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.

_____Ірина ЖУРАВСЬКА

« __ » _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

РОЗРОБКА ТА ВПРОВАДЖЕННЯ ВЕЛОКОМП'ЮТЕРА НА БАЗІ ESP32 LILYGO S3 T-DISPLAY ДЛЯ МОНІТОРИНГУ ТА ЗБОРУ ПОКАЗНИКІВ СПОРТСМЕНА

Спеціальність 123 Комп’ютерна інженерія

Освітня програма «Комп’ютерна інженерія»

Здобувач _____Богдан ПИЛИПЧУК
підпис
«__» _____ 2024 р.

Керівник д-р техн. наук, проф. _____ Ірина ЖУРАВСЬКА
підпис
«__» _____ 2024 р.

Миколаїв – 2024

Завдання на виконання кваліфікаційної магістерської роботи

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«___»_____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____Пилипчука Богдана Віталійовича_____

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Розробка та впровадження велокомп'ютера на базі ESP32 LilyGO S3 T-Display для моніторингу та збору показників спортсмена.

Затверджена наказом ректора ЧНУ ім. Петра Могили від 16.09.2024 № 236.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є: апаратно-програмний комплекс моніторингу фізичної підготовки велоспортсмена на основі модуля ESP32 LilyGO S3 T- Display, який може бути використаний для вирішення актуальної проблеми контролю фізичних навантажень велоспортсменів, що дозволило б покращити якість підготовки наших майбутніх олімпійських призерів._____

4. Перелік питань, що підлягають розробці:
1) огляд сучасних підходів та систем моніторингу фізичних показників спортсмена;
2) аналіз переваг та недоліків існуючих систем;_____
3) розробити апаратну частину комплексу, яка складається з модуля ESP32 LilyGO S3 та датчиків швидкості та каденсу;_____

4) розробити програмну частину комплексу, яка складається з прошивки для модуля ESP32 LilyGO S3 T- Display, обробки даних, застосунок для відображення та керування комплексом._____

5. Перелік графічних матеріалів

_слайди презентації_____

6. Завдання до спеціальної частини

7. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Керівник роботи _____ Ірина ЖУРАВСЬКА
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач Богдан ПИЛИПЧУК
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «_20_» _____вересня_____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної магістерської роботи

Тема:_ Розробка та впровадження велокомп'ютера на базі ESP32 LilyGO S3 T-Display для моніторингу та збору показників спортсмена

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	01.09.2024	16.09.2024	*Виконано*
2.	Огляд літератури за темою роботи	17.09.2024	24.09.2024	*Виконано*
3.	Складання календарного плану КМР	25.09.2024	27.09.2024	*Виконано*
4.	Аналіз предметної області	28.09.2024	04.10.2024	*Виконано*
5.	Розробка проєктних рішень	05.10.2024	13.10.2024	*Виконано*
6.	Моделювання	14.10.2024	26.10.2024	*Виконано*
7.	Конструювання АПК	27.10.2024	03.11.2024	*Виконано*
8.	Перевірка працездатності, тестування та апробація розробленого АПК	02.11.2024	08.11.2024	*Виконано*
9.	Аналіз результатів тестування	08.11.2024	09.11.2024	*Виконано*
10.	Розробка керівництва користувача	10.11.2024	14.11.2024	*Виконано*
11.	Відгук керівника КМР	15.11.2024	17.11.2024	*Виконано*
12.	Оформлення КМР та презентації	18.11.2024	26.12.2024	*Виконано*
13.	Попередній захист	28.11.2024	28.11.2024	*Виконано*
14.	Рецензування	02.12.2024	11.12.2024	*Виконано*
15.	Захист кваліфікаційної роботи	19.12.2024	20.12.2024	*Виконано*

Керівник роботи _____ Ірина ЖУРАВСЬКА
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Богдан ПИЛИПЧУК
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Розробка та впровадження велокомп'ютера на базі ESP32 LilyGO S3 T-Display для моніторингу та збору показників спортсмена»

Здобувач гр. 605м Пилипчук Богдан Віталійович

Керівник: д-р техн. наук, професор Журавська Ірина Миколаївна

Актуальність теми кваліфікаційної роботи полягає у зростаючому значенні контролю фізичного стану спортсменів у сучасному світі, де технології відіграють ключову роль в управлінні тренувальними процесами. Велосипедний спорт, як один із найпопулярніших олімпійських видів спорту, вимагає точного контролю фізичних показників спортсменів, що є важливим для досягнення високих результатів. Сучасні напрямки велоспорту, такі як маунтенбайк та BMX, продовжують розвиватися, а використання інноваційних технологій, зокрема Інтернету речей, стає невід'ємною частиною цього процесу.

Розробка велокомп'ютера на базі ESP32 LilyGO S3 T-Display для моніторингу фізичних показників спортсменів відповідає актуальним тенденціям цифровізації спортивної підготовки. Така система дозволить підвищити ефективність тренувань за рахунок точного збору даних про фізичний стан спортсмена, що сприятиме оптимізації навантажень і запобіганню перевтоми. В умовах постійної потреби у збереженні фізичного здоров'я, такий комплекс стане корисним інструментом для підтримки спортсменів на високому рівні готовності до змагань.

Об'єкт дослідження: процеси вимірювання, передачі, обробки та відображення даних спортсмена.

Предмет дослідження (розробки): методи, засоби та технології вимірювання, передачі, обробки та відображення даних про спортсмена на основі модуля ESP32 LilyGO S3 T- Display та Інтернету речей.

Мета: розробити велокомп'ютер на базі ESP32 LilyGO S3 T- Display для моніторингу та збору показників спортсмена, який дозволить вимірювати та відображати основні показники, такі як швидкість, середню швидкість, каденс, пульс, час та інше.

Для досягнення поставленої мети було поставлено такі завдання:

− оглянути сучасні системи моніторингу фізичних показників велоспортсмена, проаналізувати переваги та недоліки існуючих систем;

− вивчити особливості модуля ESP32 LilyGO S3 T- Display та його можливості для реалізації системи моніторингу;

− розробити апаратну частину комплексу, яка складається з модуля ESP32 LilyGO S3 та датчиків швидкості та каденсу;

− розробити програмну частину комплексу, яка складається з прошивки для модуля ESP32 LilyGO S3 T- Display, обробки даних, вебзастосунок для відображення та керування апаратно-програмним комплексом (АПК);

− провести експериментальні випробування комплексу та оцінити його функціональність, точність, надійність та енергоефективність.

Кваліфікаційна робота містить: перелік скорочень, вступ, чотири розділи, висновки, перелік джерел посилання та два додатки.

Вступ містить основні обґрунтування актуальності розробки обраної теми, об’єкт, предмет дослідження, мету та завдання, які необхідно виконати для досягнення поставленої мети.

В першому розділі проведено аналіз різноманітних методів, засобів та аналогів моніторингу фізичних показників велоспортсмена, які використовуються для вирішення важливих завдань забезпечення здоров’я та фізичної підготовки спортсмена. Визначено переваги та обмеження, а також особливості аналізу на основі технології Інтернету речей (англ. Internet of Things – IoT). Розроблено специфікацію вимог до апаратно-програмного комплексу.

Другий розділ містить опис процесу проєктування АПК.

Третій розділ містить результати проєктування програмної частини АПК.

У четвертому розділі проведено експериментальні дослідження (тестування, перевірка працездатності АПЗ тощо), виконано аналіз результатів розробки.

У висновку описано результати виконання кваліфікаційної роботи.

Додатки містять код програмного забезпечення та інформацію про апробацію кваліфікаційної роботи.

Наукова новизна роботи полягає в удосконаленні алгоритму збору та передачі даних – фізичних показників велоспортсменів у реальному часі за рахунок інтеграції з Інтернетом речей, що дозволяє використовувати розроблену систему для різних дисциплін велоспорту та в реальних тренувальних умовах.

Кваліфікаційна робота містить 71 сторінку (без додатків), 40 рисунків, 32 джерела посилання, 2 додатки.

Ключові слова: *моніторинг фізичних показників велоспортсмена, IoT, ESP32 LilyGO S3 T-Display, датчик каденсу, датчик швидкості.*

ABSTRACT

of the Master’s Thesis

"Development and implementation of a cycling computer based on ESP32 LilyGO S3 T-Display for monitoring and collecting athlete indicators"

Applicant: Pylypchuk Bohdan Vitaliiovych

Supervisor: D.Sc. (Techn.), Professor Zhuravska Iryna Mykolaivna

The relevance of the topic of qualification work lies in the growing importance of monitoring the physical condition of athletes in the modern world, where technologies play a key role in the management of training processes. Cycling, as one of the most popular Olympic sports, requires precise control of athletes' physical performance, which is important for achieving high results. Modern directions of cycling, such as mountain biking and VMX, continue to develop, and the use of innovative technologies, in particular the Internet of Things, is becoming an integral part of this process.

The development of the cycling computer based on the ESP32 LilyGO S3 T- Display for monitoring the physical indicators of athletes corresponds to the current trends of digitalization of sports training. Such a system will make it possible to increase the effectiveness of training due to the accurate collection of data on the athlete's physical condition, which will contribute to the optimization of loads and the prevention of overfatigue. In conditions of constant need to maintain physical health, such a complex will be a useful tool for supporting athletes at a high level of readiness for competition.

Object of research: processes of measurement, transmission, processing and display of athlete data.

Subject of research (development): methods, means and technologies of measurement, transmission, processing and display of athlete data based on the ESP32 LilyGO S3 T-Display module and the Internet of Things.

Purpose: to develop a cycling computer based on the ESP32 LilyGO S3 T-Display for monitoring and collecting the performance of an athlete, which will allow to measure and display the main indicators such as speed, average speed, cadence, heart rate, time and others.

To achieve the goal, the following tasks were set:

− inspect modern systems for monitoring the physical indicators of cyclists;

− analyze the shortcomings of the existing systems;

− study the features of the ESP32 LilyGO S3 T-Display module and its possibilities for implementing the monitoring system;

− develop the hardware part of the complex, which consists of the ESP32 LilyGO S3 module and speed and cadence sensors;

− develop the software part of the complex, which consists of firmware for the ESP32 LilyGO S3 T-Display module, data processing, web applications for displaying and controlling the hardware and software complex (APK);

− conduct experimental tests of the complex and evaluate its functionality, accuracy, reliability and energy efficiency.

The qualification paper contains: a list of abbreviations, an introduction, four chapters, a conclusion, a list of reference sources and three appendices.

The introduction contains the main reasons for the relevance of the development of the chosen topic, the object, the subject of research, the goal and the tasks that must be performed to achieve the goal.

In the first section, an analysis of various methods, means and analogues of monitoring the physical indicators of a cyclist, which are used to solve important tasks of ensuring the health and physical training of an athlete, is carried out. Advantages and limitations, as well as features of IoT-based analysis, are identified. The specification of requirements for the hardware and software complex has been developed.

The second section contains a description of the hardware and software complex design process.

The third section contains the results of designing the software part of the APC.

In the fourth chapter, experimental studies were conducted (testing, checking the performance of the APS, etc.), and the analysis of the development results was performed.

The conclusion describes the results of the qualification work.

Appendices contain software code and information on the approval of the qualification work.

The scientific novelty of the work lies in improving the data collecting and transmission algorithm - physical indicators of cyclists - in real time through integration with the Internet of Things, which allows using the developed system for various cycling disciplines and in real training conditions.

The qualification work contains 71 pages (without appendices), 40 figures, 32 reference sources, 2 appendices.

Keywords*: monitoring of physical indicators of a cyclist, IoT, ESP32 LilyGO S3 T-Display, cadence sensor, speed sensor.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 4
ВСТУП 5
1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ, ЗАСОБІВ ТА АНАЛОГІВ МОНІТОРИНГУ ФІЗИЧНИХ ПОКАЗНИКІВ ВЕЛОСПОРТСМЕНА 8
1.1 Аналіз методів та засобів оцінювання фізичної підготовки 8
1.2 Порівняльний аналіз існуючих технологій 9
1.3 Особливості аналізу на основі IoT 11
1.4 Огляд та аналіз наявних аналогів 13
1.5 ESP32 LilyGO S3 T-Display – особливості та характеристики 18
1.6 Формування специфікації вимог до АПК 20
Висновок до розділу 1 21
2 ПРОЄКТУВАННЯ АПАРАТНОЇ ЧАСТИНИ МОНІТОРИНГУ ФІЗИЧНИХ ПОКАЗНИКІВ ВЕЛОСПОРТСМЕНА 23
2.1 Вибір та обґрунтування компонентів АПК 24
2.2 Розробка схеми електричного зв’язку компонентів АПК 29
2.3 Розробка друкованого корпусу для АПК 31
2.4 Монтаж та налагодження АПК 36
Висновок до розділу 2 37
3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ ЧАСТИНИ АПК 39
3.1 Вибір та обґрунтування мови програмування та середовища розробки для АПК 39
3.2 Розробка алгоритму роботи АПК 40
3.3 Бібліотеки для реалізації програмної частини АПК 50
Висновок до розділу 3 51

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ. АНАЛІЗ РЕЗУЛЬТАТІВ РОЗРОБКИ 53
4.1 USER GID по завантаженню та налагодження програмного коду на АПК53
4.2 Тестування апаратної та програмної частини АПК 57
Висновок до розділу 4 64
ВИСНОВКИ 66
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 68
ДОДАТОК А Код програми 72
ДОДАТОК Б Публікації за темою роботи 86
Б.1 Стаття у журналі категорії Б 86
Б.2 Свідоцтво про реєстрацію авторського права на твір 96

ПЕРЕЛІК СКОРОЧЕНЬ

АКБ – акумуляторна батарея
АПК – апаратно-програмний комплекс
ЕКГ – електрокардіограма
МТБ – маунтенбайк
ПЗ – програмне забезпечення
САПР – система автоматизованого проєктування

ANT+ – Adaptive Network Topology
BMX – Bicycle Moto Cross
FFC – Flexible Flat Cable
FTS – Foster's Training Scale
GPS – Global Positioning System
IoT – Internet of Things
RPE – Rating of Perceived Exertion
TFT – Thin-film-transistor
VAMs – Vertical Ascent Meters per Second

ВСТУП

Велосипедний спорт – один з найпопулярніших олімпійських видів спорту.

Перший світовий чемпіонат з велоспорту відбувся у 1893 р. Велоспорт увійшов до програми змагань уже на перших Олімпійських іграх 1896 р. [1]. У 1980-х роках в США почали набирати популярність нові напрямки велоспорту − МТБ (маунтенбайк, гірський велосипед) та BMX (байк-крос). З 1990 р. проводяться чемпіонати світу з маунтенбайку, а в 1996 р. цей вид спорту був включений до програми Олімпійських ігор. Чемпіонати світу з BMX почали проводити з 1982 року, а з 2003 р. BMX став олімпійською дисципліною і вперше був представлений на Олімпіаді 2008 р. в Пекіні.

У сучасному світі, коли пандемії, війни та інші надзвичайні події стали частиною нашої реальності, збереження фізичного здоров'я та контроль за фізичними навантаженнями набувають особливого значення. Це особливо актуально для спортсменів, які мають підтримувати свої тренування і рівень підготовки незалежно від зовнішніх обставин. У цьому контексті велика увага приділяється розробці систем моніторингу фізичних навантажень для велоспортсменів, що сприяє підвищенню якості їх підготовки та можливості досягнення олімпійських висот.

Тому актуальною є задача розробки та впровадження велокомп'ютера на базі ESP32 LilyGO S3 T-Display для моніторингу та збору показників спортсмена, яка б використовувала сучасні технології та можливості Інтернету речей. Така система дозволила б отримувати оперативну та достовірну інформацію про підготовку спортсмена, виявляти та запобігати їх травмуванню та покращувати їх результати.

Мета роботи: розробити велокомп'ютер на базі ESP32 LilyGO S3 T- Display для моніторингу та збору показників спортсмена, який дозволить

вимірювати та відображати основні показники, такі як швидкість, середню швидкість, каденс, пульс, час та інше.

Об'єкт дослідження: процеси вимірювання, передачі, обробки та відображення даних спортсмена.

Предмет дослідження: методи, засоби та технології вимірювання, передачі, обробки та відображення даних про спортсмена на основі модуля ESP32 LilyGO S3 T- Display та Інтернету речей.

Завдання роботи:

– проаналізувати аналогові пристрої які вже існують, порівняти їх з власним;

– вивчити особливості модуля ESP32 LilyGO S3 T- Display та його можливості для реалізації системи моніторингу;

– розробити апаратну частину комплексу, яка складається з модуля ESP32 LilyGO S3 та датчиків швидкості та каденсу;

– розробити програмну частину комплексу, яка складається з прошивки для модуля ESP32 LilyGO S3 T- Display, обробки даних, вебзастосунок для відображення та керування комплексом;

– провести експериментальні випробування комплексу та оцінити його функціональність, точність, надійність та енергоефективність.

Практичне значення роботи полягає у розробленні апаратно-програмного комплексу моніторингу фізичної підготовки велоспортсмена на основі модуля ESP32 LilyGO S3 T- Display, який може бути використаний для вирішення актуальної проблеми контролю фізичних навантажень велоспортсменів, що дозволило б покращити якість підготовки наших майбутніх олімпійських призерів. Комплекс може бути також адаптований за допомогою модулю ANT+ для інших цілей та застосувань, пов'язаних з моніторингом фізичних показників.

Публікації. Основні положення та результати роботи опубліковані у журналі «Вісник Хмельницького національного університету», 2024, Т. 333, № 2, С. 329–337 [2]. На розроблену комп’ютерну програму «Cyclist Monitoring System» отримано Свідоцтво про реєстрацію авторського права на твір № 123174 (2024) [12].

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ, ЗАСОБІВ ТА АНАЛОГІВ МОНІТОРИНГУ ФІЗИЧНИХ ПОКАЗНИКІВ ВЕЛОСПОРТСМЕНА

Фізична активність – це будь-який рух тіла, який здійснюється за допомогою скелетних м'язів і вимагає витрат енергії [3].

Фізичні навантаження є невід'ємною частиною спортивного тренування у всіх дисциплінах, в тому числі і у велоспорті. Ці навантаження включають різні види тренувань, спрямовані на розвиток фізичної підготовленості та вдосконалення витривалості, швидкості, сили та інших фізичних якостей спортсмена.

Оскільки фізичні навантаження можуть бути різної інтенсивності та тривалості, їх контроль є важливим завданням для тренерів та спортсменів. При правильному підході до тренувань можна досягти максимальних результатів і знизити ризик отримання травм.

1.1 Аналіз методів та засобів оцінювання фізичної підготовки

Для оцінювання фізичної підготовки використовуються різні шкали оцінювання, основними із яких є:

– шкала FTP [4];
– шкала Вамса (VAMs) [5];
– шкала Фостера (FTS);
– шкала Ратінга спроможності (RPE);
– шкала Борхардта.

Щоб досягти оптимальних результатів у велоспорті, тренування повинні бути ретельно сплановані та контрольовані. Важливо враховувати особливості кожного велосипедиста, його фізичні можливості та тренувальні цілі.

Фізичну активність спортсменів можна контролювати за допомогою різних методів, включаючи вимірювання частоти серцевих скорочень, середньої швидкості під час тренувань, пройденої дистанції, каденсу та інших

показників. Розробка та провадження велокомп’ютера забезпечить можливість отримання та зберігання точних показників спортсмена під час тренувань.

1.2 Порівняльний аналіз існуючих технологій

1.2.1 IoT-рішення

Це технологія, яка використовує Інтернет речей (далі – IoT) для збору, передачі, обробки та візуалізації даних про фізичні показники у реальному часі. Наприклад, можна використовувати модуль ESP32 LilyGO S3 T- Display, який є компактним, енергоефективним, надійним та простим у використанні пристроєм, здатним за допомогою окремих датчиків вимірювати різні параметри (каденс, швидкість, частоту серцевих скорочень), передавати дані бездротовим шляхом та візуалізувати їх на веб-інтерфейсі. Переваги IoT-рішень полягають у високій точності, швидкості, надійності, енергоефективності, низькій вартості та зручності. Обмеження IoT-рішень полягають у складності інтеграції, потребі в стабільному зв’язку та живленні, ризику втрати або пошкодження даних, а також у відсутності стандартизації та регулювання [6].

1.2.2 Лабораторний аналіз

Тестування фізичного стану шосейних велосипедистів у лабораторії зазвичай включає кілька етапів для оцінки різних фізіологічних параметрів. Основними складовими таких тестів є:

1.2.2.1 Тестування на велоергометрі або тредмілі

Велоспортсмен сідає на спеціальний велоергометр, який відтворює умови шосейної їзди. Велоергометр має можливість регулювання навантаження (потужності), яке поступово збільшується. Водночас

спортсмени можуть бути підключені до обладнання для вимірювання важливих параметрів [7].

1.2.2.2 Вимірювання VO_2 max (максимального споживання кисню)

VO_2 max – це максимальна кількість кисню, яку організм може засвоїти за хвилину під час максимальних навантажень. Це один з ключових показників аеробної витривалості. Під час тестування спортсмен дихає через спеціальну маску, яка вимірює кількість кисню та вуглекислого газу. Це дає можливість оцінити ефективність роботи легенів і серцево-судинної системи.

1.2.2.3 Лактатний поріг

Під час збільшення інтенсивності велоспортсмена також вимірюють рівень лактату (молочної кислоти) в крові. Це дозволяє визначити лактатний поріг – той момент, коли рівень лактату починає швидко зростати, що є індикатором переходу організму від аеробного до анаеробного етапу роботи. Для цього під час тесту можуть брати зразки крові з пальця чи вени на різних етапах навантаження.

1.2.2.4 Електрокардіограма

Під час тесту спортсмени підключені до обладнання для запису електрокардіограми (ЕКГ), яке фіксує роботу серця на різних етапах навантаження. Це допомагає виявити можливі порушення серцевого ритму та оцінити, як серце реагує на стресові умови.

1.2.2.5 Оцінка потужності та ефективності педалювання

Важливим аспектом є також оцінка потужності, яку велосипедист може підтримувати протягом тривалого часу. Велоергометр дозволяє виміряти точну кількість ват, яку спортсмен може генерувати. Додатково можна

аналізувати техніку педалювання, щоб визначити, наскільки ефективно спортсмен використовує свою енергію.

Ці дослідження дозволяють тренерам і лікарям оцінити загальний стан велоспортсмена, виявити його сильні та слабкі сторони та розробити індивідуальний план тренувань і відновлення для підвищення результатів. Але таку процедуру не можливо виконувати кожен день, тому він не надає можливість отримувати певні показники миттєво та в реальному часі під час щоденних тренувань.

1.3 Особливості аналізу на основі IoT

Останнім часом все більшу популярність набувають моніторингові системи на основі Інтернету речей (IoT), які використовують сучасні технології та можливості для збору, передачі, обробки та візуалізації даних у реальному часі. Такий моніторинг та збір показників спортсмена має ряд переваг, таких як:

Висока точність – датчики, які використовуються для вимірювання параметрів, мають високу точність та стабільність, а також можуть бути калібровані та налаштовані відповідно до потреб;

Висока швидкість – дані можуть бути передані бездротовим шляхом до веб-сервера за допомогою Wi-Fi або LoRa, а також можуть бути доступні для перегляду та аналізу на веб-інтерфейсі у реальному часі;

Висока надійність – система моніторингу фізичних показників IoT може працювати автономно та безперебійно, а також мати функції самодіагностики та відновлення після збоїв;

Низьке енергоспоживання – система може використовувати енергозберігаючі технології та джерела живлення;

Низька вартість – система моніторингу може використовувати недорогі та доступні компоненти, такі як модуль ESP32 LilyGO S3 T- Display,

який є компактним, енергоефективним, надійним та легким у використанні пристроєм, здатним вимірювати різні параметри спортсмена, передавати дані бездротовим шляхом та візуалізувати їх за допомогою веб-інтерфейсу;

Висока зручність – система моніторингу IoT може бути легко встановлена, налаштована та керована за допомогою вебінтерфейсу, який надає інтуїтивний та інтерактивний спосіб перегляду та аналізу даних про фізичні параметри на карті, графіках, таблицях тощо.

Однак, система моніторингу фізичник показників спортсмена, також має деякі обмеження та недоліки, такі як:

Складність інтеграції – система моніторингу вимагає сумісності та інтеграції різних компонентів, таких як датчики, шлюзи, комунікаційні протоколи, платформи IoT, веб-сервери, веб-інтерфейси тощо, що може бути складним та трудомістким процесом;

Потенційні ризики безпеки – система IoT може бути вразливою до кібератак, які можуть перехопити, змінити або видалити дані, а також викликати несправність або пошкодження системи, що може призвести до неправильної оцінки;

Недостатність стандартів та регулювання – система IoT може зіткнутися з відсутністю або невідповідністю стандартів та регулювання, які визначають вимоги та критерії для вимірювання, передачі, обробки та візуалізації даних про фізичні показники, а також захисту прав та інтересів зацікавлених сторін.

Таким чином, система моніторингу IoT є перспективним та інноваційним засобом для покращення та оптимізації моніторингу фізичних показників велосипедиста, але вона також потребує подальшого дослідження та розвитку, щоб подолати існуючі недоліки.

1.4 Огляд та аналіз наявних аналогів

Існують різні аналоги та технології моніторингу фізичних показників велосипедиста.

1.4.1 Велокомп'ютер Garmin

Велокомп'ютер Garmin – це багатофункціональний пристрій, призначений для велосипедистів, що дозволяє відстежувати та аналізувати їхню активність. Він оснащений різними датчиками, які вимірюють швидкість, дистанцію, час, пульс та інші показники (рис. 1.1.). Завдяки вбудованому GPS велокомп'ютер Garmin також записує маршрут поїздки.

Рисунок 1.1 – Велокомп’ютер Garmin [8]

Основні особливості:

– GPS-відстеження маршруту;

– аналіз тренувань: фіксує потужність, каденс, пульс та інші важливі дані, допомагаючи аналізувати ефективність тренувань;

– дисплей: на екрані в реальному часі відображаються важливі дані, такі як швидкість, відстань та час;

– синхронізація з мобільними застосунками Garmin Connect для збереження даних;

– водонепроникність та тривалий час автономної роботи, що дозволяє використовувати пристрій навіть в складних умовах.

Переваги велокомп'ютерів Garmin:

– можливість навігації: деякі моделі пропонують навігацію в реальному часі, що полегшує пошук нових маршрутів або підтримання напрямку під час поїздки;

– багатофункціональність: вимірюють не тільки базові показники, але й такі, як каденс, потужність, що корисно для професійних тренувань;

– водонепроникність: велокомп'ютери Garmin зазвичай мають високий рівень захисту від води, що дозволяє використовувати їх в будь-яку погоду;

– довговічність: пристрої створені для тривалого використання та здатні працювати в складних умовах (бруд, дощ, холод тощо).

Недоліки велокомп'ютерів Garmin:

– ціна: велокомп'ютери Garmin є одними з найдорожчих на ринку, що може стати бар'єром для багатьох користувачів;

– складність інтерфейсу: через велику кількість функцій, налаштування і використання може бути складним для новачків або тих, хто не знайомий з подібною технікою;

– розмір та вага: деякі моделі можуть бути громіздкими, що не завжди зручно для тих, хто шукає компактність;

– точність у важких умовах: у місцях зі слабким сигналом GPS (наприклад, у горах чи лісах) може спостерігатися зниження точності вимірювань;

– проблеми з синхронізацією: іноді можуть виникати збої або затримки в синхронізації даних між пристроєм та програмними застосунками.

Цей пристрій непогано підходить для професійних тренувань, але все ж таки має ряд недоліків.

1.4.2 Велокомп'ютер Wahoo ELEMNT Roam

Велокомп'ютер Wahoo ELEMNT Roam – це пристрій для велосипедистів, який забезпечує достатньо високий рівень функціональності, надійності та зручності використання, особливо для тих, хто захоплюється тривалими поїздками (рис. 1.2). Цей велокомп'ютер є флагманською моделлю Wahoo і підходить як для професійних спортсменів, так і для серйозних аматорів.

Рисунок 1.2 – Wahoo ELEMNT Roam [9]

Основні особливості Wahoo ELEMNT Roam:

– велокомп'ютер має інтегрований GPS з підтримкою карт у реальному часі та функцією «Turn-by-Turn» (покрокова навігація), що допомагає легко слідувати запланованому маршруту або шукати нові шляхи;

– 2,7-дюймовий кольоровий екран з антибліковим покриттям забезпечує добру видимість навіть при яскравому сонячному світлі;

– синхронізація з мобільними застосунками: завдяки застосунку Wahoo Companion, користувач може легко налаштовувати велокомп'ютер, планувати маршрути;

– Wahoo ELEMNT Roam підтримує підключення до різних сенсорів через ANT+ і Bluetooth, що дозволяє відстежувати каденс, пульс, потужність та інші важливі показники;

– функція «Back on Track»: якщо велосипедист зіб'ється з маршруту, пристрій автоматично перерахує маршрут і поверне користувача на правильний шлях.

Переваги Wahoo ELEMNT Roam:

– реалізація навігації серед велокомп'ютерів завдяки функціям «Back on Track» та синхронізації з картами;

– чіткий і зручний екран, добре видимий навіть під прямим сонячним світлом.

Недоліки Wahoo ELEMNT Roam:

– висока ціна: пристрій є одним із найдорожчих на ринку, що може бути недоліком;

– менший екран у порівнянні з конкурентами: хоча дисплей чіткий, він дещо менший, ніж у деяких конкурентів (наприклад, Garmin);

– обмежена кількість спортивних режимів: порівняно з іншими велокомп'ютерами (наприклад, Garmin), Roam має менший вибір режимів для різних видів спорту, зосереджуючись переважно на велоспорті;

– менш потужний інтерфейс карт: хоча навігація хороша, карти не такі деталізовані, як у деяких конкурентів.

Даний велокомп’ютер є конкурентом пристрою Garmin, має деякі особливі функції, яких немає у інших виробників.

Практична значимість роботи полягає у розробці інноваційного велокомп'ютера на основі ESP32 LilyGO S3 T-Display з інтеграцією технологій Інтернету речей (IoT) для моніторингу фізичних показників велоспортсменів. Основні нові елементи (переваги) даної системи:

− **високий рівень адаптивності** – на відміну від багатьох існуючих рішень, система дозволяє підлаштовувати параметри для різних дисциплін велоспорту, таких як маунтенбайк, BMX та інші, що робить її універсальною для широкого кола спортсменів;

− **модульна структура** – використання відкритої платформи ESP32 LilyGO дозволяє розширювати функціональність системи завдяки додатковим сенсорам та модулям, що підвищує її універсальність і точність у зборі даних. Це надає більшу свободу в порівнянні з закритими рішеннями, де можливості інтеграції обмежені;

− **енергоефективність і автономність** – розробка передбачає тривалу роботу без потреби в частому підзаряджанні, що є важливим для спортсменів під час польових тренувань і змагань. Це рішення враховує мобільність та потреби спортсменів, які часто перебувають на тренування, змаганнях;

− **інтуїтивний інтерфейс з можливістю кастомізації** – використання дисплея T-Display дозволяє гнучко налаштовувати інтерфейс відповідно до індивідуальних потреб користувача, що сприяє зручності використання в реальних тренувальних умовах;

− **інтеграція з Інтернетом речей** – можливість передачі даних у реальному часі та віддаленого моніторингу підвищує ефективність тренувального процесу, даючи змогу тренерам оперативно коригувати тренування.

1.5 ESP32 LilyGO S3 T-Display – особливості та характеристики

ESP32 LilyGO S3 T-Display – це потужна плата для розробки на базі мікроконтролера ESP32-S3 з підтримкою Wi-Fi і Bluetooth від LilyGO, оснащена кольоровим дисплеєм та має вбудовані функції для IoT-проєктів.

– ESP32-S3 підтримує різні протоколи безпеки для Wi-Fi і Bluetooth, такі як WPA, WPA2, WAPI, AES, SHA, RSA і ECC1;

– ESP32-S3 має вбудований криптографічний сопроцесор, який прискорює шифрування і дешифрування даних;

– ESP32-S3 має спеціальний сопроцесор з ультранизьким енергоспоживанням, який може виконувати фонові завдання, такі як вимірювання датчиків і пробудження основної системи;

– ESP32-S3 має різні режими живлення і сну, які дозволяють оптимізувати енергоефективність і продовжити час роботи від батареї;

– ESP32-S3 може працювати в різних діапазонах напруги від 2,2 В до 3,6 В.

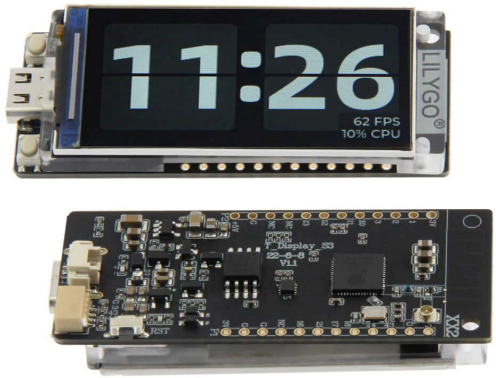


Рисунок 1.4 – ESP32 LilyGO S3 T-Display [10]

Характеристики та особливості ESP32:

– двоядерний 32-бітний процесор Tensilica Xtensa LX7;

– тактова частота – до 240 МГц;

– оперативна пам’ять – 520 кбайт SRAM;

– вбудована флеш-пам'ять: 16 Мбайт;

– дисплей: 1,9-дюймовий кольоровий TFT-дисплей (240 × 135 пікселів) ;

– інтерфейси зв'язку: Wi-Fi (2.4 GHz), Bluetooth (5.0 BLE);

– 34 програмованих GPIO-піни;

– USB інтерфейс: USB Type-C для програмування та живлення;

– підтримка батареї: роз'єм для підключення літій-полімерного акумулятора;

– живлення: вхідна напруга від 3,3 В до 5 В (підтримка живлення від акумулятора);

– максимальний струм споживання 260 мА, в сплячому режимі – 10 мА;

– 12-бітний АЦП і ЦАП;

– інтерфейси SPI, I2C, UART, I2S, CAN.

Також ESP32-S3 ***має багато переваг*** порівняно з іншими платами, такими як Arduino або Raspberry Pi :

– ESP32-S3 має вбудовані Wi-Fi та Bluetooth, в той час для Arduino необхідно підключати додаткові модулі;

– ESP32-S3 забезпечує більший обсяг пам’яті, вищу швидкість та розширені можливості порівняно з Arduino;

– ESP32-S3 споживає мало енергії і може працювати від батареї, тоді як Raspberry Pi вимагає постійного джерела живлення;

– ESP32-S3 підтримує різні операційні системи, такі як FreeRTOS, Arduino, MicroPython та ESP-IDF, у той час як Raspberry Pi працює виключно на Linux.

Звичайно, ESP32-S3 ***також має деякі недоліки:***

– ESP32-S3 має менше GPIO-пінів, ніж Arduino та Raspberry Pi;

– ESP32-S3 має менше графічних можливостей, ніж Raspberry Pi;

– ESP32-S3 має менше документації і прикладів, ніж Arduino або Raspberry Pi.

Дослідивши ESP32-S3 та порівнявши його з іншими платами такими як Arduino та Raspberry Pi, було вирішено, що ESP32-S3, незважаючи на деякі недоліки, має перевагу над іншими аналогами.

1.6 Формування специфікації вимог до АПК

В першу чергу треба визначитись з усіма компонентами АПК, які будуть використанні у проєкті.

Зробивши аналіз наявних компонентів та їх характеристик, було зроблено такий вибір:

– плата ESP32 LilyGO S3 T-Display;

– датчик каденсу CYCPLUS Cadence Sensor [11];

– датчик швидкості CYCPLUS Speed Sensor [11];

– акумуляторна батарея li-ion 950 mAh;

– плата TP4056 для зарядки Li-Ion батареї;

– з'єднувальні дроти 26 AWG;

– конектор для АКБ JST 1.25 mm 2pin;

– роздрукований корпус на 3D-принтері для плати ESP32 LilyGO S3 T- Display.

Це основні компоненти які будуть використані.

Актуальністю розробки даного АПК є те, що завдяки йому тренери зможуть швидше давати оцінку фізичної підготовки спортсмена, що дозволить пришвидшити та покращити коригування загальних відновлювальних процесів для організму спортсмена.

АПК повинно вимірювати параметри спортсмена, такі як швидкість та каденс, також розраховувати середню швидкість, пройдений шлях та будувати

карту. Також можна збільшити кількість параметрів, якщо взяти додаткові датчики, наприклад датчик пульсу. АПК повинно передавати параметри спортсмена через Wi-Fi або Bluetooth до сервера або мобільного застосунку, де вони можуть бути оброблені, візуалізовані та збережені. АПК повинно бути енергоефективним, надійним, точним, адаптивним та безпечним.

Висновок до розділу 1

У даному розділі проведено аналіз різноманітних методів, засобів та аналогів моніторингу фізичних показників велоспортсмена, які використовуються для вирішення важливих завдань забезпечення здоров’я та фізичної підготовки спортсмена. Визначено переваги та обмеження, а також особливості аналізу на основі IoT.

Проведено порівняльний аналіз різних аналогів та технологій моніторингу фізичних показників спортсмена, таких як Wahoo ELEMNT Roam, Garmin, які використовують сучасні технології та можливості для збору, передачі, обробки та візуалізації даних про фізичну підготовку у реальному часі, але вимагають уваги до особливостей кожної з систем. Вказано на їх переваги та обмеження, а також на можливість адаптації до різних сценаріїв та вимог.

Обґрунтовано вибір ESP32 LilyGO S3 T-Display, як оптимального мікроконтролера для проєкту моніторингу фізичних показників велоспортсмена на основі модуля ESP32, оскільки він має вбудований Wi-Fi і Bluetooth, має більше пам’яті, швидкості і функціональності, низьке споживання енергії і може працювати від батареї, низьку вартість, підтримку різних операційних систем, а також переваги порівняно з іншими платами, такими як Arduino та Raspberry Pi.

Описано процес формування специфікації вимог до АПК, які планується розробити для моніторингу фізичних показників. Було визначено цілі та задачі АПК, а також компоненти, які будуть використовуватись для реалізації.

Сформовано специфікацію вимог до АПК, яка включає різні розділи, що описують функціональність, характеристики, обмеження та взаємодію апаратно-програмного забезпечення з іншими компонентами системи.

2 ПРОЄКТУВАННЯ АПАРАТНОЇ ЧАСТИНИ МОНІТОРИНГУ ФІЗИЧНИХ ПОКАЗНИКІВ ВЕЛОСПОРТСМЕНА

Апаратна частина апаратно-програмного комплексу (АПК) моніторингу фізичних показників є важливим елементом системи, яка забезпечує вимірювання та передачу даних про параметри спортсмена, такі як швидкість, середню швидкість, каденс, пульс, час та інше. Ці параметри відображають стан фізичної підготовки та можуть вказувати на наявність втоми або зрівноважених параметрів, які відповідають чудовій фізичній підготовці. Апаратна частина АПК складається з модуля ESP32 LilyGO S3 T-Display, який виконує функції мікроконтролера та набору датчиків, які підключаються до модуля за допомогою аналогових та цифрових інтерфейсів. Модуль ESP32 LilyGO S3 T-Display є компактним, енергоефективним та функціональним пристроєм, який базується на двоядерному процесорі Tensilica LX7 та має вбудовані модулі Wi-Fi та Bluetooth. Датчики, які використовуються в АПК, мають високу точність, стабільність та довговічність, а також можуть працювати в широкому діапазоні температур та вологості.

Метою другого розділу є проєктування апаратної частини АПК моніторингу фізичних показників, яка забезпечує збір, обробку та передачу даних за допомогою модуля ESP32 LilyGO S3 T-Display та датчиків. Для досягнення цієї мети необхідно виконати наступні завдання:

– вибрати та обґрунтувати компоненти АПК, які відповідають вимогам специфікації;

– розробити схему електричного зв’язку компонентів АПК, яка забезпечує їх сумісність та надійність роботи;

– розробити друкований корпус для АПК, яка забезпечує компактність, ергономічність та захист від зовнішніх впливів;

– змонтувати та налаштувати АПК, перевірити його функціональність та відповідність вимогам.

2.1 Вибір та обґрунтування компонентів АПК

Для проєктування апаратної частини АПК моніторингу фізичних показників спортсмена було вибрано наступні компоненти:

2.1.1 Cycplus C3

Cycplus C3 – це бездротовий датчик 2 в 1, що відстежує швидкість та каденс обертання педалей велосипеда. Він використовує технології Bluetooth 4.0 та ANT+ для передачі даних на велокомп'ютер або смартфон. Датчик має компактний та легкий дизайн, а також водонепроникність за стандартом IP67 (рис. 2.1).

Рисунок 2.1 – Датчики Cycplus C3

Характеристики:

- *тип датчика:* 2 в 1 (каденс та швидкість);
- *технології бездротового зв'язку:* Bluetooth 4.0, ANT+;
- *сумісність:* велокомп'ютери та смартфони з підтримкою Bluetooth 4.0 або ANT+;
- *водонепроникність:* IP67;

- *час роботи:* до 300 годин;
- *тип батареї:* CR2032 (включена);
- *вага:* 9,2 г (з батареєю);
- *розміри:* 38 мм × 29,8 мм × 9,5 мм.

Функції:

- відстежування каденсу обертання педалей (кількість обертів за хвилину);
- відстеження швидкості;
- передача даних на велокомп'ютер або смартфон;
- зміна режиму роботи між реєстрацією каденсу та швидкості (за допомогою світлодіодів);
- водонепроникність.

Переваги:

- компактний та легкий дизайн;
- водонепроникність за стандартом IP67;
- тривалий час роботи;
- простий монтаж;
- сумісність з багатьма велокомп'ютерами та смартфонами.

Недоліки:

- немає вбудованого GPS.

2.1.2 Li-Pol акумулятор

Li-Pol акумулятор – це літій-іонно-полімерний акумулятор, який використовується для живлення електронних пристроїв. Він має компактний та легкий дизайн, а також високу щільність енергії (рис. 2.2).

Рисунок 2.2 – Li-Pol акумулятор

Характеристики:

- *тип:* Li-Pol (літій-іонно-полімерний);
- *ємність:* 950 мА год;
- *номінальна напруга:* 3,7 В;
- *типова енергія***:** 3,515 Вт;
- *робочий діапазон температур:* від мінус 20 °С до 60 °С;
- *кількість циклів:* до 500 циклів зарядки/розрядки;
- *застосування:* широкий спектр електронних пристроїв, таких як смартфони, планшети, фотоапарати, портативні колонки та інші.

Переваги:

- компактний та легкий дизайн;
- висока щільність енергії;
- низький саморозряд;
- довгий термін служби;
- безпечний у використанні.

Недоліки:

– може бути чутливим до екстремальних температур;

– необхідно використовувати спеціальний зарядний пристрій;

– може втратити ємність з часом.

2.1.3 Плата зарядки TP4056

Плата зарядки TP4056 – це недорогий та простий у використанні модуль для зарядки літій-іонних акумуляторів. Вона використовує мікросхему TP4056 для керування процесом зарядки, забезпечуючи безпечну та ефективну зарядку акумулятора (рис. 2.3).

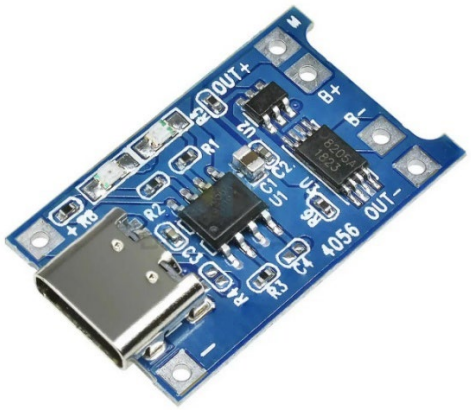


Рисунок 2.3 – Плата зарядки TP4056

Характеристики:

– *вхідна напруга:* 4,5–5,5 В;

– *напруга заряду:* 4,2 В;

– *зарядний струм:* 1 А (максимальний);

– *точність зарядки:* 1,5 %;

– *індикація:* червоний світлодіод (заряджається), зелений світлодіод (заряджено);

– *захист:*

1) захист від перезаряду;

2) захист від перерозряду;

3) захист від короткого замикання;
4) захист від зворотної полярності.

- *розмір:* 17,5 мм × 10 мм × 4 мм.

Застосування:

Плата зарядки TP4056 може використовуватися для зарядки літій-іонних акумуляторів у різних пристроях, таких як:

- power-банки;
- портативні зарядні пристрої;
- світлодіодні ліхтарі;
- іграшки;
- електронні гаджети.

Переваги:

- низька вартість;
- простота використання;
- компактний розмір;
- захисні функції;
- ефективна зарядка.

Недоліки:

- немає можливості регулювати зарядний струм;
- немає вбудованого роз'єму для підключення акумулятора;
- необхідні додаткові компоненти для повної реалізації.

2.1.4 З’єднувальні проводи

З’єднувальні проводи − це проводи, які використовуються для з’єднання компонентів між собою або з джерелами живлення, перемикачами, роз’ємами тощо (рис. 2.4). Проводи складаються з мідного сердечника, який переносить електричний струм, та ізоляційної оболонки, яка захищає від короткого

замикання або ураження електричним струмом. Проводи можуть бути різної довжини, товщини, кольору, матеріалу, гнучкості та типу з’єднання.

Рисунок 2.4 – З’єднувальні проводи 26 AWG

У цьому проєкті використовуються проводи з перетином 26 AWG [20], такі дроти є гнучкими і досить тонкими, що робить їх зручними для використання у вузьких або обмежених просторах, також вони чудово паяються та зроблені з термостійкого матеріалу.

Характеристики:

– *діаметр дроту:* 0,405 мм;

– *максимальний струм:* приблизно до 2,2 ампер (при короткій довжині, довжина впливає на пропускну здатність);

– *тип ізоляції:* зазвичай з ПВХ або іншого термостійкого матеріалу;

– *використання:* з'єднання на макетних платах, малопотужні ланцюги, зв'язок між компонентами.

2.2 Розробка схеми електричного зв’язку компонентів АПК

Схема електричного зв’язку компонентів АПК − це документ, який показує, як підключаються до модуля ESP32 LilyGO S3 дисплей, датчики та АКБ, які використовуються для відображення та вимірювання параметрів. Схема електричного зв’язку дозволяє визначити необхідні типи, кількість та довжину проводів, а також розпізнати можливі помилки або несумісності у

підключенні. Схема електричного зв’язку також є основою для розробки та проєктування друкованого корпусу для АПК.

Схема електричного зв’язку показує, як підключаються кожен компонент до мікроконтролера за допомогою різнокольорових проводів (рис. 2.5–2.6):

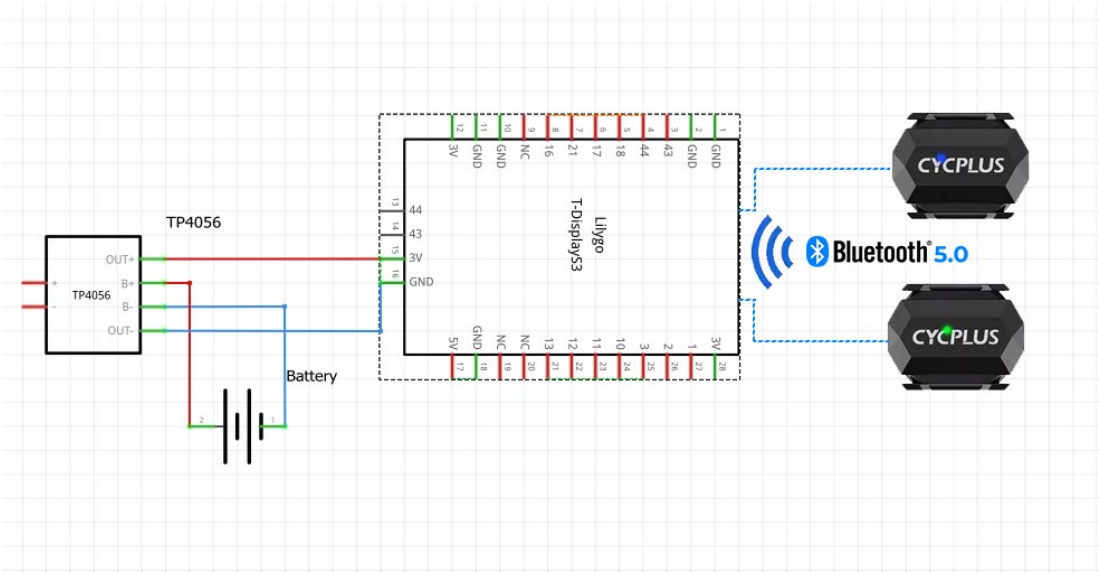


Рисунок 2.5 – Схема електрична принципова

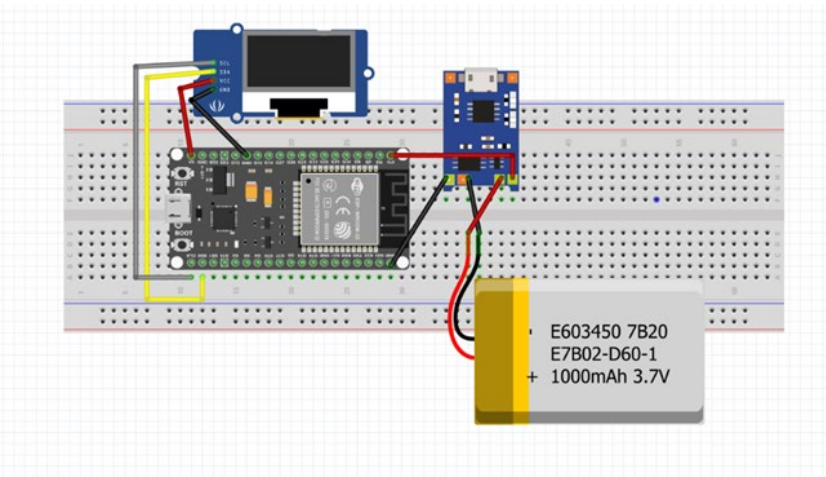


Рисунок 2.6 – Схема підключення АПК

– TFT-дисплей [22] підключається до мікроконтролера за допомогою чотирьох проводів: чорного (GND), червоного (VCC), сірого (SCL) та жовтого (SDA). Чорний провід йде від виводу GND дисплея до виводу GND

мікроконтролера. Червоний провід йде від виводу VCC дисплея до конектора V5. Сірий провід йде від виводу SCL дисплея до виводу SCL мікроконтролера; Жовтий провід йде від виводу SDA дисплея до виводу SDA мікроконтролера;

– мікроконтролер підключається до джерела живлення 3,7 В за допомогою двох проводів: червоного (+) та чорного (-). Червоний провід йде від виводу VIN мікроконтролера до джерела живлення. Чорний провід йде від виводу GND мікроконтролера до джерела живлення;

– датчики швидкості та каденсу підключається до мікроконтролера за допомогою Bluetooth;

– плата зарядки TP4056 – «*OUT+»* на TP4056 підключається до позитивного входу живлення ESP32-S3, зазвичай це контакт 5V або VBAT. «*OUT-«* на TP4056 − підключається до GND (землі) на ESP32-S3;

– підключення АКБ – «*B+»* на TP4056 − підключається до позитивного полюса акумулятора (плюс). «*B-«* на TP4056 − підключається до негативного полюса акумулятора (мінус).

2.3 Розробка друкованого корпусу для АПК

Друкований корпус − це основа для розміщення та з’єднання електронних компонентів, які входять до складу АПК. Друкований корпус забезпечить механічну фіксацію компонентів та захист від зовнішніх впливів. Друкована плата також впливає на функціональні характеристики АПК, такі як компактність, ергономічність та захист від зовнішніх впливів тощо.

Для розробки та моделювання друкованого корпусу для АПК було використано систему автоматизованого проєктування і розрахунку (САПР) – Autodesk Fusion 360 [23], яка дозволяє створювати 3D-моделі, які можна роздрукувати за допомогою 3D-принтеру [24]. Розробка корпусу складалася з нижчезазначених етапів.

2.3.1 Технічні вимоги до корпусу

Перед початком моделювання необхідно було визначити технічні вимоги до корпусу:

– корпус повинен точно відповідати розмірам плати та її компонентів. Для цього враховуються точні виміри плати, дисплея, кнопок, конекторів і отворів для кріплення;

– важливо передбачити отвори для екрану, кнопок, роз'єму живлення (USB Type-C) та контактів GPIO для підключення зовнішніх модулів;

– корпус повинен мати кріплення для фіксації плати, у даному випадку було обрано зробити защіпки на корпусі.

2.3.2 Процес моделювання корпусу в Fusion 360

Для моделювання корпусу використовується програмне забезпечення Autodesk Fusion 360, яке дозволяє створювати точні 3D-моделі та працювати з параметричними даними.

2.3.2.1 Створення основного ескізу

Розробка корпусу починається зі створення базового ескізу на основі розмірів плати ESP32 LilyGO S3 T-Display. У Fusion 360 створюється прямокутник, що відповідає довжині та ширині плати, з урахуванням необхідних відступів для стінок корпусу. Також додаються місця для монтажних отворів, через які буде фіксуватись плата.

2.3.3 Екструзія основи корпусу

Після створення ескізу він витягується за допомогою команди «**Extrude»** на висоту, яка відповідає товщині плати з урахуванням висоти компонентів, розташованих на поверхні (дисплей, конектори, кнопки, акумуляторної батареї). Для захисту компонентів висота стінок корпусу має перевищувати висоту самої плати (рис. 2.7).

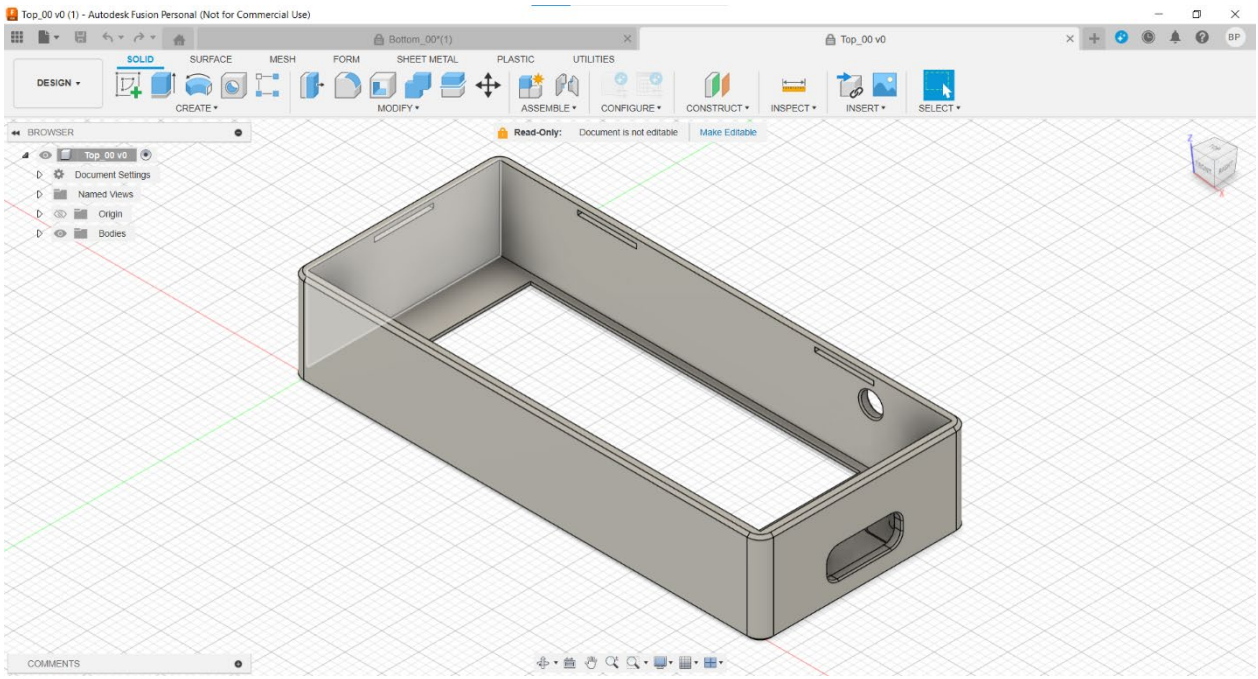


Рисунок 2.7 – 3D-модель корпусу в Autodesk Fusion 360

2.3.4 Створення отворів для компонентів

На основі ескізів у верхній частині корпусу додаються отвори для кнопок, екрану та роз'єму USB Type-C. Отвори розміщуються згідно з розмірами та розташуванням елементів плати, щоб забезпечити зручний доступ до них. Для цього використовуються команди «**Cut**» та «**Extrude**», що дозволяють створювати необхідні вирізи.

2.3.5 Кришка корпусу

Після створення основної частини корпусу моделюється кришка. Вона буде знімною. Ескіз кришки створюється на основі зовнішніх розмірів корпусу (рис. 2.8).

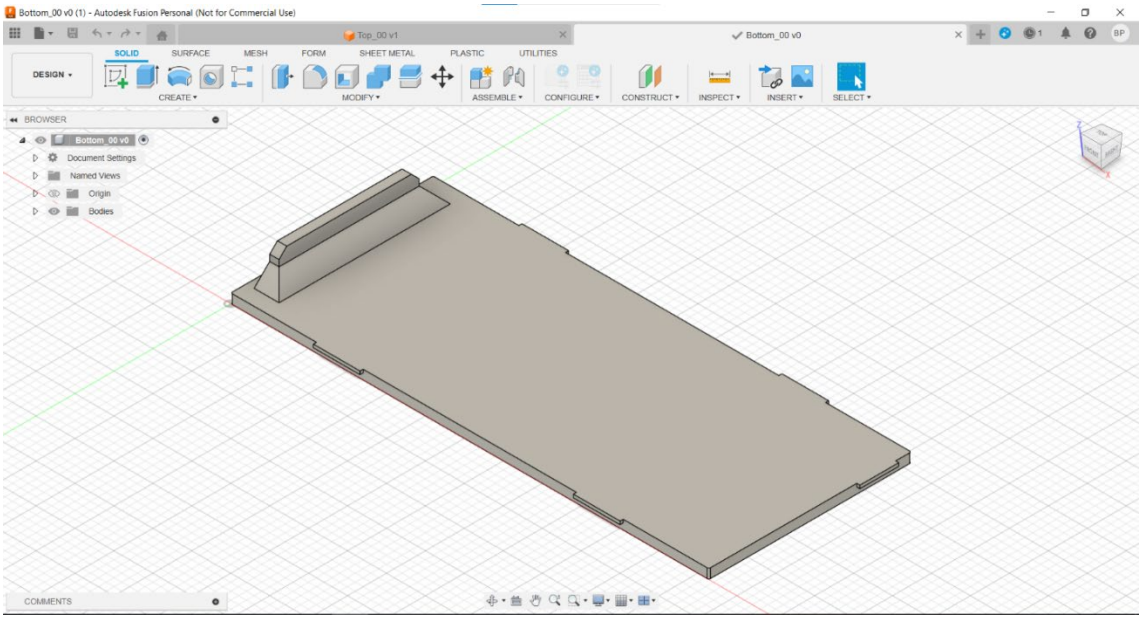


Рисунок 2.8 – 3D-модель кришки корпусу в Autodesk Fusion 360

2.3.6 Фіксуючі елементи

Для надійної фіксації плати всередині корпусу було додано спеціальні защіпки. Це забезпечує стабільність плати всередині корпусу та запобігає її руху.

Також, було розроблено кріплення, за допомогою якого корпус з електронікою буде кріпитись до кріплення велосипеду (рис. 2.9).

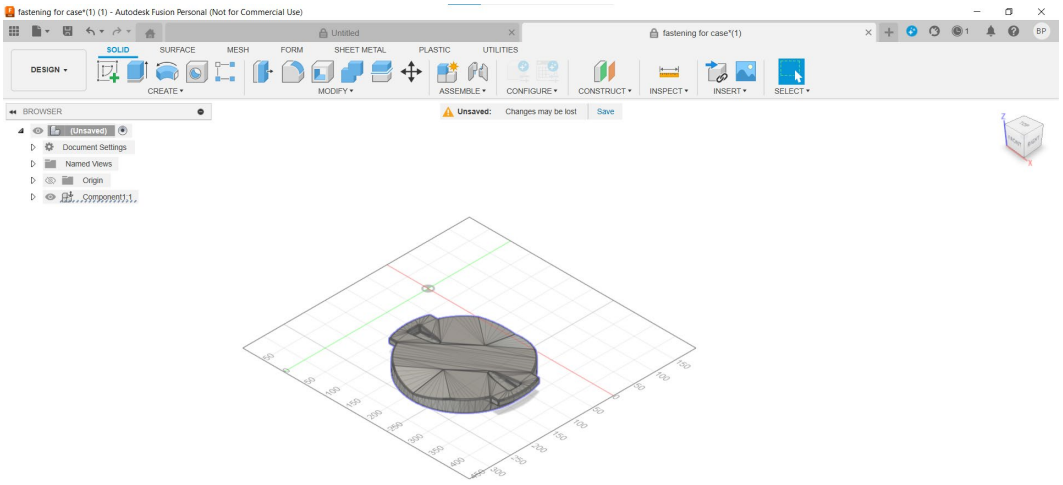


Рисунок 2.9 – 3D-модель кріплення корпусу до велосипеда в Autodesk Fusion 360

2.3.7 Тестування і верифікація моделі

Після завершення моделювання виконується перевірка на відповідність всім технічним вимогам. Модель тестується на правильність розмірів, розташування отворів та інших елементів. За допомогою функції «**Assembly**», у Fusion 360 перевіряється, як окремі частини (корпус і кришка) взаємодіють між собою.

2.3.8 Підготовка до 3D-друку

На завершальному етапі модель експортується у формат «**STL**», який є сумісним із більшістю 3D-принтерів. Далі модель перевіряється у програмі для нарізки «slicer» (рис. 2.10) для визначення оптимальних параметрів друку, таких як товщина шару, заповнення і підтримка.

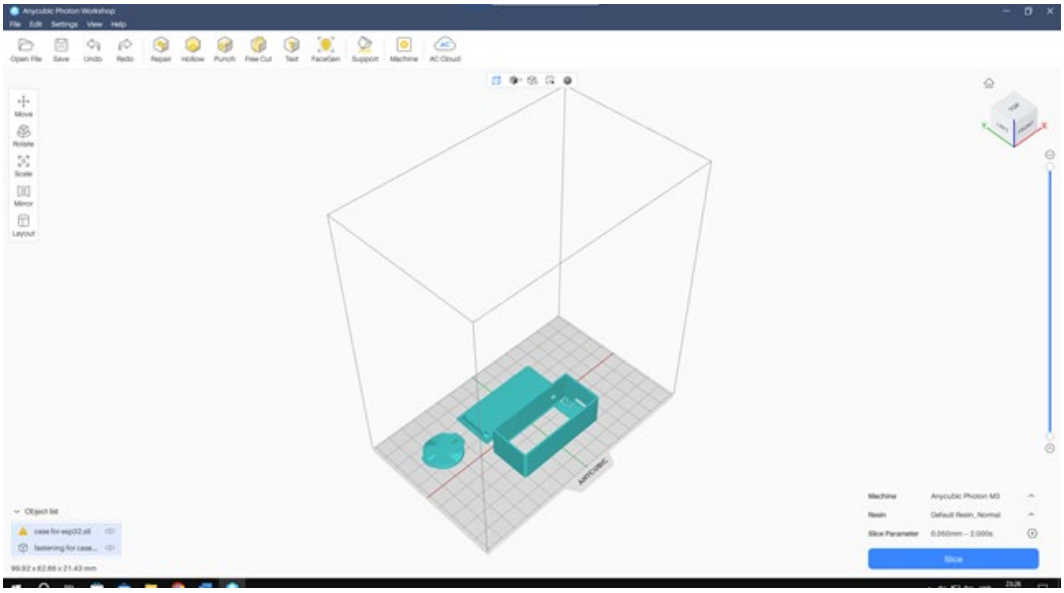


Рисунок 2.10 – Модель в slicer

Результатом моделювання є готова 3D-модель корпусу для плати ESP32 LilyGO S3 T-Display, яка відповідає всім вимогам щодо функціональності, зручності доступу до елементів та надійності фіксації.

2.4 Монтаж та налагодження АПК

Збірка АПК, яка включає в себе пайку компонентів, перевірку якості з’єднань, встановлення плати в корпус, підключення плати до джерела живлення та Bluetooth-датчиків. Для пайки компонентів було використано паяльник, флюс, олово, пінцет, провід, термоусаджувальна трубка та інші інструменти. Встановлення плати та фіксація в корпус виконується за допомогою заздалегідь спроєктованих кріплень (защепок) в корпусі. Підключення плати до джерела живлення та інших пристроїв виконується за допомогою проводів та конектора «jst 1.25 mm 2pin» [25].

Використовуючи плату ESP32 LilyGO S3 T-Display, не потрібно виконувати підключення TFT дисплею, оскільки виробник це вже зробив за нас. Дисплей підключений за допомогою конектора FFC, який підключається відповідно до виводів: GND, VCC, SCL, SDA. Дисплей встановлений на полікарбонатну проставку та зафіксований за допомогою гвинтів.

Збірка модуля зарядки для літій-іонного акумулятору, яка включає в себе пайку виводів АКБ до модуля заряди на базі TP4056 та пайки роз’єму FFC, за допомогою якого підключаємось до плати ESP32. Перед встановленням акумулятор має бути заряджений приблизно на 50 % для безпечного монтажу.

Для підключення датчиків швидкості та каденсу використовуємо бездротову технологію Bluetooth. Такий підхід дозволяє отримувати дані з цих датчиків без використання проводів, що зручно для розробки мобільних і компактних пристроїв для моніторингу активності.

Перевірка роботи АПК, яка включає в себе перевірку відображення даних на TFT-дисплеї, перевірку реакції на натискання кнопок. Також плата повинна бути перевірена на правильність роботи з акумулятором, а саме на функцію зарядки та захисту від перезарядження і глибокого розряду та перевірку стабільності роботи АПК. Для запуску АПК необхідно підключити

джерело живлення 3,7 В до конектора FFC, та натиснути кнопку RESET на платі. Для перевірки відображення даних на TFT дисплеї та роботу радіомодулів, було завантажено тестову прошивку, яка підключається до Wi-Fi та відображає поточний час на дисплеї. Для перевірки стабільності роботи АПК необхідно працювати з АПК протягом певного часу та переконатися, що він не видає помилок, працює стабільно від АКБ та має достатню ємність, не підвисає та не гріється.

Виявлення та усунення несправностей в апаратно-програмному комплексі (АПК) передбачає діагностику та визначення місця пошкодження, заміну або ремонт несправних компонентів, а також повторну перевірку роботи системи. Для діагностики застосовуються методи візуального огляду, вимірювання технічних параметрів, аналіз сигналів, тестування окремих елементів та пошук за кодами помилок.

Висновок до розділу 2

У другому розділі було проведено проєктування апаратної частини АПК моніторингу фізичних показників, яка забезпечує збір, обробку та передачу даних показників спортсмена за допомогою модуля ESP32 LilyGO S3 T-Display та датчиків каденсу та швидкості. Для досягнення цієї мети було виконано наступні завдання:

– вибір та обґрунтування компонентів АПК, які відповідають вимогам специфікації, такі як точність, стабільність, енергоефективність, компактність та сумісність. Було вибрано модуль ESP32 LilyGO S3 T-Display, який має вбудовані модулі Wi-Fi та Bluetooth та TFT дисплей , двоядерний процесор, низьке споживання енергії та велику кількість виводів. Було вибрано плату з TFT дисплеєм, оскільки він має високу контрастність, яскравість, кут огляду та швидкість оновлення і найголовніше має мале енергоспоживання. Було вибрано датчики швидкості та каденсу, які мають високу точність,

стабільність, довговічність, водонепроникність – що дає можливість працювати в будь-яких погодних умовах. Було вибрано тип проводів, які використовуються для з’єднання компонентів між собою та роз’єм для підключення джерелами живлення до плати;

– розробка схеми електричного зв’язку компонентів АПК, яка показує, як підключаються до модуля ESP32 LilyGO S3 T-Display дисплей, датчики та модуль зарядки АКБ, які використовуються для відображення та вимірювання фізичних показників спортсмена. Схема електричного зв’язку дозволяє визначити необхідні типи, кількість та довжину проводів, а також розпізнати можливі помилки або несумісності у підключенні. Схема електричного зв’язку також є основою для розробки друкованого корпусу для АПК;

– розробка друкованого корпусу для АПК, забезпечить механічну фіксацію компонентів та захист від зовнішніх впливів;

– монтаж та налагодження АПК, які включають в себе пайку компонентів, перевірку якості пайки, встановлення плати в корпус, підключення джерела живлення до плати та датчиків для вимірювання показників, перевірку роботи АПК, усунення можливих несправностей АПК.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ ЧАСТИНИ АПК

У третьому розділі було проведено проєктування програмної частини АПК моніторингу фізичних показників спортсмена, яка забезпечує збір, обробку та передачу даних за допомогою модуля ESP32 та датчиків. Було розглянуто вибір мови програмування, розроблено алгоритм роботи, середовища розробки, написано програмний код та його налагодження на мікроконтролер ESP32.

3.1 Вибір та обґрунтування мови програмування та середовища розробки для АПК

Для розробки програмного забезпечення для АПК моніторингу фізичних показників велоспортсмена було обрано мову програмування C++ та середовище розробки − Arduino IDE.

Мова програмування C++ вирізняється високою продуктивністю і дозволяє ефективно використовувати апаратні ресурси [14]. Завдяки своїм можливостям, вона широко застосовується у розробці вбудованих систем, таких як АПК.

Arduino IDE − це відкрите програмне середовище, яке підтримує C++ і надає широкий набір інструментів для програмування мікроконтролерів, зокрема ESP32, що використовується в цьому проєкті. Середовище має зручний і простий інтерфейс, а також велику кількість бібліотек, що полегшують процес розробки [15].

Таким чином, вибір мови C++ та Arduino IDE є логічним і обґрунтованим. Це поєднання забезпечує високу продуктивність, ефективне використання апаратних ресурсів, зручність розробки та доступність необхідних інструментів.

3.2 Розробка алгоритму роботи АПК

Комплекс забезпечує підключення до датчиків швидкості та каденсу через BLE [16], обробку отриманих даних, візуалізацію на дисплеї, збереження даних у файловій системі та їх передачу на платформу Strava [17] через Wi-Fi.

Алгоритм роботи апаратно-програмного комплексу можна розділити на такі ключові етапи:

- *ініціалізація апаратної платформи;*
- *пошук і підключення до BLE-датчиків;*
- *обробка даних від датчиків;*
- *візуалізація на дисплеї;*
- *збереження даних у файловій системі;*
- *формування GPX-файлу;*
- *передача даних на платформу Strava.*

Use Case діаграма (діаграма випадків використання) − це візуальне зображення, яке показує, як користувачі (або інші системи) взаємодіють із системою. Вона є частиною Unified Modeling Language (UML) і використовується для моделювання функціональних вимог до системи.

Основні компоненти Use Case:

а) **актори** − це користувачі або зовнішні системи, які взаємодіють із системою (зазвичай позначаються у вигляді людиноподібної іконки);

б) **випадки використання (Use Cases)** − це функції або завдання, які виконує система (позначаються у вигляді еліпсів із назвою всередині);

в) **система** − це межа (rectangle), яка визначає, що входить до системи. (все всередині прямокутника − це функціональність системи);

г) **зв’язки** − показують, як актори взаємодіють із випадками використання:

1) *асоціація* − це зв’язок між актором і випадком використання;

2) *include* – один випадок використання включає інший (використовується як частина);

3) *extend* – показує додаткову поведінку, що виконується лише за певних умов;

4) *документування проєкту* – важливий елемент технічної документації.

Use Case для АПК велокомп’ютера зображений на рис. 3.1.

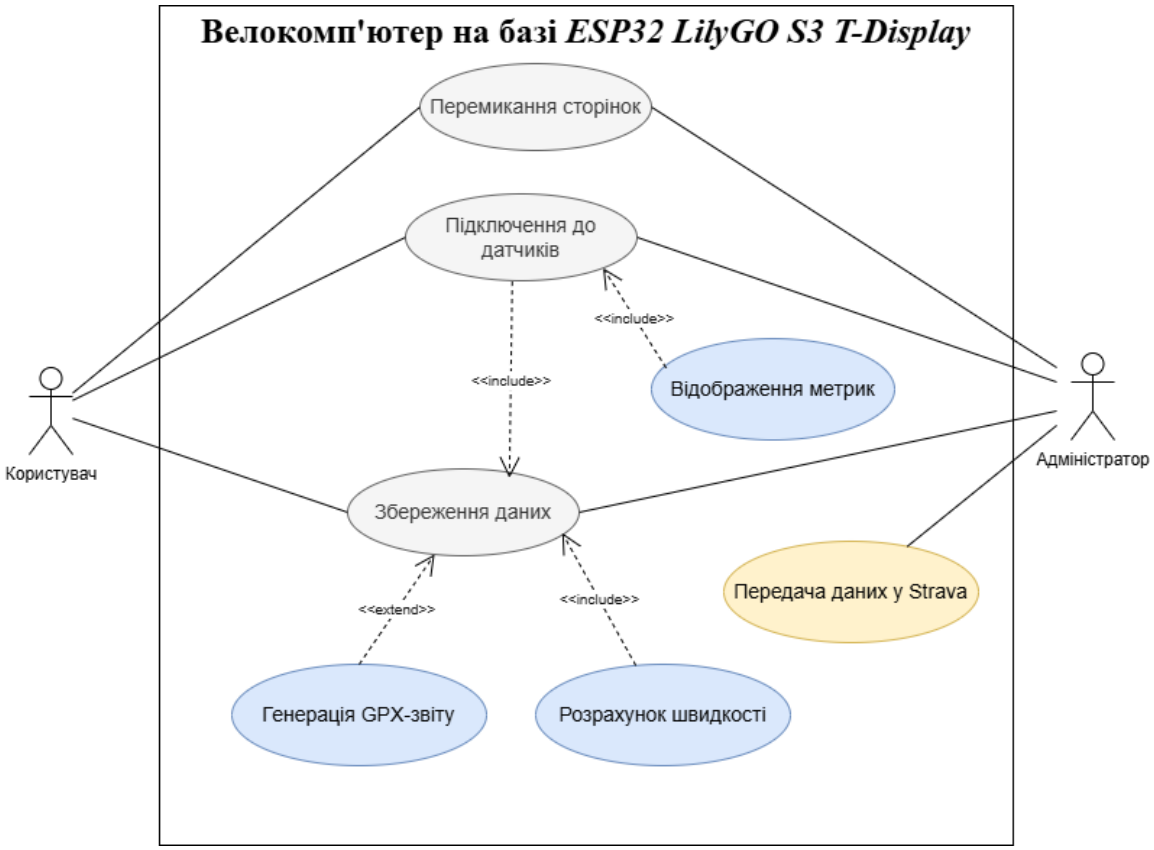


Рисунок 3.1 – Use Case діаграма

3.2.1 Ініціалізація апаратної платформи

На етапі ініціалізації виконується:

- ініціалізація BLE-модуля для пошуку та підключення до датчиків;
- ініціалізація файлової системи SPIFFS [18] для збереження даних;
- ініціалізація WiFi-модуля для підключення до мережі;
- ініціалізація TFT-дисплея для відображення інформації.

```
void setup() {
    Serial.begin(115200);
    tft.init();
    tft.setRotation(1);
    showStatus("Starting BLE Scan...");

    BLEDevice::init("");
    scanner = BLEDevice::getScan();
    scanner->setAdvertisedDeviceCallbacks(new
AdvertisedDeviceCallbacks());
    scanner->setInterval(1600);
    scanner->setWindow(1500);
    scanner->setActiveScan(true);

    if (!SPIFFS.begin(true)) {
        Serial.println("SPIFFS Mount Failed");
        return;
    }
    Serial.println("SPIFFS initialized.");
}
```

Рисунок 3.2 – Лістинг ініціалізація системи

Основний опис дій функції:

- налаштовує серійний порт для діагностики;
- ініціалізує дисплей і виводить повідомлення статусу;
- налаштовує BLE-сканер для пошуку пристроїв;
- ініціалізує файлову систему для роботи з файлами на Flash-пам'яті.

3.2.2 Сканування та підключення до BLE-датчиків

BLE-модуль сканує доступні пристрої. При знаходженні датчиків швидкості та каденсу виконується підключення через відповідні характеристики. Дані зчитуються через функції зворотного виклику (*notifyCadenceCallback* і *notifySpeedCallback*).

```
      class AdvertisedDeviceCallbacks : public
BLEAdvertisedDeviceCallbacks {
          void onResult(BLEAdvertisedDevice advertisedDevice) {
              String address =
advertisedDevice.getAddress().toString().c_str();
              if (address.equals(CADENCE_SENSOR_ADDRESS) &&
!cadenceConnected) {
                  cadenceDevice = new
BLEAdvertisedDevice(advertisedDevice);
              } else if (address.equals(SPEED_SENSOR_ADDRESS) &&
!speedConnected) {
                  speedDevice = new
BLEAdvertisedDevice(advertisedDevice);
              }
          }
};
```

Рисунок 3.3 – Лістинг функції пошуку і підключення до BLE-датчиків

Основний опис дій функції:

- перевіряє знайдені BLE-пристрої;
- якщо знайдений пристрій відповідає одному з очікуваних сенсорів (за MAC-адресою) і ще не підключений, зберігає його дані в об'єкті cadenceDevice або speedDevice.

3.2.3 Обробка даних від датчиків

Отримані дані з BLE-датчиків використовуються для розрахунку швидкості та каденсу. Зчитування відбувається за допомогою спеціальних BLE-характеристик.

Каденс – це частота обертання педалей велосипеда, яка вимірюється в обертах за хвилину (RPM, англ. revolutions per minute). Для розрахунку каденсу використовуються дані, отримані з BLE-датчику каденсу, які включають:

- **кумулятивну кількість обертів** *(cumulativeCrankRevcumulativeCrankRev)* − загальна кількість обертів, яку зафіксував датчик.

– **час останнього оберту педалей** *(lastCrankTimelastCrankTime)* − час у частках секунди (1/1024 секунди) від моменту запуску датчика.

Формули розрахунку:

− **різниця обертів педалей між двома зчитуваннями**:

$$\Delta rotations = cumulativeCrankRev - prevCrankRev \quad (3.1)$$

де, prevCrankRev*prevCrankRev* – попереднє значення кумулятивної кількості обертів.

− **різниця часу між двома зчитуваннями**:

$$\Delta time = lastCrankTime - prevCrankTime \quad (3.2)$$

де, prevCrankTime*prevCrankTime* – попередній час останнього оберту.

− **час у хвилинах**:

$$time_{mins} = \frac{\Delta time}{1024 \times 60} \quad (3.3)$$

− **каденс**:

$$cadence = \frac{\Delta rotations}{time_{mins}} \quad (3.4)$$

Таким чином, каденс обчислюється на основі кількості обертів педалей (ΔrotationsΔ*rotations*) і часу між ними (timemins*timemins*).

```
    void notifyCadenceCallback(BLERemoteCharacteristic*
pBLERemoteCharacteristic, uint8_t* data, size_t length, bool
isNotify) {
        if (data == nullptr || length < 5) return;
        int cumulativeCrankRev = (data[1] | (data[2] << 8));
        int lastCrankTime = (data[3] | (data[4] << 8));
        int deltaRotations = cumulativeCrankRev - prevCrankRev;
        int timeDelta = lastCrankTime - prevCrankTime;

        if (timeDelta > 0) {
            double timeMins = ((double)timeDelta) / 1024.0 / 60.0;
            cadence = (int)(((double)deltaRotations) / timeMins);
        }
        prevCrankRev = cumulativeCrankRev;
        prevCrankTime = lastCrankTime;}
```

Рисунок 3.4 – Лістинг функції обробки даних з датчику каденсу

Основний опис дій функції:

− отримує дані каденсу від BLE-сенсора;

− розраховує поточну частоту обертання педалей на основі змін у кумулятивній кількості обертів і часу;

− оновлює значення каденсу для подальшого використання (наприклад, для відображення на дисплеї).

Швидкість – це швидкість руху велосипеда, яка вимірюється в кілометрах за годину (км/год). Для розрахунку швидкості використовуються дані з BLE-датчику швидкості, які включають:

− **кумулятивна кількість обертів колеса** *(cumulativeWheelRevcumulativeWheelRev)* − загальна кількість обертів колеса, яку зафіксував датчик;

− **час останнього оберту колеса** *(lastWheelTimelastWheelTime)* − час у частках секунди (1/1024 секунди).

Формули розрахунку:

− **різниця обертів колеса між двома зчитуваннями**:

$$\Delta wheelRev = cumulativeWheelRev - prevWheelRev \quad (3.5)$$

де, prevWheelRev*prevWheelRev* – попереднє значення кумулятивної кількості обертів.

− **різниця часу між двома зчитуваннями**:

$$\Delta time = lastWheelTime - prevWheelTime \quad (3.6)$$

де, prevWheelTime*prevWheelTime* – попередній час останнього оберту.

− **час у секундах**:

$$time_{\text{seconds}} = \frac{\Delta time}{1024} \quad (3.7)$$

− **швидкість у метрах за секунду**:

$$speed_{m/s} = \frac{\Delta wheelRev \times wheelCircumference}{time_{\text{seconds}}} \quad (3.8)$$

де, wheelCircumference*wheelCircumference* – окружність колеса у метрах (наприклад, для колеса 700×23c окружність становить приблизно 2,105 м).

- **швидкість у кілометрах за годину**:

$$speed_{km/h} = speed_{m/s} \times 3{,}6 \quad (3.9)$$

Таким чином, швидкість розраховується на основі кількості обертів колеса (ΔwheelRevΔwheelRev), окружності колеса (wheelCircumferencewheelCircumference) і часу між ними (timesecondstimeseconds).

```
     Void notifySpeedCallback(BLERemoteCharacteristic*
pBLERemoteCharacteristic, uint8_t* data, size_t length, bool
isNotify) {
          int cumulativeWheelRev = (data[1] | (data[2] << 8));
          int lastWheelTime = (data[5] | (data[6] << 8));
          int deltaWheelRev = cumulativeWheelRev - prevWheelRev;
          int timeDelta = lastWheelTime - prevWheelTime;
          if (timeDelta > 0) {
           double wheelCircumference = 2.105;
           double timeSeconds = (double)timeDelta / 1024.0;
              speed = ((deltaWheelRev * wheelCircumference) /
timeSeconds) * 3.6;}
          prevWheelRev = cumulativeWheelRev;
          prevWheelTime = lastWheelTime;}
```

Рисунок 3.5 – Лістинг функції обробки даних з датчику швидкості

Основний опис дій функції:

- отримує дані швидкості від BLE-сенсора;
- обчислює швидкість велосипеда в км/год на основі змін у кількості обертів колеса та часу;
- оновлює значення для подальшого використання (наприклад, для виведення на дисплей).

3.2.4 Візуалізація на дисплеї

Дисплей використовується для відображення параметрів:

- час тренування;
- швидкість (поточна, середня, максимальна);
- каденс;
- кількість спалених калорій.

Також, інтерфейс має кілька сторінок, між якими можна перемикатись.

Ця функція (рис. 3.6) забезпечує інтуїтивне відображення даних, дозволяючи перемикатися між різними показниками на екрані дисплея.

```
void updateDisplay() {
    if (currentPage == 0) {
        tft.fillScreen(TFT_BLACK);
        tft.setCursor(0, 0);
        tft.print("Speed: ");
        tft.print(speed);
        tft.print(" km/h");
        tft.setCursor(0, 30);
        tft.print("Cadence: ");
        tft.print(cadence);
        tft.print(" rpm");
    } else if (currentPage == 1) {
        tft.fillScreen(TFT_BLACK);
        tft.print("Distance: ");
        tft.print(totalDistance);
        tft.print(" km");
    } else if (currentPage == 2) {
        tft.fillScreen(TFT_BLACK);
        tft.print("Calories: ");
        tft.print(caloriesBurned);
        tft.print(" kcal");
    }
}
```

Рисунок 3.6 – Лістинг функції відображення даних на дисплеї

Основний опис дій функції:

- динамічне оновлення дисплея залежно від активної сторінки;

– відображення різних даних (швидкість, каденс, відстань, калорії) для користувача.

3.2.5 Збереження даних у файловій системі

Дані зберігаються у вигляді CSV-файлу. Кожен запис містить час, швидкість, каденс, потужність і пройдену відстань.

```
     void logData(float speed, int cadence, int power, float
distance) {
         fs::File file = SPIFFS.open("/data.csv", FILE_APPEND);
         file.printf("%ld,%.2f,%d,%d,%.2f\n", millis(), speed,
cadence, power, distance);
         file.close();}
```

Рисунок 3.7 – Лістинг функції зберігання даних у вигляді CSV-файлу

Основний опис дій функції:

– збереження ключових параметрів (швидкість, каденс, потужність, відстань) у вигляді структурованого CSV-файлу;

– збирання даних для подальшого аналізу на комп'ютері (наприклад, у таблицях Excel або аналізу програмами).

3.2.6 Формування GPX-файлу

Збережені дані з CSV-файлу використовуються для формування GPX-файлу, сумісного з платформою Strava.

```
     void createGpxFile() {
       fs::File file = SPIFFS.open("/activity.gpx", FILE_WRITE);
       file.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
       file.println("<gpx version=\"1.1\"><trk><trkseg>");
         fs::File dataFile = SPIFFS.open("/data.csv", FILE_READ);
         while (dataFile.available()) {
             String line = dataFile.readStringUntil('\n');
             file.printf("<trkpt lat=\"0.0\"
lon=\"0.0\"><time>%ld</time></trkpt>\n", millis());}
         file.println("</trkseg></trk></gpx>");
         file.close();}
```

Рисунок 3.8 – Лістинг функції формування GPX-файлу

Основний опис дій функції:

− запис активності в GPX-файл, який можна використовувати для аналізу активності, побудови треків на карті або експорту в інші системи;

− CSV-дані з файлу data.csv обробляються для генерації треків у GPX.

3.2.7 Передача даних на платформу Strava

За допомогою Wi-Fi дані з GPX-файлу передаються на платформу Strava через HTTP-запит.

Для передачі файлу на платформу Strava використовується HTTP POST-запит. Відправка виконується у форматі multipart/form-data, що дозволяє передати файл разом із метаданими. Для автентифікації використовується токен доступу Strava.

```
 HTTPClient http;
      http.begin("https://www.strava.com/api/v3/uploads");
      String boundary = "------------------------ESP32Boundary";
      http.addHeader("Authorization", "Bearer " + stravaToken);
      http.addHeader("Content-Type", "multipart/form-data; boundary="
+ boundary);
      String bodyStart = "--" + boundary + "\r\n";
      bodyStart += "Content-Disposition: form-data; name=\"file\";
filename=\"activity.gpx\"\r\n";
      bodyStart += "Content-Type: application/gpx+xml\r\n\r\n";
      String bodyEnd = "\r\n--" + boundary + "--\r\n";
      WiFiClient* stream = http.getStreamPtr();
      stream->print(bodyStart);
      // Передача файлу
      while (file.available()) {stream->write(file.read());}
      stream->print(bodyEnd);
      file.close();
```

Рисунок 3.9 – Лістинг функції передачі даних на платформу Strava

Основний опис дій функції:

− завантажування GPX-файлу на сервер Strava через API;

− ефективно передавати великий файл у форматі multipart/form-data завдяки потоковій передачі.

3.3 Бібліотеки для реалізації програмної частини АПК

Для розробки програмного забезпечення для системи моніторингу фізичних показників велоспортсмена на базі модуля ESP32, було використано мову програмування C++ та середовище розробки Arduino IDE. У межах цього проєкту застосовано значну кількість бібліотек, які забезпечують функціональність різних компонентів апаратно-програмного комплексу. Нижче наведено перелік основних бібліотек, які використовуються в проєкті, разом із їхнім докладним описом:

Arduino.h – є ключовою бібліотекою для роботи з платформою Arduino, яка надає доступ до апаратних функцій, зокрема цифрових і аналогових входів/виходів, таймерів, переривань тощо [26]. Вона попередньо встановлена в Arduino IDE і забезпечує використання таких функцій, як pinMode, digitalWrite, analogRead, delay, Serial та інших, необхідних для управління платою ESP32 і її пінами.

WiFi.h – це бібліотека Arduino, розроблена для роботи з Wi-Fi модулем ESP32 [27]. Вона надає можливість підключатися до бездротових мереж і працювати з протоколами TCP/IP. Бібліотека входить до стандартного набору Arduino IDE. Завдяки їй можна використовувати такі функції, як WiFi.begin, WiFi.status, WiFiClient.connect, WiFiClient.print та інші, які забезпечують підключення до Wi-Fi і передачу даних, наприклад, на сервер STRAVA.

BLEDevice.h– використовується для роботи з Bluetooth Low Energy (BLE) на пристроях ESP32 [28]. Вона є частиною екосистеми ESP32 і надає функціонал для створення BLE-застосунків, включаючи роль клієнта, сервера або обох одночасно. Відповідна для розробки IoT-проєктів, таких як розумні датчики, передача даних через BLE-канал або віддалене управління. Часто використовується разом із мобільними застосунками для взаємодії з ESP32.

TFT_eSPI.h – бібліотека призначена для роботи з TFT-дисплеями на основі контролерів, таких як ILI9341, ST7735, ST7789, SSD1351 тощо [29].

Вона оптимізована для мікроконтролерів ESP32 та ESP8266 і забезпечує швидке та зручне управління графічними інтерфейсами. Перед використанням необхідно налаштувати файл User_Setup.h, де вказується: модель дисплея (наприклад, #define ILI9341_DRIVER), SPI-піни для підключення (MOSI, SCLK, CS, DC, RST), орієнтація екрану, кольори, роздільна здатність.

SPIFFS.h *(SPI Flash File System)*– це файлова система, розроблена спеціально для роботи з флеш-пам’яттю, яка дозволяє зберігати файли безпосередньо в пам’яті мікроконтролера [30]. У контексті платформ ESP8266 та ESP32, ця бібліотека дозволяє працювати з файлами, записаними в пам’яті флеш, як із файлами на звичайному накопичувачі. За допомогою неї, можна зберігати конфігураційні файли, веб-сторінки, налаштування, або інші невеликі файли.

FS.h – це базова бібліотека для роботи з файловими системами в екосистемі Arduino, зокрема на платформах ESP8266 та ESP32 [31]. Вона надає інтерфейс для доступу до файлових систем, таких як SPIFFS, LittleFS або FATFS, дозволяючи працювати з файлами на флеш-пам’яті або зовнішніх накопичувачах.

HTTPClient.h – це бібліотека для роботи з протоколом HTTP на мікроконтролерах, таких як ESP32 та ESP8266. Вона дозволяє створювати HTTP-запити (GET, POST, PUT, DELETE та інші) і обробляти відповіді, що є зручно для інтеграції з вебсервісами, REST API або для роботи з веб-серверами. Бібліотека є ідеальним вибором для проєктів, які потребують інтеграції з інтернет-сервісами. Вона підходить як для простих GET/POST-запитів, так і для складних API-взаємодій [32].

Висновок до розділу 3

У третьому розділі роботи було виконано проєктування програмної частини АПК для моніторингу фізичних показників спортсмена.

Для створення програмного забезпечення було обрано мову програмування C++ та середовище розробки Arduino IDE. Це забезпечило високу продуктивність, гнучкість і доступність необхідних бібліотек для роботи з апаратними компонентами ESP32. Використання C++ дозволило оптимізувати роботу з обмеженими ресурсами мікроконтролера, а Arduino IDE сприяло зручній і швидкій розробці завдяки інтуїтивному інтерфейсу.

Також було удосконалено алгоритм збору та передачі даних – фізичних показників велоспортсменів – у реальному часі за рахунок інтеграції з Інтернетом речей. До алгоритму роботи комплексу після ініціалізації апаратної платформи, було додано пошук і підключення до BLE-датчиків, розташованих на різних частинах велосипеду, а також безпосередньо на спортсмені. Після обробки даних здійснюється їх візуалізація на TFT-дисплеї, збереження у файловій системі та передача на платформу Strava. Виконане таким чином удосконалення дозволяє використовувати розроблену систему для різних дисциплін велоспорту та в реальних тренувальних умовах

Було реалізовано механізми збору та обробки даних з BLE-датчиків швидкості та каденсу. Алгоритми обчислення швидкості та частоти обертання педалей дозволять підвищити точність отриманих результатів.

Забезпечено автоматичне завантаження GPX-файлів на платформу Strava через Wi-Fi. Це розширює функціонал системи та надає користувачам можливість відстежувати свої результати у популярному спортивному сервісі.

Розроблене ПЗ забезпечує ефективну роботу апаратно-програмного комплексу, відповідно до поставлених завдань. Реалізація продемонструвала, що комплекс є готовим до використання для моніторингу фізичних показників спортсменів, що підтверджує доцільність обраних технічних рішень.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ. АНАЛІЗ РЕЗУЛЬТАТІВ РОЗРОБКИ

4.1 USER GID по завантаженню та налагодження програмного коду на АПК

4.1.1 Завантаження репозиторію з GitHub

Для розгортання програмного коду в систему на базі ESP32, необхідно клонувати репозиторій з GitHub [21]. Для цього необхідно виконати нижче наведені кроки:

4.1.1.1 Отримання посилання на репозиторій:

1) потрібно виконати переході за посиланням на репозиторій GitHub;

2) обрати меню **Code** та скопіювати HTTPS або SSH посилання.

4.1.1.2 Клонування репозиторію:

1) відкрити термінал (або Git Bash на Windows);

2) за допомогою команди ***cd*** перейти у бажану директорію (папку) на комп’ютері (рис. 4.1);

```
cd path_to_your_directory
```

Рисунок 4.1 – Вибір директорії для клонування репозиторію

3) виконати команду для клонування репозиторію (рис. 4.2).

```
git clone https://github.com/username/project-name.git
```

Рисунок 4.2 – Команда для клонування репозитарію

4.1.2 Налаштування середовища розробки

Для подальшої роботи з кодом та взаємодії з платаю ESP32 необхідно встановити середовище розробки Arduino IDE та виконати налаштування плати.

4.1.2.1 Встановлення Arduino IDE:

1) потрібно завантажити Arduino IDE з офіційного сайту *https://www.arduino.cc/en/software;*

2) встановити IDE відповідно до інструкцій для вашої операційної системи (рис. 4.3).

Рисунок 4.3 – Arduino IDE

4.1.2.2 Налаштування плати ESP32:

1) відкрийте **File → Preferences**;

2) у полі «***Additional Boards Manager URLs»*** потрібно вставити посилання на файл з метаінформацією для завантаження та встановлення плати ESP32 у середовищі Arduino IDE:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json ;

3) перейти до **Tools → Board → Boards Manager** та знайти там «ESP32» і натиснути **Install**.

4.1.2.3 Встановлення необхідних бібліотек:

1) потрібно відкрити **Tools → Manage Libraries**;

2) встановити наступні бібліотеки :

- TFT_eSPI;
- WiFi.h;
- BLEDevice;
- SPIFFS;
- FS.h;
- HTTPClient.

4.1.3 Завантаження коду на ESP32

4.1.3.1 Підключення ESP32:

1) для початку потрібно підключити ESP32 LilyGO S3 T-Display до комп’ютера через USB-кабель;

2) відкрити **Tools → Port** та обрати відповідний COM-порт.

4.1.3.2 Налаштування параметрів плати:

1) потрібно обрати плату: **Tools → Board → ESP32 → LilyGO T-Display S3**;

2) наступним кроком потрібно задати такі параметри (табл. 4.1):

Таблиця 4.1 – Параметри для плати

Arduino IDE Setting	Value
Board	**LilyGO T-Display S3**
Port	Your port
USB CDC On Boot	Enable
CPU Frequency	240MHZ(WiFi)
Core Debug Level	None
USB DFU On Boot	Disable
Erase All Flash Before Sketch Upload	Disable
Events Run On	Core1
Flash Mode	QIO 80MHZ
Flash Size	**16MB(128Mb)**
Arduino Runs On	Core1
USB Firmware MSC On Boot	Disable
Partition Scheme	**16M Flash(3M APP/9.9MB FATFS)**
PSRAM	**OPI PSRAM**
Upload Mode	**UART0/Hardware CDC**
Upload Speed	921600
USB Mode	**CDC and JTAG**

4.1.3.3 Завантаження коду:

1) відкрити файл *«mycadence-arduino.ino»* у Arduino IDE;

2) натиснути кнопку **Upload**;

3) якщо з’являється повідомлення про помилку завантаження, потрібно утримувати кнопку **«BOOT»** на платі під час завантаження.

4.1.4 Налагодження програмного коду

4.1.4.1 Моніторинг серійного порту:

1) потрібно відкрити **Tools → Serial Monitor**;

2) налаштуйте швидкість передачі даних на 115200 BAUD;

3) серійний монітор показуватиме стан роботи пристрою (підключення до BLE, Wi-Fi, обчислення параметрів).

4.1.4.2 Діагностика підключення BLE-датчиків:

1) у серійному моніторі перевірте повідомлення:

а) «Found Cadence Sensor» − датчик каденсу знайдено;

б) «Found Speed Sensor» − датчик швидкості знайдено.

2) якщо датчики не знаходяться, перевірте правильність MAC-адрес у файлі:

```
#define CADENCE_SENSOR_ADDRESS "d8:a6:1e:63:b4:03"
#define SPEED_SENSOR_ADDRESS "e1:45:3d:a6:f6:d0"
```

Рисунок 4.4 – Встановлення MAC-адреси датчиків

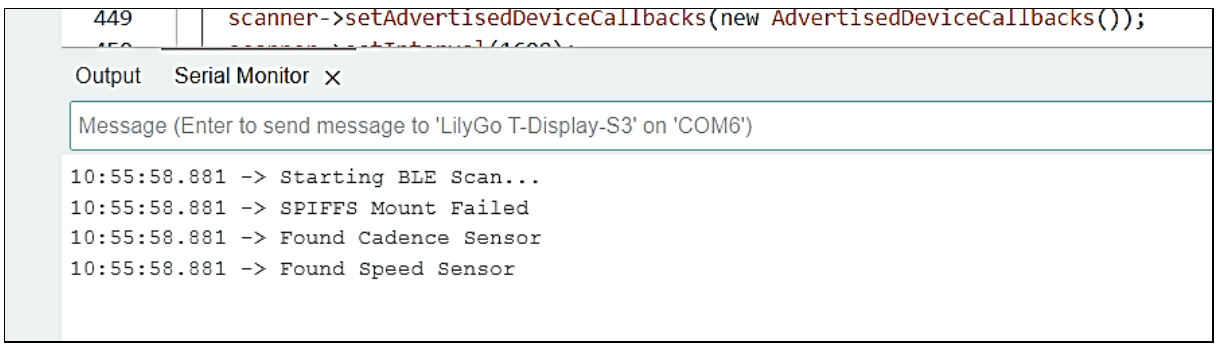


Рисунок 4.5 – Перевірка пошуку датчиків

4.1.4.3 Діагностика зчитування даних

Наступним кроком, потрібно перевірити, чи передаються дані з датчиків, перейшовши до вкладки «Serial Monitor» (рис. 4.6).

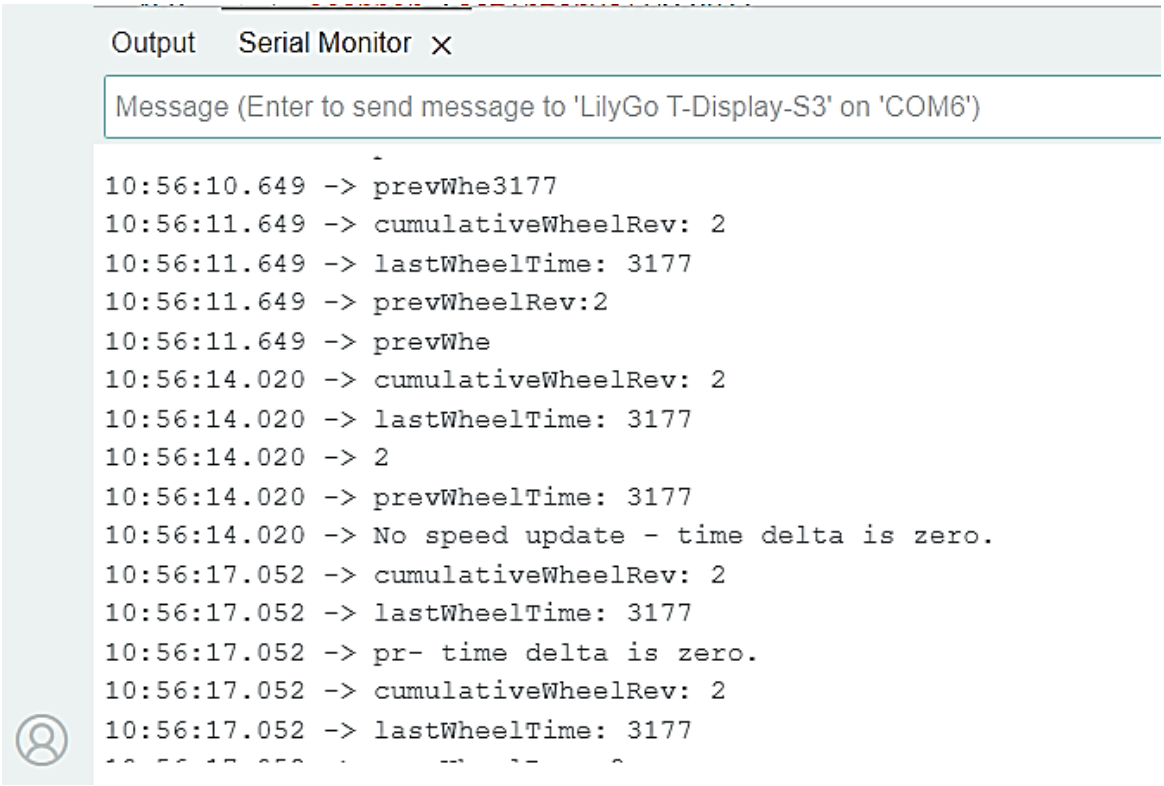


Рисунок 4.6 – Перевірка надходження даних з датчиків

Як можна побачити з рис. 4.6, дані з датчиків успішно надходять до плати, тому можна перейти до наступного пункту – тестування АПК та перевірки правильності обробки отриманих та виводу даних на дисплей.

4.2 Тестування апаратної та програмної частини АПК

Тестування роботи апаратно-програмного комплексу (АПК) є важливим етапом для перевірки його функціональності та відповідності заданим технічним вимогам. На цьому етапі здійснюється оцінка всіх ключових компонентів системи, таких як підключення до датчиків, обчислення параметрів, відображення інформації на дисплеї, збереження даних та їх передача.

4.2.1 Тестування підключення до датчиків

Потрібно перевірити з’єднання з BLE-датчиками (каденсу та швидкості). По-перше, потрібно увімкнути велокомп’ютер та на дисплеї з’явиться напис «Starting BLE scan…» (рис. 4.7).

Рисунок 4.7 – Повідомлення підчас включення велокомп’ютера

Наступним кроком потрібно у ввімкнути та перевести датчики каденсу та швидкості у відповідні режими (рис. 4.8), після ініціалізації датчиків платою на дисплеї з’явитись відображення основних функцій комп’ютера – це означатиме, що датчики були успішно підключені (рис. 4.9).

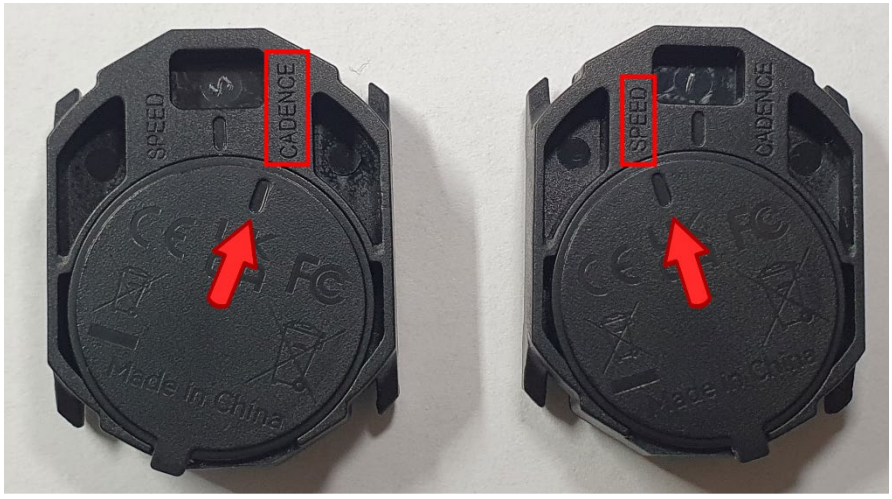


Рисунок 4.8 – Перемикання датчиків у відповідний режим роботи

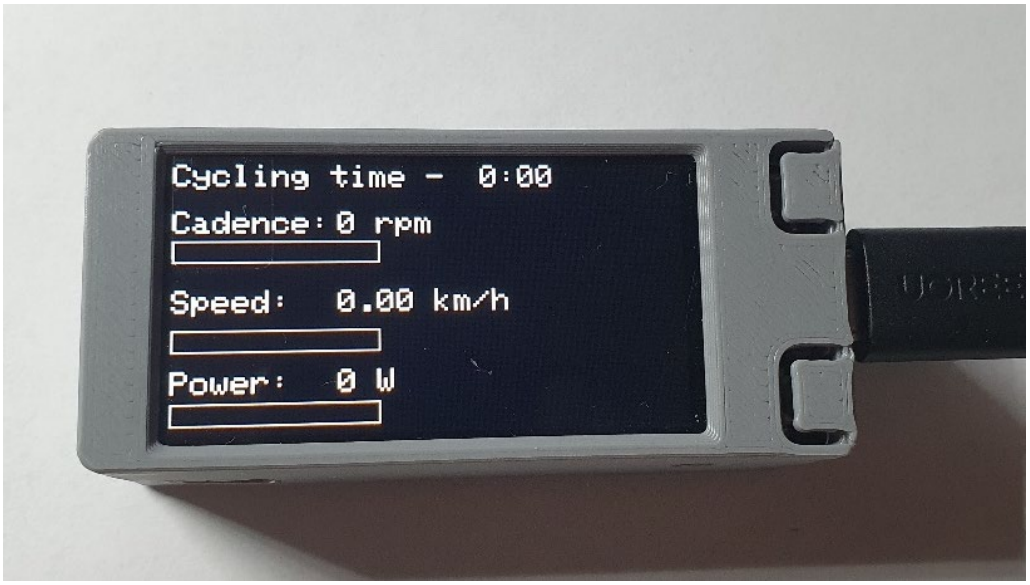


Рисунок 4.9 – Головне меню велокомп’ютера

Ще однією з головних перевірок є тест на перепідключення датчиків після режиму «сну», оскільки спортсмен під час тренувань може зупинитись на певний час. Важливо, щоб датчики автоматично і без збоїв перепідключались до плати та продовжували надавати дані для подальшого запису показників тренувань. Дану особливість було враховано під час розробки програмної частини та успішно протестовано у готовій збірці АПК.

4.2.2 Тестування АПК

Для перевірки роботи АПК, потрібно закріпити датчики каденсу та швидкості у відповідних місцях на велосипеді. Датчик «Cadence» – потрібно закріпити на шатуні велосипеду, а датчик «Speed» – потрібно закріпити на втулці переднього або заднього колеса велосипеда, як це показано на рис. 4.10 – 4.11.

Рисунок 4.10 – Приклад закріплення датчику каденсу на шатуні

Рисунок 4.11 – Приклад закріплення датчику швидкості на втулці колеса

Для контролю правильності відображення показників на розробленому АПК, було взято проводовий велокомп’ютер *«Techwell Delta-2»* (рис. 4.12).

Рисунок 4.14 – Велокомп’ютер Techwell Delta-2

Провівши тест для порівняння точності фірмового провідного велокомп’ютера Delta-2 та розробленого велокомп’ютера на базі ESP32 LilyGO S3 T-Display, було отримані результати продемонстровані на рис. 4.15–4.17.

Рисунок 4.15 – Результати тесту

Рисунок 4.16 – Результати тесту

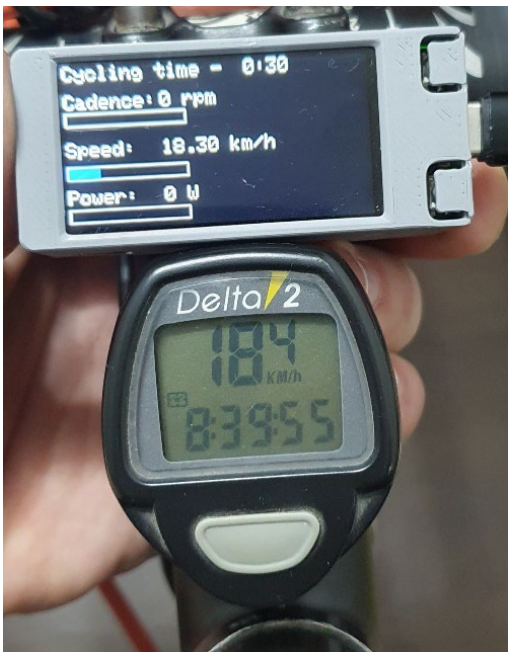


Рисунок 4.17 – Результати тесту

З результатів досліджень, можна зробити висновок, що розроблений безпровідний велокомп’ютер на базі ESP32 має високу точність у вимірах та має конкурентну спроможність з провідними велокомп’ютерами. До недоліків можна віднести те, що іноді відбувається мінімальна затримка у передачі даних, але цей недолік відноситься до всіх бездротових систем.

Також, було перевірено перемикання сторінок меню велокомп’ютера та відображення і розрахунок загальної дистанції, середньої швидкості та максимальної швидкості АПК (рис. 4.18).

Рисунок 4.18 – Відображення сторінок меню

Останньою перевіркою системи, були відправка та отримання даних на платформі Strava (рис. 4.19–4.20).

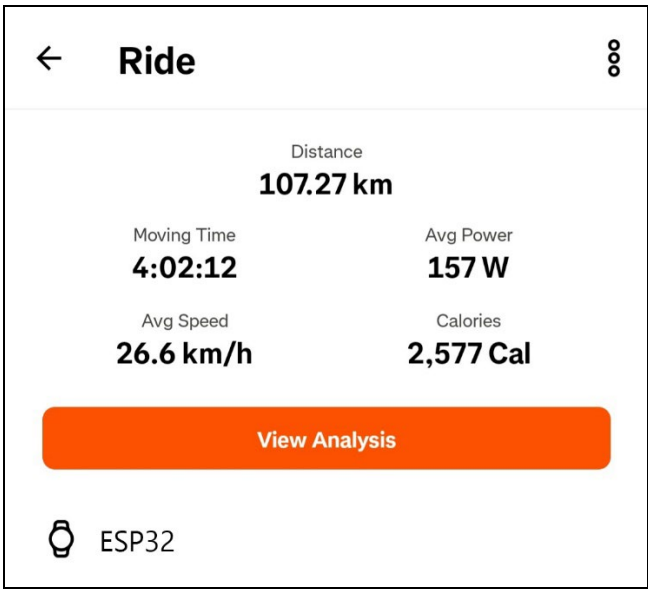


Рисунок 4.19 – Відображення головних показників

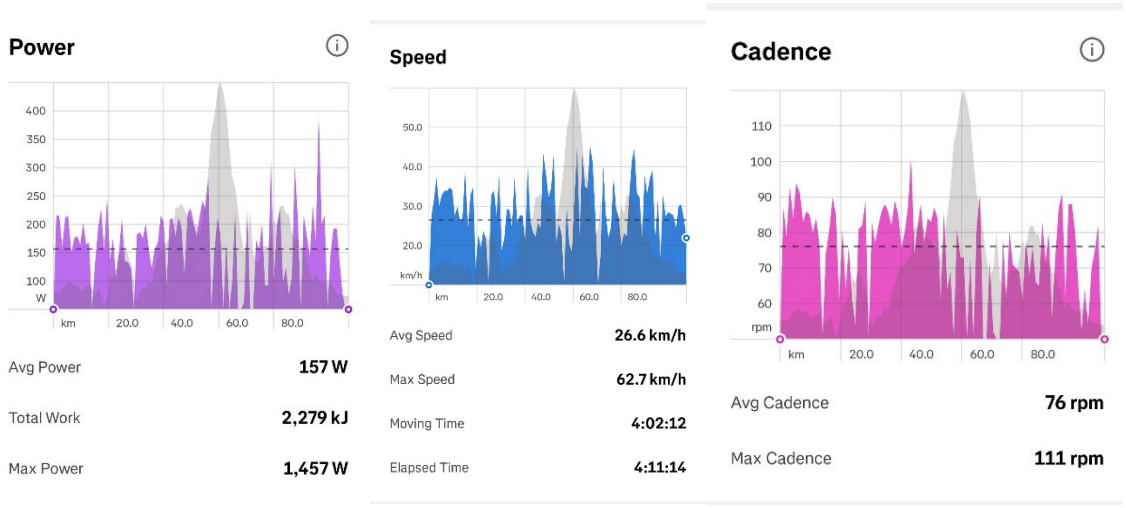


Рисунок 4.20 – Відображення графіків з отриманих даних

Як можна побачити на рис. 4.19–4.20, дані надходять до Strava та успішно відображаються на сервері.

Висновок до розділу 4

У четвертому розділі проведено експериментальні дослідження АПК для моніторингу фізичних показників спортсмена та здійснено аналіз результатів його розробки. На основі виконаної роботи було оцінено функціональність та ефективність системи.

Розроблений User Guide надає чіткі інструкції щодо завантаження та налаштування програмного забезпечення. Усі етапи – від клонування репозиторію з GitHub до налагодження роботи з платою ESP32 – докладно описані, що спрощує впровадження системи навіть для користувачів із базовим рівнем технічної підготовки. Інструкції з налаштування Arduino IDE, встановлення необхідних бібліотек і роботи з серійним монітором забезпечують стабільну роботу комплексу та полегшують усунення можливих помилок.

Тестування підтвердило, що система коректно працює з BLE-датчиками каденсу та швидкості. Перевірка автоматичного перепідключення після

виходу датчиків із режиму сну показала, що комплекс відповідає вимогам практичного використання під час тренувань. Це дозволяє спортсмену зосередитися на процесі, не відволікаючись на технічні аспекти.

Результати тестування апаратно-програмного комплексу в реальних умовах свідчать про високу точність вимірювань. У порівнянні з провідним велокомп’ютером Techwell Delta-2 розроблений комплекс демонструє аналогічні результати, що підтверджує його конкурентоспроможність. Мінімальна затримка передачі даних, характерна для бездротових систем, не вплинула на загальну функціональність комплексу.

Інтерфейс системи успішно виконує свої завдання, забезпечуючи відображення ключових показників, таких як швидкість, каденс, дистанція та спалені калорії. Додатково перевірено коректність перемикання сторінок меню та розрахунків, що підтвердило їх відповідність технічним вимогам. Також було протестовано передачу та відображення даних зібраних під час тренувань з апаратно-програмного комплексу на платформу STRAVA.

Аналіз результатів показав, що розроблений АПК на базі ESP32 LilyGO S3 T-Display є ефективним, точним і зручним у використанні. Його функціональні можливості та якість роботи підтверджують відповідність поставленим вимогам, а використання бездротових технологій забезпечує додаткові переваги в умовах реального використання. Комплекс має потенціал для подальшого вдосконалення та впровадження в практичну діяльність.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено апаратно-програмний комплекс (АПК) для моніторингу фізичних показників велоспортсменів, який базується на використанні сучасних технологій IoT. Результати проведених досліджень і розробок підтверджують, що створена система відповідає поставленим вимогам, забезпечує високу точність і функціональність, а також є зручною та ефективною у використанні.

Для досягнення поставленої мети розробки зазначеного комплексу було виконано такі **завдання:**

− розглянуто сучасні системи моніторингу та збору фізичних показників спортсмена;

− проаналізовано недоліки існуючих систем;

− розроблено апаратну частину АПК велокомп'ютера, яка складається з модуля ESP32 LilyGO S3 та датчиків швидкості та каденсу;

− розроблено програмну частину АПК велокомп'ютера, яка складається з прошивки для модуля ESP32 LilyGO S3 T-Display, обробки даних, застосунок для відображення та керування комплексом;

− розроблено 3D-модель для зборки АПК.

На основі аналізу аналогів і існуючих рішень було обґрунтовано вибір апаратної платформи ESP32 LilyGO S3 T-Display, яка завдяки інтегрованим модулям Wi-Fi і Bluetooth, TFT-дисплею та низькому енергоспоживанню є оптимальним рішенням для реалізації даного проєкту. Сформована специфікація вимог дозволила визначити цілі, задачі та компоненти, необхідні для створення АПК.

Проведено проєктування апаратної частини комплексу, включаючи вибір компонентів, розробку електричної схеми та створення друкованого корпусу. Було виконано монтаж і налагодження системи, що забезпечило стабільну роботу апаратної частини та її відповідність технічним вимогам.

Розроблено програмну частину комплексу з використанням мови програмування C++ та середовища Arduino IDE. Було вдосконалено алгоритми збору та передачі даних за рахунок інтеграції з Інтернетом речей. До алгоритму роботи комплексу було додано пошук і підключення до BLE-датчиків, розташованих на різних частинах велосипеду, а також безпосередньо на спортсмені. Після обробки отриманих даних та їх відображення на дисплеї реалізовано автоматичне завантаження на платформу Strava, що розширює практичне застосування комплексу в різних дисциплінах велоспорту.

Тестування системи в реальних умовах підтвердило її працездатність і зручність використання. У порівнянні з провідним велокомп'ютером Techwell Delta-2 розроблений комплекс показав аналогічні результати, що підтверджує його конкурентоспроможність.

Розроблений АПК для моніторингу фізичних показників велоспортсменів є сучасним, доступним та ефективним інструментом для отримання оперативної та достовірної інформації про підготовку спортсмена, виявлення та запобігання їх травмування, підтримки здоров'я та покращення спортивних результатів. Його функціональність, точність вимірювань та можливість подальшого вдосконалення відкривають перспективи для впровадження у реальну практику.

Наукова новизна роботи полягає в удосконаленні алгоритму збору та передачі даних – фізичних показників велоспортсменів – у реальному часі за рахунок інтеграції з Інтернетом речей, що дозволяє використовувати розроблену систему для різних дисциплін велоспорту та в реальних тренувальних умовах.

Основні положення та результати роботи **опубліковані** у журналі «Вісник Хмельницького національного університету» кат. Б [2]. На розроблену комп’ютерну програму «Cyclist Monitoring System» отримано Свідоцтво про реєстрацію авторського права на твір № 123174 (2024) [12].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Daam Van. Water level reeth. The Economics of Professional Road Cycling. Springer Nature, 2022. P. 363–385. ISBN 978-3-031-11258-4.

2. Пилипчук Б. В., Журавська І. М. Технології інтернету речей для моніторингу фізичних навантажень велоспортсменів. *Вісник Хмельницького національного університету. Серія: Технічні науки.* 2024, Т. 333, № 2. С. 329–337. DOI: 0.31891/2307-5732-2024-333-2-52.

3. Global recommendations on physical activity for health. World Health Organization. Geneva, Switzerland, 2020. URL: http://www.who.int/ncds/prevention/physical-activity/en/ (Last accessed: 01.10.2024).

4. What is FTP and why does it matter for cyclists? BikeRadar. URL: https://www.bikeradar.com/advice/fitness-and-training/what-is-ftp-and-why-it-matters-for-cyclists/ (Last accessed: 02.10.2024).

5. De Marchi D., Schena F. The evaluation of mountain bike cross-country performance. *International Journal of Sports Physiology and Performance*. 2021. No. 2 (4). P. 360–372. DOI: 10.1123/ijspp.2021-0541.

6. Bike computer on ESP32. URL: https://health-ua.com/cardiology/mizdisciplinarni-problemi/61895-veloergometrya-upraktichnj-meditcin (Last accessed: 03.10.2024).

7. Bicycle ergometry in practical medicine. URL: https://github.com/jamesmontemagno/mycadence-arduino?tab=readme-ov-file (Last accessed: 03.10.2024).

8. Garmin Edge 530 Bundle. URL: https://garmin-official.com/p/1053120558-velonavigator-edge-530-bundle/?o=IdqJM12uYI1w7gYwUF6puQ==&gad_source=1&gclid=Cj0KCQjwpP63BhDYARIsAOQkATZFzmj26wBd44xbs3fSYV1hlZ93oZlNPB5b-Da8NY_tnjxql4YVY_IaAurTEALw_wcB (Last accessed: 03.10.2024).

9. Wahoo ELEMNT Roam. URL: https://bikestory.com.ua/ua/velokompyuter-wahoo-elemnt-roam-gps/?gad_source=1&gclid=Cj0KCQjwpP63BhDYARIsAOQkATa9o_8c3fma7OJd32ZNP1XMHYN834YRH-62FzrzAPXfZuM-j_RGs1MaAhH1EALw_wcB (Last accessed: 03.10.2024).

10. ESP32 LilyGO S3 T-Display. URL: https://www.lilygo.cc/products/t-display-s3-ca?_pos=7&_sid=d9b4a890c&_ss=r (Last accessed: 04.10.2024).

11. CYCPLUS Speed Sensor. URL: https://www.cycplus.com/products/cycplus-smart-speed-cadence-sensor?srsltid=AfmBOorzWkAWC-ZyzjhAnuFDCzmuPyonh5b8v4TIn30_SEU0ISS-P1YK (Last accessed: 04.10.2024).

12. Свідоцтво про реєстрацію авторського права на твір 123174. Комп’ютерна програма «Cyclist Monitoring System» : комп’ютерна програма / Б. В. Пилипчук, І. М. Журавська ; дата реєстр. 24.01.2024, Бюл. № 80.

13. Пилипчук Б. В., Журавська І. М. Моніторинг фізичних навантажень велоспортсмена. *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. 47–49.

14. Marius Bancila, Modern C++ Programming Cookbook: Master modern C++ including the latest features of C++23 with 140+ practical recipes , Packt Publishing, 2024.

15. Drymonitis, A. (2024). Introduction to Arduino. In: Digital Electronics for Musicians. Maker Innovations Series. Apress, Berkeley, CA. URL: https://doi.org/10.1007/979-8-8688-0394-9_2 (Last accessed: 25.10.2024).

16. Cameron, N. (2023). BLE Beacons. In: ESP32 Formats and Communication. Maker Innovations Series. Apress, Berkeley, CA. URL: https://doi.org/10.1007/978-1-4842-9376-8_5 (Last accessed: 27.10.2024).

17. Strava − Record. Sweat. Share. Kudos. URL: https://www.strava.com/features?hl=en-EN (Last accessed: 27.10.2024).

18. Cameron, N. (2021). OTA and saving data to EEPROM, SPIFFS, and Excel. In: Electronics Projects with the ESP8266 and ESP32. Apress, Berkeley, CA. URL: https://doi.org/10.1007/978-1-4842-6336-5_20 (Last accessed: 27.10.2024).

19. Strava − Record. Sweat. Share. Kudos. URL: https://www.strava.com/features?hl=en-EN (Last accessed: 27.10.2024).

20. AWG − American wire gauge. URL: https://keyboard.net.ua/stati/scho-take-awg.html (Last accessed: 27.10.2024).

21. Що таке GitHub і як з ним працювати. URL: https://training.qatestlab.com/blog/technical-articles/what-is-github-and-how-to-work/ (Last accessed: 08.11.2024).

22. K. Nakazawa, M. Katayama, N. Kondo, M. Okamoto, K. Yano and M. Hijikigawa, "The 8.6 inch-diagonal TFT-LCDs of symmetric sub-dot design," Conference Record of the 1991 International Display Research Conference, San Diego, CA, USA, 1991, pp. 119-121, doi: 10.1109/DISPL.1991.167447.

23. Timmis, H. (2021). Modeling with Fusion 360. In: Practical Arduino Engineering. Apress, Berkeley, CA. URL: https://doi.org/10.1007/978-1-4842-6852-0_3 (Last accessed: 11.11.2024).

24. Van Looy, A. (2024). 3D Printing. In: From Emerging Technologies to Business Opportunities. Springer, Cham. URL: https://doi.org/10.1007/978-3-031-59770-1_8 (Last accessed: 11.11.2024).

25. JST PH 1.25mm connector with wires for self-soldering. URL: https://www.a7lab.in.ua/home/electronics-for-3d-printers/jst-ph1-25-2p-connector-on-wire/?v=5269f4d75f5b (Last accessed: 11.11.2024).

26. Cores Arduino.h at master. URL: https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/Arduino.h (Last accessed: 15.11.2024).

27. Library WiFi.h. URL: https://docs.arduino.cc/libraries/wifi/ (Last accessed: 15.11.2024).

28. BLEDevice.h issues. URL: https://forum.arduino.cc/t/bledevice-h-issues/1236482 (Last accessed: 15.11.2024).

29. TFT_eSPI library. URL: https://docs.arduino.cc/libraries/tft_espi/ (Last accessed: 15.11.2024).

30. SPIFFS in ESP32. URL: https://www.tutorialspoint.com/esp32_for_iot/esp32_for_iot_spiffs_storage.htm (Last accessed: 15.11.2024).

31. ESP32: Write Data to a File (LittleFS) – Arduino. Tutorials. URL: https://randomnerdtutorials.com/esp32-write-data-littlefs-arduino/ (Last accessed: 15.11.2024).

32. HttpClient − Arduino Reference. URL: https://reference.arduino.cc/reference/en/libraries/httpclient/ (Last accessed: 15.11.2024).

ДОДАТОК А

Код програми

File «mycadence-arduino.ino»:

```
#include <TFT_eSPI.h>
#include "Arduino.h"
#include "BLEDevice.h"
#include "device.h"
#include "SPIFFS.h"
#include <WiFi.h>
#include <HTTPClient.h>
#include <FS.h>

#define CADENCE_SENSOR_ADDRESS "d8:a6:1e:63:b4:03"
#define SPEED_SENSOR_ADDRESS "e1:45:3d:a6:f6:d0"
#define LED_BUILTIN 2

const char* ssid = "Filial_ept";
const char* password = "corsair125#";
String stravaToken = "3bf823f8fc50682d6a63ac9ac758b49a612119bf";

// Статуси підключення до кожного з датчиків
static boolean cadenceConnected = false;
static boolean speedConnected = false;

static BLERemoteCharacteristic* cadenceCharacteristic = nullptr;
static BLERemoteCharacteristic* speedCharacteristic = nullptr;

static BLEAdvertisedDevice* cadenceDevice = nullptr;
static BLEAdvertisedDevice* speedDevice = nullptr;

static BLEClient* cadenceClient = nullptr;
static BLEClient* speedClient = nullptr;

static BLEScan* scanner = nullptr;

static int cadence = 0;
static double speed = 0.0;
static int power = 0;
static unsigned long runtime = 0;
static unsigned long last_millis = 0;

// Параметри для сторінок
int currentPage = 0;
#define BUTTON_PIN 0
```

```
// Параметри для обчислення пробігу та середньої швидкості
double totalDistance = 0.0;  // Загальний пробіг у км
double totalRuntime = 0.0;   // Загальний час у годинах
double maxSpeed_total = 0.0; // Максимальна швидкість у км/год
double caloriesBurned = 0.0; // Кількість спалених калорій

#define maxCadence 120
#define maxSpeed 55

TFT_eSPI tft = TFT_eSPI();

void showStatus(const char* message) {
    tft.fillScreen(TFT_BLACK);
    tft.setTextSize(2);
    tft.setTextColor(TFT_WHITE);
    tft.setCursor(0, 0);
    tft.println(message);
    Serial.println(message);
}

void notifyCadenceCallback(BLERemoteCharacteristic* pBLERemoteCharacteristic,
uint8_t* data, size_t length, bool isNotify) {
    if (data == nullptr || length < 5) return; // Перевірка валідності даних

    // Кількість обертів (для каденсу)
    int cumulativeCrankRev = (data[1] | (data[2] << 8));
    int lastCrankTime = (data[3] | (data[4] << 8));

    // Розрахунок каденсу
    static int prevCrankRev = 0;
    static int prevCrankTime = 0;

    int deltaRotations = cumulativeCrankRev - prevCrankRev;
    int timeDelta = lastCrankTime - prevCrankTime;
    if (deltaRotations < 0) {
        deltaRotations += 65535;  // Обробка переповнення
    }
    if (timeDelta < 0) {
        timeDelta += 65535;  // Обробка переповнення
    }

    if (timeDelta > 0) {
        double timeMins = ((double)timeDelta) / 1024.0 / 60.0;
        cadence = (int)(((double)deltaRotations) / timeMins);
    } else {
        cadence = 0;
    }
```

```
    prevCrankRev = cumulativeCrankRev;
    prevCrankTime = lastCrankTime;
}

void notifySpeedCallback(BLERemoteCharacteristic* pBLERemoteCharacteristic,
uint8_t* data, size_t length, bool isNotify) {
    if (data == nullptr || length < 7) {
        Serial.println("Invalid speed data received.");
        return;
    }

    bool hasWheel = (data[0] & 0x01);  // Перевірка наявності даних про колесо
    int wheelRevIndex = 1;
    int lastWheelTimeIndex = 5;  // Використовуємо індекс 5 для lastWheelTime

    if (hasWheel) {
        // Отримуємо дані обертів колеса та час останнього оберту
        int cumulativeWheelRev = (data[wheelRevIndex] | (data[wheelRevIndex + 1]
<< 8));
        int lastWheelTime = (data[lastWheelTimeIndex] | (data[lastWheelTimeIndex
+ 1] << 8));

        // Змінні для збереження попередніх значень
        static int prevWheelRev = -1;
        static int prevWheelTime = -1;

        // Обчислюємо різницю обертів та часу
        int deltaWheelRev = cumulativeWheelRev - prevWheelRev;
        int timeDelta = lastWheelTime - prevWheelTime;

        // Діагностика
        Serial.print("cumulativeWheelRev: ");
        Serial.println(cumulativeWheelRev);
        Serial.print("lastWheelTime: ");
        Serial.println(lastWheelTime);
        Serial.print("prevWheelRev: ");
        Serial.println(prevWheelRev);
        Serial.print("prevWheelTime: ");
        Serial.println(prevWheelTime);

        // Обробка переповнення обертів та часу
        if (deltaWheelRev < 0) {
            deltaWheelRev += 65536;  // обертів
        }
        if (timeDelta < 0) {
            timeDelta += 65536;  // часу
        }
```

```
        // Розрахунок швидкості, якщо зміна часу відбувається
        if (timeDelta > 0) {
            double wheelCircumference = 2.105;  // Колесо, 2.105 м (можливо,
потрібно скоригувати)
            double timeSeconds = (double)timeDelta / 1024.0;  // час у секундах

            // Розрахунок швидкості (в м/с) і перетворення в км/год
            speed = ((deltaWheelRev * wheelCircumference) / timeSeconds) * 3.6;

            // Обмеження швидкості для реалістичних значень
            if (speed > 120.0) {  // Обмеження на максимальну швидкість 120
км/год
                Serial.println("Unrealistic speed value detected, setting speed
to 0.");
                speed = 0.0;
            } else {
                Serial.print("Speed calculated: ");
                Serial.println(speed);
            }
        } else {
            // Немає зміни часу
            speed = 0.0;
            Serial.println("No speed update - time delta is zero.");
        }

        // Оновлення попередніх значень
        prevWheelRev = cumulativeWheelRev;
        prevWheelTime = lastWheelTime;
    } else {
        speed = 0.0;
        Serial.println("No wheel data available.");
    }
}

bool connectToDevice(BLEAdvertisedDevice* device, BLEClient*& client,
BLERemoteCharacteristic*& characteristic, BLEUUID serviceUUID, BLEUUID charUUID,
void (*notifyCallback)(BLERemoteCharacteristic*, uint8_t*, size_t, bool)) {

    client = BLEDevice::createClient();
    client->connect(device);

    if (!client->isConnected()) {
        delete client;
        client = nullptr;
        return false;
    }

    BLERemoteService* remoteService = client->getService(serviceUUID);
```

```
    if (remoteService == nullptr) {
        client->disconnect();
        delete client;
        client = nullptr;
        return false;
    }

    characteristic = remoteService->getCharacteristic(charUUID);
    if (characteristic == nullptr) {
        client->disconnect();
        delete client;
        client = nullptr;
        return false;
    }

    characteristic->registerForNotify(notifyCallback);
    return true;
}

// Клас для обробки знайдених пристроїв
class AdvertisedDeviceCallbacks : public BLEAdvertisedDeviceCallbacks {
    void onResult(BLEAdvertisedDevice advertisedDevice) {
        String address = advertisedDevice.getAddress().toString().c_str();

        if (address.equals(CADENCE_SENSOR_ADDRESS) && !cadenceConnected) {
            Serial.println("Found Cadence Sensor");
            cadenceDevice = new BLEAdvertisedDevice(advertisedDevice);
        } else if (address.equals(SPEED_SENSOR_ADDRESS) && !speedConnected) {
            Serial.println("Found Speed Sensor");
            speedDevice = new BLEAdvertisedDevice(advertisedDevice);
        }
    }
};

int calculatePower(int cadence, int resistance) {
    if (cadence == 0 || resistance == 0) return 0;
    double a = 3.737;   // Коефіцієнт, який можна налаштувати
    double b1 = 1.023;  // Коефіцієнт для каденсу
    double b2 = 1.096;  // Коефіцієнт для опору
    return (int)(a * pow(b1, cadence) * pow(b2, resistance));
}

void logData(float speed, int cadence, int power, float distance) {
    fs::File file = SPIFFS.open("/data.csv", FILE_APPEND);  // Використовуйте
`fs::File`
    if (!file) {
        Serial.println("Failed to open file for appending");
        return;}
```

```
    file.printf("%ld,%.2f,%d,%d,%.2f\n", millis(), speed, cadence, power,
distance);
    file.close();
    Serial.println("Data written to file.");
}

void createGpxFile() {
    fs::File file = SPIFFS.open("/activity.gpx", FILE_WRITE);  // Використовуйте
`fs::File`
    if (!file) {
        Serial.println("Failed to open file for writing");
        return;
    }

    file.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
    file.println("<gpx version=\"1.1\" creator=\"ESP32\">");
    file.println("<trk><name>Activity</name><trkseg>");

    fs::File dataFile = SPIFFS.open("/data.csv", FILE_READ);  // Використовуйте
`fs::File`
    while (dataFile.available()) {
        // Зчитуємо дані, записані у CSV і форматуємо для GPX
        String line = dataFile.readStringUntil('\n');
        float speed, distance;
        int cadence, power;
        long timestamp;
        sscanf(line.c_str(), "%ld,%f,%d,%d,%f", &timestamp, &speed, &cadence,
&power, &distance);

        file.printf("<trkpt lat=\"0.0\"
lon=\"0.0\"><ele>0</ele><time>%ld</time>", timestamp);
        file.printf("<extensions><speed>%.2f</speed><cadence>%d</cadence><power>%
d</power></extensions></trkpt>\n", speed, cadence, power);
    }

    dataFile.close();
    file.println("</trkseg></trk></gpx>");
    file.close();
}

void uploadToStrava() {
    // Перевірка підключення до Wi-Fi
    if (WiFi.status() != WL_CONNECTED) {
        Serial.println("WiFi not connected, trying to connect...");
        WiFi.begin(ssid, password);
        while (WiFi.status() != WL_CONNECTED) {
            delay(1000);
```

```
            Serial.println("Connecting to WiFi...");
        }
        Serial.println("Connected to WiFi.");
    }

    // Відкриття GPX-файлу для читання
    fs::File file = SPIFFS.open("/activity.gpx", FILE_READ);
    if (!file) {
        Serial.println("Failed to open GPX file for reading.");
        return;
    }

    // Початок HTTP-запиту
    HTTPClient http;
    http.begin("https://www.strava.com/api/v3/uploads");

    // Формування запиту з параметрами
    String boundary = "------------------------ESP32Boundary";
    http.addHeader("Authorization", "Bearer " + stravaToken);
    http.addHeader("Content-Type", "multipart/form-data; boundary=" + boundary);

    // Формування тіла запиту
    String bodyStart = "--" + boundary + "\r\n";
    bodyStart += "Content-Disposition: form-data; name=\"file\";
filename=\"activity.gpx\"\r\n";
    bodyStart += "Content-Type: application/gpx+xml\r\n\r\n";

    String bodyEnd = "\r\n--" + boundary + "--\r\n";

    // Відправлення початкової частини
    WiFiClient* stream = http.getStreamPtr();
    stream->print(bodyStart);

    // Передача GPX-файлу
    while (file.available()) {
        stream->write(file.read());
    }
    file.close();

    // Завершення запиту
    stream->print(bodyEnd);

    // Отримання відповіді
    int httpResponseCode = http.POST("");
    if (httpResponseCode > 0) {
        String response = http.getString();
        Serial.printf("HTTP Response code: %d\n", httpResponseCode);
        Serial.println("Response:");
```

```
        Serial.println(response);
    } else {
        Serial.printf("Error: %d\n", httpResponseCode);
    }

    http.end();
}


void updateDisplay() {
    char buf[10];  // Оголошення буфера для використання у всій функції

    if (currentPage == 0) {
        // Сторінка 0: Поточні показники
        tft.fillScreen(TFT_BLACK);
        tft.setCursor(0, 0);
        tft.setTextColor(TFT_WHITE);
        tft.print("Cycling time - ");

        char buf[5];
        const int minutes = int(runtime / 60000);
        itoa(minutes, buf, 10);
        tft.setCursor(190, 0);
        tft.print(buf);
        tft.print(":");

        const int seconds = int((runtime % 60000) / 1000);
        if (seconds < 10) {
            buf[0] = '0';
            itoa(seconds, &buf[1], 10);
        } else {
            itoa(seconds, buf, 10);
        }
        tft.print(buf);

        // Відображення каденсу
        tft.setCursor(0, 30);
        tft.print("Cadence: ");
        itoa(cadence, buf, 10);
        tft.setCursor(100, 30);
        tft.print(buf);
        tft.print(" rpm");

        uint8_t progress = (cadence >= maxCadence) ? 100 : (uint8_t)((100 *
cadence) / maxCadence);
        tft.fillRect(0, 49, 126, 14, TFT_BLACK);
        tft.fillRect(0, 49, progress, 14, TFT_GREEN);
        tft.drawRect(0, 49, 126, 14, TFT_WHITE);
```

```
        // Відображення швидкості
        tft.setCursor(0, 80);
        tft.print("Speed: ");
        dtostrf(speed, 4, 2, buf);
        tft.setCursor(100, 80);
        tft.print(buf);
        tft.print(" km/h");

        uint8_t speedProgress = (speed >= maxSpeed) ? 100 : (uint8_t)((100 *
speed) / maxSpeed);
        tft.fillRect(0, 105, 126, 14, TFT_BLACK);
        tft.fillRect(0, 105, speedProgress, 14, TFT_BLUE);
        tft.drawRect(0, 105, 126, 14, TFT_WHITE);

        // Відображення потужності
        tft.setCursor(0, 130);
        tft.print("Power: ");
        itoa(power, buf, 10);
        tft.setCursor(100, 130);
        tft.print(buf);
        tft.print(" W");

        const int maxPower = 1000;
        uint8_t powerProgress = (power >= maxPower) ? 100 : (uint8_t)((100 *
power) / maxPower);
        tft.fillRect(0, 150, 126, 14, TFT_BLACK);
        tft.fillRect(0, 150, powerProgress, 14, TFT_RED);
        tft.drawRect(0, 150, 126, 14, TFT_WHITE);

    } else if (currentPage == 1) {
        // Сторінка 1: Загальний пробіг, середня швидкість, максимальна швидкість
        tft.fillScreen(TFT_BLACK);
        tft.setCursor(0, 0);
        tft.print("Total distance:");

        dtostrf(totalDistance, 6, 2, buf);  // Загальний пробіг у км
        tft.setCursor(0, 30);
        tft.print(buf);
        tft.print(" km");

        // Відображення середньої швидкості
        tft.setCursor(0, 60);
        tft.print("Avg speed:");
        double averageSpeed = (totalRuntime > 0) ? (totalDistance / totalRuntime)
: 0;
        dtostrf(averageSpeed, 4, 2, buf);
        tft.setCursor(0, 90);
```

```
        tft.print(buf);
        tft.print(" km/h");

        // Відображення максимальної швидкості
        tft.setCursor(0, 120);
        tft.print("Max speed:");
        dtostrf(maxSpeed_total, 4, 2, buf);
        tft.setCursor(0, 150);
        tft.print(buf);
        tft.print(" km/h");

    } else if (currentPage == 2) {
        // Сторінка 2: Кількість спалених калорій
        tft.fillScreen(TFT_BLACK);
        tft.setCursor(0, 0);
        tft.print("Calories burned:");

        dtostrf(caloriesBurned, 5, 2, buf);
        tft.setCursor(0, 30);
        tft.print(buf);
        tft.print(" kcal");
    }
}

void setup() {

    Serial.begin(115200);
    Serial.flush();
    delay(50);

    tft.init();
    tft.setRotation(1);
    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, LOW);

    showStatus("Starting BLE Scan...");
    BLEDevice::init("");
    scanner = BLEDevice::getScan();
    scanner->setAdvertisedDeviceCallbacks(new AdvertisedDeviceCallbacks());
    scanner->setInterval(1600);
    scanner->setWindow(1500);
    scanner->setActiveScan(true);

    Serial.begin(115200);
    if (!SPIFFS.begin(true)) {
        Serial.println("SPIFFS Mount Failed");
        return;
    }
```

```
    Serial.println("SPIFFS initialized.");
}

void loop() {
    // Перевірка стану підключення до кожного з датчиків
    if (cadenceConnected && !cadenceClient->isConnected()) {
        Serial.println("Cadence sensor disconnected. Attempting to
reconnect...");
        cadenceConnected = false;
        delete cadenceClient;  // Очистимо попереднє з'єднання
        cadenceClient = nullptr;
    }
    if (speedConnected && !speedClient->isConnected()) {
        Serial.println("Speed sensor disconnected. Attempting to reconnect...");
        speedConnected = false;
        delete speedClient;  // Очистимо попереднє з'єднання
        speedClient = nullptr;
    }

    // Перепідключення до датчиків, якщо підключення втрачено
    if (!cadenceConnected || !speedConnected) {
        //showStatus("Attempting to reconnect sensors...");
        scanner->start(5, false);
        scanner->stop();

        if (!cadenceConnected && cadenceDevice != nullptr) {
            cadenceConnected = connectToDevice(cadenceDevice, cadenceClient,
cadenceCharacteristic, serviceUUID, notifyUUID, notifyCadenceCallback);
            if (cadenceConnected) {
                Serial.println("Reconnected to Cadence Sensor");
                delete cadenceDevice;
                cadenceDevice = nullptr;
            }
        }

        if (!speedConnected && speedDevice != nullptr) {
            speedConnected = connectToDevice(speedDevice, speedClient,
speedCharacteristic, serviceUUID, notifyUUID, notifySpeedCallback);
            if (speedConnected) {
                Serial.println("Reconnected to Speed Sensor");
                delete speedDevice;
                speedDevice = nullptr;
            }
        }

        delay(5000); // Затримка перед повторною спробою, якщо датчики не
підключені
    } else {
```

```
        // Якщо датчики підключені, продовжуємо основні функції
        unsigned long now = millis();

        // Перевірка кнопки для перемикання сторінки
        if (digitalRead(BUTTON_PIN) == LOW) {  // Перевірка натискання кнопки
            delay(200);  // Дебаунс затримка
            currentPage = (currentPage + 1) % 3;  // Перемикання між сторінками 0
та 1,2
            tft.fillScreen(TFT_BLACK);  // Очищення екрану при перемиканні
сторінки
        }

        // Підрахунок часу активності, якщо велосипед у русі (швидкість або
каденс більше 0)
        if (speed > 0 || cadence > 0) {
            runtime += (now - last_millis);  // Додаємо час з попереднього циклу
            totalDistance += (speed * (now - last_millis)) / 3600000.0;  //
Відстань у км
            totalRuntime += (now - last_millis) / 3600000.0;  // Час у годинах

            if (speed > maxSpeed_total) {
                maxSpeed_total = speed;  // Оновлення максимальної швидкості,
якщо досягнута нова
            }

            caloriesBurned += (power * ((now - last_millis) / 3600000.0)) *
0.001;  // Спалені калорії
        }
        last_millis = now;  // Оновлюємо значення останнього часу

        // Розрахунок потужності в реальному часі
        int resistance = 10;  // Замініть на актуальне значення, якщо є
        power = calculatePower(cadence, resistance);  // Розрахунок потужності

        // Оновлення дисплея на основі поточної сторінки
        updateDisplay();
        delay(200);  // Затримка між оновленнями дисплея
    }
}
```

File «device.h»:

```
#ifndef DEVICE_H
#define DEVICE_H

#include <BLEDevice.h>
#include <TFT_eSPI.h>
```

```
// UUID для сервісу швидкості та каденсу
static BLEUUID serviceUUID("00001816-0000-1000-8000-00805f9b34fb");
static BLEUUID notifyUUID("00002a5b-0000-1000-8000-00805f9b34fb");

// Задані MAC-адреси для каденсу та швидкості
#define CADENCE_SENSOR_ADDRESS "d8:a6:1e:63:b4:03"
#define SPEED_SENSOR_ADDRESS "e1:45:3d:a6:f6:d0"

static BLEAdvertisedDevice* devices[20];
static int device_count = 0;

#define BUTTON_PIN 0
#define BUTTON_DEBOUNCE_TIME 50
#define BUTTON_SHORT_PRESS_TIME 400

// Додаємо пристрій до списку, лише якщо його MAC-адреса відповідає одному з потрібних датчиків
void addDevice(BLEAdvertisedDevice* device) {
  String address = device->getAddress().toString().c_str();

  // Перевіряємо, чи це один з потрібних пристроїв за MAC-адресою
  if (!address.equals(CADENCE_SENSOR_ADDRESS) && !address.equals(SPEED_SENSOR_ADDRESS)) {
    return;
  }
  // Уникнення додавання дублікатів
  for (uint8_t i = 0; i < device_count; i++) {
    if (address.equals(devices[i]->getAddress().toString().c_str())) {
      return;
    }
  }
  // Додаємо пристрій до масиву
  if (device->haveServiceUUID() && device->isAdvertisingService(serviceUUID)) {
    devices[device_count] = device;
    device_count++;
  }
}
// Вибір пристрою з доступних у масиві
BLEAdvertisedDevice* selectDevice(TFT_eSPI& tft) {
  tft.setTextSize(1);

  if (device_count == 0) return nullptr;
  if (device_count == 1) return devices[0];

  int selected = 0;

  while (true) {
    if (selected > device_count - 1) selected = 0;
```

```
    int start = selected < 2 ? 0 : selected - 1;
    tft.fillScreen(TFT_BLACK);
    for (int i = start; i < device_count; i++) {
      if (i == selected) {
        tft.setCursor(0, i * 10);
        tft.print("-> ");
      } else {
        tft.setCursor(0, i * 10);
        tft.print("   ");
      }
      tft.println(devices[i]->getName().c_str());
    }
    int lastState = LOW;
    int currentState;
    unsigned long pressedTime = 0;

    while (true) {
      currentState = digitalRead(BUTTON_PIN);
      if (lastState == HIGH && currentState == LOW) { // кнопка натиснута
        pressedTime = millis();
        Serial.println("DOWN");
      } else if (pressedTime != 0 && lastState == LOW && currentState == HIGH) {
// кнопка відпущена
        long pressDuration = millis() - pressedTime;
        Serial.println("UP");
        if (pressDuration < BUTTON_DEBOUNCE_TIME) {
          pressedTime = 0;
        } else if (pressDuration < BUTTON_SHORT_PRESS_TIME) {
          Serial.println("Коротке натискання");
          pressedTime = 0;
          selected++;
          break;
        } else {
          Serial.println("Довге натискання");
          pressedTime = 0;
          return devices[selected];
        }
      }
      lastState = currentState;
    }
  }
  return nullptr;
}
// Функція для отримання швидкості з датчика
float getSpeedFromSensor(BLEClient* client, BLEAdvertisedDevice* device); //
Тільки оголошення

#endif
```

ДОДАТОК Б

Публікації за темою роботи

Б.1 Стаття у журналі категорії Б

ISSN 2307-5732
DOI 10.31891/2307-5732

НАУКОВИЙ ЖУРНАЛ **2.2024**

ВІСНИК

Хмельницького національного університету

Технічні науки

Technical sciences

SCIENTIFIC JOURNAL
HERALD OF KHMELNYTSKYI NATIONAL UNIVERSITY

2024, Issue 2, Volume 333

Хмельницький

Технічні науки *ISSN 2307-5732*

DOI 10.31891/2307-5732-2024-333-2-52
УДК 004.77:796.6

ЖУРАВСЬКА ІРИНА
Чорноморський національний університет ім. Петра Могили
https://orcid.org/0000-0002-8102-9854
e-mail: iryna.zhuravska@chmnu.edu.ua
ПИЛИПЧУК БОГДАН
Чорноморський національний університет ім. Петра Могили
https://orcid.org/0009-0003-3893-5311
e-mail: pylypchyk50@ukr.net

ТЕХНОЛОГІЇ ІНТЕРНЕТУ РЕЧЕЙ ДЛЯ МОНІТОРИНГУ ФІЗИЧНИХ НАВАНТАЖЕНЬ ВЕЛОСПОРТСМЕНІВ

*В роботі розкрито принципи застосування технології Інтернет речей у сучасному спорті з акцентом на організацію режиму тренувань та змагань у велоспорті. Ця технологія є надійним способом збору даних, однак передбачає вибір певних шкал оцінювання фізичних навантажень спортсменів та втручання тренера в коригування системи тренувань, обраної за допомогою кіберфізичної системи. Найбільш розповсюдженими серед спортсменів є шкала FTP та шкала Вамса (VAMs). Найбільш придатними для поєднання особливостей застосування технології IoT та зазначених шкал є давачі, вбудовані в спортивні гаджети виробництва компанії Garmin – годинники, фітнес-браслети, GPS-пристрої тощо, які фіксують різноманітні дані під час активності.. Доступ до необхідної інформації, яка зберігається вебзастосунком Garmin Connect, можливо отримати шляхом експорту з сайту файлу з даними у форматі *.csv, які підпадають під категорію Big Data. Для розуміння спортивного прогресу розроблено програмне забезпечення (ПЗ), що включає клієнтський застосунок та прошарок роботи з даними, реалізовані мовою програмування Python з використанням відповідних бібліотек. ПЗ націлено на обробку та візуалізацію статистичних даних задля полегшення оцінювання фізичної підготовки та корегування тренувань велоспортсменів. У запропонованому рішенні враховано, що технології IoT мають передбачати підключення різноманітних спеціалізованих модулів та давачів, у т. ч. власної розробки, а не тільки розповсюджених серед велоспортсменів датчиків фірми Garmin. Тому в основі розробленого ПЗ лежать такі можливості, як масштабованість та гнучкість, які у подальшому забезпечать обмін даними між давачами спорядження велоспортсменів та комп'ютерною системою в автоматичному режимі з використанням стандартних протоколів зв'язку відповідно до концепції IoT.*

Ключові слова: Інтернет речей, кіберфізична система, датчики, велоспорт, моніторинг фізичних навантажень, інформаційна система, Big Data.

ZHURAVSKA IRYNA, PYLYPCHUK BOHDAN
Petro Mohyla Black Sea National University

THE INTERNET OF THINGS TECHNOLOGIES FOR MONITORING THE PHYSICAL LOADS OF CYCLISTS

*The work reveals the principles of using the Internet of Things technology (IoT) in modern sports with an emphasis on the organization of training regimes and competitions in cycling sport. This technology is a reliable way of collecting data, but it involves the selection of certain scales for evaluating the physical loads of athletes and the correction of the coach in adjusting the training regimes selected with the help of the cyber-physical system. The most common among athletes are the FTP scale and the VAMs scale. The most suitable for combining the features of the application of IoT technology and the selected scales are sensors built into sports gadgets manufactured by Garmin – watches, fitness bracelets, GPS devices, etc., which record various data during cyclist activity. This data have stored by the Garmin Connect web application and can been classificated as Big Data. Access to the necessary information can be obtained by exporting a data file in *.csv format from the site. The architecture of the software includes a client application and a layer of working with data, implemented in the Python programming language using appropriate libraries. The software is aimed at processing and visualizing statistical data to facilitate the assessment of physical fitness and training adjustments of cyclists. The proposed solution takes into account that IoT technologies should provide for the connection of various specialized modules and sensors, including those of our own development, and not only Garmin sensors populared among cyclists. Therefore, the basis of the developed software are such possibilities as scalability and flexibility, which will in the future ensure the exchange of data between the embedded sensors into cyclists' equipment and the computer system in automatic mode using standard communication protocols in accordance with the concept of IoT.*

Keywords: Internet of Things, cyber physical system, sensors, cycling sport, physical loads monitoring, information system, Big Data.

Постановка проблеми

Впровадження технологій Internet of Things (IoT) в процес оцінювання фізичного навантаження велоспортсменів – один із пріоритетів сучасного розвитку цієї галузі. Інтернет речей в спорті спроможний забезпечити оптимізацію процесу на всіх етапах: від моменту передачі даних від спортсмена під час тренування до фіксації показників його стану під час змагань. За допомогою технології Інтернету речей показники фізичного стану спортсмена збирають через мережу підключених до інтернету пристроїв. Така мережа побудована на основі кіберфізичного підходу: пристрої, які мають вбудовані давачі, збирають інформацію і обмінюються нею між собою і хмарним сховищем, а також надають спортсмену команди щодо зміни режиму тренування або поведінки на трасі. Всі ці пристрої взаємопов'язані між собою, мають можливість зчитування та відображення параметрів, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини за рахунок використання інтелектуальних інтерфейсів. Але для розуміння спортивного прогресу необхідно розробити спеціалізоване програмне забезпечення (ПЗ), що обробляє велику кількість даних, які підпадають під категорію Big Data. До того ж, отримані дані

потребують якісної візуалізації для можливості швидкого коригування з боку тренера режиму тренувань / стратегії виступів на змаганні.

Об'єктом роботи (розробки) є процеси моніторингу фізичних навантажень велоспортсмена з використанням технологій IoT.

Предмет роботи – методи та засоби створення кіберфізичної системи, що виконує функції обробки даних з давачів, вбудованих у спорядження велоспортсмена, щодо фізичного навантаження та коригування плану тренувань велоспортсменів.

Мета – оцінювання фізичних навантажень велоспортсмена шляхом обробки статистичних даних, отриманих з сенсорів велоспорядження за допомогою технологій Інтернету речей, та подальша візуалізація даних засобами розробленого програмного забезпечення (ПЗ) для можливості коригування режимів тренувань велоспортсмена.

Аналіз останніх джерел

Спортивні технології відразу підхопили тенденцію до цифровізації та використання комунікаційних технологій. Особливість застосування IoT у спорті – технологія використовується на всіх рівнях підготовки спортсменів: від отримання його фізичних показників до планування його досягнень [1]. Ось приклади, як впроваджують Інтернет речей у велоспорт [2]:

давачі, встановлені з безпосереднім контактом з тілом спортсмена та на велосипеді, відправляють дані обліковій системі, що виключає людський фактор;

стан спортсмена та велосипеда перебуває під постійним контролем;

система датчиків дає змогу відстежувати й контролювати навантаження на спортсмена.

інформаційна система віддалено відстежує фізичні показники спортсмена та реєструє їх;

інформаційна система розраховує відповідні показники згідно з обраною шкалою оцінювання та на їх основі планує режим тренувань;

тренер віддалено керує обладнанням системи й оперативно коригує проблеми й усуває помилки у призначеному режимі тренувань.

Фізичні навантаження можуть бути різної інтенсивності та тривалості, тому контроль за ними є дуже важливим завданням для тренерів та спортсменів [3]. Для вирішення таких задач доцільно використання технологій Інтернету речей, які дозволяють реєструвати та передавати стандартними протоколами зв'язку параметри моніторингу навантажень велоспортсмена з давачів, які вбудовані в найбільш поширені для велоспорядження пристрої від компанії Garmin [4]. Вірний підхід до тренувань дозволяє досягнути максимальних результатів та зменшити ризик отримання травм.

Враховуючі те, що велоспорт – це загальний термін, який використовується для багатьох окремих змагальних дисциплін (велоспорт на треку, гірський велосипед, шосейний велоспорт, гонку на час, велокрос, велоспорт на гравії, вільний стиль BMX, перегони BMX, паравелоспорт, артистичний велоспорт тощо), для оцінювання фізичної підготовки велоспортсмена використовуються різні шкали оцінювання та параметри моніторингу, що спричиняє необхідність обробки великої кількості даних, які підпадають під категорію Big Data. Основними шкалами є [5; 6]:

- шкала FTP;
- шкала Вамса (VAMs);
- шкала Фостера (FTS);
- шкала Ратінга спроможності (RPE);
- шкала Борхардта.

Найбільш поширеними серед професійних велосипедистів є шкала FTP та Шкала Вамса (VAMs – скорочення від «Velocita Ascensionale Media», що означає середню підйомну швидкість італійською) [5]. Тому для обробки параметрів, використовуваних зазначеними шкалами, окремих досліджень потребують кінцеві функції для роботи зі шкалами. В таких функціях мають бути реалізовані всі необхідні методи для роботи з обраною шкалою, а саме функція оцінки фізичної підготовки спортсмена, візуалізація даних та їх експорт – все це передбачає програмну реалізацію методів опрацювання великих даних (Big Data). Big Data, зібрані від Garmin-давачів, – це дуже великі та складні набори даних, які не можуть бути оброблені або проаналізовані за допомогою традиційних технік обробки даних [7].

Три особливості великих даних, які мають бути враховані при їх обробці спеціалізованим ПЗ, це обсяг даних, швидкість їх реєстрації та їх різноманітність.

Виклад основного матеріалу

Проєкт розробки програмного забезпечення (ПЗ) націлений на створення інформаційної системи для обробки та візуалізації статистичних даних, спрямованої на полегшення оцінювання фізичної підготовки та корегування тренувань велоспортсменів. Застосовані бібліотеки Pandas та NumPy для ефективної обробки даних і вирішення недоліків аналогів, таких як відсутність візуалізації та глибокого аналізу фізичної підготовки.

Архітектура інформаційної системи включає клієнтський застосунок та прошарок роботи з даними, реалізовані мовою програмування Python з використанням відповідних бібліотек. Інтерфейс користувача відповідає стандартам UI та UX, сприяючи ефективній роботі тренерів та покращенню фізичної підготовки велоспортсменів через аналіз та візуалізацію статистичних даних.

На рис. 1 показано архітектуру програми для роботи з Big Data, яка адаптована до вимог програмного забезпечення з аналізу даних для оцінки фізичних показників [8].

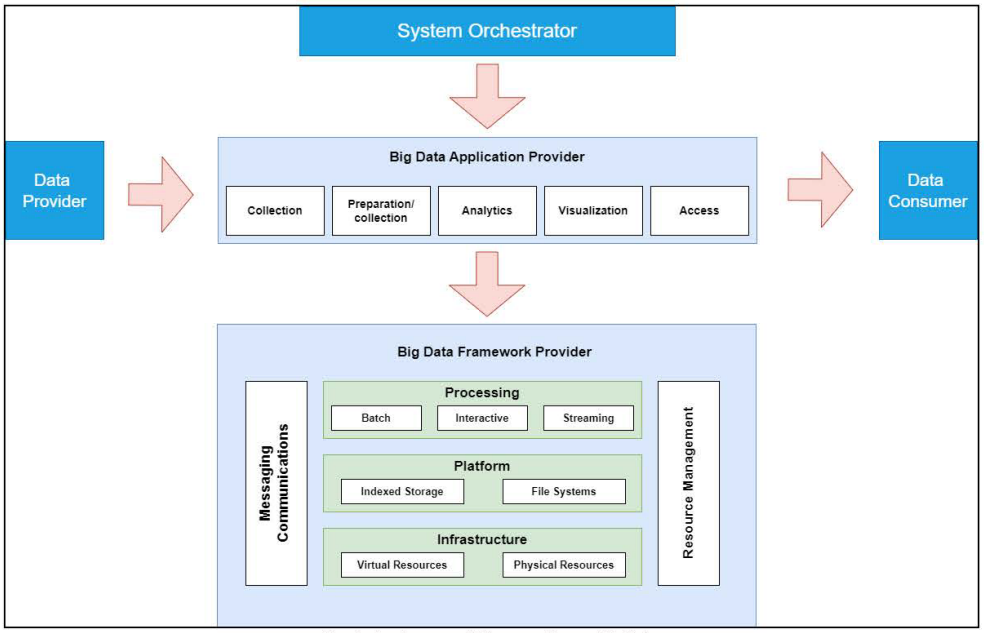


Рис.1. Архітектура ПЗ для роботи з Big Data

Опис головних компонентів схеми, наведеної на рис. 1:

System Orchestrator гарантує, що різні програми, дані та компоненти інфраструктури працюють разом;

Data Provider надає нові дані, що надходять з різних джерел, у систему великих даних для перетворення їх у оброблювані Data Set-и [9];

Big Data Application Provider містить бізнес-логіку та функціональні можливості, необхідні для перетворення даних у цінні знання за допомогою п’яти основних дій:

- збір;
- підготовка;
- аналітика;
- візуалізація;
- доступ.

Big Data Framework Provider має ресурси та сервіси для зберігання та обробки даних;

Data Consumer використовує інтерфейси сервісів, наданих Big Data Application Provider для доступу до необхідної інформації.

Garmin – відомий виробник спортивних гаджетів, таких як годинники, фітнес-браслети та GPS-пристрої, які фіксують різноманітні дані під час активності. Ці дані, такі як серцевий ритм, відстань, швидкість та багато інших параметрів, збираються і зберігаються вебзастосунком Garmin Connect. Завдяки обладнанню Garmin та вебзастосунку Garmin Connect можливо отримати доступ до необхідної інформації шляхом експорту файлу з даними з сайту, який після обробки та аналізу розробленим ПЗ допоможе в розумінні свого спортивного прогресу та прийнятті більш обгрунтованих рішень щодо покращення своїх тренувань та здоров'я.

Також у майбутньому планується відійти від залежності у використанні модулів від побічних виробників обладнання та розробити власні спеціалізовані модулі та датчики. Тому у основі розроблюваного ПЗ лежать такі можливості, як масштабованість та гнучкість, які у подальшому допоможуть розвивати та оновлювати функціонал та можливості застосунку й проєкту в цілому.

Отримані дані є файлами з розширенням *.csv* – тут зібрані дані велосипедиста протягом тренувального сезону (12 місяців). В даному файлі записані показники: Activity Type, Date, Favorite, Title, Distance, Calories, Time, Avg HR, Max HR, Aerobic TE, Avg Speed, Max Speed, Total Ascent, Total Descent, Avg Stride Length, Avg Vertical Ratio, Avg Vertical Oscillation, Avg Ground Contact Time, Avg Bike Cadence, Max Bike Cadence, Normalized PowerB (NPB), Training Stress ScoreB, Max Avg Power (20 min), Avg Power, Max Power, Grit, Flow, Total Strokes, Avg Swolf, Avg Stroke Rate, Total Reps, Dive Time, Min Temp, Surface Interval, Decompression, Best Lap Time, Number of Laps,Max Temp, Avg Resp, Min Resp, Max Resp, Moving Time, Elapsed Time, Min Elevation, Max Elevation.

Pandas – це бібліотека для аналізу, обробки і маніпуляції з даними [10].

Matplotlib – це потужна бібліотека для створення графіків у Python, яка дозволяє створювати статичні, анімовані та інтерактивні візуалізації [11]. Вона легко інтегрується з NumPy та різними графічними бібліотеками, такими як Tkinter або PyQt [12–14]. За допомогою цієї бібліотеки можна побудувати різноманітні типи графіків.

NumPy – це бібліотека, яка розширює стандартні структури даних шляхом додавання багатовимірних масивів та матриць, а також надає велику колекцію математичних функцій для роботи з цими масивами [12].

При розробці застосунків на мові програмування Python, правильна архітектура відіграє важливу роль у створенні ефективного і легкозмінного програмного рішення.

Основна ідея полягає в тому, щоб розділити функціональність програми на окремі модулі, які виконують специфічні завдання (рис. 2). Це дозволяє покращити читабельність, підтримку і розширюваність проекту, враховуючи всі його потоки даних (рис. 3).

```
def plot_distance_by_month():...

def FTP():...

def export_to_excel(data):...


def VAMs():...

def export_to_excel(data):
    save_path = filedialog.asksaveasfilename(defaultextension='.xlsx')
    if save_path:
        with ExcelWriter(save_path) as writer:
            data.to_excel(writer, index=False)
        messagebox.showinfo("Export Successful", "Data exported to Excel successfully.")
```

Рис. 2. Розбиття коду на модулі (функції)

Саме таких принципів та шаблону структури проекту було дотримано під час розробки застосунку з моніторингу фізичних навантажень велоспортсмена.

Основною задачею даного прошарку є різноманітна робота та оперування даними, аби не ускладнювати розробку інтерфейсу зайвою логікою. Вирішення досягається шляхом створення функцій – методів для кожного методу оцінки фізичної підготовки, які б містили усі необхідні методи для читання, запису, перетворення даних тощо. Загальна логіка даного шару прописана в базовому класі і перевизначається у дочірніх класах. При цьому можна легко створити різні допоміжні функції всередині похідних методів, які б були локально корисними, без залежності від зовнішніх факторів.

Першою і основною функцією є читання даних. Якщо сталася помилка при завантаженні файлу (наприклад, файл не має розширення *.csv* або має неправильний формат), з'являється діалогове вікно повідомлення про помилку *(messagebox.showerror)* з повідомленням «Помилка» та текстом «Не вдалося завантажити файл. Перевірте, чи файл має розширення *.csv* та спробуйте ще раз».

Наступною важливою функцією є *def format_file*. Дана функція призначена для форматування файлу з даними. Вона приймає один аргумент – об'єкт *DataFrame* під назвою *df*. Видаляється певний набір стовпців з датафрейму *df*, які не потрібні для реалізації методів оцінки фізичної підготовки. Стовбці, що видаляються, мають такі назви: *Activity Type, Favorite, Aerobic TE, Avg Stride Length, Avg Vertical Ratio, Avg Vertical Oscillation, Avg Ground Contact Time, Grit, Flow, Total Strokes, Avg. Swolf, Avg Stroke Rate, Total Reps, Dive Time, Min Temp, Surface Interval, Decompression, Best Lap Time, Number of Laps, Max Temp, Avg Resp, Min Resp, Max Resp, Moving Time, Elapsed Time, Min Elevation.* Ці стовбці перераховані у вигляді списку рядків і виключаються з датафрейму. В результаті цих операцій датафрейм *df* змінюється, що пришвидшує роботу застосунку.

Ключовими ж являються кінцеві функції для роботи зі шкалами. Нижче наведено код функції для розрахунку фізичної підготовки за **шкалою FTP**. В ній реалізовані всі необхідні методи для роботи з даною шкалою, а саме функція оцінки фізичної підготовки спортсмена, візуалізація даних та їх експорт. Дана функція FTP() виконує розрахунок FTP (функціональний поріг) на основі даних з файлу, вибраного користувачем (рис. 4), та відображає результати розрахунку у вікні програми.

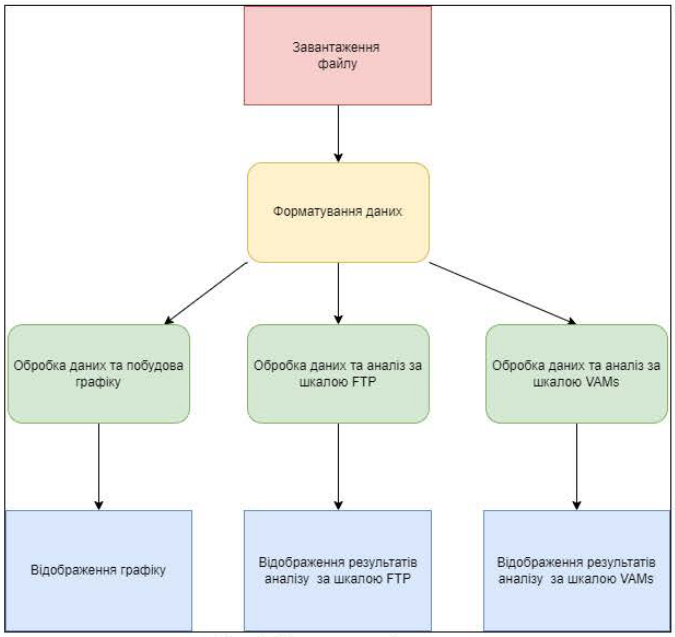


Рис. 3. Діаграма потоків даних

```
def FTP():
    """
    Фунуція обробки даних для аналізй за шкалою FTP
    """
    data = pd.read_csv(file_path)
    data = format_file(data)
    data['Avg Power'] = data['Avg Power'].str.replace(',', '').astype(float)  # Перетворення на float
    data['FTP'] = data['Avg Power'] / 60 * 0.95
    # Додавання нового стовбця для рейтингу FTP на основі значення FTP
    conditions = [
        (data['FTP'] < 2.23),
        (data['FTP'] >= 2.23) & (data['FTP'] <= 2.78),
        (data['FTP'] >= 2.79) & (data['FTP'] <= 3.92),
        (data['FTP'] >= 3.93) & (data['FTP'] <= 5.04),
        (data['FTP'] > 5.04)
    ]
    ratings = ['Непідготовлений', 'Задовільно', 'Добре', 'Відмінно', 'Чудово']
    data['Rating'] = np.select(conditions, ratings, default='Unknown')

    # Створення нового вікна верхнього рівня для результатів розрахунку FTP
    ftp_window = tk.Toplevel(root)
    ftp_window.title("FTP Calculation Results")

    # Створення рамки у вікні верхнього рівня за допомогою менеджера геометрії сітки
    table_frame = tk.Frame(ftp_window)
    table_frame.grid(padx=10, pady=10)

    # Створення віджету таблиці та відображення результату обчислення FTP
    table = Table(table_frame, dataframe=data[['Date', 'FTP', 'Rating']], showtoolbar=True, showstatusbar=True)
    table.grid(row=0, column=0, padx=10, pady=10)
    table.show()

    # Створення кнопки експорту
    export_button = tk.Button(ftp_window, text="Export to Excel", command=lambda: export_to_excel(data))
    export_button.grid(row=1, column=0, pady=10)

def export_to_excel(data):
    save_path = filedialog.asksaveasfilename(defaultextension='.xlsx')
    if save_path:
        with ExcelWriter(save_path) as writer:
            data.to_excel(writer, index=False)
        messagebox.showinfo("Export Successful", "Data exported to Excel successfully.")
```

Рис. 4. Лістинг функції розрахунку FTP (функціонального порогу)

Основні дії функції:

1) зчитується файл з даними за допомогою функції *pd.read_csv()* і зберігається у змінну *data;*

2) викликається функція *format_file(data),* яка форматує дані, виконуючи певні операції зі стовбцями датафрейму *data*. Оновлений датафрейм знову зберігається у змінну *data*;

3) виконується обробка стовбця даних. Кома в значеннях замінюється на порожній рядок за допомогою методу *str.replace(),* а потім значення перетворюються на числовий тип даних *(float)* за допомогою методу *astype()* та виконується збереження значень;

4) розраховується значення FTP;

5) додається новий стовбець «Rating» для оцінки FTP на основі значення FTP. Використовується функція *np.select()* для вибору відповідного рейтингу залежно від діапазону FTP. Рейтинги та умови розрахунку відповідають діапазонам FTP, і значення «Unknown» використовується як значення за замовчуванням, якщо жодна з умов не співпадає;

6) створюється нове вікно верхнього рівня *(tk.Toplevel)* для відображення результатів розрахунку FTP;

7) створюється фрейм всередині вікна за допомогою менеджера геометрії *'grid' (table_frame.grid(padx=10, pady=10));*

8) створюється віджет таблиці *(Table)* з бібліотеки *tkinter* у фреймі *table_frame* і відображаються результати розрахунку FTP з обраними стовпцями «Date», «FTP» та «Rating». Таблиця показує панель інструментів та рядок стану. Розташовується за допомогою *table.grid().* Відображається таблиця за допомогою *table.show();*

9) створюється кнопка експорту до Excel (Export to Excel), яка викликає функцію *export_to_excel(data)* при натисканні. Кнопка розташовується у вікні FTP за допомогою *export_button.grid().*

Функція для роботи зі **шкалою VAMs** є майже аналогічна, але з певними незначними особливостями. Дана функція VAMs() виконує обробку даних для аналізу за шкалою VAMs (Vertical Ascent Meters per Second) на основі даних з файлу, вибраного користувачем. Результати обчислень відображаються у вікні програми. Основні дії функції:

1) зчитується файл з даними за допомогою функції pd.read_csv() і зберігається у змінну data;

2) викликається функція *format_file(data)*, яка форматує дані, виконуючи певні операції зі стовпцями датафрейму *data*. Оновлений датафрейм знову зберігається у змінну *data*;

3) виконується обробка стовпця «*Max Elevation*». Кома в значеннях замінюється на порожній рядок за допомогою методу *str.replace()*, а потім значення перетворюються на числовий тип даних (*float*) за допомогою методу *astype()*. Результати зберігаються назад у стовпець *«Max Elevation»*;

4) стовпець *«Time»* перетворюється на тип *datetime*, використовуючи *pd.to_datetime()*, з форматом часу *«%H:%M:%S».* Потім значення округлюються до найближчої години за допомогою методу *round()* та *dt.time*. Оновлені значення зберігаються назад у стовпець *«Time»;*

5) створюється стовпець *«Seconds»*, де обчислюється тривалість часу у секундах на основі значення стовпця *«Time»*. Для цього використовується метод *apply()* та *datetime.timedelta().* Результати зберігаються у стовпець *«Seconds»;*

6) обчислюється значення VAMs (метри вертикального підйому на секунду) шляхом ділення значення *«Max Elevation»* на значення *«Seconds».* Результати зберігаються у новому стовпці *«VAMs»;*

7) створюється нове вікно верхнього рівня *(tk.Toplevel)* для відображення результатів розрахунку VAMs;

8) створюється фрейм *table_frame* у вікні верхнього рівня, використовуючи менеджер геометрії *grid();*

9) створюється віджет таблиці *(Table)* та відображаються результати обчислення VAMs за допомогою передачі відповідних стовпців датафрейму *data* у параметр *dataframe* конструктора *Table*. Віджет таблиці розміщується у фреймі *table_frame* за допомогою *table.grid().* Панель інструментів та рядок стану також відображаються, оскільки параметри *showtoolbar=True* та *showstatusbar=True*;

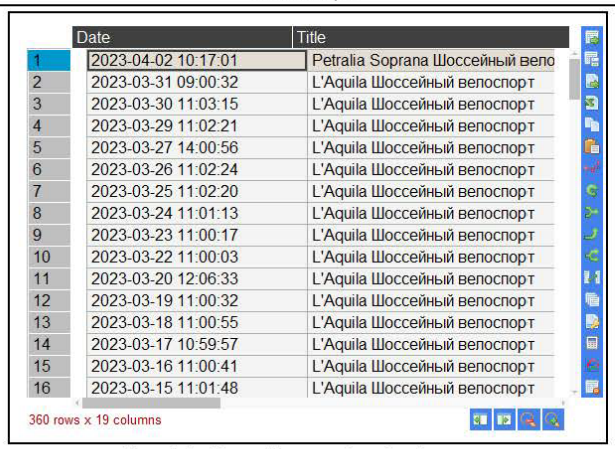
10) викликається метод *table.show()*, щоб відобразити таблицю з результатами;

11) створюється кнопка експорту до Excel з текстом «Export to Excel», яка викликає функцію *export_to_excel(data)* при натисканні. Кнопка розташовується у вікні *vams_window* за допомогою *export_button.grid().* Функція *export_to_excel(data)* виконує експорт даних у форматі Excel *(*.xlsx).* Функція відкриває діалогове вікно для вибору шляху збереження файлу за допомогою *filedialog.asksaveasfilename().* Якщо шлях вибраний, то дані зберігаються у файл Excel за допомогою *data.to_excel().* Після успішного експорту виводиться повідомлення про успішне експортування за допомогою *messagebox.showinfo().*

Також було реалізовано метод вибору шкали оцінювання як варіанту списку з *combobox* (рис. 5).

```
def ok_button():
    if variable.get() == "Графік пройденої відстані по місяцях":
        plot_distance_by_month()
    elif variable.get() == "Шкала FTP":
        FTP()
    elif variable.get() == "Шкала VAMs":
        VAMs()
    else:
        root.destroy()
ok_button = tk.Button(frame, text="Розрахувати", command=ok_button)
ok_button.grid(row=0, column=2, padx=15, pady=15)
```

Рис. 5. Лістинг коду вибору варіанту списку з combobox



	Date	Title
1	2023-04-02 10:17:01	Petralia Soprana Шоссейный вело
2	2023-03-31 09:00:32	L'Aquila Шоссейный велоспорт
3	2023-03-30 11:03:15	L'Aquila Шоссейный велоспорт
4	2023-03-29 11:02:21	L'Aquila Шоссейный велоспорт
5	2023-03-27 14:00:56	L'Aquila Шоссейный велоспорт
6	2023-03-26 11:02:24	L'Aquila Шоссейный велоспорт
7	2023-03-25 11:02:20	L'Aquila Шоссейный велоспорт
8	2023-03-24 11:01:13	L'Aquila Шоссейный велоспорт
9	2023-03-23 11:00:17	L'Aquila Шоссейный велоспорт
10	2023-03-22 11:00:03	L'Aquila Шоссейный велоспорт
11	2023-03-20 12:06:33	L'Aquila Шоссейный велоспорт
12	2023-03-19 11:00:32	L'Aquila Шоссейный велоспорт
13	2023-03-18 11:00:55	L'Aquila Шоссейный велоспорт
14	2023-03-17 10:59:57	L'Aquila Шоссейный велоспорт
15	2023-03-16 11:00:41	L'Aquila Шоссейный велоспорт
16	2023-03-15 11:01:48	L'Aquila Шоссейный велоспорт

360 rows x 19 columns

Рис. 6. Фрейм з табличною візуалізацією даних

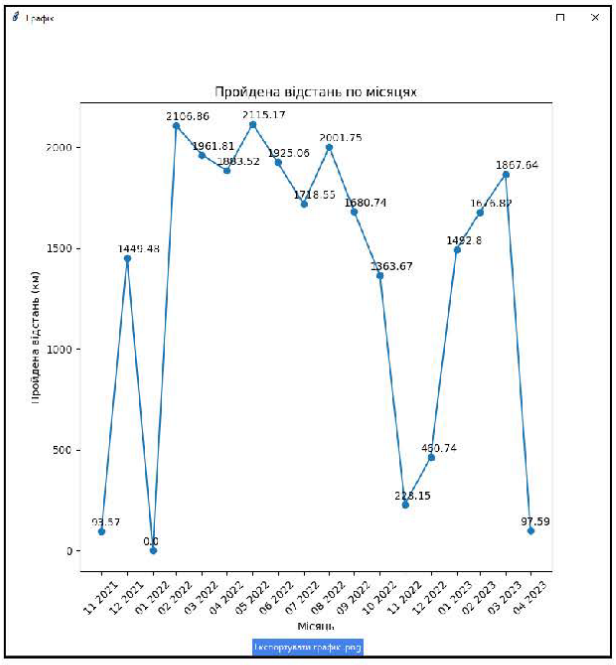


Рис. 7. Фрейм з візуалізацією даних

Дана функція необхідна для того, щоб можна було за значенням із combobox, що використовується для обрання методу оцінки, виконати обраний метод.

Наступним фреймом є таблична візуалізація (за можливістю) обраних даних, зображена на рис. 6. Фрейм з візуалізацією табличних даних, обраних користувачем, виконується одразу після вибору файлу з даними для аналізу.

Наступним фреймом є візуалізація даних з попереднього фрейму, зображена на рис. 7. Під зображенням реалізовано меню для роботи з ним, головною функцією в якому є саме можливість експорту зображення.

Візуалізація відбувається завдяки класам *Figure* та *FigureCanvasTkAgg* бібліотеки *matplotlib*. Перший відповідає за власне фігуру та її зображення, а другий дозволяє отримати віджет для подальшого відображення в фреймі. Приклад суміщення роботи цих класів для отримання результату, зображеного на рис. 11, наведено на лістингу нижче. Також на ньому вказано, як необхідно додавати зображення до фрейму. Відбувається це через отримання віджету завдяки функції *get_tk_widget()*, наведеної на рис. 8.

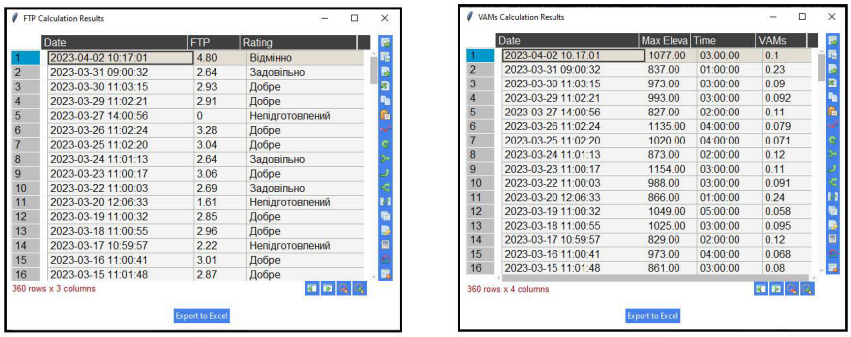
```
# Створюємо нове вікно (frame) для відображення графіка
    graph_window = tk.Toplevel(root)
    graph_window.title('Графік')

    # Створюємо об'єкт FigureCanvasTkAgg для відображення графіка в новому вікні
    canvas = FigureCanvasTkAgg(fig, master=graph_window)
    canvas.draw()
    canvas.get_tk_widget().pack()
    # Додавання кнопки експорту графіку
    def export_graph():
        save_path = filedialog.asksaveasfilename(defaultextension='.png')
        if save_path:
            fig.savefig(save_path)
            print("Графік експортовано у форматі .PNG")

    export_button = tk.Button(master=graph_window, text="Експортувати графік .png", command=export_graph)
    export_button.pack()
```

Рис. 8. Лістинг коду функції *get_tk_widget()*

Головними фреймами є фрейми з результатами оцінювання спортсмена за різними шкалами, як показано на рис. 9.



а) **б)**

Рис. 9. Фрейм з результатами оцінювання: а – за шкалою FTP; б – за шкалою VAMs

Можна було також зробити глобальний словник, але тут є певна проблема, що не дозволить реалізувати все через те, що елементи на початку інтерпретованого файлу нічого не знають про елементи після них. Це можна виправити імпортом анотацій з модулю __future__, але це спрацює лише всередині функцій. Поза ними дані маніпуляції будуть неуспішними і IDE буде видавати помилку, оскільки посилання на функцію чи клас є невизначеним.

Висновки

У роботі розглянуто проблеми та можливості застосування технології Інтернету речей (IoT) для збору даних з давачів, вбудованих у спорядження спортсмена з акцентом на проблеми сучасних видів велоспорту. передбачає вибір певних шкал оцінювання фізичних навантажень спортсменів Розроблено програмне забезпечення (ПЗ), що обробляє велику кількість даних, які підпадають під категорію Big Data. Архітектура ПЗ включає клієнтський застосунок та прошарок роботи з даними, реалізовані мовою програмування Python з використанням відповідних бібліотек. Проаналізовано особливості опрацювання отриманих даних із застосуванням шкал FTP та VAMs оцінювання фізичних навантажень спортсменів.

Розроблене ПЗ націлено на обробку та візуалізацію статистичних даних задля полегшення оцінювання фізичної підготовки та корегування тренувань велоспортсменів. У запропонованому рішенні враховано, що технології IoT мають передбачати підключення різноманітних спеціалізованих модулів та давачів, у т. ч. власної розробки, а не тільки розповсюджених серед велоспортсменів датчиків фірми Garmin. Тому в основі розробленого ПЗ лежать такі можливості, як масштабованість та гнучкість, які у подальшому забезпечать обмін даними між давачами велоспортсменів та комп'ютерною системою в автоматичному режимі з використанням стандартних протоколів зв'язку відповідно до концепції IoT.

Розробка має практичне значення для тренерів, дозволяючи швидше оцінювати фізичну підготовку спортсменів та поліпшувати коригування відновлювальних процесів.

References

1. Du M., Liu Z. Efficient feature recognition and matching technology for IoT-enabled sports training. *Internet Technology Letters*. Nov. 2023. DOI: 10.1002/itl2.490.

2. Thomas Fallon T., Heron N. A systematic review protocol of injuries and illness across all the competitive

cycling disciplines : preprint. Jan. 2024. 9 p. DOI: 10.21203/rs.3.rs-3909153/v2.

3. Municio E., Daneels G., De Brouwer M., Ongenae F., De Turck F. Continuous athlete monitoring in challenging cycling environments using IoT technologies. *IEEE Internet of Things Journal.* Sep. 2019. Vol. 99. P. 1–14. DOI: 10.1109/JIOT.2019.2942761.

4. Classification of Garmin heart rate monitors: analysis, tips, recommendations. URL: https://prostobzor.com/garmin-hrms-rating/

5. Global Recommendations on Physical Activity for Health, World Health Organization. Geneva, Switzerland, 2009. URL: http://www.who.int/ncds/prevention/physical-activity/en/

6. FTP (Functional Threshold Power). URL: https://velojournal.net/kak-izmerit-progress-v-velosporte-ftp-test-i-drugie-metody

7. Nunes F. P., Domingues P., Frade M. Post-mortem digital forensic analysis of the Garmin Connect application for Android. *Forensic Science International Digital Investigation*. Sep. 2023. Vol. 47. DOI: 10.1016/j.fsidi.2023.301624.

8. Big data architectures. URL: https://learn.microsoft.com/en-us/azure/architecture/data-guide/big-data/.

9. Data set. URL: https://www.techtarget.com/whatis/definition/data-set

10. Getting started – Pandas documentation. URL: https://www.w3schools.com/python/pandas/default.asp

11. Matplotlib: Visualization with Python. URL: https://matplotlib.org/

12. NumPy Introduction. URL: https://www.w3schools.com/python/numpy/numpy_intro.asp .

13. Tkinter – Interface Python до Tcl/Tk. URL: https://docs.python.org/uk/3/library/tkinter.html.

14. PyQt Documentation. URL: https://doc.qt.io/qtforpython-6/

Б.2 Свідоцтво про реєстрацію авторського права на твір

УКРАЇНА

СВІДОЦТВО

про реєстрацію авторського права на твір

№ 123174

Комп'ютерна програма «Cyclist Monitoring System»

(вид, назва твору)

Автор (співавтори) **Пилипчук Богдан Віталійович, Журавська Ірина Миколаїна**

(прізвище, ім’я, по батькові (за наявності), псевдонім (за наявності))

Авторські майнові права належать спільно **Пилипчук Богдан Віталійович, вул. Космонавтів, 67, кв. 2, м. Миколаїв, 54028; Журавська Ірина Миколаїна, вул. Чкалова, 107, м. Миколаїв, 54055**

(прізвище, ім’я, по батькові (за наявності) фізичної особи / найменування юридичної особи, адреса)

Дата реєстрації 24 січня 2024 р.

Виконувач обов'язків Директора Державної організації «Український національний офіс інтелектуальної власності та інновацій»

Богдан ПАДУЧАК

УКРНОІВІ