Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютерна система бортової телеметрії та адаптивної візуалізації стану БПЛА на основі платформи QGroundControl*

Виконав: студент 4 курсу, групи СІ-42
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

(підпис) Козачук К.О (прізвище та ініціали)

Керівник (підпис) Лещишин Ю.З (прізвище та ініціали)

Нормоконтроль (підпис) Луцик Н.С (прізвище та ініціали)

Завідувач кафедри (підпис) Осухівська Г.М. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
«25» квітня 2026 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студента *Козачук Карини Олегівни*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютерна система бортової телеметрії та адаптивної візуалізації стану БПЛА на основі платформи QGroundControl*

Керівник роботи *к. т. н. Лещишин Юрій Зіновійович*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від
« 24 » квітня 2026 року № 4/9-188
2. Термін подання студентом завершеної роботи *15.06.2026р*
3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз технічного завдання
2. Проєктна частина
3. Практична частина
4. Безпека життєдіяльності, основи охорони праці.
Висновки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Структурна схема комп'ютерної системи бортової телеметрії БПЛА.
2. Схема електрична принципова бортового модуля телеметрії на основі STM32.
3. Блок-схема алгоритму збору, обробки та передавання телеметричних даних.
4. UML-діаграма стану
5.UML-діаграма компонентів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	*Сенчишин В.С. к.т.н., доцент*		

7. Дата видачі завдання *25.04.2026 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Розробка технічного завдання*	*26.01 – 02.02*	
2.	*Робота над першим розділом «Аналіз технічного завдання»*	*03.02 – 15.02*	
3.	*Робота над другим розділом «Проєктна частина»*	*20.04 – 25.04*	
4.	*Робота над третім розділом «Практична частина»*	*26.04 – 05.05*	
5.	*Робота над четвертим розділом «Безпека життєдіяльності, основи охорони праці»*	*07.05 – 25.05*	
6.	*Оформлення пояснювальної записки і графічного матеріалу*	*26.05 – 7.06*	
7.	*Перевірка на академічний плагіат, перевірка керівником та консультантами*	*8.06 – 14.06*	
8.	*Попередній захист кваліфікаційної роботи бакалавра*	*15.06 – 21.06*	
9.	*Захист кваліфікаційної роботи бакалавра*	*25.06.2026р.*	

Студент _____ *Козачук К.О*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Лещишин Ю.З*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Козачук К.О. Комп’ютерна система бортової телеметрії та адаптивної візуалізації стану БПЛА на основі платформи QGroundControl: робота на здобуття кваліфікаційного ступеня бакалавра: спец. 123 - комп’ютерна інженерія. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2026.

Ключові слова: БПЛА, бортова телеметрія, STM32, Raspberry Pi, QGroundControl, UDP, MAVLink.

Кваліфікаційна робота бакалавра складається з чотирьох розділів.

У першому розділі виконано аналіз технічного завдання, визначено вимоги до комп’ютерної системи бортової телеметрії БПЛА та обґрунтовано вибір основних апаратних і програмних засобів.

У другому розділі описано проєктування системи, розроблено її структурну та принципову електричну схеми, а також подано алгоритм збору, обробки й передавання телеметричних даних.

У третьому розділі виконано програмну реалізацію основних модулів системи, організовано передавання телеметрії через Raspberry Pi та перевірено приймання даних у QGroundControl.

Четвертий розділ описує питання безпеки життєдіяльності та основи охорони праці.

ANNOTATION

Kozachuk K.O. Computer system of onboard telemetry and adaptive visualization of UAV state based on the QGroundControl platform: Bachelor’s Graduation Thesis: speciality 123 - computer engineering. Ternopil: Ternopil Ivan Puluj National Technical University, 2026.

Keywords: UAV, onboard telemetry, STM32, Raspberry Pi, QGroundControl, UDP, MAVLink.

The bachelor's qualification work consists of four sections.

The first section analyzes the technical task, defines the requirements for a UAV onboard telemetry computer system and substantiates the choice of the main hardware and software tools.

The second section describes the system design, develops its structural and electrical schematic diagrams, and presents the algorithm for collecting, processing and transmitting telemetry data.

The third section implements the main software modules of the system, organizes telemetry transmission through Raspberry Pi and verifies data reception in QGroundControl.

The fourth section describes the issues of life safety and the basics of labor protection.

ЗМІСТ

					КС КРБ 123.177.00.00 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Козачук К.О.			Комп’ютерна система бортової телеметрії та адаптивної візуалізації стану БПЛА на основі платформи QGroundControl Пояснююча записка	Літ.	Арк.	Аркушів
Перевір.		Лещишин Ю.З.				Н	6	2
Реценз.						ТНТУ, каф. КС, гр. СІ-42		
Н. Контр.		Луцик Н.С.						
Затверд.		Осухівська Г.М.						

СПИСОК СКОРОЧЕНЬ

ARM — Advanced RISC Machine

GPIO — General—Purpose Input/Output

GPS — Global Positioning System

HAL — Hardware Abstraction Layer

I2C — Inter-Integrated Circuit

IDE — Integrated Development Environment

IMU — Inertial Measurement Unit

MAVLink — Micro Air Vehicle Link

MCU — Microcontroller Unit

MEMS — Micro-Electro-Mechanical Systems

PWM — Pulse-Width Modulation

QGC — QGroundControl

RPi — Raspberry Pi

STM32 — сімейство 32-бітних мікроконтролерів STMicroelectronics

SWD — Serial Wire Debug

UART — Universal Asynchronous Receiver-Transmitter

UDP — User Datagram Protocol

БПЛА — безпілотний літальний апарат

ПЗ — програмне забезпечення

ВСТУП

Бортова телеметрія є однією з основних складових безпілотного літального апарата, оскільки саме вона забезпечує передавання інформації про його поточний стан до наземної станції. Для оператора важливо отримувати не лише окремі числові значення, наприклад координати або висоту, а й розуміти загальний стан телеметричного потоку: чи оновлюються дані, чи стабільно працює зв'язок, чи є актуальними покази сенсорів і чи можна використовувати їх для оцінки положення БПЛА. Якщо такі параметри відображаються без урахування їх достовірності, оператор може неправильно оцінити ситуацію, особливо під час втрати GPS-фіксації, затримки передавання або нестабільної роботи окремих вимірювальних модулів.

У системах малого класу для побудови телеметричних вузлів доцільно використовувати доступні мікроконтролери, одноплатні комп'ютери та сенсорні модулі. Такий підхід дає можливість створити макетну апаратно-програмну систему, у якій окремі функції розподілені між кількома рівнями. Мікроконтролер STM32 виконує зчитування даних із бортових сенсорів, первинну підготовку вимірювань і формування телеметричного пакета. Raspberry Pi використовується як проміжний обчислювальний вузол, який приймає дані від STM32, виконує їх фільтрацію, додаткову обробку та передавання на комп'ютер оператора. Наземна станція на основі QGroundControl забезпечує приймання й відображення телеметричних параметрів у зручній для аналізу формі.

Актуальність теми зумовлена потребою у створенні комп'ютерної системи, яка поєднує збір телеметричних даних, їх обробку, передавання та адаптивну візуалізацію стану БПЛА. На відміну від простого виведення числових значень, адаптивне відображення має враховувати стан самих даних: їх актуальність, стабільність оновлення та наявність можливих помилок.

Такий підхід дозволяє не лише показати параметри польоту або положення апарата, а й вказати, чи є ці параметри придатними для використання в поточний момент.

У кваліфікаційній роботі розробляється комп'ютерна система бортової телеметрії та адаптивної візуалізації стану БПЛА на основі платформи QGroundControl. У складі системи передбачено використання мікроконтролера STM32F401CCUx, Raspberry Pi 3 Model B Rev 1.2, інерціального модуля, магнітометра, GPS-модуля та барометричного датчика. Передавання даних між STM32 і Raspberry Pi організовується через інтерфейс USB CDC, а UART1 залишається резервним каналом для налагодження.Така архітектура дозволяє відокремити задачі збору даних від задач мережевого обміну та візуалізації.

Практичне значення роботи полягає у створенні апаратно-програмного рішення, яке може бути використане для перевірки принципів збору, обробки й передавання телеметричних параметрів БПЛА. Розроблена система дає змогу поетапно перевіряти роботу сенсорного вузла, мікроконтролера, проміжного вузла Raspberry Pi та приймання даних у QGroundControl. Це створює основу для подальшого вдосконалення бортових телеметричних комплексів і розширення їх функціональності відповідно до вимог конкретного застосування.

Комплексне лабораторне тестування спроєктованого тракту підтвердило стабільність наскрізного проходження сигналу та коректність роботи маркерів адаптивної візуалізації, що практично доводить життєздатність та ефективність обраної багаторівневої архітектури. Отримані результати можуть бути безпосередньо використані під час проєктування спеціалізованих бортових систем моніторингу для малих безпілотних апаратів, де критичним є не лише збір показників, а й надійність їхньої доставки кінцевому користувачу в умовах нестабільного зв'язку.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Аналіз вимог до комп'ютерної системи бортової телеметрії БПЛА

Бортова телеметрія забезпечує передавання інформації про поточний стан БПЛА від бортової частини до наземної станції. Для оператора важливими є не лише координати або висота, а повніший набір параметрів, який дозволяє оцінити положення апарата, його рух, стан сенсорів і стабільність надходження даних. Якщо телеметрія оновлюється із затримкою або окремі параметри втрачають актуальність, числові значення самі по собі можуть створювати неправильне уявлення про стан системи.

Розроблювана комп'ютерна система має забезпечувати збір даних із кількох груп сенсорів. Інерціальний модуль використовується для отримання прискорень і кутових швидкостей, магнітометр — для визначення курсу, GPS-модуль — для отримання координат і швидкості руху, а барометричний датчик — для вимірювання атмосферного тиску та оцінки зміни висоти [20]. Разом ці параметри формують телеметричний набір, достатній для відображення основного стану БПЛА на наземній станції.

Сенсорна частина системи повинна залишатися модульною. Це дає можливість підключати окремі вимірювальні модулі через стандартні інтерфейси та, за потреби, замінювати їх сумісними аналогами без зміни загальної архітектури. Для обміну з датчиками використовуються поширені інтерфейси I2C, та UART, які підтримуються мікроконтролерами STM32 і більшістю сенсорних модулів.

Нижній рівень системи реалізується на мікроконтролері STM32F401CCUx [15]. Його задача полягає у зчитуванні даних із сенсорів, первинній підготовці вимірювань і формуванні телеметричного пакета.

Передавання та подальша обробка телеметрії виконуються через проміжний вузол Raspberry Pi. Він приймає сформовані пакети від STM32, виконує фільтрацію, додаткову обробку даних і передає їх на комп'ютер оператора через мережевий канал. Це дозволяє не перевантажувати мікроконтролер задачами мережевого обміну та відокремити збір даних від їх передавання до наземної станції.

Наземна частина системи базується на QGroundControl. Це середовище використовується для приймання телеметрії та відображення параметрів стану БПЛА. Для цієї роботи важливо, щоб на наземній станції відображалися не тільки самі значення, а й стан їх оновлення. Якщо пакети перестають надходити, GPS-дані втрачають актуальність або покази сенсорів стають нестабільними, інтерфейс повинен виділяти такий стан окремо.

Окремою вимогою є можливість поетапної перевірки системи. Спочатку контролюється робота сенсорних модулів і обмін із STM32, далі — формування телеметричного пакета, передавання на Raspberry Pi, обробка та фільтрація даних, а після цього - приймання й відображення параметрів [3] у QGroundControl. Такий підхід спрощує пошук помилок і дозволяє перевіряти систему не одним великим блоком, а за окремими функціональними рівнями.

Система має забезпечувати збір, первинну підготовку, фільтрацію, передавання та візуалізацію телеметричних даних БПЛА. Основними вимогами до неї є модульність, підтримка стандартних інтерфейсів, розподіл функцій між STM32 і Raspberry Pi, передавання даних у форматі, придатному для приймання на ПК, а також відображення параметрів у QGroundControl з урахуванням актуальності та достовірності телеметрії.

1.2 Огляд рішень для збору, передавання та візуалізації телеметричних даних

Телеметрична система БПЛА охоплює не лише сенсори, що вимірюють окремі параметри, а й увесь шлях проходження даних: від збору вимірювань на борту до їх передавання на наземну станцію та відображення оператору. Тому під час аналізу рішень доцільно розглядати три складові: засоби збору даних, канали передавання та програмні засоби візуалізації.

Подібний принцип побудови використовується і в інших комп'ютерних системах моніторингу, де сенсорні вузли збирають параметри об'єкта або середовища, після чого дані передаються для обробки та відображення користувачу [4, 5].

Для збору телеметрії часто використовують готові польотні контролери. Вони мають необхідні інтерфейси для підключення сенсорів, GPS-модулів і каналів зв'язку, а також підтримують стандартні протоколи обміну. Перевагою такого підходу є наявність готової екосистеми та відпрацьованих алгоритмів роботи. Водночас значна частина логіки прихована всередині готової прошивки, тому змінювати структуру даних або реалізовувати власний порядок їх обробки складніше.

Іншим підходом є побудова вузла збору телеметрії на основі мікроконтролера. У цьому випадку розробник самостійно визначає набір сенсорів, періодичність їх опитування, формат пакета та спосіб передавання даних. Такий варіант потребує більшої кількості розробницької роботи, але дає кращий контроль над логікою системи.

Сенсори можна підключати і безпосередньо до одноплатного комп'ютера, наприклад Raspberry Pi. Такий варіант зручний для обробки даних і роботи з мережевими інтерфейсами, однак менш придатний для задач нижнього рівня, де потрібне передбачуване опитування сенсорів. Оскільки Raspberry Pi працює під керуванням операційної системи, часові затримки під

час роботи з периферією можуть бути менш стабільними, ніж у мікроконтролерній системі. Через це доцільним є розділення функцій: STM32 виконує збір і первинну підготовку даних, а Raspberry Pi забезпечує фільтрацію, додаткову обробку та передавання на ПК.

Передавання телеметрії може виконуватися через послідовний інтерфейс UART, радіомодем або мережевий канал. UART є простим і зручним для обміну між STM32 та Raspberry Pi, а також для налагодження формату телеметричного пакета. Радіомодеми наближені до реального застосування БПЛА, але потребують окремого обладнання та налаштування каналу зв'язку. Для макетної реалізації доцільно спочатку забезпечити стабільне формування пакетів і передавання їх між вузлами системи, після чого дані можуть пересилатися на комп'ютер оператора через UDP[2] або у форматі MAVLink.

Для візуалізації телеметрії можуть використовуватися термінальні програми, власні застосунки або готові наземні станції. Виведення даних у термінал зручне для первинного налагодження, але не є повноцінним інтерфейсом оператора. Створення власної наземної станції дає повний контроль над інтерфейсом, однак потребує значного обсягу додаткової розробки. Більш практичним варіантом є використання QGroundControl, оскільки це середовище вже орієнтоване на роботу з БПЛА та підтримує приймання телеметричних повідомлень через MAVLink.

Для розроблюваної системи важливо не лише відображати числові значення параметрів, а й показувати стан самих даних. Якщо телеметричні пакети перестають надходити, GPS-дані втрачають актуальність або покази сенсорів стають нестабільними, оператор повинен бачити це як окремий стан. Такий підхід підвищує інформативність наземної станції, оскільки дозволяє оцінювати не тільки параметри БПЛА, а й придатність отриманої телеметрії для використання.

Порівняння основних підходів наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння підходів до збору телеметричних даних БПЛА

Варіант рішення	Переваги	Обмеження
Готовий польотний контролер	Готова екосистема; підтримка стандартних протоколів; наявність відпрацьованих алгоритмів	Менший контроль над внутрішньою логікою та структурою експериментальних даних
Мікроконтролерний вузол	Контроль над опитуванням сенсорів, форматом пакета та алгоритмами обробки	Потребує самостійної розробки програмного забезпечення
Підключення сенсорів до Raspberry Pi	Зручна обробка даних і робота з мережевими інтерфейсами	Менш передбачуване опитування сенсорів через роботу операційної системи
UART-з'єднання	Простота реалізації, зручність налагодження та передавання пакетів між STM32 і Raspberry Pi	Обмежене використання без додаткового шлюзу або приймальної програми
Радіомодем телеметрії	Наближеність до реального використання БПЛА	Потребує окремого радіоканалу та додаткового налаштування
Raspberry Pi як шлюз	Розділення задач між збором даних і мережевим передаванням	Потребує окремого програмного модуля приймання, обробки та пересилання даних
Термінал або тестова програма	Зручність для первинної перевірки потоку даних	Не підходить як повноцінний інтерфейс оператора
Власна програма візуалізації	Повний контроль над інтерфейсом	Великий обсяг розробки, що зміщує акцент із телеметричної системи
QGroundControl	Готове середовище для роботи з БПЛА, підтримка MAVLink, можливість перевірки приймання телеметрії	Потребує узгодження формату переданих даних із підтримуваними MAVLink-повідомленнями

Для цієї системи обрано поєднання мікроконтролерного збору даних, проміжного вузла Raspberry Pi та візуалізації в QGroundControl. Така архітектура дозволяє зберегти контроль над сенсорним рівнем, винести обробку та передавання на окремий вузол і не створювати наземну станцію повністю з нуля.

1.3 Обґрунтування вибору апаратної платформи STM32, Raspberry Pi та сенсорних модулів

Для побудови комп’ютерної системи бортової телеметрії БПЛА обрано розподілену апаратну структуру, у якій задачі збору, обробки, передавання та візуалізації даних виконуються окремими вузлами. Такий підхід є доцільним, оскільки телеметрична система повинна одночасно працювати з кількома сенсорними модулями, формувати узгоджений потік даних і передавати його на наземну станцію у придатному для відображення форматі.

Нижній рівень системи реалізується на основі мікроконтролера STM32F401CCUx. Його використання пов’язане з необхідністю регулярного опитування сенсорів і роботи з периферійними інтерфейсами. Мікроконтролер не залежить від операційної системи загального призначення, тому краще підходить для задач зчитування даних із датчиків, первинної підготовки вимірювань і формування телеметричного пакета.

Для передавання та подальшої обробки телеметрії використовується Raspberry Pi. Цей вузол приймає дані від STM32 через послідовний інтерфейс, виконує фільтрацію та додаткову обробку параметрів, після чого передає їх на комп’ютер оператора через мережевий канал. Завдяки такому розподілу STM32 зосереджується на роботі з сенсорами, а Raspberry Pi — на обробці та комунікації з наземною станцією.

Сенсорна частина системи формується з модулів, які забезпечують отримання основних параметрів стану БПЛА. Інерціальний модуль використовується для визначення прискорень і кутових швидкостей, магнітометр - для оцінки курсу, GPS-модуль — для отримання координат, а барометричний датчик — для визначення атмосферного тиску та зміни висоти. Такий набір дає змогу сформувати телеметричний пакет, достатній для контролю положення, руху й навігаційного стану апарата.

Важливою умовою вибору апаратної платформи є підтримка стандартних інтерфейсів обміну даними. Для підключення сенсорів і взаємодії між вузлами системи використовуються I2C, UART та UDP. Це спрощує підключення модулів, полегшує налагодження та залишає можливість заміни окремих компонентів сумісними аналогами без зміни загальної архітектури.

Обрана структура відповідає задачам роботи, оскільки поєднує стабільність мікроконтролерного збору даних із гнучкістю одноплатного комп'ютера під час обробки та передавання телеметрії. Детальніше склад апаратних модулів, їх підключення та використані інтерфейси розглядаються в проєктній частині роботи.

1.4 Аналіз середовища QGroundControl для адаптивної візуалізації параметрів стану модулів БПЛА

Для відображення телеметричних параметрів БПЛА потрібне середовище, яке може приймати дані від бортової частини, показувати їх оператору та працювати з типовими для безпілотних систем форматами обміну. У цій системі для наземної частини використовується QGroundControl, оскільки він уже орієнтований на роботу з БПЛА та підтримує приймання телеметричних повідомлень [10]

Перевагою QGroundControl є наявність готових засобів для відображення польотної інформації, карти, стану зв'язку та основних параметрів апарата. Це дозволяє не створювати наземну станцію повністю з нуля, а зосередити увагу на формуванні, передаванні та правильному поданні телеметричних даних. Для розроблюваної системи це важливо, оскільки основна задача полягає не у створенні окремого графічного застосунку, а в побудові повного ланцюга передавання телеметрії від сенсорів до наземної станції.

QGroundControl також зручний для перевірки приймання даних у форматі MAVLink. Через MAVLink Inspector можна контролювати, чи надходять телеметричні повідомлення, з якою частотою вони оновлюються та які значення передаються. Це дає змогу перевірити роботу Raspberry Pi як проміжного вузла, правильність формування повідомлень і наявність зв’язку між бортовою та наземною частинами системи.

Приклад інтерфейсу QGroundControl для роботи з телеметричними параметрами наведено на рисунку 1.1.

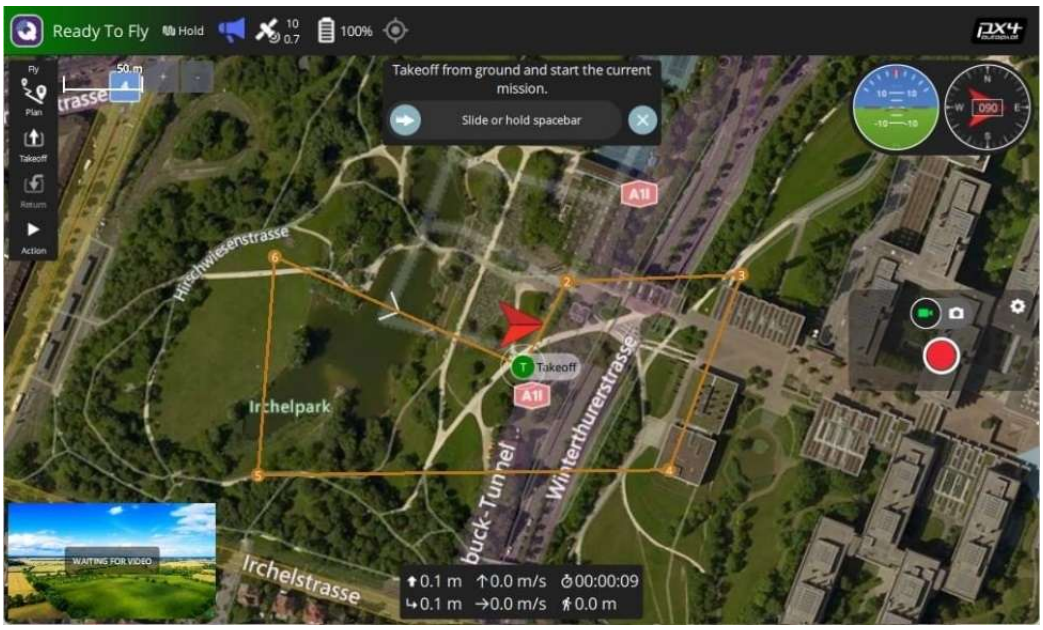

Рисунок 1.1 – Приклад інтерфейсу QGroundControl для відображення параметрів БПЛА

Застосування QGroundControl дозволяє контролювати не лише сам факт надходження телеметрії, а й її достовірність. Стабільне оновлення гарантує актуальність показників, тоді як будь-які затримки миттєво сигналізують про апаратні збої чи втрату зв'язку. Суть адаптивної візуалізації полягає в тому, що навігаційні координати, висота та швидкість виводяться в інтерфейс виключно разом з індикаторами їхньої цілісності. Завдяки цьому платформа ефективно виконує роль наземної станції.

РОЗДІЛ 2 ПРОЄКТНА ЧАСТИНА

2.1 Розробка структурної схеми комп'ютерної системи бортової телеметрії

Структурна схема комп'ютерної системи бортової телеметрії БПЛА визначає загальну архітектуру системи та взаємодію між її основними функціональними вузлами. На цьому етапі не деталізуються електричні з'єднання окремих елементів, а показується шлях проходження телеметричних даних від сенсорних модулів до наземної станції.

Система має розподілену структуру і складається з бортової та наземної частин. Бортова частина виконує зчитування параметрів із сенсорів, формування телеметричного пакета та передавання даних через проміжний вузол. Наземна частина приймає телеметрію та відображає її в середовищі QGroundControl.

Нижній рівень бортової частини реалізується на мікроконтролері STM32F401CCUx. До нього підключаються сенсорні модулі, які забезпечують отримання основних параметрів стану БПЛА, а саме інерціальних даних, барометричного тиску, координат, швидкості та напрямку. Для обміну з датчиками використовуються стандартні інтерфейси I2C, та UART. Після зчитування даних STM32 виконує їх первинну підготовку, формує телеметричний пакет і передає його на Raspberry Pi.

Raspberry Pi у структурі системи виконує роль проміжного обчислювального та комунікаційного вузла. Він приймає дані від STM32 через UART, виконує фільтрацію та додаткову обробку телеметричних параметрів, після чого передає їх на персональний комп'ютер оператора через мережевий канал. Для передавання даних до наземної станції використовується UDP або

формат MAVLink.

Наземна частина системи представлена персональним комп'ютером із програмним забезпеченням QGroundControl. На цьому рівні виконується приймання телеметричних повідомлень і візуалізація параметрів стану БПЛА. До таких параметрів належать координати, висота, швидкість, орієнтація, стан надходження даних і службові ознаки актуальності телеметрії.

Структурну схему комп'ютерної системи бортової телеметрії БПЛА наведено на рисунку 2.1.

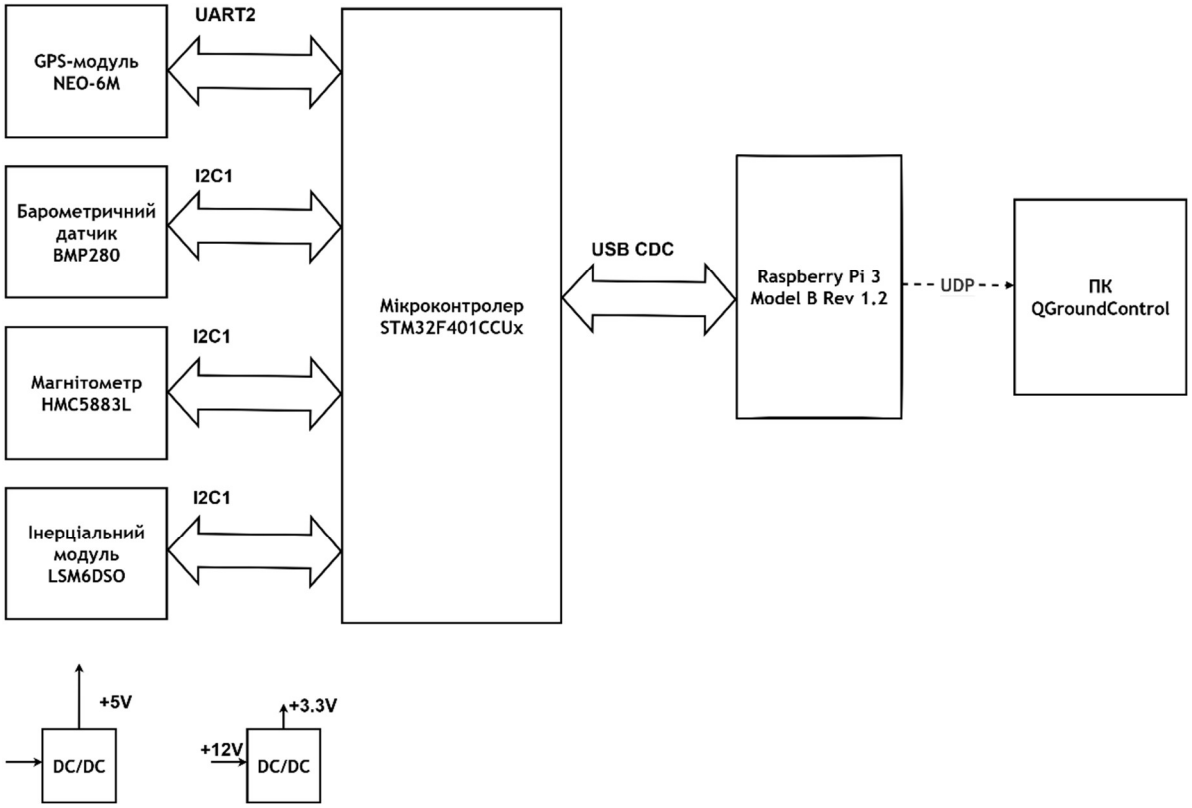

Рисунок 2.1 – Структурна схема комп'ютерної системи бортової телеметрії БПЛА

Подана схема відображає послідовний шлях проходження даних у системі: сенсорні модулі передають вимірювання на STM32, мікроконтролер формує телеметричний пакет, Raspberry Pi виконує обробку та передавання, а QGroundControl забезпечує приймання і відображення параметрів оператору.

Такий розподіл функцій спрощує налагодження системи, оскільки кожен рівень можна перевіряти окремо: сенсорний вузол, мікроконтролер, проміжний вузол Raspberry Pi та наземну станцію.

Модульна структура також залишає можливість подальшого розширення системи. За потреби окремі сенсори можуть бути замінені сумісними модулями, а спосіб передавання телеметрії може бути змінений без повної переробки архітектури. Це важливо для макетної реалізації, де система повинна бути придатною до поетапного тестування та подальшого вдосконалення.

2.2 Обґрунтування вибору апаратного забезпечення системи

Апаратне забезпечення комп’ютерної системи бортової телеметрії БПЛА вибирається з урахуванням задач збору, первинної підготовки, обробки, передавання та відображення телеметричних даних. Система має працювати з кількома групами параметрів: інерціальними вимірюваннями, навігаційними даними, барометричним тиском і напрямком руху апарата. Для цього використовується розподілена апаратна структура, у якій кожен вузол виконує окрему функцію.

Основним вузлом нижнього рівня обрано мікроконтролер STM32F401CCUx. Він відповідає за підключення сенсорів, регулярне зчитування даних, первинну підготовку вимірювань і формування телеметричного пакета. Використання STM32 є доцільним, оскільки мікроконтролер працює без операційної системи загального призначення та забезпечує більш передбачувану роботу з периферійними інтерфейсами. Наявність I2C та UART дозволяє підключити основні сенсорні модулі без використання складної додаткової обв’язки.

Зовнішній вигляд мікроконтролерної плати STM32F401CCUx наведено на рисунку 2.2.

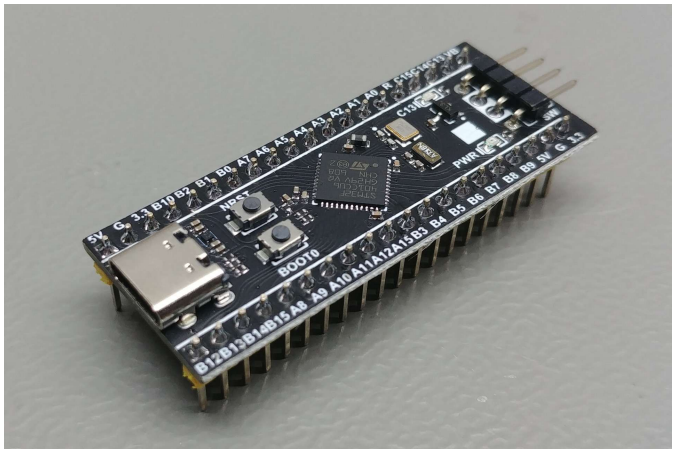

Рисунок 2.2 – Мікроконтролерна плата STM32F401CCUx

У розробленій системі плата на базі такого мікроконтролера виконує роль первинного сенсорного вузла. Він не займається ресурсомістким передаванням даних безпосередньо до наземної станції, натомість його головна задача — забезпечити строгі часові інтервали опитування периферії по шинах I2C та UART2, гарантуючи відсутність пропусків вимірювань, після чого сформувати єдиний структурований телеметричний кадр для відправки на наступний обчислювальний рівень.

Для подальшої маршрутизації та обробки телеметрії використовується одноплатний мікрокомп'ютер Raspberry Pi 3 Model B Rev 1.2 [16]. Основним каналом обміну даними між мікроконтролером та Raspberry Pi виступає інтерфейс USB CDC, який забезпечує високу надійність з'єднання, хоча класичний UART залишено як резервний канал для діагностики. Роль Raspberry Pi полягає у безперервному прийманні сформованого STM32 потоку, виконанні програмної фільтрації шумів та кінцевій підготовці параметрів

Зовнішній вигляд одноплатного комп’ютера Raspberry Pi 3 Model B Rev 1.2 наведено на рисунку 2.3.

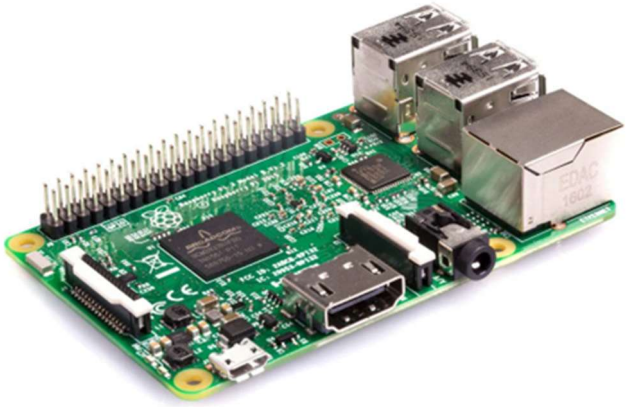

Рисунок 2.3 – Одноплатний комп’ютер Raspberry Pi 3 Model B Rev 1.2

Використання Raspberry Pi дозволяє відокремити задачі роботи з сенсорами від задач мережевого передавання. Це спрощує налагодження системи, оскільки роботу STM32, передавання через UART і пересилання даних до QGroundControl можна перевіряти окремими етапами. Крім того, наявність повноцінної операційної системи на базі Linux забезпечує гнучкість у виборі протоколів та дозволяє реалізовувати складні алгоритми буферизації без втрати загальної продуктивності.

Для отримання інерціальних параметрів у системі використовується модуль серії LSM6DS. Цей мікроелектромеханічний датчик інтегрує в одному корпусі триосьовий акселерометр та триосьовий гіроскоп, що дає змогу з високою точністю фіксувати лінійні прискорення та кутові швидкості. Ці динамічні показники важливі для безперервної оцінки просторового положення, крену, тангажу та загального вектора руху БПЛА. Модуль оснащений стандартним цифровим інтерфейсом і підключається до керівного мікроконтролера за допомогою шини I2C, що мінімізує кількість задіяних сигнальних ліній на платі.

Зовнішній вигляд інерціального модуля LSM6DSO наведено на рисунку 2.4.

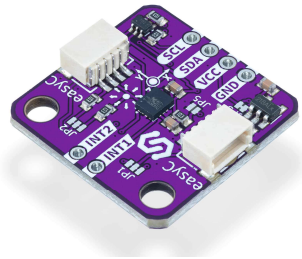

Рисунок 2.4 – Інерціальний модуль LSM6DSO

У розробленій системі цей інерціальний модуль виступає основним джерелом первинних даних про динаміку польоту. Мікроконтролер STM32 здійснює його безперервне циклічне опитування, після чого отримані сирі значення проходять базову перевірку на валідність і включаються до загального телеметричного пакета. Важливою особливістю роботи з такими датчиками є необхідність забезпечення жорстких таймінгів зчитування по шині I2C, щоб уникнути втрати перехідних процесів та спотворень під час різких маневрів апарата.

Оскільки гіроскоп схильний до накопичення похибки з часом, для точного визначення абсолютного напрямку та компенсації дрейфу курсу додатково використовується магнітометр GY-273 на базі мікросхеми HMC5883L. Цей датчик вимірює напруженість магнітного поля Землі по трьох осях, що дозволяє розрахувати магнітний азимут БПЛА. Він також підключається через інтерфейс I2C, що дає можливість ефективно використовувати спільну лінію зв'язку разом з інерціальним датчиком та барометром, оптимізуючи апаратні ресурси мікроконтролера. Проте під час розміщення магнітометра на борту необхідно враховувати його високу чутливість до наведених електромагнітних полів від силових кабелів та двигунів.

Зовнішній вигляд магнітометра HMC5883L наведено на рисунку 2.5.

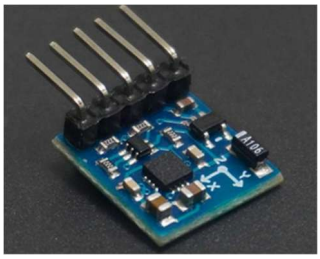

Рисунок 2.5 – Магнітометр HMC5883L

У складі телеметричної системи магнітометр доповнює інерціальні дані та дозволяє передавати на наземну станцію інформацію про орієнтацію апарата відносно магнітного поля.

Для отримання навігаційних даних використовується GPS-модуль NEO-6M [19]. Він забезпечує отримання координат, швидкості руху та службових даних про стан супутникової фіксації. Підключення GPS-модуля до STM32 виконується через UART, що відповідає типовому способу передавання навігаційних повідомлень.

Зовнішній вигляд GPS-модуля NEO-6M наведено на рисунку 2.6.

Рисунок 2.6 – GPS-модуль NEO-6M

У системі GPS-модуль використовується для формування координатної частини телеметрії. Окрім самих координат, важливо передавати також стан GPS-фіксації, оскільки за її відсутності навігаційні дані не повинні відображатися як достовірні.

Для вимірювання атмосферного тиску використовується барометричний датчик BMP280 [20]. На основі зміни тиску можна оцінювати відносну зміну висоти. Датчик є компактним, доступним і підтримує цифровий інтерфейс I2C, що робить його зручним для використання у складі макетної телеметричної системи.

Зовнішній вигляд барометричного датчика BMP280 наведено на рисунку 2.7.

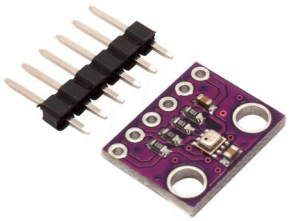

Рисунок 2.7 – Барометричний датчик BMP280

У розроблюваній системі BMP280 використовується для отримання барометричних параметрів, які передаються до Raspberry Pi та можуть бути використані для оцінки зміни висоти. Оскільки атмосферний тиск залежить не лише від висоти, а й від зовнішніх умов, ці дані потребують фільтрації та контролю стабільності.

2.3 Створення принципової електричної схеми системи

На основі структурної схеми було створено принципову електричну схему комп’ютерної системи бортової телеметрії БПЛА. Якщо структурна схема показує загальну архітектуру та напрям передавання даних, то принципова схема визначає конкретне підключення мікроконтролера, сенсорних модулів, вузла живлення, роз’ємів зв’язку та службових елементів налагодження.

Центральним елементом схеми є мікроконтролер STM32F401CCUx. До нього підключаються сенсорні модулі, які формують телеметричні параметри БПЛА, а також інтерфейси для передавання даних до Raspberry Pi. Мікроконтролер виконує зчитування даних із датчиків, первинну підготовку вимірювань і формування телеметричного пакета.

Для підключення сенсорів у схемі використано цифрові інтерфейси I2C та UART. Інерціальний модуль підключається через I2C, що забезпечує швидкий обмін даними під час зчитування прискорень і кутових швидкостей. Магнітометр і барометричний датчик підключаються до шини I2C, оскільки такий інтерфейс дозволяє використовувати спільні лінії SDA та SCL для кількох пристроїв із різними адресами. GPS-модуль підключається через UART, що відповідає типовому способу передавання навігаційних даних.

Передавання телеметричних даних від STM32 до Raspberry Pi реалізовано через окремий UART-інтерфейс. Через цей канал мікроконтролер передає сформовані пакети до проміжного вузла, який виконує фільтрацію, додаткову обробку та подальше передавання даних на комп'ютер оператора. Такий поділ дозволяє не перевантажувати STM32 мережевими задачами та залишити за ним роботу з бортовими сенсорами.

Живлення схеми організовано від джерела напруги +5 В. Для отримання рівня +3,3 В використовується стабілізатор напруги AMS1117-3.3. Ця напруга необхідна для живлення мікроконтролера та сенсорних модулів. У схемі також передбачено фільтрувальні конденсатори, які підвищують стабільність живлення та зменшують вплив пульсацій на роботу цифрових вузлів.

Для програмування та налагодження мікроконтролера передбачено роз'єм SWD. Він використовується для завантаження прошивки, перевірки роботи програми та пошуку помилок під час тестування. Додатково в схемі передбачено лінії BOOT0 і NRST, які дають змогу вибирати режим запуску та виконувати апаратне скидання мікроконтролера.

Принципову електричну схему комп’ютерної системи бортової телеметрії БПЛА наведено на рисунку 2.8.

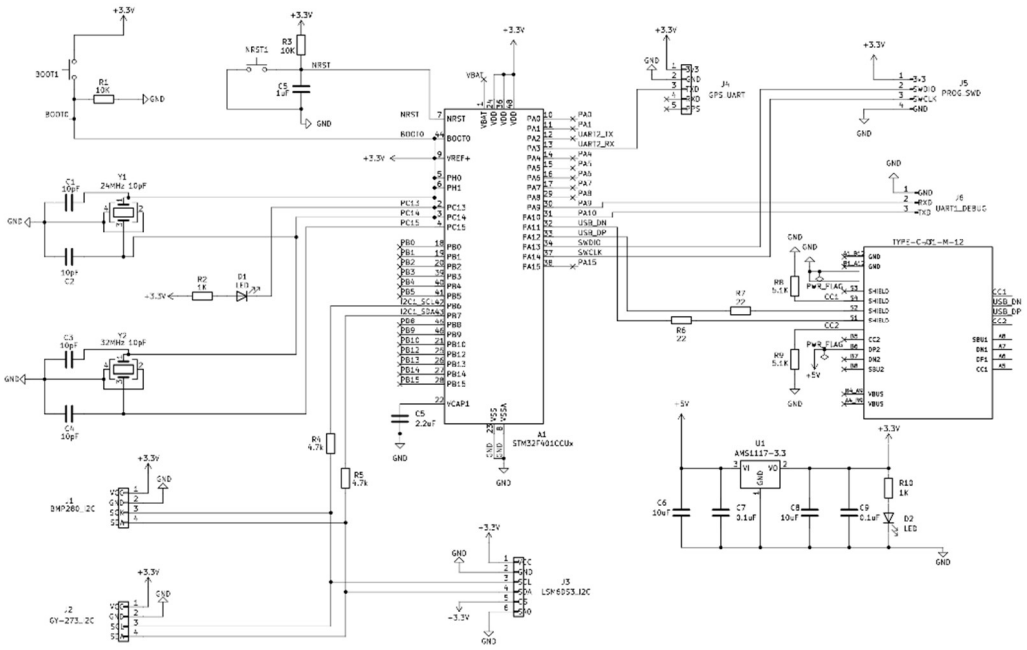

Рисунок 2.8 – Схема електрична принципова комп’ютерної системи бортової телеметрії БПЛА

Подана схема показує апаратну реалізацію бортового телеметричного вузла. Вона поєднує мікроконтролер STM32, вузол живлення, інтерфейси підключення сенсорів, GPS-модуля, каналу зв’язку з Raspberry Pi та засоби налагодження. Така побудова відповідає структурній схемі системи й забезпечує можливість подальшої програмної реалізації збору та передавання телеметричних даних.

2.4 Опис інтерфейсів і протоколів передавання даних

Для забезпечення високої надійності та швидкодії у комп’ютерній системі бортової телеметрії БПЛА застосовується багаторівнева структура інтерфейсів та протоколів обміну даними [6, 8]. Їх вибір безпосередньо продиктований розподіленою архітектурою комплексу. Сенсорні модулі

збирають фізичні показники та передають їх на мікроконтролер. Далі він формує базовий пакет і транслює його на Raspberry Pi, а одноплатний комп'ютер виконує фінальну маршрутизацію телеметрії на персональний комп'ютер (ПК) оператора для візуалізації в середовищі QGroundControl.

Збір первинної інформації від сенсорної периферії повністю реалізовано на базі єдиної послідовної шини I2C1. До цієї шини паралельно підключені барометричний датчик BMP280, магнітометр GY-273 та інерціальний модуль LSM6DSO. Таке схемотехнічне рішення дозволяє значно зекономити кількість задіяних виводів мікроконтролера, оскільки всі три пристрої використовують лише дві спільні лінії зв'язку. Апаратна роздільна здатність на шині забезпечується тим, що кожен модуль має власну унікальну адресу. Мікроконтролер по черзі звертається до необхідного датчика, швидко зчитуючи показники лінійних прискорень, кутових швидкостей, атмосферного тиску та магнітного поля без виникнення конфліктів на лінії.

Отримання навігаційних даних від приймача NEO-6M здійснюється через окремий асинхронний послідовний інтерфейс UART2. GPS-модуль безперервно генерує повідомлення у вигляді стандартизованого текстового потоку, який надходить на відповідні лінії мікроконтролера. Після програмного розбору цього потоку виділяються точні географічні координати, поточна швидкість апарата та службова інформація про стан супутникової фіксації.

Організація каналу зв'язку між основним збирачем даних (STM32F401CCUx) та вузлом мережевої маршрутизації (Raspberry Pi 3 Model B Rev 1.2) виконана за допомогою сучасного підключення. З програмної точки зору цей канал налаштований як віртуальний COM-порт на базі технології USB CDC.

Трансляція сформованого телеметричного потоку від Raspberry Pi до ПК оператора відбувається через бездротовий мережевий канал на базі протоколу UDP. Вибір протоколу транспортного рівня без встановлення постійного

з'єднання є найефективнішим для систем реального часу. Для телеметрії БПЛА критичною є мінімальна затримка та регулярність оновлення показників. Якщо один пакет даних загубиться під час передачі, система не витрачатиме час на його повторний запит, а просто дочекається наступного пакета з новішими даними.

Для безпосередньої взаємодії з програмним середовищем QGroundControl застосовується спеціалізований протокол MAVLink [12]. Raspberry Pi пакує зібрані мікроконтролером дані у валідні MAVLink-повідомлення, додаючи до них необхідні заголовки та контрольні суми. Кожен обраний інтерфейс вирішує своє вузьке завдання: I2C1 та UART2 забезпечують зчитування показників, USB CDC гарантує надійний міжапаратний зв'язок, UDP мінімізує мережеві затримки, а MAVLink забезпечує сумісність із наземною станцією.

Такий чіткий розподіл функцій між мікроконтролером жорсткого реального часу та мікрокомп’ютером із повноцінною операційною системою дозволяє уникнути вузьких місць у процесі обробки інформації. Якщо STM32 зосереджений виключно на підтримці апаратних таймінгів та безперебійному опитуванні датчиків, то Raspberry Pi бере на себе всі високорівневі завдання. Це включає не лише мережеве пакування даних, але й забезпечує потенційну можливість розширення системи. Наприклад, завдяки наявності обчислювального запасу на Raspberry Pi можна реалізувати локальний запис логів польоту або застосувати складніші алгоритми математичної фільтрації без ризику перевантажити мікроконтролер.

Кінцевою ланкою спроєктованого комунікаційного ланцюга є комп’ютер оператора із середовищем QGroundControl. Оскільки дані надходять у стандартизованому вигляді, програма автоматично розпізнає потік і відображає його у відповідних віджетах віртуальної приладової панелі, показуючи комплексний стан борту.

2.5 Проєктування алгоритму збору обробки та передавання телеметричних даних

Алгоритм роботи комп’ютерної системи бортової телеметрії визначає послідовність дій від запуску бортового вузла до передавання даних у QGroundControl. Він охоплює ініціалізацію апаратних модулів, зчитування параметрів із сенсорів, формування телеметричного пакета, передавання даних на Raspberry Pi та подальшу підготовку телеметрії для наземної станції.

Після запуску системи мікроконтролер STM32 виконує налаштування тактування, портів вводу-виводу та інтерфейсів обміну з підключеними модулями. Далі перевіряється готовність сенсорної частини. Якщо модуль відповідає на запити, він переходить у робочий режим. Якщо відповідь не отримана, у телеметрії фіксується відповідний службовий стан, щоб помилка окремого вузла не зупиняла роботу всієї системи.

В основному циклі STM32 послідовно зчитує дані з інерціального модуля, магнітометра, GPS-модуля та барометричного датчика. Отримані значення приводяться до узгодженого вигляду та об'єднуються в єдиний телеметричний пакет. До нього включаються також службові ознаки: маркери оновлення даних, стан фіксації супутників, доступність сенсорів на шині I2C і правильність отриманих значень. Така структура алгоритму гарантує відмовостійкість нижнього рівня: навіть при втраті зв'язку з одним із датчиків, формування пакета не переривається, а наземна станція отримує достовірну інформацію про апаратний стан борту. Наприкінці кожної ітерації передбачена програмна затримка, яка запобігає перевантаженню комунікаційних шин та підтримує стабільну частоту оновлення параметрів.

Блок-схему алгоритму збору, обробки та передавання телеметричних даних наведено на рисунку 2.9.

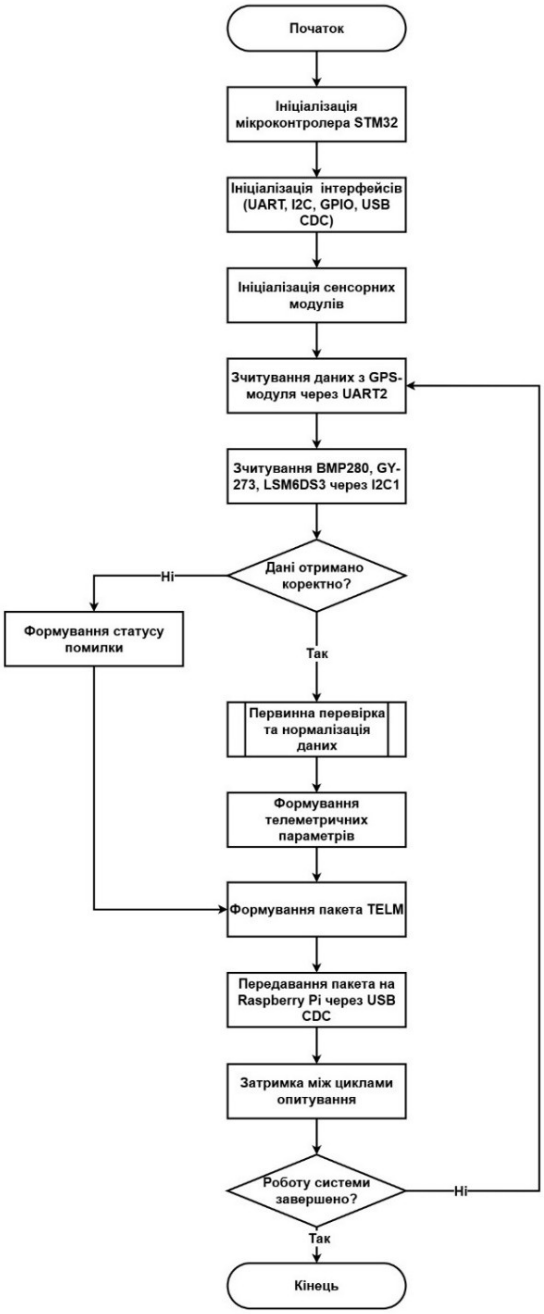

Рисунок 2.9 – Блок-схема алгоритму збору, обробки та передавання телеметричних даних

Після формування телеметричного пакета STM32 передає його на Raspberry Pi через USB CDC. На цьому рівні Raspberry Pi виконує розбір прийнятих даних, фільтрацію, додаткову обробку та підготовку параметрів до передавання на комп’ютер оператора. Такий розподіл функцій дозволяє

залишити за STM32 роботу з сенсорами, а задачі обробки та мережевого обміну перенести на проміжний вузол.

Raspberry Pi формує вихідний телеметричний потік і передає його через UDP або у форматі MAVLink. У випадку використання MAVLink дані можуть бути прийняті в QGroundControl, де перевіряється їх надходження, частота оновлення та основні значення параметрів. Якщо пакети не надходять протягом заданого часу або окремі поля не оновлюються, такий стан розглядається як затримка або втрата актуальності телеметрії.

Особливістю алгоритму є те, що система не припиняє роботу при недоступності одного з параметрів. Наприклад, за відсутності GPS-фіксації можуть продовжувати передаватися інерціальні та барометричні дані, але координати позначаються як недостовірні. Це потрібно для адаптивної візуалізації, оскільки оператор має бачити не лише значення параметрів, а й стан їх придатності для використання.

Основний цикл повторюється протягом усього часу роботи системи. STM32 регулярно формує телеметричні пакети, Raspberry Pi обробляє та пересилає їх, а QGroundControl приймає і відображає параметри стану БПЛА. Така послідовність забезпечує повний шлях проходження даних від сенсорних модулів до наземної станції.

2.6 Опис програмної організації роботи системи

Для уточнення роботи програмної частини комп’ютерної системи бортової телеметрії було розроблено UML-моделі, які відображають логіку функціонування системи та взаємодію її основних програмних компонентів. Вони дозволяють окремо показати стани роботи системи та склад програмних модулів, що беруть участь у зборі, обробці й передаванні телеметричних даних.

UML-діаграма станів показує основні етапи роботи системи після запуску. У ній відображено ініціалізацію STM32 та інтерфейсів обміну, зчитування даних GPS і сенсорних модулів, перевірку їх коректності, формування телеметричного пакета та його передавання на Raspberry Pi. У разі отримання некоректних даних система переходить у стан помилки й виконує повторне опитування модулів.UML-діаграму станів роботи комп'ютерної системи наведено на рисунку 2.10.

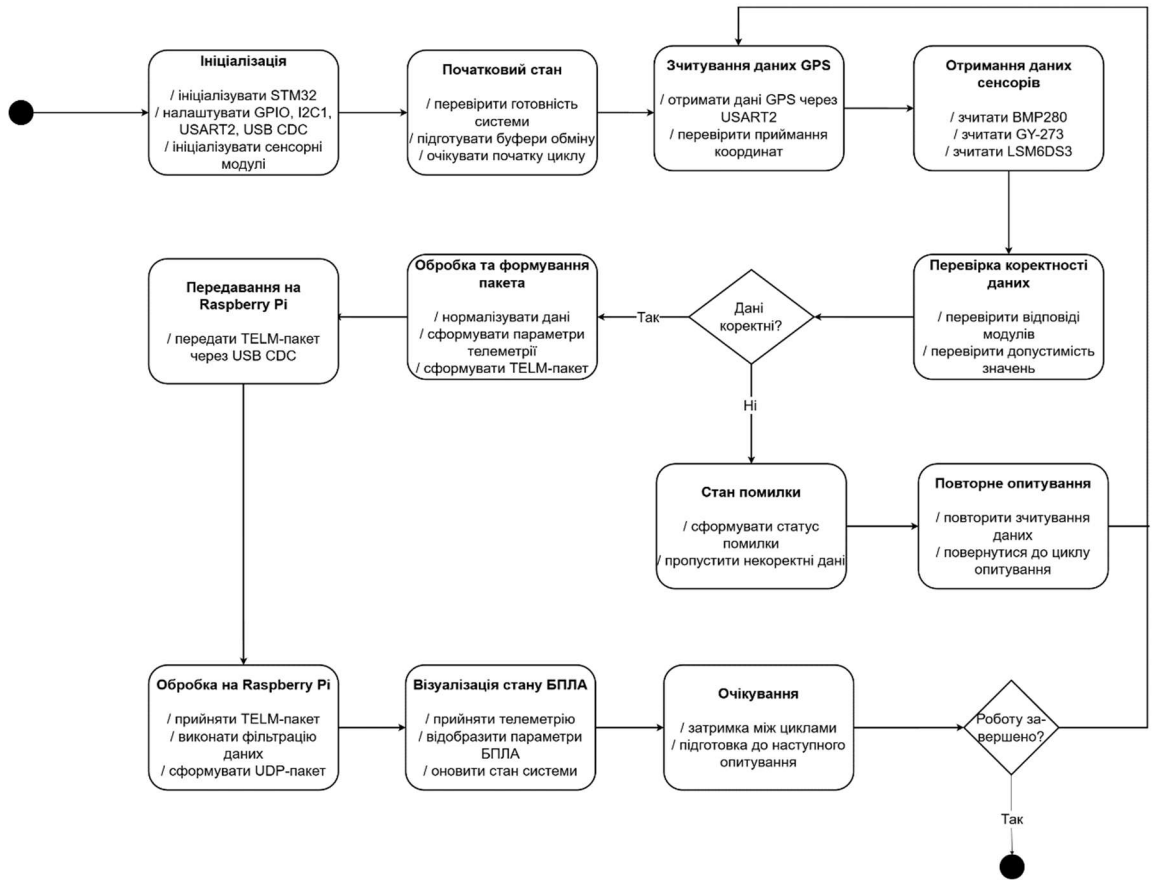

Рисунок 2.10 – UML-діаграма станів роботи комп'ютерної системи

UML-діаграма компонентів описує програмну структуру системи. Центральним компонентом є модуль TelemetrySystemCore, який координує роботу драйверів сенсорних модулів, виконує перевірку коректності даних і формує телеметричний пакет. Окремими компонентами подано модулі роботи з GPS, барометричним датчиком, магнітометром та інерціальним модулем.

Також у схемі відображено модуль обробки на Raspberry Pi та модуль візуалізації у QGroundControl.

UML-діаграму компонентів наведено на рисунку 2.11.

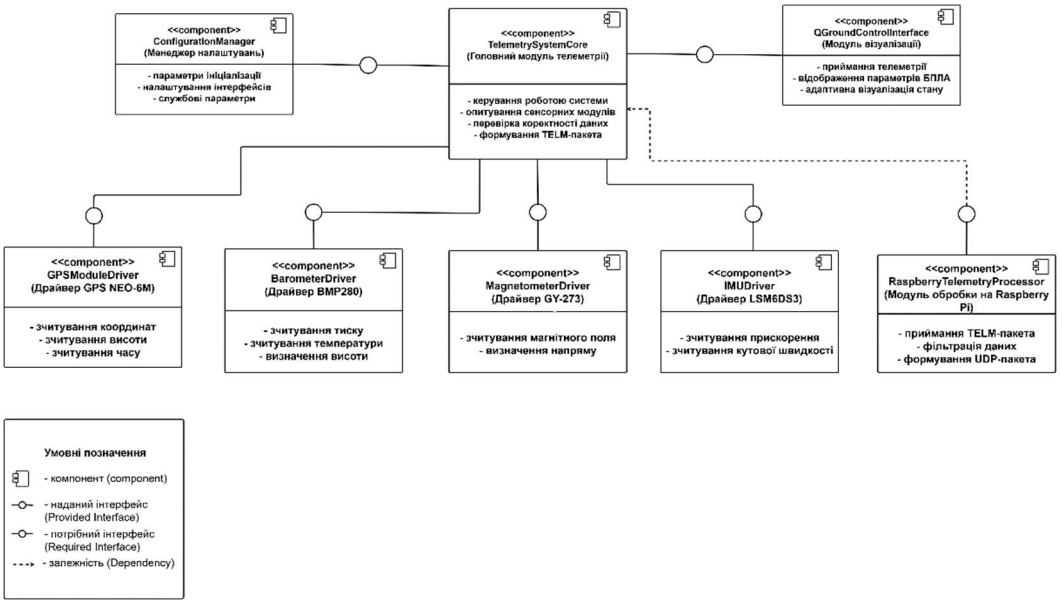

Рисунок 2.11 – UML-діаграма компонентів комп’ютерної системи

Подані UML-моделі доповнюють алгоритмічний опис системи. Діаграма станів показує, як система переходить між етапами опитування, перевірки, обробки та передавання даних, а діаграма компонентів уточнює, які програмні модулі забезпечують виконання цих функцій.

2.8 Проєктування адаптивної візуалізації телеметричних даних у QGroundControl

Після розроблення алгоритму збору, обробки та передавання телеметричних даних необхідно визначити спосіб їх подання на стороні наземної станції. У розроблюваній системі для цього використовується середовище QGroundControl, яке приймає телеметричні дані.

Адаптивна візуалізація полягає не лише у відображенні числових значень, а й у поданні стану самих даних. Для оператора важливо бачити не тільки координати, висоту, швидкість або орієнтацію, а й розуміти, чи є ці дані актуальними. Тому до телеметричного потоку додаються службові ознаки, які показують стан оновлення пакетів, наявність GPS-фіксації та працездатність окремих сенсорних модулів.

На стороні Raspberry Pi виконується приймання даних від STM32, їх попередня обробка, фільтрація та підготовка до передавання на комп'ютер оператора. Після цього телеметричні параметри передаються мережею у форматі, придатному для приймання в QGroundControl. Такий підхід дає змогу винести мережеву взаємодію на Raspberry Pi, а мікроконтролер STM32 залишити відповідальним за опитування сенсорів і формування первинного пакета даних.

Структура адаптивної візуалізації передбачає приймання телеметрії від Raspberry Pi, розбір отриманих параметрів, перевірку їх актуальності та передавання підготовлених значень в інтерфейс QGroundControl. Якщо телеметрія надходить стабільно, параметри відображаються у штатному режимі. Якщо пакет не оновлюється протягом визначеного часу, GPS-дані відсутні або один із сенсорів передає некоректні значення, відповідний параметр може бути позначений як недостовірний або неактуальний.

Такий спосіб відображення дозволяє оператору швидко та об'єктивно оцінювати загальний апаратний стан безпілотника. Замість безперервного аналізу окремих числових значень, користувач одразу бачить комплексну картину роботи.

Завдяки візуальним маркерам стану повністю виключається ризик того, що оператор прийматиме рішення на основі застарілих координат чи неактуальної висоти у разі тимчасової втрати зв'язку або апаратної відмови окремого сенсора..

Структуру модуля адаптивної візуалізації телеметричних даних наведено на рисунку 2.12.

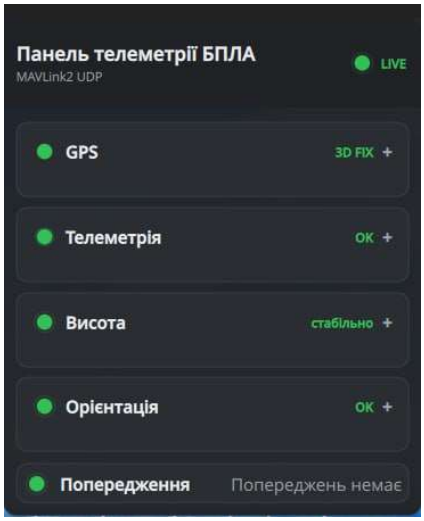

Рисунок 2.12 – Структура модуля адаптивної візуалізації телеметричних даних у QGroundControl

У штатному режимі QGroundControl відображає координати, швидкість, висоту, орієнтацію та інші параметри. Якщо телеметричний потік переривається, GPS фіксація відсутня або окремий сенсор передає нестабільні дані, інтерфейс повинен показати відповідний стан. При цьому QGroundControl не замінює роботу бортової частини, а використовується як наземна станція для приймання й відображення телеметрії, що надходить за ланцюгом, який складається послідовно з сенсорів, STM32, Raspberry Pi та самого комп'ютера.

Для реалізації такого підходу використовується колірна та текстова індикація статусних прапорців. Наприклад, зелений маркер свідчить про штатну роботу сенсорної шини, тоді як зміна кольору сигналізує про втрату зв'язку з конкретним модулем по I2C або відсутність валідних NMEA-повідомлень від навігаційного приймача Такий підхід до візуалізації кардинально знижує когнітивне навантаження, дозволяючи зосередитися на виконанні завдання та оперативному реагуванні на апаратні збої.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Розробка програмного забезпечення мікроконтролера STM32

Практична реалізація системи починається з розробки програмного забезпечення для мікроконтролера STM32. На цьому рівні виконується ініціалізація периферії, зчитування даних із підключених модулів, формування телеметричного повідомлення та передавання його на Raspberry Pi через послідовний інтерфейс.

Програмне забезпечення створено у середовищі STM32CubeIDE з використанням бібліотек HAL. У проєкті налаштовуються інтерфейси, необхідні для роботи з сенсорними модулями та передавання даних: I2C, UART, USB CDC і GPIO [14]. Шина I2C використовується для обміну з сенсорними модулями, USART2 — для приймання даних GPS-модуля, а USB CDC — для передавання сформованого телеметричного пакета на Raspberry Pi. Інтерфейс UART1 у системі залишено як резервний канал, який може використовуватися для налагодження, перевірки службових повідомлень або контролю роботи програми під час тестування.

Після генерації початкового коду до програми додаються функції опитування модулів, перевірки отриманих значень і формування вихідного телеметричного рядка. Основна логіка прошивки побудована циклічно: після запуску мікроконтролер налаштовує периферію, перевіряє доступність підключених модулів, зчитує дані з GPS і сенсорів, після чого формує пакет з телеметричними параметрами та службовими ознаками стану. Такий алгоритм гарантує, що у разі апаратної відмови окремого датчика мікроконтролер не зависне в очікуванні відповіді, а натомість продовжить безперервно передавати на Raspberry Pi актуальні дані.

Структуру проєкту програмного забезпечення мікроконтролера STM32 подано на рисунку 3.1.

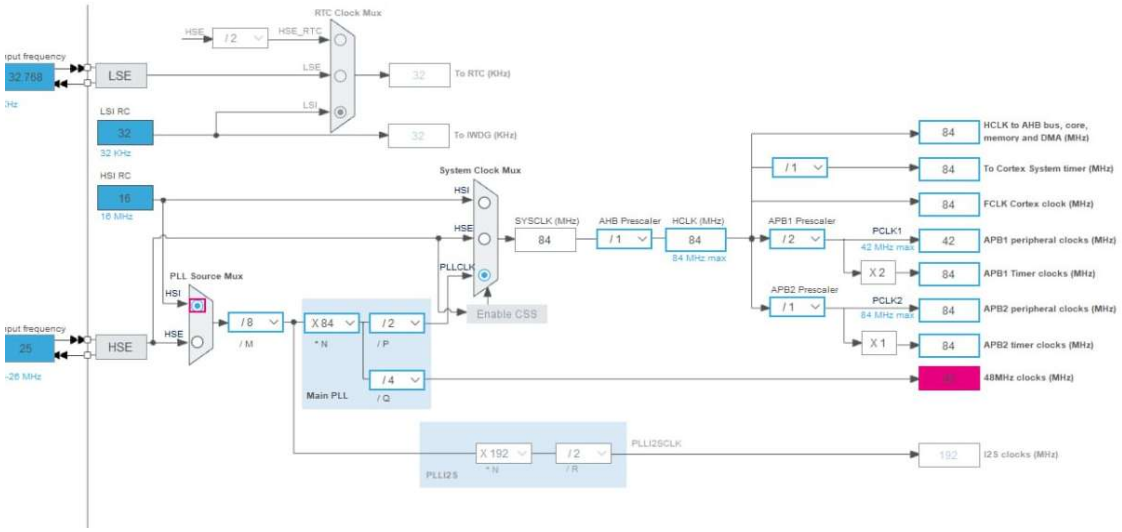

Рисунок 3.1 – Структура проєкту програмного забезпечення мікроконтролера STM32

У програмі мікроконтролера можна виділити кілька основних етапів роботи. Спочатку виконується налаштування системного тактування, портів вводу-виводу та інтерфейсів обміну. Далі програма переходить до ініціалізації підключених модулів. Якщо окремий модуль не відповідає або передає некоректні дані, цей стан фіксується у службових полях телеметричного повідомлення, але робота всієї системи не зупиняється.

Основний цикл програми виконує періодичне зчитування доступних параметрів. Дані з сенсорів приводяться до узгодженого вигляду та об’єднуються в один телеметричний пакет. До нього входять значення, які характеризують стан БПЛА, а також службові ознаки актуальності даних. Це потрібно для подальшої обробки на Raspberry Pi та відображення в QGroundControl.

Фрагмент коду формування телеметричного повідомлення наведено в лістингу на рисунку 3.2.

```
char telm_line[160];

snprintf(telm_line, sizeof(telm_line),
         "TELM,%lu,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld\r\n",
         HAL_GetTick(),
         ax_mg,
         ay_mg,
         az_mg,
         gx_mdps,
         gy_mdps,
         gz_mdps,
         pressure_pa,
         alt_cm,
         lat_e7,
         lon_e7,
         course_deg10);

HAL_UART_Transmit(&huart2,
                  (uint8_t *)telm_line,
                  strlen(telm_line),
                  HAL_MAX_DELAY);
```

Рисунок .3.2 – Фрагмент коду формування телеметричного повідомлення на STM32

Після формування повідомлення STM32 передає його на Raspberry Pi. На етапі прототипування для цього використовувався інтерфейс UART, проте у фінальній архітектурі основним каналом зв'язку виступає надійніший USB CDC. Мікроконтролер виконує виключно задачі нижнього рівня: працює з периферією, збирає дані та формує первинний пакет. Подальша фільтрація та передавання телеметрії на комп'ютер оператора виконуються вже на стороні Raspberry Pi.

Для перевірки роботи програми використовується виведення сформованих повідомлень у консоль. Такий діагностичний контроль дозволяє візуально пересвідчитися у правильності зчитування сенсорів та цілісності пакета ще до повної інтеграції борту з наземною станцією QGroundControl.

Приклад такого виведення наведено на рисунку 3.3.

```
=== UAV telemetry STM32 model started ===
Chain: Sensors -> STM32 -> Raspberry Pi -> UDP -> PC -> QGroundControl
TELM,time_ms,ax_mg,ay_mg,az_mg,gx_mdps,gy_mdps,gz_mdps,pressure_pa,alt_cm,lat_e7,lon_e7,course_deg10
TELM,123,-88,-93,9775,-489,-491,-494,100861,3911,495530000,255940000,0
TELM,1214,21,-29,9822,-390,-407,-437,100872,3818,495530003,255940005,0
TELM,2305,-70,35,9790,-291,-323,-379,100883,3726,495530006,255940010,0
TELM,3396,39,99,9837,-192,-239,-322,100893,3641,495530009,255940015,0
```

Рисунок 3.3 – Приклад формування телеметричних повідомлень мікроконтролером STM32

Як видно з наведеного термінального виведення, мікроконтролер формує структурований текстовий рядок, що починається з ідентифікатора «TELM». Ця послідовність містить ключові польотні дані, зокрема системний час, сирі значення з інерціального модуля, показники атмосферного тиску та навігаційні координати, які розділені комами для зручності подальшого програмного парсингу. Відсутність пропусків або спотворених символів у консолі свідчить про коректне налаштування тактування периферії.

3.2 Розробка програмного шлюзу Raspberry Pi для передавання телеметрії

Для передавання телеметричних даних між STM32 та наземною станцією використовується проміжний вузол на базі Raspberry Pi 3 Model B Rev 1.2. Його використання дозволяє фізично розділити процеси жорсткого реального часу та задачі мережевої маршрутизації. Raspberry Pi виступає в ролі інтелектуального шлюзу: він приймає безперервний потік байтів від мікроконтролера через інтерфейс USB CDC, здійснює його буферизацію, кінцеву обробку та готує дані до відправлення на комп'ютер оператора.

Перед початком роботи було перевірено модель використаної плати Raspberry Pi. Це потрібно для фіксації апаратної платформи, на якій реалізовано комунікаційний вузол системи.

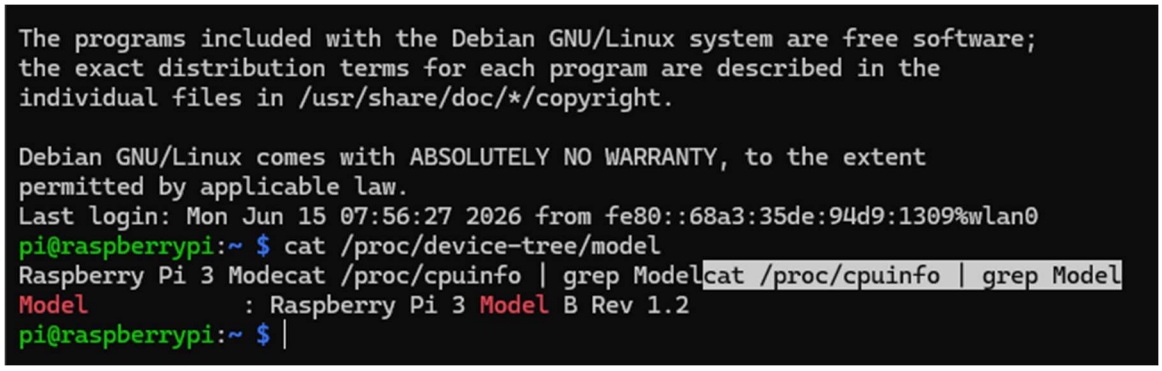

Рисунок 3.4 – Перевірка моделі використаної плати Raspberry Pi

Діагностичне виведення підтвердило, що комунікаційний вузол побудовано на базі Raspberry Pi 3 Model B ревізії 1.2. Обчислювальних потужностей цього мікрокомп'ютера цілком достатньо для забезпечення безперебійної роботи мережевих протоколів та виконання скриптів обробки даних без створення критичних затримок у передаванні телеметрії на наземну станцію.

Після апаратної верифікації було реалізовано сам програмний шлюз для приймання та трансляції показників. Його алгоритм роботи базується на безперервному читанні віртуального послідовного порту, який ініціалізується під час підключення STM32 по кабелю Type-C. Вхідний потік являє собою текстові масиви формату TELM. Програма на Raspberry Pi виконує посимвольне зчитування до символу кінця рядка, після чого розбиває отриманий кадр за розділювачами. На цьому етапі відбувається виділення окремих змінних: просторової орієнтації, висоти, широти, довготи та курсу. Після успішного парсингу скрипт перевіряє цілісність масиву, виконує необхідну математичну фільтрацію викидів і форматує змінні відповідно до структури мережевих пакетів для їх подальшого відправлення по UDP.

Фрагмент програмного шлюзу Raspberry Pi наведено в лістингу, який зображено на рисунку 3.5.

```
filters = TelemetryFilters()

master = mavutil.mavlink_connection(
    QGC_UDP,
    source_system=1,
    source_component=mavutil.mavlink.MAV_COMP_ID_AUTOPILOT1
)

for raw in raw_packets:
    processed = process_packet(raw, filters)
    send_mavlink(master, processed)

    print(
        "TX MAVLink | "
        f"t={processed.time_ms} ms | "
        f"alt={processed.alt_cm / 100.0:.2f} m | "
        f"lat={processed.lat_deg:.7f} | "
        f"lon={processed.lon_deg:.7f}"
    )

    time.sleep(SEND_INTERVAL_SEC)
```

Рисунок 3.5 – Фрагмент програмного шлюзу для передавання телеметрії через Raspberry Pi

Як видно з наведеного лістингу , основу програмного шлюзу становить безперервний цикл обробки вхідного потоку даних. На етапі ініціалізації створюється об'єкт фільтрації, який відповідає за згладжування шумів від апаратних датчиків, та налаштовується мережеве MAVLink-з'єднання з наземною станцією. Вибір протоколу UDP продиктований необхідністю швидкої трансляції телеметрії у режимі реального часу без встановлення жорстких сесій, що зменшує накладні витрати на передачу.

Паралельно з мережевим передаванням реалізовано виведення ключових метрик, таких як поточний час, висота та географічні координати, безпосередньо у термінал.

Приклад роботи програмного шлюзу Raspberry Pi під час активного приймання та обробки телеметричного потоку наведено на рисунку 3.6.

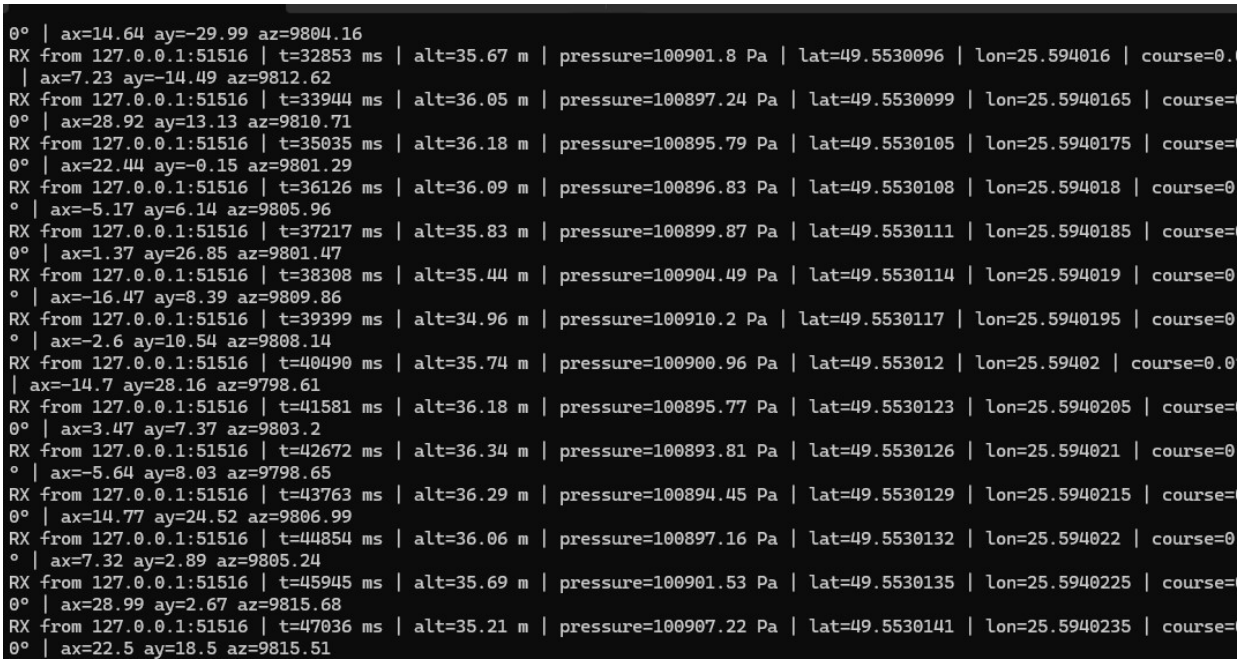

Рисунок 3.6 – Передавання телеметричних даних через Raspberry Pi

Наявність стабільного потоку повідомлень у терміналі, наочно підтверджує коректність роботи проміжного комунікаційного вузла. У консоль виводяться розпаковані значення ключових польотних параметрів: атмосферний тиск, розрахована барометрична висота, географічні координати та просторова орієнтація. Систематичне оновлення цих даних свідчить про те, що мікрокомп'ютер Raspberry Pi успішно синхронізується з потоком байтів від STM32.

Реалізований підхід підтверджує ефективність обраної розподіленої архітектури системи. Мікроконтролер нижнього рівня повністю звільнений від ресурсомістких завдань мережевої маршрутизації, зосередивши обчислювальні потужності виключно на опитуванні сенсорів I2C та підтримці жорстких таймінгів. Зі свого боку, Raspberry Pi, маючи повноцінну операційну систему, бере на себе всю роботу з фільтрацією, форматування пакетів за протоколом MAVLink та обслуговування мережевого стека.

3.3 Реалізація адаптивної візуалізації у програмі QGroundControl

На стороні наземної станції було реалізовано відображення телеметричних параметрів у середовищі QGroundControl. Для цього використовується користувацька панель, яка приймає підготовлені дані від Raspberry Pi та відображає основні параметри стану БПЛА у зручному для оператора вигляд.

Вікно панелі адаптивної візуалізації в середовищі QGroundControl наведено на рисунку 3.7.

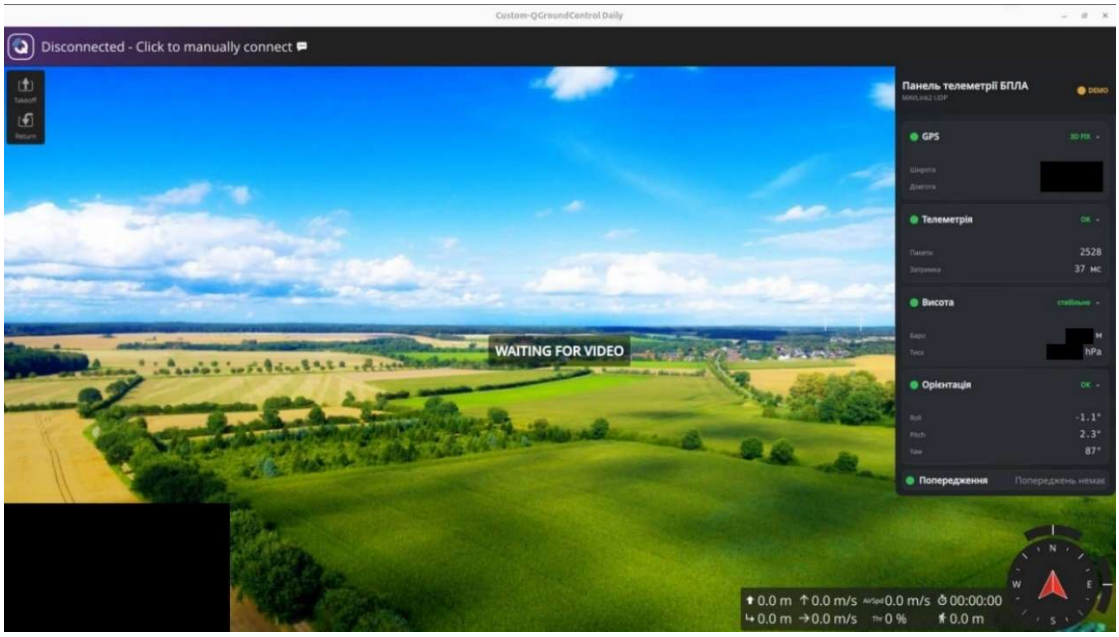

Рисунок 3.7 – Панель адаптивної візуалізації телеметричних параметрів у QGroundControl

Панель телеметрії розміщується поверх стандартного інтерфейсу QGroundControl і не змінює основну логіку роботи наземної станції [21]. Її задача полягає у виведенні параметрів, які надходять із бортової частини системи: GPS-координат, висоти, орієнтації, стану телеметричного каналу та службових повідомлень. Дані згруповано за призначенням, щоб оператор міг швидко оцінити стан окремих частин системи.

Панель відображає не лише числові значення, а й стан актуальності отриманих даних. У штатному режимі оператор бачить оновлювані параметри без додаткових попереджень. Якщо телеметричний потік переривається, GPS-фіксація відсутня або окремий параметр не оновлюється, відповідний блок переходить у попереджувальний стан. Завдяки цьому застарілі або недостовірні значення не подаються як актуальні.

Для перевірки надходження телеметрії також використовується вікно MAVLink Inspector, у якому можна побачити типи повідомлень, що приймаються QGroundControl, та частоту їх оновлення [13].

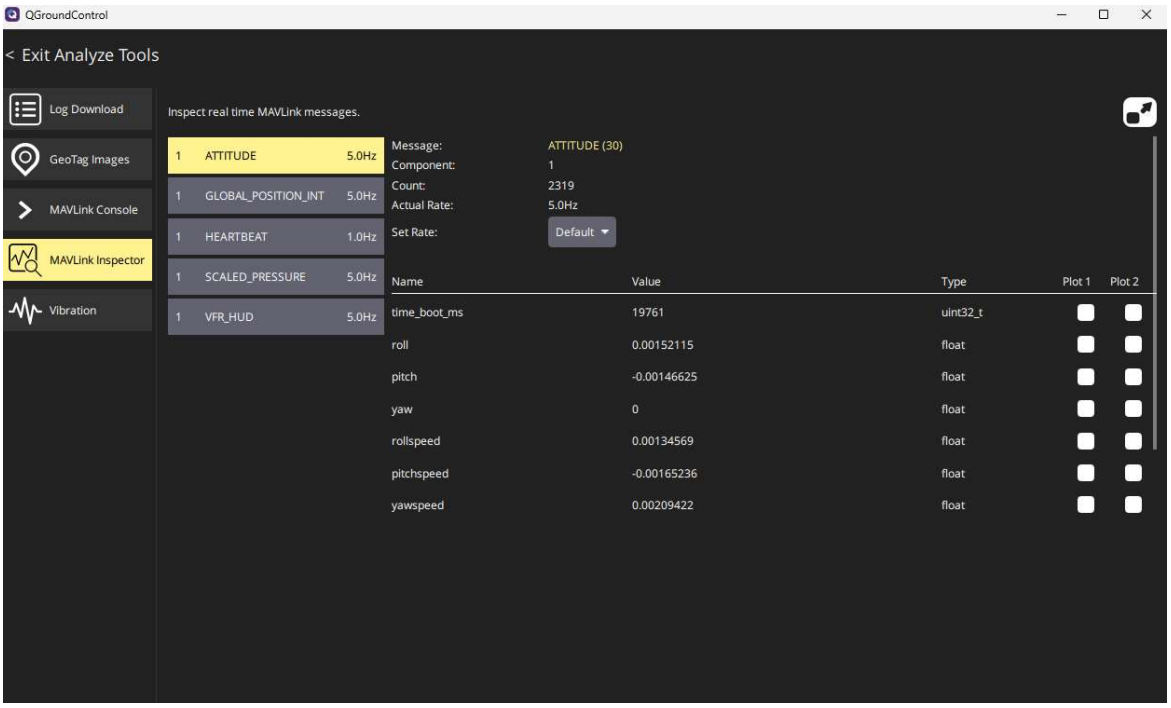

Рисунок 3.8 – Перевірка приймання MAVLink-повідомлень у QGroundControl

Наявність повідомлень у MAVLink Inspector підтверджує, що телеметричні дані проходять шлях від Raspberry Pi до наземної станції. Панель адаптивної візуалізації використовується для подання цих даних у більш зрозумілій формі: окремо для GPS, висоти, орієнтації, стану телеметричного каналу та попереджень.

3.4 Комплексне тестування апаратно-програмної системи

Після реалізації програмної частини системи було виконано комплексну перевірку проходження телеметричних даних через усі основні обчислювальні вузли.

На першому етапі тестування оцінювалася коректність роботи мікроконтролера. Перевірялося, чи правильно STM32 ініціалізує периферійні шини I2C та UART під час запуску, а також чи стабільно відбувається циклічне опитування підключених сенсорів. У терміналі послідовного порту фіксувалося формування базового телеметричного рядка у форматі TELM. Отримана в результаті симуляції послідовність повідомлень містила системний час, сирі інерціальні параметри, дані про атмосферний тиск і розраховані координати.

Другий етап був присвячений валідації комунікаційного моста на базі Raspberry Pi. Оскільки основним каналом передавання від STM32 обрано інтерфейс USB CDC, спочатку перевірялося розпізнавання пристрою в операційній системі та стабільність читання потоку байтів. Програмний шлюз успішно приймав вхідні масиви, виконував їх розбір і пакування у стандартизовані структури для трансляції. Контроль коректності роботи шлюзу здійснювався шляхом виведення діагностичної інформації у консоль мікрокомп’ютера.

Заключний третій етап охоплював перевірку наземної станції. Головним критерієм успіху була коректна інтерпретація вхідного мережевого потоку. У середовищі QGroundControl за допомогою вбудованого інструмента MAVLink Inspector контролювалася поява відповідних пакетів, а також візуальне оновлення значень у графічній панелі телеметрії. Це дозволило підтвердити працездатність наскрізного тракту: від генерації пакета мікроконтролером до його виведення на монітор оператора.

Результати перевірки основних вузлів зведено у таблиці 3.1.

Таблиця 3.1 – Результати тестування телеметричної системи

Етап перевірки	Предмет перевірки	Результат
Формування телеметрії на STM32	Наявність повідомлень формату TELM	Повідомлення формуються та виводяться в консоль
Передавання через Raspberry Pi	Приймання, обробка та пересилання даних	Дані передаються через UDP/MAVLink
Приймання в QGroundControl	Наявність MAVLink-повідомлень	Повідомлення відображаються в MAVLink Inspector
Відображення параметрів	Оновлення координат, висоти, орієнтації та станів	Параметри виводяться в панелі телеметрії
Перевірка службових станів	Втрата телеметрії, GPS-стан, нестабільні дані	Стани можуть бути відображені через попередження

Окрему увагу під час тестування мережевої взаємодії було приділено специфіці протоколу UDP. Враховуючи відсутність вбудованого механізму підтвердження доставки пакетів, перевірялася реакція наземної станції на штучно ініційовану затримку чи втрату частини даних. Експериментально підтверджено, що архітектура зв'язку коректно обробляє такі пропуски: замість зависання інтерфейсу відбувається плавне оновлення показників при надходженні наступного валідного пакета. Це доводить доцільність використання саме UDP для трансляції динамічної телеметрії, де затримка у часі є критичнішою, ніж втрата окремого кадру вимірювань.

У межах реалізації концепції адаптивного відображення тестування підтвердило успішне функціонування маркерів якості даних. Під час штучного відключення датчиків або втрати сигналу від GPS-модуля на стороні STM32, система продовжувала передавати телеметричний кадр, але зі зміненими статусними прапорцями. На екрані оператора в QGroundControl це відображалося як виведення попереджувальних повідомлень та індикація неактуальності показників. Такий підхід гарантує, що оператор БПЛА завжди має об'єктивну оцінку достовірності інформації та відрізняє реальні значення від затриманих або помилкових.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Заходи щодо забезпечення безпеки при проведенні дослідних робіт

Забезпечення безпеки під час проведення дослідних робіт є надзвичайно важливим завданням, оскільки будь-яка інженерна та технічна діяльність є потенційно небезпечною. Під час роботи небезпека може існувати як явно, так і приховано, що з часом може призвести до травм, професійних захворювань або погіршення загального самопочуття. Головна мета усіх заходів безпеки полягає у тому, щоб звести ризик виникнення небезпечних ситуацій до прийнятного рівня. Базові правові засади у цій сфері регулюються Законом України «Про охорону праці», який гарантує право кожного працівника на належні, безпечні та здорові умови праці [23]. Відповідно до законодавства, життя та здоров'я працівника визнаються найвищою соціальною цінністю, тому вони мають беззаперечний пріоритет над результатами будь-якої виробничої чи дослідницької діяльності. За створення умов, що відповідають нормативно-правовим актам, повну відповідальність несе роботодавець або керівник навчального закладу чи лабораторії [29].

Специфіка розробки комп’ютерної системи бортової телеметрії вимагає врахування цілого комплексу небезпечних та шкідливих виробничих факторів. Дослідницький процес включає взаємодію з електронними компонентами, мікроконтролерами, мікрокомп'ютерами та радіочастотними модулями, що супроводжується ризиком ураження електричним струмом. Відповідно до вимог електробезпеки, всі налагоджувальні роботи з платами STM32 та Raspberry Pi повинні проводитися на антистатичних килимках із застосуванням індивідуальних браслетів заземлення, оскільки розряди статичної електрики є небезпечними як для чутливої апаратури, так і для

людини при непередбачуваних пробоях ізоляції. Живлення дослідних стендів має здійснюватися через блоки живлення із захистом від короткого замикання та гальванічною розв'язкою від побутової електромережі. Будь-які маніпуляції з підключенням чи відключенням периферійних пристроїв, зміною конфігурації шин I2C або UART необхідно виконувати виключно при знеструмленому обладнанні.

Окрему увагу під час роботи з компонентами безпілотних літальних апаратів слід приділяти пожежній безпеці. Тестування систем часто вимагає використання літій-полімерних акумуляторних батарей, які відрізняються високою енергоємністю та схильністю до самозаймання при механічних пошкодженнях, перезарядженні або короткому замиканні. Згідно з правилами пожежної безпеки в Україні, у приміщеннях, де проводяться роботи з такою технікою, обов'язковою є наявість вуглекислотних вогнегасників. Їх використання дозволено для гасіння електроустановок під напругою, на відміну від порошкових або пінних аналогів, які можуть призвести до незворотного пошкодження електронних плат та посилення струмопровідності в зоні загоряння. Акумуляторні батареї під час циклів заряджання та розряджання повинні знаходитися у спеціальних вогнетривких захисних пакетах під постійним наглядом оператора.

Значну частину процесу створення системи займають монтажно-складальні роботи, зокрема пайка контактних з'єднань. Цей етап характеризується виділенням у робочу зону шкідливих хімічних речовин, зокрема парів свинцю, олова та каніфолі, які утворюють токсичний аерозоль. Для запобігання інтоксикації робоче місце повинно бути обладнане місцевою витяжною вентиляцією. Контроль за станом повітря у зоні дихання працівника, яка визначається як простір у радіусі п'ятдесяти сантиметрів від обличчя, є критичним параметром виробничої санітарії. Згідно з санітарними нормами, концентрація шкідливих домішок у цьому об'ємі не повинна перевищувати гранично допустимих значень [25]. Для підтримання нормального мікроклімату в лабораторії температура повітря має

підтримуватися на рівні двадцяти двох градусів за Цельсієм, а відносна вологість повинна становити від сорока до шістдесяти відсотків.

Проте сама лише наявність безпечного обладнання та витяжних систем не вирішує всіх проблем. Статистика вказує на те, що більшість інцидентів трапляється через нездатність персоналу розпізнати приховану небезпеку. Тому ключовим організаційним заходом є проведення регулярних інструктажів з охорони праці. Відповідно до типового положення про порядок проведення навчання, працівник або дослідник повинен пройти вступний інструктаж при допуску в лабораторію, а також первинний інструктаж безпосередньо на робочому місці [26]. Це навчання супроводжується перевіркою знань щодо правил експлуатації вимірювальних приладів, осцилографів, паяльних станцій та правил надання першої домедичної допомоги при ураженні електричним струмом або термічних опіках.

4.2 Профілактика захворювань, які викликані напруженням органів систем організму та невірним положенням тіла при роботі

Розробка програмного забезпечення для мікроконтролерів, написання скриптів обробки телеметрії та налаштування інтерфейсу QGroundControl вимагають тривалої і безперервної роботи за персональним комп'ютером. Постійне перебування у вимушеній сидячій позі в поєднанні з високим зоровим та нервово-емоційним навантаженням створює серйозний ризик розвитку професійних захворювань. Відповідно до державних санітарних правил і норм роботи з візуальними дисплейними терміналами, робота програміста та інженера-електроніка класифікується як праця з високим ступенем напруженості. Основний удар приймає на себе кістково-м'язова система, що призводить до остеохондрозу шийного та поперекового відділів хребта, а також міофасціального больового синдрому.

Коли людина сидить, схилившись над клавіатурою або платою, її міжхребцеві диски відчувають нерівномірний тиск, що у кілька разів

перевищує навантаження у положенні стоячи. Це призводить до порушення кровообігу, спазму м'язів спини та поступової деформації хрящової тканини. Другою шкідливою умовою є локальне м'язове перенапруження кистей рук. Тисячі одноманітних дрібних рухів під час набору коду провокують карпальний тунельний синдром, який проявляється запаленням сухожиль, затисканням серединного нерва, болем і онімінням пальців. Для запобігання цим патологіям критично важливим є ергономічне облаштування робочого місця. Робочий стіл повинен мати висоту від сімдесяти двох до сімдесяти п'яти сантиметрів, а його поверхня має не відблискувати світло. Крісло повинно обов'язково регулюватися по висоті сидіння та куту нахилу спинки, забезпечуючи підтримку попереку для збереження природного лордозу хребта. Клавіатуру і мишу слід розміщувати так, щоб передпліччя мали точку опори на столі або підлокітниках, що знімає статичне навантаження з плечового пояса [27].

Окрему групу ризику складають захворювання органів зору. Через необхідність годинами фокусувати погляд на дрібному шрифті програмного коду або схемах розпіновки компонентів на екрані монітора розвивається астенопія, відома також як комп'ютерний зоровий синдром. Цей стан характеризується швидкою втомлюваністю очей, відчуттям піску під повіками, почервонінням склер та тимчасовим зниженням гостроти зору. Причиною є не лише постійна напруга війкового м'яза ока, але й зниження частоти кліпання при високій концентрації уваги, що призводить до пересихання рогівки. Згідно з гігієнічними вимогами, екран монітора повинен знаходитися на відстані не менше шістдесяти сантиметрів від очей користувача, а його верхній край має розташовуватися на рівні очей або трохи нижче. Екран монітора має розташовуватися на зручній для користувача відстані, а його положення і яскравість повинні забезпечувати стабільне зображення без відблисків та надмірного напруження зору [27]. Загальне освітлення у приміщенні повинно доповнюватися місцевим під час роботи з

друкованими платами, при цьому коефіцієнт пульсації джерел світла не має перевищувати п'яти відсотків.

Ефективна профілактика зазначених захворювань неможлива без суворого дотримання режиму праці та відпочинку. Дослідник часто ігнорує втому під впливом нервово-емоційного напруження під час розв'язання складних алгоритмічних задач. Ефективна профілактика зазначених захворювань неможлива без дотримання регламентованих перерв протягом робочої зміни [27].Під час цих мікропауз категорично не рекомендується залишатися на робочому місці. Необхідно встати, змінити положення тіла, виконати базові вправи для розтягнення м'язів спини, шиї та плечового пояса. Для зняття зорової втоми ефективним є переведення погляду на віддалені об'єкти за вікном, часте кліпання та виконання кругових рухів очима. Заключним, але не менш важливим етапом профілактики є регулярне проходження медичних оглядів у профільних спеціалістів — невропатолога та офтальмолога, що дозволяє виявити ранні ознаки перенапруження організму ще до їх переходу в хронічну стадію та вчасно скоригувати умови праці.

Додатковим фактором, що пришвидшує втомлюваність під час розробки та налагодження бортових систем, є високе нервово-емоційне напруження. Для зниження когнітивного навантаження доцільно застосовувати принцип чергування видів діяльності: наприклад, змінювати етапи безперервного написання коду на фізичну роботу з апаратною частиною, пайку контактів або аналіз технічної документації. Додатковим фактором, що пришвидшує втомлюваність під час розробки та налагодження бортових систем, є фоновий шум від систем охолодження комп’ютерів і лабораторного обладнання. Рівень шуму на робочому місці має відповідати вимогам санітарних норм виробничого шуму, оскільки тривалий вплив акустичного навантаження посилює нервово-емоційне напруження, знижує концентрацію уваги та прискорює загальну втому [28].

ВИСНОВКИ

У кваліфікаційній роботі розроблено комп’ютерну систему бортової телеметрії БПЛА. Побудовано наскрізний шлях проходження даних: від первинних сенсорних модулів через мікроконтролер STM32 та проміжний вузол Raspberry Pi до наземного комп’ютера оператора. Такий розподіл завдань дозволив відокремити жорстко-реальночасове зчитування апаратних параметрів від мережевої передачі.

Нижній рівень системи реалізовано на базі STM32, до якого по шині I2C підключено датчик тиску BMP280, магнітометр GY-273 (HMC5883L) та інерціальний модуль LSM6DSO, а також GPS-приймач NEO-6M через UART2. Мікроконтролер опитує датчики, перевіряє валідність значень і формує первинний телеметричний пакет. Основним каналом передавання даних до Raspberry Pi визначено USB CDC через роз’єм Type-C, тоді як класичний UART1 залишено виключно як резервний та налагоджувальний інтерфейс.

Raspberry Pi виконує роль комунікаційного моста, який приймає пакети від мікроконтролера, виконує їх підготовку та транслює на комп’ютер оператора за допомогою протоколу UDP для мінімізації затримок. Приймання та адаптивна візуалізація польотних параметрів реалізовані в середовищі QGroundControl. Відображаються не лише самі метрики, а й оцінка їхньої актуальності, стан GPS-фіксації та статус працездатності сенсорів.

Проведені випробування підтвердили коректність проходження даних по всьому спроєктованому тракту: від фізичного зчитування до відображення в наземному інтерфейсі. Створена конфігурація показала стабільну роботу комунікаційних каналів і має практичне значення як готова підсистема збирання телеметрії, яку можна інтегрувати в існуючі комплекси моніторингу БПЛА без заміни їхньої апаратної бази.–

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жаровський Р.О., Луцик Н.С., Осухівська Г.М., Паламар А.М., Тиш Є.В. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль: ТНТУ, 2024. 39 с.

2. Микитишин А.Г., Митник М.М., Стухляк П.Д., Пасічник В.В. Комп'ютерні мережі. Книга 1. Львів: «Магнолія 2006», 2024. 256 с.

3. Паламар М.І., Стрембіцький М.О., Паламар А.М. Проектування комп'ютеризованих вимірювальних систем і комплексів. Навчальний посібник. Тернопіль: ТНТУ, 2019. 150 с.

4. Palamar A., Karpinski M., Palamar M., Osukhivska H., Mytnyk M. Remote Air Pollution Monitoring System Based on Internet of Things. CEUR Workshop Proceedings, 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022), Ternopil, Ukraine, November 22–24, 2022. Vol. 3309. P. 194–204.

5. Palamar A., Palamar M., Osukhivska H. Real-time Health Monitoring Computer System Based on Internet of Medical Things. CEUR Workshop Proceedings, 3rd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2023), Ternopil, Ukraine, Opole, Poland, November 22–24, 2023. Vol. 3628. P. 106–115.

6. Voloshchuk A., Osukhivska H. Adaptive multi-protocol communication for energy systems. Scientific Journal of TNTU. Ternopil: TNTU, 2025. Vol. 119, No. 3. P. 97–106. https://doi.org/10.33108/visnyk_tntu2025.03.097.

7. Оконський М.В., Лупенко С.А., Паламар А.М. Комп'ютерна система для моніторингу метеорологічних параметрів на основі IoT. Актуальні задачі сучасних технологій: збірник тез доповідей X міжнародної науково-практичної конференції молодих учених та студентів. Тернопіль: ТНТУ, 2021. С. 112.

8. Долгушин Я.В., Тиш Є.В. Розробка алгоритму адаптації протоколу Modbus для передачі даних у системи IIoT через протокол MQTT. Матеріали XIII науково-технічної конференції «Інформаційні моделі, системи та технології». Тернопіль: ТНТУ, 2025. С. 113.

9. Слюз І., Жаровський Р. Принципи та основні етапи комплексного тестування комп'ютерної інформаційної системи. Матеріали X науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології». Тернопіль: ТНТУ, 2022. С. 93.

10. QGroundControl Documentation. QGroundControl User Guide. URL: https://docs.qgroundcontrol.com/master/en/qgc-user-guide/ (дата звернення: 13.02.2026).

11. QGroundControl Developer Guide. QGroundControl Custom Build and Plugin Documentation. URL: https://docs.qgroundcontrol.com/master/en/qgc-dev-guide/ (дата звернення: 13.02.2026).

12. MAVLink Developer Guide. Micro Air Vehicle Communication Protocol. URL: https://mavlink.io/en/ (дата звернення: 14.02.2026).

13. MAVLink Common Message Set. MAVLink Documentation. URL: https://mavlink.io/en/messages/common.html (дата звернення: 14.02.2026).

14. STMicroelectronics. STM32F401xB/C and STM32F401xD/E advanced Arm-based 32-bit MCUs: Reference manual RM0368. Rev. 6. STMicroelectronics, 2025. URL: https://www.st.com/resource/en/reference_manual/rm0368-stm32f401xbc-and-stm32f401xde-advanced-armbased-32bit-mcus-stmicroelectronics.pdf (дата звернення: 15.02.2026).

15. STMicroelectronics. STM32F401CC. Product overview and datasheet. STMicroelectronics. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f401cc.html (дата звернення: 15.02.2026).

16. Raspberry Pi Ltd. Raspberry Pi Documentation. Hardware and configuration documentation. URL: https://www.raspberrypi.com/documentation/ (дата звернення: 20.04.2026).

17. STMicroelectronics. LSM6DSO iNEMO inertial module: datasheet. STMicroelectronics. URL: https://www.st.com/en/mems-and-sensors/lsm6dso.html (дата звернення: 21.04.2026).

18. Honeywell. 3-Axis Digital Compass IC HMC5883L: Data Sheet. Honeywell International Inc., 2013. 36 p.

19. u-blox. NEO-6 series: GPS modules data sheet. u-blox AG. URL: https://www.u-blox.com/en/product/neo-6-series (дата звернення: 24.04.2026).

20. Bosch Sensortec. BMP280 Digital Pressure Sensor: Data Sheet. Bosch Sensortec GmbH. URL: https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/bmp280/ (дата звернення: 03.05.2026).

21. Qt Group. Qt 6 Documentation. QML and Qt Quick documentation. URL: https://doc.qt.io/qt-6/ (дата звернення: 03.05.2026).

22. Kitware. CMake Documentation. URL: https://cmake.org/documentation/ (дата звернення: 04.05.2026).

23. Про охорону праці : Закон України від 14.10.1992 № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 23.05.2026).

24. Про затвердження Правил пожежної безпеки в Україні : наказ МВС України від 30.12.2014 № 1417. URL: https://zakon.rada.gov.ua/laws/show/z0252-15 (дата звернення: 23.05.2026).

25. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99 : постанова Головного державного санітарного лікаря України від 01.12.1999 № 42. URL: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 23.05.2026).

26. Про затвердження Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці : наказ Держнаглядохоронпраці України від 26.01.2005 № 15. URL: https://zakon.rada.gov.ua/laws/show/z0231-05 (дата звернення: 23.05.2026).

27. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями : наказ Міністерства соціальної політики України від 14.02.2018 № 207. URL: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 23.05.2026).

28. Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99 : постанова Головного державного санітарного лікаря України від 01.12.1999 № 37. URL: https://zakon.rada.gov.ua/rada/show/va037282-99 (дата звернення: 23.05.2026).

29. Правила безпечної експлуатації електроустановок споживачів : наказ Держнаглядохоронпраці України від 09.01.1998 № 4. URL: https://zakon.rada.gov.ua/laws/show/z0093-98 (дата звернення: 23.05.2026).

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем і програмної інженерії

Кафедра комп’ютерних систем та мереж

“Затверджую”

Завідувач кафедри КС

_____ Осухівська Г.М.

“ 2 ” лютого 2026 р.

Комп’ютерна система бортової телеметрії та адаптивної візуалізації стану БПЛА на основі платформи QGroundControl

ТЕХНІЧНЕ ЗАВДАННЯ

на _7_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп’ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студентка групи СІ-42
_____ к.т.н., доц. Лещишин Ю.З.	_____ Козачук К.О.
“ 2 ” лютого 2026 р.	“ 2 ” лютого 2026 р.

Тернопіль 2026

1. Назва та підстава для виконання роботи.

1.1. Комп’ютерна система контролю повітряного потоку в приміщенні.

1.2. Підставою для виконання кваліфікаційної роботи бакалавра (КРБ) є Наказ по Університету (№ 4/9-188 від 24.04.2026р.).

2. Виконавець

Студентка групи СІ-42, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Козачук К.О.

3. Мета роботи

Метою роботи є розроблення структури та програмно-апаратної частини комп’ютерної системи бортової телеметрії БПЛА з передаванням даних через Raspberry Pi та їх відображенням у середовищі QGroundControl.

4. Склад виробу

До складу системи повинні входити:

1. мікроконтролерний вузол збору телеметричних даних;
2. GPS-модуль;
3. барометричний датчик;
4. магнітометр;
5. інерціальний модуль;
6. одноплатний комп’ютер Raspberry Pi;
7. модуль живлення;

8. інтерфейси USB CDC, UART, I2C та UDP;
9. програмне забезпечення для обробки і передавання телеметрії;
10. комплект документації.

5. Технічні вимоги

5.1. Вимоги по призначенню

5.1.1. Комп’ютерна система повинна забезпечувати збирання телеметричних даних із сенсорних модулів і GPS-модуля.

5.1.2. Система повинна забезпечувати приймання навігаційних даних від GPS-модуля NEO-6M через інтерфейс UART.

5.1.3. Система повинна забезпечувати зчитування параметрів із барометричного датчика BMP280, магнітометра HMC5883L та інерціального модуля LSM6DSO через інтерфейс I2C.

5.1.4. Мікроконтролер STM32 повинен виконувати первинну перевірку отриманих даних і формування телеметричного пакета.

5.1.5. Передавання телеметричних даних від STM32 до Raspberry Pi повинно виконуватись через інтерфейс USB CDC з використанням роз’єму USB Type-C.

5.1.6. UART1 повинен бути передбачений як резервний канал для налагодження та перевірки роботи системи.

5.1.7. Raspberry Pi повинен забезпечувати приймання телеметричних пакетів, їх підготовку до передавання та пересилання на комп’ютер оператора через мережевий канал UDP.

5.1.8. На стороні комп’ютера оператора повинно бути передбачено приймання та відображення телеметричних параметрів у середовищі QGroundControl.

5.1.9. Система повинна передбачати контроль актуальності та коректності отриманих даних, зокрема стану GPS-фіксації та працездатності сенсорних модулів.

5.2. Вимоги до умов експлуатації

5.2.1. Система призначена для роботи в лабораторних умовах під час налагодження та демонстрації роботи телеметричного каналу.

5.2.2. Температура експлуатації повинна бути в межах від 0 до +40 °C.

5.2.3. Відносна вологість повітря — до 80 % при температурі +25 °C.

5.3. Конструктивні вимоги

5.3.1. Розроблення корпусу системи у кваліфікаційній роботі не передбачається.

5.3.2. При побудові системи необхідно передбачити розміщення роз’ємів живлення, програмування, налагодження та обміну даними.

5.3.3. Конструкція макету повинна забезпечувати доступ до основних компонентів під час тестування і налагодження.

5.3.4. Живлення вузлів системи повинно передбачати формування напруг +5 В та +3,3 В.

5.4. Вимоги до надійності

5.4.1. Система повинна забезпечувати стабільне передавання телеметричних даних між STM32, Raspberry Pi та комп’ютером оператора.

5.4.2. У разі отримання некоректних даних система повинна передбачати повторне опитування відповідного модуля без зупинки всієї роботи системи.

5.5. Вимоги метрології
5.5.1. Перевірка параметрів системи під час налагодження повинна виконуватись із використанням стандартних вимірювальних приладів та програмних засобів моніторингу.

6. Економічні показники

Собівартість макету системи повинна бути не більше 10000 грн.

7. Вимоги до документації

7.1. Конструкторська та програмна документація повинна відповідати вимогам ЄСКД, ДСТУ та вимогам до оформлення кваліфікаційної роботи бакалавра.

7.2. До складу документації повинно входити:

1. пояснювальна записка;
2. схема електрична структурна;
3. схема електрична принципова;
4. блок-схема алгоритму роботи системи;
5. UML-діаграма станів роботи системи;
6. UML-діаграма компонентів комп’ютерної системи;
7. перелік елементів.

8. Стадії та етапи розробки КРБ

8.1 Стадії та етапи виконання КРБ наведенні в таблиці 1.

Таблиця 1

№	Назва етапу	Строк виконання	
		початок	кінець
1	Технічне завдання	—	до 26.03.
2	Розділ 1	26.03.	10.06.
3	Розділ 2	28.03.	10.05.
4	Розділ 3	02.04.	13.04.
5	Розділ охорона праці	16.04.	27.04.
6	Нормоконтроль	21.05.	11.06.
7	Попередній захист	11.06. 26	18.06.26
8	Захист	з 20.06. 21	—

9. В дане ТЗ можуть вноситись зміни по узгодженню сторін.

Додаток Б

Переліки елементів

Додаток В

Код програми

```
#include "main.h"
#include "usb_device.h"
#include "usbd_cdc_if.h"  1.5 11шрифт

#define TELM_HEADER       0x54454C4D
#define SENSOR_OK         1
#define SENSOR_ERROR      0
#define MAX_ERROR_COUNT   5

typedef struct
{
    float latitude;
    float longitude;
    float altitude;
    float speed;
    uint8_t fix;
    uint8_t valid;
} GPS_Data_t;

typedef struct
{
    float pressure;
    float temperature;
    float mag_x;
    float mag_y;
    float mag_z;
    float acc_x;
    float acc_y;
    float acc_z;
    float gyro_x;
    float gyro_y;
    float gyro_z;

    uint8_t bmp280_status;
    uint8_t gy273_status;
    uint8_t LSM6DSO_status;
} Sensor_Data_t;

typedef struct
{
    uint32_t header;
    uint32_t time_ms;
    GPS_Data_t gps;
    Sensor_Data_t sensors;
    uint8_t system_status;
    uint16_t checksum;
} Telemetry_Packet_t;

I2C_HandleTypeDef hi2c1;
UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;

GPS_Data_t gps_data;
Sensor_Data_t sensor_data;
Telemetry_Packet_t telemetry_packet;
```

```
uint8_t gps_error_count = 0;
uint8_t bmp280_error_count = 0;
uint8_t gy273_error_count = 0;
uint8_t LSM6DSO_error_count = 0;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_USART2_UART_Init(void);

void BMP280_Init(void);
void GY273_Init(void);
void LSM6DSO_Init(void);

uint8_t GPS_ReadData(GPS_Data_t *gps);
uint8_t BMP280_ReadData(Sensor_Data_t *data);
uint8_t GY273_ReadData(Sensor_Data_t *data);
uint8_t LSM6DSO_ReadData(Sensor_Data_t *data);

uint8_t CheckTelemetryData(GPS_Data_t *gps, Sensor_Data_t *sensors);
uint16_t CalculateChecksum(uint8_t *data, uint16_t length);
void FormTelemetryPacket(Telemetry_Packet_t *packet);
void SendTelemetryPacket(Telemetry_Packet_t *packet);
void SendDebugMessage(char *message);
void ReinitSensorModules(void);

int main(void)
{
    HAL_Init();
    SystemClock_Config();

    MX_GPIO_Init();
    MX_I2C1_Init();
    MX_USART1_UART_Init();
    MX_USART2_UART_Init();
    MX_USB_DEVICE_Init();

    BMP280_Init();
    GY273_Init();
    LSM6DSO_Init();

    SendDebugMessage("System start\r\n");

    while (1)
    {
        if (GPS_ReadData(&gps_data) != SENSOR_OK)
        {
            gps_data.valid = 0;
            gps_error_count++;
        }
        else
        {
            gps_data.valid = 1;
            gps_error_count = 0;
        }

        if (BMP280_ReadData(&sensor_data) != SENSOR_OK)
        {
            sensor_data.bmp280_status = SENSOR_ERROR;
            bmp280_error_count++;
        }
```

```
        else
        {
            sensor_data.bmp280_status = SENSOR_OK;
            bmp280_error_count = 0;
        }

        if (GY273_ReadData(&sensor_data) != SENSOR_OK)
        {
            sensor_data.gy273_status = SENSOR_ERROR;
            gy273_error_count++;
        }
        else
        {
            sensor_data.gy273_status = SENSOR_OK;
            gy273_error_count = 0;
        }

        if (LSM6DSO_ReadData(&sensor_data) != SENSOR_OK)
        {
            sensor_data.LSM6DSO_status = SENSOR_ERROR;
            LSM6DSO_error_count++;
        }
        else
        {
            sensor_data.LSM6DSO_status = SENSOR_OK;
            LSM6DSO_error_count = 0;
        }

        if (CheckTelemetryData(&gps_data, &sensor_data) == SENSOR_OK)
        {
            telemetry_packet.system_status = SENSOR_OK;
        }
        else
        {
            telemetry_packet.system_status = SENSOR_ERROR;
        }

        if ((bmp280_error_count > MAX_ERROR_COUNT) ||
            (gy273_error_count > MAX_ERROR_COUNT) ||
            (LSM6DSO_error_count > MAX_ERROR_COUNT))
        {
            ReinitSensorModules();
        }

        FormTelemetryPacket(&telemetry_packet);
        SendTelemetryPacket(&telemetry_packet);

        HAL_Delay(1000);
    }
}

uint8_t GPS_ReadData(GPS_Data_t *gps)
{
    char rx_buffer[128];

    if (HAL_UART_Receive(&huart2, (uint8_t *)rx_buffer, sizeof(rx_buffer),
100) != HAL_OK)
    {
        return SENSOR_ERROR;
    }

    gps->latitude = ParseLatitude(rx_buffer);
```

```
    gps->longitude = ParseLongitude(rx_buffer);
    gps->altitude = ParseAltitude(rx_buffer);
    gps->speed = ParseSpeed(rx_buffer);
    gps->fix = ParseFixStatus(rx_buffer);

    if (gps->fix == 0)
    {
        return SENSOR_ERROR;
    }

    return SENSOR_OK;
}

uint8_t BMP280_ReadData(Sensor_Data_t *data)
{
    uint8_t raw_data[6];

    if (HAL_I2C_Mem_Read(&hi2c1, BMP280_ADDRESS, BMP280_DATA_REG,
                         I2C_MEMADD_SIZE_8BIT, raw_data, 6, 100) != HAL_OK)
    {
        return SENSOR_ERROR;
    }

    data->pressure = ConvertBMP280Pressure(raw_data);
    data->temperature = ConvertBMP280Temperature(raw_data);

    return SENSOR_OK;
}

uint8_t GY273_ReadData(Sensor_Data_t *data)
{
    uint8_t raw_data[6];

    if (HAL_I2C_Mem_Read(&hi2c1, GY273_ADDRESS, GY273_DATA_REG,
                         I2C_MEMADD_SIZE_8BIT, raw_data, 6, 100) != HAL_OK)
    {
        return SENSOR_ERROR;
    }

    data->mag_x = ConvertMagneticX(raw_data);
    data->mag_y = ConvertMagneticY(raw_data);
    data->mag_z = ConvertMagneticZ(raw_data);

    return SENSOR_OK;
}

uint8_t LSM6DSO_ReadData(Sensor_Data_t *data)
{
    uint8_t raw_data[12];

    if (HAL_I2C_Mem_Read(&hi2c1, LSM6DSO_ADDRESS, LSM6DSO_DATA_REG,
                         I2C_MEMADD_SIZE_8BIT, raw_data, 12, 100) != HAL_OK)
    {
        return SENSOR_ERROR;
    }

    data->acc_x = ConvertAccelerationX(raw_data);
    data->acc_y = ConvertAccelerationY(raw_data);
    data->acc_z = ConvertAccelerationZ(raw_data);

    data->gyro_x = ConvertGyroX(raw_data);
    data->gyro_y = ConvertGyroY(raw_data);
```

```
    data->gyro_z = ConvertGyroZ(raw_data);

    return SENSOR_OK;
}

uint8_t CheckTelemetryData(GPS_Data_t *gps, Sensor_Data_t *sensors)
{
    if (gps->valid == 0)
    {
        return SENSOR_ERROR;
    }

    if (sensors->bmp280_status == SENSOR_ERROR)
    {
        return SENSOR_ERROR;
    }

    if (sensors->gy273_status == SENSOR_ERROR)
    {
        return SENSOR_ERROR;
    }

    if (sensors->LSM6DSO_status == SENSOR_ERROR)
    {
        return SENSOR_ERROR;
    }

    if ((gps->latitude < -90.0f) || (gps->latitude > 90.0f))
    {
        return SENSOR_ERROR;
    }

    if ((gps->longitude < -180.0f) || (gps->longitude > 180.0f))
    {
        return SENSOR_ERROR;
    }

    if ((sensors->temperature < -40.0f) || (sensors->temperature > 85.0f))
    {
        return SENSOR_ERROR;
    }

    return SENSOR_OK;
}

void FormTelemetryPacket(Telemetry_Packet_t *packet)
{
    packet->header = TELM_HEADER;
    packet->time_ms = HAL_GetTick();

    packet->gps = gps_data;
    packet->sensors = sensor_data;

    packet->checksum = CalculateChecksum((uint8_t *)packet,
                                         sizeof(Telemetry_Packet_t) -
sizeof(uint16_t));
}

void SendTelemetryPacket(Telemetry_Packet_t *packet)
{
    CDC_Transmit_FS((uint8_t *)packet, sizeof(Telemetry_Packet_t));
}
```

```
void SendDebugMessage(char *message)
{
    HAL_UART_Transmit(&huart1, (uint8_t *)message, strlen(message), 100);
}

void ReinitSensorModules(void)
{
    if (bmp280_error_count > MAX_ERROR_COUNT)
    {
        BMP280_Init();
        bmp280_error_count = 0;
    }

    if (gy273_error_count > MAX_ERROR_COUNT)
    {
        GY273_Init();
        gy273_error_count = 0;
    }

    if (LSM6DSO_error_count > MAX_ERROR_COUNT)
    {
        LSM6DSO_Init();
        LSM6DSO_error_count = 0;
    }

    SendDebugMessage("Sensor reinitialization\r\n");
}

uint16_t CalculateChecksum(uint8_t *data, uint16_t length)
{
    uint16_t checksum = 0;

    for (uint16_t i = 0; i < length; i++)
    {
        checksum += data[i];
    }

    return checksum;
}

};
```