№ 2(55) 2025

МЕТОДИ ТА ПРИЛАДИ КОНТРОЛЮ ЯКОСТІ

ISSN 1993-9981 print
ISSN 2415-3575 online

https://mpky.nung.edu.ua

Прийнято 10.12.2025. Прорецензовано 24.12.2025. Опубліковано 29.12.2025.

УДК 004.415 DOI: 10.31471/1993-9981-2025-2(55)-190-201

ПРОГРАМНА РЕАЛІЗАЦІЯ MAVLINK-ТЕЛЕМЕТРІЇ ДЛЯ БЕЗПІЛОТНИХ АВІАЦІЙНИХ СИСТЕМ НА БАЗІ КОНТРОЛЕРІВ PIXHAWK

Назаренко О. Г.
Аспірант
АТЗТ «Українські мотори»
61038, Салтівське шосе, 43, м. Харків, Україна
https://orcid.org/0009-0002-2373-1159
e-mail: nazarenko.oleksii@gmail.com

Ушкаренко О. О.
доктор технічних наук, професор
Національний університет кораблебудування імені адмірала Макарова
54007, проспект Героїв України, 9, м. Миколаїв, Україна
https://orcid.org/0000-0002-3159-330X
e-mail: oleksandr.ushkarenko@nuos.edu.ua

Дьяконов О. С.
кандидат технічних наук, доцент
Національний університет кораблебудування імені адмірала Макарова
54007, проспект Героїв України, 9, м. Миколаїв, Україна
https://orcid.org/0000-0001-7438-7066
e-mail: alex.s.dyakonov@gmail.com

Сірівчук А. С.
кандидат технічних наук, доцент
Національний університет кораблебудування імені адмірала Макарова
54007, проспект Героїв України, 9, м. Миколаїв, Україна
https://orcid.org/0000-0003-2927-2600
e-mail: sirivchuka@gmail.com

Запропоноване посилання: Назаренко, О. Г., Ушкаренко, О. О., Дьяконов, О. С., Сірівчук, А. С. & Обрубов, А. В. (2025). Програмна реалізація MAVlink-телеметрії для безпілотних авіаційних систем на базі контролерів Pixhawk. Методи та прилади контролю якості, 2(55), 190-201. doi: 10.31471/1993-9981-2025-2(55)-190-201

* Відповідальний автор

Обрубов А. В.
доктор технічних наук, доцент
Національний університет кораблебудування імені адмірала Макарова
54007, проспект Героїв України, 9, м. Миколаїв, Україна
https://orcid.org/0000-0001-9667-1703
e-mail: andrii.obrubov@nuos.edu.ua

Анотація. У статті представлено комплексне дослідження процесів отримання, парсингу, обробки та аналітичного використання телеметричних даних, що надходять із польотних контролерів стандарту Pixhawk, які широко застосовуються у сфері безпілотних авіаційних систем. Детально описано архітектуру відкритої апаратної платформи Pixhawk, її модульну структуру, принципи взаємодії між сенсорами, контролерами та зовнішніми обчислювальними модулями. Розглянуто склад і функціональні особливості основних сенсорів, які забезпечують формування телеметричних потоків у режимі реального часу. Особливу увагу приділено аналізу комунікаційного протоколу MAVLink, який є базовим для передачі даних між апаратною частиною дрону та наземними станціями. Наводяться практичні приклади реалізації парсингу телеметрії з використанням мови програмування Python та спеціалізованих бібліотек, таких як pymavlink та DroneKit-Python. Аналізуються можливості застосування отриманих даних для моніторингу стану безпілотного апарату в реальному часі, пост-польотного аналізу та розробки систем підвищення автономності та безпеки польотів. У роботі проведено розбір структури пакетів, типів повідомлень, принципів їх кодування, а також описано механізми виявлення та корекції помилок. У дослідженні оцінюються можливості застосування оброблених телеметричних даних для моніторингу технічного стану безпілотного апарату в реальному часі, виконання постпольотного аналізу та оптимізації параметрів керування.

Ключові слова: безпілотний літальний апарат (БПЛА), Pixhawk, телеметрія, MAVLink, інтелектуальне управління, парсинг даних, багатоканальна система збору даних, координати, кібербезпека, автоматична орієнтація, хмарна обробка даних.

Роль телеметрії в сучасних безпілотних авіаційних системах

Стратегічне значення та стрімкий розвиток БПЛА

Безпілотні літальні апарати упродовж останнього десятиліття еволюціонували з вузькоспеціалізованих військових розробок до технології, що має фундаментальне значення для широкого спектра цивільних і оборонних застосувань [1]. Відзначається експоненційне зростання обсягів виробництва та масштабів використання у світі, а в Україні цей процес демонструє особливу динаміку та стратегічну вагу. Поряд із військовими задачами, які охоплюють розвідку, спостереження, коригування вогню та ураження цілей, швидко розширюються цивільні напрями застосування. Дрони слугують інструментами для агромоніторингу, логістики, протипожежних заходів, охорони громадського порядку, картографування та кіновиробництва. Універсальність платформи підкреслює її статус як технології подвійного призначення з мультиплікативним впливом на економіку та безпеку держави.

Телеметрія як основа управління та безпеки польотів

Функціонування сучасного БПЛА спирається на безперервний двобічний обмін даними між апаратом і наземною станцією керування або бортовим комп'ютером. Такий обмін визначається як телеметрія, тобто високоавтоматизований процес збирання даних із віддалених або важкодоступних точок і їх передавання на приймальне обладнання для вимірювання, моніторингу, відображення та запису. Візуальний потік з камери є важливим для оператора, проте не завжди є критичним для забезпечення самого польоту, тоді як телеметрія має визначальне значення. За відсутності стабільного телеметричного каналу безпечне та ефективне управління апаратом стає практично неможливим.

Телеметрія виконує роль цифрової нервової системи БПЛА і забезпечує інформацію, без якої неможливі ключові функції. У режимі реального часу оператор одержує просторові координати, висоту, кути орієнтації, лінійні та кутові швидкості, параметри стану бортових систем, рівень заряду акумулятора та індикатори якості

зв'язку, що дозволяє ухвалювати обґрунтовані рішення щодо керування польотом. Для роботи автопілота [2] телеметрія формує основне джерело навігаційних даних для руху за маршрутом, стабілізації просторового положення, автоматизованого зльоту та посадки, а також інших запрограмованих маневрів, причому алгоритми оцінювання стану на кшталт розширеного фільтра Калмана безперервно інтегрують дані сенсорів з метою підвищення точності. Безперервний моніторинг параметрів підвищує рівень безпеки завдяки своєчасному виявленню відхилень, зокрема надмірної вібрації, перегріву приводів, критичного падіння напруги або втрати супутникового сигналу, що уможливлює автоматичну активацію аварійних сценаріїв на кшталт повернення у точку старту у режимі Return to Launch. Записані телеметричні журнали становлять основу для постпольотного аналізу, діагностики відмов, з'ясування причин інцидентів, тонкого налаштування регуляторів і подальшої оптимізації продуктивності апарата.

Здатність своєчасно одержувати, коректно декодувати та аналітично опрацьовувати телеметричні дані є ключовою компетенцією розробників, дослідників і операторів БПЛА [3].

Мета роботи полягає у наданні комплексного аналізу методів та інструментів для парсингу телеметрії з відкритої апаратної платформи Pixhawk з використанням протоколу MAVLink і мови програмування Python. Для досягнення цієї мети в межах дослідження проаналізовано архітектуру та сенсорне оснащення польотних контролерів стандарту Pixhawk, детально розглянуто структуру та принципи функціонування протоколу MAVLink з урахуванням відомих вразливостей безпеки, наведено і зіставлено практичні підходи до парсингу MAVLink-повідомлень у Python на основі низькорівневих і високорівневих бібліотек, а також продемонстровано напрями використання телеметричних даних для аналізу та керування польотом.

Основою дослідження є поєднання трьох відкритих технологій, що формують цілісну екосистему. Використовується доступне та стандартизоване апаратне забезпечення Pixhawk, яке стало дослідницькою платформою для університетів і стартапів. Залучається відкрите програмне забезпечення автопілотів ArduPilot і PX4, яке забезпечує професійний функціонал без ліцензійних бар'єрів [4]. Застосовується відкритий протокол зв'язку MAVLink як уніфікована мова взаємодії компонентів. Синергія цих елементів демократизувала доступ до передових БПЛА-технологій і змістила фокус досліджень з розроблення пропрієтарних систем на створення інтелектуальних алгоритмів і прикладних рішень на базі надійної стандартної платформи. Наведене дослідження є прикладом результатів, що стали можливими завдяки відкритості та доступності зазначеної екосистеми.

Архітектура системи збору телеметричних даних на базі Pixhawk

Pixhawk як відкритий апаратний стандарт

Pixhawk (рис. 1) слід розглядати не як окрему модель польотного контролера, а як незалежний відкритий апаратний проєкт, який формує та підтримує стандарти проєктування автопілотів. У межах цього проєкту визначаються специфікації апаратної частини, що охоплюють принципові схеми, типи компонентів, їх розміщення і набір інтерфейсів, завдяки чому забезпечується висока сумісність контролерів різних виробників зі стеками прошивок і засобами наземного керування. Еволюція стандарту відбивається у версіях Flight Management Unit, серед яких FMUv3, FMUv4, FMUv5, FMUv6C, FMUv6X. Кожне нове покоління пропонує продуктивніший процесор, розширену номенклатуру сенсорів і додаткові інтерфейси, при цьому на рівні прошивки зберігається зворотна сумісність. Для сучасних зразків, що відповідають вимогам FMUv6C і FMUv6X, зокрема Pixhawk 6C і Pixhawk 6X, характерне використання мікроконтролерів класу STMicroelectronics STM32H743 з ядром Arm Cortex-M7 на частоті 480 МГц або платформи NXP i.MXRT1176 із двома ядрами Cortex-M7 на

1 ГГц та Cortex-M4 на 400 МГц. Така елементна база забезпечує достатній обчислювальний резерв для реалізації алгоритмів навігації, керування і потокової обробки сенсорних даних у режимі реального часу.

Рисунок 1 – Pixhawk 5X

Компонентна база та сенсорне оснащення

Первинні телеметричні дані формуються сенсорним комплексом, інтегрованим на платі польотного контролера, причому для підвищення надійності часто застосовується резервування окремих вимірювальних каналів. Ключову роль у визначенні просторової орієнтації відіграє інерційний вимірювальний блок IMU, який зазвичай містить два або три незалежні набори тривісних акселерометрів і тривісних гіроскопів, наприклад ICM-42688-P, BMI055, BMI088. Акселерометри вимірюють лінійні прискорення з урахуванням гравітаційної складової, а гіроскопи вимірюють кутові швидкості, що у сукупності дозволяє отримувати оцінку стану апарата з необхідною точністю. Для зменшення впливу вібрацій на мультироторних платформах IMU монтують на віброзахисті, а стабільність характеристик підвищують завдяки вбудованим підігрівачам, які підтримують оптимальний тепловий режим чутливих елементів. Визначення курсу доповнюється даними тривісного магнітометра, поширеними є датчики IST8310 і BMM150. Відомо, що точність вбудованих магнітометрів у корпусах контролерів Pixhawk може бути нижчою за еталонні лабораторні прилади, що потрібно враховувати під час розроблення високоточної навігації, тому на практиці часто застосовують зовнішні компаси, винесені від силових кабелів і джерел електромагнітних завад. Висоту за тиском визначає барометр, наприклад MS5611 або BMP390, який забезпечує високу чутливість до малих змін тиску, хоча підпадає під вплив повільного дрейфу через погодні фактори. Положення і швидкість відносно землі, а також еталонний час надає зовнішній модуль супутникової навігації GPS або GNSS, що підключається до контролера через стандартні послідовні інтерфейси.

З архітектурного погляду значна частина контролерів класу Pixhawk використовує двопроцесорну схему. Основний обчислювальний модуль FMU виконує задачі високого рівня, серед яких запуск операційної системи реального часу NuttX, виконання прошивки ArduPilot або PX4, реалізація алгоритмів сенсорної фузії на основі розширеного фільтра Калмана та планування місій [5]. Допоміжний процесор вводу-виводу, типовим прикладом є STM32F103, бере на себе низькорівневі функції керування приводами, генерацію ШІМ сигналів для двигунів і сервоприводів, а також оброблення сигналів з приймача радіокерування [6]. Такий розподіл обов’язків між ядрами зменшує затримки у критичних контурах керування і підвищує загальну відмовостійкість системи.

Екосистема програмного забезпечення: ArduPilot та PX4

Апаратна платформа Pixhawk слугує базою, на якій функціонує одна з двох домінуючих відкритих прошивок firmware – ArduPilot або PX4. Обидві системи перетворюють апаратний контролер на повноцінний автопілот [7], проте відрізняються архітектурними підходами та ліцензійними моделями.

ArduPilot належить до найстаріших і найпоширеніших відкритих прошивок, має велику спільноту і широко підтверджену надійність експлуатації. Архітектура ближча до монолітної, а спектр підтримуваних платформ охоплює мультикоптери, літаки, гелікоптери, наземні ровери, човни та підводні апарати. Завдяки тривалій історії

практичного використання код вважають перевіреним у реальних умовах, що робить ArduPilot поширеним вибором для комерційних застосувань з пріоритетом надійності. Розповсюдження відбувається за ліцензією GPLv3, яка вимагає відкриття похідних робіт.

PX4 походить з дослідницького середовища ETH Zurich і підтримується Dronecode Foundation під егідою Linux Foundation. Система має сучасну модульну архітектуру та використовує механізм обміну повідомленнями uORB, що працює за принципом видавець – підписник. Окремі модулі, такі як драйвери сенсорів, оцінювачі стану і контролери польоту, функціонують як незалежні процеси та взаємодіють через стандартизовані теми topics. Такий підхід забезпечує високу гнучкість, спрощує кастомізацію і інтеграцію нових компонентів, що є привабливим для академічних досліджень і просунутих комерційних розробок. PX4 поширюється за більш дозвільною ліцензією BSD, яка дозволяє створювати закриті комерційні продукти на основі відкритого ядра.

Сила екосистеми полягає у симбіозі трьох відкритих стандартів – апаратної платформи Pixhawk, програмного забезпечення ArduPilot або PX4 та відкритого протоколу зв'язку MAVLink [8]. Апаратний стандарт забезпечує стабільну і передбачувану основу для виконання відкритих прошивок, програмне забезпечення формує критично важливі телеметричні дані, а протокол MAVLink надає уніфікований спосіб доступу до цих даних незалежно від конкретної реалізації апаратури чи вибраної прошивки. Взаємозамінність компонентів створює ключову перевагу платформи. Розробник може замінити контролер Holybro на аналог від CUAV або перейти з ArduPilot [9] на PX4, після чого Python-скрипт, що взаємодіє мовою MAVLink, з великою ймовірністю потребуватиме лише мінімального коригування або взагалі працюватиме без змін. Саме ця глибока інтероперабельність визначає інноваційність і гнучкість усієї екосистеми.

Протокол MAVLink: Структура, механізми передачі та аналіз безпеки

Загальна характеристика та структура повідомлень

MAVLink Micro Air Vehicle Link визначається як легковаговий бінарний протокол серіалізації повідомлень, створений для комунікації між безпілотними апаратами, наземними станціями та іншими компонентами системи. Ключовими перевагами вважаються висока ефективність, що дає змогу працювати на каналах із низькою пропускною здатністю на кшталт радіомодемів, а також розширюваність, яка забезпечує додавання нових типів повідомлень. У спільноті відкритих рішень для дронів MAVLink сформувався як де-факто стандарт.

Структура пакета у версії MAVLink 1.0 має чітко визначені поля. Початок кадру передається одним байтом із фіксованим значенням 0xFE. Довжина корисного навантаження задається одним байтом і може набувати значень від нуля до двохсот п'ятдесяти п'яти. Порядковий номер пакета також кодується одним байтом у діапазоні від нуля до двохсот п'ятдесяти п'яти та використовується для виявлення втрат. Ідентифікатор системи передається одним байтом і визначає джерело або приймач, зокрема за усталеною конвенцією наземна станція часто використовує значення двісті п'ятдесят п'ять, а перший апарат у мережі має значення один. Ідентифікатор компонента всередині системи кодується одним байтом і вказує модуль на кшталт автопілота, камери чи бортового комп'ютера, причому для автопілота поширеним є значення один. Ідентифікатор типу повідомлення кодується одним байтом і визначає формат корисних даних, наприклад для HEARTBEAT застосовується значення нуль, а для GLOBAL_POSITION_INT значення тридцять три. Поле корисного навантаження має змінну довжину від нуля до двохсот п'ятдесяти п'яти байтів, а його структура задається у файлах опису протоколу у форматі XML, зокрема у файлах common.xml та ardupilot.xml. Контрольна сума передається двома байтами і обчис-

Таблиця 1 – Ключові повідомлення телеметрії MAVLink та їх корисне навантаження

ID повідом-лення	Назва повідомлення	Ключові поля корисного навантаження	Опис
0	HEARTBEAT	type, autopilot, base_mode, custom_mode, system_status	"Пульс" системи.
1	SYS_STATUS	voltage_battery, current_battery, battery_remaining, drop_rate_ comm	Детальний статус системи, включаючи напругу, струм та залишок заряду батареї, а також відсоток втрачених пакетів зв'язку.
30	ATTITUDE	time_boot_ms, roll, pitch, yaw, rollspeed, pitchspeed, yawspeed	Дані про кутову орієнтацію (крен, тангаж, рискання) та кутові швидкості апарату. Виражені в радіанах.
33	GLOBAL_ POSITION_INT	lat, lon, alt, relative_alt, vx, vy, vz, hdg	Глобальні координати.
74	VFR_HUD	airspeed, groundspeed, heading, ?ed, groundspeed, heading, throt tle, alt, climb	Дані, оптимізовані для відо-браження на Heads-Up Display (HUD)

люється за алгоритмом CRC16-CCITT над усім пакетом без стартового байта з додаванням секретного байта MAVLINK_CRC_EXTRA, що забезпечує перевірку цілісності та відповідності формату.

Версія MAVLink 2.0 розширює наведений формат і водночас зберігає зворотну сумісність із MAVLink 1.0. Новий початок кадру кодується значенням 0xFD. Діапазон ідентифікаторів типів повідомлень розширено до двадцяти чотирьох бітів, що теоретично дозволяє визначити понад шістнадцять мільйонів різновидів. Передбачено механізм доповнення вже наявних повідомлень новими полями без порушення сумісності зі старими реалізаціями. Додано можливість підписування пакетів для автентифікації та посилення контролю цілісності, однак у типових конфігураціях ця функція використовується нечасто.

Для практичної роботи з телеметрією доцільно володіти переліком ключових повідомлень, за допомогою яких передається основна інформація про стан апарата, оскільки саме вони формують мінімально необхідний набір даних для керування, моніторингу та подальшого аналітичного опрацювання.

Аналіз вразливостей та проблеми безпеки

Попри високу ефективність і широку популярність стандартна реалізація протоколу MAVLink має фундаментальні вразливості безпеки, що робить її чутливою до низки кібератак, і ця проблема активно досліджується в академічному середовищі. Парадокс полягає в тому, що саме простота, легкість та відкритість, які забезпечили успіх протоколу, перетворилися на ключові слабкі місця у сучасних сценаріях застосування безпілотників із критичними вимогами до безпеки. Історично проєкт приділяв пріоритетну увагу ефективності передавання даних, а не їх захисту, тому швидке промислове впровадження випередило розвиток комплексних механізмів безпеки.

Основні вектори атак охоплюють пасивне перехоплення трафіку унаслідок відсутності шифрування за умовчанням, що дає змогу зловмиснику в зоні дії радіоканалу отримувати телеметрію та відновлювати конфіденційні відомості про координати апарата, маршрут, місцезнаходження оператора і стан місії. Відсутність вбудованої автентифікації відправника створює можливості для ін’єкції пакетів, коли сто-

ронній вузол формує синтаксично коректні повідомлення та нав'язує команди зміни режиму польоту, коригування маршрутних точок або вимкнення двигунів, що веде до перехоплення керування або катастрофи. Додаткову загрозу становлять атаки типу людина посередині, під час яких зловмисник активно втручається у канал зв'язку між легітимними сторонами, модифікує повідомлення на льоту та ретранслює їх далі. Виявлені також ризики, пов'язані з фаззінгом, коли надсилання наборів невалідних або напіввалідних повідомлень уразливим реалізаціям парсерів на боці автопілота чи наземної станції спричиняє збої, відмову в обслуговуванні або навіть виконання довільного коду.

Запропоновані у науковій літературі підходи до підвищення захищеності передбачають інтеграцію криптографічних механізмів із мінімальним впливом на затримки та енергоспоживання. Розглядаються варіанти застосування симетричного шифрування на основі AES або ChaCha20 для забезпечення конфіденційності, а також спеціалізовані легковагові рішення на кшталт MAVShield, оптимізовані для ресурсно обмежених мікроконтролерів польотних контролерів. Попри наявність обнадійливих прототипів і експериментальних імплементацій стан справ поки не набув статусу загальноприйнятого стандарту з широким промисловим впровадженням, тому питання безпеки MAVLink залишається відкритим напрямом для подальших досліджень і стандартизації.

Практична реалізація парсингу телеметрії з використанням Python

Python як інструмент для розробки додатків для БПЛА

Вибір мови Python для роботи з телеметрією БПЛА є обґрунтованим з огляду на низку факторів [11]. Це високорівнева мова з читабельним синтаксисом, що прискорює розроблення і прототипування. Вона має розвинену екосистему наукових і інженерних бібліотек, серед яких NumPy для числових обчислень, Pandas для оброблення даних, Matplotlib і Seaborn для візуалізації, що робить її зручною для аналізу телеметричних журналів [10]. Активна спільнота підтримує спеціалізовані пакети для взаємодії з безпілотними платформами, унаслідок чого зменшується бар'єр інтеграції з апаратним забезпеченням.

Типовою архітектурою виконання Python-скриптів на борту є використання так званого бортового комп'ютера, зокрема одноплатної системи Raspberry Pi під керуванням Linux. Такий модуль під'єднують до одного з послідовних портів UART польотного контролера Pixhawk і організовують обмін даними за протоколом MAVLink. Подібна конфігурація переносить обчислювально складні задачі на більш потужний процесор бортового комп'ютера і розвантажує мікроконтролер автопілота, що дає змогу повною мірою користуватися можливостями екосистеми Python для комп'ютерного зору, оптимізації траєкторій і методів машинного навчання.

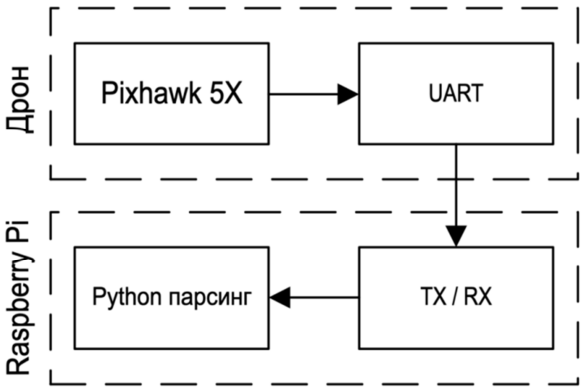

Рисунок 2 – Схема передачі даних

Для парсингу повідомлень MAVLink у Python застосовують два підходи, які відповідають різним рівням абстракції. Низькорівневий підхід базується на бібліотеці pymavlink і надає прямий доступ до структури протоколу з можливістю тонкого контролю процесу декодування. Високорівневий підхід використовує фреймворк DroneKit-Python і пропонує зручні об'єктні інтерфейси до типових сутностей на кшталт параметрів, місій і телеметричних потоків, що прискорює прикладну розробку за мінімальних накладних витрат на інтеграцію.

Низькорівневий підхід: бібліотека pymavlink

Бібліотека pymavlink є фундаментальною реалізацією протоколу MAVLink на Python. Вона надає інструменти для генерації, відправки, прийому та парсингу сирих MAVLink-повідомлень. Це дає розробнику повний контроль над комунікаційним процесом, але вимагає глибшого розуміння самого протоколу.

Практична реалізація парсера телеметрії за допомогою pymavlink зазвичай включає наступні кроки:

1. **Встановлення з'єднання:** Створюється об'єкт з'єднання за допомогою функції mavutil.mavlink_connection. Як аргумент передається рядок, що описує тип з'єднання. Для підключення до Pixhawk через USB-кабель на Linux-системі це може бути шлях до послідовного порту, наприклад, '/dev/ttyACM0', з вказанням швидкості передачі (baud=57600). Для підключення через мережу (наприклад, до симулятора або через Wi-Fi міст) використовується UDP, наприклад, 'udp:127.0.0.1:14550'.

```
Python
from pymavlink import mavutil

# Створення об'єкта з'єднання
master = mavutil.mavlink_
connection('/dev/ttyACM0', baud=57600)
```

2. **Очікування Heartbeat:** Перед початком роботи необхідно переконатися, що зв'язок з автопілотом встановлено. Найнадійніший спосіб – дочекатися першого повідомлення HEARTBEAT.

```
Python
# Очікування першого heartbeat
master.wait_heartbeat()
print(f"Heartbeat from system (system
{master.target_system} component
{master.target_component})")
```

3. **Цикл прийому та парсингу повідомлень:** Основна логіка парсера реалізується в нескінченному циклі. Метод master.recv_match() блокує виконання до отримання нового повідомлення. Він автоматично обробляє сирий потік байтів, знаходить MAVLink-пакети, перевіряє контрольну суму та повертає розпарсений об'єкт повідомлення.

```
Python
while True:
  try:
    # Очікування нового повідомлення
    msg = master.recv_match(blocking=True)
    if not msg:
      continue
    # Фільтрація за типом повідомлення
    if msg.get_type() == 'ATTITUDE':
      # Вилучення даних з полів
      roll_rad = msg.roll
      pitch_rad = msg.pitch
      yaw_rad = msg.yaw
      print(f"Attitude: Roll={roll_rad:.2f},
Pitch={pitch_rad:.2f}, Yaw={yaw_rad:.2f}")

    if msg.get_type() ==
'GLOBAL_POSITION_INT':
      lat = msg.lat / 1e7
      lon = msg.lon / 1e7
      alt = msg.relative_alt / 1000.0 # в мет-
рах
      print(f"Position: Lat={lat}, Lon={lon},
Alt={alt}m")
  except Exception as e:
    print(e)
```

Цей підхід надає максимальну гнучкість, оскільки дозволяє працювати з абсолютно будь-яким типом MAVLink-повідомлень, включаючи нестандартні (діалектні) або нові, ще не підтримувані високорівневими бібліотеками.

Високорівневий підхід: фреймворк DroneKit-Python

DroneKit-Python – це фреймворк, що є високорівневою обгорткою над pymavlink. Його головною метою є спрощення розробки автономних додатків для БПЛА, абстрагуючись від складнощів протоколу MAVLink. Замість того, щоб вручну ловити та парсити повідомлення, DroneKit на-

дає об'єктно-орієнтований API, де дрон представляється у вигляді об'єкта vehicle, а його телеметричні дані – як атрибути цього об'єкта.

Ключова перевага цього підходу полягає в тому, що розробнику не потрібно писати низькорівневий код для обробки MAVLink-повідомлень. Фреймворк робить це "під капотом", автоматично підтримуючи актуальний стан об'єкта vehicle.36

Приклад отримання телеметрії за допомогою DroneKit-Python:

1. *Імпорт та підключення:* Процес підключення схожий на pymavlink, але використовується функція connect. Параметр wait_ready=True гарантує, що функція поверне керування лише після того, як буде встановлено зв'язок та завантажено основні параметри з автопілота.

```
Python
from dronekit import connect
import time
# Підключення до апарату
vehicle = connect('/dev/ttyACM0',
wait_ready=True, baud=57600)
```

2. *Прямий доступ до телеметрії:* Після підключення телеметричні дані доступні як атрибути об'єкта vehicle. Наприклад, для отримання висоти, орієнтації та напруги батареї можна просто звернутися до відповідних полів.

```
Python
print(f"Altitude:
{vehicle.location.global_relative_frame.alt}m")
print(f"Attitude: {vehicle.attitude}")
print(f"Battery: {vehicle.battery.voltage}V")
```

3. *Використання слухачів (Listeners):* Для додатків, що працюють в реальному часі, постійне опитування атрибутів у циклі є неефективним. DroneKit пропонує більш елегантний механізм "слухачів" (або "спостерігачів"), який дозволяє зареєструвати функцію зворотного виклику (callback), що буде автоматично викликатися при оновленні певного атрибута.

```
Python
def attitude_callback(self, attr_name, value):
   print(f"ATTITUDE update: {value}")
# Додавання слухача для атрибуту 'attitude'
vehicle.add_attribute_listener('attitude',
attitude_callback)
# Головний цикл програми може займатися іншими справами
while True:
   time.sleep(1)
```

Цей асинхронний підхід є набагато більш ефективним і є рекомендованим для розробки складних додатків.

Вибір інструменту

Вибір між pymavlink і DroneKit-Python є типовим інженерним компромісом між повнотою контролю та зручністю розроблення. Ці засоби належать до різних рівнів абстракції однієї програмної драбини. Бібліотека pymavlink відповідає нижчому рівню, надає доступ до кожного байта протоколу і забезпечує максимальну продуктивність, однак вимагає вищої кваліфікації, ручної реалізації логіки оброблення повідомлень і ретельного управління станами. Фреймворк DroneKit-Python відповідає вищому рівню, пропонує інтуїтивні об’єктні інтерфейси і прискорює прикладну розробку, проте ціною стає часткова втрата гнучкості, оскільки не всі рідкісні типи MAVLink-повідомлень доступні безпосередньо, а накладні витрати можуть бути більшими.

Доцільність використання кожного інструмента визначається характером завдання. За потреби створити алгоритм керування на основі нестандартних типів повідомлень або виконати глибокий протокольний аналіз є сенс обрати pymavlink як базовий інструмент. Для розроблення типових автономних сценаріїв на кшталт місій за маршрутними точками або режимів супроводження цілі практичною буде робота з DroneKit-Python, оскільки час до працездатного прототипу істотно скорочується, а обсяг коду зменшується.

Таблиця 2 – Порівняльний аналіз pymavlink та DroneKit-Python

Критерій	**pymavlink**	**DroneKit-Python**
Рівень абстракції	Низький	Високий
Простота використання	Складна, вимагає знання протоколу MAVLink	Проста, інтуїтивно зрозумілий об'єктно-орієнтований API
Гнучкість	Дуже висока, повний контроль над усіма аспектами протоколу	Середня, обмежена функціоналом, наданим API vehicle
Продуктивність	Висока, мінімальні накладні витрати	Середня, додатковий шар абстракції може вносити невелику затримку
Підтримка MAVLink	Повна, підтримує всі стандартні та діалектні повідомлення	Підтримка основних повідомлень, необхідних для керування та телеметрії
Основний сценарій	Розробка GCS, аналізаторів протоколу, реалізація нестандартних функцій	Швидка розробка автономних місій, освітні проєкти, прототипування

Висновки та перспективи подальших досліджень

Поєднання відкритої апаратної платформи Pixhawk, стандартизованого протоколу MAVLink і мови програмування Python формує гнучку та доступну екосистему для одержання і аналітичного опрацювання телеметрії безпілотних літальних апаратів. Аналіз архітектури Pixhawk і його сенсорного комплексу свідчить, що сучасні польотні контролери надають повний набір вимірювань для реалізації алгоритмів навігації та керування високої складності. Порівняння підходів до парсингу показало, що pymavlink забезпечує повний контроль і є корисним для дослідницьких задач і нестандартних рішень, тоді як DroneKit-Python значно спрощує і прискорює створення прикладних автономних систем завдяки зручним інтерфейсам. Вибір між цими підходами слід здійснювати з урахуванням вимог конкретного проєкту і прийнятного балансу між швидкістю розроблення та гнучкістю.

Окрему увагу приділено питанням безпеки MAVLink. Виявлено, що відсутність вбудованих механізмів конфіденційності та автентифікації у стандартній реалізації залишає канал телеметрії вразливим до перехоплення і ін’єкції команд, що зумовлює потребу у подальшій стандартизації захисних рішень і впровадженні перевірених криптографічних механізмів.

Подальші дослідження доцільно спрямувати на створення і впровадження легковагових криптографічних надбудов над MAVLink із гарантованим балансом між рівнем захисту та затримками передавання, а також з урахуванням обмежених ресурсів мікроконтролерів автопілотів. Перспективним є застосування методів глибинного навчання для аналізу телеметричних потоків з метою виявлення аномалій і прогнозування відмов за допомогою рекурентних архітектур і трансформерів. Значний науковий і прикладний інтерес становить побудова децентралізованих систем керування роєм із прямим обміном телеметрією між апаратами у межах протоколу MAVLink, що відкриває шлях до узгодженої колективної поведінки, кооперативного уникнення зіткнень і виконання спільних завдань без залежності від централізованої наземної станції.

Подяки
Відсутні.

Конфлікт інтересів
Відсутні.

Список використаних джерел / References

1. Akib A. S. M. A. S. et al. Efficient Route Planning and Navigation in Drones Using Pixhawk Autopilot. *2025 6th International Conference on Artificial Intelligence, Robotics and Control (AIRC)*. IEEE. 2025. doi: 10.1109/airc64931.2025.11077519

2. ArduPilot. ArduPilot Development Team. URL: https://ardupilot.org/

3. Unmanned aerial vehicle. URL: https://uk.wikipedia.org/wiki/Безпілотний_літальний_апарат [in Ukrainian].

4. ArduPilot vs PX4: The Ultimate Guide to Open-Source Flight Control. A-Bots. URL: https://a-bots.com/blog/PX4-vs-ArduPilot.

5. Feng L., Fangchao Q. Research on the Hardware Structure Characteristics and EKF Filtering Algorithm of the Autopilot PIXHAWK. 2016 International Conference on Intelligent, Mechatronics and Control Engineering (IMCCC). 2016. doi: 10.1109/IMCCC.2016.128

6. Gharibi M., Boutaba R., Waslander S. L. Internet of Drones. *IEEE Access*. 2016. Vol. 4. P. 1148–1162. doi: 10.1109/ACCESS.2016.2537208

7. Holybro Pixhawk 6C. PX4 Autopilot. URL: https://docs.px4.io/main/en/flight_controller/pixhawk6c.html.

8. Holybro Pixhawk 6X-RT. Holybro. URL: https://holybro.com/products/pixhawk-6x-rt (дата звернення: 20.07.2025).

9. ArduPilot vs PX4: The Ultimate Guide to Open-Source Flight Control. A-Bots. URL: https://a-bots.com/blog/PX4-vs-ArduPilot.

10. Kumar A. et al. PI Drone using Python. ResearchGate. URL: https://www.researchgate.net/publication/358227145_PI_Drone_using_Python

11. Villacis J. L. P. et al. Telemetry and Video Surveillance System in a UAV for the Control and Monitoring of Long-Distance Missions. *Lecture Notes in Computer Science*. 2020. Vol. 12253. P. 556–569.

SOFTWARE IMPLEMENTATION OF MAVLINK TELEMETRY FOR UNMANNED AIRCRAFT SYSTEMS USING PIXHAWK FLIGHT CONTROLLERS

Nazarenko O. H.
Postgraduate student
ATZT "Ukrainian Motors"
61038, Saltivske Shosse, 43, Kharkiv, Ukraine
https://orcid.org/0009-0002-2373-1159
e-mail: nazarenko.oleksii@gmail.com

Ushkarenko O. O.
Doctor of Technical Sciences, Professor
Admiral Makarov National University of Shipbuilding
54007, Heroes of Ukraine Avenue, 9, Mykolaiv, Ukraine
https://orcid.org/0000-0002-3159-330X
e-mail: oleksandr.ushkarenko@nuos.edu.ua

Dyakonov O. S.
Candidate of Technical Sciences, Associate Professor
Admiral Makarov National University of Shipbuilding
54007, Heroes of Ukraine Avenue, 9, Mykolaiv, Ukraine
https://orcid.org/0000-0001-7438-7066
e-mail: alex.s.dyakonov@gmail.com

Sirivchuk A. S.
Candidate of Technical Sciences, Associate Professor
Admiral Makarov National University of Shipbuilding
54007, Heroes of Ukraine Avenue, 9, Mykolaiv, Ukraine
https://orcid.org/0000-0003-2927-2600
e-mail: sirivchuka@gmail.com

Obrubov A. V.
Doctor of Technical Sciences, Associate Professor
National University of Shipbuilding Admiral Makarov
54007, Heroes of Ukraine Avenue, 9, Mykolaiv, Ukraine
https://orcid.org/0000-0001-9667-1703
e-mail: andrii.obrubov@nuos.edu.ua

Abstract. This article presents a comprehensive study of the acquisition, parsing, processing, and analytical use of telemetry data from Pixhawk-standard flight controllers, which are widely employed in unmanned aircraft systems (UAS). The architecture of the open Pixhawk hardware platform, its modular structure, and the principles of interaction among sensors, controllers, and external compute modules are described in detail. We examine the composition and functional characteristics of the primary sensors that generate real-time telemetry streams. Special attention is paid to the MAVLink communication protocol, which underpins data transfer between the drone's hardware and ground stations. Practical examples of telemetry parsing implemented in Python with specialized libraries such as pymavlink and DroneKit-Python are provided. We analyze how the resulting data can be used for real-time health monitoring of the UAV, post-flight analysis, and the development of systems that enhance flight autonomy and safety. The paper dissects packet structures, message types, and their encoding principles, and describes error-detection and correction mechanisms. The study also evaluates the use of processed telemetry for real-time monitoring of the UAV's technical condition, for post-flight analysis, and for optimizing control parameters.

Keywords: unmanned aerial vehicle (UAV), Pixhawk, telemetry, MAVLink, intelligent control, data parsing, multichannel data acquisition system, coordinates, cybersecurity, automatic orientation, cloud data processing.