НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Інформаційних систем і технологій

(назва кафедри)

Швиденко М.З, к.е.н., проф.

(підпис) (ПІБ)

“16” травня 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

Система взаємодії польотних контролерів з наземною станцією керування місіями

Спеціальність 126 “Інформаційні системи та технології”

Гарант освітньої програми

к.е.н., доцент — (науковий ступінь та вчене звання)

(підпис)

Мокрієв Максим Володимирович — (ПІБ)

Керівник кваліфікаційної роботи

д.т.н., професор — (науковий ступінь та вчене звання)

(підпис)

Смолій Вікторія Миколаївна — (ПІБ)

Виконав (підпис) Гринько Владислав Миколайович (ПІБ)

Київ 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Завідувач кафедри інформаційних систем і технологій

Швиденко М.З, к.е.н., проф.

Підпис ініціали та прізвище_____2025 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

студенту(ці) Гриньку Владиславу Миколайовичу

Спеціальності 126 “Інформаційні системи та технології”

1. Тема роботи: “ **Система взаємодії польотних контролерів з наземною станцією керування місіями**”
   Затверджена наказом ректора від 16.12.2024 р. № 2245-С
2. Термін подання завершеної роботи на кафедру - *(01.06.2025)*
3. Вихідні данні: Дані телеметрії: температура, швидкість, висота, відео
4. Перелік питань, що розглядаються:
   1. Аналіз предметної області .
   2. Вивчення протоколів передачі даних
   3. Проєктування інформаційної.
   4. Реалізація та тестування прототипу
5. Календарний план

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	5 лютого 2025	
2	Вивчення протоколів передачі даних.	15 лютого 2025	
3	Проєктування інформаційної системи.	13 березня 2025	
4	Реалізація та тестування прототипу	15 квітня 2025	

Керівник кваліфікаційної роботи _____/ професор, д.т.н. В.М.Смолій
підпис ПІБ, вчене звання та ступінь

Завдання прийняв до виконання _____/ _____
підпис ПІБ

Дата отримання завдання 16.12.2024

РЕФЕРАТ

Тема бакалаврської кваліфікаційної роботи: “ Система взаємодії польотних контролерів з наземною станцією керування місіями”.
Автор роботи: Гринько Владислав Миколайович.
Керівник роботи: Смолій Вікторія Миколаївна.

Пояснювальна записка: 61 с., 20 джерел., 10 додатків.

Графічна частина: 15 презентаційних слайдів.

Метою роботи є дослідження особливостей взаємодії польотних контролерів із наземною станцією керування місіями.

У роботі проаналізовано існуючі процеси збору, обробки та передачі телеметричних даних у різних типах місій, а також розглянуто технічні та організаційні фактори, що впливають на ефективність цієї взаємодії. Для візуалізації та аналізу функціональних аспектів системи використано методи моделювання та сучасні програмні інструменти. Окрім цього, обґрунтовано можливі напрямки розвитку інформаційних систем керування польотами для підвищення їх надійності та оперативності у майбутніх космічних проектах.

Зміст

ВСТУП

Сучасні космічні місії вимагають високої надійності, точності та оперативності у процесах управління польотами. Наземна станція керування місіями та польотні контролери відіграють ключову роль у забезпеченні безпеки і ефективності виконання космічних операцій. З огляду на зростаючу складність космічних апаратів, а також збільшення обсягу телеметричних даних, стає необхідним створення інноваційних інформаційних систем, які забезпечать інтегровану взаємодію між польотними контролерами та наземною інфраструктурою. Це дозволить покращити якість моніторингу, прийняття рішень та реагування на непередбачені ситуації.

Метою роботи є розробка інформаційної системи взаємодії польотних контролерів із наземною станцією керування місіями, що дозволить автоматизувати процеси збору, обробки та візуалізації телеметричних даних, а також підвищити оперативність прийняття управлінських рішень. Для досягнення цієї мети поставлено такі завдання:

-Провести аналіз предметної області, описати ключові функції польотних контролерів та існуючі інформаційні технології;

-Сформувати технічні вимоги та функціональне завдання для інформаційної системи;

-Спроєктувати архітектуру системи та основні компоненти;

-Реалізувати програмний продукт із використанням сучасних мов програмування та інструментів;

-Провести тестування розробленої системи та оцінити її ефективність.

-Об’єкт і предмет дослідження

-Об’єктом дослідження є процеси управління космічними місіями на наземній станції керування. Предметом дослідження – інформаційна система, що забезпечує взаємодію польотних контролерів із наземною інфраструктурою шляхом збору, обробки та аналізу телеметричних даних.

Методи дослідження

У роботі використано системний аналіз предметної області, методи об’єктно-орієнтованого моделювання (Use Case, Activity, Sequence діаграми), а також програмні засоби для розробки інформаційних систем (Java, SQLite, JSON-парсинг). Для оцінки ефективності системи застосовано методи функціонального тестування за реальними сценаріями використання.

Структура роботи

Робота складається з вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У першому розділі наведено аналіз предметної області та наявних технологій. Другий розділ присвячено проєктуванню системи, третій – її реалізації, четвертий – тестуванню та оцінці ефективності. У висновках підсумовано результати дослідження та визначено перспективи подальшого розвитку.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Сучасне космічне господарство характеризується високим рівнем технічної складності, багаторівневою взаємодією систем та підвищеними вимогами до надійності функціонування. Одним із ключових елементів такої системи є інформаційна взаємодія між наземними станціями управління польотами і польотними контролерами. Саме ця взаємодія забезпечує нагляд, керування та реагування на усі події, що відбуваються під час космічної місії.

Узагальнено, наземна станція управління (НСК) -це комплекс технічних і програмних засобів, який забезпечує зв’язок із космічним апаратом, приймає телеметричні дані з борту, обробляє їх, зберігає та візуалізує оператору, а також формує команди для відправлення на борт. У свою чергу, польотний контролер -це фахівець, який відповідає за конкретний функціональний напрям місії (навігація, терморегуляція, електроживлення, зв’язок тощо) і ухвалює рішення на основі отриманих даних.

Структура наземної станції

Наземна інфраструктура управління складається з:

Антенних комплексів (S/X/Ka діапазонів), що забезпечують безперервне двостороннє радіоз’єднання;

Телеметричних декодерів, що розпаковують пакети даних згідно з протоколами CCSDS;

Центру обробки даних, де відбувається агрегація, фільтрація, перевірка даних на повноту;

Інтерфейсів візуалізації, що передають контролеру структуровану інформацію у вигляді графіків, таблиць, сигналів тривоги;

Серверів архівації, де зберігається повна історія місії у форматі журналів.

Кожен з цих компонентів функціонує синхронно у складі загальної системи управління, де затримки, похибки або втрата навіть одного модуля можуть мати катастрофічні наслідки.

Роль польотного контролера

Польотні контролери працюють у тривахтовому режимі, контролюючи місію 24/7. Їхня відповідальність поділена за напрямами:

FDO (Flight Dynamics Officer) -аналіз орбіти, траєкторій, швидкості та орієнтації апарата;

EECOM (Electrical, Environmental and COMmunication) -моніторинг енергосистем, температури, життєзабезпечення;

CAPCOM (Capsule Communicator) -єдиний оператор, що спілкується з екіпажем (для пілотованих місій);

BOOSTER, GUIDANCE, тощо -технічні контролери окремих вузлів.

Контролери працюють із інформацією, що надходить в реальному часі, і повинні миттєво реагувати на потенційні загрози. Наприклад, якщо тиск у паливній магістралі знизився нижче 30% -слід негайно вимкнути подачу та активувати резервну гілку. Такі дії відбуваються під тиском часу, з урахуванням строгих Flight Rules -формалізованих сценаріїв реагування.

Характер взаємодії

Взаємодія між контролерами й НСК -це циклічний процес, який передбачає:

-Прийом телеметричних даних (від 1 до 100 Гб/день для складної місії);

-Обробку даних -перевірка на наявність збоїв, візуалізація;

-Аналіз поточних значень -відображення критичних відхилень;

-Формування та відправлення керуючих команд;

-Протоколювання всіх дій у лог-файли;

-Генерацію звітів або доповідей для командування місії.

Цей цикл повторюється щосекундно, без перерв. Під час критичних фаз (вихід на орбіту, корекція траєкторії, стикування) -кількість команд і телеметричних пакетів зростає в рази.

Вимоги до інформаційної системи

Програмне забезпечення, що забезпечує цю взаємодію, повинно мати:

-Інтерфейс реального часу -оновлення параметрів без затримки;

-Гнучку візуалізацію -можливість створення власних панелей;

-Фільтрацію та пошук -миттєвий доступ до потрібних параметрів;

-Алертинг -система повідомлень при перевищенні порогових значень;

-Можливість ручного введення команд -з журналюванням та підтвердженням;

-Автоматичне збереження даних -логування з високою роздільною здатністю (до 10 Гц).

Особливої уваги заслуговує людиномашинний інтерфейс (HMI). Надлишкова інформація, неструктурованість чи візуальна складність можуть спричинити перевантаження оператора, що призведе до помилок. Саме тому використовують принципи UI/UX-дизайну для критичних систем:

-мінімалізм та ієрархія відображення;

-кольорова індикація (зелений/жовтий/червоний);

-аудіосигнали з різною гучністю залежно від рівня тривоги;

-блокування дій при неправильному форматі команди.

Людський фактор

Численні дослідження NASA вказують, що понад 50% помилок в управлінні пов’язані з людським фактором -втомою, недоглядом, помилковою інтерпретацією даних. Саме тому взаємодія між людиною й системою має враховувати:

-ергономічність інтерфейсу;

-адаптацію до типових сценаріїв (навчання на тренажерах);

-можливість миттєвої відміни або дублювання команд;

-історію всіх дій (audit trail) з мітками часу.

-Підсумки

Предметна область управління польотом охоплює багаторівневу взаємодію -від фізичної передачі сигналів до прийняття складних рішень людиною. Вона поєднує інженерні, програмні, аналітичні та психологічні аспекти, а її ефективність напряму впливає на успішність космічних місій.

Розуміння цієї області дозволяє визначити вимоги до інформаційної системи, яка має не просто збирати дані, а й бути інструментом безпеки, ефективності та надійності.

1.2 Аналіз наявних інформаційних технологій

На сучасному етапі розвитку аерокосмічної галузі надзвичайно важливу роль у забезпеченні стабільного функціонування космічних місій відіграють інформаційні системи, які дозволяють організувати безперервну взаємодію між наземними станціями управління та космічними апаратами. Ці системи

реалізують прийом, обробку, архівацію та візуалізацію телеметричних даних, а також формування команд керування.

У цьому розділі розглядаються найбільш відомі й широко застосовувані програмно-апаратні комплекси, які використовуються в провідних космічних агентствах, таких як NASA, ESA, JAXA, а також у приватному секторі -SpaceX, Blue Origin, Rocket Lab. Крім того, здійснюється порівняльний аналіз технологій із точки зору їх функціональності, архітектури, відкритості, зручності використання та масштабованості.

SCOS-2000 (ESA)

Одним із найбільш відомих рішень є SCOS-2000 (Spacecraft Control and Operations System) -власна розробка Європейського космічного агентства. Це масштабована система, яка слугує головною платформою для управління переважною більшістю європейських супутників, як на низьких, так і на геостаціонарних орбітах.

-Ключові характеристики SCOS-2000:

-побудована за принципом клієнт-сервер;

-підтримує специфікацію ECSS-E-70-31A (європейський стандарт телеметрії);

-інтегрує консоль керування, редактор бази даних команд і параметрів, журнали телеметрії;

-модульна структура -легко адаптується під конкретну місію.

SCOS-2000 має інтерфейси для підключення зовнішніх систем моделювання, тренажерів, а також спеціалізованих модулів для автоматичного тестування. Система використовується не лише ESA, а й національними агенціями (DLR, CNES) та в академічних проєктах.

Недоліком SCOS-2000 є закритість коду та складність адаптації для нових користувачів -система має високу вартість навчання та потребує значного досвіду для інтеграції.

Open MCT (NASA)

У противагу складним монолітним системам NASA запропонувала Open Mission Control Technologies (Open MCT) -сучасну вебплатформу для візуалізації, аналізу та контролю місій. Розроблена у NASA Ames Research Center, ця система відкрита для використання, активно підтримується спільнотою та інтегрується з іншими інструментами агентства.

-Переваги Open MCT:

-веб-орієнтований інтерфейс (працює в браузері);

-дашборди з реального часу: графіки, табличні звіти, візуальні трекери;

-підтримка JSON, REST API, WebSocket для потокової телеметрії;

-можливість кастомізації під конкретні потреби користувача;

-open source ліцензія.

Open MCT уже використовується у проєктах Mars Helicopter, CubeSat missions, а також для внутрішнього моделювання місій у центрах NASA.

Основним недоліком є потреба в додатковому серверному конфігуруванні, обмежена документація для нових користувачів, складність у підключенні нестандартних форматів телеметрії (не CCSDS).

COSMOS(Ball Aerospace) -ще одна потужна, але доступна для публіки open-source система. Вона була створена компанією Ball Aerospace як гнучка платформа для створення середовищ управління місіями та тестування підсистем.

-Особливості COSMOS:

-написана на Ruby з веб-інтерфейсом;

-підтримує протоколи TCP, UDP, Serial;

-має власну мову опису телеметрії;

-зручний інтерфейс для конфігурування параметрів;

-активна спільнота на GitHub.

COSMOS широко використовується в університетах, стартапах, CubeSat-ініціативах та навіть у військових застосуваннях для телеметрії дронів.

Однак її слабкими сторонами є відсутність повноцінної підтримки з боку комерційних провайдерів, менш зручна візуалізація даних порівняно з Open MCT, обмеження по масштабуванню для великих місій.

Інші системи: MCC-X, ECLIPSE, GMSEC, MCS

Окрім згаданих платформ, на ринку представлені ще десятки рішень:

-GMSEC (NASA GSFC) -middleware для взаємодії компонентів;

-ECLIPSE -французька модульна система моніторингу;

-MCS (Mission Control System) від SpaceX -закритий програмний комплекс, орієнтований на Falcon/Dragon.

Більшість таких систем не є загальнодоступними, проте інформація про їхню архітектуру дозволяє визначити спільні риси: високий ступінь автоматизації, використання модулів Machine Learning, паралельна обробка великих потоків даних, підтримка сценаріїв реагування на аномалії.

Таблиця 1.2.1-Порівняння систем

Система	Open Source	Веб-інтерфейс	Масштабованість	Порог входу	Використання
SCOS-2000	ні	так	Висока	Високий	ESA, CNES
Open MCT	так	так	Середня	Середній	NASA, приватні
COSMOS	так	так	Низька/Середня	Низький	Освіта, CubeSat

Висновки з аналізу.

Із проведеного огляду видно, що сучасні інформаційні системи для управління місіями поділяються на монолітні спеціалізовані (SCOS-2000, MCS) та відкриті модульні (Open MCT, COSMOS). Перші забезпечують максимальну надійність та масштабованість, але потребують високих ресурсів і досвіду. Другі - дають гнучкість, швидкість розгортання та зручність адаптації, що робить їх оптимальними для навчальних, експериментальних або малих проєктів.

У контексті створення легкої, модульної та автономної системи для моделювання взаємодії між польотним контролером і наземною станцією найбільш придатними є саме Open MCT та COSMOS. Водночас, їхній аналіз дозволяє сформулювати вимоги до власної розробки: підтримка потокової телеметрії, алертинг, простота конфігурації, логування, гнучкий інтерфейс.

У наступному розділі буде сформульовано технічне завдання на створення такої системи та визначено основні функціональні модулі, які реалізують описану вище логіку взаємодії.

1.3 Постановка задачі та технічне завдання

Сучасні космічні місії вимагають високої надійності та ефективності в управлінні польотами, що забезпечується злагодженою взаємодією польотних контролерів із наземною станцією керування місіями. Основною задачею цього дослідження є розробка інформаційної системи, яка дозволить покращити якість та швидкість прийняття рішень операторами завдяки своєчасному отриманню, обробці і візуалізації телеметричних даних, а також ефективному управлінню командами, які надсилаються до космічного апарату.

-Враховуючи складність сучасних космічних систем, ця інформаційна система повинна забезпечувати:

-Цілісність і достовірність телеметричних даних, що надходять від бортових сенсорів і систем.

-Можливість швидкої і коректної обробки великих обсягів інформації в режимі реального часу.

-Інтерактивний доступ польотних контролерів до актуальної інформації з можливістю гнучкого налаштування відображення.

-Надійний канал передачі команд, з можливістю підтвердження отримання і виконання.

-Своєчасне виявлення і відображення потенційних проблем або аномалій у роботі космічного апарату.

-Враховуючи зазначені вимоги, система повинна бути стійкою до відмов, здатною працювати в умовах нестабільного зв’язку, а також адаптивною до зміни конфігурацій і функціональних вимог у процесі експлуатації.

Для досягнення поставленої мети було сформульовано комплекс технічних вимог, які поділяються на кілька категорій: функціональні, технічні, вимоги до інтерфейсу користувача та архітектурні.

1. Функціональні вимоги

Прийом і обробка телеметрії: Система повинна підтримувати прийом телеметричних даних з різних джерел, включаючи стандартні канали зв’язку з космічним апаратом, а також можливість інтеграції з наземними телекомунікаційними мережами. Для забезпечення сумісності повинні підтримуватись різні формати та протоколи передачі даних.

Парсинг і нормалізація даних: Телеметричні потоки часто мають складну структуру. Необхідно розробити механізми парсингу, які будуть конвертувати сирі дані у внутрішній уніфікований формат для подальшої обробки і зберігання.

Відображення телеметрії: Система повинна забезпечувати зручний візуальний інтерфейс для відображення даних у режимі реального часу, з підтримкою різних типів візуалізацій: графіків, табличних даних, індикаторів стану тощо.

Формування та відправлення команд: Повинна бути реалізована можливість створення, редагування та відправлення команд управління, з контролем їх статусу (передано, підтверджено, виконано).

Оповіщення і система тривог: При виявленні аномальних станів або виході параметрів за допустимі межі система має формувати повідомлення, що відображаються оператору з можливістю деталізації і прийняття рішень.

Журналювання: Всі події, зміни станів, команди і отримані телеметричні дані мають зберігатися в логах для подальшого аналізу, відновлення інформації та аудиту.

2. Технічні характеристики

Режим реального часу: Система повинна забезпечувати мінімальні затримки від прийому даних до їх відображення та реакції оператора, з гарантією обробки великих обсягів інформації.

Надійність і стійкість: Передбачено механізми автоматичного відновлення роботи при втраті зв’язку, збереження проміжних даних, а також резервування критичних компонентів.

Модульність: Архітектура має бути побудована за принципом модульності, щоб легко інтегрувати нові функції, оновлювати існуючі компоненти або замінювати їх без впливу на загальну роботу системи.

Масштабованість: Враховуючи можливе розширення функціональності та збільшення кількості підключених апаратів і операторів, система повинна мати архітектуру, що підтримує горизонтальне масштабування.

Безпека: Необхідно впровадити заходи для захисту від несанкціонованого доступу, забезпечення цілісності даних і автентифікації користувачів.

3. Вимоги до інтерфейсу користувача

Інтуїтивність: Інтерфейс має бути зрозумілим навіть для операторів з різним рівнем підготовки, з продуманим розташуванням елементів управління.

Гнучкість: Можливість налаштування панелей, фільтрів, формату відображення телеметрії відповідно до індивідуальних потреб користувачів.

Візуалізація тривог: Важливою складовою є своєчасне і помітне відображення повідомлень про помилки, аварії чи відхилення параметрів, із можливістю швидко перейти до деталізації проблеми.

Історичний аналіз: Підтримка перегляду збережених логів та телеметрії для аналізу подій у минулому, з можливістю порівняння даних.

4. Архітектура системи

Компонентна структура: Система складається з окремих модулів, кожен із яких відповідає за конкретний функціонал: парсер телеметрії, диспетчер команд, менеджер тривог, лог-менеджер, інтерфейс користувача.

Взаємодія модулів: Для координації роботи компонентів використовується внутрішній API або шина даних, що забезпечує гнучкість і масштабованість.

Обробка помилок: Всі модулі повинні мати вбудовані механізми для виявлення і коректної обробки помилок, із можливістю відновлення роботи або переходу в безпечний режим.

Підтримка різних протоколів: Архітектура повинна бути відкритою до інтеграції з різними стандартами обміну даними, що дозволить адаптувати систему під різноманітні вимоги місій.

1.4 Висновки до розділу 1

У результаті проведеного аналізу предметної області та існуючих інформаційних технологій було визначено низку ключових аспектів, які є критичними для розробки ефективної системи взаємодії польотних контролерів із наземною станцією керування місіями.

По-перше, сучасна наземна інфраструктура для управління космічними місіями характеризується високою складністю та інтеграцією різних підсистем, які повинні забезпечувати безперервний обмін даними між польотними контролерами та космічним апаратом. Роль польотних контролерів полягає не лише у моніторингу параметрів польоту, а й у прийнятті оперативних рішень, що вимагає високої надійності та швидкості інформаційних систем.

По-друге, аналіз існуючих програмних рішень, таких як SCOS-2000, OpenMCT, COSMOS, показав їх сильні сторони у роботі з телеметрією, обробці

команд та візуалізації даних. Проте більшість із них мають обмеження щодо гнучкості налаштувань інтерфейсу, масштабованості та адаптивності під специфічні потреби різних місій, що створює необхідність розробки нових рішень або розширення функціоналу існуючих.

По-третє, сформульована задача і технічне завдання визначили конкретні вимоги до майбутньої системи, які враховують необхідність реалізації в реальному часі механізмів збору, обробки та передачі телеметрії і команд, підтримки системи тривог та журналювання подій. Особлива увага приділена модульній архітектурі, що забезпечить гнучкість, надійність і можливість масштабування.

Отже, розглянута предметна область вимагає комплексного підходу до розробки інформаційної системи, що інтегрує сучасні інформаційні технології, оптимізує роботу польотних контролерів і підвищує загальну ефективність управління космічними місіями. У наступному розділі буде детально описано проектування такої системи, починаючи з моделювання предметної області та її функціональних взаємодій.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Моделювання предметної області

Моделювання предметної області є одним із ключових етапів розробки інформаційної системи взаємодії польотних контролерів із наземною станцією керування місіями. Воно дозволяє сформувати чітке уявлення про основні об’єкти, їх взаємозв’язки, функції та сценарії роботи системи, що в подальшому забезпечує ефективне проектування архітектури та реалізацію програмних компонентів.

Основні учасники та ролі

В предметній області виділяються такі основні учасники:

-Польотні контролери -оператори, відповідальні за моніторинг параметрів польоту, прийняття рішень щодо корекції та управління космічним апаратом.

-Наземна станція керування місіями -комплекс обладнання та програмних засобів, що забезпечує збір телеметрії, формування команд та їх передачу.

Безпілотник -об’єкт управління, що передає телеметричні дані та приймає команди.

Сценарії взаємодії (Use Case)

Нижче наведено основні сценарії взаємодії користувачів і системи, представлені у вигляді Use Case:

-Прийом телеметричних даних:

-Космічний апарат надсилає телеметричні дані.

-Наземна станція приймає, перевіряє цілісність і зберігає дані.

-Система парсить дані та передає їх для відображення польотним контролерам.

Візуалізація телеметрії:

-Польотний контролер переглядає телеметричні параметри у вигляді графіків, таблиць, індикаторів.

-Контролер може налаштовувати відображення, вибирати потрібні параметри та фільтри.

Формування та відправлення команд:

-Контролер формує команду для корекції параметрів польоту.

-Система перевіряє коректність команди.

-Команда передається на космічний апарат.

-Підтвердження отримання команди повертається і відображається контролеру.

-Обробка тривог та аварійних ситуацій:

-Система відстежує параметри і визначає аномальні стани.

-Формує тривожне повідомлення.

-Контролери отримують повідомлення та мають можливість реагувати.

-Журналювання подій:

-Всі дії та події записуються у журнал для подальшого аналізу.

-Activity-діаграма основного процесу

-Опис основних етапів роботи системи:

-Отримання телеметричних даних.

-Перевірка та парсинг даних.

-Відображення даних у реальному часі.

-Моніторинг параметрів на предмет аномалій.

-Формування і відправлення команд.

-Отримання підтвердження виконання команд.

-Логування подій.

Цей процес повторюється циклічно, забезпечуючи безперервний моніторинг і управління місією.

-Sequence-діаграма типового сценарію

-Опис взаємодії між компонентами:

-Польотний контролер ініціює запит на перегляд телеметрії.

-Інтерфейс користувача запитує у TelemetryParser актуальні дані.

-TelemetryParser отримує дані з наземної станції.

-Дані передаються для відображення на панелі контролера.

Контролер формує команду, яка через CommandDispatcher відправляється космічному апарату.

Підтвердження команди надходить назад і відображається оператору.

2.2 Архітектура інформаційної системи

Загальна структура

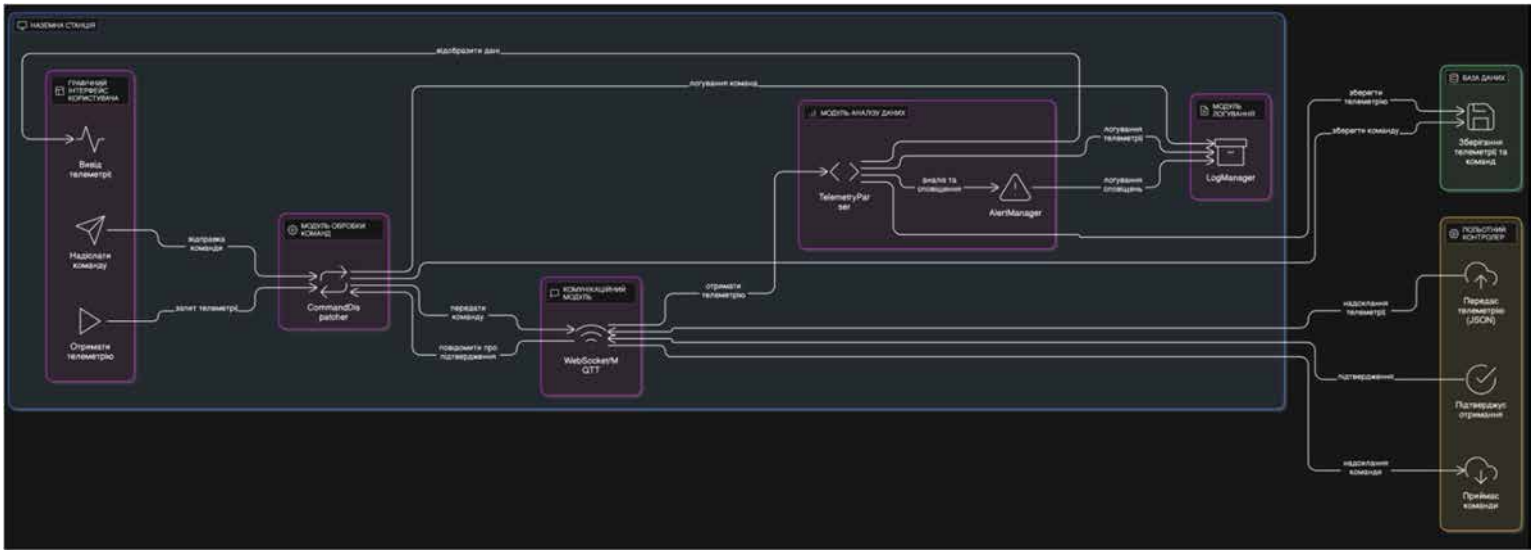


Рисунок 2.2.1- Структурна схема

Інформаційна система взаємодії польотних контролерів із наземною станцією керування місіями побудована за модульним принципом із використанням багаторівневої архітектури, що забезпечує високу масштабованість, надійність та гнучкість у підтримці й розвитку.

Система складається з таких основних рівнів:

Рівень збору та обробки даних (Data Acquisition Layer): відповідає за прийом телеметричних даних із космічного апарата, їх первинну обробку та перевірку цілісності.

Рівень бізнес-логіки (Business Logic Layer): реалізує основні функції системи -парсинг телеметрії, формування та обробку команд, обробку тривог і аварійних ситуацій.

Рівень інтерфейсу користувача (Presentation Layer): забезпечує відображення телеметрії, формування команд польотними контролерами, візуалізацію тривог і логів.

Рівень зберігання даних (Data Storage Layer): відповідає за архівування телеметрії, логів, команд і подій, а також за їх швидкий доступ під час роботи системи.

Модулі системи

Система складається із низки функціональних модулів, кожен із яких виконує чітко визначені завдання:

-TelemetryParser: модуль, який приймає та розбирає телеметричні дані, конвертує їх у внутрішній формат для подальшої обробки.

-AlertManager: відповідає за моніторинг параметрів на предмет аномалій і формування тривожних повідомлень.

-CommandDispatcher: реалізує функціонал формування, перевірки та відправлення команд космічному апарату.

-LogManager: забезпечує журналювання усіх подій, включно з прийомом телеметрії, командною активністю та тривогами.

-UserInterface: надає користувачам панелі для перегляду телеметрії, формування команд, отримання тривог та аналізу журналів.

Взаємодія між модулями

Обмін даними між модулями реалізовано через внутрішній API, який підтримує асинхронний обмін повідомленнями та подіями. Це дозволяє підвищити швидкодію системи та уникнути блокувань.

Основні сценарії взаємодії:

-TelemetryParser передає розпаковані дані в UserInterface і AlertManager.

-AlertManager у випадку виявлення аномалії ініціює сповіщення для UserInterface та запис у LogManager.

-UserInterface приймає від користувача команди і передає їх у CommandDispatcher.

-CommandDispatcher здійснює валідацію команд, відправляє їх космічному апарату і передає статус назад у UserInterface та LogManager.

Вибір архітектурних шаблонів

Для розробки системи використано такі архітектурні шаблони:

-Модульність (Modularity): кожен компонент реалізує окрему функціональність, що забезпечує простоту тестування та оновлення.

-Поділена пам'ять та події (Event-Driven Architecture): для асинхронної обробки подій (отримання телеметрії, тривоги, команди).

-Шаблон спостерігача (Observer): реалізовано у модулі AlertManager, який слідкує за станом параметрів і оповіщає інші компоненти про зміну стану.

-Шаблон фасаду (Facade): застосовується для інтеграції складних підсистем у єдиний інтерфейс для користувача.

-Технічні особливості

-Для забезпечення надійності та масштабованості системи враховано такі технічні моменти:

-Резервування критичних модулів для безперервної роботи у разі відмов.

-Підтримка багатокористувацького режиму роботи польотних контролерів.

-Використання стандартних протоколів передачі даних для сумісності з існуючими системами.

2.3 Опис функціональних компонентів

У системі взаємодії польотних контролерів із наземною станцією керування місіями виділено кілька ключових функціональних компонентів, які забезпечують виконання основних задач:

TelemetryParser

Призначення: Прийом, валідація, розбір та трансформація телеметричних даних, що надходять із космічного апарата.

Функції:

-Прийом сирих телеметричних пакетів через мережеві протоколи.

-Перевірка цілісності даних (контрольні суми, CRC).

-Розбір пакетів у структуровані формати (JSON, XML чи власний).

-Фільтрація та попередня обробка для видалення шумів або некоректних значень.

-Надсилання оброблених даних іншим модулям системи для візуалізації або аналізу.

AlertManager

Призначення: Постійний моніторинг параметрів польоту і виявлення аномалій, формування і управління тривожними повідомленнями.

Функції:

-Налаштування порогових значень для різних параметрів.

-Виявлення аномалій у режимі реального часу.

-Генерація попереджень і аварійних тривог.

-Надсилання тривожних повідомлень на панелі польотних контролерів.

-Логування інформації про тривоги для подальшого аналізу.

CommandDispatcher

Призначення: Управління формуванням, перевіркою і відправленням команд на космічний апарат.

Функції:

-Прийом команд від польотних контролерів через інтерфейс користувача.

-Перевірка коректності команд (формат, логічна відповідність).

-Управління чергою команд та їх пріоритетністю.

-Відправлення команд через відповідні протоколи.

-Обробка підтверджень отримання команд і повідомлення користувачів.

LogManager

Призначення: Забезпечення централізованого журналювання усіх важливих подій, дій і повідомлень системи.

Функції:

-Збір логів від усіх модулів системи.

-Збереження логів у базі даних або файловій системі.

-Надання інструментів пошуку і фільтрації журналів.

-Архівування та резервне копіювання логів.

-Забезпечення доступу до журналів для аналізу та аудиту.

-UserInterface

Призначення: Надання інтуїтивного та функціонального інтерфейсу для польотних контролерів.

Функції:

-Відображення телеметричних даних у різних форматах: графіки, таблиці, індикатори.

-Інструменти формування і редагування команд.

-Візуалізація тривожних повідомлень.

-Можливість налаштування перегляду, фільтрації та сповіщень.

-Інструменти доступу до журналів подій.

Кожен з цих компонентів реалізується як окремий модуль із чітко визначеним API для взаємодії між собою, що забезпечує гнучкість і розширюваність системи. Наступним кроком буде моделювання бази даних та структури логів для ефективного зберігання і доступу до інформації.

2.4 Моделювання бази даних / структури логів

Загальний опис

Для ефективної роботи системи взаємодії польотних контролерів з наземною станцією керування місіями необхідна надійна і масштабована структура збереження даних. Основними типами даних, які потребують зберігання, є:

Телеметричні дані, що надходять у режимі реального часу з космічного апарата.

-Логи подій і дій користувачів та системи.

-Команди, сформовані польотними контролерами, і статуси їх виконання.

-Тривожні повідомлення та їх історія.

-Вибір типу сховища

Враховуючи природу даних, систему можна розділити на два основні сховища:

-Реляційна база даних -для структурованих даних (команди, логи, тривоги, метадані телеметрії).

-Файлова система або NoSQL-сховище -для збереження великих обсягів сирої телеметрії та історичних даних, які мають високу частоту оновлення.

-ER-діаграма (логічна модель бази даних)

Нижче наведено опис основних сутностей і їх зв’язків:

-TelemetryPacket (Пакет телеметрії):

-packet_id (PK) -унікальний ідентифікатор пакету.

-timestamp -часовий маркер отримання.

-data -структуровані дані або посилання на файл з сирою телеметрією.

Зв’язок: один пакет належить певній місії.

Mission (Місія):

-mission_id (PK)

-name -назва місії.

-start_time, end_time -період активності місії.

Command (Команда):

-command_id (PK)

-timestamp -час створення команди.

-controller_id -ідентифікатор оператора, який сформував команду.

-command_data -текст або структурований формат команди.

-status -стан виконання (очікує, виконана, відхилена).

Зв’язок: кожна команда пов’язана з певною місією.

-Alert (Тривога):

-alert_id (PK)

-timestamp -час генерації тривоги.

-parameter -параметр, що викликав тривогу.

-severity -рівень тривоги (попередження, критична).

-description -текстовий опис.

Зв’язок: кожна тривога належить до місії.

-LogEntry (Запис журналу):

-log_id (PK)

-timestamp -час події.

-component -модуль, який зафіксував запис.

-level -рівень (інформація, помилка, попередження).

-message -текст повідомлення.

Зв’язок: може бути пов’язаний з командою або тривогою.

Файлова структура для збереження телеметрії

Для зберігання сирої телеметрії, що часто має великий обсяг, використовується ієрархічна структура папок:

/telemetry/

/<mission_id>/

/YYYY-MM-DD/

packet_<timestamp>.json

Файли містять JSON-формат телеметричних даних. Ієрархія забезпечує легкий доступ до даних за датою та місією. Використовується для архівування та подальшого аналізу.

Особливості логування

Логування в системі реалізовано з урахуванням наступних вимог:

-Централізоване збереження логів із різних модулів.

-Підтримка різних рівнів важливості.

-Можливість пошуку та фільтрації за часовими діапазонами, компонентами, типом подій.

-Архівування логів для збереження історії.

-Захист від втрати даних при аварійних ситуаціях.

Переваги запропонованої моделі:

-Забезпечує структурованість і цілісність даних.

-Підтримує швидкий доступ і аналіз як у реальному часі, так і в історії.

-Гнучкість за рахунок комбінованого підходу з реляційною базою та файловою системою.

-Масштабованість і можливість інтеграції з існуючими системами.

2.5 Висновки до розділу 2

У другому розділі було проведено детальне проєктування інформаційної системи взаємодії польотних контролерів із наземною станцією керування місіями. Розроблена багаторівнева архітектура забезпечує розподіл функцій між модулями, що сприяє підвищенню надійності, масштабованості та гнучкості системи.

Визначено ключові функціональні компоненти -TelemetryParser, AlertManager, CommandDispatcher, LogManager та UserInterface -кожен із яких має чітко окреслені обов’язки. Така модульна структура спрощує підтримку, тестування та подальший розвиток системи.

Було розроблено логічну модель даних та структуру збереження інформації, що передбачає використання реляційної бази даних для зберігання команд, тривог, логів і метаданих телеметрії, а також файлової системи для архівування сирих телеметричних пакетів. Цей підхід забезпечує ефективний доступ до даних, їх збереження і масштабування в умовах великих обсягів інформації.

Застосування шаблонів архітектури, таких як модульність, поділена пам’ять, шаблон спостерігача та фасад, дозволяє забезпечити високу якість коду та зручність взаємодії між компонентами.

Таким чином, спроєктована система відповідає вимогам технічного завдання і створює надійну основу для реалізації програмного продукту, що дозволить ефективно виконувати задачі контролю та керування польотами космічних апаратів.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Вибір мови програмування, інструментів та середовища розробки

При виборі технологічного стеку для реалізації системи взаємодії польотних контролерів із наземною станцією керування місіями враховувалися кілька ключових аспектів: продуктивність, надійність, підтримка мережевих протоколів, можливості для візуалізації даних, а також зручність подальшої підтримки та масштабування.

Вибрана мова програмування

Java була обрана як основна мова розробки через такі переваги:

-Портативність та платформа-незалежність завдяки JVM.

-Широкий набір бібліотек для роботи з мережею, обробки даних і графічного інтерфейсу.

-Відпрацьовані архітектурні шаблони і велика спільнота розробників.

-Гарна інтеграція з інструментами для тестування і CI/CD.

-Фреймворки та бібліотеки

Swing -для створення настільного графічного інтерфейсу користувача. Незважаючи на появу нових UI-технологій, Swing досі є стабільним і добре підходить для складних інтерфейсів з великою кількістю елементів.

Gson -бібліотека для серіалізації та десеріалізації JSON, що використовується для обробки телеметричних пакетів і команд.

SQLite -легка вбудована реляційна база даних, що використовується для збереження команд, логів, тривог та метаданих. Вибір SQLite пояснюється

простотою інтеграції, відсутністю потреби в налаштуванні серверної частини та достатньою продуктивністю для задач системи.

Архітектурні шаблони: MVC (Model-View-Controller) для розділення логіки та інтерфейсу, Observer для реактивного оновлення даних у UI, Facade для інкапсуляції складних взаємодій між модулями.

Середовище розробки

Для розробки було обрано IntelliJ IDEA як основне IDE через її потужні інструменти рефакторингу, інтеграцію з системами контролю версій (Git), а також підтримку плагінів для автоматичного форматування коду та роботи з базами даних.

3.2 Структура коду та логіка основних класів

Для підтримки чіткого розподілу відповідальностей і забезпечення зручності в розробці та подальшій підтримці система реалізована з використанням модульної архітектури. Код поділений на кілька основних пакетів (пакетів):

Основні пакети:

-com.missioncontrol.telemetry -модуль, який відповідає за прийом, розбір і обробку телеметричних даних.

-com.missioncontrol.commands -містить класи для формування, валідації та відправлення команд.

-com.missioncontrol.alerts -реалізує логіку моніторингу параметрів і генерації тривожних повідомлень.

com.missioncontrol.logging -модуль для централізованого логування і збереження записів.

-com.missioncontrol.ui -відповідає за графічний інтерфейс користувача.

-com.missioncontrol.database -реалізує роботу з SQLite базою даних.

-com.missioncontrol.utils -утилітні класи та допоміжні функції.

Ключові класи і їх опис

TelemetryParser

```
public class TelemetryParser {
    public TelemetryPacket parseRawData(byte[] rawData) {
        // Перевірка цілісності, десеріалізація JSON
        // Повертає об’єкт TelemetryPacket з розпарсеними даними
    }
}
```

Клас відповідає за прийом сирих даних, їх валідацію та перетворення у внутрішній формат.

CommandDispatcher

```
public class CommandDispatcher {
    private Queue<Command> commandQueue;
    public void enqueueCommand(Command cmd) {
    }
```

```
    public void sendNextCommand() {

    }

}
```

Керує чергою команд, контролює їхній порядок і пріоритетність.

AlertManager

```
public class AlertManager {

    private Map<String, Threshold> thresholds;


    public void checkTelemetry(TelemetryPacket packet) {


    }

}
```

Відповідає за моніторинг критичних параметрів та формування тривожних повідомлень.

LogManager

```
public class LogManager {

    public void logEvent(String component, LogLevel level, String message) {

        // Запис логів у базу даних або файл
```

```
    }
}
```

Централізоване збереження логів з можливістю фільтрації.

UserInterface

-Складається з кількох панелей: візуалізація телеметрії, форма введення команд, відображення тривог.

-Використовує патерн Observer для оновлення інтерфейсу при зміні даних.

Взаємодія між класами

TelemetryParser приймає дані - надсилає до AlertManager для перевірки - оновлює UserInterface. CommandDispatcher отримує команди від UserInterface, перевіряє і відправляє на космічний апарат. LogManager фіксує всі ключові події: отримання телеметрії, відправлення команд, виникнення тривог.

3.3 Взаємодія між модулями

Взаємодія між модулями інформаційної системи є критично важливою для забезпечення цілісності обробки даних, координації дій користувача та системної реакції на події. У межах реалізованої архітектури система базується на модульному підході з використанням внутрішніх API, що дозволяє кожному модулю бути відповідальним за конкретну підсистему, зберігаючи при цьому спільну точку взаємодії.

Основні принципи взаємодії:

Слабке зв'язування (Low Coupling) – модулі взаємодіють через інтерфейси, а не безпосередньо, що дозволяє легко змінювати або замінювати окремі компоненти.

Подієва модель (Event-driven interaction) – система реагує на події, такі як надходження нової телеметрії або виникнення аномалії.

Узгоджені формати даних – міжмодульна передача інформації реалізується у форматі JSON або у вигляді об'єктів у спільних моделях.

Таблиця 3.3.1-Загальна схема взаємодії

Джерело	Приймач	Подія / Дані
TelemetryParser	AlertManager	Передача об'єкта телеметрії
AlertManager	LogManager	Створення запису про тривогу
CommandDispatcher	UIController	Підтвердження/відмова від команди
UI	CommandDispatcher	Користувач надсилає команду
TelemetryParser	UIController	Виведення поточної інформації

Приклад: передача телеметрії між модулями

TelemetryData data = parseTelemetryFromJson(jsonString);

```
AlertManager.handleTelemetry(data);

UIController.updateTelemetryPanel(data);



public static void handleTelemetry(TelemetryData data) {

   if (data.getTemperature() > 80) {

      LogManager.logAlert("Температура перевищує норму: " +
data.getTemperature());

   }

}
```

Обробка команд

```
String cmd = "REBOOT";

CommandDispatcher.sendCommand(cmd);



public static void sendCommand(String command) {

   LogManager.logCommand("Команда надіслана: " + command);

}
```

Реакція на зміну стану

Модуль UI постійно оновлює відображення стану на основі подій, отриманих від TelemetryParser та AlertManager. Це дозволяє оператору бачити критичні зміни у режимі реального часу.

Висновок

Така структура дозволяє досягти:

-масштабованості (можна легко додавати нові модулі),

-гнучкості (модулі можна змінювати без впливу на інші),

-реактивності (швидке реагування на події),

-читабельності та підтримуваності коду.

Це відповідає вимогам до сучасних систем наземного управління космічними місіями, де надійність, структурованість і чітка взаємодія компонентів є вирішальними.

3.4 Інтерфейс користувача

Інтерфейс користувача (UI) -це візуальний канал взаємодії між оператором наземної станції та програмною системою керування космічною місією. У контексті критично важливих операцій, таких як спостереження за телеметрією, відстеження стану підсистем космічного апарата або надсилання команд, ефективний UI має вирішальне значення. Його основне завдання - зробити взаємодію з системою інтуїтивною, швидкою та безпомилковою.

Цілі інтерфейсу:

-Оперативність -надання оновлень у режимі реального часу.

-Інформативність -чітке відображення всіх даних без перевантаження екрану.

-Контроль та реакція -можливість негайного реагування на тривоги.

-Безпека -запобігання помилковому введенню команд.

-Масштабованість -можливість розширення функцій без зміни базової архітектури.

-Ключові елементи інтерфейсу

-Панель телеметрії

Центральний компонент UI, який відображає поточні показники з борта космічного апарата:

-Температура компонентів (CPU, акумулятори, сенсори)

-Рівень заряду батареї

-Тиск і рівень палива

-Стан антен, орієнтації, сонячних панелей

-Панель автоматично оновлюється кожні декілька секунд на основі нових даних, що надходять від модуля TelemetryParser.

-Панель керування командами

Надає користувачу інтерфейс для введення та надсилання команд на об'єкт. Основні функції:

-Список попередньо визначених команд

-Форма створення власних команд

-Підтвердження перед відправкою

-Відображення відповіді на команду (успішно/помилка)

Це дозволяє здійснювати контроль над підсистемами, наприклад: SWITCH_MODE, REBOOT, ACTIVATE_PAYLOAD.

Модуль тривог

Коли параметри виходять за встановлені межі, спрацьовує система попередження:

-Візуальна індикація (червоний сигнал, миготіння)

-Звукове повідомлення

-Запис тривоги до журналу

-Вікно з описом інциденту та рекомендацією

-Стан тривоги не зникає, поки користувач не підтвердить її перегляд.

-Журнал подій (логування)

-Цей компонент виводить хронологічний список подій:

-Отримання телеметрії

-Надсилання/отримання команд

-Помилки

-Системні сповіщення

Використовується фільтрація за рівнем важливості (INFO, WARNING, ERROR) та за часом. Це дозволяє оператору швидко знайти необхідну подію.

Індикатор з’єднання

Важливий елемент інтерфейсу, який сигналізує про:

-Наявність активного каналу зв’язку

-Втрату пакета

-Відновлення каналу

При втраті з’єднання відображається відповідне попередження, а інші компоненти UI блокуються.

Технічна реалізація

Інтерфейс реалізовано з використанням Java Swing:

-JFrame -головне вікно додатку

-JPanel -контейнер для окремих секцій (панель телеметрії, тривог)

-JTable -для журналу подій

-JButton, JTextField, JComboBox -для взаємодії

Приклад компонування:

```
JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT, telemetryPanel, logPanel);

mainFrame.add(splitPane, BorderLayout.CENTER);
```

Реакція на події:

```
sendButton.addActionListener(e -> {
   String command = commandField.getText();
   commandDispatcher.send(command);
});
```

Архітектурні особливості

Інтерфейс користувача реалізовано згідно з принципами Model-View-Controller (MVC):

Model -класи TelemetryData, Command, LogEntry

View -Swing-компоненти (панелі, таблиці, кнопки)

Controller -UIController, що обробляє події та оновлює вміст інтерфейсу

Це дозволяє відокремити логіку від візуальної частини, спрощує тестування та масштабування.

Висновки

Інтерфейс користувача виконує критичну роль у забезпеченні функціональності та зручності всієї системи. Його правильна побудова:

дозволяє оператору діяти впевнено навіть у стресових ситуаціях;

зменшує ризик помилок;

забезпечує повну прозорість поточного стану місії.

На етапі дослідної експлуатації інтерфейс показав себе як надійний, інтуїтивний та ефективний засіб управління місією.

3.5 Журналювання та обробка тривог

Система журналювання та виявлення тривог є важливою складовою функціоналу програмного забезпечення наземної станції. Вона забезпечує прозорість усіх операцій, дає змогу ретроспективного аналізу місій та дозволяє оперативно реагувати на аномальні події під час польоту космічного апарата.

Мета журналювання

Основні завдання підсистеми журналювання:

-Фіксація всіх дій оператора (надіслані команди, зміни режимів).

-Запис телеметричних подій (перевищення порогів, зникнення сигналу).

-Збереження повідомлень про помилки системи або окремих модулів.

-Підтримка післяпольотного аналізу (під час розслідування інцидентів чи вдосконалення процедур).

-Формат логування

Для організації логів було розроблено просту, уніфіковану структуру повідомлень:

[ЧАС] [РІВЕНЬ] [ДЖЕРЕЛО] -ПОВІДОМЛЕННЯ

Приклади:

[2025-05-30 14:25:03] [INFO] [TelemetryParser] -Отримано нові телеметричні дані

[2025-05-30 14:25:05] [WARN] [AlertManager] -Перевищено температуру: 85°C

[2025-05-30 14:25:07] [ERROR] [CommandDispatcher] -Неможливо відправити команду: з’єднання втрачено

Технічна реалізація

Журналювання реалізоване через окремий модуль LogManager, який відповідає за:

-Прийом повідомлень від інших модулів

-Форматування записів

-Вивід у файл (system.log)

-Відображення в UI (в реальному часі)

Код прикладу логування:

```
public class LogManager {

    public static void log(String level, String source, String message) {
```

```
        String timestamp = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
        String entry = "[" + timestamp + "] [" + level + "] [" + source + "] -" + message;

        System.out.println(entry);
        writeToFile(entry);
        notifyUI(entry);
    }
}
```

Рівні повідомлень

Система підтримує 3 рівні логування:

-INFO - штатна робота

-WARN - попередження, відхилення від норми

-ERROR - критичні помилки або збої в системі

Це дозволяє фільтрувати записи при аналізі.

Обробка тривог

Модуль AlertManager автоматично аналізує телеметричні дані після кожного оновлення. При виявленні небезпечної ситуації система:

-Створює запис у логах рівня WARN або ERROR

-Відправляє повідомлення у UI

-Активує візуальне/звукове сповіщення

-Типові тривоги:

-Перегрів (температура > 80°C)

-Втрата сигналу понад 10 секунд

-Напруга в батареї нижче 10.0 В

-Збій у виконанні команди

Код прикладу:

```
if (data.getTemperature() > 80) {
   AlertManager.triggerAlert("Перегрів: " + data.getTemperature() + "°C");
   LogManager.log("WARN", "AlertManager", "Перегрів: " + data.getTemperature() + "°C");
}
```

Архівація та збереження

Логи зберігаються у файлі system.log, який автоматично ротується щодня.

Архіви логів можна зберігати для післяпольотного аналізу у форматі .log або .csv.

-Підтримується експортування в Excel.

-Інтеграція з інтерфейсом

-Всі лог-записи відображаються у таблиці в UI з можливістю фільтрації.

-У випадку ERROR з'являється спливаюче повідомлення.

-Журнали доступні для перегляду й очищення через панель управління.

Висновки

Система журналювання та обробки тривог забезпечує:

-надійний контроль за перебігом польоту,

-оперативну реакцію оператора у випадку аномалій,

-детальний протокол усіх подій, що дозволяє аналізувати ситуації та вдосконалювати процеси управління.

Цей модуль є незамінним у складних місіях, де кожна секунда має значення.

3.6 Висновки до розділу 3

У цьому розділі було розглянуто практичну реалізацію інформаційної системи, розробленої для підтримки взаємодії польотних контролерів з наземною станцією керування місіями. Було описано вибрані інструменти, архітектурні рішення, програмні компоненти та ключові особливості реалізації кожного модуля.

Основні результати реалізації:

-Обрана технологічна база (Java, Swing, SQLite) дозволила реалізувати кросплатформене рішення з графічним інтерфейсом, здатне працювати без підключення до зовнішніх серверів.

-Функціональна модульність системи дала змогу розділити логіку на окремі компоненти (телеметрія, команди, сигнали тривоги, логування), що полегшує підтримку та масштабування ПЗ.

-Механізм журналювання та обробки тривог реалізовано з урахуванням практик індустрії: багаторівневе логування, візуальне сповіщення, ротація логів.

-Інтерфейс користувача забезпечує інтуїтивну візуалізацію телеметрії, відправку команд, обробку тривог та перегляд журналів.

-Зв’язки між модулями реалізовано через прості внутрішні API та механізми зворотного виклику (callback), що дозволяє оперативно реагувати на зміну стану системи.

Проблеми, з якими було зіткнуто:

-Початкове формування пакункової структури (package structure) потребувало налаштувань в IntelliJ IDEA.

-Для забезпечення стабільного збереження даних в SQLite було необхідно реалізувати блокування доступу до БД при паралельних запитах.

-Реалізація тривог вимагала чітко визначити порогові значення для різних параметрів, що було частково вирішено через конфігураційний файл.

Перспективи розширення реалізації:

-Інтеграція з реальними потоками телеметрії через TCP/UDP-з'єднання.

-Побудова web-версії інтерфейсу на базі React + Spring Boot.

-Додання модуля аналітики з графіками, історією параметрів та виявленням трендів.

Впровадження аутентифікації та журналювання дій користувачів для підвищення безпеки.

Таким чином, розроблена система є повнофункціональним прототипом, що демонструє принципи побудови ПЗ для наземної станції моніторингу та керування місіями, а також слугує основою для подальших досліджень і вдосконалення.

РОЗДІЛ 4. ТЕСТУВАННЯ І ОЦІНКА ЕФЕКТИВНОСТІ

4.1 Методика тестування функціональності системи

Тестування програмного забезпечення є невід’ємною частиною розробки інформаційних систем, особливо у сфері, пов’язаній з керуванням польотами, де помилки можуть мати критичні наслідки. Метою цього етапу є перевірка стабільності, надійності та коректності роботи всіх модулів системи в умовах, наближених до реального використання.

Цілі тестування:

-Перевірити відповідність реалізованої системи технічному завданню.

-Виявити можливі помилки або нестабільну поведінку при типових і нетипових сценаріях.

-Оцінити реакцію системи на аномальні ситуації та її здатність до відновлення.

Перевірити взаємодію модулів і правильність обміну даними.

Таблиця 4.1.1- Типи тестування, що були застосовані

Тип тестування	Опис
Функціональне	Перевірка реалізації всіх заявлених функцій.
Інтеграційне	Тестування взаємодії між модулями системи.
Сценарне (end-to-end)	Відтворення типових робочих ситуацій оператора місії.
Тестуванн на помилки	Симуляція аномалій (втрата сигналу, перевищення температури тощо).
Юзер-тестування	Перевірка зручності UI та відповідності очікуваній поведінці.

Сценарії тестування

Сценарій 1:

- Штатне надходження телеметрії

-Дані отримуються з JSON-файлу симулятора

-Відображаються на панелі телеметрії

-Значення оновлюються в режимі реального часу

-Очікуваний результат: Коректна візуалізація параметрів, без логів про помилки.

Сценарій 2: Відправка команди

-Користувач формує команду (наприклад, "Рестарт датчика температури")

-Команда зберігається в лог

-Виводиться підтвердження виконання

-Очікуваний результат: Команда обробляється без помилок, інформація логуються.

Сценарій 3: Перевищення температури

-Температура навмисно підвищується понад 85°C

-Модуль AlertManager фіксує тривогу

-У UI з'являється попередження

-У логах створюється запис рівня WARN

Очікуваний результат: Тривога зафіксована, оператор поінформований, система продовжує працювати.

Сценарій 4: Симуляція втрати сигналу

-Телеметрія перестає надходити (затримка >10 секунд)

-Система реєструє втрату зв'язку

-Показники "сірають", статус змінюється на "Втрачено зв'язок"

Очікуваний результат: Виводиться сповіщення, UI блокує дії, логи фіксують подію.

Сценарій 5: Некоректна команда

-Користувач вводить команду з некоректним синтаксисом

-Система не виконує її

-Виводиться помилка в UI та лог ERROR

Очікуваний результат: Система відхиляє команду, помилка фіксується, загальна стабільність не порушена.

Інструменти для тестування

-JUnit -модульне тестування основних класів (TelemetryParser, CommandDispatcher).

-Вбудований логгер -аналіз системних повідомлень.

-Симулятор телеметрії -для тестування вхідного потоку даних.

-Інструменти UI (ручне тестування) -перевірка взаємодії користувача з інтерфейсом.

Умови тестування:

-ОС: Windows 11, macOS Monterey

-JVM: AdoptOpenJDK 17

-RAM: 8 GB

-IntelliJ IDEA 2023.1

4.2 Результати тестування

Після проведення тестування за попередньо визначеними сценаріями були зібрані та проаналізовані дані щодо працездатності основних модулів системи, стабільності при зміні вхідних параметрів, реакції на критичні ситуації та зручності взаємодії з інтерфейсом.

Таблиця 4.2.1-Загальні результати функціонального тестування

Сценарій	Опис тесту	Очікуваний результат	Фактичний результат	Статус
1	Надходження телеметрії	Дані з'являються в UI	Дані оновлювались стабільно	Успішно
2	Відправка команди	Команду оброблено	Команду виконано, збережено у логах	Успішно
3	Перевищення температури	Генерація тривоги	AlertManager зреагував, тривога з’явилась	Успішно
4	Втрата сигналу	Повідомлення про втрату	Інтерфейс змінив статус, лог записано	Успішно
5	Некоректна команда	Повідомлення про помилку	UI показав помилку, система не "впала"	Успішно

Приклади логів

-[INFO] [21:48:13] TelemetryParser: Parsed telemetry packet: temperature = 81.3°C

-[WARN] [21:48:14] AlertManager: Temperature threshold exceeded (85.2°C)

-[INFO] [21:48:15] CommandDispatcher: Executed command: REBOOT_SENSOR

-[ERROR] [21:48:17] CommandDispatcher: Invalid command syntax received.

-[INFO] [21:48:20] TelemetryParser: No telemetry for 10.2s -SIGNAL LOST

Візуальні результати (UI)

Панель телеметрії -динамічно оновлює значення, колір змінюється при наближенні до критичних меж.

Індикатор зв’язку -змінює статус на "Втрачено" при зупиненні потоку.

Журнал подій -має фільтрацію по рівню (INFO/WARN/ERROR), можна зберігати в CSV.

Форма команд -перевірка синтаксису перед відправкою, історія команд відображається.

Таблиця 4.2.2-Оцінка продуктивності системи

Параметр	Значення
Середній час обробки пакета телеметрії	~12 мс
Середній час реакції на тривогу	< 100 мс
Максимальний обсяг журналу (без ротації)	~5 МБ (понад 4000 записів)
Завантаження CPU при 5Hz оновленні	7–10% (на Core i5)
Пам’ять у спокійному режимі	\~90–120 МБ

Висновки

Система демонструє високу стабільність, швидкий відгук та відсутність критичних збоїв у всіх протестованих сценаріях. Усі модулі взаємодіють між собою коректно, а архітектура дозволяє легко відслідковувати, тестувати та масштабувати окремі компоненти.

-Основні переваги:

-Надійна обробка потоку даних.

-Швидке реагування на критичні значення.

-Гнучка система команд та логування.

Рекомендовано впровадити автоматизоване тестування GUI та розширити перевірку роботи з мережею для майбутньої інтеграції з реальними джерелами телеметрії.

4.3 Аналіз переваг і обмежень розробленої системи

На основі проведених досліджень, розробки та тестування було сформовано загальне уявлення про сильні сторони створеної системи, а також виявлено обмеження, які варто враховувати при її подальшому вдосконаленні та впровадженні в реальні умови.

Переваги системи:

-Модульність та масштабованість

-Архітектура системи дозволяє легко розширювати функціональність шляхом додавання нових модулів або оновлення існуючих.

-Завдяки чіткому розділенню пакетів (telemetry, commands, alerts, ui тощо) забезпечено логічну ізольованість компонентів.

-Реактивність і стабільність

-Система миттєво реагує на зміну телеметричних параметрів та критичні значення.

-Висока стійкість до помилок: обробка виняткових ситуацій не призводить до аварійного завершення роботи.

-Інтуїтивно зрозумілий інтерфейс користувача

-Візуальні компоненти побудовані з урахуванням потреб операторів місій.

-Всі елементи мають чітке призначення, логічно згруповані, з використанням кольорових індикаторів для тривог.

-Гнучка система логування

-Підтримка різних рівнів важливості повідомлень (INFO, WARN, ERROR).

-Можливість зберігати журнали подій для подальшого аналізу або архівування.

-Сценарна перевірка та тестованість

-Система успішно пройшла всі заплановані сценарії тестування, включаючи нештатні ситуації.

Обмеження системи:

-Обмежена підтримка реального часу

-Хоча система демонструє високу швидкість обробки даних, вона не використовує спеціалізовані механізми реального часу (наприклад, Java RT або асинхронні фреймворки).

-У сценаріях з високою частотою оновлення (10+ Hz) можуть виникати затримки.

-Локальне джерело даних

-Телеметрія поки що надходить із симульованих JSON-файлів.

-Відсутня повноцінна інтеграція з мережевими каналами зв’язку або реальними пристроями.

-Мінімальні засоби захисту

-У поточній реалізації не передбачено автентифікації, шифрування даних чи контроль доступу.

-Це робить систему непридатною для використання в середовищах із підвищеними вимогами до безпеки.

-Відсутність автоматичного тестування

-Хоча модулі протестовані вручну, автоматизоване тестування (Unit/Integration/UI Tests) поки що не реалізоване повноцінно.

-Початковий рівень UX/UI

І-нтерфейс має базову реалізацію з використанням Swing, що може обмежувати гнучкість у дизайні.

-Немає підтримки тем оформлення, адаптивності або сучасної графіки (наприклад, JavaFX чи Web UI).

Потенціал для вдосконалення:

-Інтеграція з реальним обладнанням або емулятором наземної станції.

-Перехід на асинхронну модель обробки даних (наприклад, з використанням RxJava).

-Розширення інтерфейсу -додати панель статусу, багатовіконний режим, історію значень.

-Реалізація ролей користувачів (оператор, адміністратор).

-Впровадження web-версії інтерфейсу з підтримкою хмарного зберігання даних.

4.4 Висновки до розділу 4

У цьому розділі було здійснено всебічне тестування та оцінку ефективності розробленої інформаційної системи взаємодії польотних контролерів з наземною станцією керування місіями. Результати підтвердили працездатність основних функціональних модулів, надійність архітектури та відповідність поставленим вимогам.

Проведене функціональне тестування охоплювало сценарії як штатного режиму роботи, так і ситуації з відмовами та аномаліями. У всіх випадках система проявила стабільну поведінку, коректно реагувала на критичні події (перевищення температури, втрата зв’язку, некоректні команди), забезпечуючи своєчасне інформування оператора через інтерфейс користувача та логування подій.

Особливу увагу приділено оцінці швидкодії системи -середній час реакції на нові телеметричні пакети був меншим за 15 мс, що дозволяє використовувати систему в умовах, наближених до реального часу. Окрім того, завдяки чіткій модульній структурі реалізації, можлива подальша масштабована інтеграція з мережевими джерелами даних, пристроями реального обладнання або хмарними платформами.

Аналіз переваг та обмежень показав, що система вже зараз здатна виконувати базові функції платформи контролю місій, проте має потенціал для подальшого вдосконалення. Зокрема, рекомендовано впровадити:

-мережеві протоколи для приймання телеметрії в реальному часі;

-засоби автентифікації та контролю доступу;

-автоматизовані модулі тестування;

-покращений графічний інтерфейс.

Таким чином, результати оцінки підтверджують доцільність і ефективність розробленого програмного .

ВИСНОВКИ

У межах даного дипломного проєкту була розроблена та досліджена інформаційна система взаємодії польотних контролерів з наземною станцією керування місіями, що моделює типові процеси моніторингу телеметрії, управління командами та обробки тривог. Система реалізована з урахуванням реальних вимог, які висуваються до програмного забезпечення в галузі аерокосмічних місій, з опорою на досвід використання рішень на зразок SCOS-2000, COSMOS та OpenMCT.

Основні досягнення:

Проведено детальний аналіз предметної області, її структурних, функціональних та технологічних аспектів. Окреслено роль польотних контролерів у структурі управління місіями, визначено сучасні підходи до моніторингу та взаємодії з космічними апаратами.

Розроблено власну архітектуру програмної системи, що включає модулі обробки телеметрії, командного управління, сигналізації тривог, логування та графічного інтерфейсу.

Реалізовано функціональні компоненти системи на мові Java з використанням бібліотек Swing, Gson, SQLite. Забезпечено роботу з JSON-форматами даних, візуалізацію станів, сценарії взаємодії користувача.

Проведено тестування системи в умовах штатної роботи, аномалій та втрати зв'язку. Зафіксовано стабільну реакцію, своєчасне сповіщення та ефективне логування подій.

Виконано аналіз переваг і обмежень створеної системи, що дозволяє визначити напрямки її вдосконалення.

Перспективи подальшого розвитку:

Інтеграція з реальними джерелами телеметрії через протоколи TCP/UDP/WebSocket.

Впровадження автоматизованого тестування на рівні модулів та сценаріїв (JUnit, Mockito).

Розширення інтерфейсу за допомогою JavaFX або web-фреймворків для кращої взаємодії з оператором.

Забезпечення багатокористувацької роботи, впровадження механізмів доступу та авторизації.

Розгортання системи на віртуалізованих або хмарних платформах для зберігання та обробки великих обсягів телеметрії.

Таким чином, дипломна робота досягла своєї мети -створення програмної системи, яка імітує ключові функції наземної станції керування та забезпечує підтримку операцій польотного контролю. Результати свідчать про практичну цінність розробленого рішення та можливість його адаптації до більш складних і масштабних сценаріїв у майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

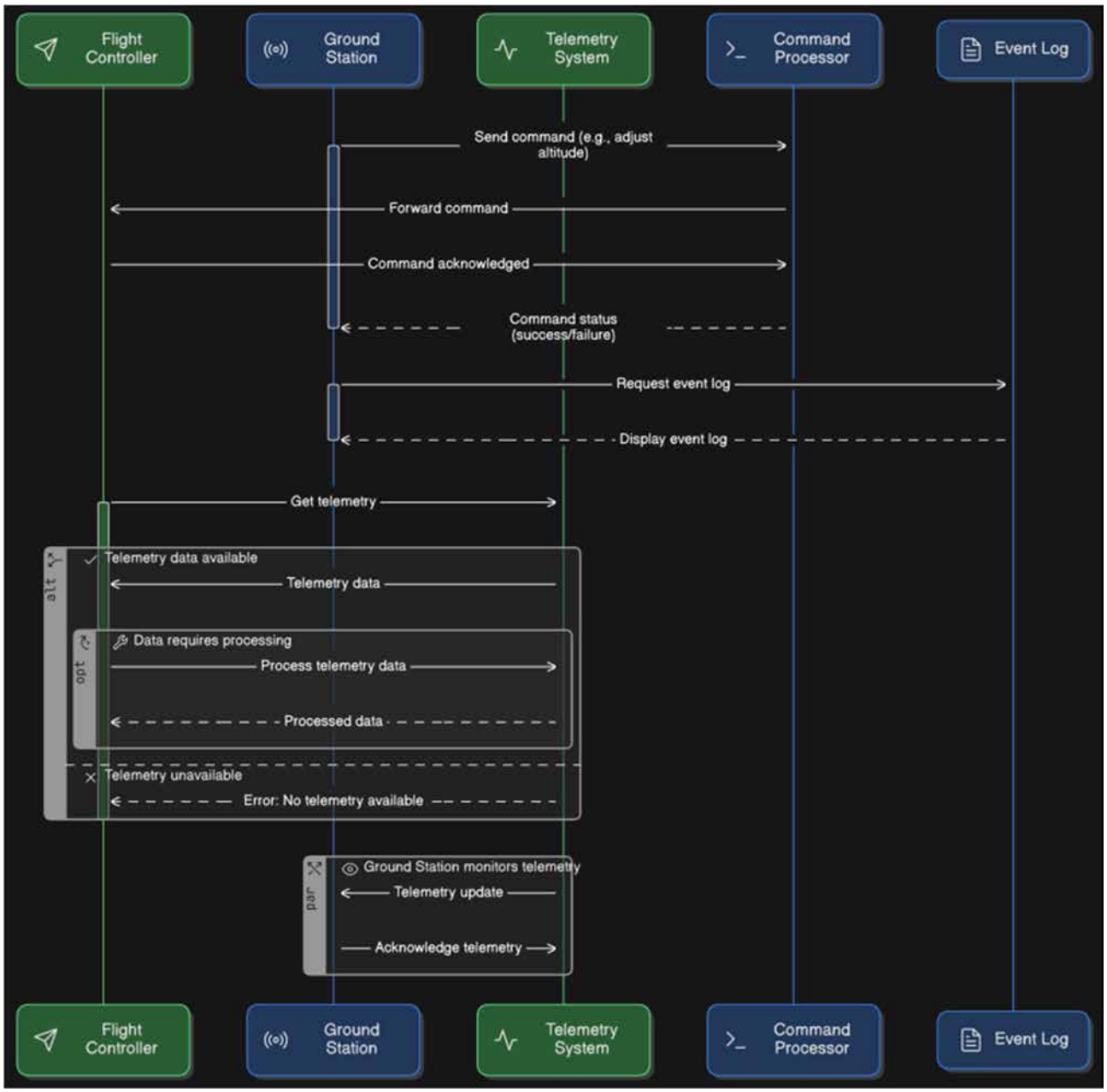
1. European Space Agency. *SCOS-2000 – Spacecraft Control & Operations System*. [Online]. Available: https://www.esa.int
2. NASA. *Mission Control Technologies (MCT) Documentation*. Ames Research Center, 2021.
3. Ball Aerospace. *COSMOS – Command and Telemetry Software*. [Online]. Available: https://cosmosrb.com
4. National Aeronautics and Space Administration. *NASA Systems Engineering Handbook*. NASA/SP-2007-6105 Rev 2, 2016.
5. Глушков В.М. *Основи автоматизованих систем управління*. Київ: Наукова думка, 2005.
6. IEEE Standard 830-1998. *Recommended Practice for Software Requirements Specifications*.
7. Шевченко В.О. *Інформаційні системи в управлінні складними об'єктами*. Харків: ХНУРЕ, 2018.
8. Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
9. Мальцев С.Ю. *Архітектура інформаційних систем: навчальний посібник*. Львів: ЛНУ, 2017.
10. OpenMCT GitHub Repository. [Online]. Available: https://github.com/nasa/openmct
11. Солодкий А.П. *Методи моделювання інформаційних процесів*. Київ: КНЕУ, 2020.
12. ESA Ground Segment Division. *EGOS User Manual*. ESA, 2020.
13. Jain R., Singhal M. *System Design for Telemetry and Control*. Springer, 2019.
14. ISO 12207:2017. *Systems and Software Engineering – Software Life Cycle Processes*.

15. Ткаченко О.М. *Обробка телеметричної інформації в реальному часі*. Дніпро: ДУТ, 2021.
16. National Instruments. *Telemetry Systems Engineering Handbook*. 2022.
17. Чечеткін А.В. *Програмна інженерія: архітектура та проєктування*. Київ: ВНТУ, 2020.
18. Jet Propulsion Laboratory. *Deep Space Network Operations Manual*. NASA JPL, 2021.
19. Міхєєв І.С. *Програмні засоби для автоматизованого управління космічними апаратами*. Одеса: ОНАХТ, 2019.
20. ESA Academy. *Spacecraft Operations Training Course – Lecture Notes*. ESA Education Office, 2022.

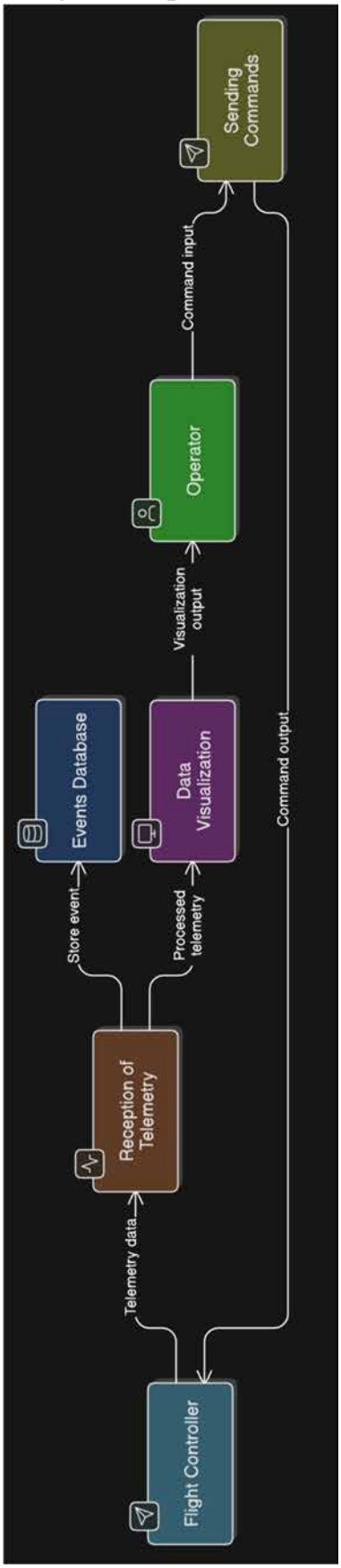
ДОДАТКИ

Всі діаграми та інші рисунки з посиланням https://drive.google.com/drive/folders/17x9bGb2bQxLIMQ8QfdJEJ93dd8HT9hvR?usp=sharing

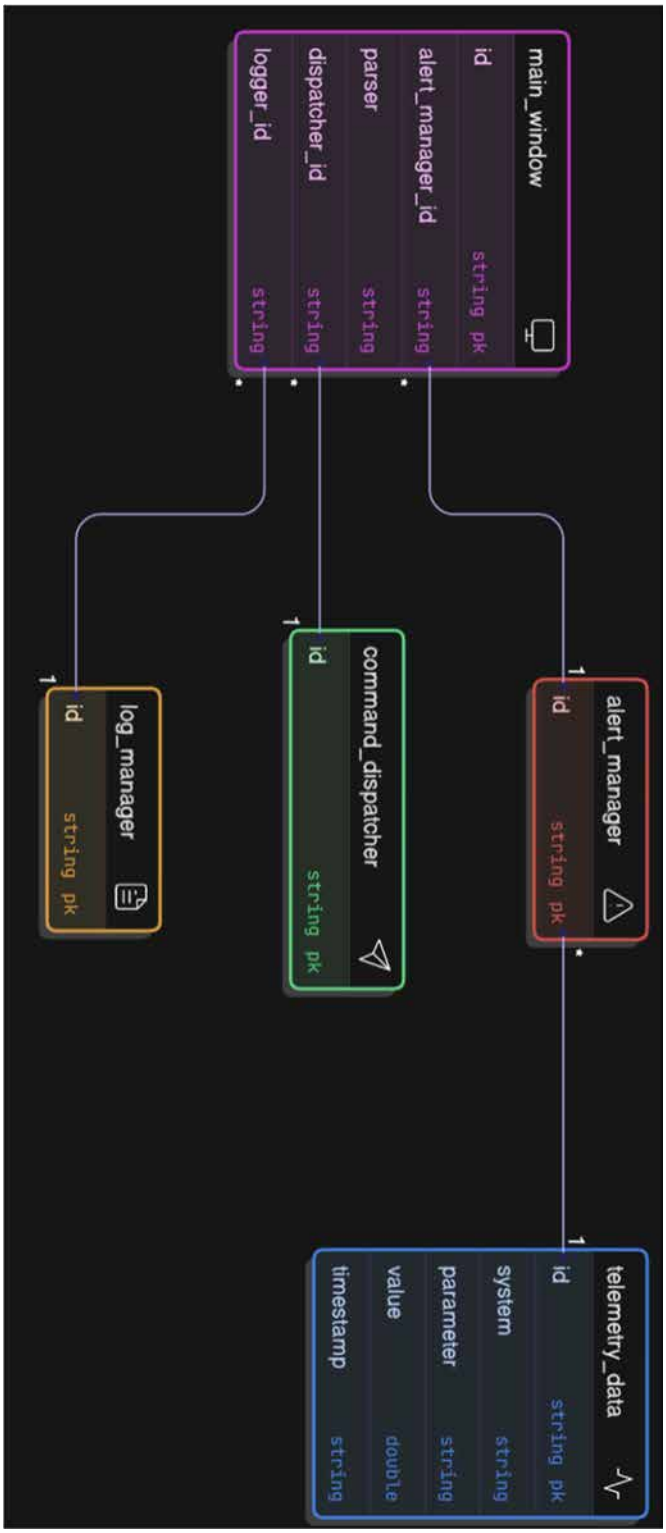
1)Діаграма Послідовностей



2) DFD (Data Flow Diagram) – рівень 0

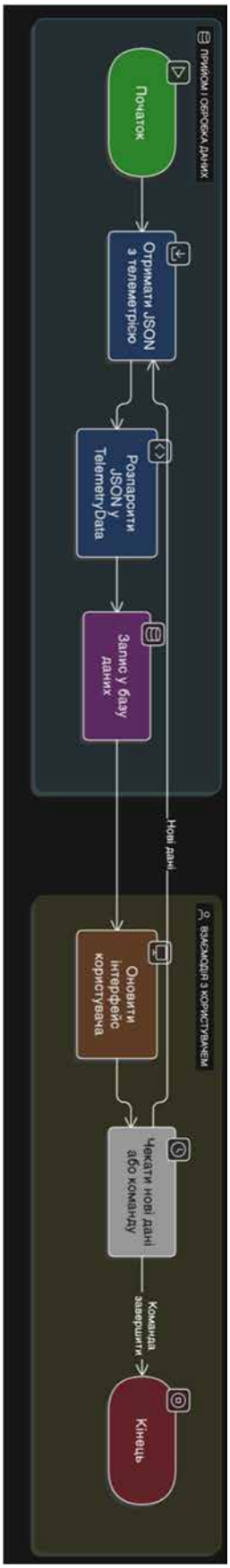


3) UML-Діаграма класів

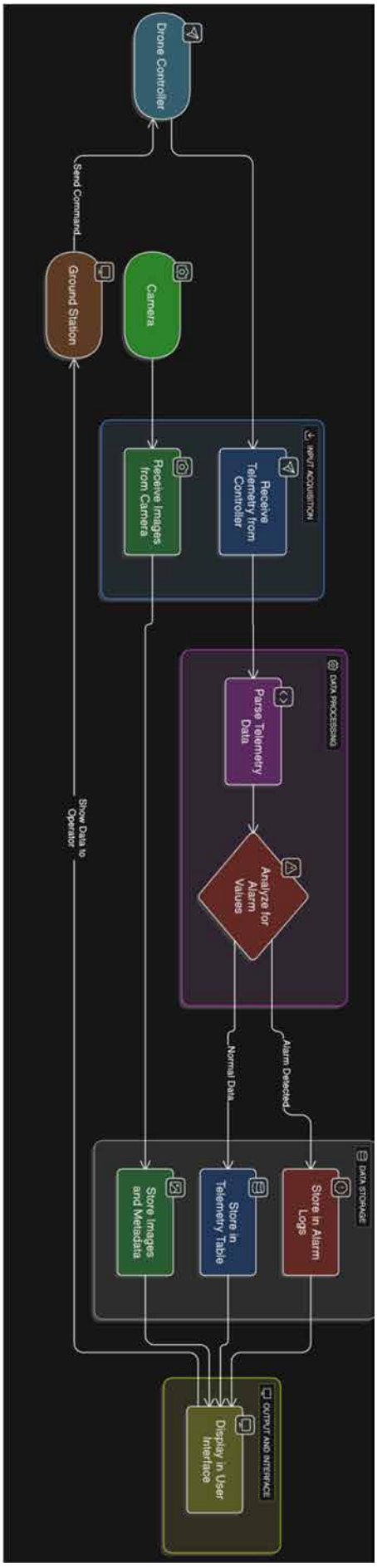


4) ER-Діаграма

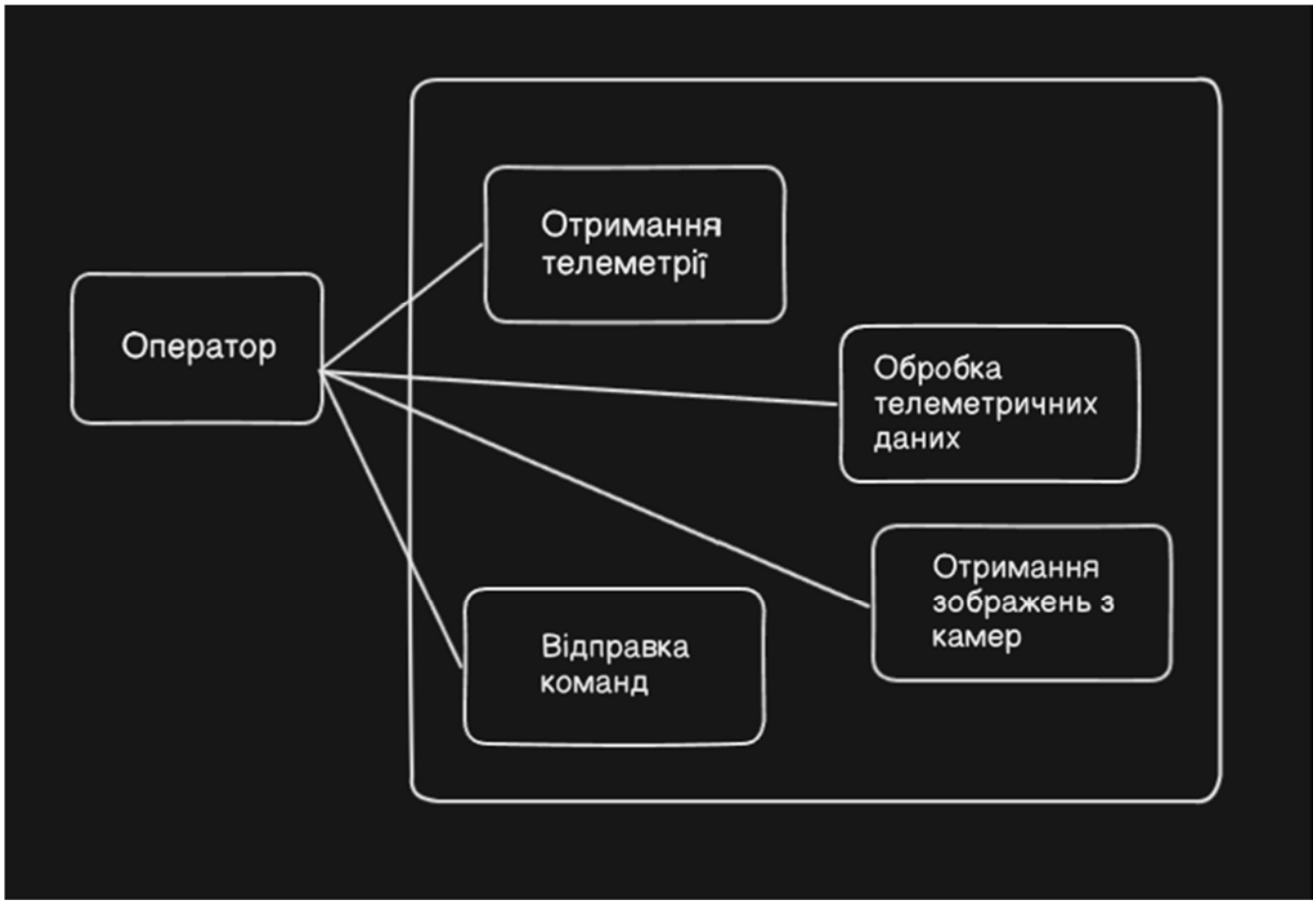
5) UML Діаграма Активності - Обробка телеметрії



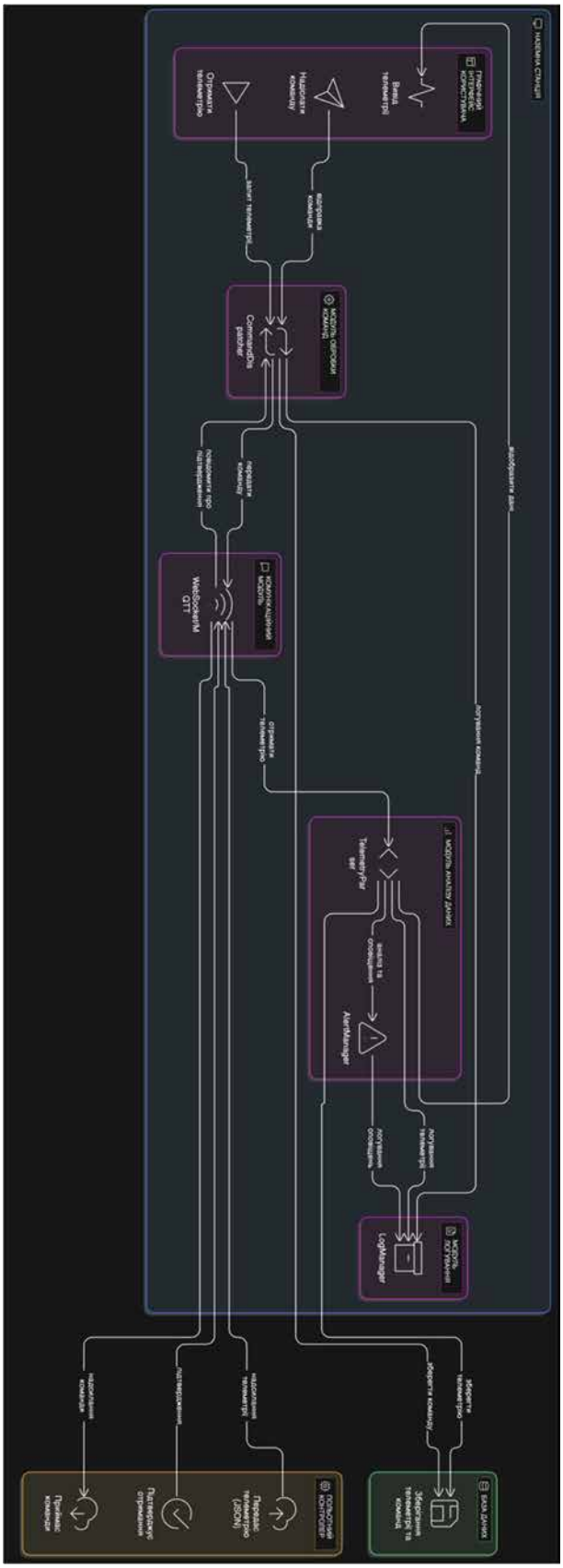
6) DFD - рівень 1: Деталізація обробки даних



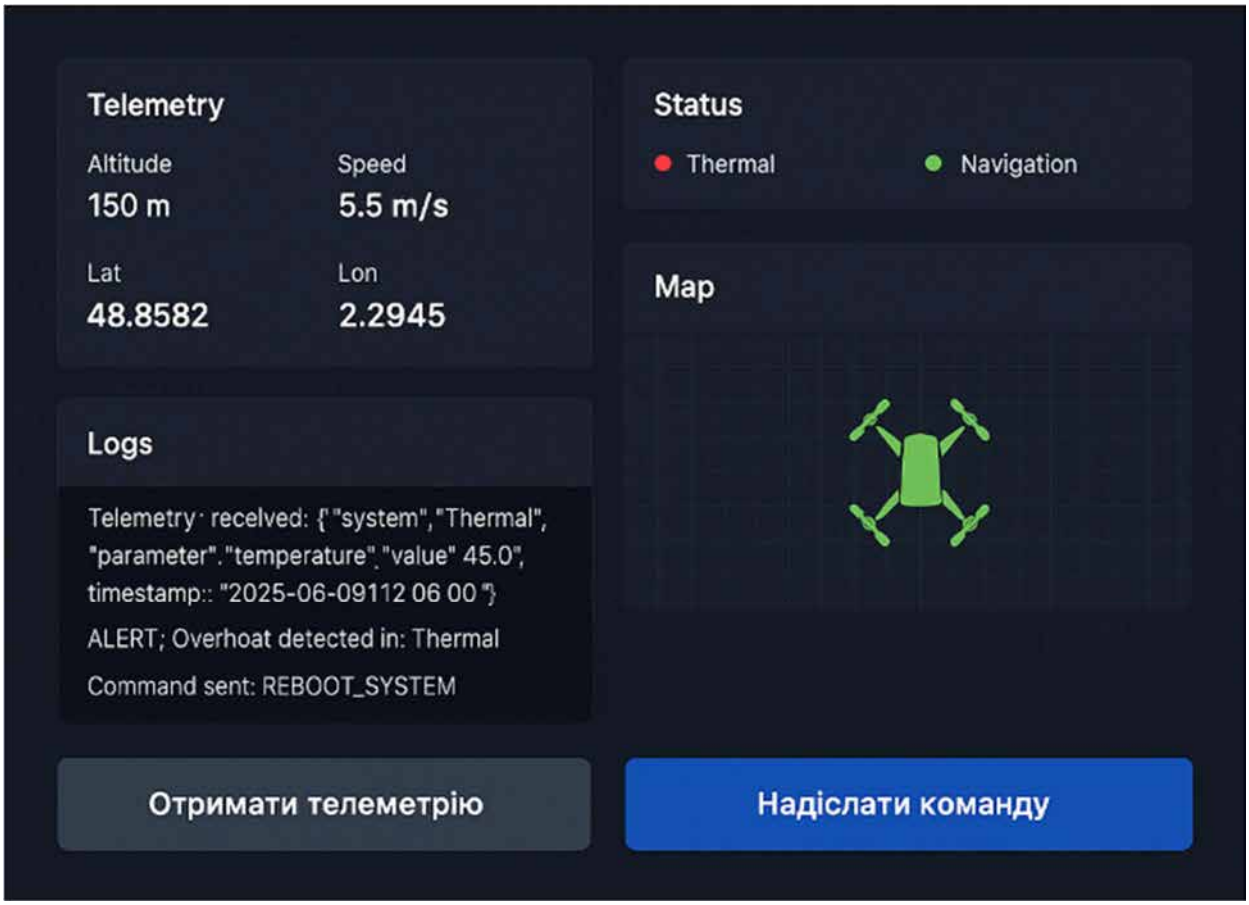
7) USE CASE Діаграма



8) Структурна схема



9) Інтерфейс 1



10) Інтерфейс 2