CDC і виводити в звичайний термінал типу PuTTY, minicom або аналогічний параметрів за замовчуванням 115200 8n1.

3.2.1 C/C++ з Pico SDK

В оригінальному документі докладно описані кроки для встановлення SDK на Linux, macOS і Windows.

Варіант із розробкою на Raspberry Pi 4B або 400 з Linux буде найпростішим, оскільки є скрипти, які зроблять початкову конфігурацію (навіть встановлення Visual Studio Code):

```
git clone https://github.com/raspberrypi/pico-setup.git
pico-setup/pico_setup.sh
```

Весь процес початкової підготовки зводиться до наступних кроків (у цьому випадку кроки виконувались на macOS та Linux):

```
# Створити загальну директорію для всього:
mkdir pico && cd pico
# Забрати Pico SDK:
git clone --recursive https://github.com/raspberrypi/pico-sdk.git
# Забрати приклади:
git clone https://github.com/raspberrypi/pico-examples.git
# Linux: встановити необхідні інструменти для збирання через "apt":
apt update && apt install cmake gcc-arm-none-eabi build-essential
# OSX: встановити інструменти за допомогою "brew" (список може трохи змінюватися в залежності від поточних встановлених пакетів):
# (gcc-arm-embedded буде встановлено в /usr/local/bin. Ця інформація буде потрібна при конфігуруванні Visual Studio Code)
brew install cmake gcc-arm-embedded
# Налаштувати змінну PICO_SDK_PATH (можна занести значення у щось типу .profile за смаком):
export PICO_SDK_PATH=`pwd`/pico-sdk
```

Можна спробувати тестову збірку миготіння світлодіодом:

```
cd pico-examples
mkdir build
cd build
cmake ..
cd blink
make -j8
```

Приклад коду для налаштування миготіння світлодіодом за допомогою Pico SDK:

```
#include "pico/stdlib.h"

int main() {
    // Конфігурація піна зі світлодіодом
    const uint LED_PIN = 25;
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);

    // Наш робочий нескінченний цикл
    while (true) {
        // Переключити світлодіод
        gpio_put(LED_PIN, 1);
        sleep_ms(250);
        gpio_put(LED_PIN, 0);
        sleep_ms(250);
    }
}
```

У разі, якщо помилок немає, в результаті буде отримано файли, серед яких буде blink.uf2. Цей файл можна скопіювати на Pico в режимі UF2 (треба утримувати кнопку BOOTSEL під час подачі живлення USB).

Після копіювання Pico автоматично перезавантажиться. У «pico-examples» є чимало цікавих прикладів, у тому числі приклади з використанням PIO.

Для створення початкового шаблону для проєкту є інструмент від Raspberry Pi. Він створює шаблони під Pico SDK, Visual Studio та додає підтримку різних бібліотек на різну периферію.

3.2.2 CircuitPython

CircuitPython є похідною мовою програмування від MicroPython, але зі своїми особливостями. Найпомітніша особливість в тому, що CircuitPython створює USB флешку зі своєю файловою системою, де можна безпосередньо редагувати скрипти на Python у зручній IDE. За будь-якого запису зміненого скрипта відбувається автоматичний перезапуск плати та виконання коду.

У терміналі можна бачити результат або запустити інтерактивний режим для виконання команд, що теж допомагає у налагодженні.

Установка CircuitPython досить проста:

1. Завантажити файл UF2 файл із CircuitPython на circuitpython.org/board/raspberry_pi_pico.

2. Перевести Pico в режим UF2 утриманням BOOTSEL під час подачі живлення USB.

3. Копіювати файл із #1 на флешку RPI-RP2, після чого Pico перезапуститься автоматично.

Після цього із системи зникне RPI-RP2 і замість нього з'явиться новий накопичувач CIRCUITPY. На цьому новому накопичувачі має бути файл code.py, з якого починається виконання коду. Також там буде порожня директорія «lib», куди можна додавати сторонні та свої бібліотеки.

Файл code.py можна змінювати прямо на цьому накопичувачі у зручному редакторі. Adafruit радить використовувати MU Editor, але у мене він зависає під час запуску.

Тестова збірка на CircuitPyhon виглядає наступним чином:

```
import board
import time
from digitalio import DigitalInOut, Direction
# Конфігурація піна зі світлодіодом
led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT

# Робочий нескінченний цикл
while True:
     # Переключити світлодіод
     led.value = not led.value
     time.sleep(1)
```

Також доступна консоль на послідовному порту з параметрами 115200 8n1. При підключенні можна отримати доступ до інтерактивного Python і виведення в консоль через «print» у скриптах.

Adafruit пропонує досить великий набір бібліотек для роботи з різним обладнанням. Рекомендується забирати .mpy версію. Це готовий байт-код, який відкомпілюваний під потрібну версію CircuitPython.

3.2.3 MicroPython

MicroPython і CircuitPython є досить близькими один до одного, але з деякими особливостями при налагодженні та API.

Інсталяція MicroPython також доволі проста:

1) завантажуємо UF2 зі свіжим релізом з офіційного сайту (вкладка «Getting started with MicroPython»);

2) переводимо Pico у режим UF2 утримуванням BOOTSEL під час подачі живлення USB;

3) копіюємо .uf2 файл з першого кроку на RPI-RP2.

На цей раз RPI-RP2 також зникає, але новий накопичувач не з'являється. Можна підключитися терміналом на порт (115200 8n1), що тільки з'явився, за яким буде доступна інтерактивна консоль Python.

Для роботи з кодом передбачається використання Thonny – інтегрованого середовища розробки для Python (рис. 3.3). Це мінімальний IDE, який може редагувати код безпосередньо на платі мікроконтролера.

Після встановлення Thonny у налаштуваннях треба підключити Pico. Це можна зробити через меню Tools – Options. На вкладці «Interpreter» потрібно вибрати «MicroPython (Raspberry Pi Pico)» у полі «Which interpreter or device», а у полі «Port» вибрати порт, у якому підключена плата Pico (рис. 3.3).

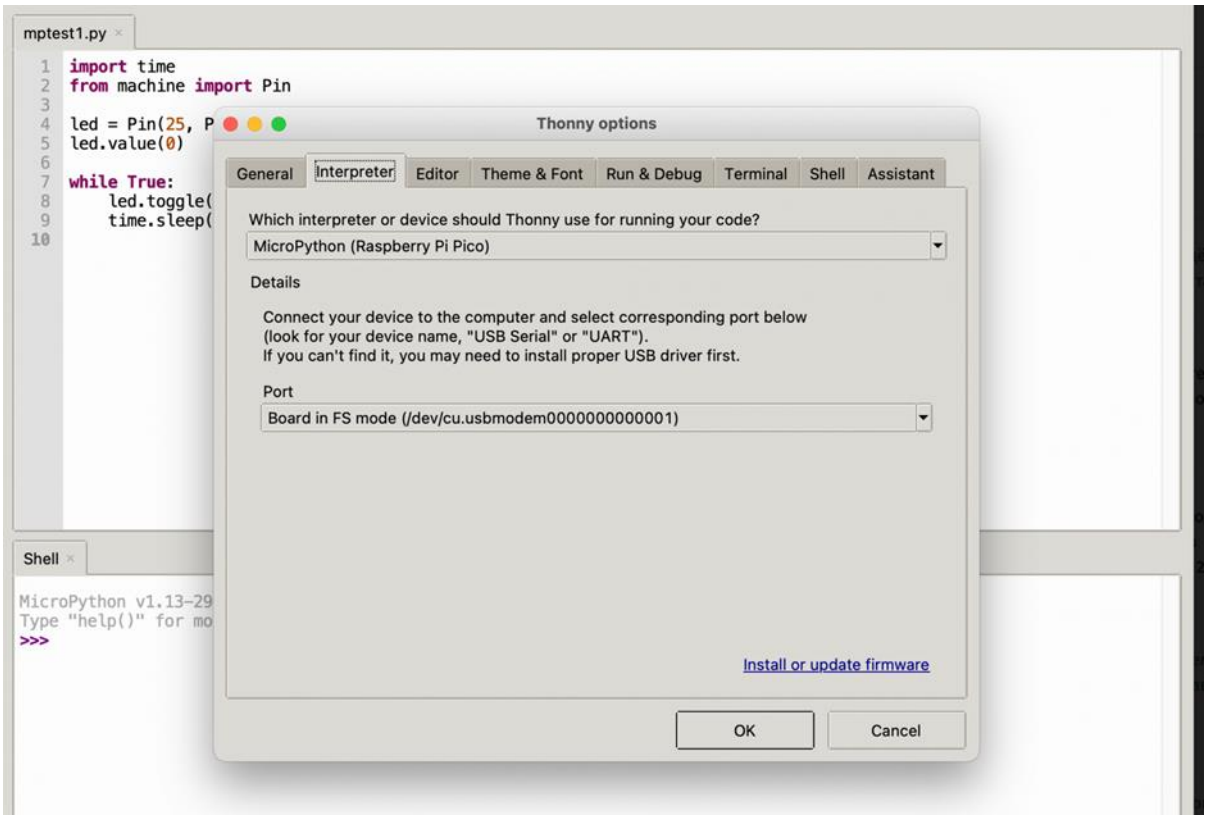

Рисунок 3.3 – Інтегроване середовище розробки Thonny

Налаштування краще виконувати при підключеній Pico, щоб Thonny міг одразу знайти потрібний порт.

Приклад для налаштування миготіння світлодіодом за допомогою MicroPython:

```
import time
from machine import Pin
# Конфігурація піна зі світлодіодом
led = Pin(25, Pin.OUT)
led.value(0)
# Робочий нескінченний цикл
while True:
    # Переключити світлодіод
    led.toggle()
    time.sleep(1)
```

3.3 Налаштування Raspberry Pi для GPS

Багато проєктів вимагають визначення положення Raspberry Pi. Іноді не можна використовувати стільникову мережу або мережу Wi-Fi для цього завдання.

У цьому випадку GPS може стати хорошою альтернативою, особливо якщо Raspberry Pi використовується на вулиці.

Потрібно налаштувати ОС Raspberry Pi, щоб мати можливість спілкуватися з GPS-приймачем. Наступні кроки стосуються саме Raspbian Jessie або новішої версії, і можуть відрізнятися для старих версій.

Починаємо із запуску raspi-config:

```
sudo raspi-config
```

Після цього побачимо екран, наведений на рис. 3.4.

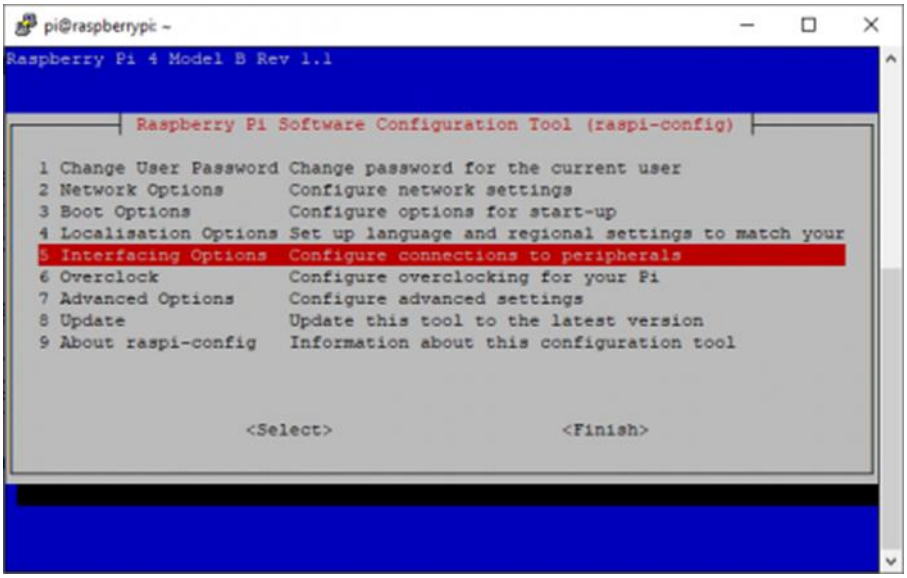

Рисунок 3.4 – Запуск інструменту налаштування програмного забезпечення Raspberry Pi

Далі, у цьому вікні потрібно вибрати «Interfacing Options», а потім пункт «Serial» (рис. 3.5).

Наступним кроком необхідно вимкнути можливість доступу до оболонки входу через послідовне з’єднання. Потім, коли запитають, чи хочете, щоб послідовні порти залишалися ввімкненими, оберіть «Так».

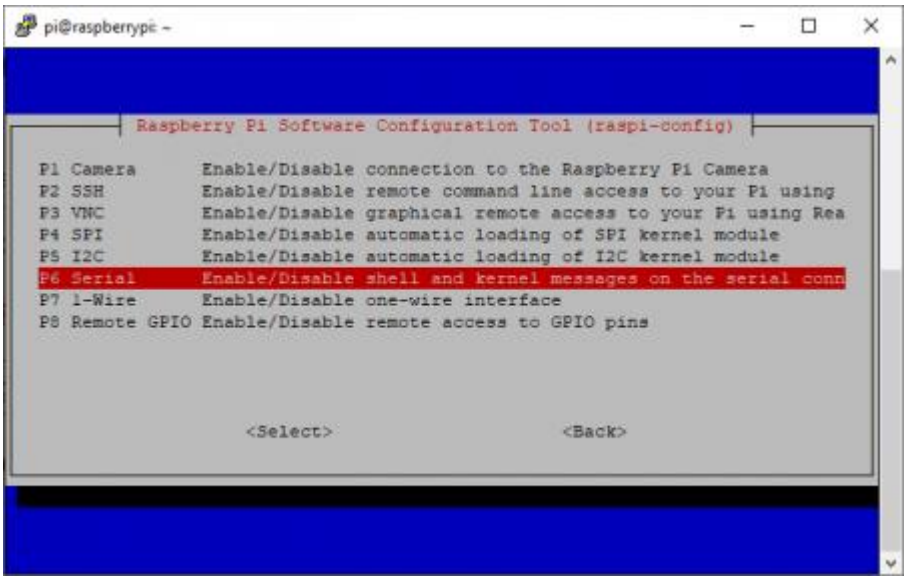

Рисунок 3.5 – Ініціалізація можливості доступу до оболонки входу через послідовне з’єднання

Після повернення до головного меню програми raspi-config, виберіть «Finish» і перезавантажте Raspberry Pi.

3.3.1 Завантаження необхідного програмного забезпечення

Встановлюємо необхідне програмного забезпечення – gpsd і gpsd-клієнт:

а) sudo apt-get install gpsd gpsd-клієнти;

б) gpsd – це демон інтерфейсу для послідовних приймачів GPS, який підтримує різні стандарти зв’язку. Його можна використовувати, щоб отримати тестове зчитування та перевірити, чи правильно працює апаратне забезпечення. Для отримання додаткової інформації про програму можна ввести man gpsd.

Після завершення встановлення переконайтеся, що можна отримувати дані від модуля GPS. Для цього виведіть дані, які він надсилає через послідовний порт:

```
cat /dev/serial0
```

Результат виконання команди для перевірки отримання даних зображено на рис. 3.6.

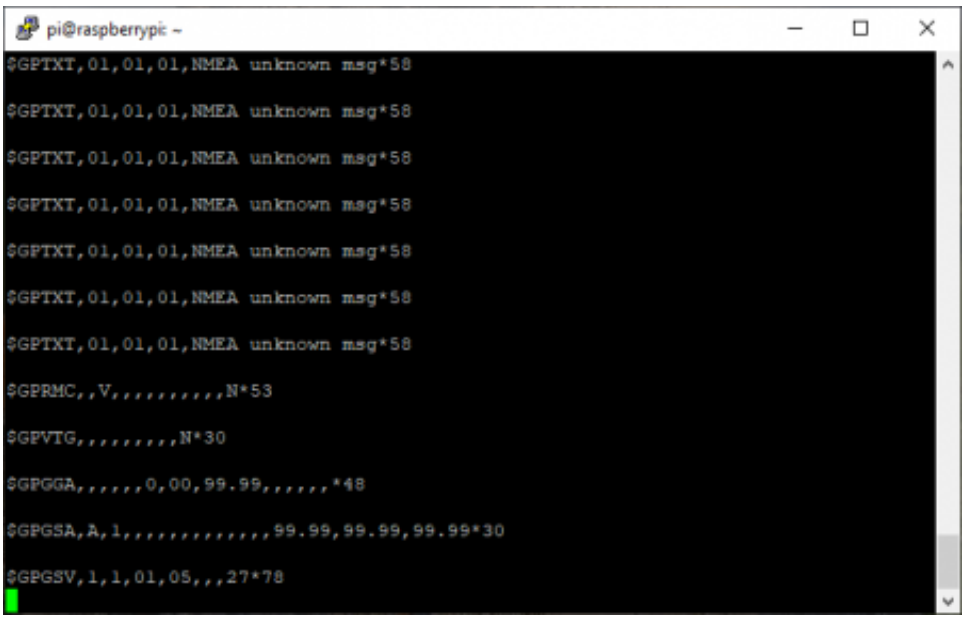

Рисунок 3.6 – Послідовний порт

На даному етапі не важливо, які данні отримуєте. Якщо порт негайно закривається або Raspberry Pi взагалі не отримує жодних даних, переконайтеся, що правильно підключили модуль.

Також повинні мати можливість виконувати цю команду, не будучи суперкористувачем. Якщо не можете, додайте pi-user до групи набору:

```
sudo adduser pi dialout
```

3.3.2 Зчитування даних про місцезнаходження

Тепер настав час визначити позицію Raspberry Pi. Потрібно ввести наступну команду, щоб зупинити службу gpsd, яка запускалася автоматично під час інсталяції gpsd. Це необхідно зробити, оскільки параметри за замовчуванням не є правильними для Raspberry Pi:

```
sudo systemctl stop gpsd.socket
```

Цю команду доведеться вводити кожного разу, коли завантажуєте систему. Але також можна повністю зупинити службу gpsd:

```
sudo systemctl disable gpsd.socket
```

Тепер потрібно запустити новий екземпляр gpsd, який перенаправляє дані правильного послідовного порту до сокета:

```
sudo gpsd /dev/serial0 -F /var/run/gpsd.sock
```

Потім можна виконати одну з наступних двох команд, щоб відобразити дані GPS:

```
sudo gpsmon
sudo cgps -s
```

Приклад даних , що повинні побачити спочатку, зображено на рис. 3.7.

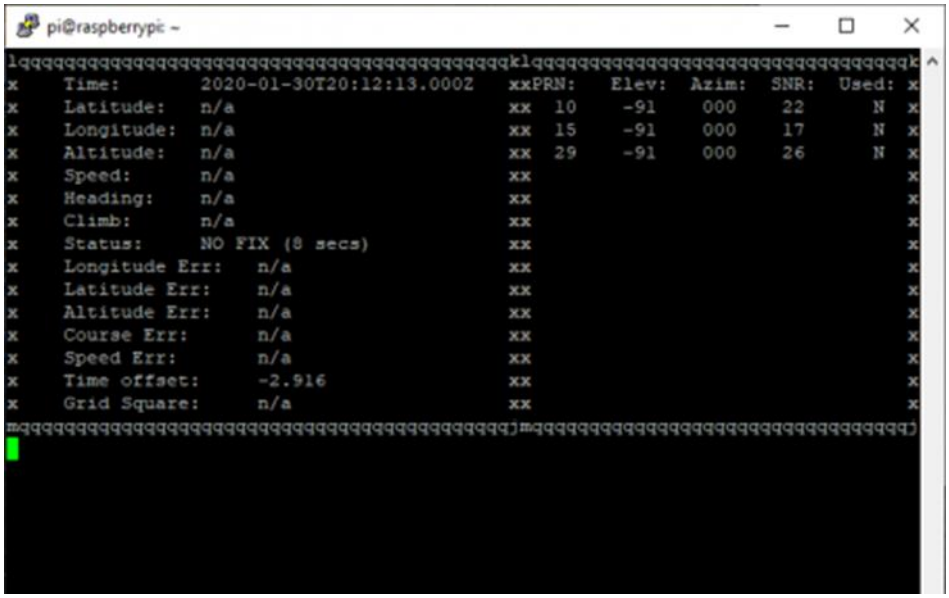

Рисунок 3.7 – Відображення даних GPS

Однак через деякий час модуль повинен зібрати достатньо даних для відображення позиції (рис. 3.8).

Може знадобитися до 30 хвилин, поки модуль не зможе визначити положення під час першого завантаження, особливо якщо перебуваєте в приміщенні. Спробуйте направити антену на вікно або, ще краще, поставити весь модуль біля вікна або на вулиці.

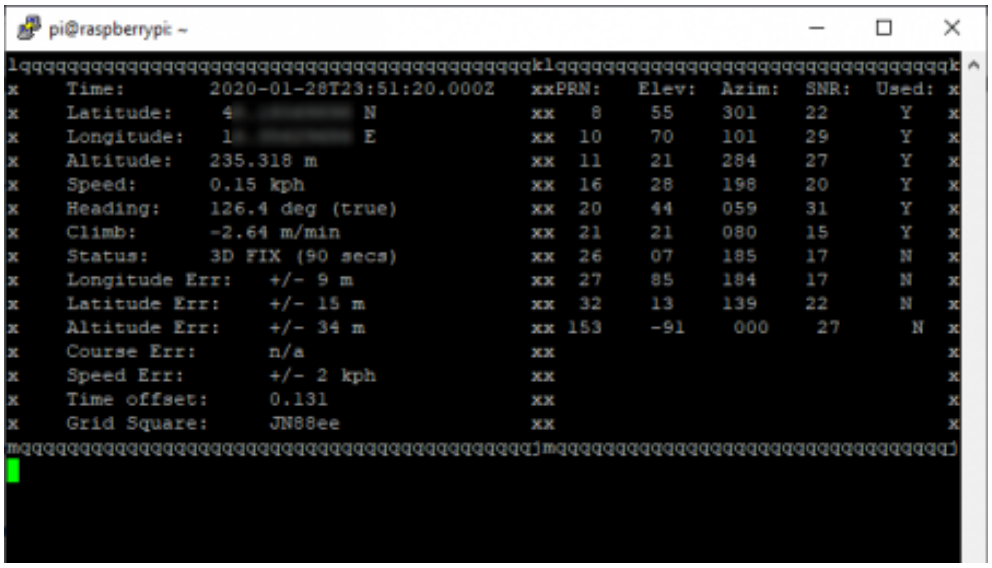

Рисунок 3.8 – Відображення більшої кількості даних GPS

Наступна команда допоможе, якщо отримуєте помилку під час запуску gpsmon і не видає результат під час запуску cgps:

```
sudo systemctl stop serial-getty@serial0.service
```

Тоді також зможете використовувати gpsmon.

Висновки до розділу 3

Під час оформлення розділу 3 було розглянуто плату мікроконтролера Raspberry Pi Pico. Також було розглянуто можливі мови програмування, серед яких можна виділити MicroPython, CircuitPython та C/C++.

MicroPython і CircuitPython досить близькі один до одного, але з деякими особливостями для налагодження та API. Для налаштування було обрано CircuitPython, оскільки ця мова програмування є похідною від MicroPython та має більш розширений фунціонал. Для розробки програмної частини використано середовище розробки Thonny.

В результаті оформлення розділу 3 було проведено підключення Raspberry Pi до середовища розробки Thonny та налаштування Raspberry Pi для роботи з GPS-модулем.

4 ОТРИМАННЯ ПОЗИЦІЇ GPS ТА ПОЛОЖЕННЯ ПРИСТРОЮ ЗА ДОПОМОГОЮ PYTHON

4.1 Отримання позиції GPS

Необроблені дані GPS (рис.4.1) містять рядки повідомлень RMS, VTG, GGA, GSA, GSV і GLL NMEA. Одночасно існує дев’ять рядків повідомлень необроблених даних GPS.

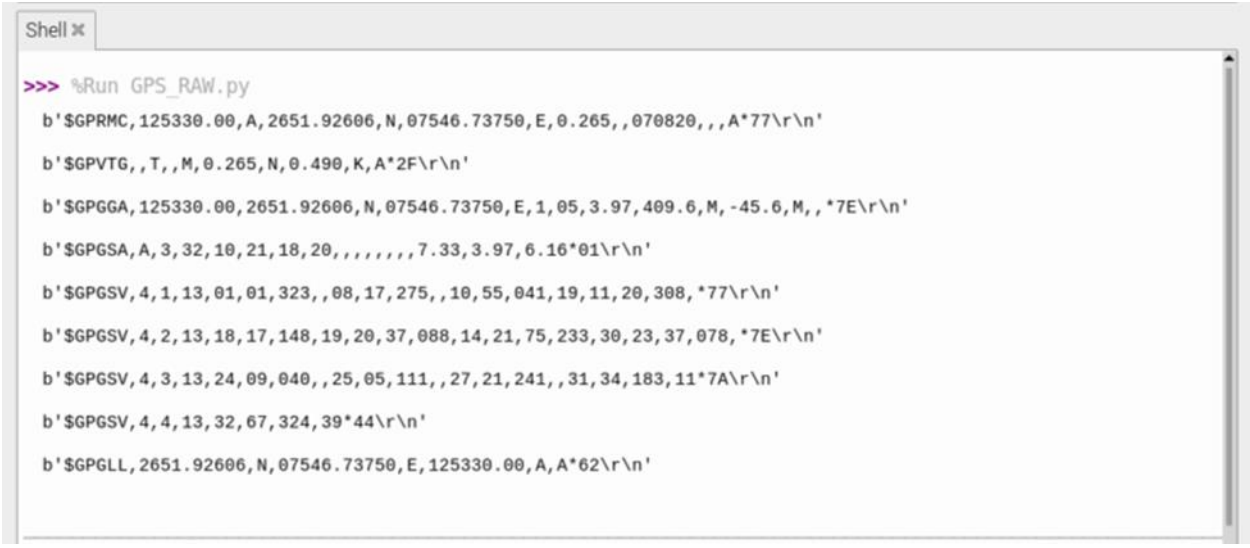

Рисунок 4.1 – Необроблені дані GPS

Щоб отримати GPS-місцезнаходження, можемо витягнути рядки повідомлення $GPGGA або $GPGLL. Обидва ці рядки містять інформацію про місцезнаходження. У наведеному прикладі було витягнуто місцезнаходження GPS із рядка повідомлення $GPGGA.

Необроблені дані GPS можна зберігати у змінній рядкового типу. Використовуючи функції обробки рядків, такі як find(), можемо шукати $GPGGA або $GPGLL в отриманому рядку.

Якщо отримані дані GPS містять один із цих рядків, можна розділити рядок повідомлення після $GPGGA або $GPGLL за допомогою функції split(). Потім розділити отримані поля даних повідомлення NMEA на масив, використовуючи кому як роздільник у функції split().

Тепер є поля даних $GPGGA або $GPGLL NMEA-повідомлення в масиві. Можна отримати час, широту та довготу за UTC, отримавши доступ до іншого індексу результуючого масиву. Широту та довготу можна конвертувати в градуси за допомогою простих математичних операцій і операцій форматування над отриманими значеннями. Перетворені значення широти та довготи також можна роздрукувати на консолі або перенести до змінної.

Нижче наведено лістинг коду Python, який витягує місцезнаходження GPS із необроблених даних GPS-модуля GPS NEO-6M:

```
import serial
from time import sleep
import sys

ser = serial.Serial (“/dev/ttyS0”)
gpgga_info = “$GPGGA,”
GPGGA_buffer = 0
NMEA_buff = 0

def convert_to_degrees(raw_value):
    decimal_value = raw_value/100.00
    degrees = int(decimal_value)
    mm_mmmm = (decimal_value – int(decimal_value))/0.6
    position = degrees + mm_mmmm
    position = “%.4f” %(position)
    return position

try:
    while True:
        received_data = (str)(ser.readline()) #read NMEA string received
        GPGGA_data_available = received_data.find(gpgga_info) #check  for  NMEA
GPGGA string
        if (GPGGA_data_available>0):
            GPGGA_buffer = received_data.split(“$GPGGA,”,1)[1]  #store    data
coming after “$GPGGA,” string
            NMEA_buff = (GPGGA_buffer.split(‘,’))
            nmea_time = []
            nmea_latitude = []
```

```
                nmea_longitude = []
                nmea_time = NMEA_buff[0]   #extract time from GPGGA string
                nmea_latitude = NMEA_buff[1]   #extract latitude from GPGGA string
                nmea_longitude = NMEA_buff[3]  #extract longitude from GPGGA string
                print(“NMEA Time: “, nmea_time,’\n’)
                lat = (float)(nmea_latitude)
                lat = convert_to_degrees(lat)
                longi = (float)(nmea_longitude)
                longi = convert_to_degrees(longi)
                print (“NMEA Latitude:”, lat,”NMEA Longitude:”, longi,’\n’)

    except KeyboardInterrupt:
        sys.exit(0)
```

Під час запуску вищевказаного сценарію було отримано наступну GPS-локацію в консолі IDLE (рис. 4.2):

Рисунок 4.2 – Відображення GPS локації в консолі

Щоб відобразити позицію GPS на Google Maps, можна використовувати бібліотеку веббраузера Python. Можна використовувати метод open() цієї бібліотеки, щоб відкрити посилання Google Maps із широтою та довготою, отриманими в попередньому прикладі.

Далі наведено лістинг коду Python, який показує місцезнаходження GPS-модуля у Google Maps:

```
    import serial
    from time import sleep
    import sys
    import webbrowser

    ser = serial.Serial (“/dev/ttyS0”)
    gpgga_info = “$GPGGA,”
    GPGGA_buffer = 0
```

```
      NMEA_buff = 0
      GPGGA_data_available = “”

      def convert_to_degrees(raw_value):
          decimal_value = raw_value/100.00
          degrees = int(decimal_value)
          mm_mmmm = (decimal_value – int(decimal_value))/0.6
          position = degrees + mm_mmmm
          position = “%.4f” %(position)
          return position

      received_data = (str)(ser.read(200)) #read NMEA string received
      GPGGA_data_available = received_data.find(gpgga_info)   #check for NMEA GPGGA
string
      if (GPGGA_data_available>0):
          GPGGA_buffer  =  received_data.split(“$GPGGA,”,1)[1]   #store  data  coming
after “$GPGGA,” string
          NMEA_buff = (GPGGA_buffer.split(‘,’))
          nmea_time = []
          nmea_latitude = []
          nmea_longitude = []
          nmea_time = NMEA_buff[0]    #extract time from GPGGA string
          nmea_latitude = NMEA_buff[1]#extract latitude from GPGGA string
          nmea_longitude = NMEA_buff[3]
          print(“NMEA Time: “, nmea_time,’\n’)
          lat = (float)(nmea_latitude)
          lat = convert_to_degrees(lat)
          longi = (float)(nmea_longitude)
          longi = convert_to_degrees(longi)
          print (“NMEA Latitude:”, lat,”NMEA Longitude:”, longi,’\n’)
          map_link = ‘http://maps.google.com/?q=’ + lat + ‘,’ + longi
          webbrowser.open(map_link)
      sys.exit(0)
```

Під час запуску вищевказаного сценарію отримали таке GPS-місцезнаходження в Google Maps (рис. 4.3):

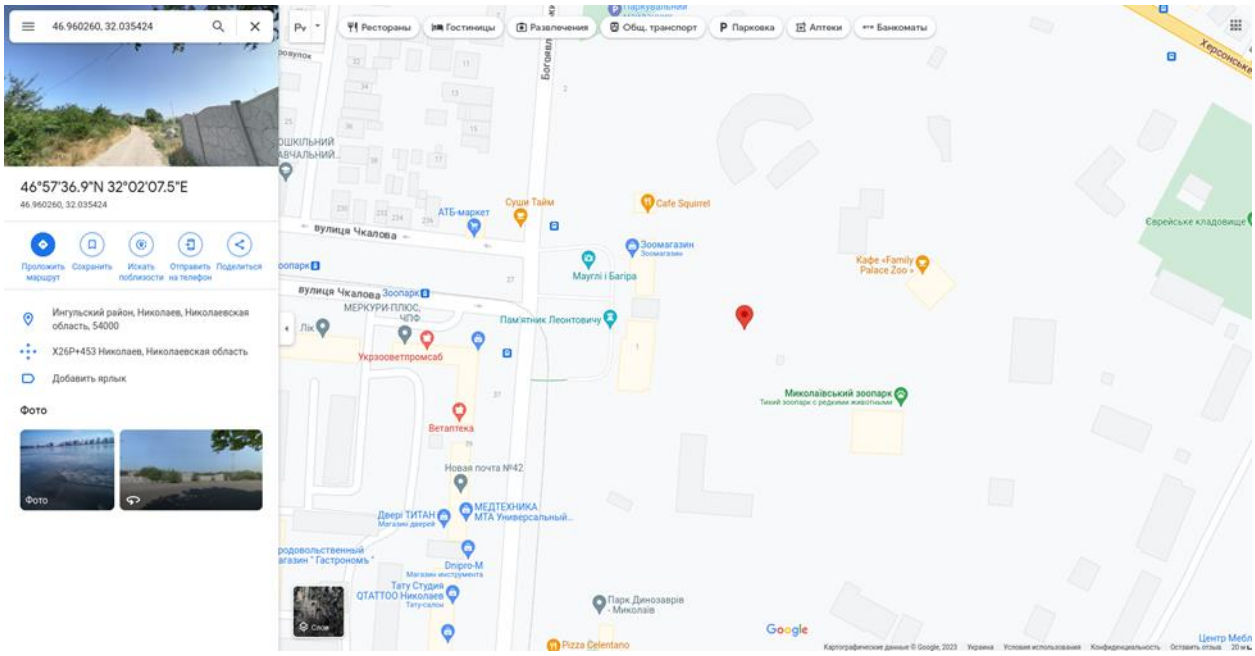

Рисунок 4.3 – GPS-місцезнаходження в Google Maps

Виходячи з отриманих даних та їх відображення у Google Maps можна зробити висновок, що GPS-модуль GPS NEO-6M було налаштовано, та він працює коректно.

4.2 Отримання положення пристрою

Датчик MPU-9250 буде спілкуватися з Raspberry Pi за допомогою протоколу I2C. Щоб читати та записувати дані через I2C, потрібно спочатку включити порти I2C на Raspberry Pi. Робимо це або за допомогою командного рядка, або перейшовши до Налаштування → Конфігурація Raspberry Pi → Інтерфейси → Увімкнути I2C.

Коли обидва пристрої MPU-9250 з’являться на детекторі I2C у вікні команд Raspberry Pi, готові прочитати MPU-6050 (адреса пристрою 0x68) і AK8963 (адреса пристрою 0x0C). При роботі с датчиком потрібні обидві адреси, щоб підключили їх до одного порту I2C. Тепер можемо використовувати адреси для керування ними в програмі.

Далі також потрібно буде збільшити швидкість передачі даних I2C (рис. 4.4), щоб отримати найшвидшу відповідь від MPU-9250. Це можна зробити, ввівши наступний рядок в командному рядку:

```
sudo nano /boot/config.txt
```

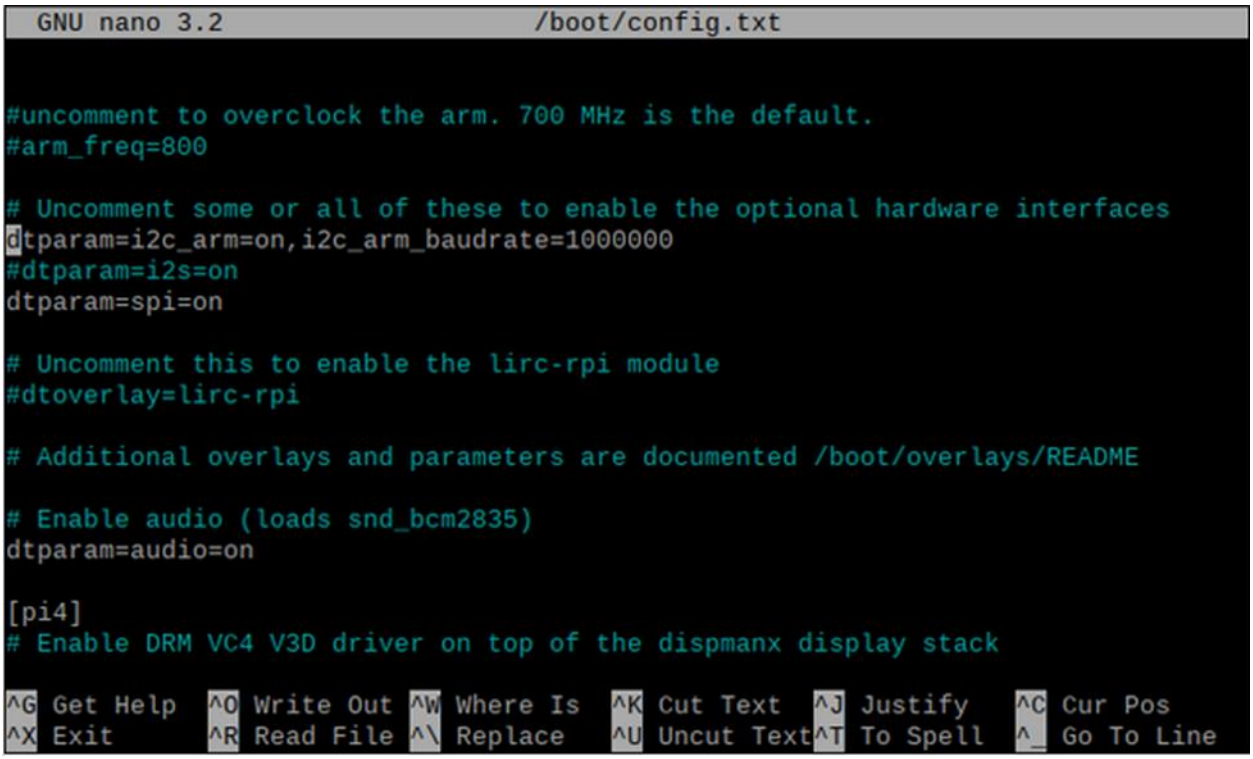

Рисунок 4.4 –Збільшення швидкості передачі даних I2C

У прикладі наведено встановлення швидкість передачі на 1 Мбіт/с. Це повинно дати частоту дискретизації приблизно 400–500 Гц (після перетворення в реальні одиниці). Можна досягти значно вищої частоти дискретизації для гіроскопа та трохи вищої частоти дискретизації для акселерометра, але для тестування пристрою вистачить і такої швидкості.

Нижче наведено лістинг коду використання разом із зразками зчитування та приклад вихідних даних (рис. 4.5).

```
from mpu9250_i2c import *

time.sleep(1) # delay necessary to allow mpu9250 to settle

print('recording data')
while 1:
    try:
        ax,ay,az,wx,wy,wz = mpu6050_conv()
        mx,my,mz = AK8963_conv()
    except:
        continue

    print('{}'.format('-'*30))
```

```
      print('accel [g]: x = {0:2.2f}, y = {1:2.2f}, z {2:2.2f}= '.format(ax,ay,az))
          print('gyro   [dps]:      x   =   {0:2.2f},   y   =   {1:2.2f},   z   =
{2:2.2f}'.format(wx,wy,wz))
          print('mag   [uT]:         x   =   {0:2.2f},   y   =   {1:2.2f},   z   =
{2:2.2f}'.format(mx,my,mz))
          print('{}'.format('-'*30))
          time.sleep(1)
```

Результат виконання вище зазначеного прикладу коду зображено на рис. 4.5.

```
accel [g]: x = 0.00, y = -0.00, z = 1.01
gyro [dps]:  x = 3.66, y = 0.37, z = 0.86
mag [uT]:   x = -6.88, y = 35.59, z = -2.69
------------------------------
------------------------------
accel [g]: x = 0.00, y = -0.00, z = 1.01
gyro [dps]:  x = 3.71, y = 0.37, z = 1.03
mag [uT]:   x = -7.03, y = 36.34, z = -2.54
------------------------------
------------------------------
accel [g]: x = -0.00, y = -0.00, z = 1.01
gyro [dps]:  x = 3.64, y = 0.24, z = 0.86
mag [uT]:   x = -7.33, y = 35.44, z = -2.84
------------------------------
------------------------------
accel [g]: x = 0.00, y = 0.00, z = 1.00
gyro [dps]:  x = 3.68, y = 0.39, z = 1.00
mag [uT]:   x = -7.63, y = 35.14, z = -1.64
------------------------------
------------------------------
accel [g]: x = 0.00, y = 0.00, z = 1.01
gyro [dps]:  x = 3.77, y = 0.27, z = 1.15
mag [uT]:   x = -7.63, y = 35.44, z = -2.84
------------------------------
------------------------------
accel [g]: x = -0.00, y = 0.00, z = 1.01
gyro [dps]:  x = 3.98, y = -0.12, z = 0.86
mag [uT]:   x = -7.18, y = 36.49, z = -4.19
------------------------------
------------------------------
accel [g]: x = 0.01, y = 0.00, z = 1.01
gyro [dps]:  x = 3.49, y = 0.64, z = 0.81
mag [uT]:   x = -6.73, y = 35.14, z = -1.94
------------------------------
------------------------------
accel [g]: x = 0.00, y = -0.00, z = 1.01
gyro [dps]:  x = 3.75, y = 0.43, z = 0.92
mag [uT]:   x = -8.52, y = 35.14, z = -2.24
------------------------------
```

Рисунок 4.5 – Вихідні дані

Вихідні дані можна використати, щоб перевірити чи працюють правильно датчик і код. Слід також зазначити наступне:

– у z-напрямку маємо значення, близьке до 1, це означає, що сила тяжіння діє у вертикальному напрямку, а позитивна – вниз.

– гіроскоп зчитує значення, близькі до 0, і в цьому випадку не перемістили пристрій, тому вони повинні бути близькі до 0.

– магнітометр показує значення від –10 мкТ до 40 мкТ у напрямках x, y, що приблизно дорівнює напруженості магнітного поля Землі в місті, де проводилися вимірювання.

Перевіривши ці значення, можна стверджувати, що датчик MPU-9250 працює коректно.

Далі отримані вихідні дані було візуалізовано за допомогою графіків як функцію часу (рис. 4.6).

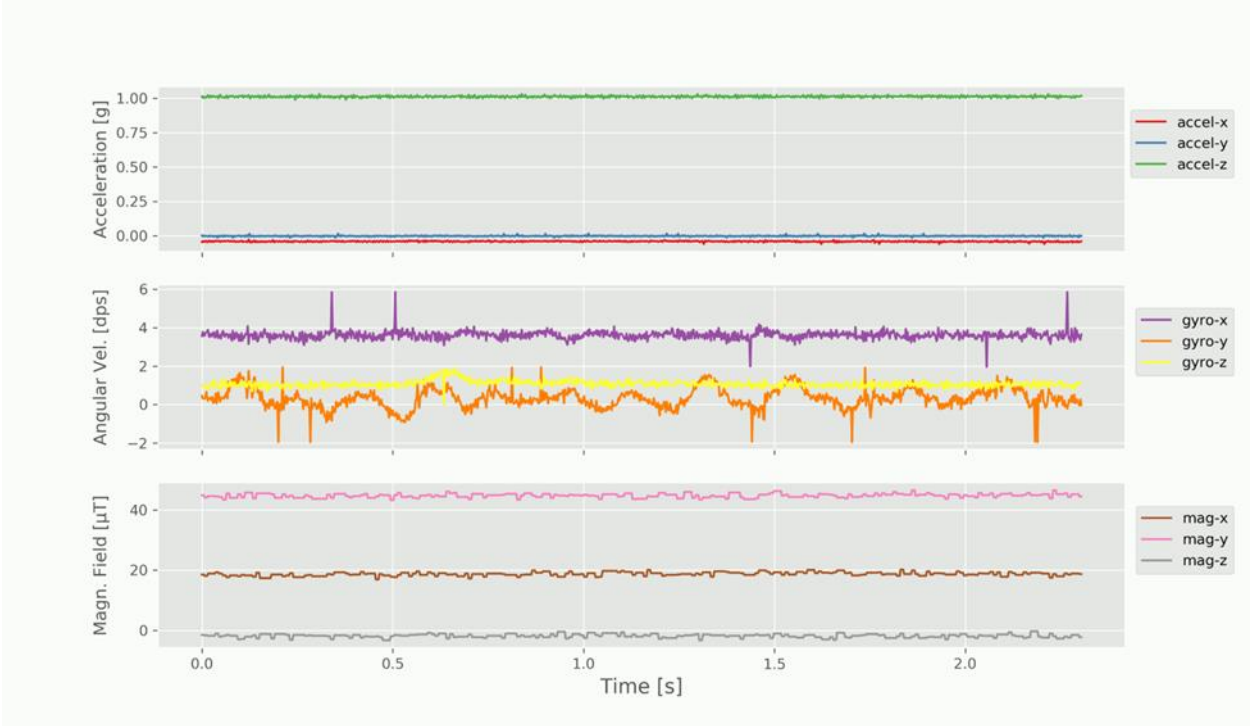

Рисунок 4.6 – Функція часу, що побудована з вихідних даних

Для тестування та отриманні іншого набору вихідних даних перевернемо датчик на бік так, щоб напрямок x був спрямований вгору. З

отриманих у цьому експерименті даних зможемо візуалізувати, як реагує кожен із 9 ступенів свободи (рис. 4.7).

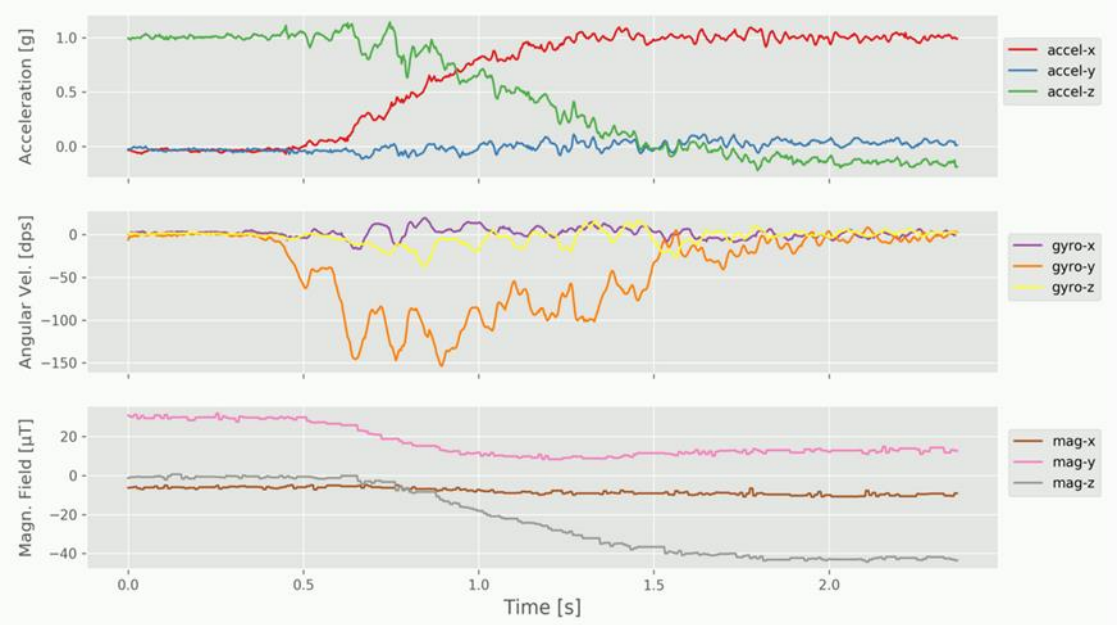

Рисунок 4.7 – Реагування на обертання пристрою

Нижче наведено лістинг коду програми:

```
from mpu9250_i2c import *
import smbus,time,datetime
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('ggplot')

time.sleep(1)

ii = 1000
t1 = time.time()


mpu6050_str = ['accel-x','accel-y','accel-z','gyro-x','gyro-y','gyro-z']
AK8963_str = ['mag-x','mag-y','mag-z']
mpu6050_vec,AK8963_vec,t_vec = [],[],[]

print('recording data')
for ii in range(0,ii):

    try:
        ax,ay,az,wx,wy,wz = mpu6050_conv()
        mx,my,mz = AK8963_conv()
    except:
```

```
            continue
        t_vec.append(time.time())
        AK8963_vec.append([mx,my,mz])
        mpu6050_vec.append([ax,ay,az,wx,wy,wz])

    print('sample rate accel: {} Hz'.format(ii/(time.time()-t1))) # print the
sample rate
    t_vec = np.subtract(t_vec,t_vec[0])

    fig,axs = plt.subplots(3,1,figsize=(12,7),sharex=True)
    cmap = plt.cm.Set1

    ax = axs[0]
    for zz in range(0,np.shape(mpu6050_vec)[1]-3):
        data_vec = [ii[zz] for ii in mpu6050_vec]
        ax.plot(t_vec,data_vec,label=mpu6050_str[zz],color=cmap(zz))
    ax.legend(bbox_to_anchor=(1.12,0.9))
    ax.set_ylabel('Acceleration [g]',fontsize=12)

    ax2 = axs[1]
    for zz in range(3,np.shape(mpu6050_vec)[1]):
        data_vec = [ii[zz] for ii in mpu6050_vec]
        ax2.plot(t_vec,data_vec,label=mpu6050_str[zz],color=cmap(zz))
    ax2.legend(bbox_to_anchor=(1.12,0.9))
    ax2.set_ylabel('Angular Vel. [dps]',fontsize=12)

    ax3 = axs[2]
    for zz in range(0,np.shape(AK8963_vec)[1]):
        data_vec = [ii[zz] for ii in AK8963_vec]
        ax3.plot(t_vec,data_vec,label=AK8963_str[zz],color=cmap(zz+6))
    ax3.legend(bbox_to_anchor=(1.12,0.9))
    ax3.set_ylabel('Magn. Field [Î¼T]',fontsize=12)
    ax3.set_xlabel('Time [s]',fontsize=14)

    fig.align_ylabels(axs)
    plt.show()
```

Отже, можна зробити кілька зауважень щодо поведінки коду зазначеного вище:

– обертання навколо осі Y призводить до прискорення сили тяжіння в напрямку X;

– обертання показує негативну кутову швидкість у напрямку Y;

– магнітне поле зміщується від напрямків X і Y до напрямків Y і Z.

Висновки до розділу 4

У розділі 4 було виконано налаштування GPS модулю NEO-6M та модуля MPU-9250. Також було протестовано коректність роботи модулів NEO-6M та MPU-9250.

Виходячи з отриманих результатів від GPS модуль NEO-6M даних та їх відображення у Google Maps можемо зробити висновок, що модуль працює коректно.

Використовуючи протокол I2C, було зчитано 9 різних змінних, по одній для кожної з трьох осей для кожного з трьох датчиків модулю

MPU-9250. Потім кожну з 9 змінних було візуалізовано та нанесено на графік для відображення зміни значень при русі пристрою.

Під час налаштування модуля MPU-9250 було виявлено наступні особливості його використання:

– обертання навколо осі Y призводить до прискорення сили тяжіння в напрямку X;

– обертання показує негативну кутову швидкість у напрямку Y;

– магнітне поле зміщується від напрямків X і Y до напрямків Y і Z.

В результаті оформлення розділу 4 було розроблено прототип системи моніторингу позиціювання. Також було протестовано розроблений прототип у різних умовах. Серед недоліків можна зазначити невелику кількість вихідних шумів у модуля MPU-9250, які можна вирішити програмно.

ВИСНОВКИ

Під час виконання кваліфікаційної магістерської роботи було проаналізовано принципи роботи аналогових рішень моніторингу позиціювання, які є актуальними на сьогоднішній час.

Було розглянуто сукупність методів і підходів до розв'язання задачі моніторингу позиціювання, а саме: акустичні, радіочастотні, магнітні, оптичні, інерційні та гібридні. Також проаналізовано аналоги та сформовані вимоги до майбутнього апаратно-програмного забезпечення.

У роботі розглянуто математичну модель роботи MEMS акселерометрів, гіроскопів та магнітометрів. Як результат було визначено, що середньоквадратична величина шуму на виході пов'язана із зазначеною у специфікації спектральною щільністю, а точніше, коренем з неї, і еквівалентною шумовою смугою пропускання.

Було обрано необхідні компоненти для створення системи моніторингу позиціювання об’єкта у просторі. Враховуючи усі переваги, було зроблено наступний вибір:

- одноплатний комп’ютер Raspberry Pi 4;
- триосьовий MEMS акселерометр;
- триосьовий MEMS магнітометр;
- MEMS гіроскоп;
- GPS модуль NEO-6MV2.

Мовою програмування обрано Python, так як ця мова ідеально підходить для програмування на Raspberry Pi та модулів для цієї плати. А саме було обрано CircuitPython, оскільки ця похідна мова програмування має більш розширений фунціонал.

У результаті розробки продемонстровано процеси підключення обраних датчиків до одноплатного комп’ютера. Виконано налаштування GPS-модуля NEO-6M і модуля MPU-9250 та протестовано коректність їх роботи. Також описано план для реалізації програмної частини та проведено

експеримент і тестування системи моніторингу позиціювання об’єкта у просторі для розуміння коректності роботи й працездатності системи, визначено ділянки для подальших покращень. Отримані данні було візуалізовано за допомогою графіків для відображення зміни значень при русі пристрою.

Під час налаштування модуля MPU-9250 було виявлено наступні особливості його використання:

– обертання навколо осі Y призводить до прискорення сили тяжіння в напрямку X;

– обертання показує негативну кутову швидкість у напрямку Y;

– магнітне поле зміщується від напрямків X і Y до напрямків Y і Z.

Результатом роботи є розроблений прототип системи моніторингу позиціювання, а також тестування розробленого прототипу в різних умовах. Серед недоліків можна зазначити невелику кількість вихідних шумів у модуля MPU-9250, які можна вирішити програмно.

Практичне значення отриманих результатів полягає у можливості використання розробленого апаратно-програмного комплексу для моніторингу позиціювання об’єкта у просторі у системах навігації, стеженні, охороні здоров’я, туризмі, виробництві, особистій безпеці тощо.

Робота пройшла апробацію під час XXV Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 07–11 листопада 2022 р.).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. WHO global report on falls prevention in older age. URL : https://bit.ly/3UBt9AY (Last accessed: 07.12.2022).

2. What is the internet of things (IoT)?. URL: https://bit.ly/3tAjYpr (Last accesse : 07.12.2022).

3. Fu Z, Delbruck T, Lichtsteiner P, Culurciello E. An address-event fall detector for assisted living applications. IEEE Transactions on Biomedical Circuits and Systems. 2008. Vol. 2, Is. 2, P. 88–96 (Last accessed: 11.12.2022).

4. Що означає модель Гауссової суміші (GMM). URL : https://bit.ly/3uBSvUC (дата звернення : 11.12.2022).

5. Приховані Марківські моделі. URL : https://bit.ly/3he545B (дата звернення : 11.12.2022).

6. Fuzzy Logic: Definition, Meaning, Examples, and History. URL : https://bit.ly/3VFXijO (Last accessed: 11.12.2022).

7. Santos G. L., Endo P. T., Monteiro K. H. de C., Rocha E. da S., Silva I., Lynn T. Accelerometer-based human fall detection using convolutional neural networks. Sensors (Switzerland). 2019. Vol. 19, Is. 7 (Last accessed: 02.01.2023).

8. Palmerini L., Klenk J., Becker C., Chiari L. Accelerometer-based fall detection using machine learning: Training and testing on real-world falls. Sensors (Switzerland). 2020. Is. 20, No. 22 (Last accessed: 02.01.2023).

9. Khojasteh S. B., Villar J. R., Chira C., González V. M., de la Cal E. Improving fall detection using an on-wrist wearable accelerometer. Sensors (Switzerland). 2018. Is. 18, No. 5 (Last accessed: 02.01.2023).

10. Video-based fall risk assessment system : пат. US20200205697A1 United States. № US16/731,025; Prior. 09.05.2020. Publ. 02.07.2020. URL: https://patents.google.com/patent/US20200205697A1/en (Last accessed: 04.01.2023).

11. Detecting falls using a mobile device : пат. US11282363B2 United States. № US16/929,028; Prior. 14.07.2020. Publ. 29.10.2020. URL: https://patents.google.com/patent/US11282363B2/en (Last accessed: 04.01.2023).

12. Fall detection systems and methods : пат. US10485452B2 United States. № US16/034,266; Prior. 12.07.2018. Publ. 22.11.2018. URL: https://patents.google.com/patent/US10485452B2/en (Last accessed: 04.01.2023).

13. Evaluation of raspberry harvesting by Raspberry-Pi / K. TOKUDA et al. The Proceedings of JSME annual Conference on Robotics and Mechatronics (Robomec). 2019. Vol. 2019. P. 1A1–D02. URL: https://doi.org/10.1299/jsmermd.2019.1a1-d02 (Last accessed: 11.01.2023).

14. Gay W. Pi Camera. Advanced Raspberry Pi. Berkeley, CA, 2018. P. 493–499. URL: https://doi.org/10.1007/978-1-4842-3948-3_26 (Last accessed: 07.01.2023).

15. Nusairat J. F. Raspberry Pi. Rust for the IoT. Berkeley, CA, 2020. P. 391–427. URL: https://doi.org/10.1007/978-1-4842-5860-6_8 (дата звернення: 07.01.2023).

16. TSUKAMOTO T. MEMS Vibratory Gyroscope. The Journal of The Institute of Electrical Engineers of Japan. 2019. Vol. 139, No. 3. P. 160–163. URL: https://doi.org/10.1541/ieejjournal.139.160 (Last accessed: 02.02.2023).

17. Firdaus F., Ismail I. Komparasi Akurasi Global Posistion System (GPS) Receiver U-blox Neo-6M dan U-blox Neo-M8N pada Navigasi Quadcopter. Elektron : Journal Ilmiah. 2020. Vol. 12, No. 1. P. 12–15. URL: https://doi.org/10.30630/eji.12.1.137 (Last accessed: 05.02.2023).

18. Huang H., Gartner G., Krisp J. M., Raubal M., Van de Weghe N. Location based services: ongoing evolution and research agenda. Journal of Location Based Services. 2018. Vol. 12, No. 2. P. 63–93 (Last accessed: 07.02.2023).

19. Klochko V. K., Smirnov S. A. Object tracking algorithm for a passive positioning system. Computer Optics. 2020. Vol. 44, No 2. P. 244–249. URL: https://doi.org/10.18287/2412-6179-co-609 (Last accessed: 07.02.2023).

20. Fielke G. J. Global positioning system integrity monitoring. URL: http://arrow.unisa.edu.au:8081/1959.8/84951 (Last accessed: 08.02.2023).

21. Soto D. TempSense: A Four-Channel Temperature Logger Based on a USB Bridge and CircuitPython. Journal of Open Hardware. 2021. Vol. 5, No. 1. URL: https://doi.org/10.5334/joh.34 (Last accessed: 08.02.2023).

22. Модель системи позиціонування та моніторингу на основі багаторівневої структури передачі даних в розподіленій мережі / О. І. Тимочко та ін. Системи озброєння і військова техніка. 2021. № 4 (68). С. 123–129. URL: https://doi.org/10.30748/soivt.2021.68.16 (дата звернення: 08.02.2023).

23. Volders L. Raspberry Pi Pico Simplified. Lulu Press, Inc., 2021 (Last accessed: 08.02.2023).

24. Bell C. Introducing the Raspberry Pi Pico. Beginning MicroPython with the Raspberry Pi Pico. Berkeley, CA, 2022. P. 1–42. URL: https://doi.org/10.1007/978-1-4842-8135-2_1 (Last accessed: 09.02.2023).

25. Everard B., Halfacree G. Get Started with MicroPython on Raspberry Pi Pico. Raspberry Pi Press, 2021 (Last accessed: 09.02.2023).

26. Ковальчук М. В., Обухова К. О. Інерційні датчики та системи позиціювання. Могилянські читання – 2022 : тези доп. XXV Всеукр. наук.-метод. конф. Миколаїв, 7–11 листоп. 2022 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2022. С. 68–70.

ДОДАТОК А

Лістинг коду програми

```
      Class SpeechRecognizer
      import speech_recognition as sr
      class SpeechRecognizer:
          def __init__(self):
              #self.lang="en-US"
              self.text_speach=""
              self.lang="uk-UA"
              self.r = sr.Recognizer()
          def run_recognizer(self):
              while 1:
                  with sr.Microphone() as source:
                      print("Say something!")
                      audio = self.r.listen(source,5,5)
                  try:

txt_answer=self.r.recognize_google(audio,language=self.lang)
                      self.text_speach=txt_answer
                  except sr.UnknownValueError:
                      print("Google Speech Recognition could not understand
audio")
                  except sr.RequestError as e:
                      print("Could not request results from Google Speech
Recognition service; {0}".format(e))
          def get_text_speach(self):
              return self.text_speach
          def clean_text_speach(self):
              self.text_speach=""
      Class Rele
      import RPi.GPIO as GPIO
      class Rele:
          def __init__(self):
              self.pin=None
              GPIO.setmode(GPIO.BCM)
          def set_pin(self,pin):
              self.pin=pin
              GPIO.setup(self.pin, GPIO.OUT, initial=1)
          def Turn_on(self):
              GPIO.output(self.pin, 0)
          def Turn_off(self):
              GPIO.output(self.pin, 1)
      from peewee import *
      class BaseModel(Model):
          class Meta:
              database   =   SqliteDatabase("/WebInterface/db.sqlite3")     #
соединение с базой, из шаблона выше
      # Определяем модель исполнителя
      class DevicesPages_language(BaseModel):
          language_id = AutoField(column_name='id')
          code = TextField(column_name='code')
          Name= TextField(column_name='Name')
```

```
    class Meta:
        table_name = 'DevicesPages_language'
class DevicesPages_translation_devices(BaseModel):
    id_dev = IntegerField(column_name='devices_id')
    language_id = IntegerField(column_name='language_id')
    Name = TextField(column_name='Name')
    class Meta:
        table_name = 'DevicesPages_translation_devices'
class DevicesPages_translation_room_names(BaseModel):
    language_id = IntegerField(column_name='language_id')
    rooms_id = IntegerField(column_name='rooms_id')
    Name = TextField(column_name='Name')
    class Meta:
        table_name = 'DevicesPages_translation_room_names'
class DevicesPages_translation_action(BaseModel):
    action_id = IntegerField(column_name='actions_id')
    language_id = IntegerField(column_name='language_id')
    Name = TextField(column_name='Name')
    class Meta:
        table_name = 'DevicesPages_translation_action'
class ActionListView_ActionList(BaseModel):
    rooms_id = IntegerField(column_name='rooms_id')
    device_id = IntegerField(column_name='device_id')
    action_id = IntegerField(column_name='action_id')
    description = TextField(column_name='description')
    class Meta:
        table_name = 'ActionListView_ActionList'
from Levenshtein import distance
class TextProcess:
    def __init__(self,dev,room,action):
        self.procesed_text={}
        self.inpt_text=""
        self.dev=dev
        self.room=room
        self.actrion=action
    def split_command(self):
        splited_command=[]
        temp_list_command=self.inpt_text.split(" ")
        for comand in temp_list_command:
            splited_command.append(comand.replace(" ",""))
        return splited_command
    def set_inpt_text(self,inpt_text):
        self.inpt_text=inpt_text
    def compare_txt(self,inpt_list,txt):
        for element_inpt_list in inpt_list.keys():
            #print(distance(element_inpt_list, txt),element_inpt_list)
            if len(txt)>3 and distance(element_inpt_list, txt)<=2  :
                return inpt_list[element_inpt_list]
            elif len(txt)<=3 and distance(element_inpt_list, txt)<=1 :
                return inpt_list[element_inpt_list]
        return False
    def proces_inpt_str(self):
        for inpt_part in self.split_command():
            dev=self.compare_txt(self.dev,inpt_part)
            room=self.compare_txt(self.room,inpt_part)
```

```
            action=self.compare_txt(self.actrion,inpt_part)
            if dev:
                self.procesed_text['dev']=dev
            if room:
                self.procesed_text['room']=room
            if action:
                self.procesed_text['action']=action
    def get_procesed_text(self):
        self.proces_inpt_str()
        return self.procesed_text
from Rele import*
from WebRequest import *
class control:
    def __init__(self):
        self.inpt_comand=""
        self.api_command=APICOMMAND()
    def set_inpt_command(self,inpt_comand):
        self.inpt_comand=inpt_comand
    def process_comand(self):
        if re.fullmatch(r'^(http|https)', self.inpt_comand.lower()):
            self.api_command.send_command(self.inpt_comand)
        else:
            rele_command=self.inpt_comand.lower().split(" ")
            self.rel=Rele()
            self.rel.set_pin(int(rele_command[1]))
            if rele_command[2]=='on':
                self.rel.Turn_on()
            else:
                self.rel.Turn_off()
```

ДОДАТОК Б

Апробація кваліфікаційної роботи

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили

«МОГИЛЯНСЬКІ ЧИТАННЯ – 2022:
Досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»

XXV Всеукраїнська науково-практична конференція

ТЕЗИ

Комп’ютерні науки.

Технічні науки

Миколаїв, 7–11 листопада 2022 року

Миколаїв – 2022

УДК 004.67

Ковальчук М. В.,
магістрант,
Обухова К. О.,
викладач,
ЧНУ ім. Петра Могили, м. Миколаїв, Україна

ІНЕРЦІЙНІ ДАТЧИКИ ТА СИСТЕМИ ПОЗИЦІЮВАННЯ

Більшість сучасних мобільних пристроїв мають тріади акселерометрів, гіроскопів та магнітометрів, часто на додаток до них ставиться і датчик атмосферного тиску. Цей набір датчиків може допомогти виявити різке погіршення фізичного стану власника мобільного пристрою, проте далеко не всі смартфони здатні отримати дані про власне позиціонування у просторі та ще менше з них можуть робити це в автономному режимі довгий час.

Об’єкт дослідження (розробки): процес отримання інформації щодо позиціювання апаратного забезпечення у просторі.

Предмет дослідження (розробки): методи та засоби отримання інформації щодо позиціювання апаратного забезпечення у просторі.

Акселерометр – це прилад, що вимірює проєкцію прискорення, що здається. Типовий акселерометр складається з трьох взаємно перпендикулярних вимірювальних осей, що реструють гравітаційне та лінійне прискорення.

За допомогою вимірювань тривісного акселерометра можна визначити його орієнтацію щодо опорного вектора, яким в цьому випадку гравітаційне прискорення. Тоді, однак, орієнтацію буде дозволено не повністю – залишиться невизначеність щодо кута повороту навколо осі, паралельного напрямку прискорення вільного падіння.

Припустимо, що в нашому розпорядженні є вимірювальний пристрій з акселерометром, що має три осі X, Y і Z. На рис. 1 осі позначені червоним, зеленим і синім кольором і утворюють ліву трійку векторів. Очевидно, що якщо для визначення орієнтації доступний лише вектор прискорення вільного падіння, то існуватиме нескінченна кількість можливих орієнтацій вимірювального пристрою, при яких вісь Z акселерометра буде вимірювати значення прискорення вільного падіння, але дозволити абсолютну орієнтацію пристрою не зможемо.

Щоб вирішити орієнтацію повністю, потрібен другий базисний вектор, який паралельний першому. Таким вектором може бути, наприклад, вектор магнітного поля планети. Якщо відомий його напрямок, то орієнтація буде вирішено однозначно.

Рисунок 1 – Векторне зображення вимірювального пристрою з акселерометром

Знаючи орієнтацію однієї системи координат щодо іншої стає можливим переводити виміри із системи координат пристрою на глобальну. А знання про прискорення у глобальній системі координат вирішується шляхом інтегрування відновити швидкість та отримати інформацію про відносне розташування.

Гіроскоп дозволяє вимірювати швидкість обертання пристрою. Відповідно для того, щоб привести швидкість до кута повороту, потрібно її інтегрувати. З цим положенням пов’язана основна проблема орієнтації тільки за допомогою гіроскопа – через постійне інтегрування не зовсім точних вимірювань кутових швидкостей, викликаних зміщенням нуля або температурними ефектами, отримаємо дрейф орієнтації, або, іншими словами, вона «відпливатиме» від справжнього значення.

Крім традиційної та добре вивченої задачі визначення орієнтації пристрою, інерційні датчики можуть використовуватися для відновлення траєкторії руху об’єкта. Таким об’єктом може бути пішохід чи автомобіль. В окремих випадках, наприклад, при кріпленні пристрою на ногі та попередньому точному калібруванні датчиків можна домогтися помилки повернення в точку початку руху, що не перевищує десятків сантиметрів для довжини шляху, що перевищує 100 метрів. Приклад відновленої траєкторії методом ZUPT (при скиданні помилки в періоди нерухомості), доповненим вимірюваннями датчика атмосферного тиску наведено на рис. 2.

Траєкторія руху включала прохід коридором, спуск сходами, ще один прохід і підйом на ліфті.

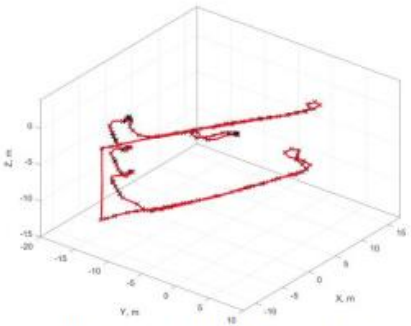

Рисунок 2 – Приклад відновленої траєкторії методом ZUPT, доповненим вимірюваннями датчика атмосферного тиску

Спираючись на отримані дані можна зробити висновок щодо поточного положення людини у просторі, а отже щодо фізичного стану людини, визначити його різке погіршення та автоматично інформувати про це відповідні екстрені контакти або спеціальні служби.

УДК 62-529:004.93

Короєв Р. В.,
бакалаврант,
Данилова О. М.,
бакалаврант,
Бурлаченко І. С.,
ст. викладач кафедри комп'ютерної інженерії,
ЧНУ ім. Петра Могили, м. Миколаїв, Україна

АВТОМОБІЛЬНА МОДЕЛЬ ДЛЯ ДИСТАНЦІЙНОГО ПОШУКУ ТА КАРТОГРАФУВАННЯ ВИБУХОНЕБЕЗПЕЧНИХ ПРИСТРОЇВ

Вибухонебезпечні предмети становитимуть загрозу життю людини та оточуючих. До них відносять: артилерійські снаряди, реактивні та