Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого
навчального закладу)

Навчально-науковий інститут інформаційних технологій та робототехніки
(повна назва інституту)

Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

_____магістра_____
(рівень вищої освіти)
на тему

_____Розробка системи лазерної телеметрії для автоперегонів___
_____
_____

Виконав: студент __6_ курсу, групи 601 ТН__
спеціальності

_____122 Комп'ютерні науки _____
(шифр і назва спеціальності)
_____Конєв В. О._____
(прізвище та ініціали)
Керівник _____ Краснобаєв В. А._____
(прізвище та ініціали)
Рецензент ____Приходько А. С. _____
(прізвище та ініціали)

Полтава – 2025 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

спеціальність 122 «Комп'ютерні науки»

на тему

«Розробка системи лазерної телеметрії для автоперегонів»

Студента групи 601-ТН Конєва Владислава Олександровича

Керівник роботи
доктор технічних
наук, професор
Краснобаєв В. А.

Консультант
кандидат фізико-
математичних
наук,
Двірна О.А.

Завідувач кафедри
кандидат фізико-
математичних
наук,
Двірна О.А.

Полтава – 2025 року

РЕФЕРАТ

Дипломна робота містить 81 сторінки, 1 таблицю, 19 рисунків, список літератури з 27 найменувань, 8 додатків.

Розробка системи лазерної телеметрії для автоперегонів типу тайм атак

Об’єктом дослідження виступає технології заміру часі кола.

Предметом дослідження є розробка системи телеметрії мовами C++ та Java.

Мета дипломної роботи полягає у створенні системи лазерної телеметрії для автоперегонів.

Відповідно до мети наукового дослідження були поставлені та розв’язані наступні завдання:

— на основі проведеного порівняльного аналізу схожих програмних продуктів описати основні вимоги, функціональність та архітектуру майбутнього продукту;
— описати проектні рішення, інструменти створення, використані технології та підходи до розробки;
— розрахувати економічну складову проекту та розглянути питання з охорони праці в галузі;
— розробити систему телеметрії та виконати тестування.

За результатами дослідження сформульовані вимоги до системи телеметрії та розроблено дану систему.

Одержані результати можуть бути використані у майбутніх користувачів системи.

Рік виконання дипломної роботи 2025р.
Рік захисту роботи 2025р.

ABSTRACT

The thesis contains 81 pages, 1 table, 19 figures, a list of references of 27 titles, 8 appendices.

Development of a laser telemetry system for time attack type auto racing

The object of the research is the technology of lap timing.

The subject of the research is the development of a telemetry system in C++ and Java.

The purpose of the thesis is to create a laser telemetry system for auto racing.

In accordance with the purpose of the scientific research, the following tasks were set and solved:

— based on the comparative analysis of similar software products, describe the main requirements, functionality and architecture of the future product;
— describe the design solutions, creation tools, technologies used and development approaches;
— calculate the economic component of the project and consider issues of labor protection in the industry;
— develop a telemetry system and perform testing.

Based on the results of the study, the requirements for the telemetry system were formulated and this system was developed.

The results obtained can be used by future users of the system.

Year of completion of the thesis 2025p.
Year of defense of the thesis 2025p.

ЗМІСТ

ВСТУП

Місцем мого натхнення став технічно-спортивний комплекс «Лтава імені Володимира Черниша», розташований у місті Полтава. Цей об’єкт має багату історію та відіграє важливу роль у розвитку автоспорту в області та Україні загалом. Картодром «Лтава» побудовано в 1988 році, і з того часу він став одним із провідних центрів для проведення змагань з різних видів технічного спорту, таких як картинг, мотоциклетний спорт та автомобільні перегони. За десятиліття роботи цей комплекс набув значної популярності як серед професійних спортсменів, так і серед аматорів, які прагнуть розвиватися у різних сфера автоспорту.[1]

Основними напрямками діяльності комплексу є організація та проведення спортивних заходів різного рівня – від аматорських перегонів до чемпіонатів України з картингу та інших дисциплін. Крім того, комплекс активно працює над розвитком власної інфраструктури для забезпечення тренувань та підготовки спортсменів. Особливу увагу приділяють підготовці молодих спортсменів, залученню молоді до занять автоспортом та популяризації здорового способу життя через участь у спортивних заходах.

Під час роботи там я отримав можливість ознайомитися з роботою команди фахівців, відповідальних за організацію та технічне забезпечення спортивних заходів. Основна увага була зосереджена на вивченні процесів забезпечення точності вимірювань та збору даних під час змагань, зокрема підрахунку часу проходження кіл» на автомобільних змаганнях типу «тайм-атак». Оскільки точність вимірювань у спортивних змаганнях має вирішальне значення для визначення переможців та дотримання спортивної справедливості, перед командою технічного забезпечення стояло завдання забезпечити максимально точний і надійний інструментарій для вимірювання часу заїздів.

Одним із ключових аспектів моєї дипломної роботи стала розробка та впровадження системи лазерної телеметрії, яка дозволяє автоматично та з

високою точністю вимірювати час проходження кожного кола. Це завдання вимагало застосування сучасних технологій та знань у галузі електроніки, а також навичок програмування для створення ефективної системи збору й аналізу даних.

Проєкт, над яким я працював, був спрямований на розробку інноваційної системи телеметрії з використанням платформи Arduino, яка забезпечує легкість роботи з різними сенсорами та електронними компонентами. Arduino є універсальною платформою для побудови подібних проєктів завдяки своїй простоті та гнучкості. У рамках реалізації проєкту я займався підключенням та налаштуванням лазерних датчиків, програмуванням мікроконтролера, а також тестуванням та оптимізацією роботи системи в реальних умовах.

Розроблена система повинна була мати здатність точно фіксувати моменти перетину фінішної лінії кожним автомобілем та передавати ці дані для подальшої обробки й аналізу. Це завдання вимагало знань у таких галузях, як програмування мовами C++ та Java, що дозволило мені створити ефективні алгоритми для обробки отриманих даних та інтеграції системи з іншими компонентами комплексу.

Однією з основних переваг системи лазерної телеметрії є можливість безконтактного вимірювання часу та автоматичної обробки даних, що знижує ймовірність людських помилок та підвищує точність результатів. Окрім цього, використання Arduino дозволяє легко модифікувати систему та адаптувати її для інших видів спортивних змагань.

У ході роботи над проєктом я також ознайомився з системами зберігання та обробки даних, що використовуються в технічному спорті. Це допомогло мені розширити свої знання у сфері систем збору даних та аналітики, що є важливими аспектами сучасної спортивної інфраструктури.

Таким чином, робота в технічно-спортивному комплексі «Лтава» стала для мене цінним досвідом, який дозволив отримати практичні навички роботи з електронними компонентами, програмування, аналізу даних та інтеграції систем. Я зміг застосувати знання, отримані під час навчання, у реальних

умовах, а також розширити власні знання та навички у написанні коду мовами C++ та Java.

Цей досвід виявився надзвичайно корисним для моєї професійної підготовки, оскільки дозволив мені зрозуміти специфіку роботи у спортивній інженерії, а також навчитися працювати в команді над комплексними завданнями, що потребують координації зусиль багатьох спеціалістів. Робота також дала мені можливість ознайомитися з новими інструментами та технологіями, що відіграють важливу роль у проведені спортивних заходів.

Загалом, це дало мені можливість не лише закріпити теоретичні знання та отримати практичні навички, а й зрозуміти, як важливо використовувати сучасні технології у спорті. Завдяки здобутому досвіду я отримав чітке уявлення про роль технологій у забезпеченні чесності та прозорості змагань, що має вирішальне значення для розвитку сучасного спорту.

РОЗДІЛ 1

Опис предметної області

1.1 Поняття телеметрії та її види

Телеметрія – це спеціально розроблена система для перегонів для заміру часу проходження дистанції. Зараз найбільш поширені три основні типи телеметрії – лазерна, GPS-телеметрія, RFID-системи, кожна з яких має свої переваги та обмеження.[2]

Лазерні системи вимірювання часу працюють за принципом перетину автомобілем лазерного променя на контрольній точці. Такі системи зазвичай встановлюються на старті, фініші та в ключових секторах траси. Їх основна перевага – висока точність вимірювання (до 0,001 секунди) та миттєва фіксація результатів. Однак лазерні системи вимагають ретельної калібровки та коректного розміщення обладнання, щоб уникнути похибок. Наприклад, навіть невеликі зміщення або перешкоди на трасі можуть вплинути на точність вимірювання. Тим не менш, лазерна телеметрія є основною технологією для фінішної фіксації в Time Attack, де різниця між результатами може бути мінімальною. Варто зазначити, що під час використання лазерної телеметрії обгони на трасі заборонені, щоб уникнути конфліктів у вимірюванні часу.

GPS-телеметрія дозволяє відслідковувати місце розташування автомобіля на трасі та визначати проходження певних зон. Вона забезпечує ширший функціонал порівняно з лазерними системами, оскільки дозволяє аналізувати траєкторію руху автомобіля та взаємодію між учасниками перегонів. Проте точність GPS може коливатися, особливо на закритих ділянках або під час сильних перешкод сигналу. Такі системи переважно використовуються для контролю позицій автомобілів, а не для фінішної фіксації часу. Їх переваги полягають у легкості використання та можливості пілотам обганяти суперників. Недоліки включають високу вартість обладнання та ризики, пов’язані з перебоями у доступі до супутників,

наприклад, під час повітряної тривоги або роботи засобів радіоелектронної боротьби (РЕБ).

RFID-системи базуються на використанні міток, прикріплених до автомобілів, та зчитувачів, встановлених на трасі. Ці системи дозволяють реєструвати проходження контрольних точок, проте їх точність обмежується часом відгуку. Для змагань, де важливі мілісекунди, це може стати значним обмеженням. RFID має свої переваги, серед яких – легкість у використанні та можливість пілотам обганяти суперників. Однак є й недоліки, такі як висока вартість обладнання та датчиків, невелика дистанція спрацювання та ризик похибок у часі. Наприклад, при встановленні антени під дорожнім покриттям система може не зафіксувати результат, якщо автомобіль перебуває у стрибку, що робить RFID менш придатним для ралі або кросу.

Таким чином, кожен тип телеметрії має свої особливості, які слід враховувати залежно від специфіки перегонів та технічних вимог.

Сучасний стан розвитку техніки і технологій зумовлює швидкий прогрес у всіх сферах людської діяльності, що, у свою чергу, сприяє створенню нових засобів та систем. Це вимагає від науковців та фахівців постійного удосконалення знань та навичок, що успішно задовольняти сучасні потреби в певних галузях.

Однією з найбільш перспективних областей розвитку техніки стала обрана тема дослідження, яка пов’язана з створенням інноваційних рішень, спрямованих на підвищення ефективності та якості виробничих процесів.

1.2 Постановка завдання

Лазерна телеметрія для перегонів типу Time Attack – це система точного вимірювання часу, що дозволяє автоматично фіксувати проходження автомобіля через контрольні точки на трасі. Дана технологія базується на використанні лазерних сенсорів, які здатні реагувати на перетин променя транспортним засобом, забезпечуючи максимальну точність вимірювань.

У змаганнях формату Time Attack головною метою є визначення найшвидшого часу проходження траси окремими учасниками. Враховуючи високу конкуренцію, де навіть тисячні частки секунди можуть стати вирішальними, система лазерної телеметрії повинна забезпечувати високу швидкість та точність фіксації. Це зумовлює необхідність розробки надійної телеметричної системи, яка коректно працює в умовах різних погодних факторів і на високих швидкостях.

В рамках виконання дипломної роботи необхідно розробити лазерну телеметричну систему для змагань Time Attack, яка буде відрізнятися від існуючих рішень високою точністю, стійкістю до зовнішніх перешкод та інтеграцією з іншими елементами інфраструктури змагань (програмне забезпечення для обробки даних, панелі результатів та протоколи).

Система повинна виконувати такі функції:

— точне вимірювання часу з мінімальною похибкою (до 0,001 с);
— легка інтеграція з іншими системами змагань (лазерні датчики Omron E3Z-T86-L/D).

Основною особливістю даної розробки є фокус на автоспортивних змаганнях формату Time Attack, де пріоритетом є точність обліку часу для визначення переможця. Крім того, система повинна бути адаптована для різних трас та умов, що дозволить використовувати її як у професійних, так і аматорських перегонах.

1.3 Історія розвитку телеметрії в автоспорті

Телеметрія в автоспорті має довгу історію, яка почалася з простих механічних систем вимірювання часу. У 1950-х роках використовувалися механічні хронометри, які фіксували час проходження автомобіля через контрольні точки. З розвитком електроніки в 1970-х роках з'явилися перші електронні системи телеметрії, які дозволяли автоматизувати процес вимірювання часу.

У 1980-х роках почали використовуватися перші GPS-системи, які дозволяли відстежувати позицію автомобіля на трасі. Це стало значним кроком вперед, оскільки дозволило аналізувати траєкторію руху автомобіля та взаємодію між учасниками змагань. У 1990-х роках з'явилися RFID-системи, які дозволяли реєструвати проходження контрольних точок за допомогою міток, прикріплених до автомобілів.

Сьогодні телеметрія в автоспорті досягла високого рівня розвитку. Використовуються сучасні лазерні системи, які забезпечують точність вимірювання до 0,001 секунди. Крім того, телеметрія інтегрується з іншими системами змагань, такими як програмне забезпечення для обробки даних, панелі результатів та протоколи.[3]

1.4 Сучасні тенденції у використанні телеметрії в автоспорті

Сучасний автоспорт вимагає високої точності та швидкості вимірювання часу, що зумовлює постійний розвиток телеметричних систем. Однією з основних тенденцій є інтеграція телеметрії з іншими системами змагань, такими як програмне забезпечення для обробки даних, панелі результатів та протоколи. Це дозволяє автоматизувати процес вимірювання часу та забезпечити оперативне оновлення результатів.

Іншою тенденцією є використання сучасних технологій, таких як штучний інтелект та машинне навчання, для аналізу даних, зібраних під час змагань. Це дозволяє отримати більш глибоке розуміння поведінки автомобіля на трасі та виявити потенційні проблеми, які можуть вплинути на результати змагань.

Крім того, сучасні телеметричні системи стають більш доступними для аматорських змагань. Раніше такі системи використовувалися переважно на професійних заходах, але зараз вони стають доступними для менших змагань, що дозволяє підвищити рівень змагань та забезпечити більш об'єктивні результати.

1.5 Вплив телеметрії на розвиток автоспорту

Телеметрія має значний вплив на розвиток автоспорту. Вона дозволяє забезпечити точність та об'єктивність результатів змагань, що є важливим аспектом для учасників та організаторів. Крім того, телеметрія дозволяє аналізувати поведінку автомобіля на трасі та виявляти потенційні проблеми, які можуть вплинути на результати змагань.

Телеметрія також сприяє популяризації автоспорту. Завдяки можливості отримувати точні результати та аналізувати дані, змагання стають більш цікавими для глядачів. Крім того, телеметрія дозволяє залучати до змагань більше учасників, оскільки забезпечує об'єктивність результатів та знижує ризик суперечок між учасниками.

У майбутньому телеметрія продовжуватиме розвиватися, що дозволить ще більше підвищити точність та об'єктивність результатів змагань. Крім того, інтеграція телеметрії з іншими технологіями, такими як штучний інтелект та машинне навчання, дозволить отримати більш глибоке розуміння поведінки автомобіля на трасі та виявити потенційні проблеми, які можуть вплинути на результати змагань.

РОЗДІЛ 2

Огляд та порівняльна харатеристика існуючих подібних систем

2.1. Моніторинг (конкурентоспроможність)

Моніторинг у сфері лазерної телеметрії передбачає проведення серії вимірювань та аналізу результатів для оцінки продуктивності й виявлення тенденцій. У контексті розробки лазерної телеметричної системи для перегонів Time Attack моніторинг конкурентних рішень дозволяє визначити переваги та недоліки існуючих технологій, виявити найкращі практики та уникнути поширених помилок.

Конкурентний моніторинг охоплює аналіз вже існуючих систем телеметрії, що використовуються в автоспорті, зокрема тих, які застосовують лазерні, GPS, RFID або інші сенсори для вимірювання часу. Мета моніторингу — знайти оптимальні рішення та визначити шляхи покращення власної розробки для досягнення конкурентних переваг.

Критерії оцінки телеметричних систем

Для аналізу конкурентних систем використовуються такі критерії:

— точність вимірювання часу: важливо, щоб система могла фіксувати результати з точністю до тисячних часток секунди, що критично для Time Attack. Недостатня точність може призвести до спірних ситуацій при визначенні переможця;

— інтеграція з іншими системами. Телеметричні системи повинні легко інтегруватися з програмним забезпеченням для протоколів змагань, інформаційними табло та мобільними додатками для глядачів і організаторів;

— легкість установки та налаштування. Для організаторів важлива простота монтажу системи на трасі та можливість швидкого налаштування без складних процедур калібрування. Це забезпечує економію часу перед початком змагань;

— вартість та економічна доцільність. оцінюється співвідношення ціни до функціональності. Надто дорогі системи можуть бути недоцільними для аматорських змагань, тоді як дешевші рішення можуть мати значні обмеження у точності.

2.2. Система телеметрії Mylaps

Одним із лідерів ринку телеметричних систем для автоспорту є компанія MyLaps, яка пропонує комплексні рішення для змагань різних форматів, включно з Time Attack. Їхні системи відомі своєю надійністю, інтеграцією з іншими сервісами та точністю. Аналіз MyLaps дозволяє оцінити, як виглядає сучасна телеметрія, що відповідає найвищим стандартам, та визначити аспекти, які можуть бути вдосконалені в нашій розробці.[4]

Ключові характеристики системи MyLaps:

— MyLaps використовує комбінацію RFID-транспондерів(Рисунок 2.1) та індукційних петель, що вбудовуються в поверхню траси. Кожен автомобіль оснащується RFID-транспондером, який активується під час перетину індукційної петлі, що дозволяє точно фіксувати час проходження;

— система забезпечує точність на рівні 0,001 секунди, що відповідає вимогам автоспортивних змагань, хоча в деяких ситуаціях чутливість RFID може поступатися лазерним системам.

Рисунок 2.1 – Транспондер телеметрії MyLaps

Надійність у різних умовах:

— система MyLaps розроблена для стабільної роботи за будь-яких погодних умов. Вона не чутлива до дощу чи зміни освітлення, оскільки технологія RFID не залежить від оптичного сигналу. Це може бути перевагою в порівнянні з лазерними рішеннями, які потребують точного налаштування.

Інтеграція з програмним забезпеченням:

— MyLaps пропонує комплексну платформу для обробки та аналізу даних у реальному часі. Результати автоматично передаються на сервер і можуть бути опубліковані на онлайн-табло, інтегровані в мобільні додатки для глядачів та організаторів;

— система також підтримує трансляцію результатів у прямому ефірі, що робить її популярною серед організаторів великих заходів.

Простота установки та налаштування:

— MyLaps пропонує відносно простий монтаж індукційних петель, які можуть бути використані на постійних або тимчасових трасах. Система

легко налаштовується та дозволяє швидко розпочати роботу після встановлення. Це важливо для організаторів, яким потрібна мінімальна підготовка перед початком змагань.

Вартість та рентабельність:

— системи MyLaps є доволі дорогими, що може обмежувати їх використання у менших аматорських змаганнях. Проте на професійних заходах їхня надійність та функціональність часто виправдовують високі витрати.

— Переваги та недоліки MyLaps для Time Attack

Переваги:

— висока точність фіксації часу (0,001 с);

— стійкість до погодних умов та зовнішніх перешкод;

— широкі можливості інтеграції з мобільними додатками та онлайн-платформами;

— підтримка обробки даних у реальному часі та трансляції результатів.

Недоліки:

— вища вартість у порівнянні з іншими рішеннями;

— наявна затримка в залежності від антени, тому є погрішність в часі;

— якщо закріпити транспортер високо, то час не буде відсікатися;

— RFID-технологія може бути менш чутливою на коротких відстанях порівняно з лазерними системами;

— індукційні петлі потребують монтажу, що може ускладнити використання на деяких тимчасових трасах.

2.3. Система телеметрії Tag Heuer Timing

Компанія Tag Heuer Timing відома своїми високоточними системами вимірювання часу для різних видів автоспорту, включаючи перегони формату Time Attack. Їхні рішення орієнтовані на професійні змагання, де особлива

увага приділяється точності фіксації та надійності під час проведення заходів світового рівня.[5]

Ключові характеристики системи Tag Heuer Timing:

— системи Tag Heuer використовують лазерні промені та фотоелементи(Рисунок 2.2), що розміщуються на ключових точках траси. Промені створюють невидимий бар'єр, який розривається при проходженні автомобіля, фіксуючи точний час із похибкою до 0,001 секунди;

— лазерні бар'єри забезпечують миттєву реакцію, що робить їх оптимальними для індивідуальних заїздів, таких як Time Attack.

Рисунок 2.2 – набір телеметрії Tag Heuer Timing

Надійність та стабільність:

— система працює ефективно незалежно від погодних умов і зовнішніх факторів. Вона оснащена додатковими резервними елементами, що

дозволяє уникнути втрати даних у разі збоїв. Це важливо для організаторів, які не можуть допустити жодних помилок у процесі вимірювань.

Інтеграція з інформаційними системами:

— Tag Heuer Timing пропонує інтеграцію з табло, мобільними додатками та програмами обробки даних у режимі реального часу. Це дозволяє організаторам швидко оновлювати результати та передавати їх глядачам і учасникам.

Простота в установці та налаштуванні:

— лазерні та фотоелементні системи легко встановлюються на трасі й не вимагають складного налаштування. Це робить їх привабливими для використання на тимчасових трасах та під час короткострокових заходів.

Вартість та рентабельність:

— система має високу вартість через використання преміум-компонентів, але це виправдано для професійних перегонів, де пріоритетом є максимальна точність та надійність.

Переваги та недоліки Tag Heuer Timing для Time Attack

Переваги:

— висока точність (0,001 секунди) завдяки лазерним променям;
— інтеграція з сучасними системами обробки даних і табло;
— надійна робота за будь-яких погодних умов;
— простота налаштування для різних видів змагань.

Недоліки:

— висока вартість, що може обмежити використання на аматорських заходах;
— необхідність забезпечення точної орієнтації лазерних елементів на трасі.

2.4. Система телеметрії RaceResult

Система RaceResult здобула популярність завдяки універсальності та можливості використання у різних видах перегонів, зокрема марафонах, велозмаганнях та автоспорті. Її рішення орієнтовані на простоту впровадження та можливість швидкої обробки даних для заходів будь-якого масштабу, включно з Time Attack.[6]

Ключові характеристики системи RaceResult:

— RaceResult використовує активні RFID-транспондери(Рисунок 2.3), які встановлюються на автомобілях і синхронізуються з фіксуючими пристроями на трасі. Транспондери активуються при перетині контрольних точок, передаючи сигнал для точного вимірювання часу;

— система забезпечує точність вимірювання на рівні 0,01 секунди, що підходить для багатьох видів перегонів, хоча може бути недостатньо для Time Attack, де важлива кожна тисячна частка секунди.

Рисунок 2.3 – Набір телеметрії RaceResult

Надійність та гнучкість у використанні:

— RaceResult працює стабільно в різних умовах і не потребує складної інфраструктури. Це дозволяє використовувати систему як на постійних трасах, так і під час тимчасових заходів, що організовуються на обмежений час.

Інтеграція з хмарними сервісами:

— RaceResult підтримує передачу даних у хмарні сховища та можливість онлайн-трансляцій результатів у режимі реального часу. Учасники й організатори можуть відстежувати прогрес прямо через веб-інтерфейс або мобільні додатки.

Простота налаштування та використання:

— система має зручний інтерфейс для налаштування та потребує мінімальних зусиль для впровадження. Її легко адаптувати під різні типи заходів, що робить її популярною серед організаторів аматорських перегонів.

Вартість та доступність:

— RaceResult є більш доступним рішенням у порівнянні з преміальними системами, такими як MyLaps або Tag Heuer. Це робить її привабливою для організаторів менших заходів, де потрібно забезпечити базову точність без значних витрат.

Переваги та недоліки RaceResult для Time Attack

Переваги:

— легка установка та налаштування;
— стабільна робота за різних умов;
— інтеграція з хмарними сервісами та онлайн-трансляціями;
— доступна вартість для аматорських заходів.

Недоліки:

— точність вимірювання (0,01 с) може бути недостатньою для професійних змагань Time Attack;

— потребує встановлення RFID-транспондерів на автомобілі, що може ускладнити підготовку.

2.5. Система телеметрії ChronoTrack

ChronoTrack спеціалізується на високоточних системах вимірювання часу, широко застосовуючи свої рішення у вело- та автоспорті. Основна перевага ChronoTrack – гнучке використання RFID-технологій для вимірювань у режимі реального часу.[7]

Ключові характеристики системи ChronoTrack:

— ChronoTrack використовує пасивні RFID-чіпи(Рисунок 2.4), що розміщуються на автомобілях. Дані зчитуються при перетині контрольних точок на трасі. Ця технологія забезпечує точність вимірювань до 0,02 секунди, чого може бути достатньо для аматорських перегонів, але недостатньо для професійних Time Attack.

Рисунок 2.4 – Набір системи телеметрії ChronoTrack

Швидке розгортання системи:

— RFID-рішення ChronoTrack потребує мінімальної інфраструктури та швидко налаштовується. Система ідеально підходить для тимчасових змагань або заходів, де важлива швидкість встановлення обладнання.

Обробка великих обсягів даних:

— ChronoTrack ефективно працює з великим потоком учасників, що важливо для масових змагань. Для Time Attack це може бути менш актуально, але для організаторів це гарантує стабільну роботу за високого навантаження.

Інтеграція з мобільними додатками:

— результати змагань можна транслювати в мобільні додатки та онлайн-платформи, що дозволяє глядачам та учасникам відстежувати прогрес у реальному часі.

Вартість та гнучкість:

— ChronoTrack має середній ціновий рівень серед телеметричних систем. Її можна використовувати як на аматорських, так і на професійних заходах завдяки універсальності технологій.

Переваги та недоліки ChronoTrack для Time Attack

Переваги:

— простота налаштування та швидке розгортання;
— можливість обробки великих обсягів даних;
— інтеграція з мобільними додатками для онлайн-трансляцій;
— гнучка вартість для різних рівнів змагань.

Недоліки:

— точність 0,02 секунди може бути недостатньою для професійних Time Attack;
— RFID-чіпи потребують ручної установки на кожен автомобіль, що ускладнює підготовку.

2.6. Система телеметрії Alge Timing

Alge Timing – відома система вимірювання часу, що широко застосовується у різних спортивних заходах, включаючи біг, велоспорт і автоспорт. Основна перевага Alge Timing – це мультисенсорні технології, які поєднують лазерні бар'єри та інфрачервоні сенсори для підвищення точності вимірювань, які зображені на рисунку 2.5.[8]

Рисунок 2.5 – Набір телеметрії Alge Timing

Ключові характеристики системи Alge Timing:

— система Alge Timing використовує поєднання інфрачервоних та лазерних датчиків, що дозволяє досягати точності до 0,001 секунди. Лазерні бар'єри встановлюються на старті, фініші та ключових точках траси, що забезпечує повний контроль над часом проходження кожної секції.

Гнучке налаштування:

— Alge Timing дозволяє налаштовувати систему під різні умови змагань. Це робить її привабливою для тимчасових та постійних трас, де важлива максимальна точність.

Резервування та захист даних:

— система оснащена функцією резервного збереження даних, що забезпечує надійність під час непередбачуваних збоїв. Це критично важливо для змагань, де будь-яка втрата даних може призвести до спірних результатів.

Інтеграція з програмним забезпеченням:

— Alge Timing пропонує програмне забезпечення для аналітики та управління результатами. Організатори можуть отримувати звіти в реальному часі та аналізувати результати кожного учасника.

Вартість:

— висока точність та надійність роблять Alge Timing преміальним продуктом, але це також означає високі витрати. Вона зазвичай використовується для професійних змагань, де важливі максимальна точність і стабільність.

Переваги та недоліки Alge Timing для Time Attack

Переваги:

— точність вимірювань до 0,001 секунди;
— можливість використання інфрачервоних та лазерних датчиків;
— надійна система резервування та захисту даних;
— програмне забезпечення для аналітики та управління результатами.

Недоліки:

— висока вартість;
— складність налаштування та впровадження на тимчасових трасах.

Дивлячись на всі переваги і недоліки конккурентних рішень, було зрозуміло що дані системи телеметрії не дуже підходять під основні вимоги

організаторів серій з TimeAttack та регламентам. Було прийняте рішення про розробку системи лазерної телемерії, врахувавши всі недоліки подібних систем.

В ході вивчення конкуруючих продуктів було визначено головні проблеми:

— ціна системи;

— точність вимірювання;

— складність підготовки системи до роботи.

Порівняльну характеристику подібних систем наведено в додатку А.

В таблиці відповідність програмного продукту заданому критерію відмічаємо «+», а відсутність критерію «–».

РОЗДІЛ 3

Виявлення вимог до системи. опис її функціональності

3.1. Основні вимоги до продукту

У процесі розробки системи телеметрії важливо ретельно визначити функціональні та нефункціональні вимоги, щоб забезпечити її ефективну роботу та відповідність потребам користувачів. Функціональні вимоги визначають, які дії повинна виконувати система, а нефункціональні – описують її якість, надійність та зручність використання.

Система лазерної телеметрії призначена для точного вимірювання результатів у спортивних заходах або змаганнях типу Time Attack. Вона складається з лазерних датчиків Omron E3Z-T86-L/D для фіксації проходження контрольних точок, платформи Arduino для збору та обробки даних, а також Java-додатку на ПК для візуалізації результатів і керування системою.

ФУНКЦІОНАЛЬНІ ВИМОГИ ЛАЗЕРНОЇ ТЕЛЕМЕТРІЇ

Функціональні вимоги визначають основні операції та завдання, які система повинна виконувати для досягнення своїх цілей.

— розпізнавання проходження контрольних точок: лазерні датчики Omron повинні точно фіксувати момент проходження об'єкта через визначені контрольні точки;
— обробка даних за допомогою Arduino: система на базі Arduino повинна отримувати сигнали від лазерних датчиків та обчислювати проміжний і фінальний час проходження;
— передача результатів у режимі реального часу: Arduino передає дані в комп'ютерний додаток через USB або бездротове з'єднання для їх обробки та відображення;

— Java-додаток для ПК: відображати результати змагань у реальному часі. Вмикати та вимикати ручний режим, який дозволяє оператору вручну додавати або коригувати результати. Вести лог всіх зафіксованих подій для подальшого аналізу;
— збереження результатів: Java-додаток повинен мати можливість зберігати результати у базі даних або у вигляді файлів для подальшого аналізу;
— налаштування параметрів: додаток дозволяє налаштувати параметри датчиків, інтервали запису та часові пороги;
— аварійне резервування: у разі збою програма має зберігати дані локально та сигналізувати про помилку оператору.

НЕФУНКЦІОНАЛЬНІ ВИМОГИ ЛАЗЕРНОЇ ТЕЛЕМЕТРІЇ

Нефункціональні вимоги описують якість роботи системи та умови її експлуатації:

— точність вимірювань: лазерні датчики повинні забезпечувати точність фіксації подій до 1 мілісекунди;
— продуктивність: Arduino та комп'ютерний додаток повинні обробляти дані без затримок, щоб результати відображалися миттєво;
— стабільність та надійність: система повинна працювати стабільно протягом тривалих сесій без переривань;
— зручність використання: інтерфейс Java-додатку має бути інтуїтивно зрозумілим, з можливістю налаштування під потреби оператора;
— сумісність: програма повинна підтримувати різні версії Windows, Linux та macOS. Arduino має бути сумісним із широким спектром периферійних пристроїв;
— швидкість налаштування: система повинна бути простою у розгортанні та налаштуванні. Усі компоненти мають швидко синхронізуватися після запуску;
— масштабованість: система повинна легко масштабуватися для роботи з більшою кількістю датчиків та збільшеним обсягом даних;

— захист від зовнішніх факторів: датчики та обладнання мають бути стійкими до пилу, вологи та інших несприятливих умов середовища;
— безпека: дані, що зберігаються, повинні бути захищені паролями та шифруванням, щоб уникнути несанкціонованого доступу.

3.2. Проектування системи лазерної телеметрії

Розробка системи лазерної телеметрії включає кілька важливих етапів, кожен з яких має вирішальне значення для досягнення цілей проєкту.

Визначення мети системи:

— основна мета – забезпечення точного та швидкого вимірювання часу проходження об'єктів через контрольні точки під час змагань.

Вибір компонентів та обладнання:

— лазерні датчики Omron E3Z-T86-L/D для точного розпізнавання подій;
— Arduino як центральний елемент обробки даних;
— Java-додаток для виведення результатів та керування системою.

Розробка програмного забезпечення:

— програма на Arduino для отримання та обробки сигналів;
— розробка Java-додатку з графічним інтерфейсом для візуалізації та керування.

Тестування та налаштування:

— проведення польових випробувань для перевірки точності та стабільності системи;
— перевірка коректності обробки даних у реальних умовах.

Розгортання системи:

— підключення та налаштування всіх компонентів на місці змагань;
— перевірка готовності системи перед стартом події.

Взаємодія з користувачами:

— навчання оператора використанню системи та інтерфейсу додатку;
— підтримка під час експлуатації та оперативне усунення можливих збоїв.

Періодичні оновлення:

— внесення покращень та оновлень програмного забезпечення на основі відгуків користувачів;
— розширення функціональності системи за потреби.

3.3. Аналіз та візуалізація вимог на діаграмі прецедентів

Діаграма прецедентів – це один з ключових інструментів моделювання в системному аналізі та проектуванні програмного забезпечення. Вона дозволяє візуалізувати функціональність системи з точки зору користувачів та їх дій.[9]

Діаграма прецедентів складається з акторів, прецедентів та зв'язків між ними.

Актори – це ролі або конкретні люди, які взаємодіють з системою. Актори можуть бути зовнішніми користувачами, внутрішніми системами або іншими програмними компонентами. Кожен актор представляється в діаграмі використанням піктограми людини або об'єкта.

Прецеденти – це дії або функції, які система виконує для користувачів. Прецеденти описують те, що робить система, а не як вона це робить. Кожен прецедент представляється в діаграмі використанням овальної форми з назвою.[10]

Зв'язки – це лінії, які показують взаємодію між акторами та прецедентами. В діаграмі використовуються три типи зв'язків:

— зв'язок між актором та прецедентом, що показує, як актор взаємодіє з прецедентом;
— зв'язок між двома прецедентами, що показує, як один прецедент використовує інший для виконання своїх завдань;
— загальний зв'язок, що показує спільність між прецедентами або акторами.

Діаграма прецедентів допомагає розробникам зрозуміти, як користувачі будуть взаємодіяти з системою та які функції повинна виконувати система для задоволення потреб користувачів.

ПРЕЦЕДЕНТ: ЗАПУСК СЕСІЇ

Актори: Оператор.

Передумова: Додаток Arduino підключено до серійного монітора.

Післяумова: Система починає відстеження переривання лазерного променя.

Сценарій:

1. Оператор вводить команду для запуску сесії.
2. Система ініціалізує облік кола.
3. Починається відстеження лазерного променя.

ПРЕЦЕДЕНТ: РЕЄСТРАЦІЯ ЗАВЕРШЕНОГО КОЛА

Актори: Лазерний сенсор, додаток Arduino.

Передумова: Сесія активна.

Післяумова: Кількість завершених кіл оновлюється.

Сценарій:

1. Лазерний сенсор фіксує переривання.
2. Arduino реєструє завершення кола для відповідного автомобіля.
3. Оновлюється лог у пам’яті.

ПРЕЦЕДЕНТ: ЗУПИНКА АВТОМОБІЛЯ

Актори: Оператор.

Передумова: Сесія активна.

Післяумова: Автомобіль зупиняється.

Сценарій:

1. Оператор вводить команду зупинки для конкретного автомобіля.
2. Arduino виконує команду зупинки.

ПРЕЦЕДЕНТ: ЗАВЕРШЕННЯ СЕСІЇ

Актори: Оператор.

Передумова: Сесія активна.

Післяумова: Система припиняє відстеження кола.

Сценарій:

1. Оператор вводить команду завершення сесії.
2. Система зберігає логи.
3. Всі активні процеси припиняються.

ПРЕЦЕДЕНТ: ВИВЕДЕННЯ ЛОГІВ

Актори: Оператор, EEPROM.

Передумова: Сесія завершена

Післяумова: Логи виведені на екран серійного монітора.

Сценарій:

1. Оператор вводить команду виводу логів.
2. Система зчитує логи з EEPROM.
3. Логи виводяться у серійний монітор.

ПРЕЦЕДЕНТ: ВИДАЛЕННЯ ЛОГІВ

Актори: Оператор, EEPROM.

Передумова: Логи існують у пам’яті.

Післяумова: Логи видалено з EEPROM.

Сценарій:

1. Оператор вводить команду видалення логів.
2. Система очищає відповідну область EEPROM.

ПРЕЦЕДЕНТ: ВИЯВЛЕННЯ ПЕРЕРИВАННЯ ЛАЗЕРНОГО ПРОМЕНЯ

Актори: Лазерний сенсор.

Передумова: Сесія активна.

Післяумова: Система фіксує переривання.

Сценарій:

1. Лазерний сенсор фіксує зміну напруги.
2. Система розпізнає переривання як завершення кола.

Діаграму прецендентів можна переглянути у додатку Б.

3.4. Опис прототипу інтерфейсу

Прототип інтерфейсу – це візуальна модель або прототип майбутнього програмного інтерфейсу, який дозволяє користувачам та розробникам перевірити та оцінити його функціональність, ефективність та зручність використання.[11]

Прототип інтерфейсу може бути статичним зображенням екрану або динамічною інтерактивною моделлю, яка відображає реакції на дії користувача. Прототип дозволяє протестувати функціональність та інтерфейсні рішення на ранніх етапах розробки, що дозволяє зекономити час та кошти на подальші коригування.

Також, прототип інтерфейсу дозволяє залучати до розробки користувачів та отримувати їхні відгуки та пропозиції щодо поліпшення інтерфейсу. Це може допомогти покращити користувальницький досвід та підвищити ефективність програмного продукту.

Вимоги до інтерфейсу користувача:

— простота та зручність використання: інтерфейс має бути легким та зрозумілим для користувачів будь-якого рівня технічної підготовки;
— мінімалізація зусиль: користувач повинен здійснювати операції з мінімальною кількістю кроків та дій.

Головне вікно програми складається з верхньої панелі, центральної секції (розділеної на блоки), панелі для логів, та кнопок керування.

Верхня панель (Top Panel):

Призначення: Вибір COM-порту для підключення до пристрою.

Елементи:

— мітка (JLabel): "Select COM Port:" для пояснення функціоналу;
— випадаючий список (JComboBox): Перелік доступних COM-портів, автоматично зчитаних за допомогою SerialPort.getCommPorts();
— кнопка (JButton): "Connect" для підключення до вибраного порту.

Дія: Користувач обирає COM-порт і натискає "Connect", після чого програма намагається встановити з'єднання.

Панель статусу (Status Panel)

Призначення: Відображення поточного статусу сесії та активних автомобілів.

Елементи:

— мітка (JLabel): "Session: Inactive" – відображає статус сесії (активна/неактивна);
— мітка (JLabel): "Cars Remaining: 0" – показує кількість автомобілів, що залишились у сесії;
— текстова область (JTextArea): "Active Cars: None" – список активних автомобілів.

Дія: При зміні статусу сесії чи активності автомобілів ці елементи оновлюються.

Панель керування (Control Panel)

Призначення: Забезпечує основні функції керування сесією.

Елементи:

— поле вводу (JTextField): Для введення кількості автомобілів, які будуть брати участь у сесії;
— поле вводу (JTextField): Для введення номера автомобіля, який треба видалити.

Кнопки (JButton):

— "Start Session" – запускає сесію з вказаною кількістю автомобілів;
— "Remove Car" – видаляє вказаний номер автомобіля із активної сесії;
— "End Session" – завершує поточну сесію.

Дія: Користувач взаємодіє з елементами для керування телеметрією.

Панель логів (Log Panel)

Призначення: Виведення логів роботи програми.

Елементи:

— Текстова область (JTextArea): Відображає всі повідомлення логів;

— скролінг (JScrollPane): Для перегляду великої кількості записів.

Дія: Всі дії та помилки зберігаються і відображаються в цьому вікні. Логи також записуються у файл.

3.5. Опис архітектури системи

Архітектура системи – це структура і організація компонентів, модулів та їх взаємодія в межах програмного або апаратного забезпечення. Вона визначає план або концепцію системи і надає основу для розробки та управління системою.[12]

Клас – це шаблон або опис, який визначає структуру та поведінку об'єктів в програмуванні. Він визначає атрибути (змінні) і методи (функції), які можуть бути використані об'єктами даного класу. Клас є основною одиницею організації коду в об'єктно-орієнтованому програмуванні і використовується для створення нових об'єктів з визначеними характеристиками і поведінкою. Об'єкти, що належать до одного класу, мають спільні властивості і можуть виконувати спільні дії, визначені в класі.[13]

Діаграма класів – це графічний засіб моделювання, який використовується для візуалізації структури класів, їх взаємозв'язків та основних елементів об'єктно-орієнтованого програмного коду. Вона надає зручний спосіб показати класи, їх атрибути (змінні) та методи (функції), а також взаємодію між ними. [14]

Діаграма класів може містити такі елементи:

— класи: представлені у вигляді прямокутників з назвою класу;

— атрибути: змінні, що належать класу, вказуються в прямокутнику під назвою класу;

— методи: функції, що належать класу, показуються в прямокутнику під атрибутами або окремо від них;

— взаємозв'язки: показуються за допомогою стрілок між класами, що вказують на залежність, агрегацію, асоціацію, спадкування та інші типи відношень між класами.

Клас Java-додатку:

Атрибути:

— serialPort: відповідає за підключення до порту Arduino;
— logArea, portList, sessionStatusLabel, carsRemainingLabel, activeCarsArea, carNumberField, carCountField: елементи GUI для відображення та взаємодії з користувачем;
— logFile: об'єкт для збереження логів;
— sessionActive: булевий індикатор, що показує статус сесії;
— activeCars: список активних автомобілів.

Методи:

— createLogFile(): створює файл для логування;
— connectToArduino(): здійснює підключення до вибраного COM-порту;
— startSession(), removeCar(), endSession(): керують сесією;
— updateStatus(): оновлює статус GUI відповідно до поточного стану;
— sendCommand(String command): надсилає команди через COM-порт;
— log(String message), saveLogToFile(String message): додають повідомлення до логів та зберігають їх у файл;
— main(String[] args): точка входу для запуску програми.

Зв'язки:

— TelemetryScreen залежить від класів бібліотек (SerialPort, JTextArea, JComboBox, JLabel, JTextField, File, List), оскільки використовує їх об'єкти.

Клас прошивки для Arduino:

Опис компонентів:

TelemetrySystem

Призначення: Основний клас, який реалізує функціонал підрахунку кіл автомобілів.

Поля:

— laserPin: Аналоговий пін для зчитування лазера;
— thresholdVoltage: Порогова напруга для визначення переривання лазера;
— maxCars: Максимальна кількість автомобілів у системі;
— eepromSize: Розмір пам’яті EEPROM;
— logEntrySize: Максимальна довжина запису логу;
— totalCars, currentCar: Кількість автомобілів та номер поточної машини;
— logCount: Кількість збережених логів;
— lastLapFinishTimes: Час завершення останнього кола кожної машини;
— lapCounts: Кількість завершених кіл для кожної машини;
— carStopped, carStarted: Стан кожної машини (зупинена/активна);
— laserInterrupted: Прапорець, чи був лазер перерваний;
— lastLapTime: Час останнього зафіксованого кола;
— debounceDelay: Мінімальна затримка між реєстраціями кіл.

Методи:

— setup() і loop(): Ініціалізація системи та основний цикл роботи;
— startNewSession(int num): Початок нової сесії для заданої кількості машин;
— logLap(): Реєстрація кола;
— stopCar(int num): Зупинка машини;
— endSession(): Завершення сесії;
— saveLogToEEPROM(String logEntry): Збереження записів у пам’яті;
— printLogs(): Виведення всіх логів;
— deleteLogs(): Очищення логів;
— getFormattedTime(unsigned long lapTimeMs): Форматування часу.

EEPROM

Призначення: Клас для роботи із збереженням даних у пам’яті.

Методи:

— put(int address, char[] data): Запис даних у вказану адресу EEPROM;

— get(int address, char[] buffer): Зчитування даних із вказаної адреси.

RTC_Millis

Призначення: Імітує функціонал реального часу (RTC).

Методи:

— begin(DateTime dt): Ініціалізує реальний час.

LaserSensor

Призначення: Представляє лазерний сенсор для виявлення перешкод (переривання лазера).

Поля:

— pin: Аналоговий пін, до якого підключений сенсор;

— thresholdVoltage: Порогова напруга для визначення переривання.

Методи:

— LaserSensor(int pin, float thresholdVoltage): Конструктор для створення сенсора;

— isInterrupted(): Визначає, чи перервано лазерний промінь;

— readVoltage(): Читає поточну напругу з сенсора.

Логіка зв'язків між класами:

— TelemetrySystem → EEPROM: зберігає дані у вбудованій пам’яті EEPROM через методи put і get;

— TelemetrySystem → RTC_Millis: використовує RTC_Millis для отримання точного часу, необхідного для фіксації часу завершення кіл;

— TelemetrySystem → LaserSensor: зчитує напругу з лазерного сенсора, щоб визначити, чи перервано лазерний промінь.

Діаграми класів можна переглянути у додатку Г.

РОЗДІЛ 4

Опис проектних рішень, інструментів та підходів до розробки

4.1. Мова програмування C++

C++ – це високопродуктивна мова програмування, яка поєднує можливості об'єктно-орієнтованого, процедурного та узагальненого програмування. Її створив Б'ярн Страуструп у 1983 році як розширення мови C, що дозволило додати об'єктно-орієнтовані можливості. C++ використовується для розробки операційних систем, ігрових рушіїв, графічного програмного забезпечення, фінансових додатків, драйверів та складних інженерних програм.[15]

Ця мова підтримує маніпуляції з пам'яттю, що робить її популярною серед розробників, які працюють з обмеженими ресурсами або потребують високої продуктивності. Оскільки C++ дає доступ до низькорівневих функцій, це дозволяє максимально ефективно використовувати апаратні ресурси, що робить її популярною у розробці вбудованих систем.

C++ також забезпечує розширюваність завдяки великій кількості бібліотек, таких як STL (Standard Template Library) і Boost. Вона має потужну спільноту розробників і підтримку в багатьох галузях, що робить її ключовим інструментом для створення надійного програмного забезпечення.

Переваги:

— висока продуктивність: код C++ компілюється в машинний код і виконується дуже швидко. Це робить мову ідеальною для критичних систем та ігор;
— кросплатформенність: програми на C++ можна запустити на багатьох операційних системах із незначними змінами коду;
— об'єктно-орієнтоване програмування: підтримка принципів інкапсуляції, успадкування та поліморфізму забезпечує модульність та повторне використання коду;

— узагальнене програмування: завдяки шаблонам код стає більш гнучким і дозволяє працювати з різними типами даних;
— контроль пам'яті: програміст отримує повний контроль над тим, як виділяється і звільняється пам'ять, що дозволяє досягти максимальної ефективності.

Недоліки:

— складний синтаксис: навчання C++ потребує багато часу через велику кількість можливостей та концепцій;
— ризик помилок з пам'яттю: робота з вказівниками може призвести до витоків пам'яті та інших складних для виявлення помилок;
— довший цикл розробки: компіляція великих проектів займає більше часу порівняно з інтерпретованими мовами;
— відсутність автоматичного керування пам'яттю: немає автоматичного збирання сміття (garbage collection), як у Java або Python, що збільшує ризик помилок;
— високий поріг входження: для новачків мова може здатися надто складною через багатство можливостей.

4.2. Мова програмування Java

Java – це об'єктно-орієнтована мова програмування, розроблена у 1995 році компанією Sun Microsystems (тепер належить Oracle). Її головною перевагою є концепція "Write Once, Run Anywhere" (WORA), що означає, що програми, написані на Java, можуть працювати на будь-якій платформі з підтримкою Java Virtual Machine (JVM). Це робить її однією з найпопулярніших мов для створення корпоративних додатків та Android-додатків.[16]

Java широко використовується в розробці веб-додатків, серверних рішень, мобільних програм та великих корпоративних систем. Мова підтримує багатопотоковість, що дозволяє створювати ефективні багатозадачні

програми. Програмістам не потрібно турбуватися про управління пам'яттю, адже Java має вбудований Garbage Collector для автоматичного очищення невикористаних об'єктів.[17]

Переваги:

— кросплатформеність: програми на Java можуть виконуватися на будь-якій операційній системі з JVM;
— безпека: JVM забезпечує високий рівень безпеки, що особливо важливо для веб-додатків;
— автоматичне керування пам'яттю: Java звільняє розробника від необхідності вручну звільняти пам'ять;
— багата екосистема: велика кількість фреймворків (Spring, Hibernate), бібліотек та інструментів полегшує розробку;
— підтримка багатопотоковості: дозволяє створювати програми, що ефективно використовують багатоядерні процесори.

Недоліки:

— нижча продуктивність у порівнянні з C++ через інтерпретацію байт-коду JVM;
— велика споживаність пам'яті: Java-додатки часто потребують більше оперативної пам'яті, ніж програми на інших мовах;
— залежність від JVM: виконання програми може залежати від версії JVM, що може створювати проблеми сумісності;
— більший обсяг коду: деякі операції вимагають більше рядків коду, ніж у Python чи Kotlin;
— платність: корпоративна версія Java підтримується Oracle, що може вимагати ліцензій.

4.3. Середовище розробки Arduino IDE

Arduino IDE – це інтегроване середовище розробки для написання та завантаження програм на плати Arduino та інші сумісні мікроконтролери.

Програмування для Arduino базується на C/C++, але з певними спрощеннями для початківців. Arduino IDE широко використовується для створення проектів автоматизації, робототехніки та IoT (інтернету речей) завдяки простоті інтерфейсу та багатій бібліотеці компонентів.[18]

Це середовище розробки підтримує багато популярних мікроконтролерів і має велику спільноту розробників, що полегшує пошук прикладів та готових рішень. Arduino IDE дозволяє програмісту швидко тестувати код та прототипувати проекти, що особливо корисно для навчальних та експериментальних завдань. Скріншот середовища розробки зображено на рисунку 4.1.

Рисунок4.1 – Середовище розробки Arduino IDE

Переваги:

— інтуїтивний інтерфейс: дозволяє швидко розпочати роботу навіть без досвіду програмування;
— широка спільнота: багато прикладів проектів та бібліотек з відкритим кодом для Arduino;
— швидкий прототипінг: розробники можуть швидко створювати прототипи пристроїв для IoT та автоматизації;

— кросплатформенність: доступна для Windows, macOS та Linux;
— сумісність із різними датчиками та модулями: підтримка великої кількості периферійних пристроїв.

Недоліки:

— обмежена продуктивність: Arduino не підходить для ресурсомістких завдань;
— відсутність інструментів для відлагодження: немає повноцінного відлагоджувача для пошуку складних помилок.
— застарілий інтерфейс: можливості налаштування обмежені порівняно з сучасними IDE;
— менш гнучке середовище: не має таких потужних інструментів, як Visual Studio Code чи IntelliJ IDEA;
— вимоги до знань електроніки: для роботи з платами Arduino потрібно знати основи електроніки.

4.4. Середовище розробки Intellij IDEA

IntelliJ IDEA – це інтегроване середовище розробки (IDE), створене компанією JetBrains, відомою своїми інноваційними інструментами для програмістів. IntelliJ IDEA вважається однією з найпотужніших і найзручніших IDE для Java та Kotlin, але також підтримує інші мови програмування, такі як Python, JavaScript, SQL, PHP, C++, та багато інших. Завдяки розширеній функціональності та інтелектуальному редагуванню, IntelliJ IDEA є вибором багатьох професійних розробників для створення корпоративних, веб- та мобільних додатків.[19]

IntelliJ IDEA забезпечує глибокий аналіз коду в реальному часі та надає пропозиції щодо оптимізації. Програма підтримує автоматичне доповнення коду, що значно скорочує час написання програм. Завдяки тісній інтеграції з Git, інструментами CI/CD та платформами розгортання, ця IDE дозволяє

ефективно працювати над великими проєктами у команді. Скріншот середовища розробки зображено на рисунку 4.2.

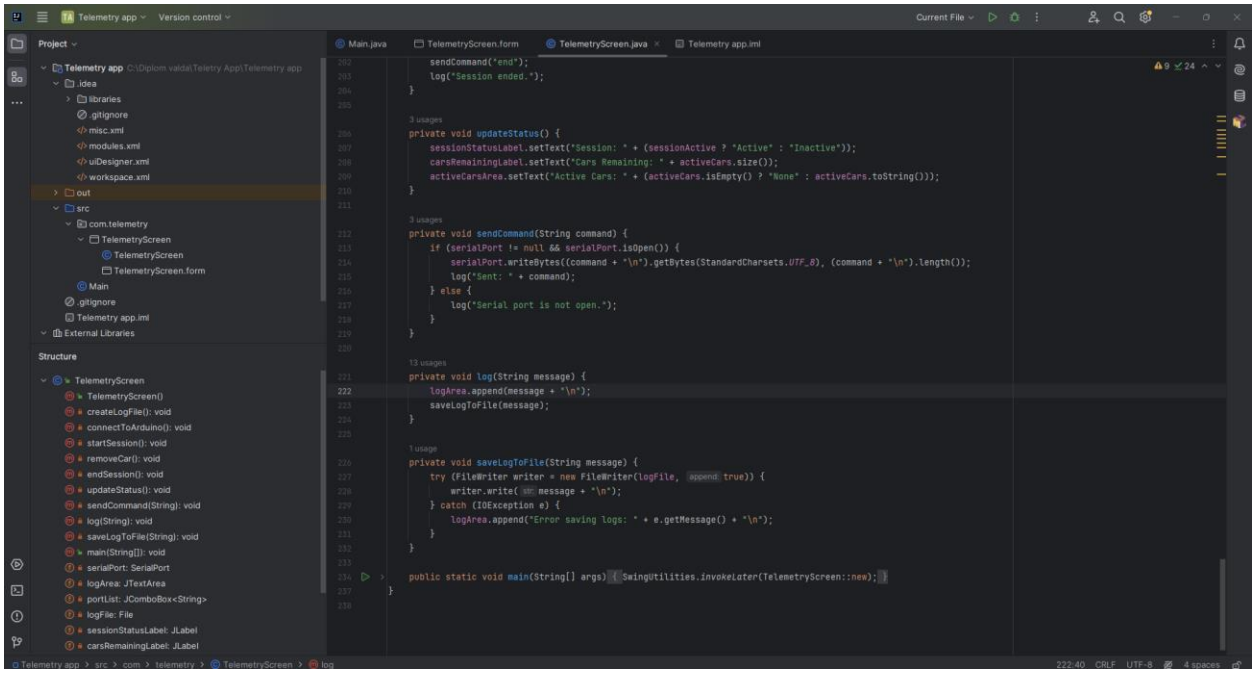


Рисунок4.2 – Середовище розробки IntelliJ IDEA

Основні можливості IntelliJ IDEA:

— інтелектуальне доповнення коду: система автоматично підказує можливі варіанти під час написання коду, включаючи методи, змінні, класи та бібліотеки;

— глибока підтримка Java і Kotlin: IntelliJ IDEA чудово інтегрується з платформами Java та Kotlin, надаючи спеціалізовані інструменти для відлагодження та компіляції;

— фвтоматична рефакторизація коду: допомагає оптимізувати код та зменшувати технічний борг завдяки автоматичній заміні та оновленню фрагментів коду;

— вбудований відлагоджувач: дозволяє крок за кроком відслідковувати виконання програм, що полегшує пошук помилок;

— інтеграція з інструментами контролю версій: підтримує Git, GitHub, SVN, Mercurial та інші системи контролю версій. Це полегшує командну роботу над проєктами;

— підтримка плагінів: користувачі можуть додавати нові функції за допомогою плагінів, розширюючи можливості IDE відповідно до потреб проєкту;
— інструменти для тестування: інтеграція з фреймворками JUnit, TestNG та іншими полегшує створення та виконання тестів для забезпечення якості коду.

Переваги IntelliJ IDEA:

— висока продуктивність розробки: автоматичне доповнення та рефакторизація коду скорочують час на написання та підтримку програм;
— широка підтримка мов і технологій: IntelliJ IDEA підтримує не лише Java, але й інші мови та фреймворки, включаючи Spring, Hibernate, JavaScript та SQL;
— інтуїтивний інтерфейс: зручний та адаптивний інтерфейс робить роботу з IDE комфортною навіть для новачків;
— потужний відлагоджувач: дозволяє ефективно знаходити помилки у великих та складних проєктах;
— активна спільнота та регулярні оновлення: JetBrains постійно оновлює IntelliJ IDEA, додаючи нові функції та підтримку сучасних технологій.

Недоліки IntelliJ IDEA:

— велике споживання ресурсів: IntelliJ IDEA потребує значної кількості оперативної пам'яті та потужного процесора для швидкої роботи, що може впливати на продуктивність старих комп'ютерів;
— вартість: хоча є безкоштовна версія Community Edition, повнофункціональна версія (Ultimate) потребує придбання ліцензії, що може бути дорогим для окремих користувачів;
— довгий час завантаження: при запуску великих проєктів IDE може працювати дещо повільно.
— складність для новачків: через багатий функціонал та можливості IntelliJ IDEA потребує часу на освоєння, особливо для початківців;

— залежність від плагінів: деякі необхідні функції доступні лише через плагіни, що може вимагати додаткових налаштувань.

IntelliJ IDEA – це потужний та гнучкий інструмент, який підходить як для початківців, так і для професійних розробників. Завдяки розширеним можливостям та підтримці багатьох мов програмування, вона стала незамінною для тих, хто працює з проєктами на Java та Kotlin. Попри те, що повнофункціональна версія може вимагати додаткових витрат, продуктивність і зручність, які забезпечує ця IDE, роблять її одним із найкращих інструментів для розробки сучасного програмного забезпечення.

4.5. Мікрокомп’ютер Arduino Uno

Arduino Uno – це одна з найпопулярніших і найуніверсальніших плат мікроконтролера в сімействі Arduino. Вона створена на основі мікроконтролера ATmega328P та ідеально підходить для початківців і досвідчених розробників. Плата забезпечує зручну платформу для створення прототипів електронних пристроїв завдяки відкритій архітектурі, простому програмуванню та широкій спільноті користувачів[20]. Плата зображена рисунку 4.3.

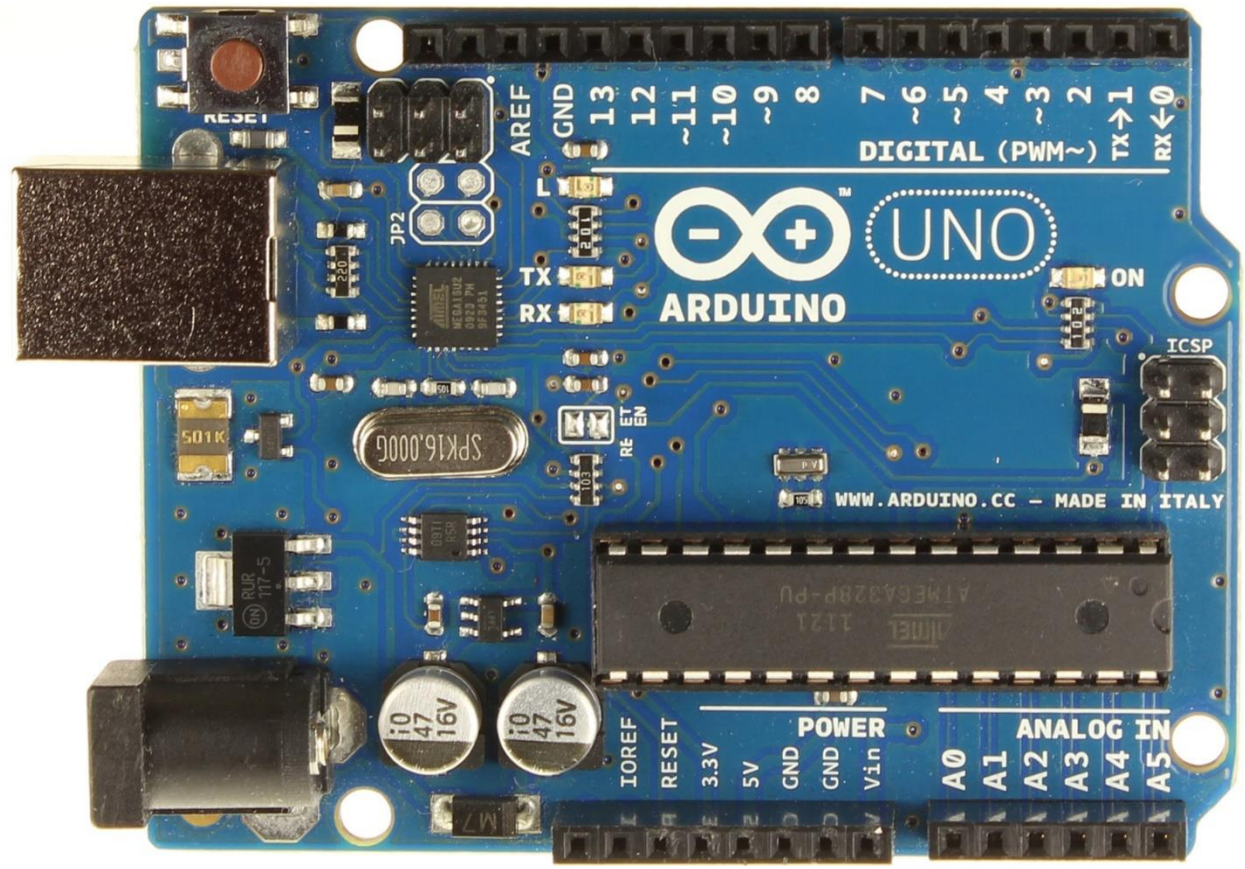


Рисунок 4.3 – Плата Arduino Uno

Основні характеристики:

— мікроконтролер: ATmega328P;

— напруга живлення: 5 В;

— вхідна напруга (рекомендована): 7–12 В;

— кількість цифрових входів/виходів (I/O): 14 (з них 6 можуть використовуватися для PWM);

— кількість аналогових входів: 6 (від A0 до A5);

— flash-пам’ять: 32 КБ (з них 0,5 КБ використовується для завантажувача);

— оперативна пам’ять (SRAM): 2 КБ;

— енергонезалежна пам’ять (EEPROM): 1 КБ;

— тактова частота: 16 МГц;

— інтерфейси: UART, SPI, I^2C;

— роз'єм живлення: DC Jack, USB, або через пін VIN;

— розміри: 68,6 × 53,4 мм.

Особливості:

— легкість програмування. Arduino Uno програмується через середовище Arduino IDE, що підтримує простий синтаксис і широкі можливості;
— широкий набір пінів. 14 цифрових пінів і 6 аналогових дозволяють підключати датчики, модулі, дисплеї та інші периферійні пристрої;
— підключення через USB. Завдяки вбудованому чіпу ATmega16U2 плата легко підключається до комп'ютера, що спрощує як живлення, так і передачу даних;
— сумісність із шилдами. Arduino Uno підтримує численні модулі-розширення, які дозволяють швидко збільшити функціонал проєкту.

Застосування: Arduino Uno широко використовується у проектах автоматизації, робототехніки, системах розумного дому, керуванні датчиками та дисплеями, а також у навчальних проектах для вивчення основ електроніки та програмування.

4.6. Підвищувальний перетворювач XL6009E1

XL6009E1 – це високоінтегрований імпульсний перетворювач постійного струму типу Boost Converter. Він здатний підвищувати вхідну напругу до бажаного рівня, що робить його ідеальним для використання в портативних пристроях, світлодіодному освітленні, системах живлення та інших електронних проектах[21]. Зображення плати показано на рисунку 4.4.

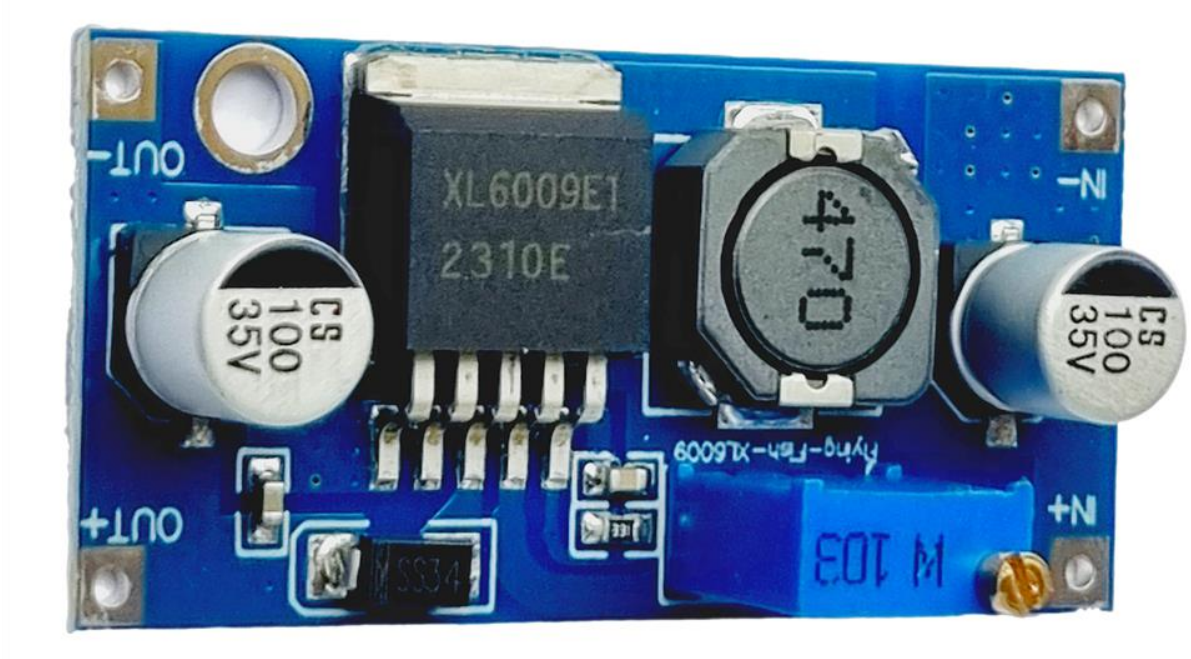


Рисунок 4.4 – Плата XL6009E1

Основні характеристики:

— тип регулятора: Підвищувальний (Boost);
— діапазон вхідної напруги: 3–32 В;
— діапазон вихідної напруги: 5–35 В (регульована);
— максимальний вихідний струм: До 4 А (рекомендується використовувати до 2 А для стабільної роботи);
— коефіцієнт ефективності: До 94%;
— частота перемикання: 400 кГц;
— інтегрований захист: Від перевантаження, короткого замикання і перегріву.

Особливості:

— регулювання напруги. Завдяки потенціометру можна плавно регулювати вихідну напругу;
— висока стабільність. Добре підходить для живлення чутливих до стабільності пристроїв;
— компактний розмір. Універсальна друкована плата дозволяє легко інтегрувати модуль у проєкти.

Застосування:

— зарядка акумуляторів;

— живлення світлодіодів;

— перетворення живлення для портативних пристроїв;

— автоматизовані системи керування.

4.7. Підвищувальний перетворювач MT3608

MT3608 — це високоефективний компактний підвищувальний DC-DC перетворювач. Його компактність і низьке енергоспоживання роблять його чудовим вибором для проектів з обмеженим простором або живленням від батарей[22]. Зображення плати показано на рисунку 4.5.

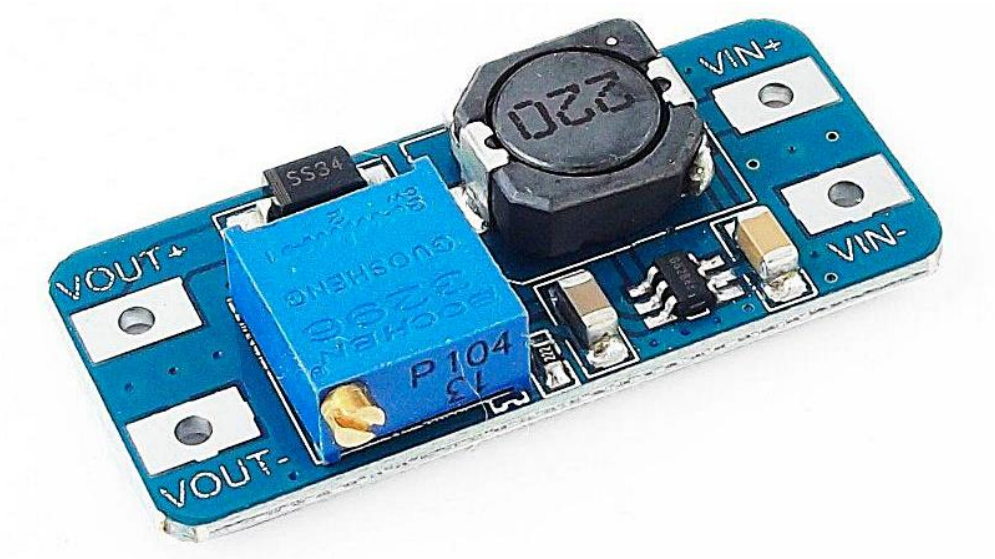


Рисунок 4.5 – Плата MT3608

Основні характеристики:

— тип регулятора: Підвищувальний (Boost);

— діапазон вхідної напруги: 2–24 В;

— діапазон вихідної напруги: 5–28 В (регульована);

— максимальний вихідний струм: До 2 А;

— коефіцієнт ефективності: До 93%;

— частота перемикання: 1,2 МГц;

— захист: Від перевантаження і перегріву.

Особливості:

— низький поріг вхідної напруги. Працює від 2 В, що дозволяє використовувати його з низьковольтними джерелами;
— простота використання. Потенціометр для швидкого налаштування вихідної напруги;
— економічність. Висока енергоефективність з мінімальними втратами.

Застосування:

— зарядні пристрої для мобільних телефонів та інших гаджетів;
— живлення датчиків у системах IoT;
— підвищення напруги для акумуляторних батарей;
— освітлювальні системи зі світлодіодами.

4.8. Фотоелектричний датчик Omron E3Z-T86

Фотоелектричний датчик Omron E3Z-T86 — це високоякісний пристрій з вбудованим підсилювачем, який забезпечує стабільну роботу навіть у складних умовах експлуатації. Він є частиною серії E3Z, яка відома своєю довговічністю та точністю. Датчик E3Z-T86 використовується для виявлення об'єктів на великій відстані, що робить його ідеальним рішенням для промислової автоматизації, логістики та інших застосувань, де потрібна надійна детекція.[23]

Основні характеристики:

— тип датчика: Through-beam (передавач + приймач);
— дальність виявлення: до 30 метрів;
— джерело світла: інфрачервоний світлодіод (870 нм);
— захист від зовнішнього світла: унікальний алгоритм запобігає впливу зовнішнього освітлення;
— захист від неправильного підключення: вбудований захист від зворотної полярності;

— ступінь захисту: IP67 (захист від пилу та води);

— робочий діапазон температур: від –40 до 55°C.

Розпіновка та підключення:

Датчик E3Z-T86 оснащений стандартним M8-роз'ємом, що значно спрощує його підключення до промислових систем. Розпіновка роз'єму наступна (див. рисунок 4.6):

— пін 1: Плюс живлення (+V);

— пін 2: Не використовується;

— пін 3: Мінус живлення (0V);

— пін 4: Вихідний сигнал.

Для коректного підключення важливо дотримуватися схеми. Ця схема демонструє, як правильно підключити датчик до джерела живлення та навантаження, щоб уникнути помилок і забезпечити стабільну роботу.

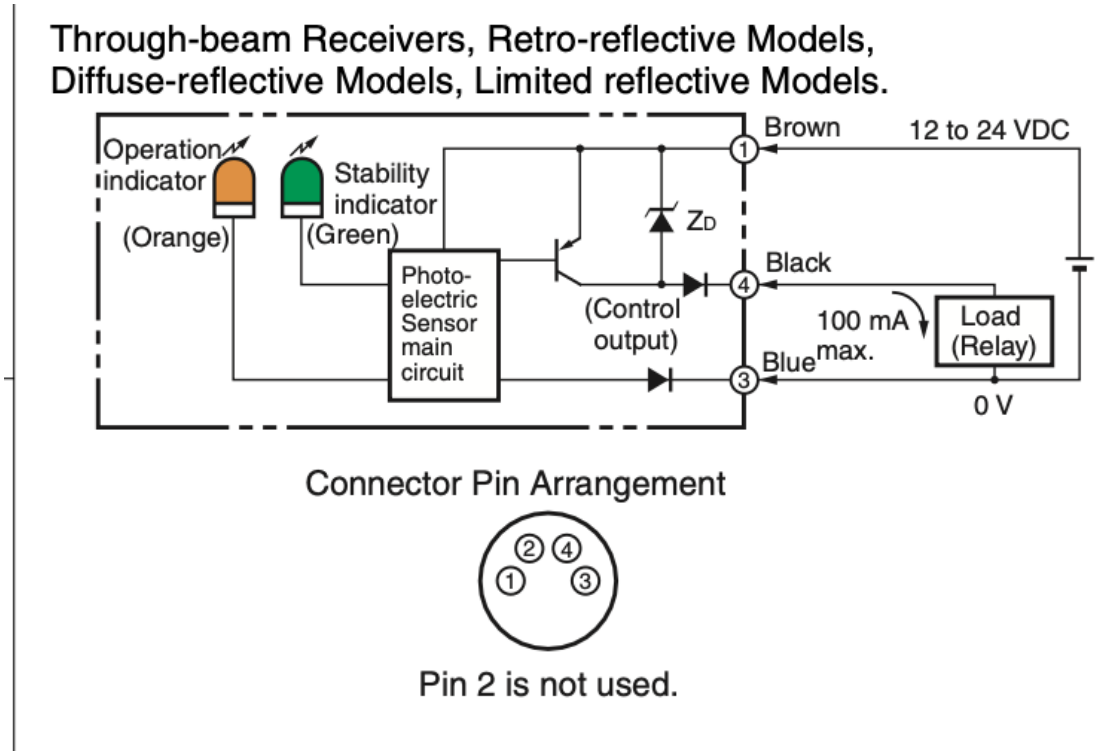


Рисунок 4.6 – Схема роботи датчика-прийомника

Переваги:

— простота установки: Завдяки мінімальному відхиленню між оптичною та механічною осями (±2.5°), налаштування датчика відбувається швидко та точно;
— надійність у холодних умовах: Датчик може працювати при температурах до –40°C, що робить його придатним для використання в холодильних складах;
— захист від помилок: Вбудований захист від зворотної полярності та короткого замикання забезпечує безпеку та довговічність пристрою;
— відповідність стандартам: Датчик повністю відповідає вимогам директив RoHS, що підтверджує його екологічну безпеку.

Застосування: датчик E3Z-T86 ідеально підходить для використання в системах автоматизації, де потрібне точне виявлення об'єктів на великій відстані. Він часто використовується в логістиці, на конвеєрних лініях, у системах сортування та інших промислових процесах.

Omron E3Z-T86 – це надійний і ефективний фотоелектричний датчик, який забезпечує стабільну роботу в різних умовах експлуатації. Його проста інтеграція, захист від помилок та відповідність міжнародним стандартам роблять його відмінним вибором для промислових застосувань. Для детальнішої інформації про підключення звертайтеся до рисунку 4.1, де наведено розпіновку та схему підключення.

4.9. Схема телеметрії

Так як Arduino Uno видає лише 5V що не вистачає для живлення фотоелектричного датчика який споживає 12V, було використано 2 плати підвищувального перетворювача. Одна плата для підвищення живлення, інша для пониження, так як датчки передає живлення по чорному кабелю і показує чи є сигнал від датчкик-відправника чи ні. Ардуіно може обчислити зміни в напрузі якщо вона 5V або менше. Плата підвищувального перетворювача

XL600E1 має наступні піни – In+,In-, Out+,Out-. Піни входу підключені до Ardino в піни 5V(+) та GND(-), вихідні піни підключені до фотоелектродатчику + до коричневого кабелю, а – до синього. Плати підвищувального перетворювача MT3608 має такі піни: Vin+, Vin-, Vout+, Vout-. Чорний кабель фотоелектродатчику під'єднано до Vin+ бо потрібно його понижувати, вхідний – береться з датчику піну IN- в платі XL6009E1. Вихідний + йде до аналогового піна Arduino A5, вихідний пін – не використовуєтсья бо – Arduino для обчислення бере з GND. Для використання іншого обладнння для пониження живлення на платі MT3608 для – потрібно використовувати вхідний +, в іншому випадку плата буде підвищувати напругу. Схема підключення телеметрії зображена на рисунку 4.7.

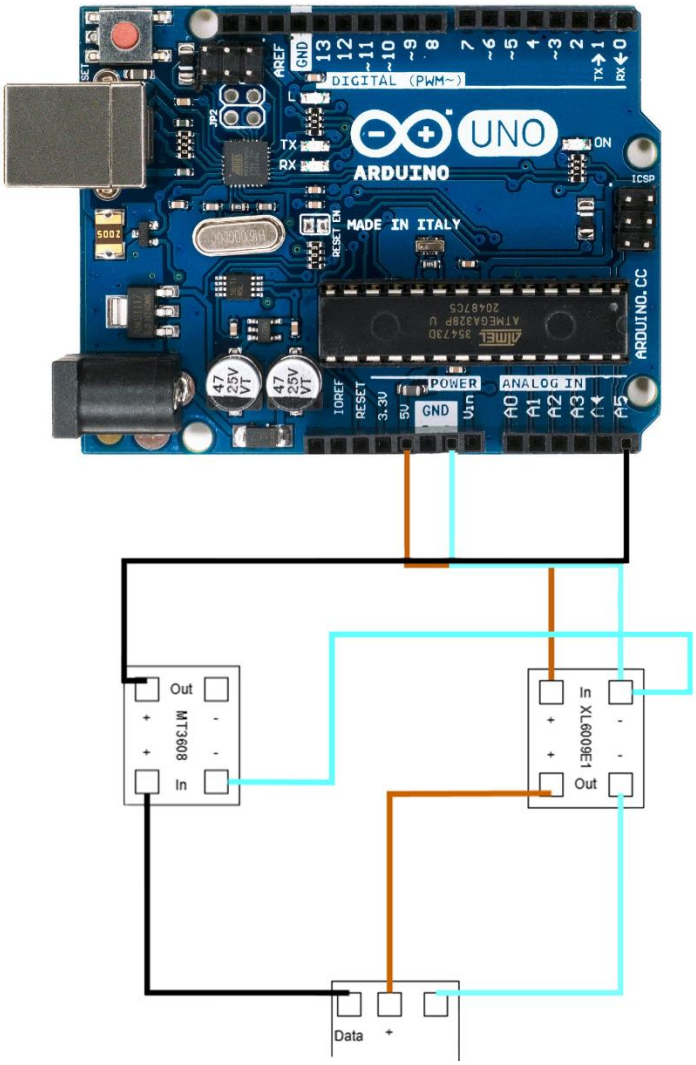


Рисунок 4.7 – Схема підключення фотоелектродатчика до ардуіно

РОЗДІЛ 5

Опис можливостей системи та реалізованого функціоналу, інструкція

5.1. Можливості системи

Можливості системи – це сукупність функцій, здатностей та сервісів, які програмне забезпечення, апаратний комплекс або інформаційний продукт може надавати чи виконувати. Вони визначають, які дії та операції доступні користувачам або іншим системам, і як вони можуть використовувати систему для досягнення своїх цілей.

У розробленій системі лазерної телеметрії для автоперегонів типу тайм-атак реалізовано наступні можливості:

— фіксація часу кола – система використовує лазерний сенсор для точного відсікання часу проходження кожного кола;
— ведення телеметрії в режимі реального часу – передача даних про час проходження кіл та інші параметри в додаток на ПК або мобільний пристрій;
— управління режимами роботи системи – активація та деактивація ручного режиму через Java-додаток;
— налаштування кількості учасників – можливість вибору кількості машин для одночасної участі у змаганнях;
— заборона обгонів – функція моніторингу та попередження порушення правил змагань;
— гнучка інтеграція із зовнішніми додатками логування даних і збереження статистики для подальшого аналізу;
— розширені можливості налаштувань – можливість адаптації параметрів системи через ПК для різних сценаріїв використання.

Ці функціональні можливості роблять систему універсальною для застосування в умовах спортивних перегонів, дозволяючи операторам і водіям отримувати точні дані для аналізу та вдосконалення результатів.

5.2. Тестування системи

Для тестування розробленого програмного продукту було обрано 4 типи тестування: monkey testing, beta testing, testcase, функціональне тестування та тестування інтерфейсу користувача.

Monkey testing(тестування мавпою) – це методика тестування програмного забезпечення, при якій випадкові вхідні дані надсилаються до тестованої програми для вивчення її поведінки та виявлення потенційних проблем. Вона полягає в генерації випадкових подій, таких як клацання, дотики, жести та натискання клавіш, та їх передачі програмі.[24]

Метою тестування мавпою є моделювання реальних сценаріїв використання та виявлення непередбачуваної або необробленої поведінки програми. Шляхом випадкового взаємодії з програмою вона спрямовується на виявлення проблем, таких як збої, зависання, нереагування, витоки пам'яті та інші аномалії, які можуть виникати в різних сценаріях використання.

Beta testing(бета-тестування) – один з етапів процесу тестування програмного забезпечення. Це вид зовнішнього тестування, коли програма, що знаходиться у фінальній стадії розробки, надсилається користувачам або обмеженій групі зовнішніх тестерів для використання в реальних умовах перед офіційним випуском. [25]

Метою бета-тестування є отримання реального відгуку користувачів щодо функціональності, продуктивності, стабільності та зручності використання програми. Це дає розробникам можливість виявити та виправити потенційні проблеми, помилки та недоліки перед широким випуском програми на ринок.

Testcase – це документ, який описує конкретні вхідні дані, дії виконавця та очікувані результати для перевірки правильності роботи певного

програмного продукту або системи. Тест-кейси використовуються в процесі тестування програмного забезпечення для перевірки функціональності, якості та надійності системи.[26]

Функціональне тестування – одним з основних видів тестування програмного забезпечення. Його основна мета полягає в перевірці функціональності програми згідно з вимогами та очікуваннями користувачів.

Під час функціонального тестування перевіряються функції, операції та можливості програмного забезпечення з точки зору правильності їх виконання. Тестувальник переконується, що програма виконує заплановані функції згідно з вимогами та специфікаціями, і надає очікувані результати.

Тестування інтерфейсу користувача – процес перевірки та оцінки користувальницького інтерфейсу програмного забезпечення. Його ціль полягає в перевірці, чи відповідає інтерфейс програми вимогам дизайну, функціональності та використання.

Під час тестування розробленого програмного продукту кожним з цих видів тестування значних помилок не виявлено. Результати testcase можна переглянути у Додатку Є. Всі помилки було виправлено.

5.3. Опис реалізованого функціоналу

Функціонал програмного забезпечення – сукупність функцій або операцій, які програма або додаток може виконувати. Він описує, які можливості та сервіси доступні користувачам через програму.

В розробленому програмному продукті розроблений наступний функціонал:

Автоматичне вимірювання часу проходження кіл:

— лазерні датчики фіксують кожен проїзд карта через фінішну лінію;
— контролер Arduino обробляє сигнал і записує час з точністю до мілісекунд;
— програмне забезпечення забезпечує миттєву передачу даних для аналізу.

Налаштування параметрів змагань:

— адміністратор може налаштовувати кількість кіл у заїзді, мінімальний час кола та інші параметри;
— можливість зміни налаштувань під час змагань без перезапуску програми.

Сповіщення про порушення або технічні проблеми:

— програма надсилає автоматичні попередження у разі виявлення проблем (наприклад, збій у фіксації часу);
— індикація можливих помилок датчиків або несправностей контролера на екрані суддів.

Інтеграція з іншими системами:

— можливість передавати результати змагань на веб-платформу комплексу «Лтава» для віддаленого перегляду;
— сумісність з іншими інформаційними системами для подальшої обробки та аналізу даних.

Безпека та конфіденційність:

— доступ до налаштувань системи та даних захищений паролем;
— дані про учасників зберігаються локально з дотриманням вимог конфіденційності.

Можливість модернізації та розширення функціоналу:

— програмне забезпечення побудоване з урахуванням можливості додавання нових функцій;
— код оптимізований для швидкої адаптації під нові вимоги та типи змагань.

5.4. Інструкція з експлуатації

Для використання системи, в першу чергу потрібно підєднати плату Arduino Uno з підключеною схемою та датчиком вхідного лазеру до пк за допомогою Usb Type-A кабелю. Потім потірібно підєднати вихідний лазер до

12V акумулятору не переплутавши полярніть, однаковими кольорами крокодилів до клем. Після цього обідва датчики засвітяться, для правильного висталення датчиків, потрібно їх направити друг на друга, коли вихідний лазер датчик напривиться точно в вхідний датчик, на вхідному датчику окрім зеленого світлодіода загориться помаранчевий. Потім потрібно запустити додаток на ПК, після вверхній частині обрати в випадаючому меню COM-порт плати. Після для запуску сесії треба ввести кількість машин на натиснути кнопку Start session. Для видалення машини з сесії потрібно ввести номер машини та натиснути кнопку Remove car. Для завершення сессії потрібно натиснути кнопку End session. Для того щоб вимкнути систему треба закрити програму на ПК, для збереження логів, вони будуть в текстовому файлі в папці «Документи». Потім потрібно відєднати плату від Пк та зняти крокодили з акумулятора.

ВИСНОВКИ

Дипломна робота на тему «Розробка системи лазерної телеметрії для картодрому» була спрямована на створення програмного рішення, яке забезпечує точне вимірювання часу проходження кіл під час змагань з тайм атак на базі технічно-спортивного комплексу «Лтава ім. Володимира Черниша». Це рішення має важливе значення для організації спортивних заходів, оскільки забезпечує об’єктивність результатів та підвищує рівень змагань.

У ході роботи було проведено дослідження сучасних підходів до вимірювання часу у спортивних дисциплінах, особливо тих, де точність відіграє критичну роль. Аналіз показав, що впровадження автоматизованих систем телеметрії є актуальним напрямом у спортивній інженерії, оскільки такі системи мінімізують можливість людських помилок та забезпечують точність до мілісекунд.

У результаті виконання дипломної роботи було створено систему лазерної телеметрії на базі платформи Arduino, яка дозволяє фіксувати моменти перетину фінішної лінії кожним учасником заїзду та передавати ці дані для подальшого аналізу. Система складається з таких компонентів:

— лазерні датчики для безконтактного виявлення картів;
— Arduino-контролер для збору та обробки сигналів від сенсорів;
— програмний модуль на C++, який відповідає за роботу логіки системи;
— Інтерфейс для відображення результатів у режимі реального часу.

Основні функції та можливості системи:

— вимірювання часу проходження кожного кола з високою точністю;
— автоматична фіксація результатів та їх обробка без участі суддів;
— збереження даних для подальшого аналізу та формування статистики;
— зручний інтерфейс для перегляду результатів під час та після змагань.

Під час реалізації проєкту було використано знання з програмування, електроніки та аналізу даних. Розроблений код написаний на C++ та Java, що

забезпечує високу продуктивність та надійність системи. Arduino виступив як гнучка платформа, яка дозволила легко інтегрувати датчики та реалізувати алгоритми обробки сигналів.

Переваги розробленої системи:

— висока точність і швидкість обробки даних;
— автоматизація процесу вимірювань, що мінімізує людський фактор;
— можливість інтеграції з іншими інформаційними системами комплексу «Лтава»;
— гнучкість у налаштуванні та адаптації системи для інших видів змагань.

Тестування та результати:

У процесі тестування системи було проведено серію експериментів під час тренувальних заїздів на картодромі «Лтава». В експериментах брали участь добровільні учасники, що дозволило оцінити функціональність та ефективність роботи системи в реальних умовах. Результати тестувань показали:

— високу точність розпізнавання проходження картів через фінішну лінію;
— швидку обробку сигналів, що забезпечує оперативне оновлення результатів у реальному часі;
— зручний інтерфейс для відображення результатів, який отримав позитивні відгуки від учасників і суддів.

Отримані результати демонструють успішність розробки та її значний потенціал для подальшого використання.

Безпека та конфіденційність:

Під час розробки системи було враховано вимоги до безпеки даних. Усі результати зберігаються локально, що забезпечує захист від несанкціонованого доступу та витоку інформації. Конфіденційність даних учасників гарантована, оскільки система не зберігає особисту інформацію.

Підсумовуючи результати дипломної роботи, можна стверджувати, що система лазерної телеметрії є важливим кроком у розвитку інфраструктури

спортивного комплексу «Лтава». Розробка забезпечує ефективне та точне проведення змагань з картингу, підвищуючи їх рівень і об'єктивність.

У майбутньому розроблену систему можна адаптувати для інших видів спортивних заходів, таких як мотоциклетні або автомобільні перегони. Крім того, впровадження подібних рішень може сприяти популяризації технічних видів спорту та залученню молоді до занять спортом.

Загалом, виконана дипломна робота зробила значний внесок у підвищення точності та автоматизації спортивних вимірювань, що сприятиме розвитку спортивної інфраструктури та підвищенню рівня змагань у майбутньому. Знімки роботи системи лазерної телеметрії та результати тестувань можна переглянути у Додатках Є та Ж.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ltavaring: [Веб-сайт]. URL: https://ltavaring.com/ (дата звернення: 09.09.2024).
2. Telemetry in Auto Racing: [Веб-сайт]. URL: https://eu.mouser.com/applications/automotive-racing-telemetry/?srsltid=AfmBOooM85bIqvxe7JBnbmN2p1BvQT1DaSmmeynPan1GExUuiAs7YTQt (дата звернення: 09.09.2024).
3. Than and Now – Telemetry in F1: [Веб-сайт]. URL: https://theparcferme.com/then-and-now-telemetry-in-f1/ (дата звернення: 09.09.2024).
4. MyLaps Telemetry: [Веб-сайт]. URL: https://www.mylaps.com/ (дата звернення: 10.09.2024).
5. Timing - Watches - FAQ - TAG Heuer: [Веб-сайт]. URL: https://faq.tagheuer.com/?l=en_US&c=Products:Timing (дата звернення: 10.09.2024).
6. RACE RESULT – timing and scoring of sport events: [Веб-сайт]. URL: https://www.raceresult.com/en-ru/home/index (дата звернення: 10.09.2024).
7. Chronotrack – world-class registration and Race: [Веб-сайт]. URL: https://chronotrack.com/ (дата звернення: 10.09.2024).
8. ALGE-TIMING: [Веб-сайт]. URL: https://alge-timing.com/ (дата звернення: 10.09.2024).
9. Діаграма Прецедентів (Use Case UML Diagram): [Веб-сайт]. URL: https://lvivqaclub.blogspot.com/2008/10/use-case-uml-diagram.html (дата звернення: 11.09.2024).
10. Sonmes J. O. Soft skills. The software developers life manual / за ред. Scott Hanselman and Robert C. Martin. SHELTER ISLAND: Manning, 2015. 506 с.

11. Rapid prototyping UX/UI продуктів – YellowArrow.Design: [Веб-сайт]. URL: https://yellowarrow.design/index.php/ua/blog-article/84-rapid-prototyping-ux-ui (дата звернення: 12.09.2024).
12. Архітектура програмного забеспечення: [Веб-сайт]. URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya (дата звернення: 13.09.2024).
13. Класи в програмуванні: ваш шлях в ООП: [Веб-сайт]. URL: https://foxminded.ua/klasy-v-prohramuvanni/ (дата звернення: 13.09.2024).
14. Діаграми класів: [Веб-сайт]. URL: https://ua5.org/oop/392-diagrami-klasiv.html (дата звернення: 13.09.2024).
15. C++ Introduction: [Веб-сайт]. URL: https://www.w3schools.com/cpp/cpp_intro.asp (дата звернення: 15.09.2024).
16. Java | Oracle: [Веб-сайт]. URL: https://www.java.com/ua/ (дата звернення: 15.09.2024).
17. Haralabids N. O. Oracle JDeveloper 11gR2 Cookbook. Birmingham: Packt Publishing, 2012. 406 с.
18. Software: Arduino IDE: [Веб-сайт]. URL: https://www.arduino.cc/en/software (дата звернення: 15.09.2024).
19. IntelliJ IDEA – the Leader Java and Kotlin IDE – JetBrains: [Веб-сайт]. URL: https://www.jetbrains.com/idea/ (дата звернення: 15.09.2024).
20. Arduino UNO R4 Minima: [Веб-сайт]. URL: https://store.arduino.cc/collections/boards-modules/products/uno-r4-minima?_pos=1&_fid=14b65ed3c&_ss=c (дата звернення: 15.09.2024).
21. Модуль XL6009E1 DC-DC 4A (max.) підвищуючий з регулюванням напруги: [Веб-сайт]. URL: https://radioformat.in.ua/ua/p1630180106-modul-xl6009e1-max.html (дата звернення: 15.09.2024).
22. Модуль MT3608 DC-DC 2A підвищуючий з регулюванням напруги: [Веб-сайт]. URL: https://radioformat.in.ua/ua/p1629198984-modul-mt3608-povyshayuschij.html (дата звернення: 15.09.2024).

23. E3Z-T86 – OMRON Industrial Automation: URL: https://www.ia.omron.com/product/item/773/ (дата звернення: 15.09.2024).
24. What Is Monkey testing: URL: https://qatestlab.com/resources/knowledge-center/monkey-testing-technique/ (дата звернення: 15.10.2024).
25. What is a Beta Test? | Definition: URL: https://www.productplan.com/glossary/beta-test/ (дата звернення: 15.10.2024).
26. Правила оформлення теми тест-кейса: URL: https://training.qatestlab.com/blog/course-materials/test-case-topic/ (дата звернення: 15.10.2024).
27. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 122 «Комп’ютерні науки»;[уклад.: М. І. Демиденко, О.А. Руденко]. – Полтава: НУПП, 2024.– 54 с

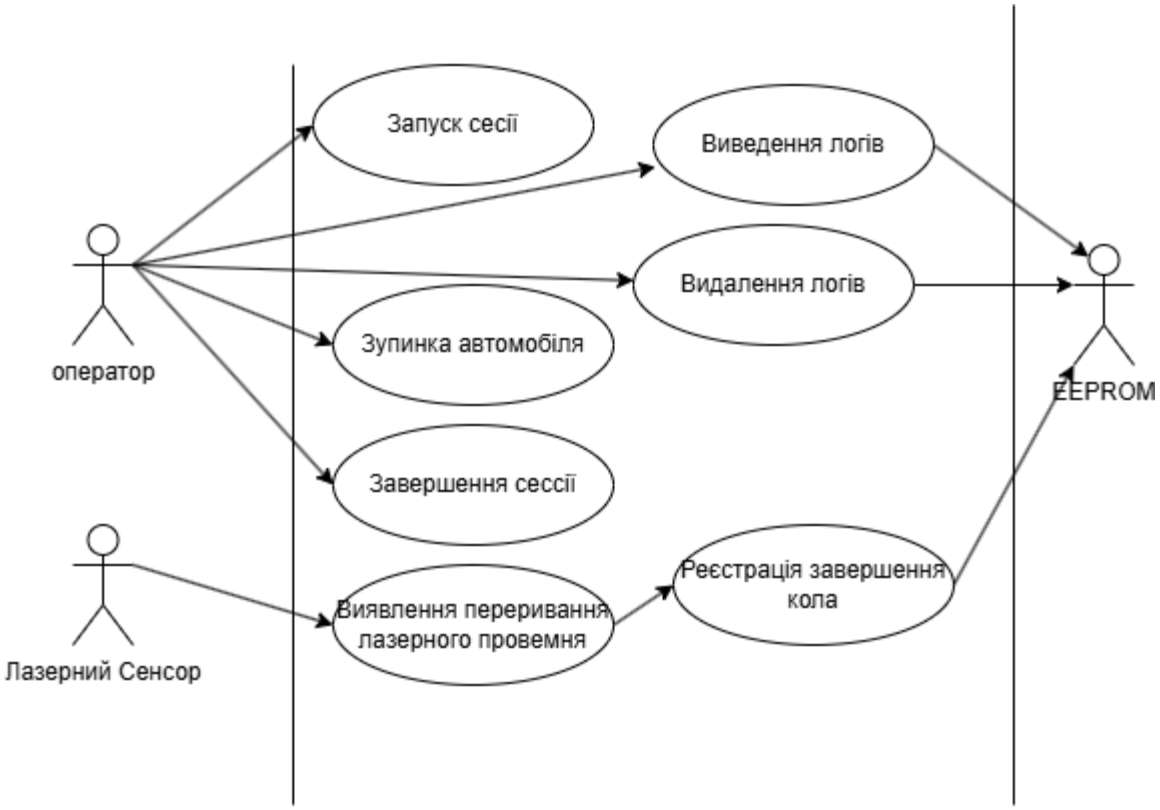
ДОДАТОК А

Порівняльна характеристика існуючих продуктів аналогічного призначення

Система	Точність вимірювання	Інтеграція з іншими системами	Простота налаштування	Вартість
MyLAPS	0.001	+	-	20000$
Tag Heuer	0.001	-	-	18000$
Raceresult	0.001	+	+	19000$
ChronoTrack	0.001	+	-	14000$
RaceLogic	0.002	-	+	22000$
Моя система	0.001	+	+	510$

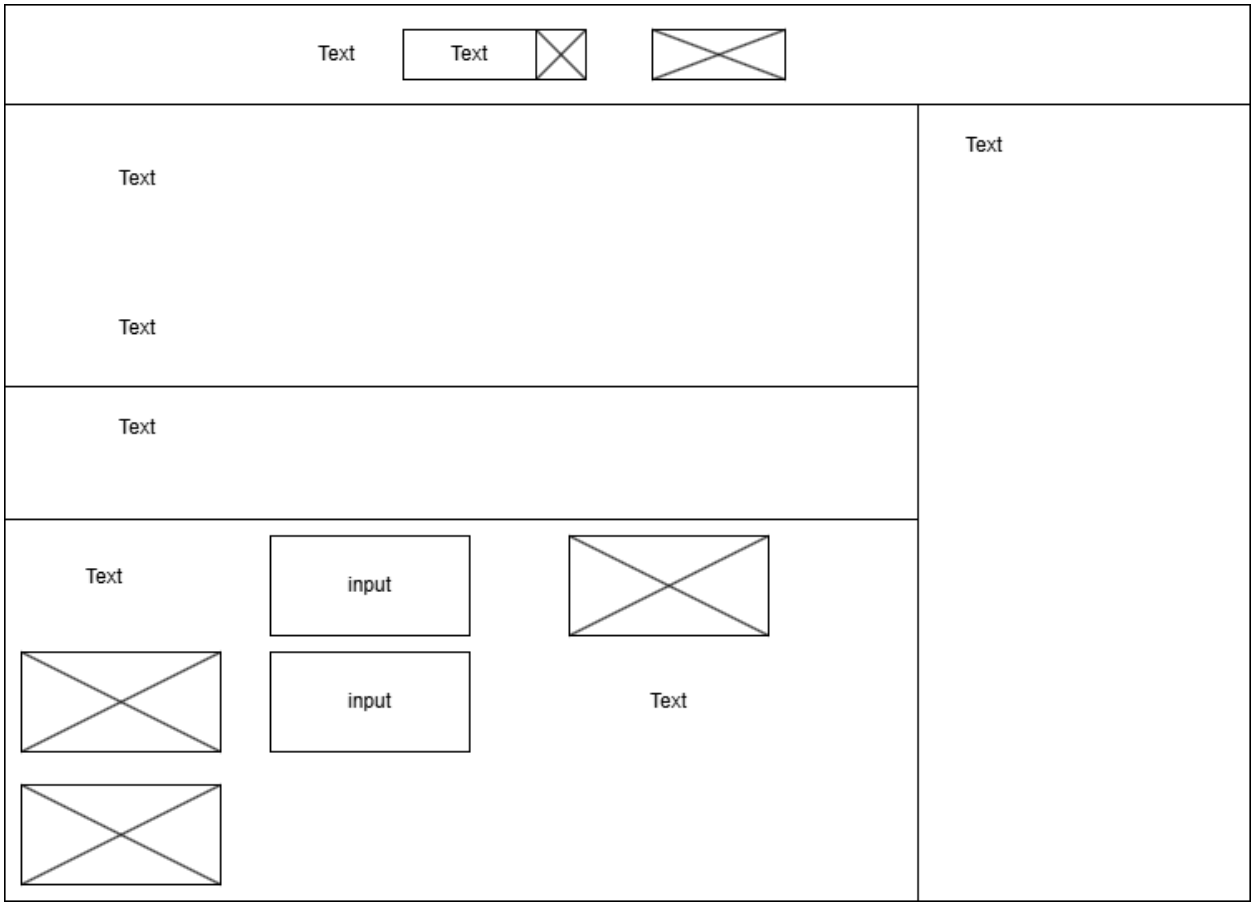
ДОДАТОК Б

Uml діаграма прецедентів



ДОДАТОК В

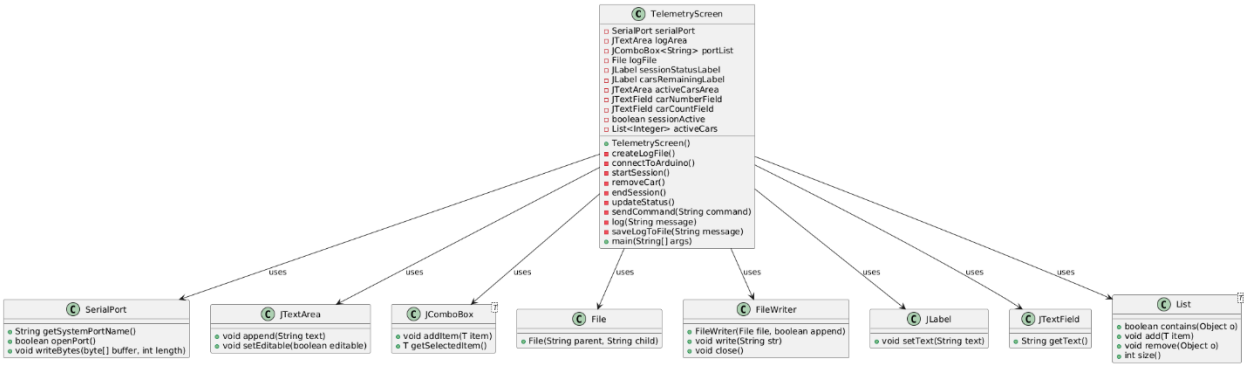
Модель інтерфейсу користувача



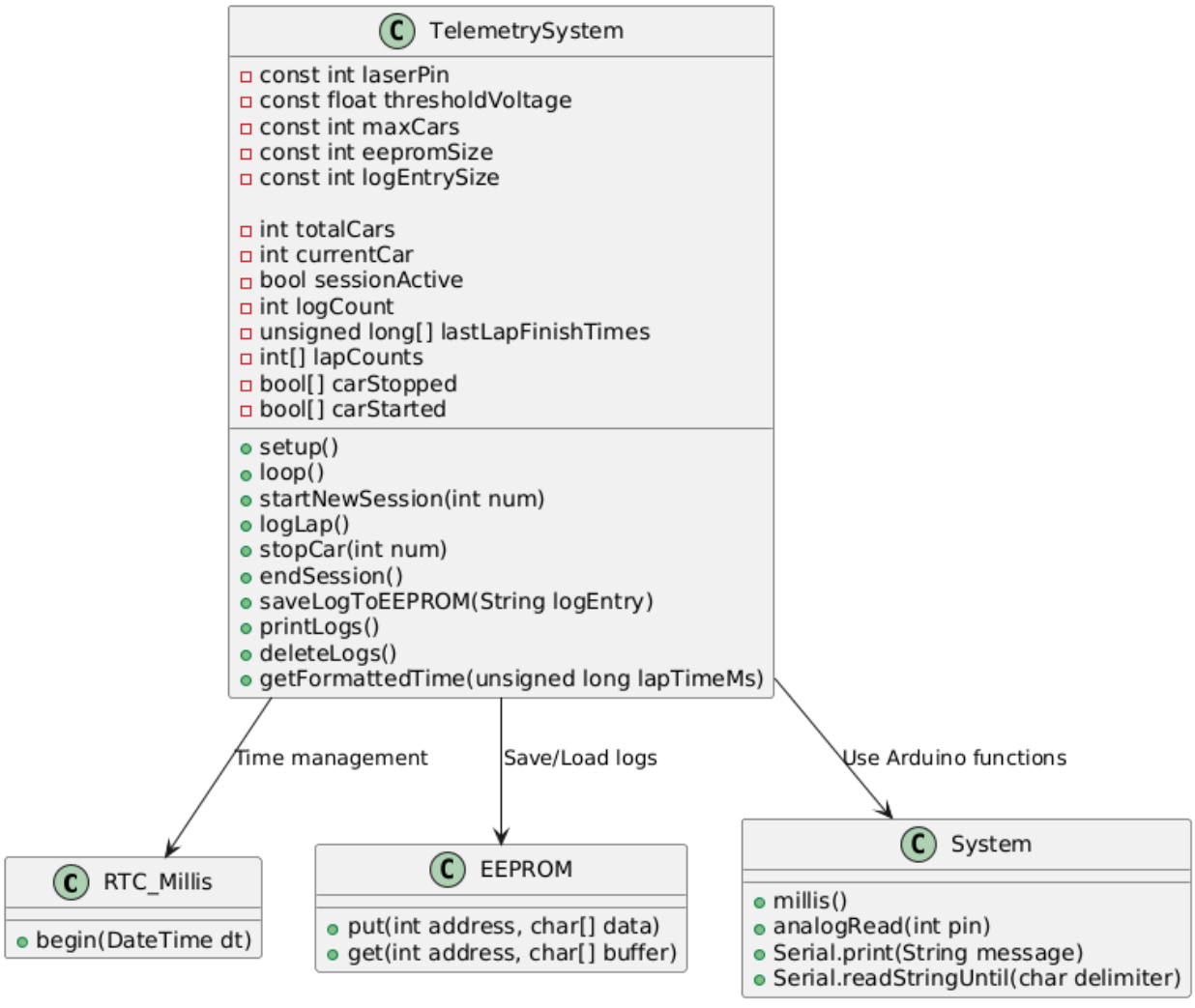
ДОДАТОК Г

Uml діаграма класів

Г.1. Діаграма класів Java-додатку

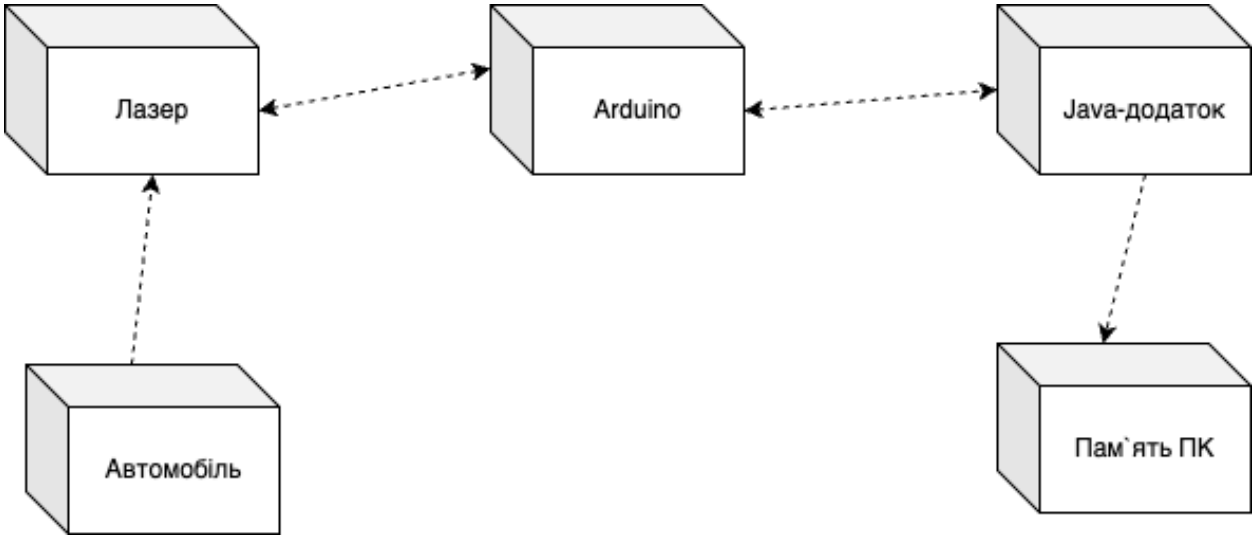


Г.2. Діаграма класів прошивки для плати Arduino Uno



ДОДАТОК Д

Uml діаграма розгортання



ДОДАТОК Е

Вихідні коди

Е.1. Вихідні коди прошивки для мікрокомп’ютера Arduino Uno

```
#include <EEPROM.h>
#include <RTClib.h>

RTC_Millis rtc;

const int laserPin = A5;             // Аналоговий пін для лазера
const float thresholdVoltage = 2.07; // Порогова напруга для виявлення об'єкта

const int maxCars = 10;              // Максимальна кількість машин
const int eepromSize = 1024;         // Максимальний розмір EEPROM
const int logEntrySize = 50;            // Максимальна довжина одного запису логу в символах

int totalCars = 0;
int currentCar = 0;
int logCount = 0;
unsigned long *lastLapFinishTimes;    // Динамічний масив для збереження часу фінішу останнього кола кожної машини
int *lapCounts;                      // Масив для зберігання кількості завершених кіл для кожного автомобіля
bool sessionActive = false;
bool *carStopped;                    // Динамічний масив для трекінгу зупинок машин
bool *carStarted;                   // Масив для визначення, чи машина вже стартувала
bool laserInterrupted = false;       // Чи був лазер перерваний?
unsigned long lastLapTime = 0;       // Час останнього зафіксованого кола
const unsigned long debounceDelay = 200; // Мінімальна затримка між колами (мс)

void setup() {
  Serial.begin(9600);
  rtc.begin(DateTime(F(__DATE__), F(__TIME__)));

  // Ініціалізація масивів
  carStopped = new bool[maxCars];
  carStarted = new bool[maxCars];
  lastLapFinishTimes = new unsigned long[maxCars];
  lapCounts = new int[maxCars];
  for (int i = 0; i < maxCars; i++) {
    carStopped[i] = false;
    carStarted[i] = false; // Спочатку жодна машина не стартувала
    lastLapFinishTimes[i] = 0;
    lapCounts[i] = 0;      // Початковий номер кола для всіх машин
  }

  Serial.println("Система підрахунку кіл запущена.");
}

void loop() {
  // Зчитуємо напругу лазера
  int analogValue = analogRead(laserPin);
  float voltage = analogValue * 5.0 / 1023.0;


  // Виявлення переривання лазера
```

```
  unsigned long currentTime = millis();
  if (voltage < thresholdVoltage) { // Лазер перервано
    if (!laserInterrupted && (currentTime - lastLapTime > debounceDelay)) {
      laserInterrupted = true;   // Запобігання дублюванню
      lastLapTime = currentTime; // Оновлення часу останнього кола
      logLap();                  // Реєстрація кола
    }
  } else {
    laserInterrupted = false;    // Готовність до наступного переривання
  }

  // Читання команд із серійного порту
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
    command.trim();

    if (command.startsWith("start")) {
      int num;
      sscanf(command.c_str(), "start %d", &num);
      startNewSession(num);
    } else if (command.startsWith("stop")) {
      int num;
      sscanf(command.c_str(), "stop %d", &num);
      stopCar(num);
    } else if (command == "end") {
      endSession();
    } else if (command == "logs") {
      printLogs();
    } else if (command == "delLogs") {
      deleteLogs();
    }
  }
}

void startNewSession(int num) {
  delete[] carStopped;
  delete[] carStarted;
  delete[] lastLapFinishTimes;
  delete[] lapCounts;
  carStopped = new bool[maxCars];
  carStarted = new bool[maxCars];
  lastLapFinishTimes = new unsigned long[maxCars];
  lapCounts = new int[maxCars];
  for (int i = 0; i < maxCars; i++) {
    carStopped[i] = false;
    carStarted[i] = false;
    lastLapFinishTimes[i] = 0;
    lapCounts[i] = 0; // Скидаємо кількість кіл
  }

  totalCars = num;
  if (totalCars > maxCars) {
    Serial.println("Кількість машин перевищує максимальний розмір масиву.");
    return;
  }

  currentCar = 0;
  sessionActive = true;
  logCount = 0;

  Serial.println("Сесія розпочата");
}
```

```
void logLap() {
  if (!sessionActive) {
    Serial.println("Сесія не активна.");
    return;
  }

  bool allStopped = true;
  int originalCar = currentCar;

  do {
    if (!carStopped[currentCar]) {
      allStopped = false;
      break;
    }
    currentCar = (currentCar + 1) % totalCars;
  } while (currentCar != originalCar);

  if (allStopped) {
    Serial.println("Всі машини зупинені.");
    return;
  }

  unsigned long currentTime = millis();
  unsigned long lapTimeMs = currentTime - lastLapFinishTimes[currentCar];

  // Якщо машина ще не стартувала
  if (!carStarted[currentCar]) {
    carStarted[currentCar] = true;
    String logEntry = "Car " + String(currentCar + 1) + " started.";
    saveLogToEEPROM(logEntry);
    Serial.println(logEntry);
  } else {
    // Оновлення номера кола для поточної машини
    lapCounts[currentCar]++;
    String logEntry = "C" + String(currentCar + 1) + " L" +
String(lapCounts[currentCar]) + " -> ";
    logEntry += getFormattedTime(lapTimeMs);
    saveLogToEEPROM(logEntry);
    Serial.println(logEntry);
  }

  lastLapFinishTimes[currentCar] = currentTime;

  do {
    currentCar = (currentCar + 1) % totalCars;
  } while (carStopped[currentCar] && currentCar != originalCar);
}

void stopCar(int num) {
  if (num >= 1 && num <= totalCars) {
    carStopped[num - 1] = true;
    String logEntry = "Car " + String(num) + " stopped.";
    saveLogToEEPROM(logEntry);
    Serial.println(logEntry);
  } else {
    Serial.println("Невірний номер машини.");
  }
}

void endSession() {
  if (!sessionActive) {
```

```
    Serial.println("Сесія не почалась.");
    return;
  }

  String logEntry = "Session ended.";
  saveLogToEEPROM(logEntry);
  Serial.println(logEntry);

  sessionActive = false;
  for (int i = 0; i < totalCars; i++) {
    carStopped[i] = false;
  }
}

void saveLogToEEPROM(String logEntry) {
  if (logEntry.length() >= logEntrySize) {
    Serial.println("Лог занадто довгий для збереження.");
    return;
  }

  int address = logCount * logEntrySize;
  if (address + logEntrySize > eepromSize) {
    Serial.println("EEPROM заповнена, логи не можуть бути записані.");
    return;
  }

  char logBuffer[logEntrySize];
  logEntry.toCharArray(logBuffer, logEntrySize);
  EEPROM.put(address, logBuffer);
  logCount++;
}

void printLogs() {
  Serial.println("Логи:");
  for (int i = 0; i < logCount; i++) {
    char logBuffer[logEntrySize];
    EEPROM.get(i * logEntrySize, logBuffer);
    Serial.println(String(logBuffer));
  }
}

void deleteLogs() {
  logCount = 0;
  currentCar = 0;
  sessionActive = false;
  for (int i = 0; i < totalCars; i++) {
    carStopped[i] = false;
  }
  Serial.println("Логи видалено.");
}

String getFormattedTime(unsigned long lapTimeMs) {
  unsigned long hours = lapTimeMs / 3600000;
  unsigned long minutes = (lapTimeMs % 3600000) / 60000;
  unsigned long seconds = (lapTimeMs % 60000) / 1000;
  unsigned long milliseconds = lapTimeMs % 1000;

  char buffer[20];
  snprintf(buffer, sizeof(buffer), "%02lu:%02lu:%02lu,%03lu", hours, minutes, seconds,
milliseconds);
  return String(buffer);
}
```

Е.2. Вихідні коди Java-додатку

```
package com.telemetry;

import com.fazecast.jSerialComm.SerialPort;

import javax.swing.*;
import java.awt.*;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class TelemetryScreen {
    private SerialPort serialPort;
    private JTextArea logArea;
    private JComboBox<String> portList;
    private File logFile;
    private JLabel sessionStatusLabel;
    private JLabel carsRemainingLabel;
    private JTextArea activeCarsArea;
    private JTextField carNumberField;
    private JTextField carCountField;
    private boolean sessionActive = false;
    private List<Integer> activeCars = new ArrayList<>();

    public TelemetryScreen() {
        // Налаштування стилю
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Створюємо лог-файл
        createLogFile();

        // Налаштування GUI
        JFrame frame = new JFrame("Telemetry Control");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);
        frame.setLayout(new BorderLayout());

        // Верхня панель для вибору COM-порту
        JPanel topPanel = new JPanel(new FlowLayout());
        topPanel.setBorder(BorderFactory.createTitledBorder("COM Port Selection"));
        JLabel portLabel = new JLabel("Select COM Port:");
        topPanel.add(portLabel);

        portList = new JComboBox<>();
        SerialPort[] ports = SerialPort.getCommPorts();
        for (SerialPort port : ports) {
            portList.addItem(port.getSystemPortName());
        }
        topPanel.add(portList);

        JButton connectButton = new JButton("Connect");
```

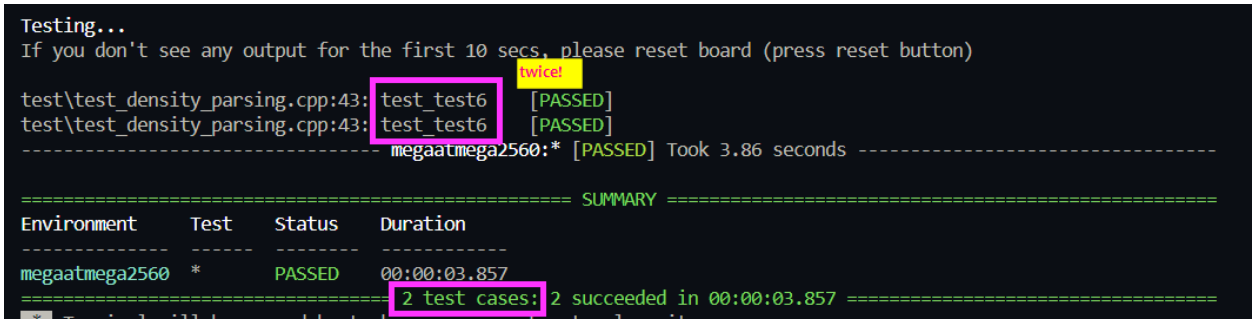
```
topPanel.add(connectButton);

frame.add(topPanel, BorderLayout.NORTH);

// Центральна панель для статусу і керування
JPanel centerPanel = new JPanel(new GridLayout(2, 1));
centerPanel.setBorder(BorderFactory.createTitledBorder("Telemetry"));

// Блок статусу
JPanel statusPanel = new JPanel(new GridLayout(3, 1));
statusPanel.setBorder(BorderFactory.createTitledBorder("Status"));

sessionStatusLabel = new JLabel("Session: Inactive");
sessionStatusLabel.setFont(new Font("Arial", Font.BOLD, 14));
statusPanel.add(sessionStatusLabel);

carsRemainingLabel = new JLabel("Cars Remaining: 0");
carsRemainingLabel.setFont(new Font("Arial", Font.BOLD, 14));
statusPanel.add(carsRemainingLabel);

activeCarsArea = new JTextArea("Active Cars: None");
activeCarsArea.setEditable(false);
activeCarsArea.setFont(new Font("Arial", Font.PLAIN, 12));
JScrollPane activeCarsScrollPane = new JScrollPane(activeCarsArea);
statusPanel.add(activeCarsScrollPane);

centerPanel.add(statusPanel);

// Панель керування кнопками
JPanel controlPanel = new JPanel(new GridLayout(3, 2, 10, 10));
controlPanel.setBorder(BorderFactory.createTitledBorder("Controls"));

JLabel carCountLabel = new JLabel("Enter Cars Count:");
carCountField = new JTextField(5);

JButton startSessionButton = new JButton("Start Session");
JButton removeCarButton = new JButton("Remove Car");
JButton endSessionButton = new JButton("End Session");

JLabel carNumberLabel = new JLabel("Car Number:");
carNumberField = new JTextField(5);

controlPanel.add(carCountLabel);
controlPanel.add(carCountField);
controlPanel.add(startSessionButton);
controlPanel.add(removeCarButton);
controlPanel.add(carNumberLabel);
controlPanel.add(carNumberField);
controlPanel.add(endSessionButton);

centerPanel.add(controlPanel);

// Панель для логів
JPanel logPanel = new JPanel(new BorderLayout());
logPanel.setBorder(BorderFactory.createTitledBorder("Logs"));
logArea = new JTextArea();
logArea.setEditable(false);
JScrollPane logScrollPane = new JScrollPane(logArea);
logPanel.add(logScrollPane, BorderLayout.CENTER);

// Лівий і правий поділ
JSplitPane splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
```

```
centerPanel, logPanel);
        splitPane.setDividerLocation(600);
        frame.add(splitPane, BorderLayout.CENTER);

        frame.setVisible(true);

        // Дії для кнопок
        connectButton.addActionListener(e -> connectToArduino());
        startSessionButton.addActionListener(e -> startSession());
        removeCarButton.addActionListener(e -> removeCar());
        endSessionButton.addActionListener(e -> endSession());
    }

    private void createLogFile() {
        String userHome = System.getProperty("user.home");
        String documentsPath = userHome + File.separator + "Documents";
        String fileName = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss").format(new
Date()) + "_logs.txt";
        logFile = new File(documentsPath, fileName);
    }

    private void connectToArduino() {
        String selectedPort = (String) portList.getSelectedItem();
        if (selectedPort == null) {
            log("No COM port selected.");
            return;
        }
        serialPort = SerialPort.getCommPort(selectedPort);
        serialPort.setBaudRate(9600);
        if (serialPort.openPort()) {
            log("Connected to " + selectedPort);
        } else {
            log("Failed to open port " + selectedPort);
        }
    }

    private void startSession() {
        String carCountText = carCountField.getText();
        try {
            int carCount = Integer.parseInt(carCountText);
            activeCars.clear();
            for (int i = 1; i <= carCount; i++) {
                activeCars.add(i);
            }
            sessionActive = true;
            updateStatus();
            sendCommand("start " + carCount);
            log("Session started with " + carCount + " cars.");
        } catch (NumberFormatException e) {
            log("Invalid car count.");
        }
    }

    private void removeCar() {
        if (!sessionActive) {
            log("No active session.");
            return;
        }
        String carNumberText = carNumberField.getText();
        try {
            int carNumber = Integer.parseInt(carNumberText);
            if (activeCars.contains(carNumber)) {
```

```
                activeCars.remove(Integer.valueOf(carNumber));
                updateStatus();
                sendCommand("stop " + carNumber);
                log("Car " + carNumber + " removed.");
            } else {
                log("Car " + carNumber + " is not active.");
            }
        } catch (NumberFormatException e) {
            log("Invalid car number.");
        }
    }

    private void endSession() {
        if (!sessionActive) {
            log("No active session.");
            return;
        }
        sessionActive = false;
        activeCars.clear();
        updateStatus();
        sendCommand("end");
        log("Session ended.");
    }

    private void updateStatus() {
        sessionStatusLabel.setText("Session: " + (sessionActive ? "Active" :
"Inactive"));
        carsRemainingLabel.setText("Cars Remaining: " + activeCars.size());
        activeCarsArea.setText("Active Cars: " + (activeCars.isEmpty() ? "None" :
activeCars.toString()));
    }

    private void sendCommand(String command) {
        if (serialPort != null && serialPort.isOpen()) {
            serialPort.writeBytes((command + "\n").getBytes(StandardCharsets.UTF_8),
(command + "\n").length());
            log("Sent: " + command);
        } else {
            log("Serial port is not open.");
        }
    }

    private void log(String message) {
        logArea.append(message + "\n");
        saveLogToFile(message);
    }

    private void saveLogToFile(String message) {
        try (FileWriter writer = new FileWriter(logFile, true)) {
            writer.write(message + "\n");
        } catch (IOException e) {
            logArea.append("Error saving logs: " + e.getMessage() + "\n");
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(TelemetryScreen::new);
    }
}
```

ДОДАТОК Є

Результати тестування



ДОДАТОК Ж

Знімки екрану