Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних систем та комп'ютерного моделювання
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи
перший (бакалаврський)
(рівень вищої освіти)

на тему: *Розроблення ГІС для екологічного моніторингу лісів*

Виконав: студент 2 курсу групи ІСТС-21
спеціальності
126 " Інформаційні системи та технології "
(шифр і назва напряму підготовки, спеціальності)

Ковальчук Р.К.
(прізвище та ініціали)

Керівник Часковський О.Г.
(прізвище та ініціали)

Рецензент Карашецький В. П.
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра інформаційних систем та комп'ютерного моделювання
Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 " Інформаційні системи та технології"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІСКМ
_____ Сторожук О.Л.
" 15 " 11 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Ковальчуку Роману Костянтиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розроблення ГІС для екологічного моніторингу лісів»*
керівник роботи *Часковський Олег Григорович, к.т.н., доцент*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від *"15" 11 2024 року № С-884*
2. Термін подання студентом роботи *12.06.2025р.*
3. Вихідні дані до роботи: Основні параметри та вимоги до налаштування дронів з використанням CLI-команд у системі Betaflight для застосування в екологічному моніторингу лісів у складі ГІС

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
1) Стан проблемної області;
2) Інформаційне та математичне забезпечення;
3) Програмне та технічне забезпечення.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація до диплому
6. Дата видачі завдання *18 листопада 2024 року*

Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літературних даних та інших джерел згідно досліджуваної теми*	*18.11.2024 – 15.12.2024*	*виконано*
2.	*Аналіз досліджуваної теми та вибір відповідних варіантів її розробки*	*16.12.2024 – 29.12.2024*	*виконано*
3.	*Постановка задачі та її формалізація*	*30.12.2024 – 12.01.2025*	*виконано*
4.	*Вибір та обґрунтування методів і засобів проведення дослідження*	*13.01.2025 – 26.01.2025*	*виконано*
5.	*Розроблення концептуальної схеми реалізації завдання*	*27.01.2025 – 16.02.2025*	*виконано*
6	*Програмна реалізація завдання*	*17.02.2025 – 13.04.2025*	*виконано*
7.	*Тестування програмного продукту та отриманих результатів*	*14.04.2025 – 27.04.2025*	*виконано*
8.	*Розробка пояснювальної записки магістерської роботи*	*28.04.2025 – 01.06.2025*	*виконано*
9.	*Коригування пояснювальної записки згідно вимог, розроблення презентації*	*02.06.2025 – 12.06.2025*	*виконано*

Студент *(підпис)* *Ковальчук Р.К.* *(прізвище та ініціали)*

Керівник роботи *(підпис)* *Часковський О.Г.* *(прізвище та ініціали)*

АНОТАЦІЯ

Дипломна робота містить 56 сторінок пояснювальної записки, 32 рисунки, 2 додатки, 15 джерел.

У роботі представлено розробку програмного комплексу для генерації CLI-команд для контролерів польоту Betaflight.

Під час проектування та створення програмного додатку використовувались мова програмування Python та бібліотека для побудови графічного інтерфейсу CustomTkinter. Результатом роботи розробленого програмного комплексу є автоматична генерація налаштувань для контролера польоту у форматі CLI-команд. Реалізовано можливість копіювання згенерованих команд та збереження їх у файл для подальшого застосування у Betaflight Configurator.

Ключові слова: *налаштування, командний рядок, Python, Betaflight, FPV, дрон, графічний інтерфейс.*

ABSTRACT

The thesis contains 56 pages of explanatory note, 32 figures, 2 appendices, 15 sources.

The work presents the development of a software complex for generating CLI commands for Betaflight flight controllers.

During the design and creation of the software application, the Python programming language and the CustomTkinter library for building a graphical user interface were used. The result of the developed software complex is the automatic generation of settings for the flight controller in CLI command format. The ability to copy generated commands and save them to a file for further use in Betaflight Configurator has been implemented.

Keywords: *object, function, AutoLisp, VisualLisp, DCL, algorithm, program code.*

ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно розробити програмне забезпечення автоматизованого налаштування дронів для виконання завдань екологічного моніторингу у складі геоінформаційної системи (ГІС). Програмне забезпечення має бути реалізоване у вигляді десктоп-додатку з графічним інтерфейсом користувача, створеного засобами Python, з використанням бібліотек PyQt5 або tkinter, та підтримкою експорту команд налаштування у форматі CLI для середовища Betaflight.

Основні функціональні можливості:

- Введення параметрів дрона: модель польотного контролера, тип приймача, відеопередавача, камери, UART-портів тощо;
- Генерація послідовності CLI-команд налаштування відповідно до заданої конфігурації;
- Збереження сформованої конфігурації у текстовий або JSON-файл для подальшого використання у Betaflight Configurator;
- Можливість імпорту раніше збережених конфігурацій та редагування;
- Вибір та створення шаблонів для типових конфігурацій дрона.

Програмне забезпечення має бути інтуїтивно зрозумілим, з мінімальними вимогами до підготовки користувача, і дозволяти швидке повторне використання конфігурацій.

Усі конфігураційні дані мають бути представлені у структурованому вигляді з можливістю генерації команди save у фіналі скрипта. Інтерфейс користувача має містити поля введення, списки вибору та кнопки з чітким функціональним призначенням.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

CLI – Command Line Interface (інтерфейс командного рядка);

FPV – First Person View (режим польоту від першої особи з передачею відео в реальному часі);

FC – Flight Controller (польотний контролер – основний обчислювальний модуль [redacted]);

GUI – Graphical User Interface (графічний інтерфейс користувача);

OSD – On-Screen Display (вивід телеметрії на відеоекран пілота);

PID-тюнінг – налаштування параметрів стабілізації польоту у Betaflight;

Blackbox – система логування польотних даних у Betaflight;

RX – Receiver (приймач радіосигналу);

TX – Transmitter (передавач радіосигналу);

UART – Universal Asynchronous Receiver-Transmitter (послідовний інтерфейс обміну даними);

VTX – Video Transmitter (відеопередавач FPV-системи).

ВСТУП

Сучасні екологічні виклики, зокрема зміна клімату, вирубка лісів, пожежі та деградація природного середовища, потребують ефективних рішень для моніторингу та управління природними ресурсами. Одним з найперспективніших напрямів є застосування [redacted] апаратів [redacted]) у поєднанні з геоінформаційними системами (ГІС). Завдяки цьому можливо оперативно отримувати, обробляти та аналізувати просторові дані щодо стану лісових масивів.

Однак для ефективного використання дронів у ГІС-системах необхідне їх точне та гнучке налаштування під конкретні задачі (аерофотозйомка, тепловізійне спостереження, моніторинг вирубок, визначення індексів рослинності тощо). В сучасних умовах процес налаштування [redacted] – зокрема у програмному середовищі Betaflight – здійснюється вручну, що вимагає значних знань, часу та не виключає помилок. Такий підхід є неефективним у випадку використання великої кількості дронів або постійної зміни умов місії.

Актуальність теми полягає у необхідності розробки програмного рішення, яке дозволить автоматизувати процес конфігурування дронів, спростити призначення параметрів, зменшити ризик людських помилок та підвищити якість екологічного моніторингу. Особливо цінною така система буде для лісового господарства, служб моніторингу надзвичайних ситуацій, наукових досліджень та природоохоронних організацій.

Об’єктом дослідження є процес створення програмного забезпечення для автоматизованого налаштування дронів, які використовуються в ГІС для екологічного моніторингу лісів.

Предметом дослідження є методи і засоби розробки програмного забезпечення на мові Python для генерації CLI-команд налаштування [redacted] у середовищі Betaflight з можливістю інтеграції в ГІС-системи.

Метою роботи є розроблення програмного додатку для автоматизованого формування конфігурацій [redacted] на базі вибраних параметрів з подальшим застосуванням у задачах екологічного моніторингу в рамках ГІС.

Практичне значення розробленого рішення полягає у можливості значного скорочення часу на підготовку [redacted] до польоту, зменшення кількості помилок при налаштуванні, стандартизації параметрів польоту та забезпеченні кращої інтеграції з ГІС-додатками.

Таким чином, розроблення інструменту для автоматизованого конфігурування [redacted] дозволяє покращити технологічний ланцюг екологічного моніторингу та зробити його більш ефективним, гнучким і масштабованим.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Актуальність завдання та характеристика об’єкта проєктування

Сучасні екологічні виклики, зокрема зміна клімату, незаконна вирубка лісу, деградація екосистем і збільшення частоти лісових пожеж, вимагають впровадження ефективних систем моніторингу природного середовища. Геоінформаційні системи (ГІС) є одним з ключових інструментів для аналізу стану лісових масивів, картографування змін, прогнозування ризиків та прийняття управлінських рішень на основі просторових даних. Особливої актуальності набуває використання [redacted] апаратів ([redacted]) як мобільного засобу збору даних для ГІС, як і для багатьох інших областей людської діяльності.

[redacted] забезпечують оперативну аерофотозйомку, тепловізійну розвідку, точковий моніторинг рослинності, виявлення ознак незаконної вирубки, оцінку біомаси, фіксацію лісових пожеж або деградації ґрунтів. Завдяки високій мобільності, компактності та автономності, вони дозволяють здійснювати зондування навіть у важкодоступних регіонах. Усе це робить [redacted] ефективним джерелом даних для екологічного моніторингу.

Проте розгортання [redacted] у масштабних екологічних проєктах пов’язане з низкою практичних труднощів. Однією з таких є конфігурування апаратної платформи – зокрема, налаштування польотного контролера. У багатьох моделях [redacted], що працюють на базі прошивки Betaflight, конфігурування здійснюється за допомогою текстових CLI-команд (Command Line Interface). Ці команди відповідають за налаштування UART, типу приймача, PID-регуляторів, відеопередавача, параметрів камери, режимів польоту та інших важливих аспектів.

Ручне введення CLI-команд є не лише технічно складним для недосвідчених користувачів, але й суттєво уповільнює розгортання [redacted] у польових умовах. Особливо це критично при масовому використанні [redacted], в прикладі ГІС це буде:

моніторинг великих лісових площ або створення регулярних маршрутів зондування.

У зв’язку з цим актуальною стає задача автоматизації генерації CLI-команд на основі введених параметрів. Це дозволить спростити процес конфігурації [redacted], зменшити ймовірність помилок, підвищити ефективність підготовки техніки до польоту та скоротити час на налаштування. Об’єктом проєктування в межах даної роботи є система автоматизованої генерації команд налаштування [redacted], які застосовуються у ГІС-системах для екологічного моніторингу лісів.

Предметом дослідження є методика створення desktop-додатку на мові Python, який забезпечує формування CLI-команд Betaflight на основі обраних параметрів у графічному інтерфейсі. Зокрема, це стосується налаштувань UART, приймача (RX), відеопередавача (VTX), камери, PID-регуляторів та інших ключових параметрів.

Метою роботи є створення desktop-застосунку для автоматизованої генерації конфігураційних CLI-команд, який спрощує процес налаштування [redacted] для задач екологічного моніторингу та інтегрується в робочі процеси ГІС-систем.

Для досягнення мети необхідно реалізувати такі завдання:

1. Провести аналіз параметрів, що налаштовуються в прошивці Betaflight (UART, RX, VTX, Camera тощо);
2. Створити графічний інтерфейс користувача для вибору конфігураційних компонентів [redacted];
3. Реалізувати логіку генерації CLI-команд на основі введених параметрів;
4. Забезпечити виведення готових скриптів у текстовий файл для подальшого імпорту в Betaflight Configurator;
5. Реалізувати функцію збереження шаблонів конфігурацій (пресетів) та їх повторного використання;

Таким чином, розробка інструменту автоматизації налаштування [redacted] значно спрощує застосування [redacted] у рамках ГІС-досліджень і робить їх використання більш доступним, швидким і надійним, в тому числі й у екологічному контексті.

1.2 Огляд існуючих рішень та підходів до налаштування дронів

Сфера [redacted] стрімко розвивається, і з кожним роком на ринку з’являються нові апаратні та програмні рішення, що дозволяють розширити можливості [redacted] у різних прикладних галузях, зокрема в екологічному моніторингу. Разом із розвитком апаратної частини особливу роль відіграє програмне забезпечення для налаштування [redacted] параметрів [redacted]. Одним із найпоширеніших та найпотужніших у цій сфері є програмне середовище Betaflight.

Betaflight – це відкрита (open-source) прошивка для польотних контролерів [redacted], орієнтована переважно на [redacted], які використовуються як у спортивних, так і у прикладних цілях.

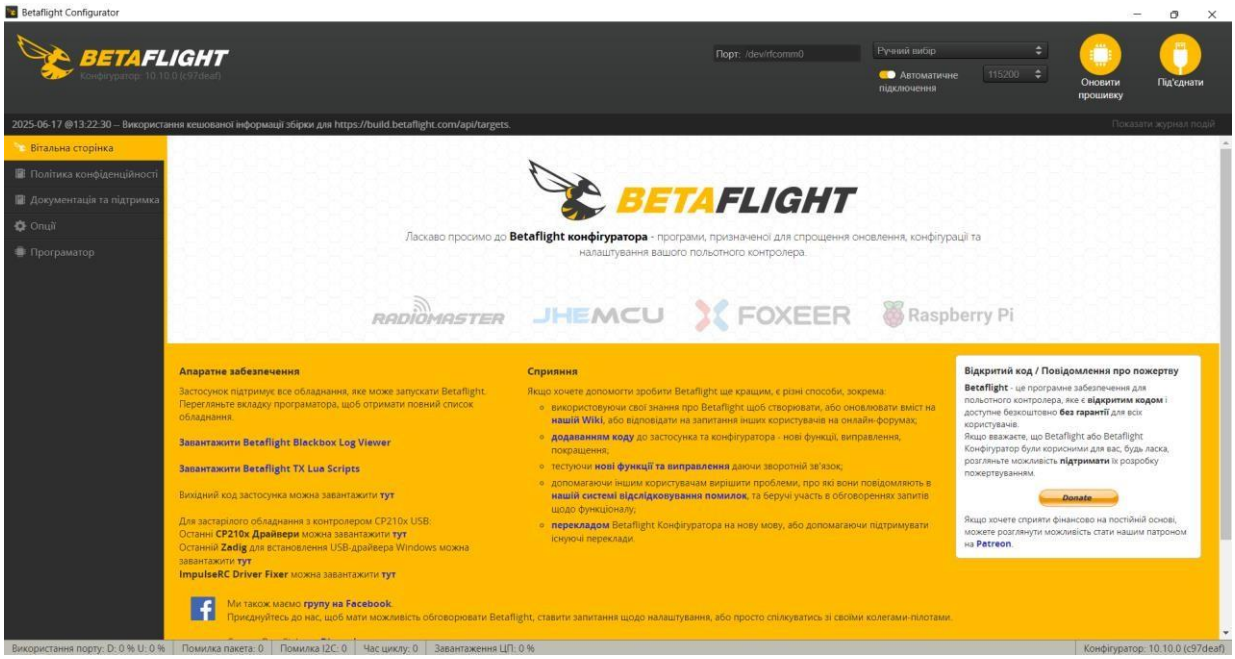


Рисунок1.1 – Вітальна сторінка програми Betaflight Configurator

Betaflight дозволяє налаштувати практично всі аспекти керування польотом [redacted], починаючи з вибору типу і прошивки польотного контролера, призначення UART-портів для підключення приймача, GPS, відеопередавача, SmartAudio або Tramp протоколу, конфігурації ESC (вибір протоколу, частоти PWM, режиму демпфування), прив'язки моторів і ресурсів до конкретних пінів контролера, налаштування PID-регуляторів і фільтрів для трьох осей стабілізації, а також визначення поведінки дрона у разі втрати сигналу (failsafe). Крім того, у Betaflight конфігурується робота акселерометра, гіроскопа, барометра та компаса, задаються режими польоту (наприклад: Angle, Horizon, Acro), прив'язуються канали керування до функцій (ARM, Flip Over After Crash, Beeper, GPS Rescue тощо), активуються осцилятори та фільтри для зменшення шуму, керується підсвіткою LED та звуковим сигналом, а також задаються параметри відеосигналу (VTX power, band, channel, pitmode). Також Betaflight дозволяє активувати або відключити режим чорної скриньки (Blackbox), зберігати конфігурації, оновлювати прошивку та контролювати стани [redacted] в реальному часі через OSD або телеметрію. Окрім цих основних налаштувань, Betaflight дозволяє конфігурувати й безліч дрібніших параметрів.

Важливою функцією Betaflight є також можливість аналізу польотних даних за допомогою Blackbox. Завдяки цьому інструменту [redacted] можуть переглядати лог-файли з даними сенсорів, PID-регуляторів, команд з передавача, руху моторів тощо. Це дозволяє не лише діагностувати проблеми з вібраціями чи нестабільною поведінкою [redacted], а й удосконалювати налаштування для досягнення кращої стабільності, зменшення лагів та підвищення загальної ефективності системи.

На додачу, Betaflight має активну спільноту користувачів та розробників, яка постійно оновлює прошивку, додає нові функції та вдосконалює існуючі алгоритми. Завдяки відкритому коду, кожен охочий може внести власні зміни або скористатися модифікаціями від інших учасників спільноти. Це робить Betaflight не лише гнучким інструментом для професійного налаштування [5-6; 8].

Betaflight пропонує два основних способи конфігурування:

1. Через графічний інтерфейс користувача – Betaflight Configurator (розширення Chrome або standalone-додаток).

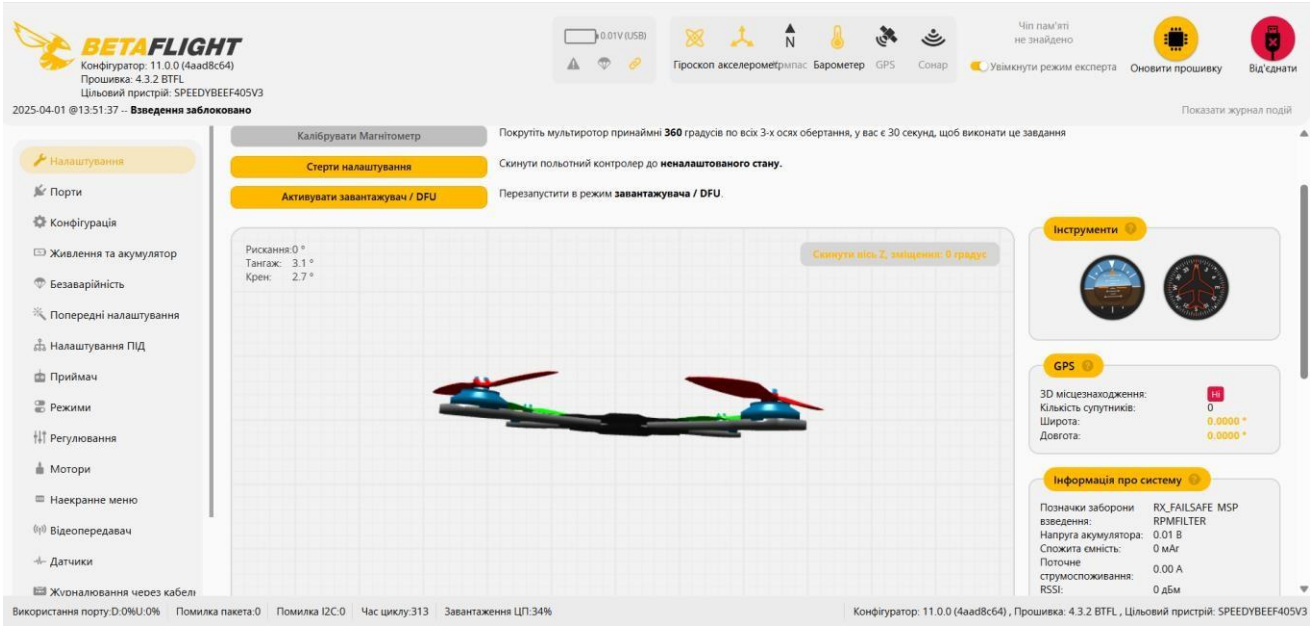


Рисунок 1.2 – Калібрування акселерометра

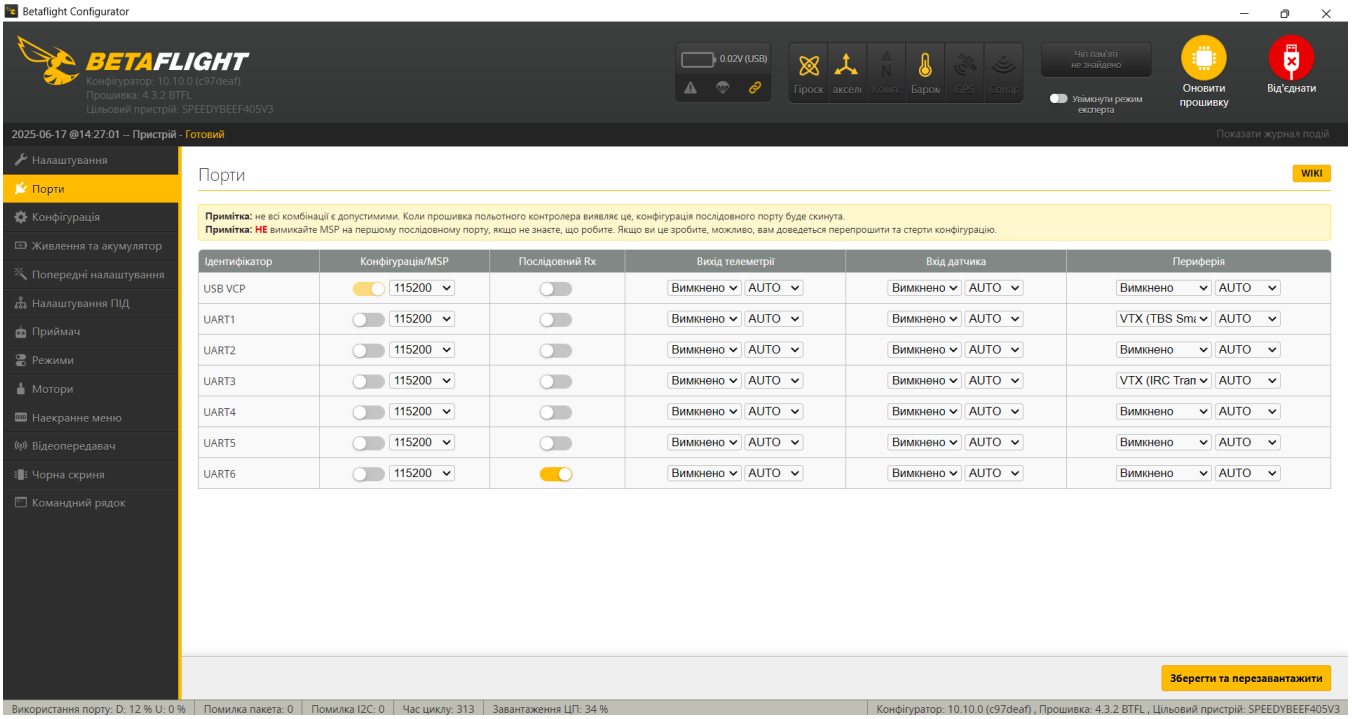


Рисунок 1.3 – Призначення портів периферій

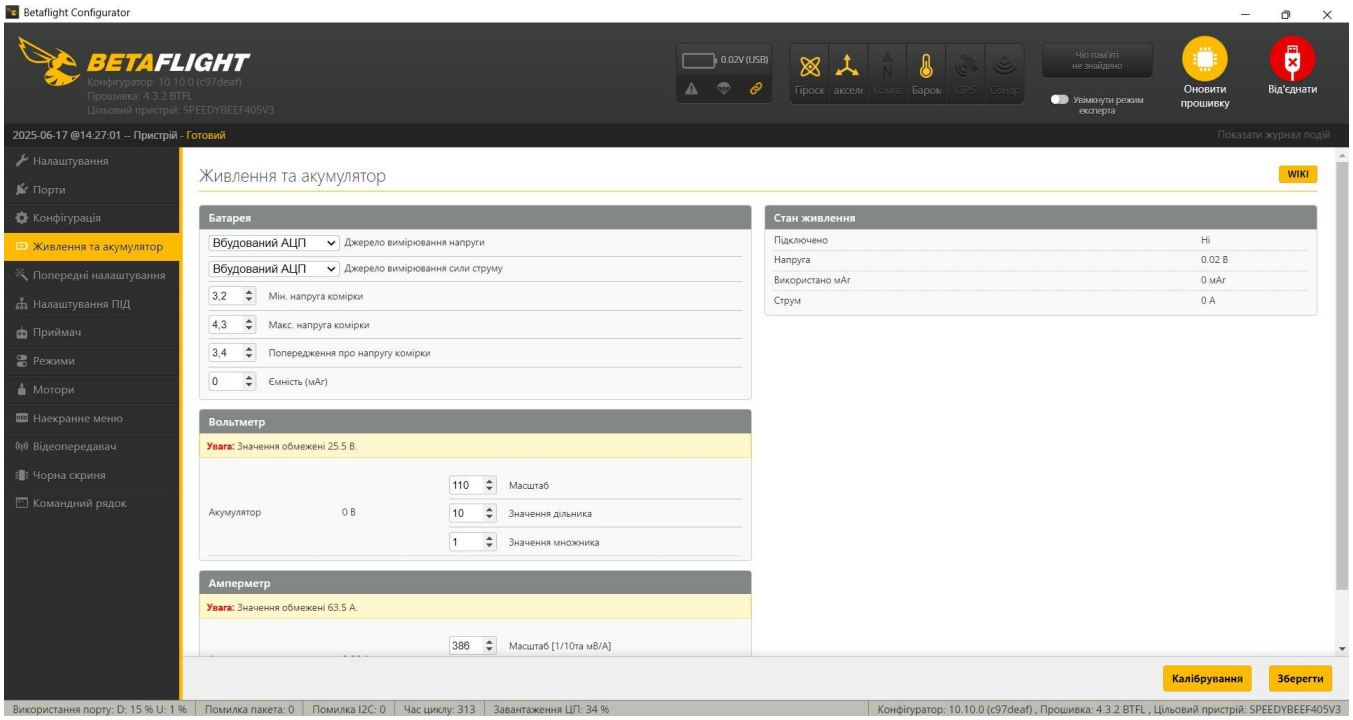


Рисунок 1.4 – Налаштування параметрів живлення

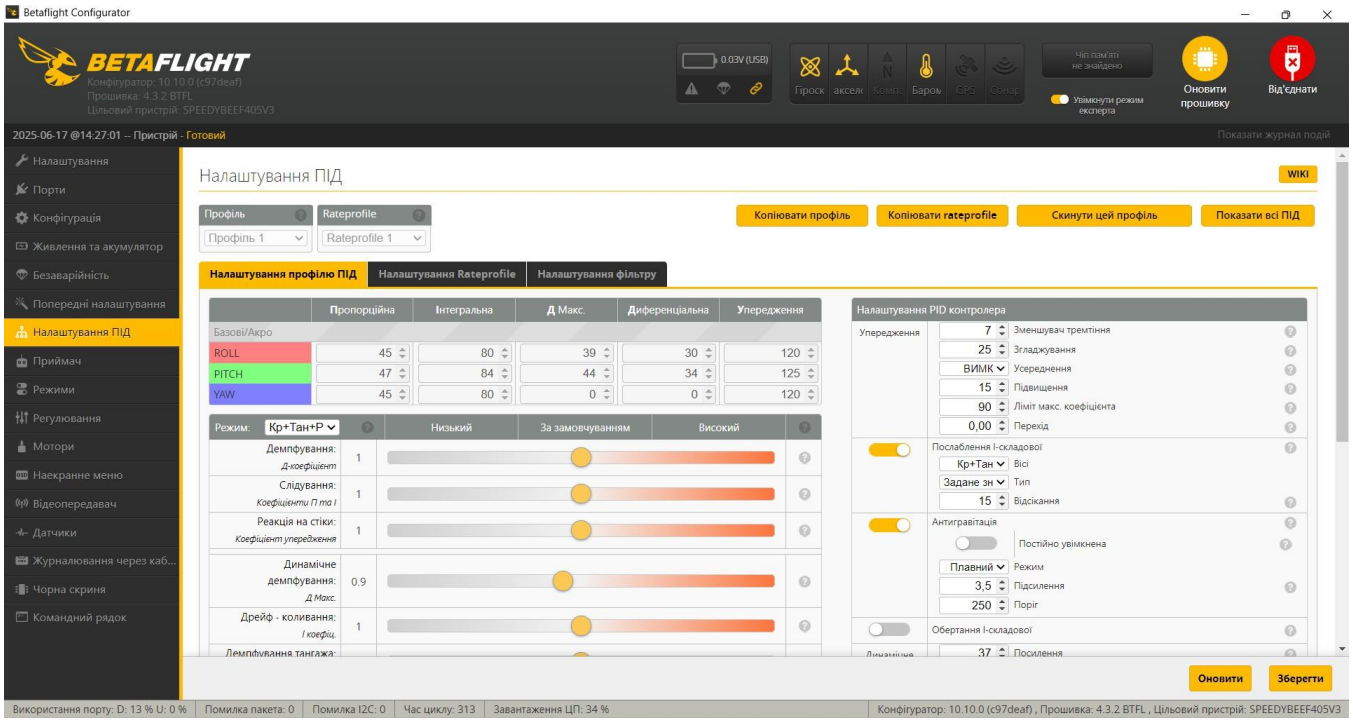


Рисунок 1.5 – Налаштування PID-ів і фільтрів

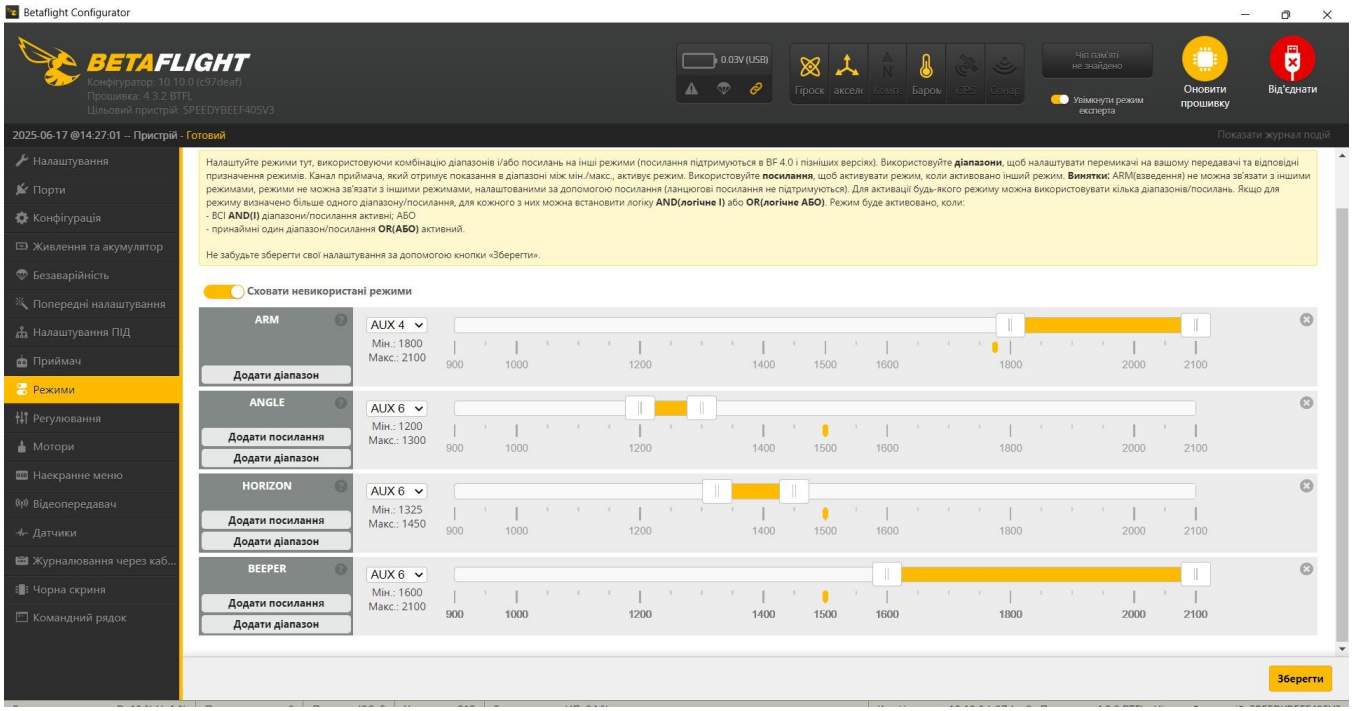


Рисунок 1.6 – Налаштування режимів керування

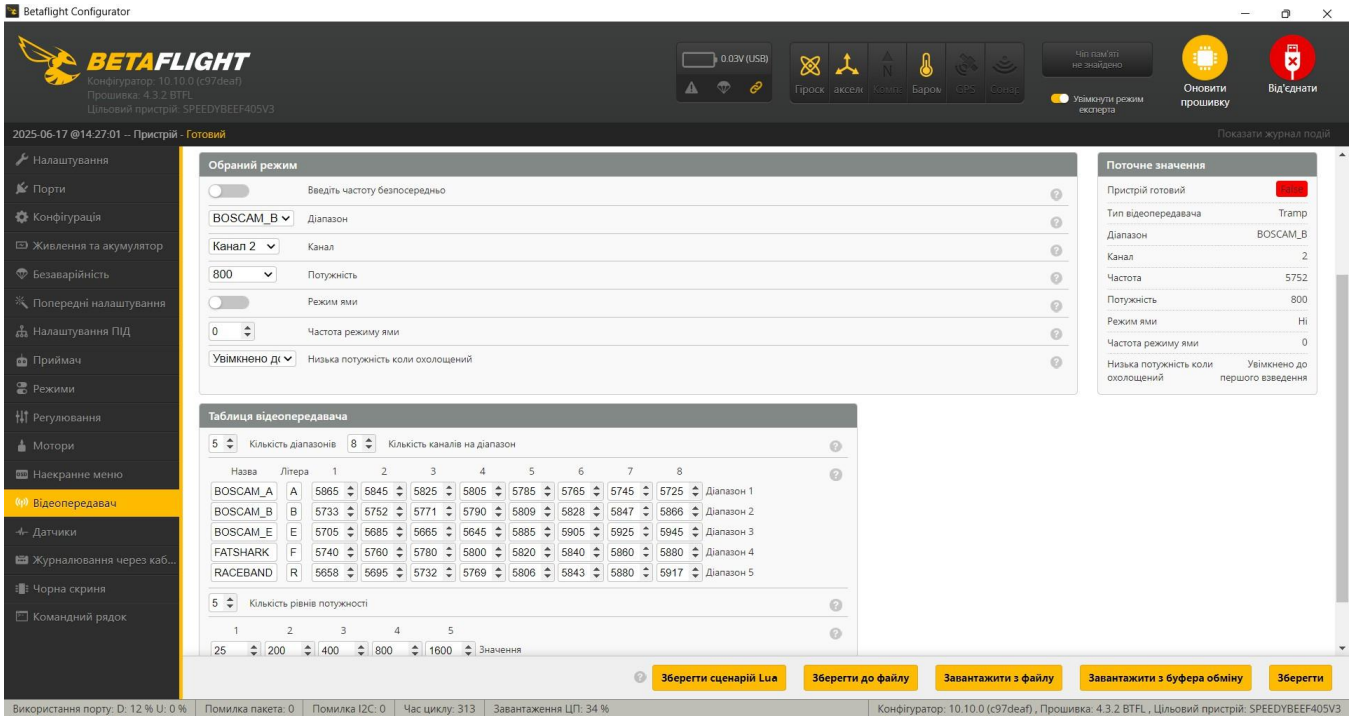


Рисунок 1.7 – Налаштування відеопередавача

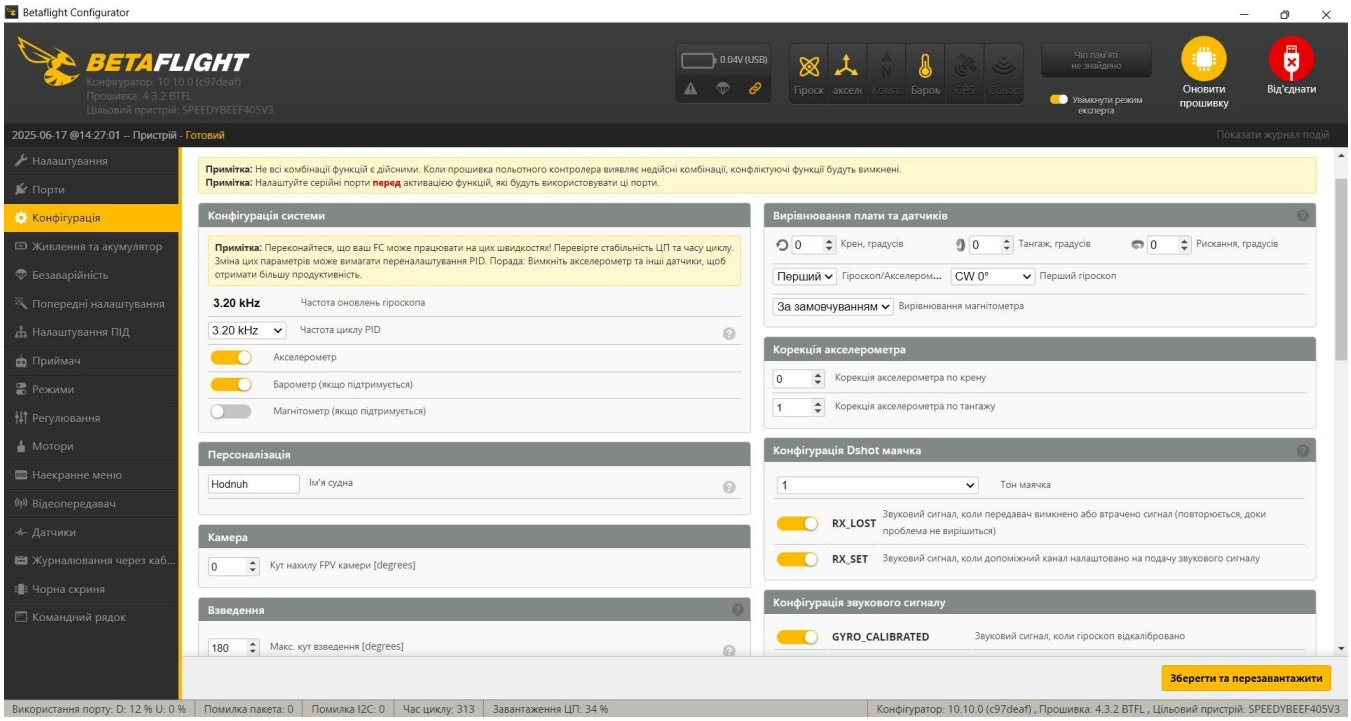


Рисунок 1.8 – Загальні коригування у вкладці «Конфігурація»

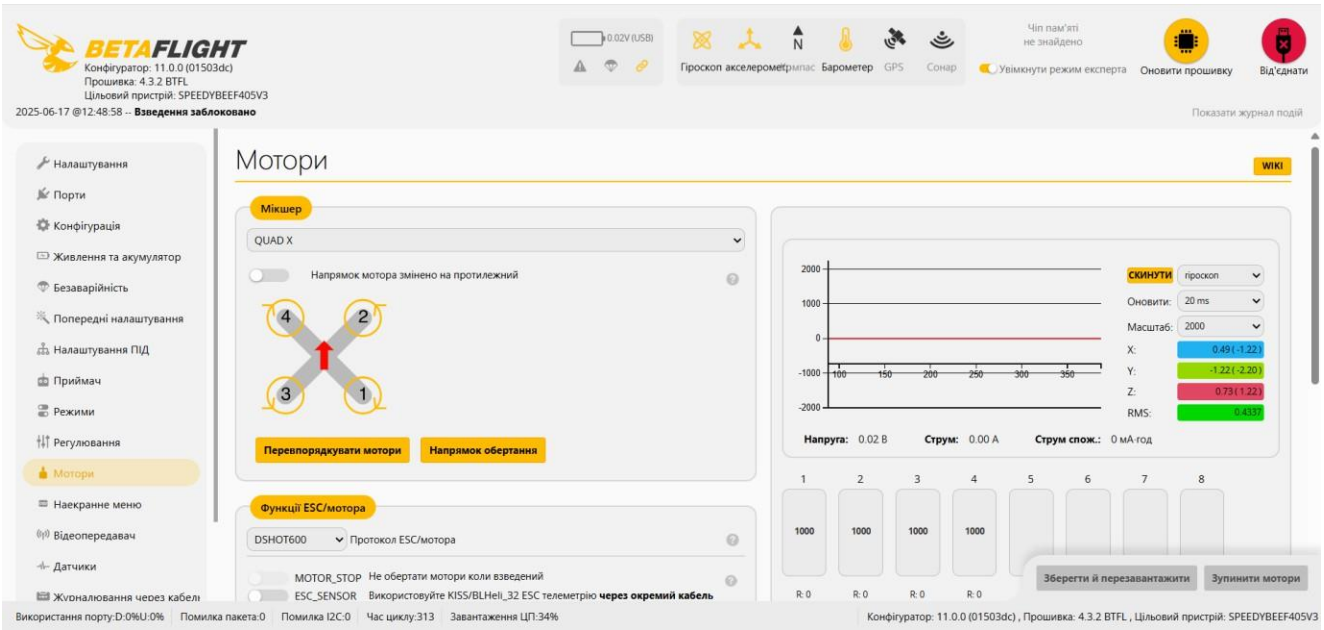


Рисунок 1.9 – Вкладка налаштування моторів

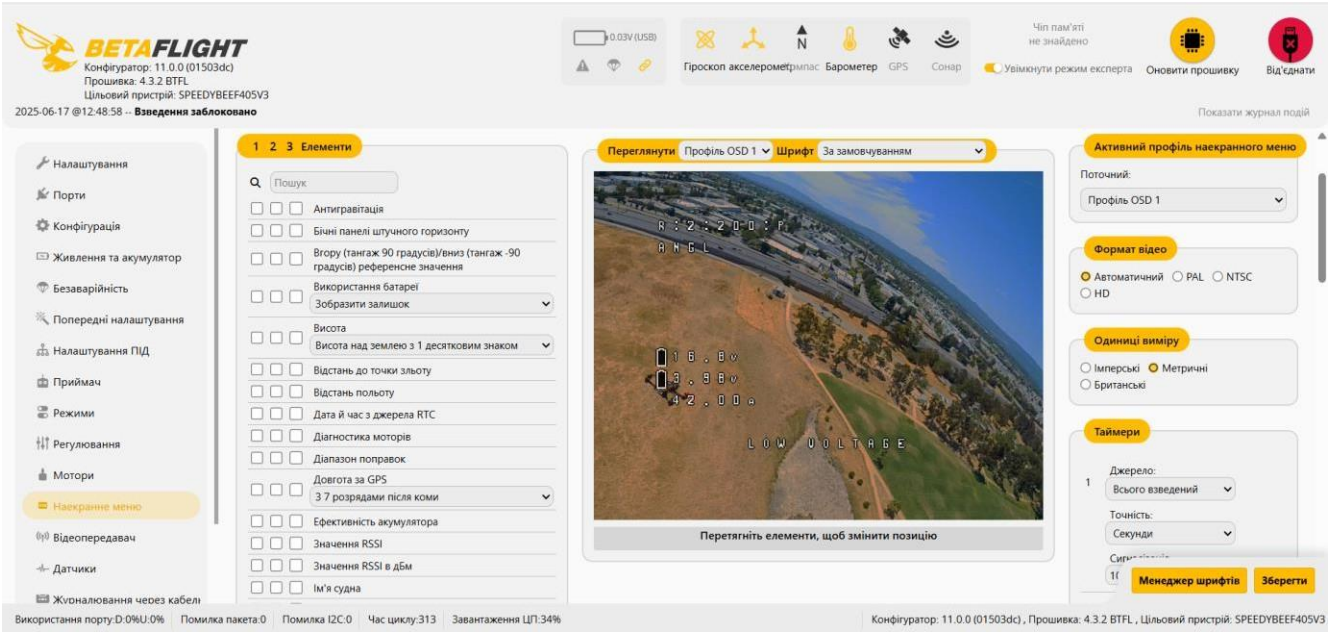


Рисунок 1.10 – Вкладка налаштування наекранного меню (OSD)

Після цих налаштувань потрібно також підкорегувати значення моторів і приймача сигналу управління в інших програмах, наприклад *BLHeliSuite32* і *ExpressLRS Configurator* відповідно. Після чого провести необхідні тести, дозібрати дрон, провести його обліт і відправляти в підрозділ.

2. Через текстовий інтерфейс CLI (Command Line Interface), де всі налаштування виконуються командами на кшталт:

set motor_pwm_protocol = DSHOT600

resource MOTOR 1 A03

serial 0 64 115200

save

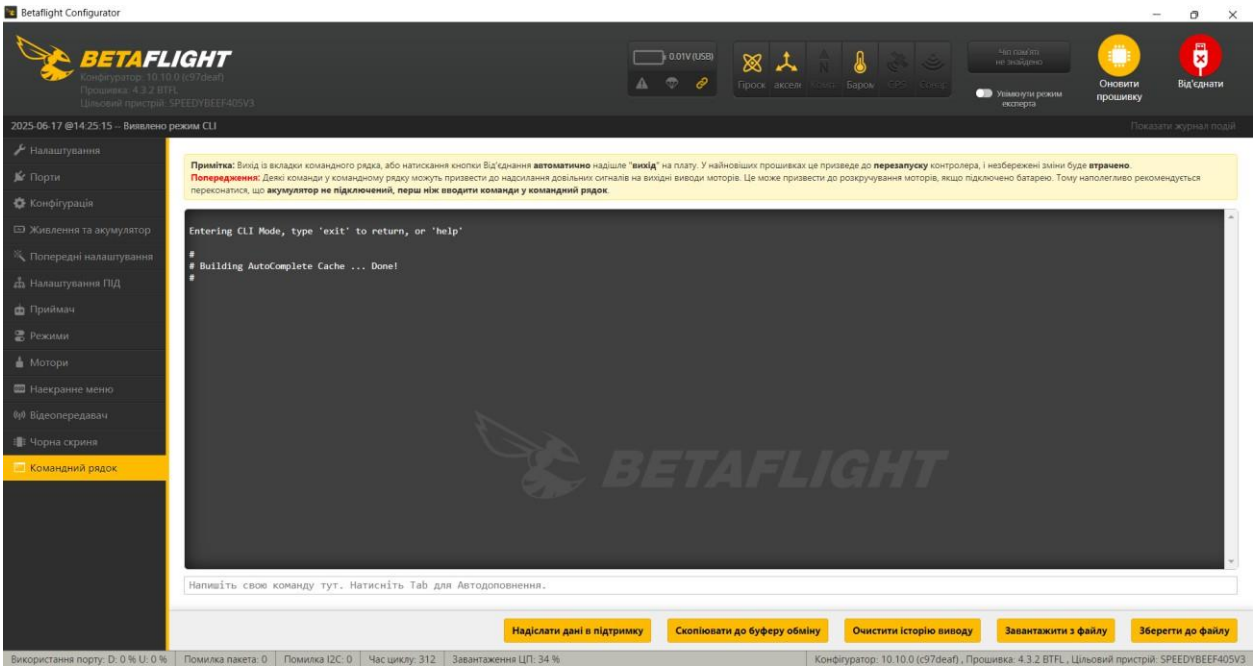


Рисунок 1.11 – Командний рядок в Betaflight

Конфігурація через CLI особливо корисна в тих випадках, коли необхідно швидко розгорнути або повторно налаштувати великий парк [redacted], оскільки вона дозволяє зберігати та повторно використовувати конфігураційні шаблони. Це є критично важливим у польових умовах, де час на налаштування та людські ресурси обмежені.

Незважаючи на наявність графічного конфігуратора, CLI залишається незамінним інструментом для швидкої зміни ключових параметрів або для масового налаштування [redacted]. Проте ручне введення CLI-команд може призводити до помилок, втрати часу, несумісностей або пропусків важливих параметрів.

У зв’язку з цим запропоноване рішення - desktop-додаток на Python із графічним інтерфейсом – покликане вирішити проблему автоматизації процесу налаштування дронів шляхом:

1. Введення користувачем параметрів польотного контролера через інтуїтивний GUI (рамка, приймач, VTX, UART, прошивка, опис тощо);
2. Формування відповідного набору CLI-команд для прошивки Betaflight;
3. Миттєвого перегляду результату у вікні програми;

4. Можливості копіювання або збереження згенерованих команд у файл;

5. Підтримки збереження пресетів, які можуть бути використані повторно при конфігурації ідентичних або схожих дронів.

Такий підхід дозволяє підвищити ефективність налаштування апаратної частини [redacted] в екологічних проєктах, що використовують ГІС. Завдяки модульності програма може бути розширена для обробки логів польоту, підключення сторонніх датчиків або створення стандартних наборів конфігурацій для типових місій. Крім того, відкритість програмної реалізації дозволяє адаптувати додаток під специфіку конкретної апаратури або задачі.

Загалом, інтеграція автоматизованого генератора CLI-команд у процес підготовки ГІС-[redacted] дозволяє підвищити надійність налаштування, зменшити залежність від людського фактору та забезпечити стандартизацію конфігурації безпілотних апаратів.

Таким чином, програма бере на себе формування правильного CLI-скрипта, знижуючи ймовірність помилок і спрощуючи роботу з Betaflight. Це особливо корисно в ситуаціях, коли конфігурується кілька однотипних [redacted] - наприклад, при польовому розгортанні дронів для зйомки, доставки чи моніторингу.

У перспективі така програма може бути інтегрована з іншими компонентами - наприклад, для попередньої підготовки дронів у рамках задач аерофотозйомки, картографування або екологічного моніторингу територій, які в подальшому аналізуються в ГІС.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Обґрунтування вибору засобів розробки

Розробка desktop-застосунку для генерації CLI-команд Betaflight вимагала використання гнучкого та доступного інструментарію. Основною вимогою до платформи стало забезпечення графічного інтерфейсу, кросплатформеності, простоти розгортання, а також широкої підтримки серед спільноти розробників [redacted].

У зв’язку з цим для реалізації застосунку було обрано мову програмування Python – популярну високорівневу мову з багатою екосистемою бібліотек. Як бібліотеку для створення графічного інтерфейсу було використано customtkinter – сучасну надбудову над стандартною бібліотекою tkinter, яка дозволяє створювати темні/світлі теми, адаптивні елементи керування та компоненти з покращеним візуальним виглядом.

На користь такого вибору також свідчить:

- швидкість розробки та простота налагодження;
- відсутність потреби в компіляції;
- можливість створення незалежного .exe-файлу через PyInstaller;
- легка інтеграція з обробкою файлів, JSON, буфером обміну та логічними модулями.

Крім того, Python дозволяє реалізовувати додаткові модулі – наприклад, для обробки логів польоту, збереження конфігурацій у файл, створення шаблонів і аналізу текстових виводів CLI. Усі ці можливості були реалізовані у межах застосунку з урахуванням принципів модульності, розширюваності та зручності використання [1-3].

2.2 Побудова дерева проблем і дерева цілей

Дерево проблем є важливим аналітичним інструментом, що дозволяє візуалізувати причинно-наслідкові зв’язки між головною проблемою та її складовими. У контексті розробки програмного забезпечення для автоматизованої конфігурації [redacted], дерево проблем охоплює ключові труднощі, пов’язані з ручним налаштуванням системи Betaflight, низьким рівнем автоматизації та складністю інтеграції в ГІС-систему.

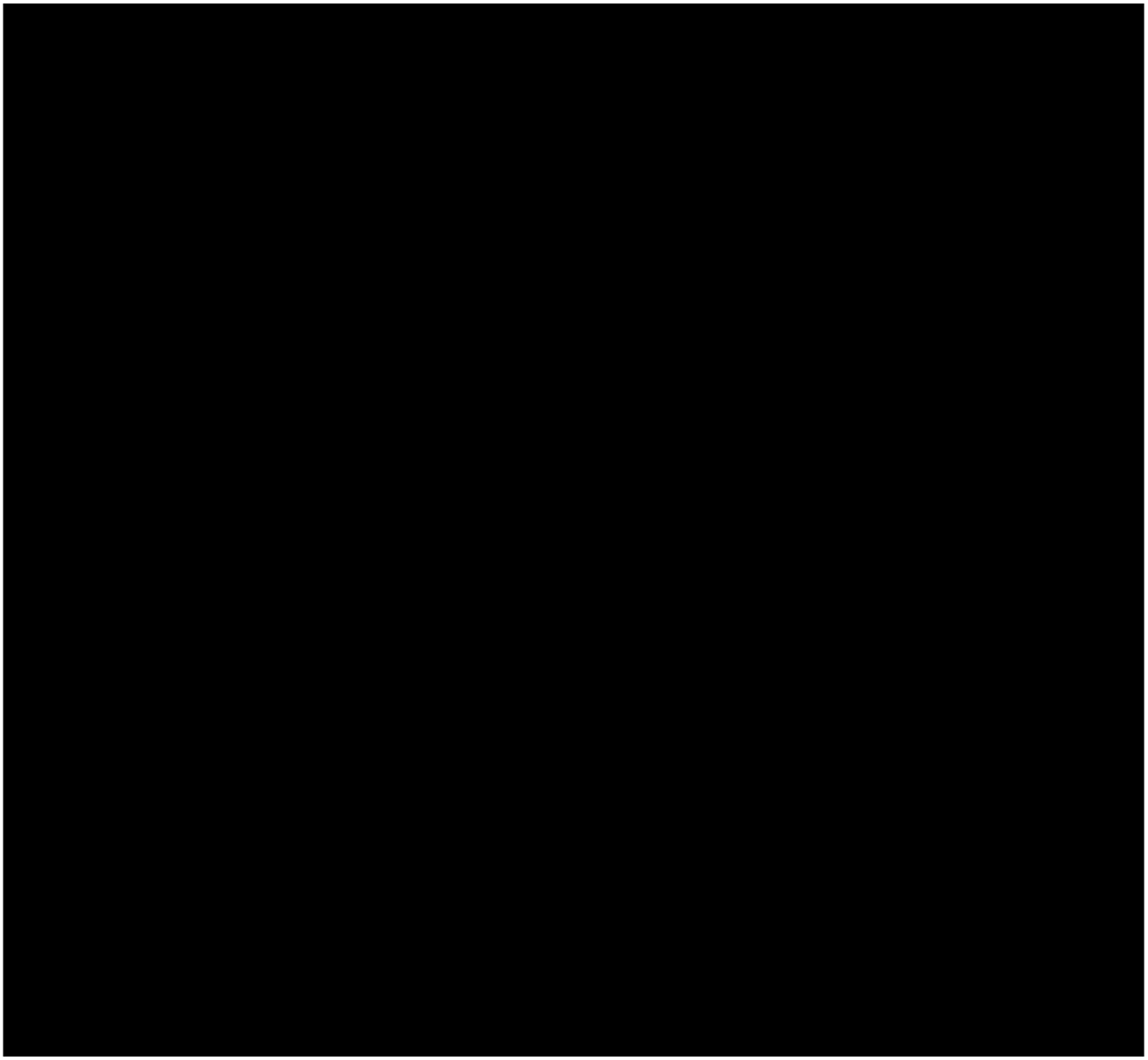


Рисунок 2.1 – Дерево проблем

Головна проблема: Складність та висока ймовірність помилок при ручному налаштуванні [redacted] через Betaflight CLI.

Причина 1: Необхідність глибоких технічних знань для налаштування.

Підпричина 1.1: Відсутність інтуїтивного інтерфейсу для складних команд.

Підпричина 1.2: Потреба розуміння синтаксису CLI команд та їхніх параметрів.

Підпричина 1.3: Складність бітових масок для конфігурації UART-портів.

Причина 2: Висока трудомісткість та час, затрачений на конфігурацію.

Підпричина 2.1: Повторювані ручні введення команд для типових налаштувань.

Підпричина 2.2: Складність відстеження та застосування власних "пресетів" налаштувань.

Підпричина 2.3: Потреба у ручному оновленні налаштувань при зміні компонентів або прошивки.

Причина 3: Ризик помилок та некоректних налаштувань.

Підпричина 3.1: Можливість введення несумісних або неправильних значень параметрів.

Підпричина 3.2: Труднощі з перевіркою коректності всієї конфігурації.

Підпричина 3.3: Проблеми з подальшим виявленням джерела помилок у складній конфігурації.

Дерево цілей є логічним відображенням дерева проблем, трансформуючи кожну виявлену проблему у конкретну ціль. Такий підхід дозволяє структурувати задачі, що мають бути реалізовані в межах проєкту, та сформувати основу для технічного завдання й архітектури програмного забезпечення.

Рисунок 2.2 – Дерево цілей

Головна ціль: Спрощення та автоматизація процесу налаштування [redacted] через Betaflight CLI.

Напрямок 1: Забезпечення інтуїтивного та зручного інтерфейсу для налаштування.

Ціль 1.1: Розробка графічного інтерфейсу користувача (GUI), який спрощує вибір та введення параметрів.

Ціль 1.2: Автоматична генерація коректних CLI-команд на основі вибору користувача.

Ціль 1.3: Перетворення складних бітових масок UART у зрозумілі опції вибору функцій.

Напрямок 2: Скорочення часу та зусиль, необхідних для конфігурації.

Ціль 2.1: Автоматизація генерації стандартних блоків команд (наприклад, для функцій, OSD).

Ціль 2.2: Впровадження системи збереження та завантаження користувацьких шаблонів конфігурацій.

Ціль 2.3: Надання можливості швидкого застосування налаштувань для різних компонентів (FC, VTX, RX) з відповідними значеннями.

Напрямок 3: Зниження ризику помилок та покращення надійності конфігурації.

Ціль 3.1: Вбудована валідація введених даних та обмеження вибору несумісних опцій.

Ціль 3.2: Генерація перевірених та працюючих пресетів для поширених конфігурацій.

Ціль 3.3: Надання інструментів для базового аналізу лог-файлів Blackbox для діагностики проблем.

2.3 Структура програми

Програма реалізована як desktop-застосунок із графічним інтерфейсом, розроблений на мові Python із використанням бібліотеки CustomTkinter. Вона побудована за принципом модульної архітектури, де інтерфейс користувача, логіка генерації команд, обробка шаблонів та аналіз логів реалізовані у вигляді взаємопов’язаних, але незалежних компонентів. Основна структура програми охоплює чотири логічні блоки:

1. Інтерфейс користувача (GUI), побудований у класі App – відповідає за створення вікон, елементів керування, вкладок і взаємодію з користувачем;

2. Логічний модуль generate_cli_commands() – реалізує формування команд на основі введених параметрів;

3. Модуль шаблонів – забезпечує збереження, редагування та завантаження конфігурацій у форматі JSON;

4. Модуль аналізу логів – реалізує базову текстову обробку польотних лог-файлів.

GUI реалізовано як табову структуру (CTkTabview), що включає чотири вкладки:

1. Генератор CLI – основна вкладка для налаштування параметрів дрона;

2. Шаблони – збереження, завантаження, перегляд готових конфігурацій;

3. Аналіз логів – завантаження Blackbox-файлів та базовий аналіз (помилки, попередження, рядки);

4. Довідка – текстова інструкція до застосунку.

Архітектурно всі введені користувачем параметри (наприклад, PID, UART, FC, VTX, RX тощо) зберігаються у словнику cli_tab_entries, з якого зчитуються під час генерації команд. Цей словник слугує проміжним буфером даних між інтерфейсом та логікою генерації.

Функція generate_cli_commands(config) формує текстовий скрипт CLI-команд, який відповідає синтаксису Betaflight. Вивід формується динамічно, з урахуванням наявних та валідних параметрів, і виводиться у текстове поле, а також може бути скопійований або збережений.

Шаблони конфігурацій зберігаються у локальному файлі templates.json, що дозволяє відновлювати параметри [redacted] навіть після перезапуску програми. Передбачено перевірку на дублікати, можливість редагування, завантаження в поля та копіювання CLI без повторного введення.

Програма також реалізує початкову обробку лог-файлів Betaflight (формати .log та .txt) із підрахунком ключових показників – кількість рядків,

помилок і попереджень. Це дозволяє швидко оцінити валідність логу та виявити потенційні проблеми.

Таким чином, архітектура програми дозволяє легко розширювати її функціонал – наприклад, додавати нові типи параметрів, вдосконалювати генерацію CLI або реалізовувати більш складну логіку аналізу логів. Застосунок розроблено з урахуванням потенційного розгортання як допоміжного інструменту до Betaflight Configurator.

2.4 Опис функціональності та особливості використання

Основна функціональна мета програми – автоматизація формування CLI-команд Betaflight для конфігурації [redacted]. Вона орієнтована на користувачів, які мають базове розуміння налаштування дронів, але бажають зменшити кількість повторюваних дій, уникнути помилок та пришвидшити процес конфігурації апаратної частини.

Користувач заповнюватиме параметри у графічному інтерфейсі: вибирає модель рами, [redacted] контролера (FC), прошивку, відеопередавач, камеру, частоту приймача, UART-порт тощо. Крім того, є можливість задати PID-налаштування та опис конфігурації. Після натискання кнопки «Генерувати CLI-команди» програма формує текстовий скрипт, який повністю сумісний із CLI-режимом Betaflight.

Згенеровані команди можна буде:

- скопіювати в буфер обміну та вставити у CLI-вікно Betaflight Configurator;
- зберегти у текстовий файл для подальшого імпорту;
- зберегти як шаблон конфігурації та використовувати у майбутньому для інших дронів.

Оскільки Betaflight Configurator підтримує вставлення CLI-команд напряму в термінал, користувачеві достатньо просто скопіювати текст із

програми та вставити його в CLI-вікно конфігуратора. Таким чином, програма працює в тандемі з офіційним інструментом налаштування – виступаючи як генератор, який спрощує підготовку конфігураційного скрипта.

У перспективі програма дозволятиме повноцінно формувати всі основні параметри дрона – включаючи повноцінну конфігурацію UART, розклад моторів, режими польоту, failsafe, OSD, LED-панель, telemetry smartports тощо. Очікується, що розширення логіки генерації дозволить створювати CLI-сценарії, які повністю готують [redacted] до [redacted] за лічені секунди.

Крім генерації CLI, застосунок реалізує систему шаблонів. Це дозволяє зберігати типові конфігурації (наприклад, для однакових моделей [redacted]), швидко їх відновлювати, адаптувати або тиражувати у проектах із кількома [redacted]. Шаблони можна редагувати, видаляти, копіювати або знову завантажувати у вкладку генерації CLI.

Вбудована вкладка аналізу логів дозволяє оцінити вміст Blackbox-файлу - визначити кількість рядків, кількість помилок і попереджень, що прискорює діагностику проблем у польоті.

Підсумовуючи, застосунок слугуватиме як зручне розширення до Betaflight Configurator, дозволяючи швидко формувати валідні CLI-команди, зменшуючи навантаження на користувача та стандартизуючи процес підготовки [redacted] до роботи.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Проектування архітектури системи

Система, що реалізовується, є десктопним додатком з графічним інтерфейсом користувача (GUI), призначеним для спрощення процесу генерації CLI-команд для [redacted] Betaflight, а також для виконання базового аналізу Blackbox логів. Програма буде реалізована мовою програмування Python, що забезпечить кросплатформність та ефективність розробки.

Архітектура програми будується таким чином, щоб чітко розмежувати логіку взаємодії з користувачем та основну функціональність. Це досягається шляхом поділу системи на два ключові модулі:

- main.py: Цей модуль відповідатиме за побудову та керування усіма елементами графічного інтерфейсу користувача. Він використовуватиме бібліотеку CustomTkinter для створення сучасного та інтерактивного вікна програми, вкладок, полів введення, випадаючих списків, чекбоксів та кнопок. Його основна функція — збір вхідних даних від користувача та відображення отриманих результатів.
- logic.py: Цей модуль міститиме всю основну бізнес-логіку програми. Сюди входитиме функціональність для генерації CLI-команд на основі отриманих від main.py даних, а також логіка для базового аналізу Blackbox логів. logic.py буде ізольований від деталей інтерфейсу, що забезпечить легкість тестування та подальшої підтримки коду.

Взаємодія між цими модулями відбуватиметься за принципом "розділення відповідальності". main.py слугуватиме "фронт-ендом" програми, збираючи дані та викликаючи відповідні функції з logic.py для виконання обчислень чи обробки. Після того, як logic.py виконає свою роботу (наприклад, згенерує CLI-команди або проаналізує лог), він повертатиме результат назад у main.py, який, у свою чергу, відображатиме його користувачу через GUI. Такий підхід дозволить

легко розвивати та модифікувати кожен аспект програми незалежно, зберігаючи при цьому її цілісність.

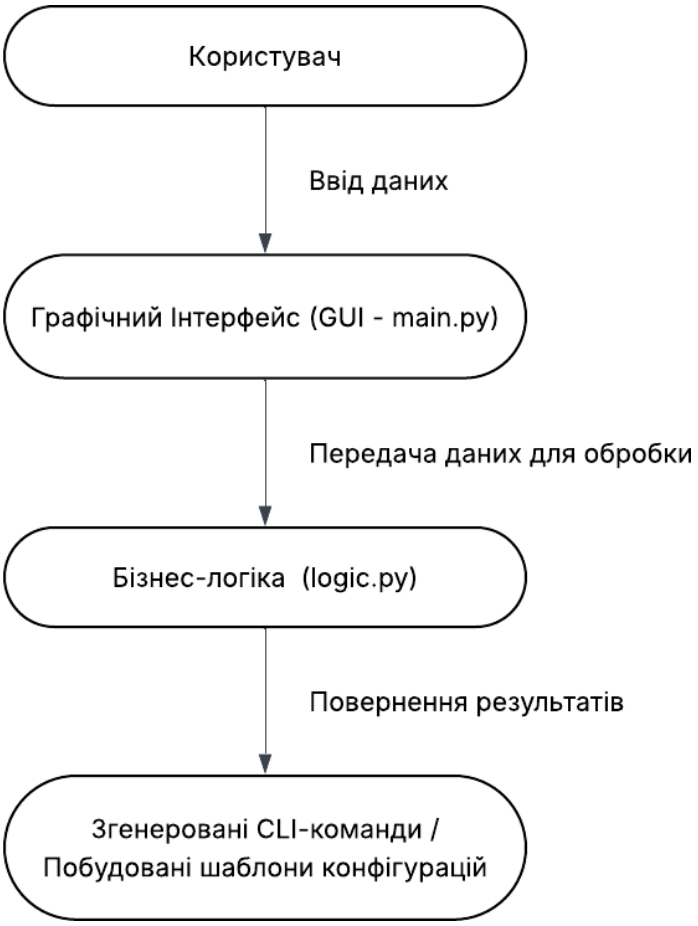


Рисунок 3.1 – Спрощена структурна схема взаємодії модулів ПЗ

3.2 Проектування інтерфейсу користувача

Проектування інтерфейсу користувача (UI) та досвіду взаємодії (UX) для розроблюваної програми буде зосереджено на кількох ключових принципах, що забезпечать її ефективність та привабливість для цільової аудиторії:

- Зручність використання: Головною метою буде створення програми, яка буде максимально простою та зрозумілою для кожного користувача, незалежно від його рівня технічних знань. Всі елементи керування будуть

розташовані логічно, а послідовність дій для генерації CLI-команд або аналізу логів буде інтуїтивно зрозумілою.

- Інтуїтивно зрозумілий дизайн: Інтерфейс буде розроблений таким чином, щоб користувач міг без додаткових інструкцій розуміти призначення кожного елемента та його вплив на функціональність програми. Використання чітких текстових міток, піктограм та візуальної ієрархії сприятиме швидкому освоєнню програми.
- Мінімалістичний та функціональний вигляд: Дизайн програми уникатиме зайвих графічних елементів, які можуть відволікати користувача або перевантажувати інтерфейс. Основний акцент буде зроблено на функціональності, забезпечуючи при цьому естетичний, чистий та сучасний вигляд. Це дозволить користувачеві зосередитися на виконанні своїх завдань без візуального шуму.
- Фокус на ефективній генерації команд: Усі дизайнерські рішення будуть спрямовані на оптимізацію процесу генерації CLI-команд. Це означає, що поля введення, випадаючі списки та чекбокси будуть зручно розташовані для швидкого доступу, а процес генерації буде максимально швидким та безперебійним, мінімізуючи кількість кроків, необхідних для отримання бажаного результату.

Для реалізації графічного інтерфейсу чудово підійде бібліотека CustomTkinter. Цей вибір обумовлений її значними перевагами у створенні сучасного та адаптивного інтерфейсу, що відповідає вищезазначеним принципам. CustomTkinter побудована на базі стандартної бібліотеки Tkinter, але розширює її можливості, надаючи доступ до більш сучасних візуальних компонентів та тем. Серед ключових переваг CustomTkinter можна виділити:

Сучасний вигляд: Бібліотека дозволяє створювати елементи інтерфейсу з сучасним "плоским" дизайном та плавними анімаціями, що значно покращує естетичне сприйняття програми порівняно зі стандартним Tkinter.

Темна та світла теми: Вбудована підтримка зміни тем дозволяє користувачам обирати між світлим та темним інтерфейсом, що є важливим елементом комфорту та доступності.

Адаптивність: Елементи CustomTkinter є гнучкими та легко адаптуються до зміни розміру вікна програми, забезпечуючи коректне відображення на різних роздільних здатностях екранів. Це допоможе забезпечити зручність використання програми як на великих моніторах, так і на менших екранах ноутбуків.

Простота використання: Незважаючи на розширений функціонал, CustomTkinter зберігає простоту синтаксису Tkinter, що спрощує процес розробки та підтримки коду інтерфейсу.

Інтеграція з Python: Як Python-бібліотека, вона легко інтегрується з основною логікою програми, забезпечуючи ефективну взаємодію між GUI та бізнес-логікою[9].

Реалізований графічний інтерефейс за допомогою бібліотеки CustomTkinter наведено на рисунку 3.2.

На зображенні інтерфейсу програми "Генератор CLI для Betaflight" застосовані наступні елементи:

Вкладки (Tabview): "Генератор CLI", "Шаблони", "Аналіз логів", "Довідка". Поля для введення тексту (Entry/Text fields):"PID P","PID I","PID D","Опис", поле "Назва дрона". Випадаючі списки (ComboBox):"Оберіть FC", "Оберіть прошивку", "Оберіть VTX", "ExpressLRS (CRSF)" (або обраний RX Protocol). Шість випадаючих списків для "UART1" - "UART6" з опціями функцій (наприклад, "None", "Serial RX", "VTX (SmartAudio/TrampHV)", "VTX (MSP)", "GPS", "Blackbox", "ESC Sensor", "MSP", "Camera Control").

А також: написи (Labels), чекбокси (Checkboxes), текстове поле (Textbox): Область для виводу згенерованих CLI-команд і скролбар (Scrollbar): Вертикальний скролбар праворуч від полів введення та випадаючих списків, що дозволяє прокручувати вміст вкладки.

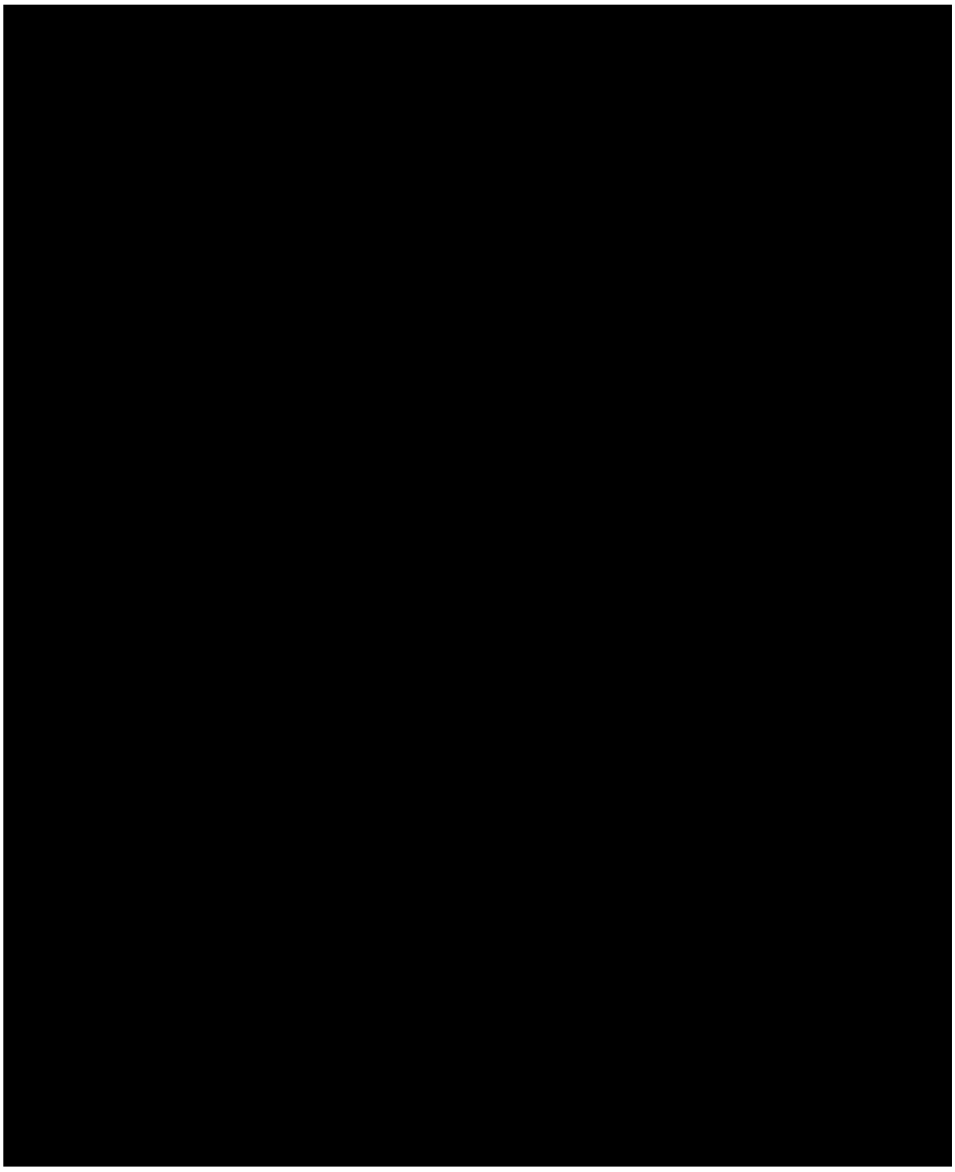


Рисунок 3.2 – Головне вікно програми

Та перед початком безпосередньої програмної реалізації було здійснено етап прототипування, що дозволило попередньо візуалізувати структуру та розташування ключових елементів інтерфейсу. Хоча повноцінні деталізовані прототипи не розроблялися, початкові ескізи та начерки, подібні до , були використані для формування загальної концепції головної вкладки. Це дозволило уточнити розміщення полів введення для параметрів [redacted], випадаючих списків для вибору компонентів та зони для виведення згенерованих CLI-команд, ще до написання основного коду інтерфейсу. Такий підхід сприяв ранньому виявленню потенційних проблем з юзабіліті та оптимізації візуального простору.

Рисунок 3.3 – Прототип головної вкладки

Рисунок 3.4 – Головна вкладка на етапі розробки

3.3 Опис основних вікон та елементів інтерфейсу

Інтерфейс програми буде організовано за допомогою вкладок (Tabview), що дозволить зручно перемикатися між різними функціональними блоками: "Генератор CLI", "Шаблони", "Аналіз логів" та "Довідка".

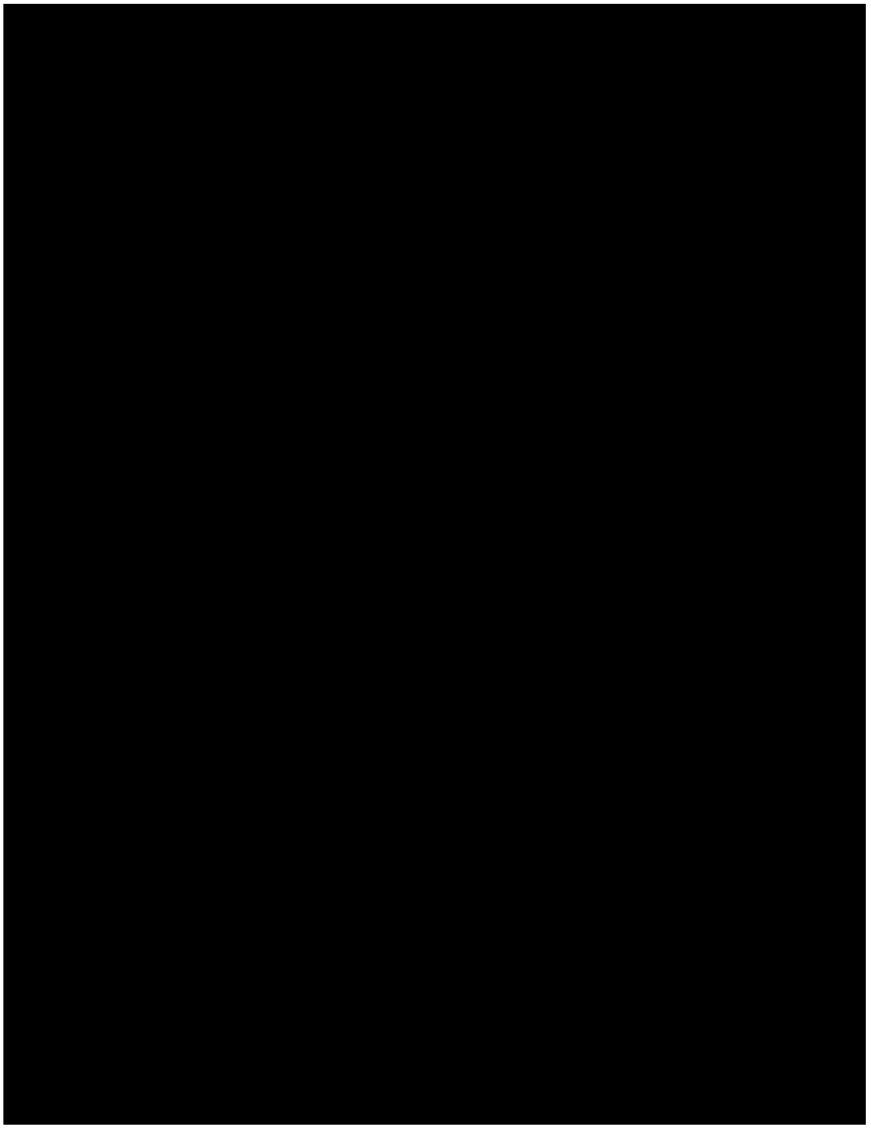


Рисунок 3.5 – Головна вкладка "Генератор CLI"

Ця вкладка є центральним елементом програми, де користувач зможе вводити всі необхідні дані для генерації CLI-команд. Її інтерфейс буде логічно поділений на кілька зон для зручності заповнення.

Блоки введення даних:

1. [redacted]

- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

Рисунок 3.6 – Основні блоки введення даних

Налаштування [redacted]: Цей підрозділ відіграватиме важливу роль у конфігурації периферійних пристроїв. Користувачеві буде надано можливість призначати функції для кожного з [redacted]. Кожен [redacted] матиме свій випадаючий список, з якого можна буде обрати наступні функції:

Рисунок 3.7 – Налаштування UART-портів

- [redacted]

[redacted]

Кожна з цих опцій має свою унікальну бітову маску, яка буде включена у команду serial в згенерованому CLI, таким чином чітко визначаючи функціонал кожного фізичного порту для Betaflight.

Додаткові налаштування: Цей блок міститить чекбокси для активації або деактивації певних функцій Betaflight.

Рисунок 3.8 – Чекбокси додаткових налаштувань

- [redacted]
- [redacted]
- [redacted]

Вибір цих чекбоксів безпосередньо впливатиме на генерацію відповідних команд set у CLI.

Рисунок 3.9 – Елементи керування

- [redacted]
- [redacted]

- [redacted]

- Кнопка "Зберегти в файл": Забезпечуватиме можливість зберегти згенеровані команди у текстовий файл на локальному диску користувача.
- Кнопка "В шаблони": Надаватиме функціональність для збереження поточної конфігурації (всіх заповнених полів, виборів та станів чекбоксів) як шаблон для подальшого використання.

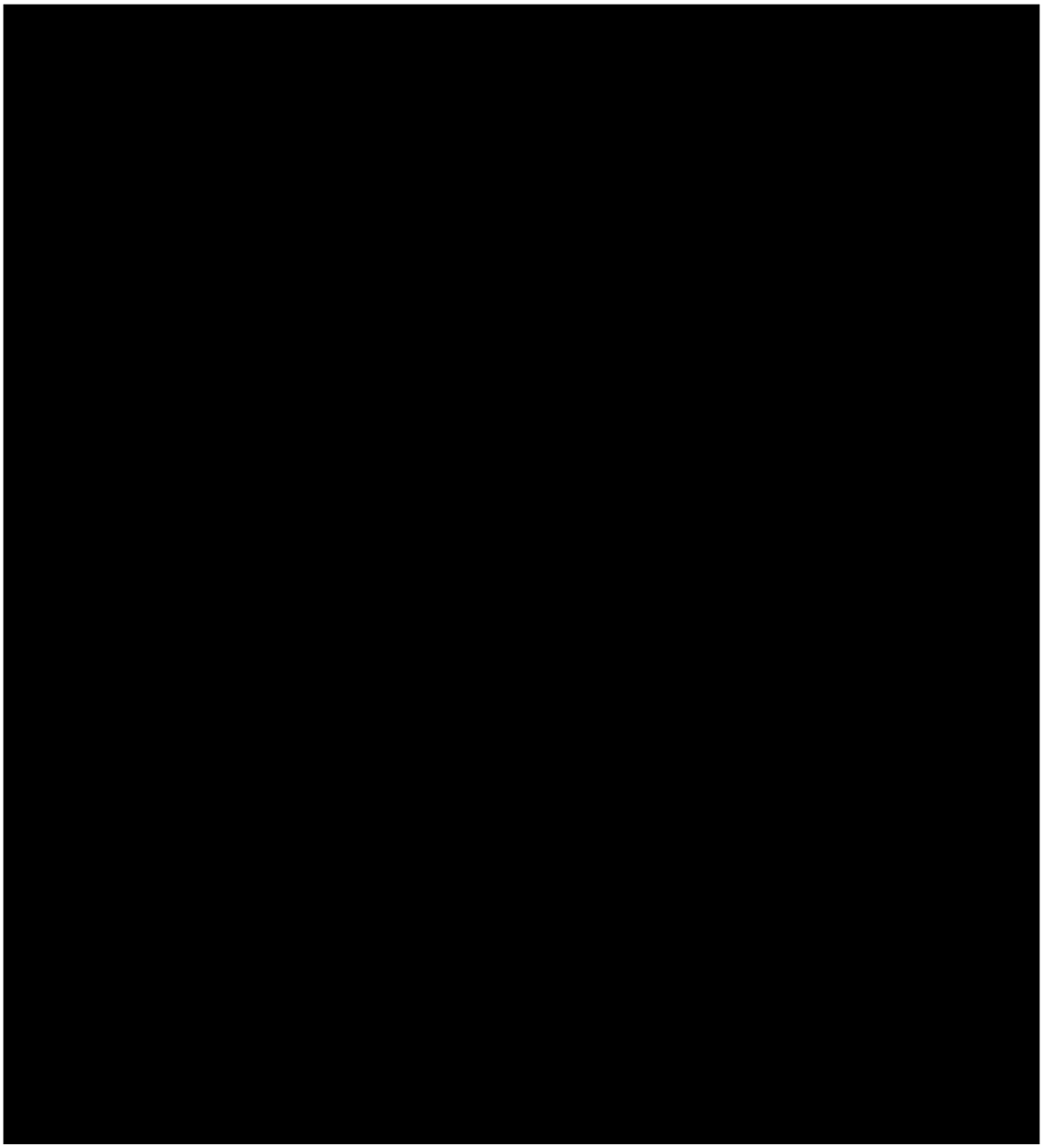


Рисунок 3.10 – Вкладка "Шаблони"

Ця вкладка (рис.3.10) призначена для ефективного керування збереженими користувачем конфігураціями.

Призначення: Вкладка слугуватиме сховищем для улюблених або часто використовуваних конфігурацій [redacted]. Це дозволить користувачеві швидко завантажувати, копіювати або видаляти попередньо збережені налаштування, уникаючи повторного ручного введення даних.

Відображення шаблонів: Кожен збережений шаблон буде відображатися у вигляді окремого блоку, який міститиме ключову інформацію: назву шаблону, короткий опис (якщо введено), модель [redacted], тип приймача [redacted] та тип рами.

Функціонал кнопок:

- "Завантажити": При натисканні цієї кнопки, всі поля на вкладці "Генератор CLI" будуть автоматично заповнені даними відповідного шаблону, що дозволить швидко повернутися до роботи з конкретною конфігурацією.
- "Скопіювати CLI": Дасть можливість скопіювати згенеровані CLI-команди для цього конкретного шаблону без необхідності переходу на вкладку "Генератор CLI".
- "Видалити": Забезпечить видалення вибраного шаблону зі списку збережених.

Зберігання даних: Усі дані шаблонів будуть зберігатися у файлі templates.json, що забезпечить їхню стійкість між сесіями використання програми.

Вкладка "Аналіз логів"

Ця вкладка буде надавати базовий функціонал для роботи з логами польотів.

Призначення: Вкладка призначена для спрощеного аналізу файлів Blackbox логів, дозволяючи користувачам отримати швидке резюме або певну інформацію з лог-файлів без використання складних зовнішніх інструментів.

Елементи:

Кнопка "Завантажити Blackbox лог": Використовуватиметься для вибору та завантаження .log або .txt файлу Blackbox з локального диска.

Текстове поле для виводу результатів аналізу: Після завантаження та обробки логу, в цій області буде відображатися стисла інформація або ключові дані з лог-файлу, що допоможе у швидкій діагностиці польотних даних.

Рисунок 3.11 – Вкладка "Довідка"

Ця вкладка слугуватиме вбудованим джерелом інформації для користувачів.

Призначення: вкладка міститить вичерпні інструкції з використання програми, її основних функцій та можливостей. Це дозволить користувачам швидко знайти відповіді на питання щодо роботи з "Генератором CLI для Betaflight", зменшуючи потребу у зовнішній документації.

3.4 Деталізація програмної реалізації

Проект має чітку та логічну структуру, яка забезпечує легкість розробки, тестування та подальшої підтримки. Основні файли проекту та їхні ролі наступні:

main.py: Цей файл є основним виконавчим модулем програми. Він відповідає за ініціалізацію головного вікна додатку, побудову всіх елементів графічного інтерфейсу користувача (GUI) за допомогою бібліотеки CustomTkinter та керування взаємодією користувача з цими елементами. Усі події, що виникають в GUI (натискання кнопок, вибір у списках, введення тексту), обробляються саме тут.

logic.py: Цей модуль містить всю основну бізнес-логіку програми. Він відповідає за генерацію CLI-команд на основі введених користувачем параметрів, а також за виконання базового аналізу Blackbox логів. logic.py є незалежним від GUI, що дозволяє тестувати його функціональність окремо.

templates.json: Цей файл використовується для зберігання користувацьких шаблонів конфігурацій. Коли користувач зберігає поточні налаштування як шаблон, дані серіалізуються у формат JSON та записуються до цього файлу. При завантаженні шаблонів, дані зчитуються з templates.json та десеріалізуються назад у Python-об'єкти.

icon.ico: Цей файл містить іконку програми, яка відображається у заголовку вікна та на панелі завдань операційної системи, забезпечуючи впізнаваність додатка.

Рисунок 3.12 – Структура проекту

Опис основних класів та функцій main.py

Основний функціонал графічного інтерфейсу реалізовано в класі App у файлі main.py.

Клас App: Є центральним класом додатка, який успадковується від customtkinter.CTk та відповідає за ініціалізацію всього графічного інтерфейсу. Він керує життєвим циклом програми, обробляє події та координує взаємодію між компонентами GUI та логікою програми.

Рисунок 3.13 – Ініціалізація класу App й конструктор налаштування вікна

Методи побудови інтерфейсу:

- __init__(self): Конструктор класу App, який виконує початкові налаштування вікна програми (розмір, заголовок, іконка), ініціалізує вкладки інтерфейсу (CTkTabview) та викликає методи для побудови вмісту кожної

вкладки. Також завантажуються існуючі шаблони з файлу templates.json під час запуску.

- [redacted]

Методи для керування CLI-виводом та шаблонами:

- [redacted]

[redacted]

Методи для збереження/завантаження даних:

[redacted]

Опис logic.py.

Модуль logic.py містить дві основні функції, що реалізують бізнес-логіку програми.

Рисунок 3.14 – Функція “generate_cli_commands” й таблиця одного із підтримуваних програмою відеопередавачів

generate_cli_commands(config, uart_configs):

Вхідні параметри:

- [redacted]

- [redacted], де значення є бітовими масками, що відповідають вибраним функціям[4].

- Логіка формування CLI-команд: Функція послідовно формує CLI-команди для Betaflight, спираючись на отримані вхідні дані. Це включає:

1. Генерацію команд name для ідентифікації конфігурації.
2. Генерацію команд set для встановлення значень PID-контролера.

- Генерацію команд serial X бітова_маска, де X - номер UART-порту, а бітова_маска - числове значення, що представляє комбінацію функцій, призначених цьому порту (наприклад, serial 1 1024 для VTX (SmartAudio/TrampHV)). Важливо, що кожна функція UART має свою унікальну бітову маску, яка коректно обробляється.
- [redacted]
- [redacted]
- Додавання фінальної команди save, яка забезпечує застосування всіх налаштувань у Betaflight.

Форматування виведеного тексту CLI: Згенеровані команди відформатовані у зручний для читання вигляд, з коментарями та розбивкою на логічні блоки, що спрощує їхнє використання.

analyze_blackbox_log(content):

Призначення: Ця функція виконує базовий аналіз текстового вмісту файлів Blackbox логів. Її метою є надання користувачеві швидкої зведеної інформації про [redacted] без необхідності відкривати лог у повноцінному Blackbox Explorer.

Деталізація: На поточному етапі функція надає загальну інформацію або заглушку для подальшої розробки більш глибокого аналізу. У майбутньому її можна розширити для виявлення аномалій, пікових значень, або надання рекомендацій на основі даних логів.

3.5 Використані інструменти та технології

Для реалізації десктопного додатку "Генератор CLI для Betaflight" було використано комплекс сучасних інструментів та технологій, що забезпечать функціональність, стабільність та зручність розробки.

Мова програмування: Основною мовою програмування для розробки всієї системи було обрано Python. Вибір Python обґрунтовується його високою читабельністю, широким спектром доступних бібліотек, швидкістю розробки та кросплатформністю, що дозволить програмі функціонувати на різних операційних системах [7,10].

Основні бібліотеки та фреймворки:

- [redacted]
- [redacted]
- [redacted]
- json: Вбудований у Python модуль, який буде використовуватись для серіалізації (запису) та десеріалізації (читання) даних. Зокрема, він буде застосований для збереження та завантаження шаблонів конфігурацій користувача у файлі templates.json, забезпечуючи стійкість даних між сесіями використання програми.

Середовище розробки (IDE): Для розробки програми буде використано інтегроване середовище розробки PyCharm. PyCharm надає широкий спектр інструментів, що спрощують процес кодування, налагодження та тестування Python-додатків, включаючи інтелектуальне автодоповнення коду, підказки, рефакторинг та інтеграцію з системами контролю версій.

Рисунок 3.15 – Інтерфейс середовища розробки PyCharm”

3.6 Опис роботи програми

Цей розділ описує типовий сценарій використання програми "Генератор CLI для Betaflight", демонструючи покрокову взаємодію користувача з її функціоналом.

1. Запуск програми: Користувач запускає програму, виконуючи виконавчий файл main.py. Після запуску відкривається головне вікно додатка з активною вкладкою "Генератор CLI".

2. Генерація CLI-команд:

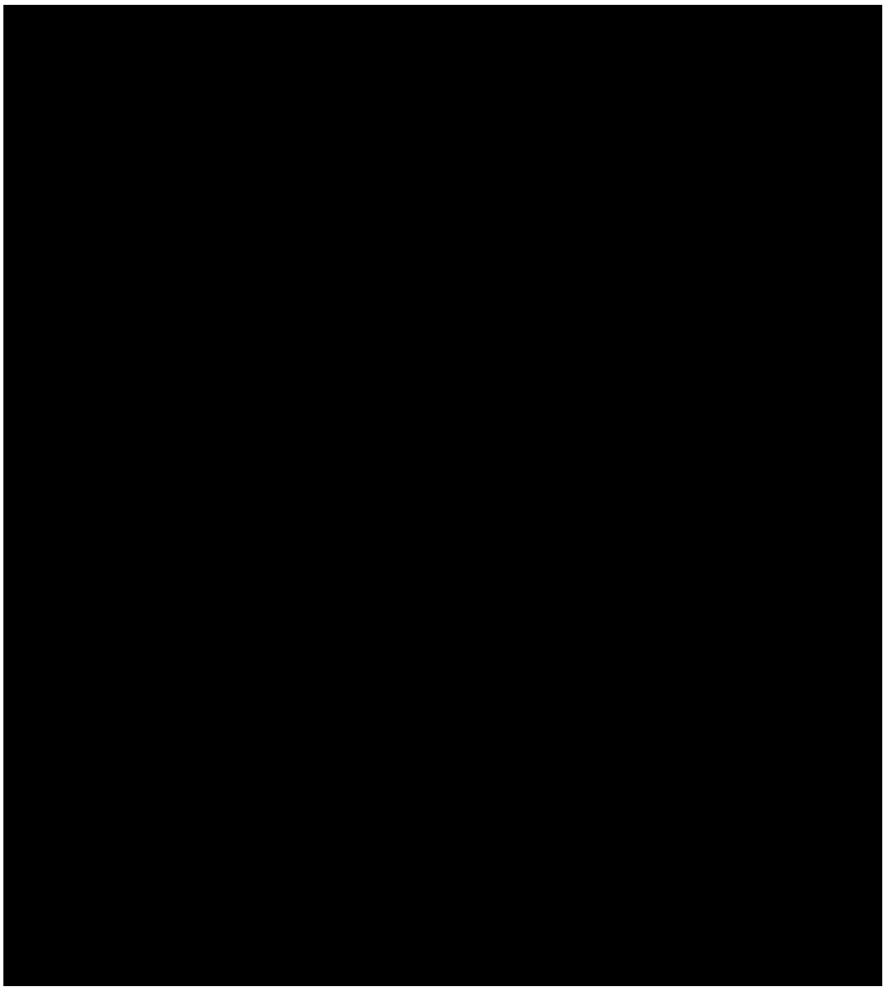


Рисунок 3.16 – Процес генерації команд

- Вибір та введення базових параметрів дрона: [redacted].
- Налаштування UART-портів: [redacted]

[redacted]

3. Керування згенерованими командами:

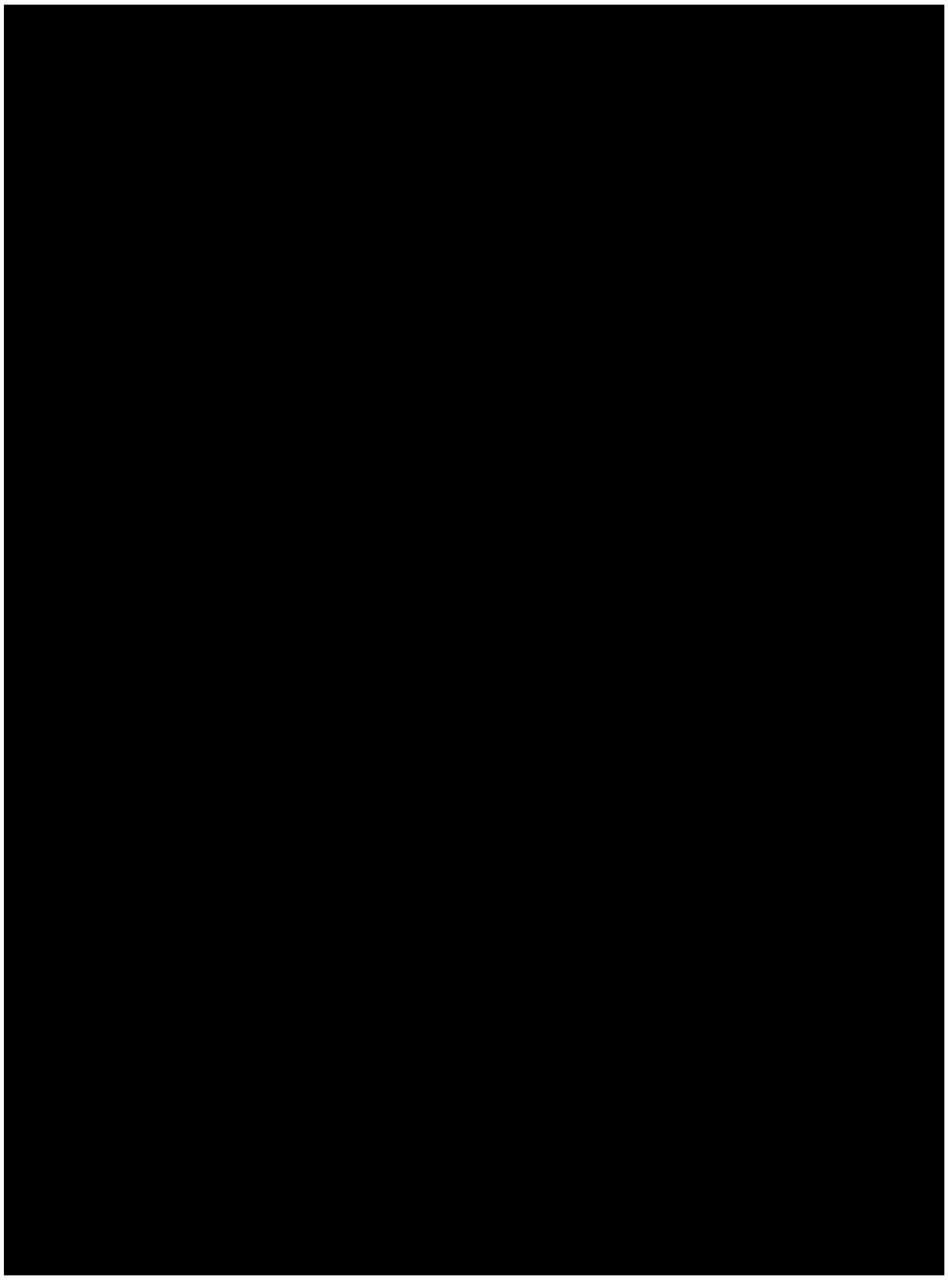


Рисунок 3.17 – Процес керування командами

- Копіювання команд у буфер обміну: За допомогою кнопки "Скопіювати" користувач може миттєво перенести весь згенерований CLI-текст до системного буфера обміну для подальшого вставлення в Betaflight Configurator.
- Збереження команд у текстовий файл: Кнопка "Зберегти в файл" дозволяє зберегти згенеровані команди у текстовий файл з розширенням .txt або .cli на локальному комп'ютері.
- Додавання поточної конфігурації до шаблонів: Кнопка "В шаблони" надає можливість зберегти поточну, повністю налаштовану конфігурацію як новий шаблон. Цей шаблон додається до списку на вкладці "Шаблони", забезпечуючи швидкий доступ до нього у майбутньому.

4. Робота з шаблонами:

Рисунок 3.18 – Процес роботи з шаблонами

- Перегляд списку збережених шаблонів: Перейшовши на вкладку "Шаблони", користувач бачить список усіх раніше збережених конфігурацій. Кожен шаблон представлений у вигляді окремого рядка з ключовою інформацією про нього (назва, модель [redacted], тип [redacted] тощо).

- Завантаження існуючих шаблонів для їх подальшого редагування або генерації CLI: При натисканні кнопки "Завантажити" поруч із шаблоном, всі поля на вкладці "Генератор CLI" автоматично заповнюються даними цього шаблону, дозволяючи користувачеві продовжити роботу з ним, внести зміни або повторно згенерувати команди.
- Видалення непотрібних шаблонів: Користувач може видалити будь-який шаблон зі списку, натиснувши кнопку "Видалити", що допомагає підтримувати порядок у збережених конфігураціях.

5. Після того як ми зберегли наші налаштування файлом (або скопіювавли в буфер) – в командному рядку betaflight виконуємо ці команди (рис. 3.18) й [redacted] налаштовано.

Рисунок 3.19 – Виконання згенерованих команд

Подальший розвиток програмного комплексу "Генератор CLI для Betaflight" з можливістю інтеграції із системами планування [redacted], телеметрії та автоматичного аналізу зібраних даних є надзвичайно доцільним та перспективним напрямком. Це розширення функціоналу перетворить програму на комплексний інструмент для керування дронами на всіх етапах, від конфігурації до експлуатації, створюючи єдину та зручну екосистему для користувача.

Інтеграція згаданих можливостей значно підвищить ефективність та зручність використання, оскільки користувачі зможуть виконувати всі операції з єдиного інтерфейсу замість кількох окремих програм. Це зменшить часові затрати, мінімізує ризик помилок та спростить загальний робочий процес. Окрім того, можливість автоматичного збору та аналізу даних телеметрії відкриє нові горизонти для оптимізації, дозволяючи програмі не лише генерувати команди, а й пропонувати рекомендації щодо покращення налаштувань на основі реальних польотних даних.

Таким чином, розширення функціоналу програмного комплексу збільшить його цінність для користувачів та підвищить конкурентоспроможність на ринку [redacted] та [redacted]. Це не лише логічно завершить функціональний цикл управління дроном, але й забезпечить програмі стратегічну перевагу, зробивши її незамінним помічником для спільноти [redacted].

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи досягнуто поставлену мету – розроблено програмне забезпечення для автоматизованого формування [redacted] для середовища Betaflight. У процесі дослідження вирішено низку теоретичних та прикладних завдань, які дозволяють оптимізувати налаштування [redacted] апаратів для типових конфігурацій [redacted].

1. Проаналізовано архітектуру конфігурації дронів у середовищі Betaflight, зокрема механізм роботи з CLI-командами, типові помилки користувача та проблеми при ручному налаштуванні.
2. Розроблено програмне забезпечення на мові Python з графічним інтерфейсом користувача, яке дозволяє здійснювати вибір конфігурацій [redacted] та генерувати CLI-команди автоматично.
3. Введено систему перевірки коректності введених користувачем даних, яка дозволяє уникнути конфліктів UART, недопустимих значень параметрів, помилок сумісності компонентів.
4. Застосовано шаблонний підхід до формування типових конфігурацій [redacted], що прискорює процес налаштування і забезпечує відтворюваність налаштувань.
5. Представлено механізм імпорту й експорту конфігурацій у форматах .txt та .json, що забезпечує збереження налаштувань та їх повторне використання.
6. Встановлено, що розроблене рішення дозволяє зменшити витрати часу на налаштування [redacted] щонайменше втричі, у порівнянні з ручною конфігурацією, та підвищити точність призначення параметрів.
7. Обґрунтовано доцільність подальшого розвитку програмного комплексу з можливістю інтеграції із системами планування [redacted], телеметрії та автоматичного аналізу зібраних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python. [Електронний ресурс] URL: https://docs.python.org/3/ (дата звернення: 02.06.2025).
2. Маттес Е. Пришвидшений курс Python. Практичний, проєктно-орієнтований вступ до програмування. Видавництво Старого Лева, 2021. 600 с.
3. W3Schools (Python). [Електронний ресурс] URL: https://www.w3schools.com/python/ (дата звернення: 05.06.2025).
4. VTX in UA. [Електронний ресурс] URL:https://vtx.in.ua/ (дата звернення: 06.06.2025).
5. Шамаю Г. Drone Theory. 2015. 170с.
6. Joshua Bardwell. FPV Drone Tutorials. [Електронний ресурс] URL: https://www.youtube.com/c/JoshuaBardwell (дата звернення: 02.06.2025).
7. Руденко В. М. Програмування мовою Python. Київ: Видавництво Ліра-К, 2021. 350 с.
8. Лисенко А. А., Кулаковський О. В., Кулаковська М. В. Системи керування безпілотними літальними апаратами. Харків: ХАІ, 2019. 280 с.
9. CustomTkinter Documentation. [Електронний ресурс] URL: https://customtkinter.tomschimansky.com/ (дата звернення: 06.06.2025).
10. Васильєв О.В. Python для початківців: Від основ до перших проектів. Київ: Київський політехнічний інститут, 2020. 200 с.
11. Colomina I., Molina P. Unmanned aerial systems for photogrammetry and remote sensing: A review. – ISPRS Journal of Photogrammetry and Remote Sensing, 2021, 79-87 p.
12. Torresan C., Berton A., Carotenuto F. et al. Forestry applications of UAVs in Europe: a review. – International Journal of Remote Sensing, 2021, 148-150 p.
13. Paneque-Gálvez J., McCall M.K., Napoletano B.M. Small drones for community-based forest monitoring: An assessment. – Remote Sensing, 2022.

14. Guo Q., Su Y., Hu T. LiDAR and UAV fusion for forest monitoring and ecosystem analysis. – Remote Sensing of Environment, 2022.
15. Tang L., Shao G. Drone remote sensing for forestry research and practices. – Journal of Forestry Research, 2022, 4-8 p.

ДОДАТКИ

ДОДАТОК А (main.py)

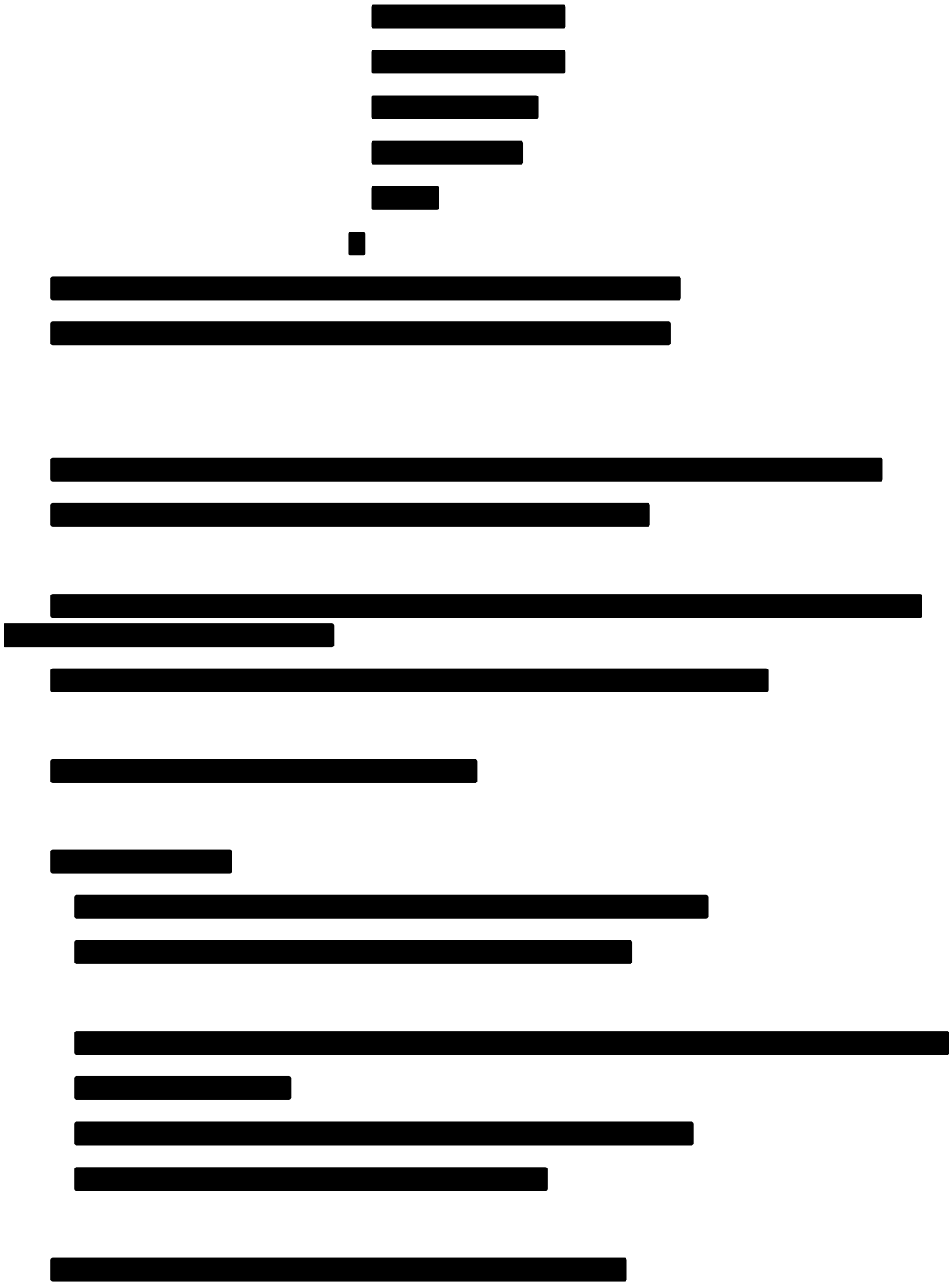
[redacted]

[redacted]

[redacted]

[redacted]

[redacted]



self.checkbox_frame = ctk.CTkFrame(self.cli_scroll_frame, fg_color="transparent")

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

ДОДАТОК Б (logic.py)

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]