Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Пояснювальна записка

до бакалаврського дипломного проекту на тему:

Мікропроцесорна система велокомп'ютера на Raspberry Pi

Виконав: студент 4 курсу, групи 2СП-20б
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма – «Комп'ютерна інженерія»
Бохенек Є.С.

Керівник к.т.н., доц. каф. ОТ
Муращенко О.Г.

Рецензент д.т.н, проф.
Ліщинська Л.Б.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

"20" 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо—кваліфікаційний рівень бакалвр

Галузь знань — 12 Інформаційні технології

Спеціальність — 123 Комп'ютерна інженерія

Освітня програма — Системне програмування

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ проф., д.т.н.

_____ Азаров О. Д.

"06" 02 2024 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКИЙ ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Бохенек Євгену Станіславовичу

1 Тема проекту: «Мікропроцессорна система велокомп'ютера на Raspberry Pi», керівник роботи Муращенко О.Г., к.т.н., доцент кафедри ОТ, затверджено наказом вищого навчального закладу від "11" березня 2024 року № 80

2 Строк подання студентом проекту 10.06.2024

3 Вихідні дані до проекту: технічний опис системи, технології розробки, технічна документація, теоретичні дані.

4 Зміст розрахунково-пояснювальної записки: вступ, огляд існуючих рішень та аналіз, проектування мікропроцесорної системи, огляд системи та його тестування, висновки, список використаних джерел.

5 Консультанти розділів проекту приведені в таблиці 1.

6 Дата видачі завдання 9.03.2024 р.

7 Календарний план приведений в таблиці 2.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	к.т.н., доц. каф. ОТ Муращенко О.Г		

Таблиця 2— Календарний план

№	Назва та зміст етапу	Термін виконання		Приміт
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БДП	09.03.2024	11.03.2024	Виконано
2	Аналіз літературних джерел. Попередня розробка основних розділів	12.03.2024	20.03.2024	Виконано
3	Розробка технічного завдання (ТЗ)	21.03.2024	07.04.2024	Виконано
4	Огляд існуючих рішень та аналіз сервісу	08.04.2024	28.04.2024	Виконано
5	Проектування мікропроцессорної системи велокомп'ютера на базі Raspberry Pi	29.04.2024	05.05.2024	Виконано
6	Огляд системи	06.05.2024	12.05.2024	Виконано
7	Оформлення пояснювальної записки та графічної частини	13.05.2024	19.05.2024	Виконано
8	Нормоконтроль	20.05.2024	09.06.2024	Виконано
9	Попередній захист БДП, доопрацювання, рецензування БДП	20.05.2024	09.06.2024	Виконано
10	Захист БДП ЕК	10.06.2024	16.06.2024	Виконано

Студент Євгеній БОХЕНЕК

Керівник роботи к.т.н., доц. Олександр МУРАЩЕНКО

АНОТАЦІЯ

УДК 621.3

Бохенек Є.С. Мікропроцесорна система велокомп’ютера на Raspberry Pi / Arduino. Бакалаврська кваліфікаційна робота зі спеціальності 123 «Комп’ютерна інженерія», освітня програма «Системне програмування». Вінниця: ВНТУ, 2024. 80 с.

На укр. мові. Бібліогр.: 21 рис.; 6 лістинга;

У даному дипломному проекті розглянуто розробку та реалізацію мікропроцесорної системи велокомп'ютера на базі платформ Raspberry Pi та Arduino. Основною метою дослідження є створення багатофункціонального пристрою, який забезпечує велоентузіастів можливістю моніторингу різних параметрів поїздки, таких як швидкість, дистанція, час, а також простір для подальшого покращення та додавання новий систем та функцій. Ключові слова: мікропроцесор , Arduino, Raspberry Pi, велокомп’ютер , SPI.

ABSTRACT

Bokhenek E.S. Microprocessor-Based Bicycle Computer System on Raspberry Pi / Arduino. Bachelor's thesis on specialty 123 "Computer engineering", educational program "System programming". Vinnytsia: VNTU, 2024. 80 p.

In Ukrainian language. Bibliographer: 21 figures; 6 listings;

In the bachelor's thesis, the development and implementation of a microprocessor-based bicycle computer system using Raspberry Pi and Arduino platforms is examined. The main objective of the research is to create a multifunctional device that provides cycling enthusiasts with the ability to monitor various trip parameters such as speed, distance, time, as well as offering space for further improvement and the addition of new systems and features. Key words: microprocessor, Arduino, Raspberry Pi, bicycle computer, SPI.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

RPI — Raspberry Pi.

IDE — Integrated Development Environment (Інтегроване середовище розробки).

IoT — Internet of Things (Інтернет речей).

GPS — Global Positioning System (Глобальна навігаційна супутникова система).

SPI — Serial Peripheral Interface (Серійний периферійний інтерфейс).

ADC — Analog-to-Digital Converter (Аналого-цифровий перетворювач).

PWM — Pulse Width Modulation (Широтно-імпульсна модуляція).

API — Application Programming Interface (Інтерфейс прикладного програмування).

GUI — Graphical User Interface (Графічний інтерфейс користувача).

LCD — Liquid Crystal Display (Рідкокристалічний дисплей).

ВСТУП

У сучасному світі велосипед стає все більш **актуальним** засобом пересування як серед тих, хто любить активний відпочинок, так і серед тих, хто використовує його для повсякденних подорожей. Це пов'язано не тільки з турботою про здоров'я та екологію, а й з прагненням до мобільності та незалежності від пробок. Велосипед дозволяє легко маневрувати в міських умовах і дозволяє дістатися до місць, де інші види транспорту можуть бути обмежені. Крім того, велоспорт сприяє фізичному розвитку і загальному поліпшенню самопочуття.

Мета: Розробка мікропроцесорної системи велокомп’ютера на Raspberry PI , що забезпечить велосипедистам можливість ефективного моніторингу параметрів поїздки, таких як швидкість, відстань, та час, з можливістю збереження даних для подальшого аналізу та розширення функціональності.

Задачі дослідження:

— зробити максимально гнучку систему щоб кожен користувач міг обрати саме те що йому потрібно в данний момент, або в майбутньому не купувати інший пристрій викидаючи на смітник старий

— можливість додавати різноманітні аксессуари

Об'єктом дослідження є процеси що протікають у систем на мікроконтролерах з різноманітними датчиками та протоколами

Предметом дослідження є методи та засоби створення власного пристрою на основі мікроконтролерів та мікрокомп’ютерів.

Практична цінність даної роботи полягає в її здатності забезпечити інноваційні рішення для велосипедистів через розробку велокомп'ютера на основі Raspberry Pi. Це дає можливість створювати персоналізовані пристрої з розширеними функціями, які не завжди доступні в комерційних продуктах. Наприклад, можна реалізувати GPS-трекінг, моніторинг серцевого ритму та аналіз маршруту в реальному часі, що підвищує комфорт і безпеку користувача.

Також хочу зазначити, що робота з Raspberry Pi має велику освітню цінність, оскільки сприяє розвитку практичних навичок в області програмування,

електроніки та системної інтеграції. Використання цього мікрокомп'ютера дозволяє зрозуміти основи роботи з апаратним та програмним забезпеченням, що є надзвичайно важливим для формування професійних компетенцій. Це особливо корисно для студентів та інженерів, які хочуть поглибити свої знання в галузі мікропроцесорних систем та Інтернету речей (IoT). Завдяки доступності та багатофункціональності Raspberry Pi, студенти можуть вивчати різні аспекти обчислювальної техніки, починаючи від базових операцій і закінчуючи розробкою комплексних інтегрованих систем.

Ця робота також відкриває можливості для подальших досліджень та інновацій у сфері велоспорту та мобільних технологій. Вона може стати основою для розробки нових додатків та сервісів, таких як аналітика даних, машинне навчання для прогнозування та оптимізації маршрутів, інтеграція з іншими IoT-пристроями. Наприклад, на основі зібраних даних можна створювати алгоритми, які допомагатимуть велосипедистам обирати найефективніші маршрути, враховуючи поточні погодні умови, дорожню ситуацію та інші фактори. Крім того, інтеграція з іншими IoT-пристроями може забезпечити розширені можливості для моніторингу та керування спортивним обладнанням, підвищуючи безпеку та комфорт під час велопоїздок.

Дослідження у цій галузі також можуть сприяти розвитку нових технологій для покращення спортивних результатів. Наприклад, використання даних з сенсорів може допомогти у створенні персоналізованих тренувальних програм, що базуються на індивідуальних фізіологічних показниках спортсмена. Це може включати контроль серцевого ритму, аналіз потужності педалювання, визначення оптимальної швидкості та каденсу для різних умов.

Таким чином, розробка мікропроцесорної системи велокомп’ютера на основі Raspberry Pi не лише має велике практичне значення для велосипедистів, але й відкриває широкі можливості для подальшого розвитку технологій у сфері спорту та IoT, сприяючи інноваціям та підвищенню ефективності у цих галузях.

1 РОЗГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛОГІВ

1.1 Визначення основних термінів

Велокомп’ютер — це електронний пристрій, призначений для збору, обробки та відображення різноманітних даних, пов'язаних з їздою на велосипеді. Він може вимірювати такі параметри, як швидкість, відстань, час, висота та інші дані, які допомагають велосипедисту оптимізувати свої тренування та підвищити безпеку на дорозі.

Raspberry Pi — мініатюрний одноплатний комп'ютер(рисунок 1.1), розроблений Raspberry Pi Foundation. Він призначений для навчання програмуванню і створення різноманітних електронних проектів. Raspberry Pi є доступним за ціною і має потужні обчислювальні можливості, що робить його ідеальним для використання в освітніх і хобі-проектах.

Рисунок 1.1 — Raspberry Pi

Мікропроцесорна система — це комплекс апаратного та програмного забезпечення, що включає мікропроцесор, пам'ять та периферійні пристрої. Ці системи використовуються для виконання обчислень та керування іншими пристроями. У контексті велокомп'ютера мікропроцесорна система забезпечує

обробку даних від датчиків і керування відображенням інформації на дисплеї, в цій роботі мікропроцессорною системою виступає Arduino Nano (рисунок 1.2).

Рисунок 1.2 — Arduino NANO

Інтернет речей (IoT) — концепція об'єднання різноманітних пристроїв в одну мережу для обміну даними і виконання завдань без участі людини. У велокомп'ютері IoT може включати інтеграцію з іншими розумними пристроями, такими як смартфони або фітнес-трекери, для більш детального моніторингу і аналізу даних.

GPS-трекінг — система глобального позиціонування, яка використовується для визначення точного місця розташування об'єкта на основі сигналів від супутників. У велокомп'ютері GPS-трекінг дозволяє відстежувати маршрут і швидкість руху, а також надавати інформацію про пройдену відстань і поточне місцезнаходження.

Датчик — пристрій, що виявляє і вимірює фізичні явища, такі як температура, вологість, тиск, швидкість (рисунок 1.3), і перетворює ці вимірювання на сигнали,

які можуть бути прочитані іншими пристроями або системами. У велокомп'ютері датчики можуть використовуватися для вимірювання швидкості, частоти обертів педалей, температури та інших параметрів.

Рисунок 1.3 — Приклад датчика Холла

1.2 Огляд предметної області

Велокомп'ютери займають важливе місце у сучасному світі велосипедного спорту та активного відпочинку. Їх основною метою є надання велосипедистам необхідної інформації для аналізу своїх поїздок, тренувань та стану здоров'я. Велокомп'ютери можуть значно відрізнятися за функціональністю, дизайном та ціною, що дозволяє задовольняти потреби різних категорій користувачів – від початківців до професійних спортсменів.

Розробка мікропроцесорної системи велокомп’ютера на Raspberry Pi є надзвичайно важливим і актуальним завданням у контексті сучасного технологічного прогресу та зростаючого інтересу до здорового способу життя. Велокомп'ютери, як пристрої, що надають велосипедистам інформацію про їхні поїздки, відіграють важливу роль у покращенні якості тренувань та повсякденного

використання велосипеда. Сучасні велокомп'ютери надають широкий спектр функцій, включаючи вимірювання швидкості, відстані, часу поїздки, а також можуть містити додаткові функції, такі як моніторинг пульсу, навігація за допомогою GPS, а також підключення до смартфонів для синхронізації даних.

Одним із основних компонентів велокомп’ютера є система вимірювання швидкості, яка зазвичай базується на використанні датчиків холла або магнітних сенсорів, встановлених на колесі велосипеда. Ці датчики зчитують обертання колеса і передають відповідні сигнали на мікропроцесор, який обробляє ці дані і обчислює поточну швидкість, а також загальну пройдену дистанцію. Для відображення цієї інформації використовуються різні типи дисплеїв, зокрема LCD або OLED екрани(рис 1.4), які забезпечують чітке і зручне відображення необхідних даних в реальному часі.

Рисунок 1.4 — Дисплей для Raspberry PI

Крім того, важливим аспектом роботи велокомп’ютера є можливість зберігання даних про поїздки. Це дозволяє велосипедистам аналізувати свої тренування, відстежувати прогрес і планувати майбутні поїздки на основі зібраних даних. Зберігання даних може бути реалізоване на внутрішній пам'яті пристрою або на зовнішніх носіях, таких як карти пам'яті SD. Більшість сучасних велокомп'ютерів також підтримують синхронізацію з мобільними додатками або веб-сервісами, що дозволяє зберігати і аналізувати дані на смартфоні або комп'ютері.

Велокомп'ютери можуть включати додаткові функції, такі як GPS-навігація, яка дозволяє велосипедистам відстежувати своє місцезнаходження, планувати маршрути і отримувати інформацію про поточну локацію. Це особливо корисно для велосипедистів, які подорожують на великі відстані або використовують велосипед для дослідження нових місць. Також сучасні велокомп'ютери можуть мати функції моніторингу здоров'я, зокрема вимірювання пульсу, що дозволяє користувачам стежити за своїм фізичним станом під час поїздки.

Розробка цієї системи велокомп’ютера на Raspberry Pi відкриває широкі можливості для розширення функціональності пристрою. Raspberry Pi є потужною платформою, яка надає розробникам можливість створювати складні і багатофункціональні системи. Зокрема, використання Raspberry Pi дозволяє інтегрувати додаткові модулі і сенсори, такі як GPS, Bluetooth, Wi-Fi, а також камери для зйомки фото і відео під час поїздки. Це забезпечує велосипедистам широкий спектр можливостей для аналізу і поліпшення своїх поїздок.

Іншим важливим аспектом розробки є енергозабезпечення велокомп'ютера. Оскільки пристрій працює в автономному режимі, важливо забезпечити його достатньою кількістю енергії для тривалої роботи. Це може бути реалізовано за допомогою використання акумуляторів з великою ємністю або систем генерації енергії під час поїздки, наприклад, за допомогою динамо-генератора(рис. 1.5), встановленого на колесі велосипеда.

Рисунок 1.5 — Динамо-генератор

Основні функції велокомп'ютерів включають вимірювання швидкості, дистанції, часу у дорозі та інших параметрів. Вони допомагають велосипедистам підтримувати бажаний темп, аналізувати середню швидкість для оцінки загального рівня фізичної підготовки, відстежувати максимальну швидкість для аналізу спринтерських можливостей, а також вимірювати пройдену відстань для планування маршрутів та контролю фізичного навантаження. Додаткові функції включають годинник та таймер для планування зупинок і активностей під час поїздки, а також вимірювання каденсу, що допомагає оптимізувати педалювання та зменшити втому.

Сучасні велокомп'ютери також можуть мати GPS-навігацію, що дозволяє відстежувати маршрут у реальному часі, планувати поїздки та зберігати дані для подальшого аналізу. Підключення датчиків серцевого ритму дозволяє контролювати інтенсивність тренування та слідкувати за здоров'ям. Інтеграція з мобільними додатками забезпечує зберігання та аналіз даних, а також можливість ділитися результатами з іншими користувачами.

Розвиток технологій у сфері електроніки та мікропроцесорних платформ значно розширив можливості сучасних велокомп'ютерів. Вони використовують передові сенсори, бездротові технології передачі даних та потужні мікропроцесори, що дозволяють реалізовувати складні функції. Наприклад, індуктивні та магнітні датчики забезпечують високу точність вимірювань швидкості та частоти обертання педалей, а використання глобальної системи позиціонування (GPS) дозволяє відстежувати маршрут у реальному часі.

Мікропроцесорні платформи, такі як Arduino та Raspberry Pi або інші, дозволяють створювати кастомні велокомп'ютери з високим рівнем налаштування та розширення функціоналу. Вони є доступними за ціною, що робить їх привабливими для розробників-аматорів та професіоналів. Спеціалізовані мікроконтролери, які використовуються у комерційних велокомп'ютерах, забезпечують високу інтеграцію функцій при низькому енергоспоживанні.

Бездротові технології, такі як Bluetooth , дозволяють підключати додаткові сенсори та синхронізувати велокомп'ютер з мобільними пристроями для зберігання та аналізу даних. Це відкриває нові можливості для велосипедистів, які прагнуть оптимізувати свої тренування та поїздки.

Сучасні велокомп'ютери стикаються з викликами, такими як висока вартість преміум-моделей, складність налаштування та використання, а також високе енергоспоживання, що потребує частого заряджання. Проте, кастомні рішення на базі платформ, таких як Raspberry Pi, дозволяють створювати індивідуальні рішення з необхідним набором функцій та налаштувань. Це забезпечує гнучкість та можливість розширення функціоналу, підключення додаткових сенсорів та інтеграцію з мобільними додатками.

1.3 Аналіз існуючих аналогів

Аналіз існуючих аналогів велокомп'ютерів на ринку допомагає зрозуміти їх основні характеристики, переваги та недоліки, а також визначити напрямки для розробки кращого кастомного рішення. Велокомп'ютери можна розділити на три основні категорії: бюджетні, середнього цінового сегмента та преміум-класу. Розглянемо кожну з цих категорій детальніше.

Бюджетні велокомп'ютери, такі як Cateye Velo 7 (рис.1.4), забезпечують базові функції вимірювання швидкості, дистанції та часу. Ці моделі орієнтовані на початківців та велосипедистів, які не потребують розширеного функціоналу.

Рисунок 1.4 — Бюджетний сегмент

Основні характеристики:

— поточна швидкість, відображає швидкість руху в режимі реального часу;

— середня швидкість, показує середню швидкість за весь час поїздки;

— дистанція, вимірює загальну пройдену відстань;

— час у дорозі, відображає загальний час руху.

Переваги:

— доступна ціна, Бюджетні моделі є доступними для широкого кола користувачів;

— простота використання, легкість налаштування та використання робить їх привабливими для новачків.

Недоліки:

— обмежене функціонування, відсутність розширених функцій, таких як GPS та моніторинг серцевого ритму.

—обмежена можливість розширення, неможливість підключення додаткових сенсорів та модулів.

Велокомп'ютери середнього цінового сегмента, такі як Garmin Edge 130 (рис.1.5), пропонують додаткові функції та кращу якість виконання, порівняно з бюджетними моделями. Вони задовольняють потреби більш вимогливих користувачів.

Рисунок 1.5 — Середній сегмент

Основні характеристики:

— GPS-навігація, вбудований GPS-модуль для відстеження маршруту.

— підключення до сенсорів, можливість підключення датчиків серцевого ритму та каденсу.

— синхронізація з мобільними додатками, інтеграція з мобільними додатками для зберігання та аналізу даних.

Переваги:

— розширений функціонал, наявність додаткових функцій, таких як GPS та моніторинг серцевого ритму.

— підвищена точність, краща точність вимірювань завдяки використанню передових сенсорів та технологій.

Недоліки:

— вища ціна, велокомп'ютери середнього цінового сегмента дорожчі, ніж бюджетні моделі.

— складність налаштування, більш складні у налаштуванні та використанні, що може вимагати певних технічних знань.

Преміум велокомп'ютери, такі як Wahoo ELEMNT ROAM (рис.1.6), забезпечують найширший спектр можливостей та найвищу якість виконання. Вони орієнтовані на професійних спортсменів та ентузіастів, які потребують максимальних функціональних можливостей.

Рисунок 1.6 — Преміум сегмент

Основні характеристики:

— детальна GPS-навігація, розширена навігаційна система з картами та маршрутами.

— аналіз тренувань, поглиблений аналіз тренувань з використанням різних датчиків.

— інтеграція з екосистемою, підключення до різних додатків та платформ для обміну даними.

Переваги:

— максимальний функціонал, найширший спектр функцій, включаючи детальну GPS-навігацію, моніторинг серцевого ритму, каденсу та інші показники.

— професійний аналіз, можливість детального аналізу тренувань для покращення результатів.

Недоліки:

— висока ціна, преміум моделі є найдорожчими на ринку.

— високе енергоспоживання, потребують частого заряджання через використання багатьох функцій та сенсорів.

Використання платформи Raspberry Pi для розробки велокомп'ютера відкриває широкі можливості для створення функціонального та доступного пристрою, який можна налаштовувати відповідно до індивідуальних потреб користувача. Це є доступною за ціною платформою, що робить її привабливою для розробників-аматорів та професіоналів.

Основні переваги:

— гнучкість і модульність, можливість підключення різноманітних додаткових модулів та сенсорів, таких як GPS, акселерометри, датчики каденсу та інші. Це забезпечує можливість розширення функціоналу велокомп'ютера відповідно до вимог користувача.

— велика та активна спільнота розробників надає доступ до численних ресурсів, навчальних матеріалів та готових проектів. Це значно полегшує процес розробки та налагодження пристрою.

– простота програмування, програмування на базі мови C/C++ робить Raspberry Pi зручним інструментом для створення кастомних рішень. Це дозволяє

користувачам налаштовувати пристрій під свої індивідуальні потреби, вносити зміни у функціонал та додавати нові можливості.

Одна з основних переваг багатьох існуючих велокомп'ютерів – це їх зручність та простота використання. Інтерфейси цих пристроїв часто розроблені з урахуванням ергономіки, що дозволяє велосипедистам швидко та легко отримувати доступ до необхідної інформації під час руху. Крім того, більшість сучасних велокомп'ютерів мають водонепроникні корпуси та високу стійкість до механічних впливів, що забезпечує їх довговічність і надійність в експлуатації навіть у складних погодних умовах.

Разом з тим, варто зазначити, що існуючі аналоги мають і певні недоліки. Одним з таких є обмежена можливість персоналізації та розширення функціоналу. Багато стандартних моделей не підтримують інтеграцію з додатковими датчиками та модулями, що може обмежити їх функціональність і зробити їх менш привабливими для користувачів, які прагнуть мати більше можливостей для аналізу та покращення своїх поїздок. Висока вартість деяких преміум-моделей також може стати перепоною для широкого кола користувачів.

Ще одним важливим аспектом є точність і надійність вимірювань. Деякі велокомп'ютери можуть мати проблеми з точністю вимірювання швидкості та дистанції, що може призводити до неправильного відображення даних. Це особливо критично для спортсменів і велосипедистів, які серйозно займаються тренуваннями і потребують точних даних для аналізу своїх результатів. Крім того, обмежена ємність акумуляторів деяких моделей може стати проблемою для користувачів, які здійснюють тривалі поїздки і не мають можливості часто заряджати пристрій.

Враховуючи всі ці аспекти, розробка нової системи велокомп'ютера на базі Raspberry Pi має на меті подолати існуючі обмеження і запропонувати користувачам більш гнучке, надійне та функціональне рішення. Використання таких потужних і водночас компактних платформ як Raspberry Pi та Arduino дозволяє створити пристрій, який можна легко адаптувати під індивідуальні потреби користувача, розширювати його функціональні можливості шляхом

додавання нових датчиків і модулів, а також забезпечити високу точність і надійність вимірювань.

Зрештою кожна з цих платформ має свої переваги та недоліки, і вибір конкретного рішення залежить від потреб користувачів, їхніх уподобань та специфіки завдань, які вони хочуть вирішити через комунікацію на мобільних платформах. При аналізі існуючих аналогів слід враховувати різні аспекти, такі як зручність використання, надійність, вартість, а також можливості розширення та адаптації до індивідуальних потреб користувачів. Сучасні велокомп'ютери пропонують широкий спектр функціональних можливостей, від базових моделей, які забезпечують вимірювання швидкості та дистанції, до більш просунутих систем, що включають GPS-навігацію (рисунок 1.7), моніторинг серцевого ритму та інтеграцію зі смартфонами.

Рисунок 1.7 — GPS модуль

Таким чином, нова система велокомп'ютера буде не лише відповідати сучасним вимогам велосипедистів, але й відкривати нові можливості для

вдосконалення тренувань та повсякденного використання велосипеда. Вона стане інструментом, який допоможе велосипедистам покращити свої результати, відстежувати прогрес та планувати нові маршрути, забезпечуючи при цьому високий рівень зручності та надійності в експлуатаці.

2 МЕТОДОЛОГІЯ ТА МЕТОДИ РОЗРОБКИ

2.1 Визначення вимог

Визначення вимог до розробки мікропроцесорної системи велокомп'ютера на базі Raspberry Pi є важливим етапом, який забезпечує успішність реалізації проекту. Цей процес починається з аналізу потреб користувачів та їх очікувань від системи. Основними вимогами є точність вимірювань, зручність використання, надійність та функціональність. Важливо визначити, які параметри мають бути виміряні та відображені, такі як швидкість, пройдена відстань, висота, температура, вологість та інтенсивність освітлення.

Функціональні вимоги включають здатність системи зчитувати дані з різних датчиків, обробляти їх та відображати результати в режимі реального часу. Важливо забезпечити можливість оновлення даних з інтервалами, які дозволять користувачам отримувати актуальну інформацію під час поїздки. Окрім цього, система повинна мати можливість збереження даних для подальшого аналізу та використання в інших додатках.

Не менш важливими є нефункціональні вимоги, такі як надійність та стабільність роботи системи. Велокомп'ютер має працювати в різних погодних умовах та бути стійким до вібрацій і ударів, які можуть виникати під час їзди. Енергоефективність також є критичним аспектом, оскільки система повинна функціонувати тривалий час без необхідності частої заміни або зарядки акумулятора.

Інтерфейс користувача повинен бути інтуїтивно зрозумілим, з чітким відображенням всіх необхідних даних. Зручність монтажу та підключення системи до велосипеда також є важливим аспектом. Вимоги до програмного забезпечення включають використання сучасних технологій, таких як Python та C, бібліотеки для роботи з датчиками та GPS, а також розробку зручного інтерфейсу для користувача.

Для забезпечення точності вимірювань система має використовувати високоякісні датчики та алгоритми обробки даних. Важливо провести тестування всіх компонентів системи в реальних умовах, щоб переконатися в їх надійності та

точності. Зокрема, варто перевірити, як система обробляє дані від датчиків швидкості, висоти, температури, вологи та освітлення.

Також необхідно врахувати вимоги до апаратного забезпечення. Raspberry Pi обрано як основну платформу через її потужність, гнучкість та підтримку широкого спектра периферійних пристроїв. Вимоги до пам'яті та обчислювальної потужності повинні бути визначені на основі обсягу даних, що обробляються, та необхідної швидкості їх обробки.

Для забезпечення безпеки даних варто передбачити механізми захисту від несанкціонованого доступу та можливості резервного копіювання інформації. Важливо також забезпечити зручні методи оновлення програмного забезпечення для додавання нових функцій та виправлення помилок.

Підсумовуючи, визначення вимог до мікропроцесорної системи велокомп'ютера на Raspberry Pi є критичним етапом, який включає аналіз потреб користувачів, вибір необхідних функцій та параметрів, забезпечення надійності та стабільності системи, а також використання сучасних технологій для досягнення високої точності та зручності у використанні.

2.1.1 Аналіз ринку та аналогів

Аналіз ринку та аналогів мікропроцесорних систем для велокомп'ютерів є важливим етапом при розробці нового продукту, оскільки він дозволяє виявити поточні тенденції, конкурентів та існуючі рішення на ринку. Це також допомагає визначити унікальні переваги та можливості для покращення майбутнього продукту.

Перше, що необхідно зробити, – це детально вивчити поточний ринок велокомп'ютерів. Це включає аналіз доступних продуктів різних цінових категорій: бюджетних, середнього цінового сегмента та преміум-класу. Аналіз включає виявлення основних характеристик, функцій та технологій, які використовуються в цих продуктах. Важливо зрозуміти, які функції є базовими і обов'язковими для всіх велокомп'ютерів, а які є додатковими та розширюють функціонал. Вище я наводив приклади аналогів та те що вони пропонують

На ринку представлено багато моделей велокомп'ютерів від різних виробників. Деякі з них забезпечують базові функції, такі як вимірювання швидкості та дистанції, тоді як інші пропонують розширені можливості, включаючи GPS-навігацію, моніторинг стану здоров'я та інтеграцію зі смартфонами. Важливо проаналізувати ключові характеристики цих продуктів, їх вартість, функціональність та відгуки користувачів.

Сучасні велокомп'ютери, як правило, використовують різні датчики для вимірювання швидкості, висоти, температури та інших параметрів. Одним із провідних брендів на ринку є Garmin, який пропонує велокомп'ютери з вбудованими GPS-модулями, можливістю синхронізації з мобільними додатками та підтримкою ANT+ для підключення додаткових датчиків. Такі продукти, як Garmin Edge, є високотехнологічними пристроями з широким набором функцій, але їх вартість часто є значною.

Інші виробники, як Sigma Sport та Cateye, пропонують велокомп'ютери з більш обмеженим набором функцій за доступнішою ціною. Вони забезпечують основні можливості, такі як вимірювання швидкості, дистанції та часу поїздки, але не мають розширених функцій, таких як GPS-навігація чи моніторинг стану здоров'я.

Аналізуючи аналоги, важливо звернути увагу на їхні переваги та недоліки. Наприклад, велокомп'ютери з GPS можуть запропонувати точне відстеження маршруту та навігацію, але їх вартість та енергоспоживання можуть бути значними. З іншого боку, простіші моделі можуть бути доступнішими та економічнішими, але їх функціональність обмежена.

Ще одним важливим аспектом є програмне забезпечення, що використовується в цих пристроях. Багато сучасних велокомп'ютерів мають супутні мобільні додатки, які дозволяють аналізувати дані поїздок, встановлювати цілі та ділитися результатами з іншими користувачами. Наприклад, додатки для велокомп'ютерів Garmin дозволяють користувачам зберігати дані на хмарі, що забезпечує зручний доступ до інформації з будь-якого пристрою.

Важливо також враховувати відгуки користувачів про існуючі продукти. Це допомагає виявити основні проблеми, з якими стикаються користувачі, та визначити можливості для покращення нового продукту. Наприклад, часті скарги на низьку точність датчиків чи складність інтерфейсу можуть стати важливими орієнтирами при розробці.

Таким чином, аналіз ринку та аналогів дозволяє отримати чітке уявлення про поточні тенденції та вимоги до велокомп'ютерів. Це є важливим кроком у розробці нового продукту, який буде конкурентоспроможним, зручним у використанні та відповідає потребам користувачів.

2.1.2 Збір вимог від користувачів

Наступний крок – це збір вимог безпосередньо від потенційних користувачів велокомп'ютера. Це включає опитування велосипедистів різного рівня підготовки — від новачків до професіоналів. Основна мета цього етапу — зрозуміти, які функції та характеристики є найбільш важливими для кінцевих користувачів.

Це є ключовим етапом при розробці мікропроцесорної системи велокомп'ютера на Raspberry Pi. Відповідний збір даних дозволяє зрозуміти потреби та очікування користувачів, що забезпечує розробку продукту, який дійсно відповідає їхнім запитам та покращує користувацький досвід.

Першим кроком у цьому процесі є визначення цільової аудиторії. Велокомп'ютер може бути корисним для різних категорій користувачів, таких як професійні велосипедисти, аматори, фітнес-ентузіасти та навіть звичайні користувачі, які використовують велосипед як основний транспортний засіб. Кожна з цих груп має свої специфічні вимоги та очікування від продукту.

Для збору вимог можна використовувати різні методи, включаючи опитування, інтерв'ю, фокус-групи та аналіз відгуків користувачів на існуючі продукти. Опитування та інтерв'ю дозволяють отримати безпосередній зворотний зв'язок від потенційних користувачів щодо їхніх потреб та побажань. Наприклад, можна дізнатися, які функції велокомп'ютера є найбільш важливими, наскільки

важлива точність вимірювань, які додаткові можливості були б бажаними (наприклад, інтеграція з мобільними додатками чи підтримка GPS-навігації).

Фокус-групи дозволяють більш детально обговорити вимоги користувачів у невеликих групах, що дає можливість отримати глибше розуміння їхніх потреб та проблем. Під час таких обговорень можна виявити не лише очевидні вимоги, але й приховані потреби, які користувачі можуть не усвідомлювати безпосередньо.

Аналіз відгуків користувачів на існуючі продукти є також важливим джерелом інформації. Це дозволяє виявити сильні та слабкі сторони конкурентних продуктів, зрозуміти, що подобається та не подобається користувачам, і які функції вони хотіли б бачити у новому продукті. Наприклад, якщо користувачі часто скаржаться на короткий час автономної роботи або складність налаштування, це може стати важливим аспектом для покращення.

Після збору всієї необхідної інформації, важливо систематизувати та проаналізувати дані, щоб визначити ключові вимоги до продукту. Це включає як функціональні вимоги (наприклад, вимірювання швидкості, дистанції, висоти), так і нефункціональні вимоги (наприклад, зручність використання, час автономної роботи, надійність).

Зрештою, на основі зібраних вимог, можна розробити технічне завдання, яке буде чітко визначати всі необхідні функції та характеристики велокомп'ютера. Це забезпечить, що розробка продукту буде орієнтована на задоволення реальних потреб користувачів та дозволить створити конкурентоспроможний продукт на ринку.

2.1.3 Визначення технічних характеристик

Визначення технічних характеристик є ключовим етапом у розробці мікропроцесорної системи велокомп'ютера на основі Raspberry Pi. Цей процес включає визначення всіх необхідних параметрів і функцій, які забезпечать виконання вимог користувачів та досягнення поставлених цілей проекту.

Перше, що потрібно врахувати, це апаратна платформа. Raspberry Pi обрана як основа для велокомп'ютера завдяки її потужності, гнучкості та підтримці різноманітних периферійних пристроїв. Основні технічні характеристики включають процесор, оперативну пам'ять, сховища даних, порти для підключення зовнішніх датчиків та модулів (GPS, Bluetooth, Wi-Fi).

Однією з ключових функцій велокомп'ютера є вимірювання швидкості та дистанції. Для цього необхідно використовувати точні датчики руху, такі як герконові датчики або датчики Холла, які можуть бути підключені до GPIO-портів Raspberry Pi. Важливо врахувати частоту опитування датчиків для забезпечення точних вимірювань в реальному часі.

Наступною важливою функцією є GPS-навігація. Модуль GPS забезпечує визначення поточного місцезнаходження, швидкості руху та інших навігаційних даних. Важливо вибрати надійний GPS-модуль, який забезпечить високу точність та швидкість оновлення даних. Для інтеграції GPS-модуля з Raspberry Pi можна використовувати стандартні інтерфейси, такі як UART(рис. 2.1) або USB.

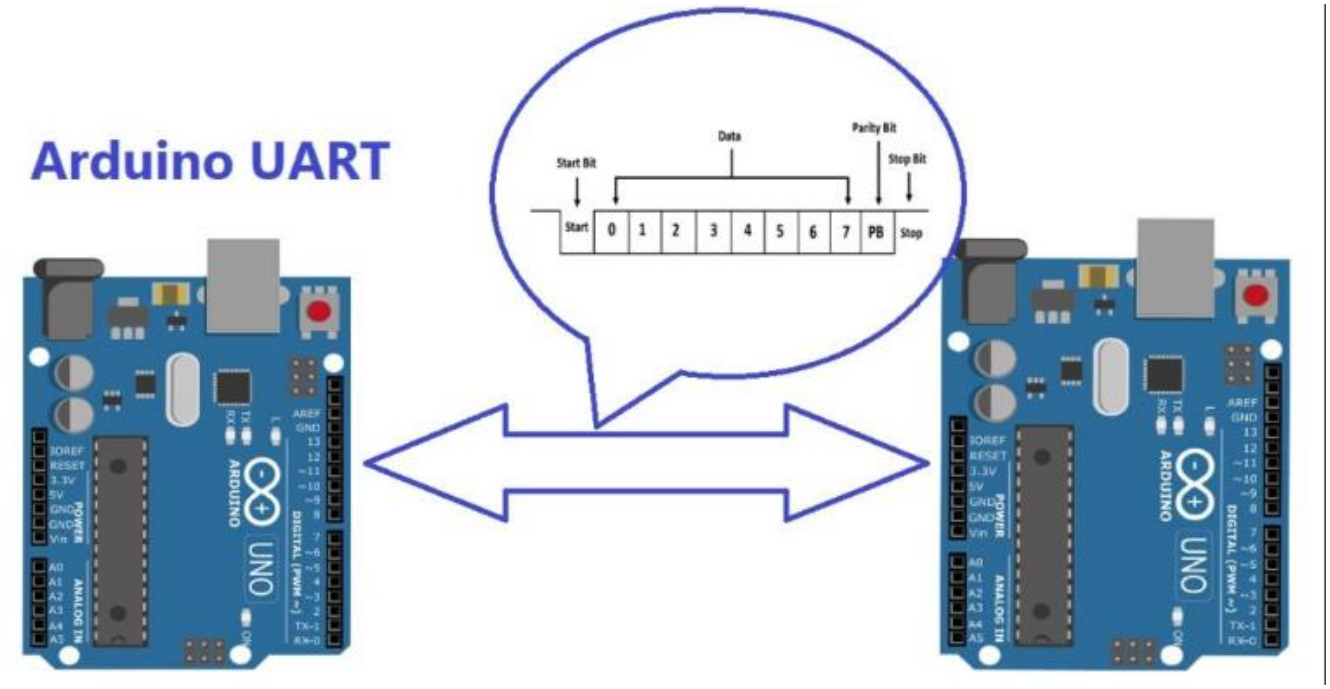

Рисунок 2.1 — Робота UART

Для зберігання даних про поїздки та інші параметри необхідно передбачити достатній об'єм пам'яті. Це можуть бути як вбудовані сховища Raspberry Pi, так і зовнішні накопичувачі, такі як microSD-карти або USB-флешки. Важливо також передбачити можливість резервного копіювання та відновлення даних.

Інтерфейс користувача має бути зручним та інтуїтивно зрозумілим. Для цього можна використовувати дисплей з високою роздільною здатністю, який буде відображати основну інформацію, таку як швидкість, дистанція, поточний час, навігаційні вказівки тощо. Крім того, можна передбачити використання кнопок або сенсорного екрану для зручного керування функціями велокомп'ютера.

Живлення системи є критично важливим аспектом. Велокомп'ютер повинен мати достатню автономність, щоб забезпечити тривалу роботу під час поїздок. Для цього можна використовувати літій-іонні або літій-полімерні акумулятори, які забезпечують високу ємність та стабільну роботу. Важливо також передбачити ефективне енергоспоживання системи та можливість заряджання акумуляторів від зовнішніх джерел, таких як сонячні панелі або генератори.

Рисунок 2.2 — Вигляд акумулятора

Комунікаційні можливості велокомп'ютера також мають велике значення. Наявність модулів Bluetooth та Wi-Fi (рис. 2.3) дозволить передавати дані на мобільні пристрої або в хмарні сервіси для подальшого аналізу та зберігання. Це забезпечить можливість синхронізації даних, віддаленого керування та отримання оновлень програмного забезпечення.

Рисунок 2.3 — Модуль Bluetooth та Wi-Fi

Забезпечення безпеки даних та захист від несанкціонованого доступу є ще одним важливим аспектом. Для цього необхідно передбачити механізми

шифрування даних, автентифікації користувачів та захисту від зломів. Це забезпечить збереження конфіденційності інформації та безпеку роботи системи.

Визначення технічних характеристик також включає розробку програмного забезпечення, яке буде забезпечувати роботу всіх компонентів системи, обробку даних з датчиків, взаємодію з користувачем та іншими пристроями. Важливо забезпечити оптимальну продуктивність програмного забезпечення, його стабільність та надійність, а також можливість подальшого оновлення та модернізації.

2.1.4 Формування функціональних вимог

Основною функціональною вимогою є вимірювання швидкості та дистанції під час поїздок на велосипеді. Система повинна точно вимірювати швидкість руху та пройдену дистанцію, використовуючи відповідні датчики руху. Це можуть бути герконові датчики або датчики Холла, які фіксують обертання колеса велосипеда та передають ці дані на Raspberry Pi для обробки.

Наступною важливою функцією є GPS-навігація(рис 2.4). Велокомп'ютер повинен бути здатний визначати поточне місцезнаходження користувача за допомогою GPS-модуля. Це дозволить відслідковувати маршрут поїздки, визначати напрямок руху та надавати користувачу навігаційні вказівки.

Рисунок 2.4 — Принцип роботи GPS

Система повинна забезпечувати точне і швидке оновлення даних GPS, щоб користувач завжди був в курсі свого поточного положення та маршруту.

Для зручності користувача, велокомп'ютер повинен мати зрозумілий та інтуїтивний інтерфейс. Дисплей з високою роздільною здатністю повинен відображати основну інформацію, таку як швидкість, дистанція, поточний час, навігаційні вказівки, стан батареї та інші важливі дані. Інтерфейс повинен бути простим у використанні, з можливістю легко переключатися між різними режимами та функціями.

Збір та зберігання даних є ще однією важливою функцією велокомп'ютера. Система повинна записувати дані про кожну поїздку, включаючи швидкість, дистанцію, маршрут, висоту над рівнем моря та інші параметри. Ці дані можуть бути збережені на внутрішній пам'яті Raspberry Pi або на зовнішніх носіях, таких як microSD-карти. Важливо передбачити можливість резервного копіювання та відновлення даних.

Комунікаційні можливості також є важливим аспектом функціональних вимог. Велокомп'ютер повинен мати можливість передавати дані на мобільні пристрої або в хмарні сервіси через Bluetooth або Wi-Fi (рис. 2.5). Це дозволить користувачу синхронізувати дані, отримувати оновлення програмного забезпечення та керувати системою віддалено. Крім того, система може підтримувати інтеграцію з додатками для фітнесу та спорту, що дозволить користувачу відслідковувати свої досягнення та планувати тренування.

Рисунок 2.5 — Хмарні сервіси

Автономність роботи системи є ще однією важливою функціональною вимогою. Велокомп'ютер повинен мати достатню автономність для тривалих поїздок, що забезпечується за рахунок використання літій-іонних або літій-полімерних акумуляторів. Система повинна бути енергоефективною, з можливістю заряджання від зовнішніх джерел, таких як сонячні панелі або генератори.

Безпека даних та захист від несанкціонованого доступу є критично важливими функціональними вимогами. Велокомп'ютер повинен забезпечувати шифрування даних, автентифікацію користувачів та захист від зломів. Це забезпечить збереження конфіденційності інформації та безпеку роботи системи.

Останнім, але не менш важливим аспектом, є можливість подальшого оновлення та модернізації системи. Програмне забезпечення велокомп'ютера повинно бути гнучким та легко оновлюваним, щоб забезпечити впровадження

нових функцій та виправлення можливих помилок. Це дозволить системі залишатися актуальною та відповідати вимогам користувачів у майбутньому.

Формування функціональних вимог є основою для розробки ефективної та надійної мікропроцесорної системи велокомп'ютера, яка задовольнить всі потреби користувачів та забезпечить високу якість і зручність використання.

2.2 Обґрунтування вибору Raspberry Pi

Під час аналізу доступних мікропроцесорних платформ для використання у велокомп'ютері, я звернув увагу на декілька ключових критеріїв, які визначають їхню придатність для мого проекту. Raspberry Pi та Arduino вирізняються на фоні інших платформ своєю популярністю, розширеними можливостями та активною спільнотою користувачів, що сприяє швидкому розвитку та підтримці проекту.

Arduino, з одного боку, здатна ефективно взаємодіяти з різноманітними сенсорами та пристроями завдяки своїй простоті та низькому рівню складності програмування. Вона пропонує простий інтерфейс для роботи з цифровими та аналоговими сигналами, що робить її ідеальним вибором для виконання базових завдань велокомп'ютера, таких як вимірювання швидкості та дистанції.

З іншого боку, Raspberry Pi відома своєю високою обчислювальною потужністю та можливістю виконувати складні програмні задачі, що вимагають більшого обсягу обробки даних. Вона пропонує повноцінну операційну систему, яка дає можливість використовувати різноманітні програмні бібліотеки та інструменти для створення розширеного функціоналу велокомп'ютера, такого як GPS-навігація, зв'язок з Інтернетом та обробка відеоданих.

Обидві платформи мають широкий вибір розширень та аксесуарів, що дозволяє вибрати оптимальний набір компонентів для конкретних потреб проекту. Крім того, вони мають велику спільноту користувачів, яка активно ділиться досвідом та знаннями, що є важливим фактором для успішної реалізації проекту та вирішення можливих проблем.

Таким чином, обрання Arduino та Raspberry Pi для мого проекту велокомп'ютера було обґрунтоване їхніми технічними характеристиками,

доступністю компонентів та підтримкою спільноти користувачів, що забезпечує успішну та ефективну реалізацію задуманого проекту.

2.3 Використання протоколу SPI задля взаємодією між Raspberry Pi та Arduino

Протокол SPI (Serial Peripheral Interface) - це синхронний протокол зв'язку, який дозволяє обмінюватися даними між мікроконтролерами або мікропроцесорами та периферійними пристроями (Рисунок 2.6). У зв'язку за допомогою SPI використовуються чотири провідника:

- SCLK (Serial Clock), це сигнал генерації тактових імпульсів, який синхронізує передавання та приймання даних між пристроями;
- MOSI (Master Out Slave In), це лінія, по якій майстер (у нашому випадку Raspberry Pi) надсилає дані до підчиненого пристрою (Arduino Nano);
- MISO (Master In Slave Out), ця лінія використовується для передачі даних від підчиненого пристрою до майстра;
- SS/CS (Slave Select/Chip Select), це сигнал, який вказує підчиненому пристрою, що він повинен приймати дані. Кожен підчинений пристрій має свій власний SS/CS сигнал;

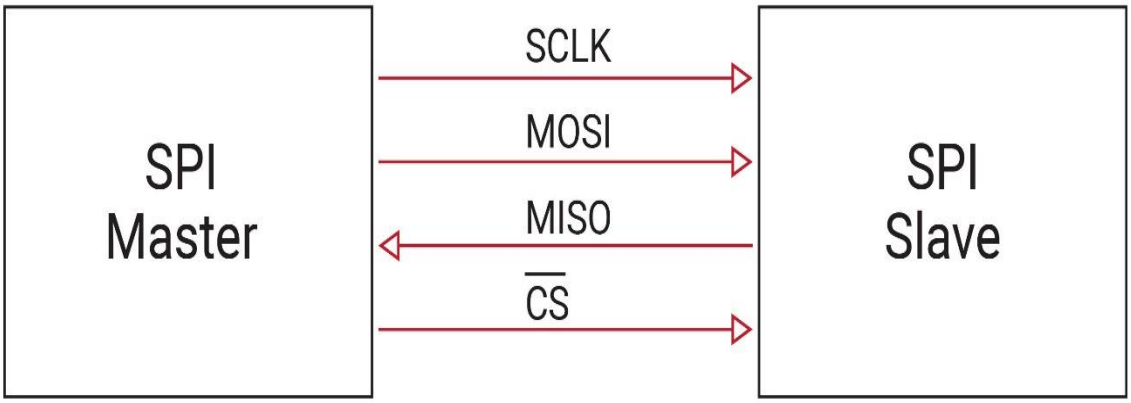

Рисунок 2.6 — Протокол SPI

Тепер, щодо підключення Raspberry Pi та Arduino Nano за допомогою SPI. На Raspberry Pi є GPIO (General Purpose Input/Output) піни, які можна налаштувати для роботи в режимі SPI. Щоб підключити Raspberry Pi до Arduino Nano, ти можеш використовувати наступну схему підключення:

- Пін SCLK Raspberry Pi підключається до піна SCK (Serial Clock) на Arduino Nano;
- Пін MOSI Raspberry Pi підключається до піна MOSI (Master Out Slave In) на Arduino Nano;
- Пін MISO Raspberry Pi підключається до піна MISO (Master In Slave Out) на Arduino Nano;
- Пін SS/CS Raspberry Pi підключається до піна SS/CS на Arduino Nano.

Крім цього, обов'язково пам'ятай про необхідність підключення живлення та землі (GND) для кожного пристрою.

Отже, коли з'єднання налаштовано, Raspberry Pi може ініціювати зчитування або передачу даних за допомогою протоколу SPI. Наприклад, як і використовується в роботі датчик на Arduino Nano, Raspberry Pi може надіслати запит на зчитування даних, а Arduino Nano відповість, відправляючи виміряні показники по лінії MISO.

Таким чином, за допомогою протоколу SPI та відповідних підключень, можна забезпечити ефективну взаємодію між Raspberry Pi та Arduino Nano для отримання даних з датчиків та виконання різних завдань у мікропроцесорній системі.

2.4 Використання Arduino IDE для розробки логіки программи

Використання Arduino IDE (Рисунок 2.7) для розробки логіки програми є ключовим аспектом мого проекту велокомп'ютера. Arduino IDE (Integrated Development Environment) — це середовище програмування, спеціально призначене для розробки програм для платформ Arduino. Це зручний інструмент, який надає широкий набір функцій та інструментів для створення та налагодження програм для мікроконтролерів Arduino.

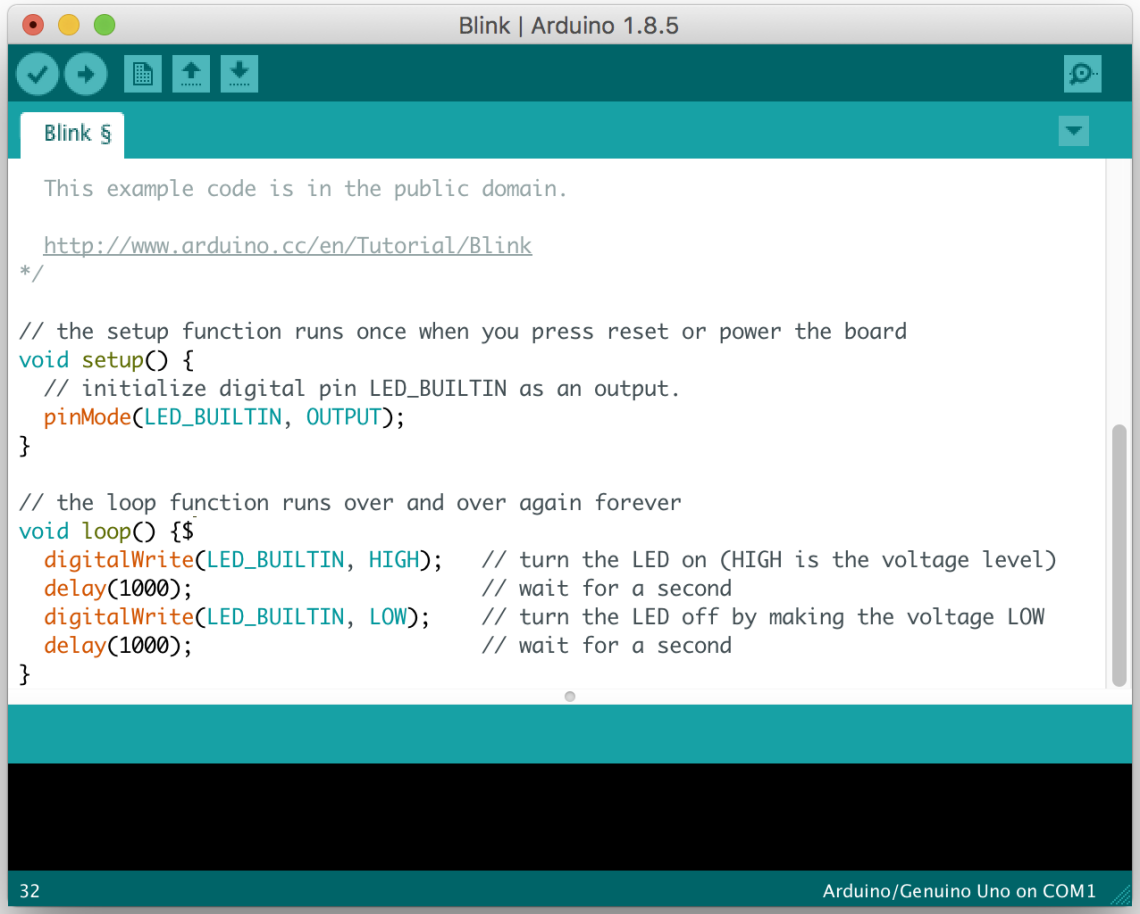

Рисунок 2.7 — Середовище ArduinoIDE

Основні переваги використання Arduino IDE включають його простоту використання та доступність для початківців, що робить його ідеальним вибором для студента, як я. Його інтерфейс має інтуїтивно зрозумілу структуру, яка дозволяє швидко освоїти основні функції та почати розробку програм.

Крім того, Arduino IDE має вбудований компілятор та середовище виконання, що дозволяє швидко перевіряти та відлагоджувати програми без необхідності встановлення додаткових програмних засобів. Це дозволяє зосередитися на розробці логіки програми та тестуванні її функціональності без зайвих витрат часу на налаштування середовища розробки.

Крім того, Arduino IDE підтримує велику кількість бібліотек та скетчів, що дозволяє використовувати готові рішення та зразки коду для швидкої реалізації

різноманітних функцій та задач. Це забезпечує ефективний процес розробки та дозволяє зосередитися на реалізації конкретної функціональності в моєму проекті велокомп'ютера.

Узагальнюючи, використання Arduino IDE надає широкі можливості для розробки програмного забезпечення для велокомп'ютера, забезпечуючи зручний та ефективний спосіб програмування мікроконтролерів та інтеграції різноманітних пристроїв та функціональностей.

3 РЕАЛІЗАЦІЯ ПРОЕКТУ ТА ЙОГО ФУНКЦІОНАЛУ

3.1 Створення проекту

Налаштування проекту в Arduino IDE, зокрема створення нового проекту, є першим кроком у розробці програмного забезпечення для велокомп'ютера. Для цього відкриваємо Arduino IDE і виконуємо наступні кроки:

відкрийте Arduino IDE на вашому комп'ютері;

у головному вікні Arduino IDE оберіть "File" (Файл) у верхньому меню, потім "New" (Новий) для створення нового скетчу (програми);

дайте проекту назву, оберіть платформу Arduino, яку ви використовуєте (наприклад, Arduino Nano), та виберіть порт, до якого підключений ваш Arduino;

Arduino IDE створить для вас новий файл скетчу з розширенням ".ino". Цей файл буде головним файлом вашого проекту, де ви будете писати код для велокомп'ютера;

збережіть свій проект на вашому комп'ютері, вибравши "File" (Файл) у верхньому меню, а потім "Save As" (Зберегти як), та оберіть шлях та назву для вашого проекту;

тепер ви готові почати розробку логіки програми для вашого велокомп'ютера. Ви можете почати писати код у файлі скетчу, використовуючи мову програмування Arduino, яка базується на мові C++.

Створення нового проекту в Arduino IDE дозволяє швидко почати роботу над вашим проектом велокомп'ютера та відразу перейти до програмування без зайвих складнощів.

Структура проекту в Arduino IDE проста і зрозуміла, що робить її ідеальною для студентів, як я. Основним елементом є головний файл скетчу з розширенням ".ino", де ми пишемо наш код. У цьому файлі ми визначаємо змінні, функції та всю логіку нашого проекту.

Крім того, ми можемо використовувати зовнішні файли бібліотек для розширення функціональності нашого проекту. Це дозволяє нам використовувати

готові рішення для роботи з датчиками, дисплеями та іншими пристроями, що прискорює розробку.

Також в нашому проекті можуть бути константи та параметри налаштувань, які можуть бути визначені окремо або в головному файлі скетчу. Це дозволяє нам змінювати параметри проекту без необхідності змінювати код.

Важливо також додавати коментарі до нашого коду, щоб пояснити, як він працює. Це допомагає нам та іншим розробникам краще розуміти функціональність проекту та швидше знаходити помилки.

Загалом, структура проекту в Arduino IDE проста і зрозуміла, що робить процес розробки приємним та ефективним для студентів, які тільки вчаться програмувати на мові Arduino. На (рис.3.1) зображено перші дії для створення нового проекту.

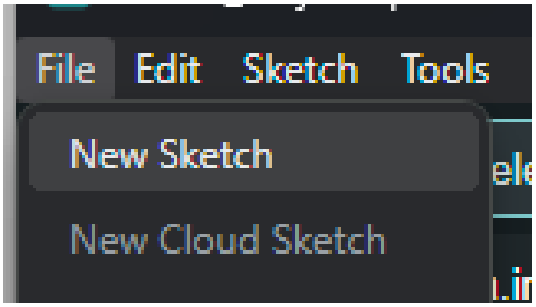

Рисунок 3.1 — Створення нового файлу

Далі у відповідному вікні обираємо «Select Board» (рис. 3.2).

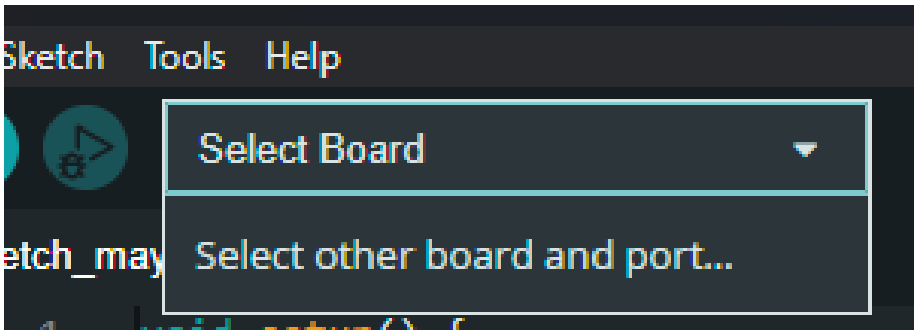

Рисунок 3.2 — Вибір своєї “плати ”

Наступним кроком потрібно знайти свою плату та підключити її(рис. 3.3).

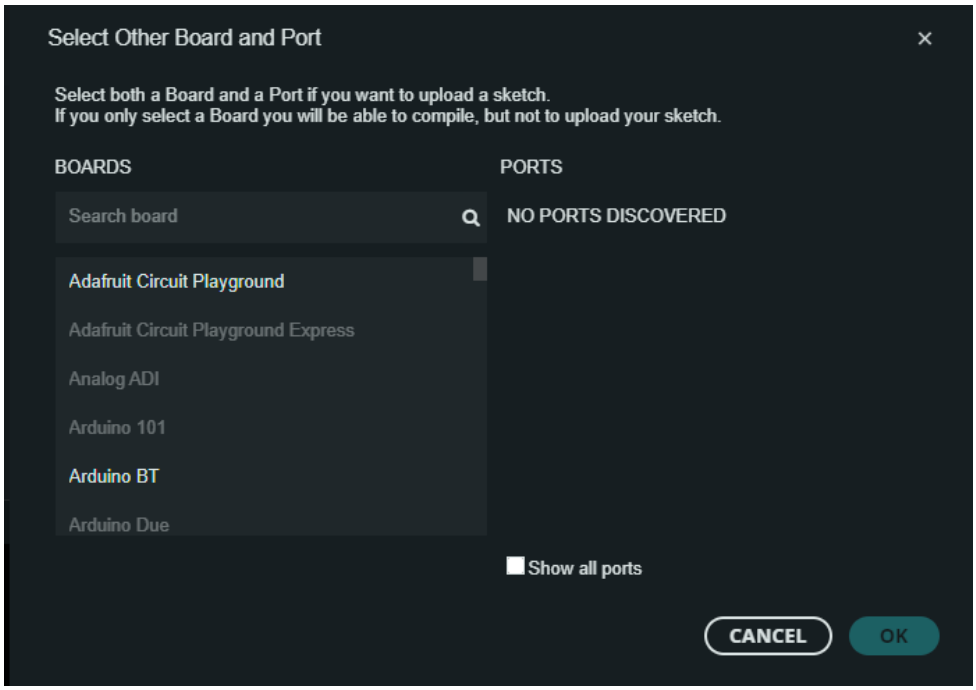

Рисунок 3.3 — Підключення плати

3.2 Додавання сторонніх бібліотек

Крок 1 — відкриття Arduino IDE. Запустіть Arduino IDE на вашому комп'ютері.

Крок 2 — вибір "Manage Libraries" (Керування бібліотеками)

У верхньому меню оберіть "Sketch" (Скетч).

У випадаючому меню оберіть "Include Library" (Включити бібліотеку).

У підменю оберіть "Manage Libraries" (Керування бібліотеками) (рис.3.4).

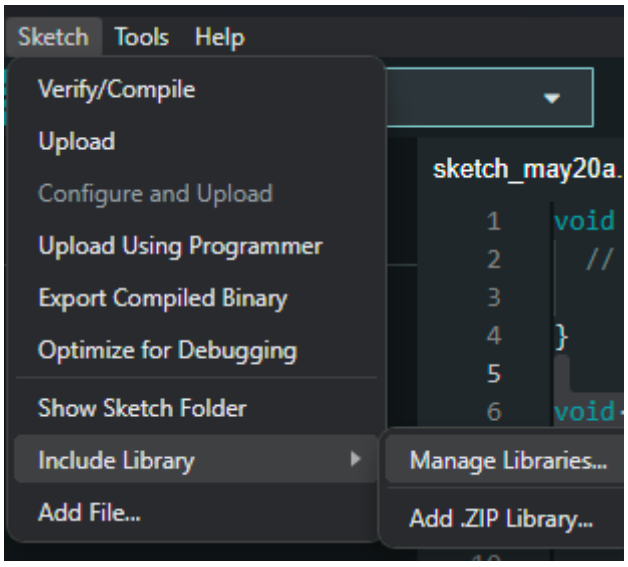

Рисунок 3.4 — Додавання бібліотек

В Arduino IDE доступний інструментарій для управління бібліотеками, який називається "Library Manager" (Менеджер бібліотек). Цей інструмент надає можливість користувачам знайти, встановити та керувати різними бібліотеками, які розширюють функціональність Arduino.

Щоб скористатися Менеджером бібліотек, користувач повинен відкрити Arduino IDE і вибрати меню "Sketch" (Скетч), після чого обрати пункт "Include Library" (Включити бібліотеку) і "Manage Libraries" (Менеджер бібліотек) (рисунок 3.5). Після цього з'явиться вікно Менеджера бібліотек, де користувач може виконати пошук бібліотек за назвою, типом або темою.

Користуючись цим інструментом, користувач може швидко знайти необхідні бібліотеки, встановити їх у своє середовище Arduino IDE та оновлювати до останніх версій. Крім того, він може керувати встановленими бібліотеками, дозволяючи видаляти ті, які більше не потрібні, або вимикаючи їх, щоб зменшити обсяг програми. Це значно спрощує процес управління залежностями та забезпечує можливість легкого налаштування середовища розробки під конкретні потреби проекту.

Завдяки цьому інструменту, користувачі отримують доступ до великої кількості перевірених і оптимізованих бібліотек, що можуть суттєво прискорити процес розробки та підвищити якість кінцевого продукту. Бібліотеки часто включають готові рішення для роботи з датчиками, модулями зв'язку, дисплеями та іншими апаратними компонентами, що дозволяє зосередитися на логіці додатка замість вирішення низькорівневих технічних питань.

Враховуючи цей інструмент, користувачам відкривається доступ до широкого спектру ресурсів та можливостей для розширення функціональності їх проектів на Arduino, сприяючи їхній ефективності та продуктивності. Це особливо важливо для тих, хто займається проектами у сфері IoT, автоматизації, робототехніки та інших областях, де час та точність виконання завдань мають критичне значення.

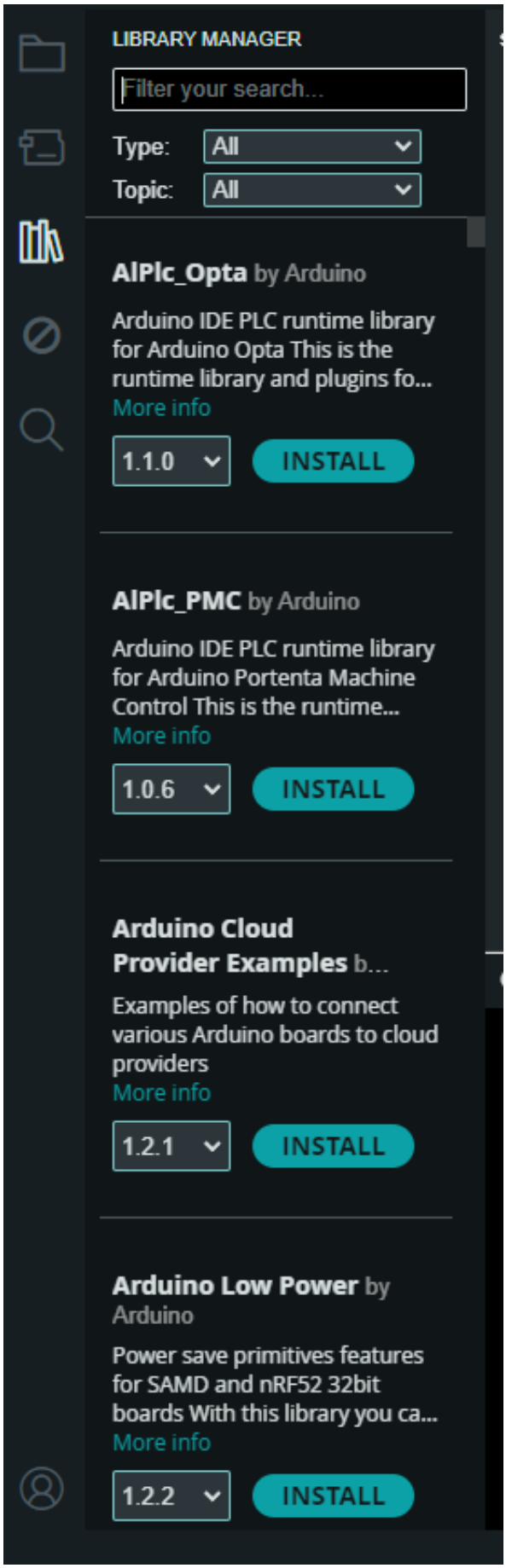

Рисунок 3.5 — Інсталяція потрібних бібліотек

Після того, як ви знайшли потрібну бібліотеку, оберіть її зі списку.

Натисніть кнопку "Install" (Встановити) біля вибраної бібліотеки та версії (рис.3.6).

Рисунок 3.6 — Інсталювання бібліотеки

Почекайте, поки Arduino IDE завантажить та встановить бібліотеку.

Після завершення встановлення оберіть "Sketch" (Скетч) у верхньому меню.

У підменю оберіть "Include Library" (Включити бібліотеку).

Оберіть бібліотеку зі списку, яку ви хочете включити до вашого проекту.

Це дозволяє вам використовувати функції та класи, які містяться у встановленій бібліотеці, у вашому коді. Наприклад, якщо ви встановили бібліотеку для роботи з LCD дисплеєм, ви можете використовувати її функції для виводу тексту та зображень на вашому дисплеї.

3.3 Основні пункти

Реалізація функціоналу велокомп'ютера на базі платформ Raspberry Pi та Arduino включає декілька ключових етапів:

1) вимірювання швидкості та дистанції:

— використання датчика холла для вимірювання швидкості;

— обробка сигналів датчика холла на платформі Arduino;

— використання алгоритму для розрахунку швидкості та відстані на основі зчитаних даних;

— передавання даниих до Raspberry

— відображення поточної швидкості, дистанції та іншої інформації на дисплеї.

2) Зберігання даних:

— використання пам'яті Arduino або зовнішніх носіїв даних для зберігання даних про поїздки.

— запис та зчитування даних для подальшого аналізу або відображення.

3) Розширюваність:

— підтримка додаткових модулів та сенсорів для розширення функціоналу.

— можливість додавання нових функцій та покращень через програмне забезпечення.

Ці етапи утворюють основу функціональності велокомп'ютера та забезпечують можливість вимірювання та відображення важливої інформації для велосипедистів.

У даному проекті використовується протокол SPI (Serial Peripheral Interface), що є потужним і широко використовуваним засобом для обміну даними між мікроконтролерами та периферійними пристроями. Він дозволяє передавати дані швидко та ефективно, що особливо корисно для реалізації проектів з вбудованими системами.

У контексті цього проекту, Python забезпечує можливість взаємодії з SPI завдяки бібліотеці `spidev`, що дає можливість зчитувати дані з пристроїв, підключених через SPI. Функція `read_sensor_data()` відкриває SPI пристрій, встановлює параметри комунікації та здійснює взаємодію з Arduino через SPI для зчитування даних про швидкість, температуру та вологість.

Одним із ключових елементів проекту є функція `update_sensor_data()`, яка періодично оновлює відображення даних на екрані. Вона забезпечує постійне зчитування даних з Arduino через SPI та їх відображення у вікні програми на Raspberry Pi. Крім того, ця функція керує оновленням візуального спідометра, який дозволяє користувачу легко візуалізувати поточну швидкість.

В цілому, завдяки використанню протоколу SPI та бібліотеки `spidev`, проект забезпечує ефективний обмін даними між Raspberry Pi та Arduino, що є ключовим аспектом багатьох проектів з вбудованими системами.

3.4 Модуль GPS

В цій роботі також використовується модуль GPS для отримання географічних координат, які можна використовувати для відстеження місцезнаходження. Отже, для включення модуля GPS в коді Python для отримання географічних координат потрібно додати ініціалізацію GPS, що забезпечить зчитування даних з GPS-модуля.

Лістинг 1.1 Модуль GPS

```
import tkinter as tk
from tkinter import ttk
import spidev
import gpsd
```

Ініціалізація з'єднання з GPS:

```
gpsd.connect()
```

Функція connect() ініціалізує з'єднання з gpsd для отримання даних GPS.

```
def update_gps():
    gps_packet = gpsd.get_current()  # Отримання поточних даних з GPS
    gps_data.set('{}, {}'.format(gps_packet.lat, gps_packet.lon))  # Оновлення значення GPS
    root.after(1000, update_gps)  # Повторення оновлення через 1 секунду
```

gpsd.get_current() повертає об'єкт gps_packet, який містить інформацію про поточні GPS-координати.

gps_packet.lat і gps_packet.lon - це latitude (широта) і longitude (довгота) відповідно, які отримуються з модуля GPS.

gps_data.set('{}, {}'.format(gps_packet.lat, gps_packet.lon)) - оновлює значення gps_data (tk.StringVar), яке використовується для відображення на GUI.

root.after(1000, update_gps) - забезпечує періодичне оновлення GPS-даних кожну секунду.

```
gps_label = ttk.Label(root, text="GPS: ")
gps_label.grid(row=6, column=0, padx=10, pady=5, sticky="w")
gps_data = tk.StringVar()
gps_display = ttk.Label(root, textvariable=gps_data)
```

gps_display.grid(row=6, column=1, padx=10, pady=5, sticky="w")

gps_data - це tk.StringVar, яка використовується для відображення GPS-координат.

gps_display - ttk.Label, який відображає значення gps_data на GUI.

Запуск оновлення GPS-даних разом з іншими даними:

update_sensor_data() # Оновлення даних сенсорів

update_gps() # Оновлення GPS-даних

root.mainloop() # Запуск основного циклу tkinter

3.5 Датчик Холла

У проекті доданий датчик Холла для вимірювання обертів колеса або інших рухливих частин. Датчик Холла працює на принципі виявлення магнітного поля, що змінюється з рухом об'єкта, до якого він прикріплений.

Лістинг 1.2 Датчик Холла

boolean reedSwitchState;

int maxReedCount = 100;

int reedCount;

reedSwitchState - змінна типу boolean, яка зберігає стан датчика Холла (замкнутий або розімкнутий).

maxReedCount - максимальне значення лічильника для відстеження часу між сигналами датчика Холла.

reedCount - лічильник для відстеження кількості імпульсів від датчика Холла.

Ініціалізація в setup функції:

reedCount = maxReedCount;

pinMode(A0, INPUT);

reedCount встановлюється на максимальне значення.

Пін A0 налаштований як вхідний, до якого підключено датчик Холла.

Нижче наведена функція обробки переривань, яка виконується на частоті 1 кГц.

ISR(TIMER1_COMPA_vect) {

```
if (analogRead(reed) >= 680 && analogRead(reed) <= 742) {
  reedSwitchState = true;
} else {
  reedSwitchState = false;
}

if (reedSwitchState) {
  if (timeCounter > 110) {
    isMoving = true;
  }

  if (reedCount == 0) {
    if (isMoving) {
      speedKmh = (56.8 * float(wheelCircumference)) / float(timeCounter) * 1.61;
      tripDistance += distanceDelta;
      distanceAccumulator += distanceDelta;
    }
    reedCount = maxReedCount;
    timeCounter = 0;
  } else {
    if (reedCount > 0) {
      reedCount -= 1;
    }
  }
} else {
  if (reedCount > 0) {
    reedCount -= 1;
  }
}
```

```
  if (timeCounter > 2000) {
    speedKmh = 0;
    isMoving = false;
  } else {
    timeCounter += 1;
  }
}
```

analogRead(reed) - зчитує значення з піну, до якого підключений датчик Холла. Якщо значення в межах 680-742, то датчик Холла замкнутий (reedSwitchState = true).

Якщо датчик Холла замкнутий і timeCounter більше 110, встановлюється isMoving = true.

Якщо reedCount дорівнює 0, розраховується швидкість у км/год і оновлюється пройдена відстань.

Якщо датчик Холла не замкнутий, зменшується лічильник reedCount.

Якщо timeCounter більше 2000, швидкість встановлюється в 0 і isMoving = false.

Лістинг 1.3 Функція loop

```
void loop(void)
{
  if (reedSwitchState) {
    if (timeCounter <= 110) {
      isMoving = false;
    } else {
      isMoving = true;
    }
  }
  updateDisplay();
  saveData();
```

```
  if (isMoving) {
    int d = (int)((millis() - previousTime) / 1000);
    timeAccumulator += d;
    tripDuration += d;
  }

  previousTime = millis();

  delay(1000);
}
```

Якщо датчик Холла замкнутий і timeCounter менше або дорівнює 110, встановлюється isMoving = false, інакше isMoving = true.

Викликаються функції updateDisplay та saveData.

Якщо isMoving = true, оновлюються накопичені значення часу та тривалості поїздки.

Таким чином, датчик Холла відіграє ключову роль у визначенні обертів колеса, що дозволяє розраховувати швидкість руху та пройдену відстань. Код відповідає за зчитування стану датчика, обробку сигналів та оновлення відповідних змінних.

3.6 Інший функціонал пристрою

Велокомп’ютер на базі Raspberry Pi реалізований з метою збору, обробки та візуалізації різноманітної інформації для велосипедистів. Основні компоненти системи включають збір даних з датчиків (швидкість, температура, вологість, висота, тиск, інтенсивність світла), отримання географічних координат за допомогою GPS, а також відображення цих даних на дисплеї.

Зчитування даних з сенсорів: Програмне забезпечення взаємодіє з Arduino через SPI для отримання даних про швидкість, температуру, вологість, висоту, тиск та інтенсивність світла.

Лістинг 1.4

```
import tkinter as tk
from tkinter import ttk
import spidev

def read_sensor_data():
    spi = spidev.SpiDev()
    spi.open(0, 0)
    spi.max_speed_hz = 1000000

    # Відправка запиту на отримання даних
    spi.xfer([0x01])

    # Зчитування даних з SPI
    received_data = spi.readbytes(3)

    # Розпакування отриманих даних
    speed = received_data[0]
    temperature = received_data[1]
    humidity = received_data[2]

    return speed, temperature, humidity

def update_speed():
    new_speed = read_sensor_data()[0]
    speed_value.set("{:.1f} км/год".format(new_speed))
    root.after(1000, update_speed)

root = tk.Tk()
```

```
root.title("Велокомп'ютер")
root.geometry("800x600")

speed_label = ttk.Label(root, text="Швидкість: ")
speed_label.grid(row=0, column=0, padx=10, pady=5, sticky="w")
speed_value = tk.StringVar()
speed_display = ttk.Label(root, textvariable=speed_value)
speed_display.grid(row=0, column=1, padx=10, pady=5, sticky="w")

update_speed()

root.mainloop()
```

Зберігання та відображення даних, дані про швидкість, температуру, вологість, висоту, тиск, інтенсивність світла та географічні координати зберігаються в пам'яті Raspberry Pi та відображаються на екрані в реальному часі.

Лістинг 1.5

```
import tkinter as tk
from tkinter import ttk

# Функція для оновлення даних швидкості
def update_speed():
    # Отримання нових даних (замінити на реальні дані)
    new_speed = 25.5  # Припустимо, що це нова швидкість

    # Оновлення значення швидкості
    speed_value.set("{:.1f} км/год".format(new_speed))

    # Реєструємо наступне оновлення через 1 секунду
```

```
    root.after(1000, update_speed)

# Функція для оновлення GPS координат
def update_gps():
    # Отримання нових GPS координат (замінити на реальні дані)
    new_lat = 49.83826
    new_lon = 24.02324

    # Оновлення значення GPS координат
    gps_data.set("Широта: {:.5f}, Довгота: {:.5f}".format(new_lat, new_lon))

    # Реєструємо наступне оновлення через 1 секунду
    root.after(1000, update_gps)

# Створення основного вікна
root = tk.Tk()
root.title("Велокомп'ютер")
root.geometry("800x600")

# Відображення даних про швидкість
speed_label = ttk.Label(root, text="Швидкість: ")
speed_label.grid(row=0, column=0, padx=10, pady=5, sticky="w")
speed_value = tk.StringVar()
speed_display = ttk.Label(root, textvariable=speed_value)
speed_display.grid(row=0, column=1, padx=10, pady=5, sticky="w")

gps_label = ttk.Label(root, text="GPS: ")
gps_label.grid(row=1, column=0, padx=10, pady=5, sticky="w")
gps_data = tk.StringVar()
gps_display = ttk.Label(root, textvariable=gps_data)
```

```
gps_display.grid(row=1, column=1, padx=10, pady=5, sticky="w")

# Запуск функцій оновлення даних
update_speed()
update_gps()

# Запуск головного циклу обробки подій
root.mainloop()

root.mainloop()
```

Як було зазначено раніше, система складається з кількох основних компонентів: Raspberry Pi, Arduino та дисплея. Нижче детально описано кожен з них:

— Raspberry Pi використовується як основний обчислювальний модуль системи. Він забезпечує обробку даних, збереження інформації та взаємодію з іншими компонентами. Raspberry Pi може бути підключений до різних сенсорів і модулів, включаючи Arduino, для збору та обробки даних. На ньому також може бути встановлене програмне забезпечення для обробки даних і візуалізації інформації.

— Arduino використовується для збору даних від датчика Холла і інших сенсорів. Він підключається до Raspberry Pi через серійний порт або інші комунікаційні інтерфейси. Arduino відповідає за зчитування сигналів від датчика Холла, обчислення обертів колеса та передачу цих даних на Raspberry Pi для подальшої обробки.

— дисплей використовується для виведення інформації про швидкість, пройдену відстань, тривалість поїздки та поточний час. Він підключений до Raspberry Pi або Arduino (залежно від конфігурації) і відображає дані в реальному часі. У коді дисплей ініціалізується та оновлюється відповідними функціями, забезпечуючи користувачу доступ до актуальної інформації.

Всі ці пристрої та загальний вид системи можна побачити на рисунку 4.1.

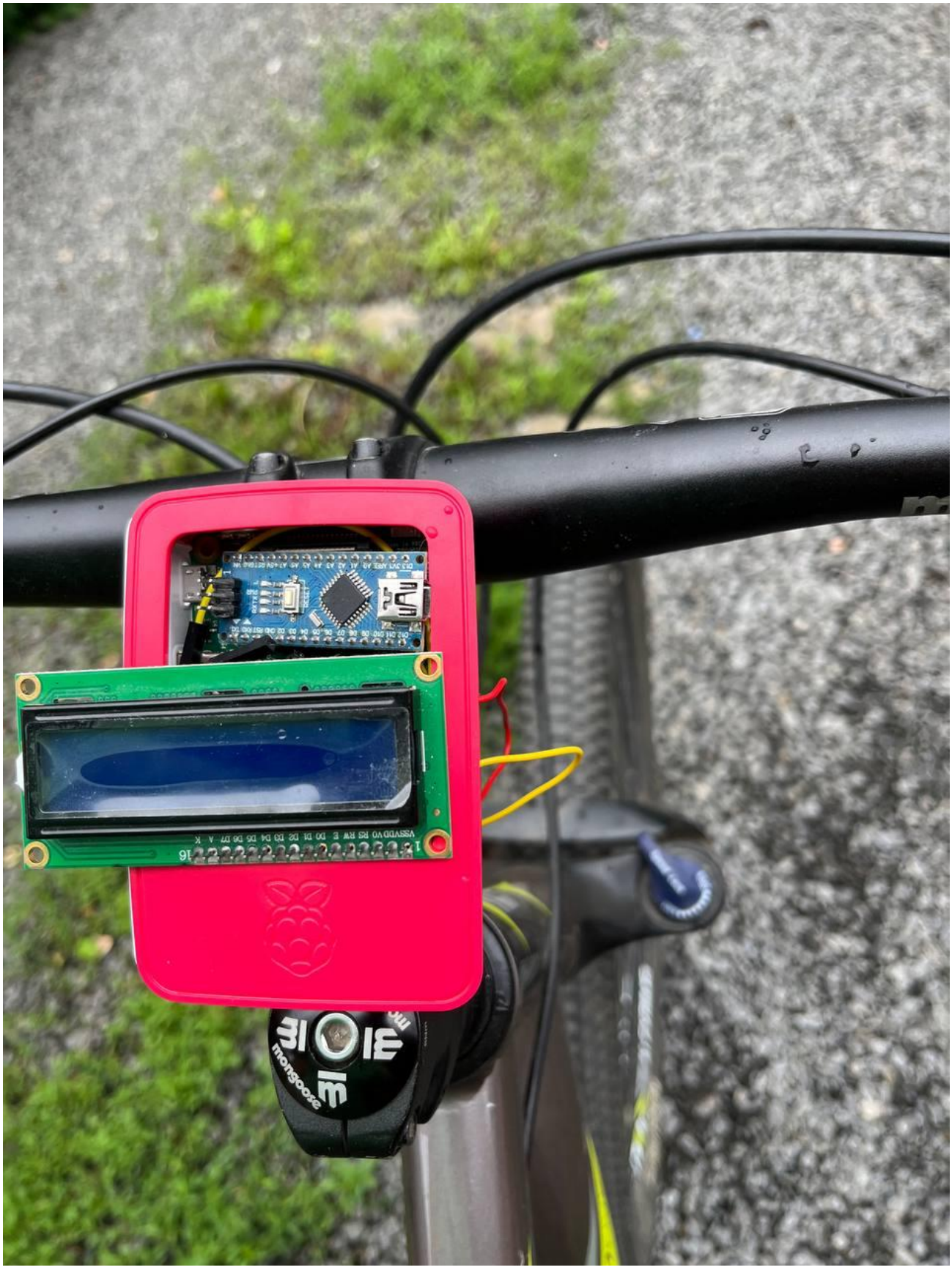

Рисунок 4.1 — Загальний вигляд

ВИСНОВКИ

У висновках проекту можна зазначити, що розробка мікропроцесорної системи велокомп'ютера на базі платформ Raspberry Pi та Arduino має великий потенціал для застосування у сфері велоспорту та рекреації. Використання цих платформ дозволяє створити пристрій, який надає велосипедистам зручний та ефективний спосіб моніторингу та аналізу різних параметрів поїздки, таких як швидкість, дистанція та час.

У ході дослідження було досягнуто важливих результатів, що свідчать про перспективність цього напрямку розробки. Наша система успішно вимірює потрібні параметри, інтегрується з різними сенсорами та модулями, а також має потенціал для подальшого розширення функціоналу. Це робить її корисним інструментом для велосипедистів будь-якого рівня.

Однак, варто враховувати деякі обмеження, зокрема, обмежені ресурси апаратної платформи, які можуть ускладнювати виконання деяких завдань. Також важливо враховувати питання енергоспоживання та живлення пристрою, особливо у довгих поїздках.

У майбутньому можливе подальше розширення функціоналу системи, наприклад, додавання нових сенсорів, підтримка зовнішніх пристроїв чи розробка спеціалізованого програмного забезпечення. Це дозволить покращити користувацький досвід та розширити сферу застосування велокомп'ютера.

Загалом, проект відкриває нові можливості для використання мікропроцесорних систем у велосипедній індустрії та сприяє розвитку сучасних технологій для поліпшення досвіду велосипедистів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. GPSD [Електронний ресурс] – Режим доступу: https://gpsd.gitlab.io/gpsd/

2. Робота з GPSD [Електронний ресурс] – Режим доступу: https://www.catb.org/gpsd/gpsd.html

3. PyGPSClient [Електронний ресурс] – Режим доступу: https://pypi.org/project/PyGPSClient/

4. Adafruit GPS Library [Електронний ресурс] – Режим доступу: https://learn.adafruit.com/adafruit-ultimate-gps

5. GPS Module with Arduino [Електронний ресурс] – Режим доступу: https://create.arduino.cc/projecthub/jeffpar0723/ultimate-gps-breakout-board-4c89e3

6. SPI Communication [Електронний ресурс] – Режим доступу: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html

7. Tkinter Documentation [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/tkinter.html

8. Python SPI Library (spidev) [Електронний ресурс] – Режим доступу: https://pypi.org/project/spidev/

9. Arduino SPI [Електронний ресурс] – Режим доступу: https://www.arduino.cc/en/reference/SPI

10. Arduino Library Manager [Електронний ресурс] – Режим доступу: https://www.arduino.cc/en/guide/libraries

11. LiquidCrystal Library for Arduino [Електронний ресурс] – Режим доступу: https://www.arduino.cc/en/Reference/LiquidCrystal

12. Arduino Timer Interrupts [Електронний ресурс] – Режим доступу: https://www.robotshop.com/community/forum/t/arduino-timer-and-interrupt-tutorial/13072

13. SD Card Module [Електронний ресурс] – Режим доступу: https://www.arduino.cc/en/Reference/SD

14. DS1302 RTC Module [Електронний ресурс] – Режим доступу: https://www.arduino.cc/reference/en/libraries/ds1302/

15. Realtime Data Display with Tkinter [Електронний ресурс] – Режим доступу: https://riptutorial.com/tkinter/example/32085/a-real-world-application---real-time-graphing

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н. О.Д. Азаров

«__» _____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

ІНТЕРАКТИВНИЙ ЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ОСОБИСТИМ БЮДЖЕТОМ

08-54.БДП.019.00.000 ПЗ

Керівник роботи: к.т.н., доц. каф. ОТ

_____Муращенко О.Г.

Виконав: студент гр. 2СП-20б

_____Бохенек Є.С.

Вінниця 2024

1 Підстава для виконання дипломної роботи (БДП):

— тема створення велокомп'ютера є актуальною через зростання популярності велосипедного спорту та активного способу життя. Велокомп'ютери забезпечують точний моніторинг швидкості, дистанції, часу та інших параметрів, допомагаючи велосипедистам покращувати свої результати, стежити за прогресом і підвищувати безпеку під час поїздок. Використання сучасних технологій, таких як GPS та сенсори, робить ці пристрої ще більш корисними та функціональними.

— наказ про затвердження теми бакалаврської дипломної роботи.

2 Мета і призначення БДП:

— метою проекту є створення практичної, дешевої та доповнюваної системи велокомп’ютера.

— призначення розробки — виконання бакалаврського дипломного проекту із подальшою підтримкою та розвитком.

3 Джерела розробки:

— технічний опис;

— технології розробки;

— технічна документація;

— теоретичні дані.

4 Технічні вимоги до виконання БДП:

— Коректна робота всієї системи;

— інтегрувати та перевірити працездатність системи.

5 Етапи роботи та очікуванні результати приведено в Таблиці А.1.

6 Матеріали, що подаються до захисту БДП: пояснювальна записка, графічні та ілюстративні матеріали, протокол попереднього захисту роботи на кафедрі, відгук наукового керівника, анотації українською та іноземною мовами, нормоконтроль про відповідність оформлення діючим вимогам.

Таблиця А.1 — Етапи БДР

№	Назва та зміст етапу	Термін виконання		Очікувані результати
		початок	закінчення	
	Вибір, узгодження та затвердження теми БДП			Задачі дослідження
	Аналіз літературних джерел. Попередня розробка основних розділів			Аналітичний огляд джерел
	Розробка технічного завдання (ТЗ)			ТЗ
	Огляд існуючих рішень та аналіз сервісу			Розділ 1
	Проектування інтерактивного застосунку для управління особистим бюджетом			Розділ 2
	Огляд застосунку та його тестування			Розділ 3
	Оформлення пояснювальної записки та графічної частини			ПЗ, презентація

7 Порядок контролю виконання та захисту БДП: виконання етапів графічної та розрахункової документації контролюється науковим керівником згідно зі встановленими термінами. Захист відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 При оформлювання БДП використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

Методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2022;

документами на які посилаються у вище вказаних.

9 Порядок виконання БДП викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21»

ДОДАТОК Б

Код Arduino

```
float tireRadius = 13.5; //тут потрібно радіус шини
boolean reedSwitchState;
long timeCounter = 0;
float speedKmh = 0.00;
float wheelCircumference;

float tripDistance = 0;

long totalTripDistance = 0;
long totalTripTime = 0;
float distanceAccumulator = 0;
long timeAccumulator = 0;
float distanceDelta;
boolean isMoving = false;
long currentTime = 0;
long previousTime = millis();
long tripDuration = 0;

DS1302 rtc(9, 8, 7);

int maxReedCount = 100;
int reedCount;

void setup(void)
{
  rtc.halt(false);
  rtc.writeProtect(false);
```

```
  pinMode(10, OUTPUT);

  reedCount = maxReedCount;
  distanceDelta = 2 * 3.415926535 * tireRadius * 0.025;
  wheelCircumference = 2 * 3.14 * tireRadius;
  pinMode(A0, INPUT);

  // НАЛАШТУВАННЯ ТАЙМЕРА - переривання таймера дозволяє точно вимірювати час спрацьовування геркона
  // для додаткової інформації про налаштування таймерів Arduino див. http://arduino.cc/playground/Code/Timer1
  cli(); //зупинити переривання

  //встановити переривання таймера1 на 1кГц
  TCCR1A = 0; // обнулити весь регістр TCCR1A
  TCCR1B = 0; // те ж саме для TCCR1B
  TCNT1  = 0;
  // встановити лічильник таймера на інкременти 1кГц
  OCR1A = 1999; // = (1/1000) / ((1/(16*10^6)) * 8) - 1
  // увімкнути режим CTC
  TCCR1B |= (1 << WGM12);
  // встановити біти CS11 для дільника на 8
  TCCR1B |= (1 << CS11);
  // увімкнути переривання порівняння таймера
  TIMSK1 |= (1 << OCIE1A);

  sei(); //дозволити переривання
  //КІНЕЦЬ НАЛАШТУВАННЯ ТАЙМЕРА
```

```
LcdInitialise();
LcdClear();

LcdString("Ініціалізація SD карти...");
pinMode(10, OUTPUT);
gotoXY(0, 1);
if (!SD.begin(chipSelect)) {
  LcdString("Карта не вдалася або не присутня");
} else {
  LcdString("Карту ініціалізовано.");
  File logFile = SD.open("logfile.txt");
  if (logFile) {
    gotoXY(0, 2);
    LcdString("читання файлу");
    while (logFile.available()) {
      totalTripDistance = logFile.parseInt();
      totalTripTime = logFile.parseInt();
    }
  } else {
    gotoXY(0, 2);
    LcdString("помилка читання файлу");
  }
}

LcdClear();
gotoXY(0, 0);
LcdString("Шв.:"); // швидкість Км/Год
gotoXY(0, 1);
LcdString("Дст.:"); // дистанція за поїздку в метрах
gotoXY(0, 2);
```

```
  LcdString("Длт.:"); // тривалість поїздки в хвилинах
  gotoXY(0, 3);
  LcdString("Год.:"); // години
}

ISR(TIMER1_COMPA_vect) { //Переривання на частоті 1 кГц для перевірки геркона
  if (analogRead(reed) >= 680 && analogRead(reed) <= 742) { //Геркон замкнуто при таких показаннях
    reedSwitchState = true;
  } else {
    reedSwitchState = false;
  }

  if (reedSwitchState) { //Геркон замкнуто
    if (timeCounter > 110) {
      isMoving = true;
    }

    if (reedCount == 0) { //Мінімальний час між імпульсами пройшов
      if (isMoving) {
        speedKmh = (56.8 * float(wheelCircumference)) / float(timeCounter) * 1.61; //кілометри в годину
        tripDistance += distanceDelta;
        distanceAccumulator += distanceDelta;
      }
      reedCount = maxReedCount; //Скидаємо reedCount
      timeCounter = 0; //Скидаємо таймер геркона
    } else {
      if (reedCount > 0) { //Не дозволяємо зменшення нижче нуля
```

```
        reedCount -= 1; //Зменшуємо reedCount
      }
    }
  } else { //Якщо геркон не замкнуто
    if (reedCount > 0) { //Не дозволяємо зменшення нижче нуля
      reedCount -= 1; //Зменшуємо reedCount
    }
  }
  if (timeCounter > 2000) {
    speedKmh = 0; //Якщо довго немає сигналів від геркона, ми стоїмо.
    isMoving = false;
  } else {
    timeCounter += 1; //Збільшуємо таймер
  }
}

void updateDisplay() {
  for (int i = 0; i <= 3; i++) {
    gotoXY(28, i);
    LcdString("       ");
  }

  char Sensor1CharMsg[8];

  gotoXY(28, 0);
  if (isMoving) {
    String Sensor1String((int)(speedKmh), DEC);
    Sensor1String.toCharArray(Sensor1CharMsg, (Sensor1String.length() + 1));
    LcdString(Sensor1CharMsg);
  } else {
```

```
    LcdString("0");
  }
  LcdString("Км/Год");

  gotoXY(28, 1);
  String Sensor1String2((int)(tripDistance), DEC);
  Sensor1String2.toCharArray(Sensor1CharMsg, (Sensor1String2.length() + 1));
  LcdString(Sensor1CharMsg);
  LcdString("м");

  gotoXY(28, 2);
  String Sensor1String3((int)(tripDuration / 60), DEC);
  Sensor1String3.toCharArray(Sensor1CharMsg, (Sensor1String3.length() + 1));
  LcdString(Sensor1CharMsg);
  LcdString("хв");

  gotoXY(28, 3);
  String Sensor1String4(rtc.getTimeStr(FORMAT_SHORT));
  Sensor1String4.toCharArray(Sensor1CharMsg, (Sensor1String4.length() + 1));
  LcdString(Sensor1CharMsg);
}

void loop(void)
{
  if (reedSwitchState) { //Геркон замкнуто
    if (timeCounter <= 110) {
      isMoving = false; //Ми зупинилися, замкнувши геркон
    } else {
      isMoving = true;
    }
```

```
  }
  updateDisplay();
  saveData();

  if (isMoving) {
    int d = (int)((millis() - previousTime) / 1000);
    timeAccumulator += d;
    tripDuration += d;
  }

  previousTime = millis();

  delay(1000);
}

void saveData() {
  while (timeAccumulator >= 60) {
    timeAccumulator -= 60;
    totalTripTime++;
  }
  while (distanceAccumulator >= 11) {
    distanceAccumulator -= 11;
    totalTripDistance += 11;
  }
}
```

Додаток В

Код Raspberry PI

```
import tkinter as tk
from tkinter import ttk
import spidev
import gpsd

# Ініціалізуємо GPS
gpsd.connect()

# Функція для зчитування даних з Arduino через SPI
def read_sensor_data():
    spi = spidev.SpiDev()
    spi.open(0, 0)
    spi.max_speed_hz = 1000000

    # Відправка запиту на отримання даних
    spi.xfer([0x01])

    # Зчитування даних з SPI
    received_data = spi.readbytes(3)

    # Розпакування отриманих даних
    speed = received_data[0]
    temperature = received_data[1]
    humidity = received_data[2]

    return speed, temperature, humidity

# Функція для зчитування додаткових даних з Arduino через SPI
```

```
def read_additional_sensor_data():
    spi = spidev.SpiDev()
    spi.open(0, 1)
    spi.max_speed_hz = 1000000

    # Відправка запиту на отримання даних
    spi.xfer([0x02])

    # Зчитування даних з SPI
    received_data = spi.readbytes(3)

    # Розпакування отриманих даних
    altitude = received_data[0]
    pressure = received_data[1]
    light_intensity = received_data[2]

    return altitude, pressure, light_intensity

# Оновлення даних датчиків
def update_sensor_data():
    # Зчитування даних з Arduino
    speed, temperature, humidity = read_sensor_data()
    altitude, pressure, light_intensity = read_additional_sensor_data()

    # Оновлення значень Label
    speed_value.set("{:.2f} m/s".format(speed))
    temperature_value.set("{:.2f} °C".format(temperature))
    humidity_value.set("{:.2f} %".format(humidity))
    altitude_value.set("{:.2f} m".format(altitude))
    pressure_value.set("{:.2f} hPa".format(pressure))
```

```
    light_intensity_value.set("{:.2f} lx".format(light_intensity))

    # Оновлення спідометра
    speed_meter.set(speed)

    root.after(1000, update_sensor_data)

# Оновлення даних GPS
def update_gps():
    gps_packet = gpsd.get_current()
    gps_data.set('{}, {}'.format(gps_packet.lat, gps_packet.lon))
    root.after(1000, update_gps)

# Створення вікна
root = tk.Tk()
root.title("Dashboard")
root.geometry("800x500")

# Додавання Label для відображення даних
speed_label = ttk.Label(root, text="Speed: ")
speed_label.grid(row=0, column=0, padx=10, pady=5, sticky="w")
speed_value = tk.StringVar()
speed_display = ttk.Label(root, textvariable=speed_value)
speed_display.grid(row=0, column=1, padx=10, pady=5, sticky="w")

temperature_label = ttk.Label(root, text="Temperature: ")
temperature_label.grid(row=1, column=0, padx=10, pady=5, sticky="w")
temperature_value = tk.StringVar()
temperature_display = ttk.Label(root, textvariable=temperature_value)
temperature_display.grid(row=1, column=1, padx=10, pady=5, sticky="w")
```

```
humidity_label = ttk.Label(root, text="Humidity: ")
humidity_label.grid(row=2, column=0, padx=10, pady=5, sticky="w")
humidity_value = tk.StringVar()
humidity_display = ttk.Label(root, textvariable=humidity_value)
humidity_display.grid(row=2, column=1, padx=10, pady=5, sticky="w")

altitude_label = ttk.Label(root, text="Altitude: ")
altitude_label.grid(row=3, column=0, padx=10, pady=5, sticky="w")
altitude_value = tk.StringVar()
altitude_display = ttk.Label(root, textvariable=altitude_value)
altitude_display.grid(row=3, column=1, padx=10, pady=5, sticky="w")

pressure_label = ttk.Label(root, text="Pressure: ")
pressure_label.grid(row=4, column=0, padx=10, pady=5, sticky="w")
pressure_value = tk.StringVar()
pressure_display = ttk.Label(root, textvariable=pressure_value)
pressure_display.grid(row=4, column=1, padx=10, pady=5, sticky="w")

light_intensity_label = ttk.Label(root, text="Light Intensity: ")
light_intensity_label.grid(row=5, column=0, padx=10, pady=5, sticky="w")
light_intensity_value = tk.StringVar()
light_intensity_display = ttk.Label(root, textvariable=light_intensity_value)
light_intensity_display.grid(row=5, column=1, padx=10, pady=5, sticky="w")

gps_label = ttk.Label(root, text="GPS: ")
gps_label.grid(row=6, column=0, padx=10, pady=5, sticky="w")
gps_data = tk.StringVar()
gps_display = ttk.Label(root, textvariable=gps_data)
gps_display.grid(row=6, column=1, padx=10, pady=5, sticky="w")
```

```
# Створення спідометра
speed_meter = ttk.Progressbar(root, orient="horizontal", length=400, mode="determinate")
speed_meter.grid(row=7, columnspan=2, padx=10, pady=5)

# Оновлення даних
update_sensor_data()
update_gps()

root.mainloop()
```

ДОДАТОК Г

ЛОГІКА РОБОТИ ПРИСТРОЯ

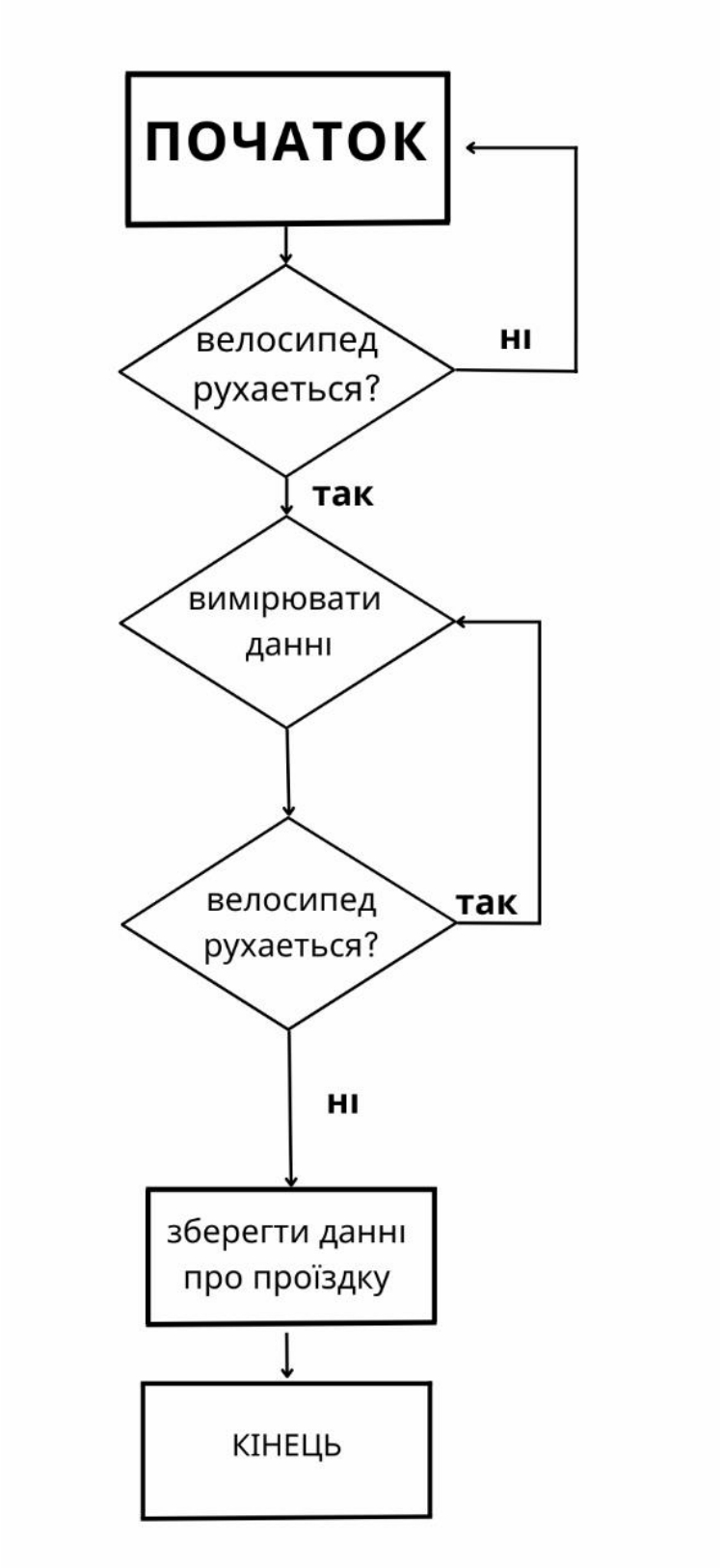

Рисунок Г 1 — Логіка роботи пристроя

ДОДАТОК Д

СХЕМА ПІДКЛЮЧЕННЯ

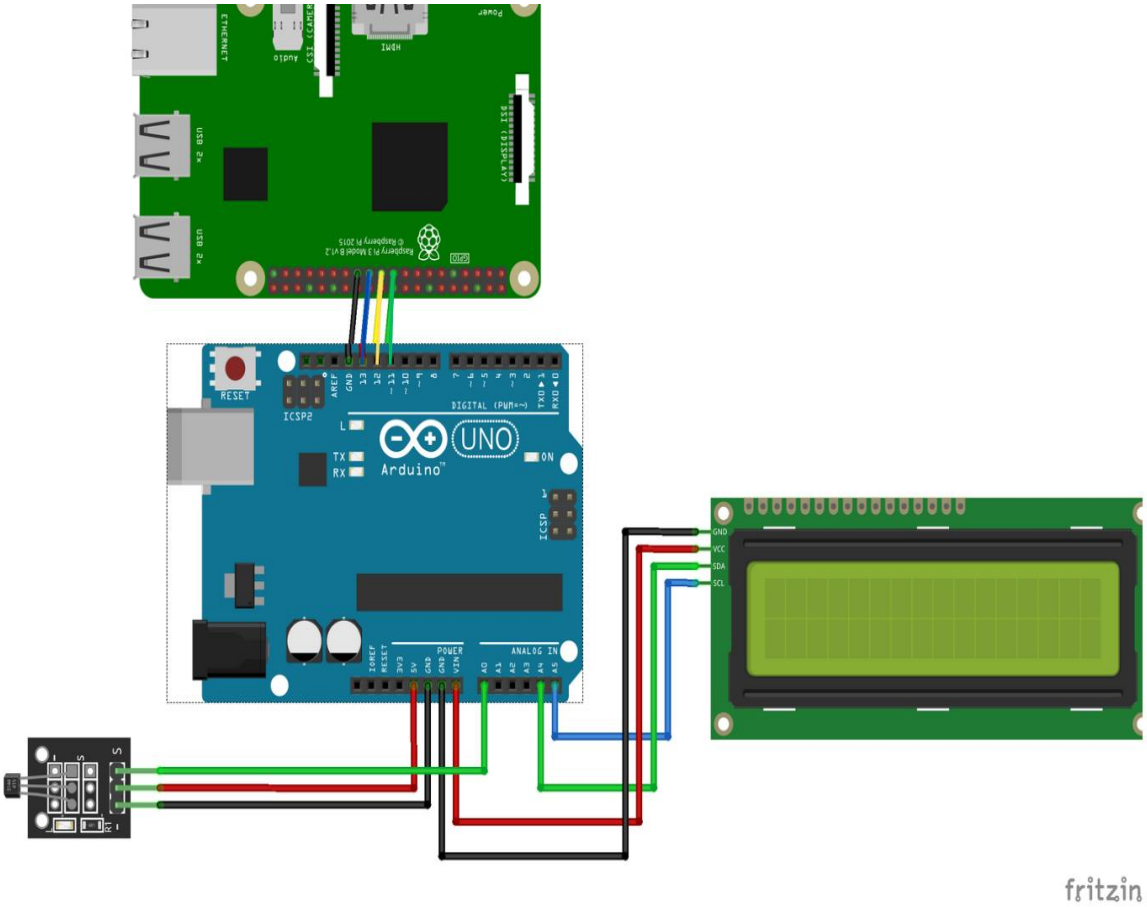

Рисунок Д 1 — Схема підключення

ДОДАТОК Е

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Мікропроцессорна система велокомп'ютера на Raspberry Pi

Тип роботи: Бакалаврський дипломний проект
(БДР,МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра,факультет)

Показники звіту подібності Unicheck

Оригінальність 99.1% Схожість 0.9%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Муращенко О.Г.
(підпис) (прізвище,ініціали)

Ознайомлен із повним звітом подібності,який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Бохенек Є.С
(підпис) (прізвище,ініціали)

Керівник роботи _____ Муращенко О.Г.
(підпис) (прізвище,ініціали)