Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютерних технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Перший (бакалаврський)
(рівень вищої освіти)

Розроблення програмного модуля для відображення телеметрії з датчиків позиціонування мобільного робота
(тема)

Виконав:
студент 4 курсу, групи АКТСІ-20-2
Калюжний М.В.
(прізвище, ініціали)
Спеціальності 151 Автоматизація та комп'ютерно інтегровані технології
(код і повна назва спеціальності)
Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Системна інженерія
(повна назва освітньої програми)
Керівник проф. Новосєлов С.П.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТАР _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

2024 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно інтегровані технології
Тип програми Освітньо-професійна
Освітня програма Автоматизація та комп'ютерно-інтегровані технології

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«18» червня 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Калюжному Максиму Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення програмного модуля для відображення телеметрії з датчиків позиціонування мобільного робота
Затверджена наказом по університету від 03.06.2024 №545 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 18.06.2024
3. Вхідні дані до роботи
3.1 Компоненти для автоматизованої системи
3.2 Додаткові компоненти, датчики
3.3 Блок живлення
3.4 Комп'ютер для розробки проекту

4. Перелік питань, що потрібно розглянути у роботі
4.1 Вступ
4.2 Огляд теми та її актуальності
4.3 Вибір та обґрунтування компонентів системи
4.4 Розробка програмної частини
4.5 Тестування та налагодження системи
4.6 Висновки та перелік джерел посилань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій
Демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt) – 12 с. формату А4.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт-ка
1	Аналіз існуючих рішень для відображення телеметрії	25.01.2024	Виконано
2	Огляд сфер застосування	12.02.2024	Виконано
3	Аналіз конкурентів на ринку	18.02.2024	Виконано
4	Вибір потрібних компонентів для розробки	25.02.2024	Виконано
5	Вибір середовища програмування	18.03.2024	Виконано
6	Розробка апаратної частини системи	28.03.2024	Виконано
7	Тестування роботи	02.06.2024	Виконано
8	Оформлення звіту з виконаної роботи	05.06.2024	Виконано

Дата видачі завдання 25.01.2024

Студент _____ Калюжний М.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ проф. Новосєлов С. П.
(підпис) (посада, прізвище, ініціали

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

"16" червня 2024 р Калюжний М.В.

РЕФЕРАТ

Пояснювальна записка містить: 71 с., 35 рис., 2 дод., 15 джерел.

ВІДОБРАЖЕННЯ, ДАТЧИКИ, ПРОГРАМНИЙ МОДУЛЬ, ПОЗИЦІОНУВАННЯ, ТЕЛЕМЕТРІЯ.

Мета роботи – створення програмного модуля для моніторингу та відображення телеметричних даних з датчиків позиціонування мобільного робота.

Об'єкт розробки – процес збору та візуалізації телеметрії з датчиків мобільного робота.

Предмет розробки – програмний модуль для відображення телеметричних даних положення, швидкості, стану живлення мобільного робота.

Проведено аналіз існуючих систем та розглянуто приклад побудови інтерфейсу моніторингу в LabVIEW.

Проведено розробку програмного забезпечення та здійснено налагодження та тестування роботи програмного модуля.

ABSTRACT

Explanatory note contains: 71 p., 35 fig., 2 app., 15 sources.

DISPLAY, POSITIONING, SENSORS, SOFTWARE MODULE, POSITIONING, TELEMETRY

The aim of the work is to create a software module for monitoring and displaying telemetry data from positioning sensors of a mobile robot.

The object of development is the process of collecting and visualizing telemetry from sensors of a mobile robot.

The subject of development is a system of automated microclimate control to improve the efficiency of working conditions and ensure comfortable living.

An analysis of existing remote control systems was conducted, and an example of building a monitoring interface in LabVIEW was considered.

Software development was carried out, and debugging and testing of the software module operation were performed.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АЦП – аналого-цифровий перетворювач;

ПЛК – програмований логічний контролер;

ПЗ – програмний засіб;

ПК – персональний комп’ютер.

API – Application Programming Interface;

IoT – Internet Of Things;

IIoT – Industrial Internet Of Things;

DTA – Digital Twin Aggregate;

DTI – Digital Twin Instance;

DTP – Digital Twin Prototypes;

PLC – Programmable logic controller;

PLM – Product Lifecycle Management;

ВСТУП

Все більше підприємств автоматизують фізичні процеси та інтегрують робототехніку, щоб оптимізувати роботу, максимізувати ефективність і зменшити витрати на робочу силу.

Автономні роботи стають все більш досконалими та здатними на неймовірні речі. Але робот, який не виконує свою роботу належним чином, не буде генерувати рентабельність інвестицій для свого бізнес-користувача. Це може навіть завдати репутаційної шкоди компанії. Тому вкрай важливо переконатися, що він працює правильно.

Таким чином моніторинг стану мобільного робота є вирішальним у багатьох галузях промисловості – від роботів-прибиральників до безпілотних літальних апаратів для перевірки об’єктів, безпілотних вантажівок до автономних підводних човнів. Система відображення телеметрії допомагає операторам отримувати дані в режимі реального часу, необхідні для ефективного керування мобільними роботами.

Засоби телеметрії дозволяють здійснювати моніторинг стану мобільних роботів в реальному часі. Інформація від датчиків щодо різних параметрів, таких як температура, вологість, тиск, швидкість руху і багато інших, збирається і передається оператору в реальному часі. Це дозволяє оперативно реагувати на будь-які зміни в стані робота і вживати необхідні заходи для попередження можливих аварій або збоїв.

Аналіз даних, зібраних від датчиків, дозволяє виявляти тенденції в зношуванні та відмовах обладнання, а також розробляти раціональні стратегії технічного обслуговування і ремонту. Це допомагає зменшити час простою машин і підвищити їхню доступність для виробничих процесів.

Таким чином, використання засобів телеметрії для контролю стану мобільних роботів на виробництві є надзвичайно актуальним та необхідним аспектом в сучасній промисловості. Ці технології дозволяють підвищити

ефективність, безпеку та надійність виробничих процесів, що робить їх незамінними для сучасних підприємств.

Метою роботи є вдосконалення технології управління мобільними роботами завдяки використанню засобів телеметрії для віддаленого контролю стану пристрою на основі показників від сенсорів та пристроїв позиціонування.

Об’єктом дослідження є автоматизоване управління мобільними роботами.

Предмет дослідження – програмні засоби для збирання та відображення даних телеметрії від автономних мобільних роботів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз існуючих систем віддаленого контролю стану робототехнічних засобів;

– розробити архітектура поєднання програмного-апаратних засобів збору та відображення телеметрії з мобільного роботу

– виконати побудову об’єктної моделі програмної системи;

– розробити структуру бази даних для зберігання отриманих даних від датчиків мобільного робота;

– розробити інтерфейс програмного засобу;

– провести експериментальні випробування розробленої програми;

– оформити пояснювальну записку згідно з рекомендаціями [1], та вимогами ДСТУ 3008:2015 [2].

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Системи віддаленого контролю стану робототехнічних засобів

Телеметрія – сукупність технічних засобів і методів вимірювання на відстані різних фізичних, технічних та інших величин у промислових, енергетичних, транспортних та інших установках та передавання визначених даних з будь-якої точки на віддалений термінал [11].

Вимірювання здійснюється за допомогою датчиків. Отримані результати автоматично передаються (у вигляді кодованих радіосигналів) каналами зв'язку на приймальні пристрої, де вони розшифровуються для наступної обробки або реєстрації.

Найчастіше використовуються датчики (перетворювачі) напруги, струму, тиску, витрати енергії, температури, опору. У типову телеметричну систему входить декілька різновидів формувачів сигналів, кожен з яких використовується для перетворення вихідного сигналу того або іншого конкретного перетворювача у стандартизований сигнал напруги від 5 до 10 В.

Система телеметрії сприймає і ретранслює електричні сигнали від багатьох датчиків одночасно завдяки мультиплексуванню – процесу ущільнення даних.

У міжнародному стандарті Міжвідомчої комісії з вимірювальних засобів IRIG прийнято три способи ущільнення даних: амплітудно-імпульсна модуляція (АІМ), частотна модуляція (ЧМ) і імпульсно-кодова модуляція (ІКМ). ІКМ є найбільш поширеною завдяки характерній для неї низькій імовірності похибок (менше 0,25% для будь-якого вимірювання).

При проектуванні систем дистанційного контролю і управління для промислових роботів часто виникає необхідність передачі даних про стан датчиків і модулів управління.

Здійснювати контроль мобільного робота по провідних лініях неможливо через відсутність даного каналу зв'язку або інтенсивного руху мобільних

об'єктів. У даному випадку набагато зручніше і ефективніше передавати дані про стан датчиків з використанням радіоканалу.

Телеметричний пульт керування працює за допомогою дротової або бездротової мережі зв'язку, що взаємодіє з мобільним роботом. Датчики, що встановлені на роботі, вимірюють різні параметри та перетворюють вимірювання в електричні сигнали. Ці електричні сигнали передаються через кабелі або бездротовим способом на пристрій для зчитування. Пристрій для зчитування збирає виміряні дані та передає їх у центральну систему через мережу зв’язку.

Центральна система отримує дані вимірювань від усіх телеметричних підсистем і обробляє їх для створення звітів і графіків. Ці звіти та графіки використовуються для аналізу різних параметрів і прийняття рішень щодо керування процесом. Крім того, центральна система також може надсилати команди телеметричним підсистемам для контролю певними параметрами [12].

У сфері електроніки та телекомунікацій дистанційне керування – це технологія, яка дозволяє дистанційно керувати роботизованим пристроями та автоматизованими системами.

При застосуванні телеметрії в роботизованих мобільних пристроях, дистанційне керування дозволяє проводити вимірювання параметрів пристрою на відстані, без необхідності фізично перебувати в місці, де розташовані вимірювальні пристрої. Це досягається завдяки використанню датчиків і вимірювального обладнання, підключеного до мережі зв’язку, яке віддалено надсилає дані до систем керування.

Можна розділити основні вимоги моніторингу роботів на три категорії [13].

Апаратний моніторинг використовується для того, щоб переконатися, що фізичні компоненти робота, такі як ЦП, пам’ять, жорсткий диск і датчики, функціонують належним чином.

Моніторинг додатків використовується для того, щоб переконатися, що програмне забезпечення виконує те, для чого воно призначене, і не викликає збоїв і проблем.

Збір детальної інформації в реальному часі про те, що робот робить у будь-який момент і наскільки ефективно він виконує заплановані завдання – це якісний моніторинг.

Усі ці аспекти моніторингу є важливими для отримання надійного розуміння того, чи ефективно робот виконує свої обов’язки. Надійний моніторинг також має вирішальне значення для діагностики помилок і вирішення проблем, оптимізації та розробки майбутніх оновлень і вдосконалень.

1.2 Аналіз аналогічних рішень

1.2.1 Платформа Rocos Robot Operations Platform

Платформа Rocos розроблена, щоб інформувати оператора про стан усього парку роботів у режимі реального часу з будь-якої точки світу.

Незалежно від того, створюєте ви роботизовані рішення чи використовуєте роботів у своєму бізнесі, Rocos Robot Operations Platform дає змогу підключати, контролювати та контролювати свій автопарк [14].

Rocos – централізована платформа, що з’єднує парк роботів із людьми та речами, які допомагають їм краще виконувати свою роботу. Даний інструмент надає миттєвий доступ до надійних і безпечних API, SDK і настроюваного веб-порталу. Підключення до хмари та локальної мережі реалізовано з коробки.

Платформа надає можливості для створення власних інформаційних панелей з телеметрією з наднизькою затримкою та налаштування сповіщення та діагностику.

1.2.2 Використання спеціальної інформаційної панелі для моніторингу та телеметрії

Однією з найбільших проблем під час моніторингу роботів є аналіз величезної кількості телеметричних даних, які вони генерують щодня. Дуже важливо, щоб ви могли виділити та вивести на поверхню конкретні дані, які вам потрібні в будь-якій ситуації.

Необхідні дані також можуть відрізнятися залежно від того, хто стежить за роботом. Інженер-мехатронік може захотіти з'ясувати, чому робот раптово вимикається в певний час, чому він втрачає рівновагу або чому певні об'єкти вислизають крізь рукоятку робота. Навпаки, операційний менеджер може захотіти витягнути дані, які допоможуть йому точно визначити області для вдосконалення в рамках ширшого процесу автоматизації бізнесу.

Rocos пропонує налаштування інформаційної панелі, щоб допомогти вирішити цю проблему. За допомогою наших інформаційних панелей можна витягувати потрібні дані та налаштовувати спосіб їх відображення, додаючи або видаляючи віджети, які візуалізують базові потоки телеметрії (рис. 1.1).

Якщо потрібна візуалізація чи інтеграція, недоступна в стандартних віджетах, можна створити свій власний за допомогою HTML, Javascript і CSS.

1.2.3 Перегляд минулих інцидентів зі збережених потоків даних

Незалежно від того, чи є роботи базовими чи дуже складними, час від часу виникають проблеми в їх роботі. В цей час оператор можете не дивитись на приладову панель, коли сталася критична помилка. Навіть якщо оператор бачить нештатну ситуацію на моніторі, може бути важко діагностувати першопричину проблеми в реальному часі. Можуть існувати сотні потенційних причинно-наслідкових зав'язків між процесами, ресурсами та середовищем, у якому працює робот.

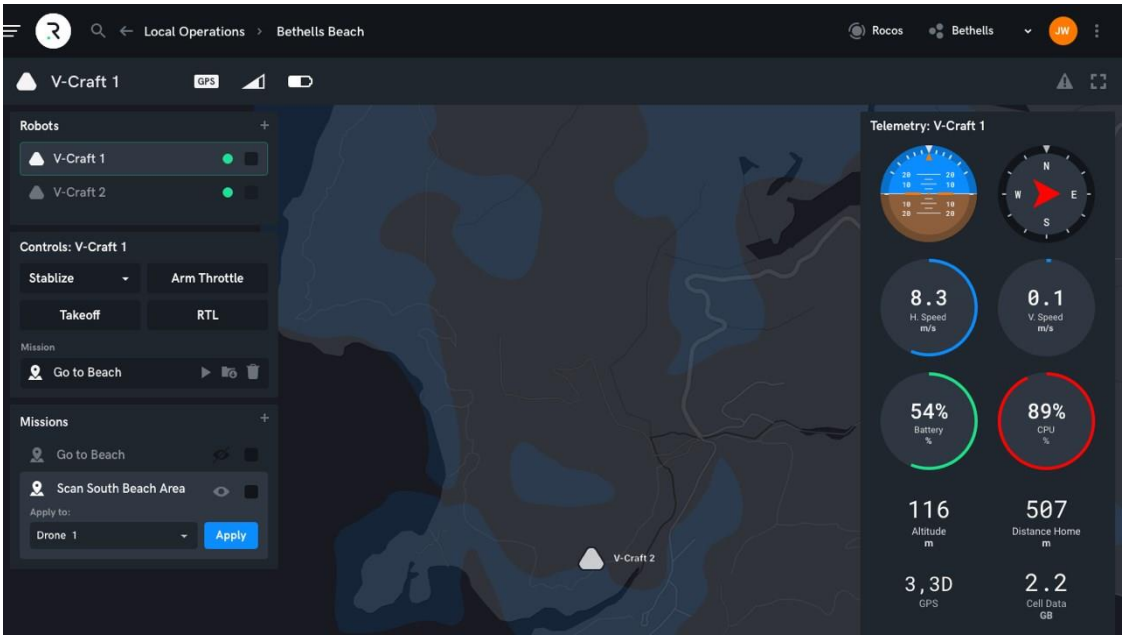


Рисунок 1.1 – Вікно відображення основних параметрів мобільного робота

Наприклад, робот зазнав раптової і, здавалося б, випадкової помилки в плануванні шляху. Цей збій повторюється щодня приблизно в один і той же час. Після тривалого дослідження можна розуміти, що світло, що проходить через вікно будівлі в певний час доби, спричиняє надмірне освітлення датчиків робота. Це значить, що робот не може точно визначити навколишнє середовище (рис 1.2).

Цей сценарій підкреслює важливість можливості аналізу історичних даних. Rocos пропонує можливість використовувати потоки зберігання – дані, які постійно збираються та зберігаються для подальшого аналізу.

Рисунок 1.2 – Перегляд минулих інцидентів зі збережених потоків даних

Коли трапиться збій, оператор може повернутися до того моменту часу та заглибитися у відповідну телеметрію, щоб зрозуміти, що сталося. Також можна порівнювати історичні тенденції та потоки даних у різних роботах і контекстах, щоб виявити важливі закономірності.

1.2.4 Відображення подій на телеметричній платформі Rocos

Щоб ефективно контролювати систему в реальному часі, потрібно повідомляти операторові, коли виникають термінові чи критичні проблеми.

На платформі Rocos подія складається з тригера та дії (рис. 1.3) [14]. Тригер визначає певний аспект телеметрії, і коли він перетинає певне порогове значення, він викликає подію. Це призводить до дії, якою зазвичай є вебхук або кінцева точка API, яка звертається до системи, щоб повідомити когось.

Наприклад, тригером може бути рівень заряду батареї робота, який впав нижче 20%. Коли це відбувається, дією може бути надсилання автоматичного SMS або повідомлення Slack оператору робота.

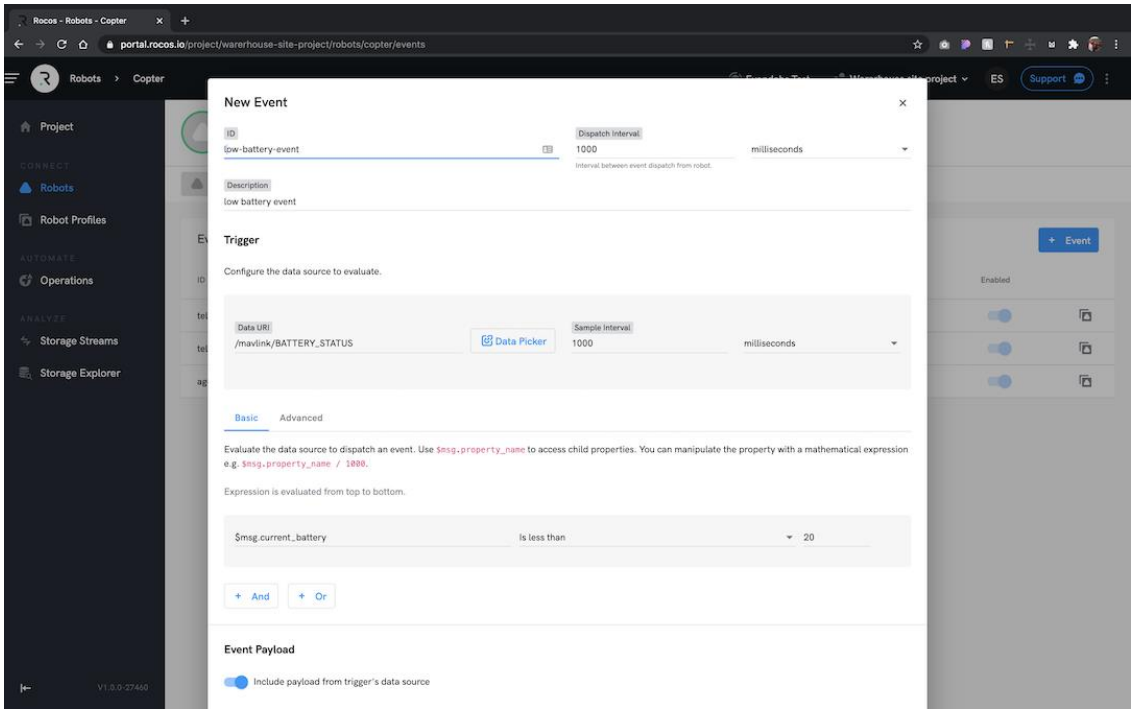


Рисунок 1.3 – Відображення подій на телеметричній платформі Rocos

Важливо, що також можна використовувати платформу Rocos, щоб надіслати команду назад до робота. Наприклад, якщо рівень заряду батареї падає нижче визначеного порогу, це може викликати команду для робота повернутися до зарядної станції. Або якщо робот досягає певної маршрутної точки GPS, платформа може наказати роботу залишатися нерухомим, а також надіслати сповіщення людині-охоронцеві, повідомляючи їм, що роботові потрібен дозвіл перед входом у наступну зону.

1.2.5 Відображення навколишнього середовища в реальному часі

Іншою важливою можливістю моніторингу, яку надає Rocos, є перегляд локальних і глобальних операцій. На глобальному рівні використовується GPS-карти, щоб у режимі реального часу показувати, де знаходяться всі роботи. На місцевому рівні надається 3D-візуалізацію того, що можуть бачити роботи (рис 1.4). Це дає вашим операторам більше інформації про оточення робота, щоб вони могли втручатися або адаптувати місії за потреби.

Рисунок 1.4 – Відображення місцевості

Цього можна досягти, імпортувавши 3D-моделі середовища (наприклад, комунального підприємства, складу чи будівельного майданчика) на телеметричну платформу, а потім використовуючи живу телеметрію від робота, точно відобразити його положення в цьому цифровому близнюку. Або якщо робот має сканер хмари точок, можна відтворити ці живі дані як реалістичне 3D-середовище.

Наприклад, якщо використовується керування автономними вантажними майданчиками, можна переглядати місцезнаходження всіх безпілотних вантажівок у будь-який момент на локальному чи глобальному рівнях. Можна створити 3D-візуалізацію кожного вантажного двору, що полегшить картографування маршрутів і оптимізацію розташування контейнерів і місця для зберігання. Також можна включити потоки з інших джерел, як-от системи відеоспостереження, на свою інформаційну панель і використовувати це для моніторингу таких речей, як безпека працівників на місці (рис. 1.5).

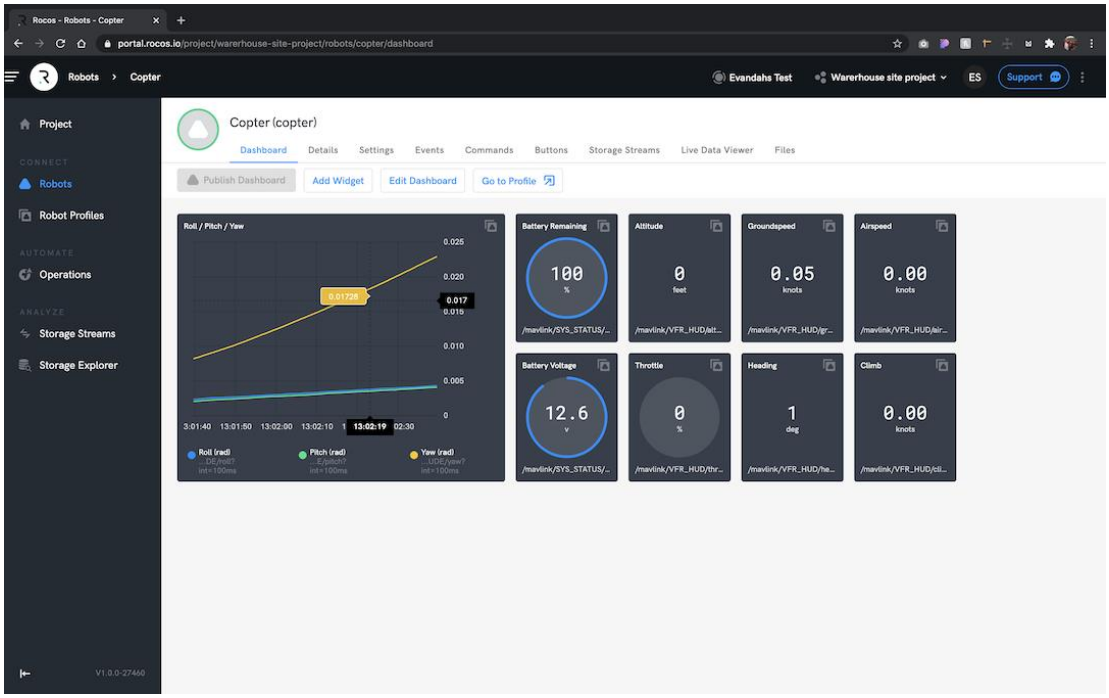


Рисунок 1.5 – Інформаційна панель з телеметрією

1.3 Приклад побудови інтерфейсу для моніторингу стану мобільного робота за допомогою LabView

Програма Dashboard, встановлена та запущена FRC Driver Station, є програмою LabVIEW, призначеною для надання командам базового зворотного зв’язку від мобільних роботів із можливістю розширення та налаштування інформації відповідно до майбутніх потреб [15]. Ця програма використовує NetworkTables і містить різноманітні інструменти, які можуть бути корисними для команд.

Інформаційна панель розділена на два основні розділи (рис. 1.6). Ліва панель призначена для відображення зображення камери. Права панель містить індикатори про стан основних компонентів мобільного роботу які оформлені у вигляді вкладок з набором необхідних графічних компонентів:

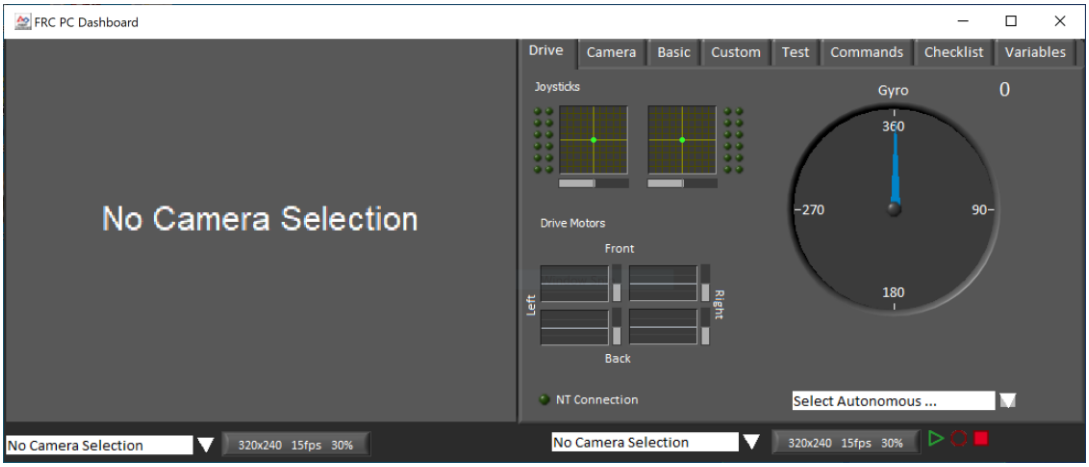


Рисунок 1.6 – Зовнішній вигляд інтерфейсу моніторингу стану мобільного робота

– вкладка Drive містить індикатори значень джойстика та двигуна приводу (підключено за замовчуванням, якщо використовується з кодом робота LabVIEW), індикатор гіроскопа, текстове поле автономного вибору, індикатор підключення та деякі елементи керування та індикатори для камери;

– основна вкладка містить деякі елементи керування та індикатори за замовчуванням;

– вкладка Камера містить засіб перегляду додаткової камери, подібний до засобу перегляду на лівій панелі;

– спеціальна вкладка для налаштування інформаційної панелі за допомогою LabVIEW;

– вкладка Тест для використання з тестовим режимом у середовищі LabVIEW;

– вкладка Команди для використання з новою системою C&C у LabVIEW;

– вкладка контрольного списку, яку можна використовувати для створення списків завдань для виконання мобільними роботами;

– вкладка Змінні, яка відображає необроблені змінні NetworkTables у форматі перегляду дерева.

Інформаційна панель LabVIEW також містить функцію запису/відтворення, розташовану внизу праворуч.

Окрема область зліва використовується для виведення зображення з відео камери. Призначення органів управління показано на рис 1.7.

Рисунок .1.7 – Управління відеокамерою

Під областю вкладок також є елементи керування та індикатори, пов’язані з камерою:

– відображення зображення камери (1);

– вибір режиму - це спадне меню дозволяє вибрати тип дисплея камери для використання. Вибір: Camera Off (Камера вимкнена), USB Camera SW (програмне стиснення), USB Camera HW (апаратне стиснення) та IP-камера (камера Axis). Зауважте, що налаштування IP-камери не працюватиме, якщо ваш ПК підключено до roboRIO через USB (2);

– параметри камери. Цей елемент керування дозволяє змінювати роздільну здатність, частоту кадрів і стиснення потоку зображення на інформаційній

панелі. Натисніть цей елемент керування, щоб відкрити спливаюче вікно конфігурації (3);

– індикатор пропускної здатності – вказує на приблизне використання пропускної здатності потоку зображень. Індикатор відображатиметься зеленим кольором для «безпечного» використання пропускної здатності, жовтим, коли командам слід бути обережними, і червоним, якщо пропускна здатність потоку виходить за межі рівня, який буде працювати на змагальному полі (4);

– частота кадрів вказує на приблизну отриману частоту кадрів потоку зображення (5).

Елементи інтерфейсу моніторингу стану мобільного роботу показано на рис 1.8.

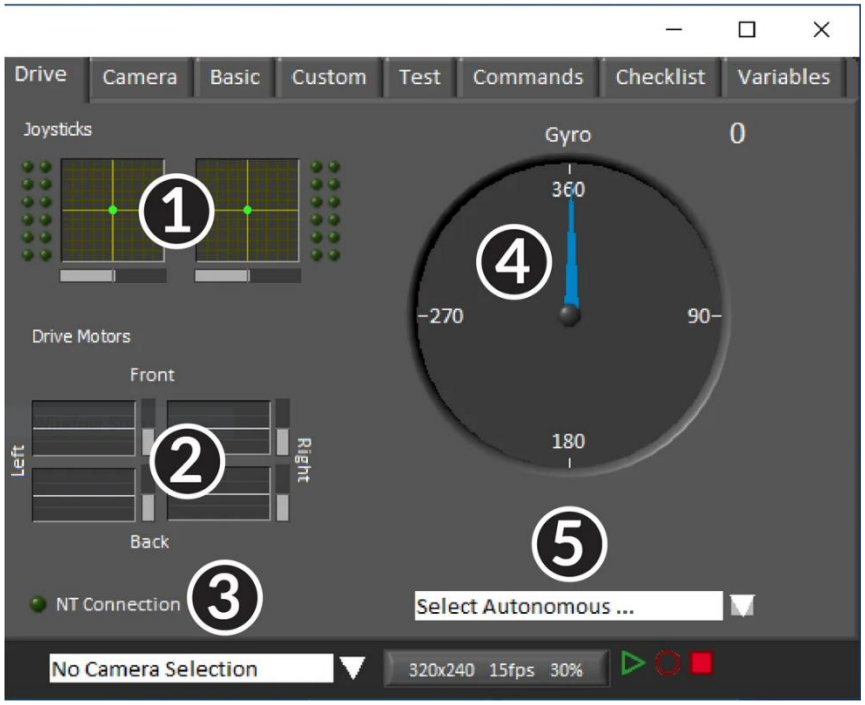


Рисунок 1.8 – Елементи інтерфейсу моніторингу стану мобільного роботу

Центральна панель містить розділ, який надає зворотній зв’язок щодо джойстиків і команд приводу при використанні з інфраструктурою LabVIEW, і розділ, який відображає статус NetworkTables і автономний селектор:

– віджет 1 відображає інформацію про X, Y і дросель, а також значення кнопок для максимум 2 джойстиків під час використання середовища LabVIEW;

– віджет 2 відображає значення, що надсилаються до контролерів двигунів під час використання середовища LabVIEW;

– віджет 3 відображає індикатор підключення для даних NetworkTables від робота;

– віджет 4 відображає значення гіроскопа;

– віджет 5 відображає текстове поле, яке можна використовувати для вибору автономних режимів. У шаблонах коду кожної мови є приклади використання цього поля для вибору з кількох автономних програм.

Ці індикатори (окрім гіроскопа) підключаються до відповідних значень за замовчуванням під час використання середовища LabVIEW.

1.4 Постановка задачі

Проведений аналіз літератури та аналогічних рішень показав актуальність впровадження телеметричної системи збору показань з датчиків мобільних роботів. Система відображення телеметрії допомагає операторам отримувати дані в режимі реального часу, необхідні для ефективного керування мобільними роботами.

В подальшій роботі потрібно зробити наступне:

– розробити архітектура поєднання програмного-апаратних засобів збору та відображення телеметрії з мобільного роботу

– виконати побудову об’єктної моделі програмної системи;

– розробити структуру бази даних для зберігання отриманих даних від датчиків мобільного робота;

– розробити інтерфейс програмного засобу;

– провести експериментальні випробування розробленої програми.

2 ОПИС ПРИНЦИПУ РОБОТИ АВТОМАТИЗОВАНОЇ СИСТЕМИ

2.1 Розробка структурної схеми поєднання компонентів системи

Архітектура автоматизованої системи відображення даних телеметрії показана на рис 2.1.

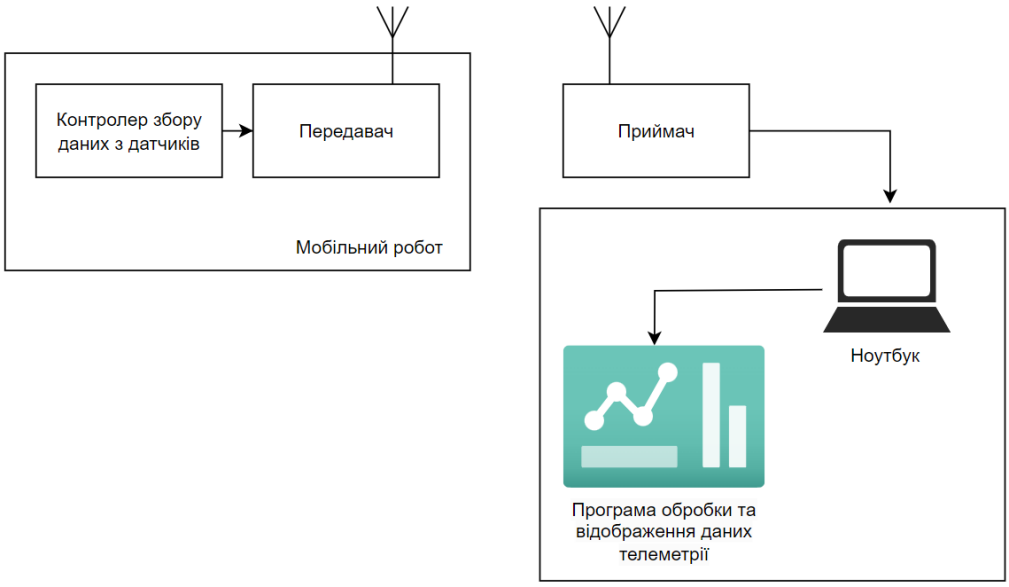


Рисунок 2.1 – Архітектура автоматизованої системи відображення даних телеметрії

Автоматизована система поєднує між собою контролер мобільного робота та ноутбук, або персональний комп’ютер за допомогою бездротової технології передавання даних. Для цього використовуються передавач та приймач, що можуть працювати в одному з дозволених діапазонів радіочастот, наприклад в діапазоне 433 МГц.

Задача контролера мобільного робота – зібрати показники з наявних датчиків, що розміщенні на борту роботизованої платформи та передати їх через послідовний протокол зв’язку на бік ноутбуку.

Задача програми, що завантажена на ноутбуці – прийняти дані за певним протоколом, обробити отримані дані та відобразити їх у зручному для оброки оператором вигляді. Інша задача – це зберігання отриманих даних в базі для подальшого використання.

Структурна схема підсистеми збору даних з датчиків мобільного робота показана на рис 2.2.

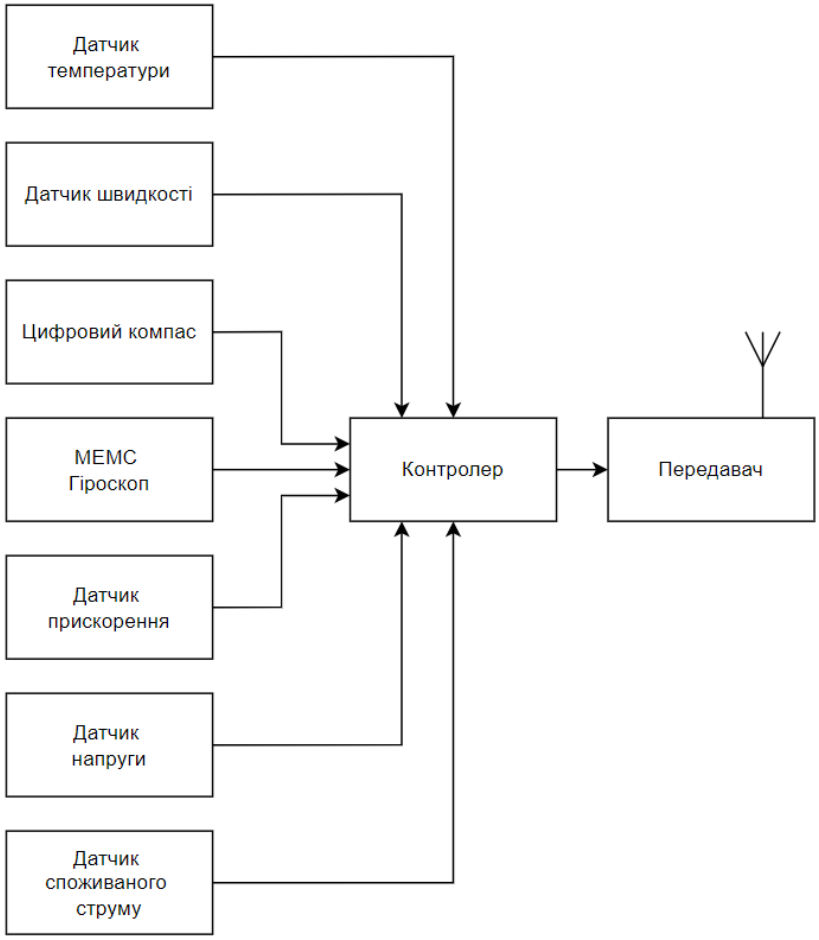


Рисунок 2.2 – Структурна схема підсистеми збору даних з датчиків мобільного робота

Контролер мобільного робота збирає дані з наступних сенсорів:

– датчик температури відсіку з акумуляторами;

– датчик швидкості руху мобільної платформи;

– цифровий компас, що використовується для орієнтації на місцевості та визначення поточного напрямку руху;

– МЕМС гіроскоп для визначення куту повороту в просторі;

– датчик прискорення;

– датчик напруги бортової мережі;

– датчик споживаного струму компонентами мобільної роботизованої платформи.

Контролер збирає дані безпосередньо з датчиків, що підключені до його контактів, або отримує інформацію через внутрішню послідовну мережу зв'язку, наприклад з використанням інтерфейсу RS-485 та протоколу modbus.

Отримані дані долучаються до загального інформаційного пакету та передаються засобами радіозв'язку до підсистеми візуалізації даних. На рис 2.3 показана структура пакету даних, що передається від мобільного робота до підсистеми відображення телеметрії.

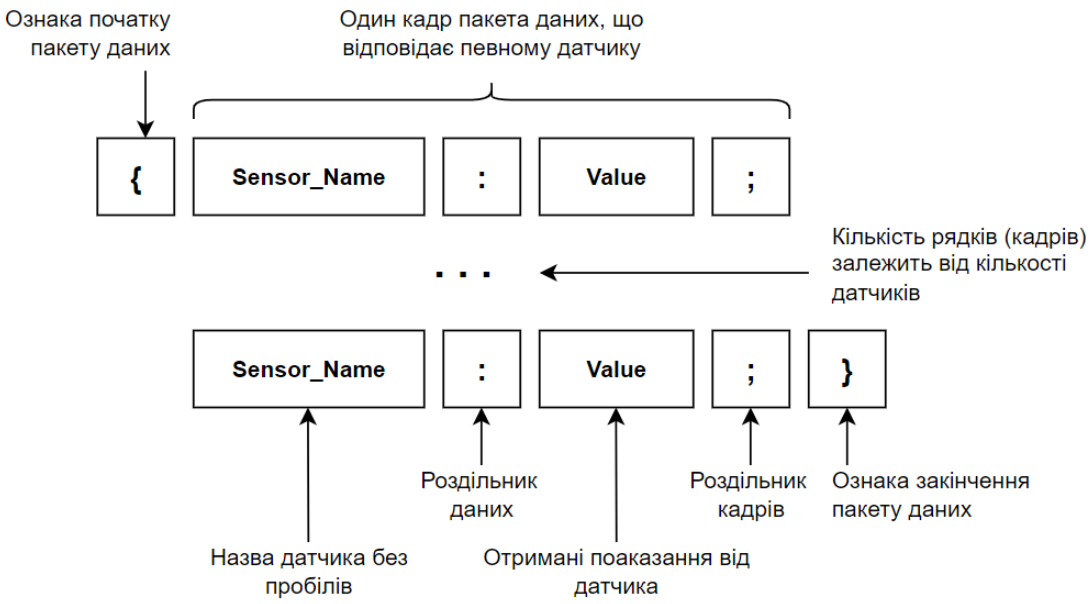


Рисунок 2.3 – Структура пакету даних

Пакет даних являє собою текстовий рядок в кодуванні ASCII певної структури. Початок пакту даних характеризується наявністю символу {. Ознакою кінця є символ }.

Між фігурними дужками знаходяться кадри пакету даних. Кожен кадр відповідає певному датчику, що надав дані для відображення телеметрії. Вміст кадру настуаний:

– назва датчика;

– розподіляючий символ :;

– виміряне значення;

– символ завершення кадру ;.

Кадр пакету даних починається з найменування датчика, наприклад SpeedSensor_1 (перший датчик швидкості), або TempSensor_3 (третій датчик температури). В назві датчика не можна використовувати пробіл.

Після назви датчика ставиться розподільник у вигляді символу ASCII :. Цей символ відділяє назву від значення.

Отримане від датчика значення наводиться у вигляді текстового рядку. Після отримання кадру на боці приймача не обхідно виконати зворотне конвертування тексту в числове значення для правильного відображення даних телеметрії.

Завершується кожен кадр символом ;. Таким чином, всі отримані дані від всіх наявних датчиків складають текстовий рядок розділений крапкою з комою для кожного окремого датчика.

Наприкінці пакету використовується символ }, що є ознакою припинення прийняття та обробки інформації з віддаленого мобільного роботизованого пристрою.

Структура підсистеми візуалізації даних телеметрії від мобільного робота показана на рис 2.4.

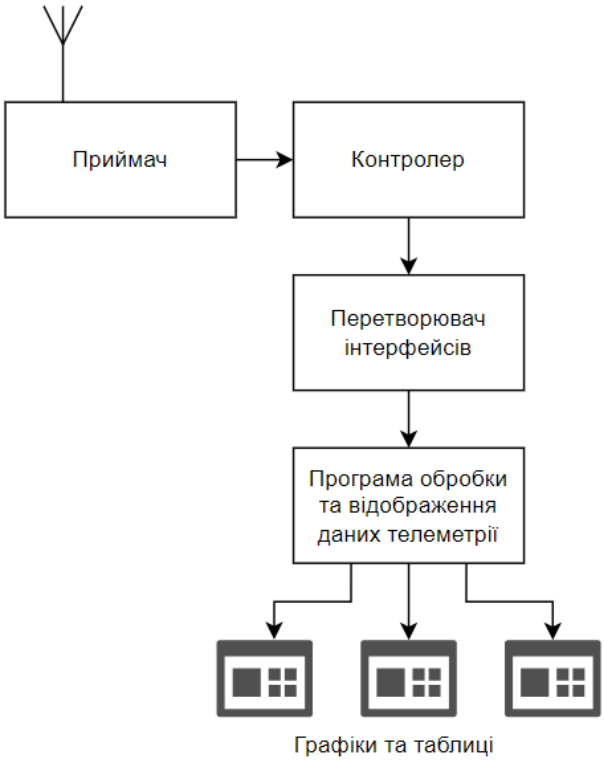


Рисунок 2.4 – Структура підсистеми візуалізації даних телеметрії

Для виконання ноутбуком функції відображення даних телеметрії необхідно попередньо прийняти дані від мобільного робота. Дані передаються через радіоефір, тому необхідно застосовувати радіоприймач, що працює в заданому діапазоні.

Приймач може бути підключений до контролера для обробки прийнятих даних та формування корисного навантаження для передачі на вхід програмного засобу. Для поєднання з програмним засобом використовуються спеціальні пристрої – конвертори інтерфейсів.

Після отримання даних програма відображає інформацію у вигляді графіків, трендів, або в табличному чи числовому форматі.

2.2 Розробка алгоритму роботи програми

На рис 2.5 показано схема алгоритму роботи програми в режимі отримання даних від мобільного роботизованого пристрою.

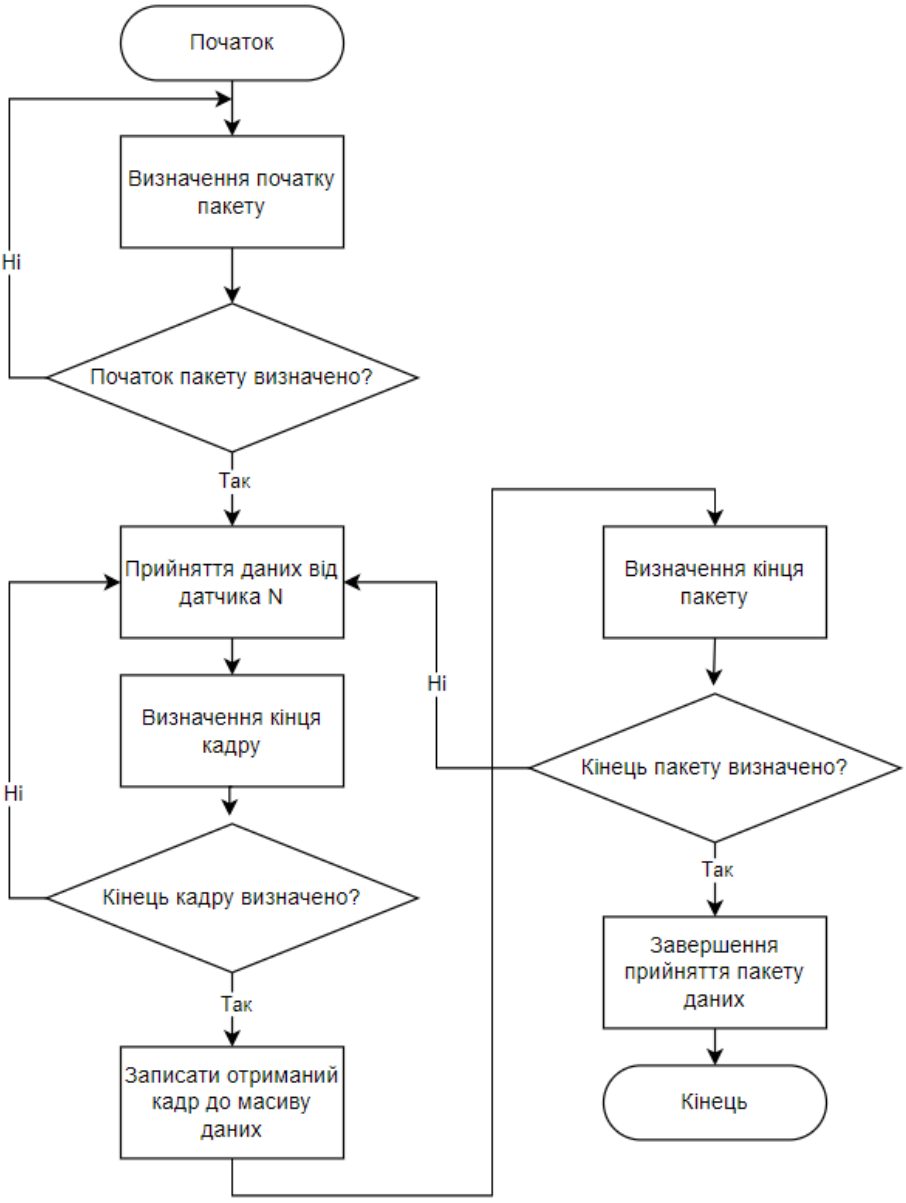


Рисунок 2.5 – Схема алгоритму роботи програми в режимі отримання даних від мобільного роботизованого пристрою

На початку роботи програма перевіряє наявність даних в буфері обміну. Якщо такі дані будуть виявлені, то перевіряється наявність спеціального символу початку пакету {.

В разі, якщо даних в буфері не виявлено, або на початку отриманого рядку тексту немає спеціального символу, програма повертається на початок головного циклу.

Після виявлення символу початку програма зчитує з буферу обміну цілий кадр до розділяючого символу ;.

Далі, отриманий кадр зберігається в спеціальну змінну Cadr та викликається підпрограма для аналізу прийнятої інформації та заповнення змінних.

Наступним кроком виконується перевірка на наявність символу кінця пакету даних }. Якщо такого символу не знайдено, то виконується нова ітерація циклу прийняття чергового кадру та його аналізу за допомогою відповідної підпрограми.

Після отримання символу, що означає кінець пакету відбувається завершення прийняття даних:

– прийняті дані записуються у відповідні комірки бази даних та створюється відповідна кількість рядків запису у сховище;

– прийняті дані аналізуються та надсилаються до відповідних візуальних графічних компонентів для відображення телеметрії на екрані персонального комп’ютера, або панелі оператора.

На рис 2.6 наведено схему алгоритму роботи підпрограми аналізу кадру даних.

Підпрограма аналізу даних розбиває кадр на дві частини за допомогою оператору Split. Ознакою частин є символ :, що розділяє команду і значення.

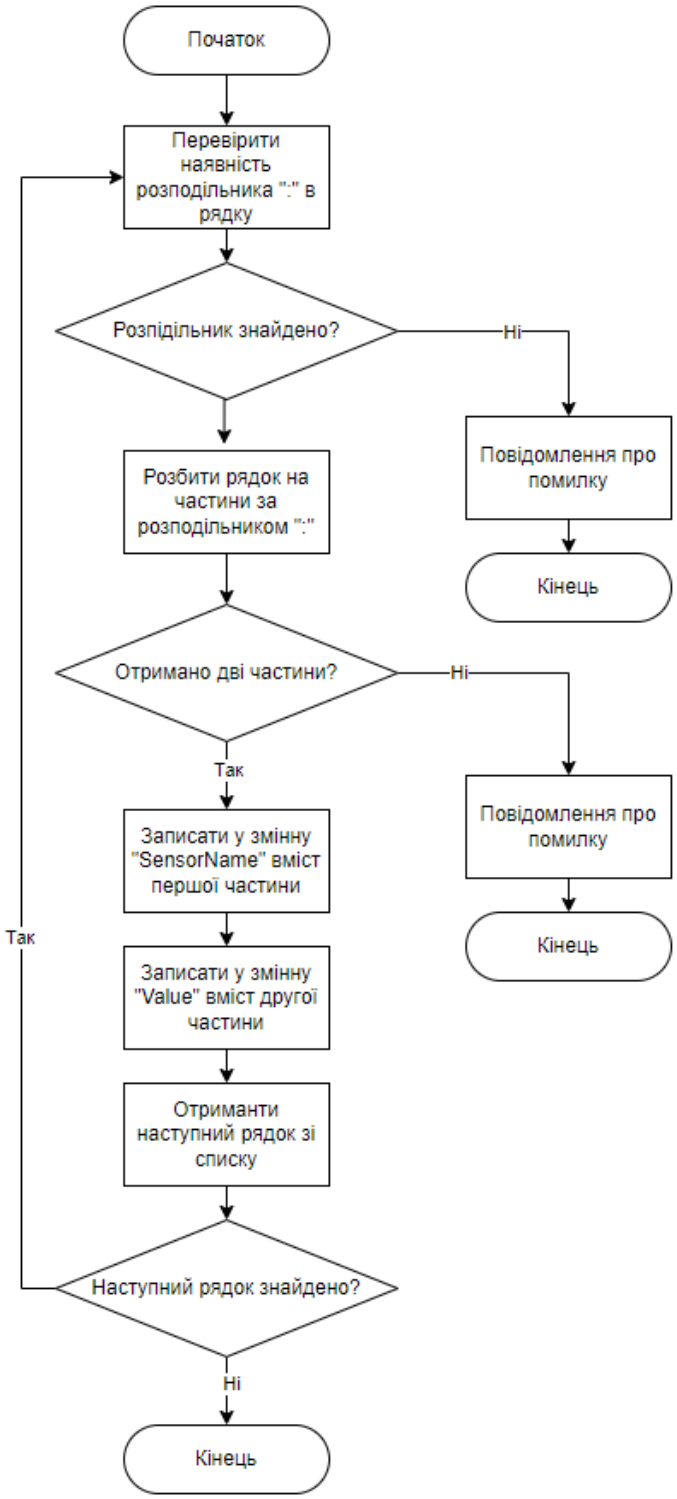


Рисунок 2.6 – Схема алгоритму роботи підпрограми аналізу кадру даних

Після розбиття текстового рядку на частини обов’язково перевіряється кількість отриманих частин. Якщо частин менше або більше двох, виводиться повідомлення про помилку та робота підпрограми завершується.

Також помилка виводиться, якщо в текстовому рядку, що надійшов на аналіз взагалі не буде знайдено роздільника.

Перша частина кадру заноситься до змінної SensorName, друга частина – до змінної Value. Сформовані змінні далі надсилаються до бази даних та на графічний інтерфейс для відображення користувачеві.

3 ВИБІР КОМПОНЕНТІВ АВТОМАТИЗОВАНОЇ СИСТЕМИ

3.1 Вибір датчика швидкості

Датчик швидкості – це комбінований елемент, що являє собою диск з прорізами (таходиск) та фотодатчик для зчитування світлових імпульсів від таходиска. Зовнішній вигляд таходиска показано на рис 3.1.

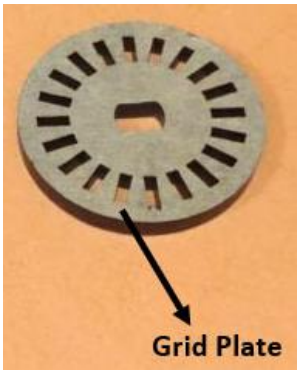


Рисунок 3.1 – Зовнішній вигляд таходиска

На рис 3.2 показано зовнішній вигляд оптичного датчика для зчитування сигналів з таходиску.

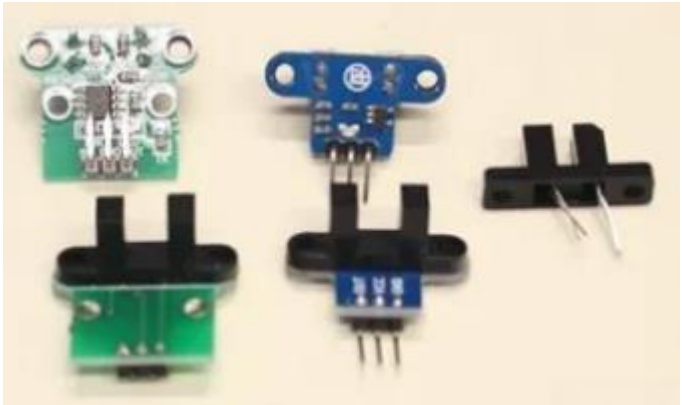


Рисунок 3.2 – Зовнішній вигляд оптичного датчика для зчитування сигналів з таходиску

На рис 3.3 показано датчик швидкості у зібраному стані.

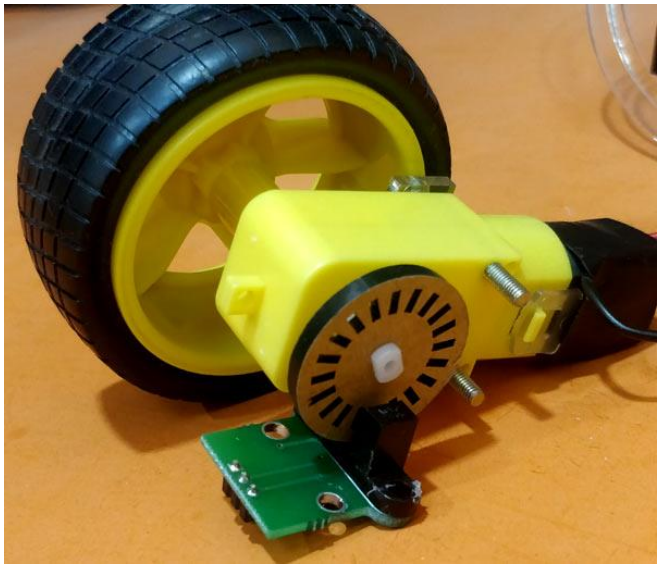


Рисунок 3.3 – Датчик швидкості у зібраному стані

Модуль датчика швидкості складається з датчика інфрачервоного світла, інтегрованого з мікросхемою компаратора напруги LM393. Модуль також складається з перфорованого диску, який повинен бути встановлений на обертовому валу двигуна.

Датчик інфрачервоного світла складається з ІЧ-світлодіода та фототранзистора, розділених невеликою перетинкою. Уся система датчиків розміщена в чорному корпусі, як показано вище.

Таходиск розташовано між проміжками датчика інфрачервоного світла таким чином, щоб датчик міг визначити проміжки в сітчастому диску. Кожна щілина в диску забезпечує спрацювання ІЧ-датчик при проходженні променю крізь щілину. Потім ці імпульси світла перетворюються в сигнали напруги за допомогою компаратора. Компаратор – це мікросхема LM393 від ON semiconductors. Модуль має три контакти, два з яких використовуються для живлення модуля, а один вихідний контакт використовується для підрахунку кількості тригерів.

Нижче наведено фрагмент коду для контролеру Arduino для визначення швидкості обертання коліс мобільного робота. По-перше, виконується ініціалізація всіх змінних:

```
const int sensorPin = 2;  // Пін, до якого підключений фотодатчик
volatile unsigned long pulseCount = 0;
unsigned long previousMillis = 0;
const unsigned long interval = 1000;  // Інтервал вимірювання у мілісекундах
const int pulsesPerRevolution = 20;  // Кількість імпульсів на один оберт
```

Далі виконується налаштування швидкості передавання даних через послідовний інтерфейс та первинні налаштування портів контролеру:

```
void setup() {
  Serial.begin(115200);
  pinMode(sensorPin, INPUT);
  attachInterrupt(digitalPinToInterrupt(sensorPin), countPulse, RISING);
}
```

В основному циклі виконується підрахунок числи обертів колеса за хвилину, та передавання даних до програми відображення телеметрії:

```
void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // Обчислюємо RPM
    float rpm = (pulseCount / (float)pulsesPerRevolution) * (60000.0 / interval);
```

```
    Serial.print("RPM: ");
    Serial.println(rpm);

    // Очищуємо лічильник імпульсів
    pulseCount = 0;
  }
}
```

Кожні 1000 мілісекунд обчислюється RPM за формулою:

$$RPM = \left(\frac{P_n}{N}\right) \times \left(\frac{60000}{T}\right), \quad (3.1)$$

де P_n– кількість імпульсів за визначений інтервал;

N – кількість імпульсів на оберт;

T – розмір інтервалу часу в мс.

Визначений результат виводиться у послідовний порт, а лічильник імпульсів очищується для наступного інтервалу вимірювання.

Функція переривання (countPulse) збільшує лічильник імпульсів щоразу, коли фотодатчик фіксує імпульс:

```
void countPulse() {
  pulseCount++;
}
```

3.2 Вибір МЕМС комбінованого датчика для визначення положення мобільного робота

Для визначення положення мобільного робота в просторі використовується комбінований МЕМС модуль GY-86 (рис. 3.4).

Рисунок 3.4 – Зовнішній вигляд комбінованого модуля GY-86

Модуль GY-86 – це набір датчиків по 9 осях для інерційної навігаційної системи (IMU) + барометр MS5611. До складу модуля входять:

– MPU6050 (3-х осьовий гіроскоп + 3-х осьовий акселерометр);

– HMC5883L (електронний 3-х осьовий компас);

– MS5611 (прецизійний датчик абсолютного атмосферного тиску).

Характеристики модуля:

– інтерфейс I2C (TWI)

– напруга живлення: +3.3 ~ +5 В

– вбудований конвертер логічних рівнів (сумісний з 3.3/5.0 В логікою).

Акселерометр – це пристрій, який вимірює правильне прискорення («силу перевантаження»). Власне прискорення не те саме, що координатне прискорення (швидкість зміни швидкості). Наприклад, акселерометр у стані спокою на поверхні Землі вимірює прискорення g= 9,81 м/с2 прямо вгору. Навпаки,

акселерометри у вільному падінні, що обертаються та прискорюються завдяки гравітації Землі, вимірюватимуть нуль.

Недолік акселерометра полягає в тому, що він досить точний і не дрейфує, але неможливо виміряти зміни навколо осі Y. Це відбувається тому, що сила тяжіння не змінюється під час вертикального обертання.

D GY-86 використовується акселерометр MPU6050 від Analog Devices. Він вимірює прискорення для всіх трьох осей (x, y, z) і має роздільну здатність до 13 біт (виявляє зміни менше 1,0°).

Мікросхема зазвичай повертає оцифровані значення датчика з роздільною здатністю 10 біт. Щоб мати можливість далі працювати з цими даними, значення потрібно перетворити на загальну одиницю, як-от G.

Цифровий компа HMC5883L побудовано на основі анізотропної магніторезистивної (AMR) технології, яка забезпечує точне вимірювання магнітних полів. Він має тривісні датчики з полями в діапазоні від ±1,3 до ±8,1 Гауса. Це означає, що він може вимірювати магнітні поля за трьома різними осями (X, Y і Z), що дає йому можливість відчути напрямок магнітного поля Землі в трьох вимірах.

HMC5883L поставляється з кількома вбудованими функціями, які спрощують інтеграцію в різні системи. Вони включають вбудований АЦП (аналогово-цифровий перетворювач), який забезпечує виведення 12-бітних даних, і інтерфейс послідовної шини I2C. Цей інтерфейс використовується для зв’язку з пристроєм, що дозволяє йому надсилати дані на мікроконтролер або інший блок обробки.

В обраному комбінованому модулі до мікросхеми магнітометра HMC5883L додано мікросхеми регулятора напруги, резистори і конденсатори в інтегральній схемі. Використовується стабілізатор напруги IC XC6206P332MR.

Робочі характеристики датчика HMC5883L:

– робоча напруга: від 3 В до 6 В постійного струму;

– робочий струм: 100-130 мкА;

– інтерфейс I2C;

– точність курсу 1-2 градуси;

– інтегрований 12-розрядний АЦП;

– максимальна швидкість передачі даних 160 Гц;

– діапазон від -8 до +8 Гаусса;

– швидкість виведення даних: 0,75, 1,5, 3, 7,5, 15 Гц.

3.3 Вибір датчика струму та напруги живлення

В якості датчика струму і напруги використовується інтегрований модуль CJMCU-219. Даний модуль призначений для вимірювання таких параметрів постійного струму як напруга, струм і споживана потужність. Модуль виконаний на мікросхемі INA219 – вимірювач струму та напруги з нульовим дрейфом і володіє малими розмірами і вагою при дуже великих можливостях та високою точністю вимірювань. Зовнішній вигляд модуля CJMCU-219 показано на рис 3.5.

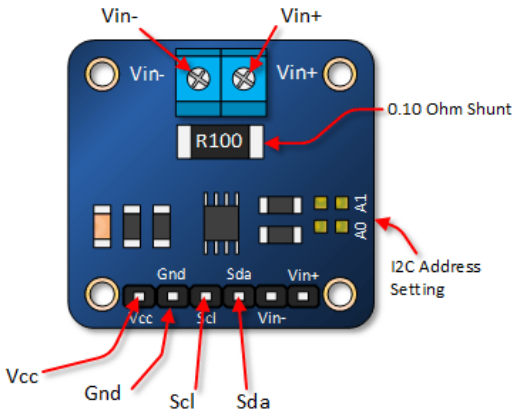


Рисунок 3.5 – Зовнішній вигляд модуля CJMCU-219

Мікросхема вимірює параметри протікання струму в будь-якому напрямку з автоматичним перемиканням полярності вимірювання. Застосувати модуль можна в системах, які контролюють процес заряду/розряду акумуляторних батарей, джерелах живлення з контролем напруги і споживаного навантаженням струму. Можливості зміни I2C адреси вимірювача дозволяють підключити на одну шину до 4-х таких пристроїв. Для збільшення точності вимірювань передбачений регістр калібрування.

Характеристики модуля:

– робоча температура: від -40С до 85С;

– дрейф в робочому температурному діапазоні: 100мкВ;

– максимальний вимірюваний струм: 3,2А;

– точність вимірювання струму: 0,8мА;

– роздільна здатність вимірювача: 12-біт;

– інтерфейс: I2C;

– швидкість інтерфейсу: 3,4МГц;

– максимальна вимірювана напруга: + -26 В;

– напруга живлення: від 3В до 5В.

Нижче показано фрагмент програми для визначення напруги, струму та споживаної потужності. Програма забезпечує постійний моніторинг параметрів електричної системи, що підключена до сенсора INA219, і виводить ці дані в зручному для аналізу вигляді на серійний порт:

```
#include <Wire.h>
Adafruit_INA219 sensor219; // Declare and instance of INA219
void setup(void) {
  Serial.begin(9600);
  sensor219.begin();  }

void loop(void) {
  float busVoltage = 0;
```

```
 float current = 0; // Measure in milli amps
 float power = 0;
 busVoltage = sensor219.getBusVoltage_V();
 current = sensor219.getCurrent_mA();
 power = busVoltage * (current/1000); // Calculate the Power
 Serial.print("Bus Voltage:   ");
 Serial.print(busVoltage);
 Serial.println(" V");
 Serial.print("Current:       ");
 Serial.print(current);
 Serial.println(" mA");
 Serial.print("Power:         ");
 Serial.print(power);
 Serial.println(" W");
 Serial.println("");
 delay(2000);
}
```

На початку програми підключається бібліотека Wire, необхідна для роботи з шиною I2C, а також створюється екземпляр об'єкта sensor219 класу Adafruit_INA219. В функції setup() ініціалізується серійний порт для виводу даних з швидкістю 9600 бод та ініціалізується сенсор INA219 за допомогою методу begin(). Це забезпечує підготовку сенсора до роботи та готовність до вимірювань.

У функції loop(), яка виконується постійно, спочатку визначаються змінні для зберігання значень напруги, струму та потужності. Потім, за допомогою методу getBusVoltage_V(), отримується значення напруги шини в вольтах. Далі за допомогою методу getCurrent_mA() отримується значення струму в міліамперах. Після цього розраховується потужність шляхом множення напруги на струм, переведеного в ампери.

Отримані значення виводяться на серійний порт для перегляду. Зокрема, значення напруги виводиться з позначенням Bus Voltage, значення струму – з позначенням Current, а значення потужності – з позначенням Power. Кожне значення виводиться з відповідними одиницями вимірювання (вольти, міліампери, вати) для зручності читання.

Після виведення даних програма робить паузу на 2 секунди, використовуючи функцію delay(2000), перед тим як повторити процес вимірювання та виводу даних. Це дозволяє регулярно оновлювати показники та відслідковувати їх зміни в реальному часі.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Опис середовища розробника

Visual Studio 2022 (VS 2022) є новітньою версією інтегрованого середовища розробки (IDE) від Microsoft, яке широко використовується для створення різноманітного програмного забезпечення. Це середовище підтримує розробку додатків для платформ Windows, iOS, Android, веб-додатків та хмарних сервісів. Visual Studio 2022 відзначається високою продуктивністю, інтуїтивним інтерфейсом і багатим набором інструментів, що робить його популярним вибором серед професійних розробників.

Visual Studio 2022 підтримує широкий спектр мов програмування, включаючи C#, C++, JavaScript, Python, TypeScript, F#, Visual Basic, і багато інших. Це дозволяє розробникам працювати з різними технологіями в одному середовищі, що значно спрощує процес розробки багатоплатформенних додатків. Однією з ключових переваг VS 2022 є його 64-бітна архітектура, яка дозволяє використовувати більше пам’яті, ніж попередні версії. Це забезпечує кращу продуктивність при роботі з великими проектами, зменшуючи кількість випадків уповільнення та дозволяючи одночасно обробляти більше задач.

Інтерфейс користувача Visual Studio 2022 було оновлено, щоб зробити його більш сучасним і інтуїтивним. Візуальні покращення включають більш чіткі шрифти, нові іконки та більш зручну навігацію, що сприяє підвищенню продуктивності розробників. Інтелектуальні інструменти для підвищення продуктивності розробки включають IntelliCode, який використовує машинне навчання для надання рекомендацій щодо коду, що базуються на практиках кодування мільйонів відкритих проектів. Функції рефакторингу спрощують реорганізацію коду, забезпечуючи швидке і безпечне внесення змін, а Live Share дозволяє спільну роботу над кодом в реальному часі з іншими розробниками, незалежно від їхнього місцезнаходження.

Редактор коду в Visual Studio 2022 підтримує багатопоточність, що забезпечує плавну роботу навіть з великими файлами. Інструменти для форматування коду, перевірки синтаксису та автоматичного завершення коду роблять процес розробки більш ефективним і менш схильним до помилок. Крім того, Visual Studio 2022 інтегрується з Microsoft Azure, що дозволяє розробникам легко створювати, тестувати та розгортати хмарні додатки. Інструменти для управління хмарними ресурсами, такими як бази даних, сховища даних і служби штучного інтелекту, роблять процес розробки хмарних додатків більш зручним і продуктивним.

Важливою особливістю Visual Studio 2022 є підтримка контейнерів та оркестрації контейнерів через Docker та Kubernetes. Це забезпечує більш просту розробку, тестування і розгортання додатків у контейнеризованих середовищах, що значно покращує гнучкість і масштабованість додатків. Інтеграція з Git та GitHub дозволяє легко керувати вихідним кодом, відстежувати зміни та співпрацювати з іншими розробниками, що сприяє кращій організації проектів і швидшому вирішенню проблем.

4.2 Опис інтерфейсу користувача

Зовнішній вигляд інтерфейсу користувача показано на рис 4.1.

Область 1 призначена для відображення відеозображення з камери, що підключена до мобільного робота.

Віджет 2 призначений для відображення поточної швидкості руху мобільної платформи.

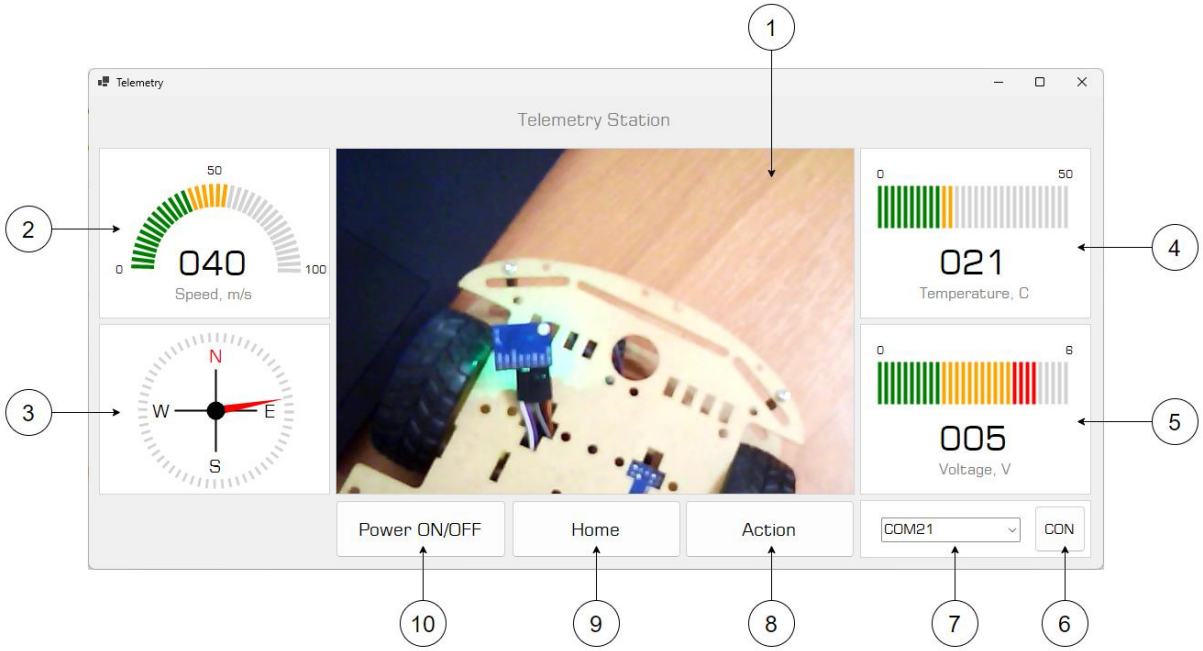


Рисунок 4.1 – Зовнішній вигляд інтерфейсу користувача

Віджет 3 – це цифровий компас і призначений для відображення поточної орієнтації мобільного пристрою в просторі.

Віджет 4 призначений для інформування оператора про поточну температуру внутрі корпусу модуля керування мобільним роботом.

Віджет 5 відображає поточне значення напруги живлення бортової мережі мобільного пристрою.

Для підключення до каналу зв’язку між робочою станцією та мобільним пристроєм використовується Bluetooth-перетворювач, або інший пристрій, що може транслювати прийняті дані до послідовного інтерфейсу ПК. З’єднання з послідовним інтерфейсом відбувається на початку роботи з програмою за допомогою кнопки CON (6). Вибір послідовного інтерфейсу здійснюється за допомогою випадаючого списку 7.

Приклад програмного коду показано на рисунку 4.2.

Коли користувач натискає кнопку, код перевіряє, чи ініціалізовано об'єкт serialPort. Якщо ні, то створюється новий екземпляр класу SerialPort. Потім перевіряється, чи не відкрито вже з'єднання.

```
private void button4_Click(object sender, EventArgs e)
{
    if (serialPort == null) serialPort = new SerialPort();

    if (!serialPort.IsOpen)
    {

        serialPort.PortName = PortName;
        if (BaudRate != "") serialPort.BaudRate = Int32.Parse(BaudRate);
        else serialPort.BaudRate = 115200;
        serialPort.Parity = Parity.None;
        serialPort.DataBits = 8;
        serialPort.StopBits = StopBits.Two;
        serialPort.Handshake = Handshake.None;
        serialPort.DtrEnable = false;
        serialPort.RtsEnable = false;

        serialPort.Open();
        serialPort.ReadTimeout = 5000;
        serialPort.WriteTimeout = 5000;

        serialPort.DataReceived += new SerialDataReceivedEventHandler(DataReceivedHandler);
    }
    else
    {
        serialPort.Close();
    }
}
```

Рисунок 4.2 – Приклад програмного підключення до послідовного інтерфейсу

Якщо з'єднання ще не відкрито, код налаштовує параметри порту: ім'я порту, швидкість передачі даних (якщо значення BaudRate не вказане, використовується значення за замовчуванням 115200), парність (відсутня), кількість біт даних (8), стопові біти (два), тип обміну (відсутній), а також вимикає управління сигналами DTR і RTS.

Після цього з'єднання відкривається, і для порту встановлюються таймаути на читання і запис (по 5000 мс). Також додається обробник події DataReceived, який викликається при отриманні даних через послідовний порт. Якщо ж з'єднання вже відкрите, воно закривається.

Кнопки 8 – 10 призначені для керування мобільним пристроєм:

– кнопка 8 ініціює роботу виконавчого пристрою;

– кнопка 9 дає команду виконавчому пристрою зайняти початкове положення;

– кнопка 10 дає команду увімкнути або вимкнути напругу живлення виконавчого пристрою.

4.3 Опис програмної реалізації віджетів графічного інтерфейсу

4.3.1 Віджет для відображення циферблату

Циферблатний дашборд, також відомий як гейдж або стрілочний дашборд, використовується для візуалізації ключових показників ефективності у вигляді аналогового індикатора, схожого на спідометр або годинник.

Основний принцип роботи циферблатного дашборда полягає в тому, що він має кругову шкалу, яка відображає діапазон значень метрики, з розбиттям на сегменти, що можуть бути пофарбовані в різні кольори для індикації нормальних, попереджувальних та критичних станів.

Стрілка або показник, що рухається по шкалі, вказує на поточне значення метрики. Положення стрілки змінюється залежно від зміни значення показника, що оновлюється з певного джерела даних, такого як база даних, API або датчики, які надають актуальну інформацію в реальному часі або з певним інтервалом.

Для полегшення сприйняття стану метрики шкала циферблатного дашборда може включати кольорові зони, де, наприклад, зелена зона означає безпечний рівень, жовта – потенційну загрозу, а червона – критичний стан.

Крім того, в дашборді можуть бути встановлені порогові значення, які визначають межі для різних зон, що дозволяє автоматично генерувати попередження або вживати заходів при перевищенні або недосягненні певних значень.

Деякі циферблатні дашборди є інтерактивними, дозволяючи користувачам налаштовувати порогові значення, вибирати різні метрики або переглядати детальну інформацію при наведенні курсору.

Циферблатні дашборди є інтуїтивно зрозумілими та візуально привабливими, надаючи користувачам швидкий і легкий спосіб оцінити стан важливих показників без необхідності в глибокому аналізі даних.

На рис 4.3 показано зовнішній вигляд розробленого віджету циферблатного віджету.

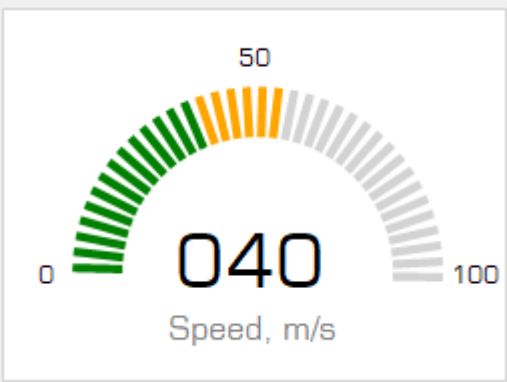


Рисунок 4.3 – Зовнішній вигляд розробленого віджету циферблатного віджету

Даний віджет має кругову шкалу з кольоровими зонами. Зелена зона відображає значення від 0 до 35 % від загального діапазону. Жовта зона – від 35 до 75 %. Червона – від 75 до 100 %.

Віджет має зону «Title» для відображення назви параметрів, що відображається.

Основна функція організації роботи даного віджета наведена нижче:

```
private void myPanel1_Paint(object sender, PaintEventArgs e)
{
   e.Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
   //Calc scale
   int lineCount = (int)((endAngle - startAngle) / angleStep);
   float scale = ((maxValue - minValue) / lineCount);
```

```
int porog = (int) (value / scale);
Graphics g = e.Graphics;
Pen pen_back = new Pen(Color.LightGray, arcWidth);
Pen pen_value = new Pen(Color.Green, arcWidth);
Pen pen_value_min = new Pen(Color.Green, arcWidth);
Pen pen_value_mid = new Pen(Color.Orange, arcWidth);
Pen pen_value_max = new Pen(Color.Red, arcWidth);
// Кольор форми
Point centerPoint = new Point(this.ClientSize.Width / 2, this.ClientSize.Height / 2);
centerPoint.Y = centerPoint.Y + (int)(centerPoint.Y / 2.5);
int porog_1 = 15;
int porog_2 = 30;
int curLine= 1;
for (float angle = startAngle + 180; angle <= endAngle + 180; angle += angleStep)
{
   // Обчислення прямокутника для малювання дуги
   float rectWidth = 2 * radius;
   float rectHeight = 2 * radius;
   float rectX = centerPoint.X - radius;
   float rectY = centerPoint.Y - radius;
   // Малюємо дугу
   if (curLine > porog)
   {   g.DrawArc(pen_back, rectX, rectY, rectWidth, rectHeight, angle, angleStep - 2);   }
   else {
      if (curLine < porog_1) g.DrawArc(pen_value_min, rectX, rectY, rectWidth, rectHeight, angle, angleStep - 2);
```

```
                else if (curLine < porog_2) g.DrawArc(pen_value_mid, rectX, rectY, rectWidth, rectHeight, angle, angleStep - 2);
                else g.DrawArc(pen_value_max, rectX, rectY, rectWidth, rectHeight, angle, angleStep - 2);     }
            curLine++;
        }
    }
```

Ця програма реалізує візуальний циферблатний дашборд у середовищі розробки Visual Studio, використовуючи бібліотеку Windows Forms і мову програмування C#. Основною метою програми є відображення значень метрик у вигляді дуги залежно від їх значень.

Програма починає роботу з події Paint панелі myPanel1, яка відповідає за малювання графічних елементів. У цій події налаштовується згладжування ліній для покращення якості діаграми. Далі розраховуються необхідні параметри для побудови дуги, такі як кількість ліній, що будуть відображені, ширина шкали, та інші.

Графічні об'єкти, такі як об'єкти типу Pen, використовуються для задання кольору та товщини ліній дуги. Для визначення центру панелі та правильного центрування дуг використовується обчислення координат.

Рисування дуг здійснюється у циклі for, де кожна дуга представляє різні значення метрики. Логіка визначення кольору дуги базується на порогових значеннях метрики, які відображаються різними кольорами: зеленим, помаранчевим та червоним.

Програма дозволяє інтерактивно відслідковувати зміни значень метрик у реальному часі, відображаючи їх у вигляді циферблатного дашборда зі згладженими лініями, що підвищує якість візуалізації.

4.3.2 Віджет для відображення компасу

Зовнішній вигляд віджету для відображення компасу показано на рис 4.4.

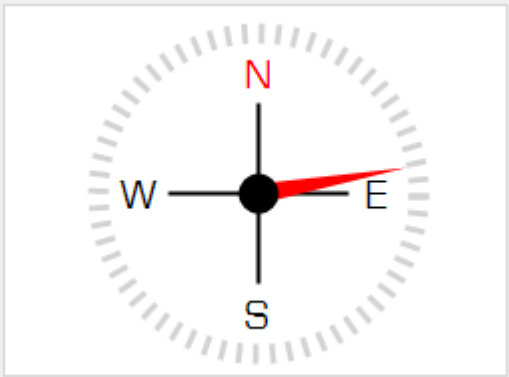


Рисунок 4.4 – Зовнішній вигляд віджету для відображення компасу

Функціонал дашборда включає шкалу, яка представлена набором дуг на окружності. Ці дуги служать для візуального позначення напрямків і метрик, що відображаються на дашборді.

Основні лінії виходять з центральної точки дашборда і показують напрямки північ, південь, захід і схід, допомагаючи користувачу орієнтуватися на дашборді (рис. 4.5).

```
float arcWidth = 10;
float arrowLength = radius - 5;
float arrowWidth = 10;

e.Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

//Рисуємо шкалу
Graphics g = e.Graphics;
Pen pen_shkala = new Pen(Color.LightGray, arcWidth);
Pen pen_arrow = new Pen(Color.Red, 1);
Pen pen_line = new Pen(Color.Black, 2);
Brush brush = new SolidBrush(Color.Red);

// Центр форми
Point centerPoint = new Point(this.ClientSize.Width / 2, this.ClientSize.Height / 2);

for (float angle = 0; angle <= 360; angle += 6)
{
    // Обчислення прямокутника для малювання дуги
    float rectWidth = 2 * radius;
    float rectHeight = 2 * radius;
    float rectX = centerPoint.X - radius;
    float rectY = centerPoint.Y - radius;
    g.DrawArc(pen_shkala, rectX, rectY, rectWidth, rectHeight, angle, 2);
}

//draw line
g.DrawLine(pen_line, centerPoint.X, centerPoint.Y, centerPoint.X, centerPoint.Y - radius + 35);
g.DrawLine(pen_line, centerPoint.X, centerPoint.Y, centerPoint.X, centerPoint.Y + radius - 35);
g.DrawLine(pen_line, centerPoint.X, centerPoint.Y, centerPoint.X - radius + 35, centerPoint.Y);
g.DrawLine(pen_line, centerPoint.X, centerPoint.Y, centerPoint.X + radius - 35, centerPoint.Y);
```

Рисунок 4.5 – Рисування шкали та ліній

Текстові позначення (N, S, W, E) розташовані на кожній стороні круга і вказують відповідні напрямки, роблячи дашборд зрозумілим для користувача (рис. 4.6).

```
//draw text
// Колір тексту
Brush brush_Text1 = new SolidBrush(Color.Red);
Brush brush_Text2 = new SolidBrush(Color.Black);

// Вивід тексту
// Встановлення шрифту
System.Drawing.Font font;
try
{
    font = new System.Drawing.Font("Square721 BT", 14);
}
catch (ArgumentException)
{
    // У випадку, якщо шрифт не знайдено, використовуємо стандартний шрифт
    font = new System.Drawing.Font(FontFamily.GenericSansSerif, 12);
}
g.DrawString("N", font, brush_Text1, centerPoint.X - 10, centerPoint.Y - radius + 10);
g.DrawString("S", font, brush_Text2, centerPoint.X - 10, centerPoint.Y + radius - 30);
g.DrawString("W", font, brush_Text2, centerPoint.X - radius + 8 , centerPoint.Y - 10);
g.DrawString("E", font, brush_Text2, centerPoint.X + radius - 30, centerPoint.Y - 10);
```

Рисунок 4.6 – Вивід текстових ззначень

Вказівник (стрілка) на дашборді є трикутною фігурою або вказівником, який вказує поточний напрямок. Він може рухатися на дашборді і вказувати на конкретний напрямок, що відображається на шкалі (рис. 4.7).

Центральна точка дашборда відокремлює показники від інших елементів і може використовуватися для відмітки центру або початкової точки дашборда.

```
//Рисуємо вказівник ===========
// Обчислення вершин трикутника (стрілки компаса)
float radians = arrowAngle * (float)Math.PI / 180;

PointF tipPoint = new PointF(
    centerPoint.X + arrowLength * (float)Math.Cos(radians),
    centerPoint.Y - arrowLength * (float)Math.Sin(radians)
);

PointF leftBasePoint = new PointF(
    centerPoint.X + arrowWidth / 2 * (float)Math.Sin(radians),
    centerPoint.Y + arrowWidth / 2 * (float)Math.Cos(radians)
);

PointF rightBasePoint = new PointF(
    centerPoint.X - arrowWidth / 2 * (float)Math.Sin(radians),
    centerPoint.Y - arrowWidth / 2 * (float)Math.Cos(radians)
);

PointF[] arrowPoints = { leftBasePoint, rightBasePoint, tipPoint };

// Малюємо трикутник
g.FillPolygon(brush, arrowPoints);
```

Рисунок 4.7 – Рисування вказівника

Кожен з цих елементів інтерфейсу керується за допомогою методів рисування, що надаються об'єктом Graphics бібліотеки Windows Forms. Властивості, такі як кольори, ширини ліній, шрифти та інші параметри, встановлюються для кожного елемента з метою досягнення необхідного вигляду дашборда. Такий підхід дозволяє створювати інтерактивні інтерфейси, що візуалізують дані і метрики у зручному для користувача форматі.

4.3.3 Віджет для відображення лінійної шкали

На рис 4.8 показано зовнішній вигляд віджету для відображення лінійної шкали.

Загальний вигляд програми є простим та ефективним, вона відображає необхідну інформацію про метрику у вигляді графічної шкали, яка легко інтерпретується користувачем.

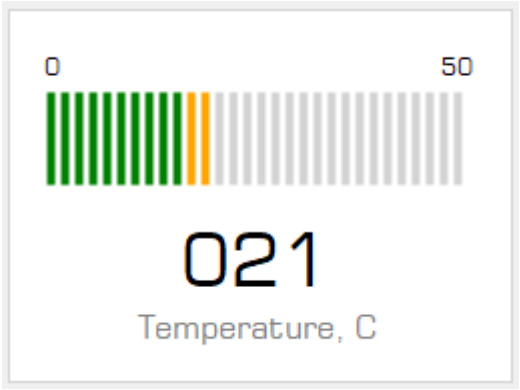


Рисунок 4.8 – Зовнішній вигляд віджету для відображення лінійної шкали.

Вихідний код програми показано на рис 4.9.

```
e.Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

float lineCount = 30;
int arcWidth = 4;
float scale = ((maxValue - minValue) / lineCount);
int porog = (int)(value / scale);
int porog_1 = 10;
int porog_2 = 21;
int margin = 20;
int margin_y = 40;

Graphics g = e.Graphics;
Pen pen_back = new Pen(Color.LightGray, arcWidth);
Pen pen_value_min = new Pen(Color.Green, arcWidth);
Pen pen_value_mid = new Pen(Color.Orange, arcWidth);
Pen pen_value_max = new Pen(Color.Red, arcWidth);

int lineSpacing = (this.ClientSize.Width - 2 * margin) / 30;
int lineHeight = this.ClientSize.Height - 7 * margin;
float lineWidth = this.ClientSize.Width / 50f;

// Встановлення пера для малювання ліній
Pen pen = new Pen(Color.Black, lineWidth);

for (int i = 0; i < 30; i++)
{
    int x = margin + i * lineSpacing;

    if (i >= porog)
    {
        g.DrawLine(pen_back, x, margin_y, x, margin_y + lineHeight);
    }
    else
    {
        if (i < porog_1) g.DrawLine(pen_value_min, x, margin_y, x, margin_y + lineHeight);
        else if (i < porog_2) g.DrawLine(pen_value_mid, x, margin_y, x, margin_y + lineHeight);
        else g.DrawLine(pen_value_max, x, margin_y, x, margin_y + lineHeight);
    }

}
```

Рисунок 4.9 – Функція відображення лінійної шкали

Ця програма відображає графічну шкалу на панелі myPanel1, використовуючи мову програмування C# та бібліотеку Windows Forms для створення користувацького інтерфейсу. Основна мета програми – візуалізувати дані метрики у вигляді горизонтальних ліній на екрані.

Так само, як і в попередній програмі, під час рисування шкали використовується антиаліасне згладжування (SmoothingMode.AntiAlias), щоб забезпечити плавність та якість ліній.

Параметри шкали, такі як кількість ліній (lineCount), ширина дуги (arcWidth), інтервали між лініями (lineSpacing), а також межі для відображення різних кольорів (за допомогою порогових значень porog_1, porog_2) обчислюються на основі параметрів, які задаються в програмі.

Лінії шкали мають різні кольори в залежності від їхнього положення на шкалі. Лінії, які відповідають значенням метрики нижче porog_1, малюються зеленим кольором (pen_value_min), лінії від porog_1 до porog_2 - помаранчевим (pen_value_mid), а лінії вище porog_2 - червоним (pen_value_max). Лінії, які знаходяться нижче поточного значення метрики, малюються з легким сірим кольором (pen_back).

Відстань між лініями (lineSpacing) обчислюється, щоб забезпечити, що шкала займає максимально можливий розмір панелі myPanel1. Кожна лінія малюється в циклі for, який проходить по всіх можливих значеннях шкали і розташовує лінії на відповідних позиціях на основі поточного значення метрики.

5 АПАРАТНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗБИРАННЯ ТЕЛЕМЕТРІЇ

5.1 Опис макету

Зовнішній вигляд макету показано на рис 5.1.

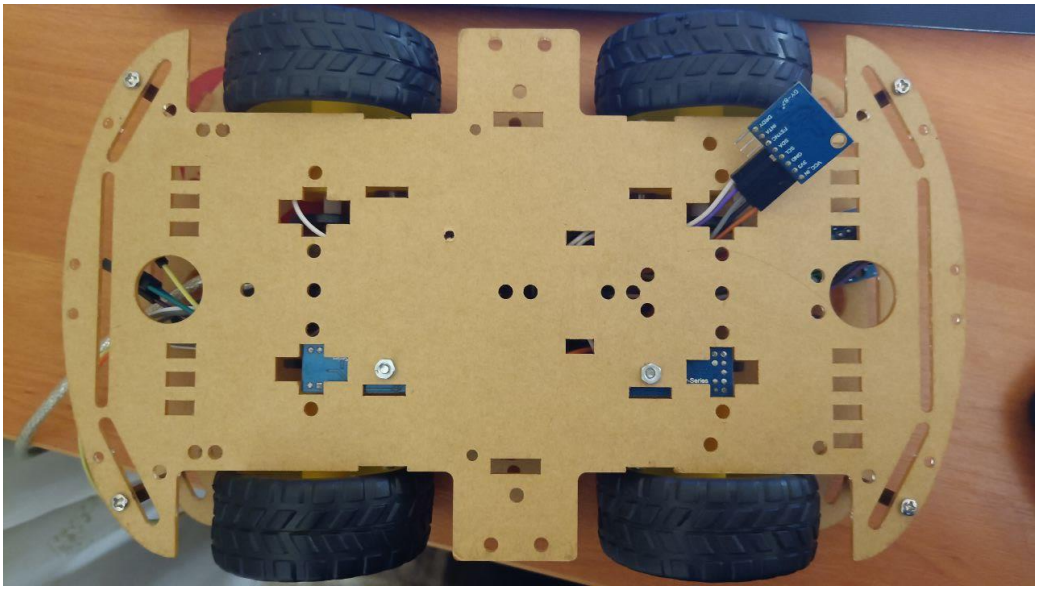


Рисунок 5.1 – Зовнішній вигляд макету

Макет являє собою мобільну роботизовану платформу з чотирма двигунами. Платформа має диференційний привод. Мотори поєднані парами. На рис 5.2 показана кінематична схема даного типу роботу.

Внутрішня кінематична модель роботу, що описує його положення на площині, визначається трикомпонентним вектором положення *q(t)*.

$$q(t) = \begin{bmatrix} x_1(t) \\ y_1(t) \\ z_1(t) \end{bmatrix}, \tag{5.1}$$

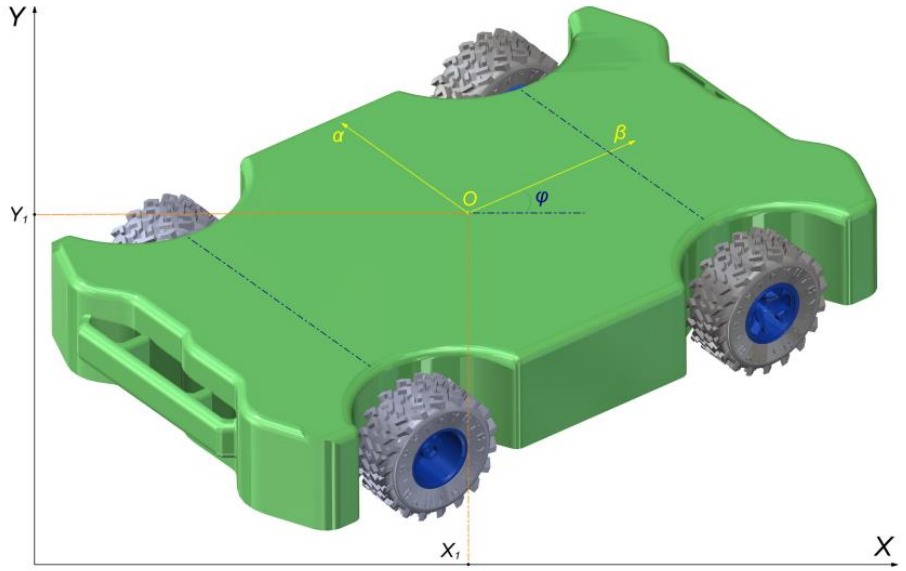


Рисунок 5.2 – Кінематична схема роботу

де x_1, y_1– координати точки О центру роботу в нерухомій системі координат $(\vec{X}, \vec{Y})$;

φ – кут між системами координат, що визначають положення рухомого базису $(\vec{\alpha}, \vec{\beta})$ щодо нерухомого базису $(\vec{X}, \vec{Y})$.

Поворот рухомої системи координат $(\vec{\alpha}, \vec{\beta})$ щодо нерухомої системи координат $(\vec{X}, \vec{Y})$ у зовнішній кінематичній моделі визначається транспонованим вектором $[X_1, Y_1]^T$ та ортогональною матрицею

$$R(\varphi) = \begin{bmatrix} cos\ \varphi & sin\ \varphi & 0 \\ -sin\ \varphi & cos\ \varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (5.2)$$

В основі конструкції лежать (рис. 5.3):

– чотири двигуна постійного струму для приводу коліс (1);

– драйвер двигунів постійного струму (2);

– стабілізатор вхідної напруги (3);

– МЕМС датчик положення з цифровим компасом, гіроскопом та акселерометром (4);

– Bluetooth модуль для організації зв’язку з ПК (5);

– плата керування Arduino UNO (6).

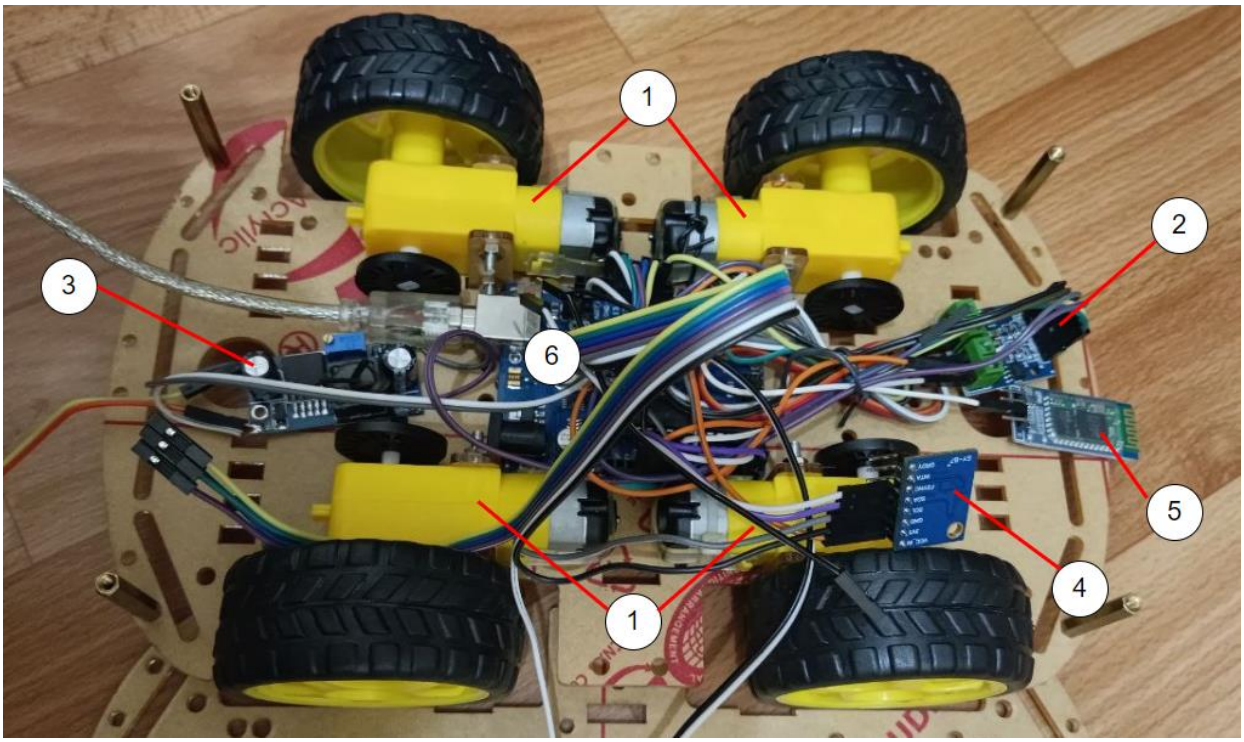


Рисунок 5.3 – Зібраний макет мобільного робота з системою телеметрії

Для контролю швидкості використовується оптичний датчик, що монтується на вісь двигуна в сукупності з диском із отворами.

На рис 5.4 наведено приклад кріплення частин датчику на корпусі мобільного робота.

Вихід датчика підключено до входу контролеру до входу 2, що має можливість відслідковувати зовнішні переривання. Таким чином, програма керування може відслідковувати та вимірювати кількість обертів на хвилину. В подальшому дана інформація перетворюється в швидкість метрів за секунду та за допомогою Bluetooth передавача передається на ПК.

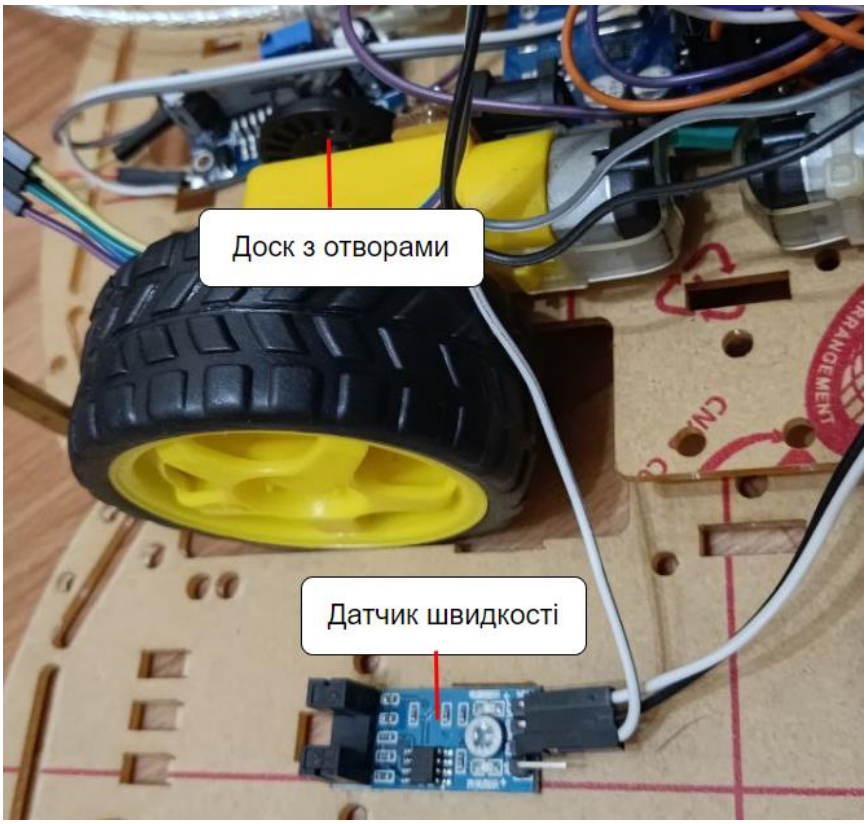


Рисунок 5.4 – Приклад кріплення частин датчику на корпусі мобільного робота

5.2 Розробка структурної схеми системи управління двигунами постійного струму

Для того щоб побудувати структурну схему автоматизованої системи управління двигуном потрібно розглянути конструкцію типового двигуна постійного струму, що застосовується в нашій конструкції (рис 5.5).

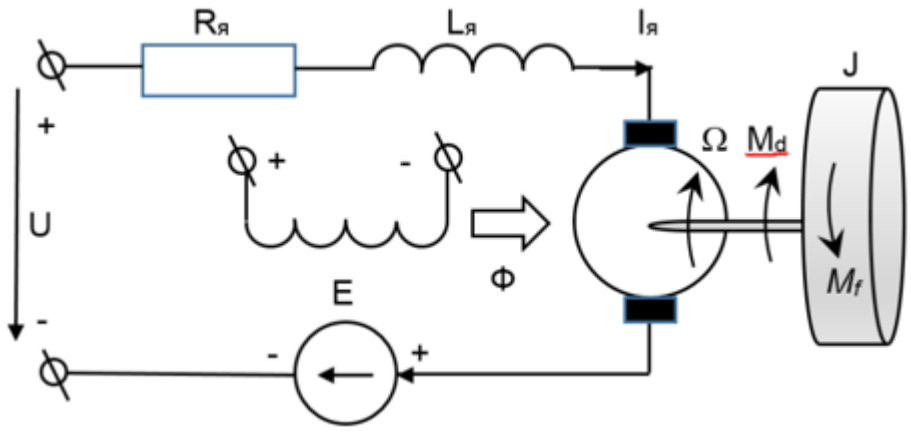


Рисунок 5.5 – Еквівалентна електрична схема двигуна постійного струму

На рис 5.5 використовуються наступні позначення:

– Ω –кутова швидкість валу двигуна;

– *J* – момент інерції рухомих частин двигуна приведені до валу;

– *Md* – рухомий момент прикладений до валу;

– *Mf* – момент прикладений до валу;

– $I_я$ – струм в колі якоря двигуна;

– $R_я$ – опір якоря двигуна;

– $L_я$ – індуктивність якоря двигуна;

– *U* – напруга на якорі.

Параметри двигуна пов’язані наступними співвідношеннями:

$$J\frac{d\Omega}{dt} = M_d - M_f\,, \tag{5.3}$$

де *J* – момент інерції рухомих частин двигуна приведені до валу;

M_d – рухомий момент прикладений до валу;

M_f – момент прикладений до валу;

Ω_0 – циклічна частота обертання;

t – час.

Формула (5.3) являє собою рівняння динаміки обертання ротора двигуна постійного струму.

Рухомий момент прикладений до валу розраховується наступним чином:

$$M_d = \frac{M_0}{I_0 \Phi_0} I\Phi, \tag{5.4}$$

де M_d – рухомий момент прикладений до валу;

M_0 – номінальний момент;

I_0 – номінальний струм;

Φ_0 – номінальний потік;

I – реальний струм;

Φ – реальний потік.

Як видно з формули (5.4) рухомий момент M_d пропорційний добутку струму I і магнітного потоку Φ. Коефіцієнт пропорційності представляють у вигляді Cm.

$$Cm = \frac{M_0}{I_0}, \tag{5.5}$$

де Cm – коефіцієнт пропорційності;

M_0 – номінальний момент;

I_0 – номінальний струм;

З формули (5.5) видно що він представляю собою відношення номінального моменту до добутку номінального струму I_0 та потоку Φ_0 залежить від конструкції двигуну та зазвичай наводиться в довіднику.

Електричні рівняння для струму в колі якоря наведене в формулі (5.6):

$$I_я R_я + \frac{dI_я}{dt} L_я = U - E, \tag{5.6}$$

де *Ія* – струм в колі якоря двигуна;

Rя – опір якоря двигуна;

t – час;

Lя – індуктивність якоря двигуна;

U – напруга на якорі;

E – зворотна ЕРС, що виникає внаслідок наводки напруги на рухомий ротор.

Чим вища швидкість обертання, тим вище значення Е:

$$E = \frac{E_0}{\Omega_0 \Phi_0} \Omega \Phi, \qquad (5.7)$$

де *E* – зворотна ЕРС;

E_0 –номінальна зворотна ЕРС;

Ω_0 – номінальна циклічна частота обертання;

Φ – реальний потік.

Ω – циклічна частота обертання

Коефіцієнт пропорційності між швидкістю зміни магнітного потоку (і, відповідно, швидкості обертання) це відношення номінальної проти ЕРС. E_0 до добутку номінальної швидкості Ω_0 та потоку Φ_0 *Ce* (5.7). Як і *Cm*, *Ce* залежить від параметрів двигуну:

$$Ce = \frac{E_0}{\Omega_0}, \qquad (5.7)$$

де E_0 – номінальна зворотна ЕРС;

Ω_0 – номінальна швидкість.

Подані рівняння не є лінійними, проте їх можна лінеаризувати за допомогою формули Тейлора. Тоді отримуємо набір рівнянь для механіки та електрики:

$$\begin{aligned}
&\Omega(s)=\left[M_d(s)-M_f(s)\right]\frac{1}{Js};\\
&M_d(s)=M_0\left[\frac{I(s)}{I_0}+\frac{\Phi(s)}{\Phi_0}\right];\\
&I_я(s)=\left[U(s)-E(s)\right]\frac{\frac{1}{R_я}}{T_я s+1};\\
&E(s)=E_0\left[\frac{\Omega(s)}{\Omega_0}+\frac{\Phi(s)}{\Phi_0}\right].
\end{aligned} \qquad (5.8)$$

де: Ω - циклічна частота обертання

M_d - рухомий момент прикладений до валу;

M_0 – номінальний момент;

J - момент інерції рухомих частин двигуна приведені до валу;

s – оператор Лапласа

I_0 – номінальний струм;

Φ_0 – номінальний потік;

I – реальний струм;

Φ – реальний потік;

$I_я$ –струм на якорі двигуна;

U - напруга на якорі;

E – зворотна ЕРС, що виникає внаслідок наводки напруги на рухомий

E_0 –номінальна зворотна ЕРС

$R_я$ - опір якоря двигуна;

$T_я$ – постійна часу якоря.

Загальна структурна схема системи управління двигуном постійного струму, що відповідає рівнянням (5.8) представлена на рисунку 5.6.

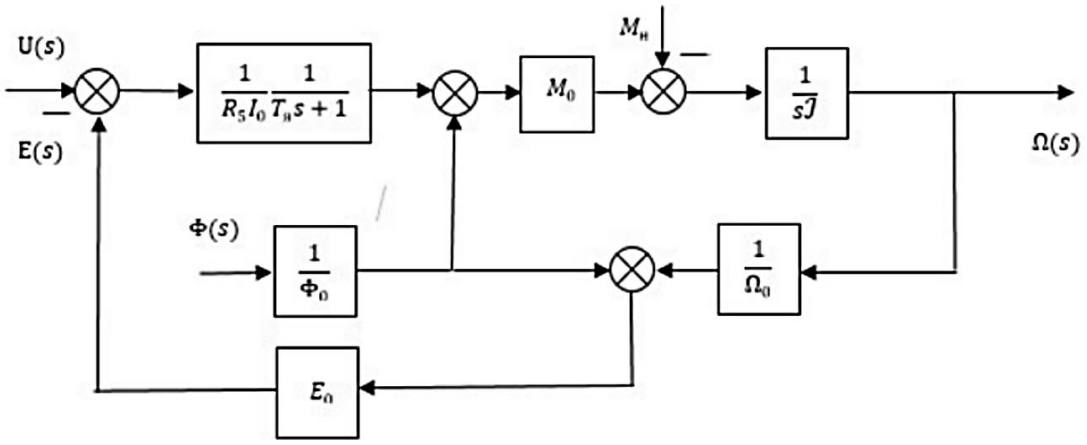


Рисунок 5.6 – Структурна схема системи управління двигуном постійного струму

5.3 Безпека життєдіяльності під час створення макету

Для забезпечення економічності системи використовується один вентилятор для забезпечення витяжки з усіх лабораторії, що в свою чергу не погіршує умов вентиляції.

З цією метою використовується вентилятор центробіжнього типу низького тиску Ц4-70. Сумарна продуктивність вентилятора L=707,68 м^3/ч. Рабочий тиск Р=40 кг*с/м^3. Для параметрів наведених вище характеристики вентилятора Ц4 - 70 такі:

– частота обертання 950 хв$^{-1}$;

– коэфіцієнт корисної дії 0,725;

– окружна швидкість колеса 28 м/с.

Потрібну потужність електродвигуна для приводу вентилятора визначаю за формулою:

$$P = \frac{L \times Pv}{3600 \times 102 \times \eta_B \times \eta_n} = \frac{707{,}68 \times 40}{3600 \times 102 \times 0{,}725 \times 0{,}95} \approx 0{,}11 \text{ кВт}, \qquad (5.9)$$

де L – продуктивність вентилятора, м^2/ч;

P_v – робочий тиск, кг*с/м^2;

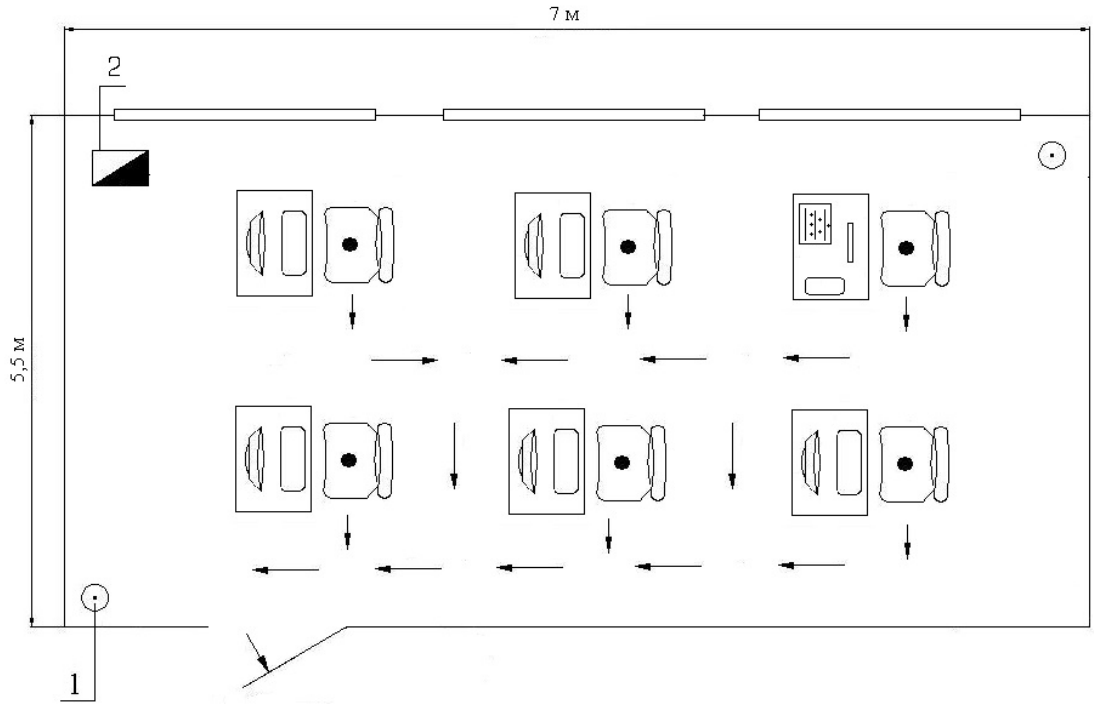
η_B – коэфіцієнт корисної дії вентилятора;

$\eta_П$ – коэфіцієнт корисної дії клиноременної передачі.

Вибір електродвигуна проводиться за заданою частотою обертання вентилятора і розрахованої потрібної потужності.

Джерелами шумів є робота пристроїв введення / виводу ПЕОМ (дисководи, кулер центрального процесора). Для забезпечення нормованих рівнів шуму в приміщенні встановлені шумопоглиначі.

Схема розміщення робочих місць і план евакуації при пожежі в лабораторії представлена на рис. 5.7.



1– вогнегасник; 2 – ящик з піском

Рисунок 5.7 – Схема розміщення робочих місць і план евакуації при пожежі

Так як робота виконується сидячи і не вимагає фізичної напруги, енерговитрати організму до 120 ккал / год, вона відноситься до робіт категорій Іа. Відповідно до ДСан ПиН для попередження втоми і підвищення працездатності осіб необхідно встановити раціональний режим праці та відпочинку. Для зменшення розумового перенапруження, перенапруження зорових аналізаторів і емоційних перевантажень слід встановити перерви по 20 хв через кожні 2 год після початку робіт, через 1,5 год і 2,5 год після обідньої перерви або ж по 5-15 хв через кожну годину роботи, також доцільно деякі перерви використовувати для виконання комплексу вправ. Загальна тривалість перерв (не рахуючи обідньої перерви) за 8-годинний день становить 60 хвилин.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи розроблено програму для відображення телеметрії з датчиків позиціонування мобільного робота.

Проведений аналіз літератури та аналогічних рішень показав актуальність впровадження телеметричної системи збору показань з датчиків мобільних роботів. Розглянуті принципи використання спеціальної інформаційної панелі для моніторингу та телеметрії. Показано, що система відображення телеметрії допомагає операторам отримувати дані в режимі реального часу, необхідні для ефективного керування мобільними роботами.

Розроблено архітектуру автоматизованої системи відображення даних телеметрії. Автоматизована система поєднує між собою контролер мобільного робота та ноутбук, або персональний комп’ютер за допомогою бездротової технології передавання даних. Для цього використовуються передавач та приймач, що можуть працювати в одному з дозволених діапазонів радіочастот.

Розроблено структурну схему підсистеми збору даних з датчиків мобільного робота та підсистеми візуалізації даних телеметрії. Розроблено схему алгоритму роботи програми в режимі отримання даних від мобільного роботизованого пристрою та схему алгоритму роботи підпрограми аналізу кадру даних.

Виконано вибір компонентів та розроблено макет для проведення експериментальних досліджень. Розроблено програму для отримання телеметрії від мобільного робота та надання графічної інформації операторові в зручному вигляді. Для цього розроблені графічні віджети: цифровий компас, циферблатний та лінійний віджети.

Виконані експериментальні включення показали працездатність програмного забезпечення відображення телеметрії з датчиків позиціонування мобільного робота.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра ля здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології освітньої програми «Системна інженерія» / Упоряд.: І. Ш. Невлюдов, О. М. Цимбал, О. В. Токарєва, А. І. Бронніков. – Харків: ХНУРЕ, 2022. – 66 с.

2. ДСТУ 3008-2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлення. – К.: Вид-во стандартів, 2016. – 26 с.

3. Положення про організацію освітнього процесу у ХНУРЕ [електронний ресурс] : Наказ ХНУРЕ від 27.11.2020 р. № 400. Режим доступу: https://nure.ua/wp-content/uploads/Main_Docs_NURE/polozhennja-pro-organizaciju-osvitnogo-procesu-v-hnure.pdf

4. Положення про організацію проведення практики здобувачів вищої освіти Харківського національного університету радіоелектроніки [Електронний ресурс] : Наказ ХНУРЕ від 03.05. 2019 р. № 222. Режим доступу : https://nure.ua/wp-content/uploads/222-vid-03.05.2019-pro-vvedennja-v-diju rishennja-vchenoi-radi-universitetu.pdf

5. Положення про академічну доброчесність [Електронний ресурс]: Наказ ХНУРЕ від 02 лютого 2021 р. № 50. – Режим доступу: https://nure.ua/wpcontent/ uploads/Main_Docs_NURE/polozhennja-pro-akademi chnu-dobrochesnist.pdf.

6. Невлюдов І.Ш. Навчальний посібник з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» : Навчальний посібник / І. Ш. Невлюдов, О. І. Филипенко, О. В. Токарєва, С. П. Новоселов, О. В. Сичова. – Харків: ХНУРЕ, 2023 . – 150 с.

7. Невлюдов І. Ш. Комп'ютерно-інтегровані технології виробництва технічних засобів автоматизації. Частина 1: підручник для студентів заклавдів

вищої осіти ; Харків. нац. ун-т радіоелектроніки. – Харків : ФОП Панов А.М., 2021. – 604 с. ISBN 978-617-7947-67-6

8. Невлюдов І.Ш. Виробничі процеси та обладнання об’єктів автоматизації: Підручник / Кривий Ріг: КК НАУ, 2017. – 444 с.

9. Невлюдов І.Ш. Виробничі процеси та обладнання об’єктів автоматизації. Збірник задач: Навчальний посібник / І.Ш. Невлюдов, А.О. Андрусевич, Г.В. Пономарьова, А.О. Функендорф. Кривий Ріг: КК НАУ. 2018. – 332 с.

10. Невлюдов І.Ш. Технічні засоби автоматизації: Підручник / І.Ш. Невлюдов, А.О. Андрусевич, О.І. Филипенко, Н.П. Демська, С.П. Новоселов. – Кривий Ріг: Криворізький коледж НАУ, 2019. – 366 с.

11. Учасники проектів Вікімедіа. Телеметрія – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Телеметрія (дата звернення: 03.05.2024).

12. Телеметрія: процес, у якому виконуються вимірювання - Polaridad.es. URL: https://polaridad.es/uk/telemedida-proceso-en-que-las-mediciones-se-realizan-en/ (дата звернення: 03.05.2024).

13. Monitor the performance of your robot fleet. Reality Capture | Drone Mapping Software | Photo Documentation. URL: https://www.dronedeploy.com/blog/monitor-the-performance-of-your-robot-fleet (date of access: 03.05.2024).

14. About the Robotics Portal. URL: https://docs-automate.dronedeploy.com/docs/concepts (date of access: 06.05.2024).

15. FRC LabVIEW Dashboard. FIRST Robotics Competition Documentation. URL: https://docs.wpilib.org/en/stable/docs/software/dashboards/labview-dashboard/driver-station-labview-dashboard.html (date of access: 06.05.2024).

16. Винокурова Л.Е. Васильчук М.В. Гаман Н.В.Основи охорони праці. Підручник для ПТНЗ