Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

FPV UAV Firmware Review & Development

Текстова частина до курсової роботи

за спеціальністю «Інженерія програмного забезпечення» - 121

Керівник курсової роботи

Старший викладач

Курочкін А.В.

_____ (Підпис)

«___» _____ 2024 року

Виконав студент

Прохоров О. Д.

«___» _____ 2024 року

Київ 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Старший викладач

_____ Курочкін А.В.

«_____» _____ 2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Прохорову Олександру Денисовичу

факультету інформатики 3 курсу бакалаврської програми

ТЕМА: FPV UAV Firmware Review & Development

Зміст ТЧ до кваліфікаційної роботи:

Індивідуальне завдання

Календарний план

Анотація

Вступ

Загальна характеристика існуючих методів

Аналіз використаних технологій

Розробка MVP

Висновки

Використані джерела

Глосарій

Дата видачі «___» _____ 2023 р.

Керівник_____

(підпис)

Завдання отримав_____

(підпис)

I. Календарний план виконання роботи

№	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Визначення теми кваліфікаційної роботи	Жовтень 2023	
2.	Отримання завдання на кваліфікаційну роботу	Жовтень 2023	
3.	Огляд літератури	Листопад - Грудень 2023	
4.	Написання теоретичної частини кваліфікаційної роботи	Січень 2024	
5.	Написання програмної реалізації	Лютий 2024	
6.	Написання практичної частини кваліфікаційної роботи	Березень - Квітень 2024	
7.	Захист кваліфікаційної роботи	Травень 2024	

Студент _____

Керівник _____ “_____”_____ 2023

II. Анотація

Дане дослідження присвячене аналізу, розробці і впровадженню прошивки для польотного контролера для FPV дрона на базі мікроконтролера Raspberry Pi Pico. Під час виконання роботи було створено програму, яка складається з декількох логічних частин, кожна з яких відповідає за певні задачі : обробка вхідних сигналів, корекція позиції у просторі, передача команд на мотори. Система працює по принципу циклічного послідовного виконання команд (прийняти, обробити, передати), що є загальноприйнятою практикою при написання програм для мікроконтролерів.

Основна програма була написана за допомогою вбудованих в C++ бібліотек, офіційного SDK для Raspberry Pi Pico та пре компільованої Rust бібліотеки для парсингу CRSF пакетів. Проте варто зазначити, що впроваджена система має великий потенціал для розширення і ускладнення логіки.

Ціллю дослідження було створення польотного контролера, який дозволить без великих грошових витрат почати практикуватись у керуванні FPV дроном.

Підсумовуючи, результати цього дослідження несуть за собою позитивні зрушення в напрямку розробки власного українського польотного контролеру, що в перспективі може стати корисним продуктом, який буде приносити не тільки дослідницьку користь, а і практичну.

III. Table of Contents

I. Introduction

A. Modern FPV Flight controller's firmware and it's usage

The flight controller (onwards FC) is a kind of firmware that goes on the FC board – a specially designed and adjusted controller, which is responsible for communicating and controlling most UAV's components.

Different firmware allows different levels of adjustment and customization options. Using the right firmware will have as much impact on the drone's performance and behavior as on the ability to personalize the pilot's experience during flight.

By 2024, there will be a lot of open sources and commercial products for FPV's software. Each individual could determine for himself which one would be the most pleasant to use, but a large amount of these products are either obsolete or did not get much popularity in the community.

Most FC's firmware is a GitHub repository, driven by the community of independent developers and enthusiasts. This approach has pros and cons. A big independent community frequently creates secure and various code, but at the same time, this way of development has a lack of control of further steps in a workflow and motivates new developers to take part in a development of existed product, instead of starting their own one's.

The only well-known commercial UAV firmware is created by Chinese companies DJI and Autel Robotics, but this is proprietary software specially developed for a specific hardware setup. Any level of changes in their software would be considered a violation of the user agreement and could even result in the refusal of further maintenance.

B. Problematics: Open-source firmware good for commercial FCs, but not for custom made ones

Since the beginning of the full-scale invasion of Ukraine, UAVs have become one of the most significant technologies to resist Russian attacks. As a response to this, we observe a large increase in demand for a variety of FCs and other parts of the drone.

FC is a main part of the drone; most controllers are manufactured in China. By the year 2024, this country mostly monopolized this field, and due to the high demand for controller boards, this resulted in an increase in defective products.

When the situation dictates the requirement for a large number of drones possibility of decreasing any part of the drone would be very profitable because the more money the manufacturer saves – the more drones they manufacture.

So, if there is a demand for a low-price custom-made FC there is also a demand for custom made software for a specific hardware. Although a noticeable price reduction will obviously result in a quality reduction, there are still fields of usage that will not only ignore quality issues but rather benefit from them.

One of the important fields where cheap FCs would be greatly beneficial is decoy drones. The main purpose of this type of UAV is to engage with opposite forces to find and determine positions of defense systems or gaps where this system cannot reach drones.

The problem is further compounded by the fact that the time frame in which a piece of tech could be imported in Ukraine varies between months. This results in a constant shortage of major parts of the FPVs and other UAVs.

The main challenge is to create effective software that would be uploaded on a cheap microcontroller. Understand what is the most significant features that

should be implemented in custom-made software for FC.

C. Research Objective: Durable firmware for microcontroller development and deployment

The goal is to have a resilient product with minimally required comfort of use for an experienced pilot, because the final destination for a drone, which uses this firmware, is to be detected and destroyed.

Firstly, it aims to understand and determine core features that any FC firmware should have to be at least usable in real-world situations.

Secondly, it aims to find what practices are frequently used in FC's firmware, what pitfalls such as firmware and how different ways of implementation impact on end product and its functionality.

The research will contribute to the development of Ukrainian own FC's firmware and will contribute to the drone research industry. It will provide valuable knowledge on the inside work of the FCs, protocols and common algorithms that are useful in such type of software.

IV. General characteristic of existing solutions

A. What is Flight Controller firmware

FC firmware is the type of software that runs on a specially designed board, called Flight Controller, and fully operate and control an FPV drone. It affects each part of drone's behavior, performance, and sensor or other peripherals operation. Depending on flight controller's firmware there are advantages and disadvantages that impacts pilot's experience.

There are a lot of FC firmware options on the internet. Each varies in their functionality and purposes; on what FC generation it operates and so on. A lot of FC software were just forks of the forks of the clones of other repositories, what absolutely fist the title of open source.

Although each firmware differs each other in a case of usage, but most known ones will be suitable in most cases because their complexity and variety of settings.

1. Betaflight

Betaflight is an open-source software designed for multirotors and maintained by nearly 400 contributors. By the year 2024, it is one of the most popular firmware for UAVs.

The history of the Betaflight begins from firmware called Multiwii, a controller software that runs on 8-bit microcontrollers and, as the name hints, used a gyro board from the Nintendo Wii controllers. After that Multiwii was ported to be able to use 32-bit boards and named Baseflight, but due to disagreements within the community later was renamed to Cleanflight.

The Betaflight originally was a fork from Cleanflight to further experiment on UAV firmware and truly succeeded in this.

By version 4.5.0 Betaflight has an incredible list of features such as PID tuning, OSD configuration, and advanced flight modes. But they slowly refuse to support older versions of FC and begin denying accept targets F3 generation and less.

One of the main advantages of this firmware is high-performance orientation during the flight. Betaflight's main focus is on the freestyle and racing quadcopters, but at the same time, it supports other types of aircraft as, as not only, fixed wing and tri/hex/octocopters.

By the 2024th Betaflight maintainers focused on expanding support for advanced GPS rescue capabilities.

In addition to supporting different types of UAVs, as mentioned earlier, Betaflight has quite an astonishing range of ESC protocols support. The main one used in every case is DShot, but users also have the ability to choose other protocols like Oneshot, Multishot, and even PWM.

Another great thing about this firmware is Precision Tuning. Whether it is tuning of a whoop drone or a 5 or 7-inch quad, there is a wide range of constants which could be tuned according to demand.

Also, Betaflight can take advantage of a wide variety of sensors that are connected to the FC board. Depending on flight mode and one's demands, one used an accelerometer, a magnetometer, a barometer, or a GPS.

At the same time, Betaflight supports managing external or internal OSD for digital and analogue signals. There are a bunch of information that could be displayed, pretty much any information about drones or sensors could be

displayed.

1. ArduPilot

ArduPilot is an open-source unmanned vehicle autopilot software with as many as 700 contributors and an active community. It originated from Chris Anderson's and Jordi Munoz's hobby projects. In 2009 their company 3D Robotics released first ArduPilot board and later code repository.

As their web-site claims, ArduPilot one of the leading auto pilot and mission planning software, also depending on one's need UAV could be partially or fully controlled by a person with a controller instead of software. It supports multi-copters, traditional helicopters, fixed-wing aircraft, rovers, submarines, and antenna trackers.

ArduPilot possesses itself as a product that can be used for any mission. This software could be used either in the real world or simulator.

The basic goal of ArduPilot is to provide control of the vehicle either autonomously, via pilot input through a radio control transmitter or ground control station, or via companion computer on board the vehicle, any of these ways which are optional, including only loading a fully autonomous mission on the vehicle for execution.

ArduPilot uses all sorts of sensors to control and coordinate UAVs. All ArduPilot-compatible autopilots have at least one or more accelerometers, barometers, and gyros integrated onboard. In case of duplicate components ArduPilot would manage this situation and switch to working one in case of unexpected issues. Typically, GPS, and in most cases, compass is mandatory to properly fulfil a mission, and could be provided externally.

2. PX4

The PX4 autopilot is another open-source project with a great amount of about 600 contributors. This software is a flexible set of tools for the development of drone firmware. PX4 aims to share technologies and tailoring solutions for drone applications.

This software provides a standard to deliver hardware for drone and software stack. This approach allows the ecosystem to build and maintain hardware and software in a scalable way.

PX4 has many useful features, such as supporting many different vehicle frames/types: multicopters, fixed-wing aircraft, VTOLs (hybrid multicopter/fixed-wing), ground vehicles, and underwater vehicles. Such a variety of abilities gained PX4 its popularity among enthusiasts.

Also, the flight stack provided by PX4 is not limited to software alone, they also have their hardware setups for the flight controller, sensors, payloads, and other peripherals, so the end-user has a wide variety of possibilities: to use many variants of hardware he could find alone or to use Pixhawk, advanced autopilot designed and made for PX4.

Flight modes provided in the software have different levels of vehicle automation and autopilot assistance for the pilot. There are autonomous modes that are fully controlled by the autopilot, and require no remote input and manual modes, where pilot controls drone movements with different levels of help from firmware.

The most frequent use cases of autonomous modes are takeoff, returning to the home position, and landing. Other autonomous modes execute pre-programmed missions, follow a GPS beacon, or accept commands from an offboard computer or ground station.

Manual modes provide for user control with partial or no assistance from autopilot. The most used and known manual modes are acro – for acrobatic tricks and maximum user control, angle – stable angle control with help from the firmware and level – maximum stabilization and angle control from firmware.

One another feature of PX4 is failsafe systems to protect and recover vehicles if some unexpected situations have taken place. Users have the ability to specify areas and conditions under which fly is considered safe, and actions that should be performed if a failsafe is triggered.

3. DJI

DJI is a leading Chinese company specializing in drones and aerial image technologies. It began from a small office in 2006 and grew to become one of the largest technology companies in the world.

DJI produces a wide range of consumer types of drones designed for professionals and enthusiasts. The advantage of their drones is low entry level for inexperienced pilots. From the box, users have easy flight controls, built-in cameras, and automated flight modes for capturing stunning aerial photos and videos.

Also, DJI produces high-quality professional drones for filmmakers and photographers, surveyors, and other professionals. These drones often advanced features with higher-resolution cameras, longer flight times, and obstacle avoidance systems to support complex aerial tasks.

This company provides a list of products that have practically no analogues such as quadcopters (Phantom and Mavic) oriented on mass customers and multi-copters (Inspire, Spreading Wings), carious controllers for unmanned aerial vehicles (A2, Naza, and others), filming equipment for stabilization and

professional cinematography (Osmo, Ronin) and platforms that are actually itself drone designers (Guidance, Matrice 100).

Also, DJI has quite impressive equipment for FPV drones: FPV drones, which are designed for immersive and high-speed flight experiences. Along with this, they provide special goggles for the live video feed from a drone camera perspective, which allows the feeling of being onboard the vehicle.

The only disadvantages of DJI are the high price and proprietary software and hardware. DJI drones could be considered as premium product with a premium price tag. Also, although their software is a high-quality product in most cases it could be used only in the DJI ecosystem. DJI's closed ecosystem, comprising proprietary software and hardware, can limit customization and interoperability with third-party products and software solutions.

This results in strict rules of use and user agreements, any intrusion in hardware or software is a straight violation of these rules and could result in the refusal of support from DJI company.

Also, DJI products may not be readily available or supported in certain regions or countries due to regulatory restrictions, import/export regulations, or geopolitical factors. This can limit access to DJI's products and services for users in these areas, forcing them to seek alternatives.

4. Autel

Autel Robotics is another Chinese company widely known around the world and in the drone community. The company was founded in 2014 and quickly grew to a global scale. They are recognized for producing a diverse range of drones tailored for both professionals and enthusiasts.

One of their key advantages is accessible, compared to other private companies, entry points for pilots of all skill levels. This company positions itself as consumer-oriented and user-friendly. This accessibility makes them popular among hobbyists and beginners looking to explore aerial photography and videography.

In addition to hobbits-level drones, Autel provides high-quality professional drones designed to meet the demanding needs of filmmakers, photographers, surveyors, and other professionals. Their drones provide advanced features such as high-resolution cameras, extended flight times, and obstacle avoidance systems, enabling users to tackle complex aerial tasks with confidence and precision.

Autel's product range extends beyond drones to include various accessories and equipment for aerial imaging and cinematography. They offer stabilizers, camera mounts, and other filming equipment to enhance the quality and stability of aerial footage.

One standout offering from Autel is their FPV drone lineup, which provides immersive and high-speed flight experiences. Paired with specialized goggles that provide a live video feed from the drone's perspective.

But Autel also has its cons, one notable disadvantage is the potential limitations of its ecosystem. Autel's software and hardware may not be as widely compatible with third-party products and solutions, potentially restricting customization options for users. Additionally, availability and support for Autel products may vary depending on regulatory restrictions and geopolitical factors in certain regions or countries.

B. Comparison

Each user chooses software according to their needs and financial capabilities. Each of the provided earlier software occupies its own niche and is good in what it was designed for.

Betaflight is perfect for racing and freestyle due to responsive flight characteristics, low latency, and precise control for acrobatic maneuvers. They provide very flexible drone control settings with a great variety of constants that can be adjusted for one's needs.

Autel and PX4 on the other hand oriented for drone independent flying and mission planning and coordination. But even this two software have their differences. While ArduPilot has a feature-rich and versatile platform, PX4 on the other hand oriented on providing a technologies stack for developing something new, based on their platform. Also, unlike ArduPilot, PX4 is under a BSD-3-Clause license, which means the software also allows proprietary use and allows the releases under the license to be incorporated into proprietary products.

DJI and Autel are both prominent companies in the drone industry, offering consumer and professional drones with advanced features. DJI has a wider product range and market presence, known for its innovative technology and comprehensive ecosystem. Autel, while smaller, provides competitive alternatives with a focus on accessibility and quality, appealing to users seeking alternatives to DJI's proprietary ecosystem.

C. Custom software

As part of this work, PicoFlight firmware for the Raspberry Pi Pico board was developed. It is a small and efficient firmware written in C++ with the addition of Rust.

PicoFlight uses a periphery sensor for getting gyroscope and accelerometer data, can decode data from Rx with ELRS firmware and is able to send commands on ESC for controlling motors.

This firmware is able to operate in acro mode with stabilization that was implemented using data from the gyroscope and the desired angular velocity that the board gets from pilot input.

Itself PicoFlight is easy-to-understand firmware with three main logic modules: motors, Rx and IMU.

The small size of the software and well-documented code make it possible to quickly understand the logic of the operation of the firmware for the drone and could be a base for developing someone's own drone firmware.

It is clear that this firmware cannot compete with major players in this field, but it is still needed for the development of this industry and the emergence of new technologies and ways of development.

V. Analysis of the technologies used

A. Used resources

The PicoFlight consists mostly of C/C++ Pico SDK with the addition of a static Rust library for decoding CRSF packets. In the project has been used DShot digital protocol for communication with ESC, Kalman filters for filtering values from MPU and PID controller for drone stabilization.

B. DSHOT

DShot (or Digital Shot) is widely used in flight controllers for ESC communication. In the quadcopter community, it is considered the standard. The protocol is to transmit the target throttle value from the flight controller to the ESC which in turn decodes it and drives the motors.

A digital protocol has major benefits in comparison to analogue ones:

- DShot's packets have a checksum that ESC checks to confirm the correctness and integrity of data and that there was no interference.
- There are four types of DShot that are distinguished by their speed of operation from 150kbit/s to 1200kbit/s.
- No oscillator drift and therefore no need for calibration.

DShot is supported on all BLHELI_S, BLEHLI_32 and KISS ESCs, by the year 2024 it will be the most used firmware for ESCs. Also, there are no extra settings on the ESC is required – it automatically detects which used, so it works accordingly.

DShot has a pretty straightforward frame structure:

Throttle | Telemetry | CRC

- Throttle: 11-bit throttle, 2048 possible values. 0 is reserved for disarmed. 1-47 are reserved for special commands. And 48 to 2047 (2000 values) for the actual throttle.
- Telemetry: telemetry bit. If 1 - telemetry data is sent back via a separate channel, usually it is a separate wire that goes from ESC.
- CRC: 4-bit CRC to validate previous bits.

As a result, DShot sends a 16-bit frame with all necessary information.

The frame length is important for digital protocol, so each DShot type has its own time frame.

DSHOT	Bitrate	T1H	T0H	Bit (µs)	Frame (µs)
150	150kbit/s	5.00	2.50	6.67	106.72
300	300kbit/s	2.50	1.25	3.33	53.28
600	600kbit/s	1.50	0.625	1.67	26.72
1200	1200kbit/s	0.625	0.313	0.83	13.28

- T1H: the duration in µs for which the signal needs to be high in order to be counted as a 1.
- T0H: the duration in µs for which the signal needs to be high in order to be counted as a 0.

C. Receiver

1. CRSF packets

CRSF, or TBS Crossfire, packets are the fundamental units of data exchange between a controller and the receiver using the ELRS (ExpressLRS) protocol with

CRSF packets. These packets encapsulate control commands, telemetry data, and other vital information necessary for the operation and monitoring of your remote-controlled vehicle, such as a drone.

CRSF are able to send different packets of different lengths, but in most cases, this is a package consisting of 16 channels, each of which is responsible for one of the switches or joysticks on the remote controller.

The structure of the package looks like this:

Sync | Len | Type | Payload | CRC

- Sync: 1 byte (0xC8 or 0xEE).
- Len: 1 byte (Length of bytes that follow, including type, payload, and CRC).
- Type: Variable length (Packet type identifier).
- Payload: Variable length (Packet payload).
- CRC: 2 bytes to validate previous bits.

This kind of structure ensures proper synchronization, length indication, and error detection for CRSF packets, facilitating reliable not cringe communication between the transmitter and receiver modules in drone systems.

2. ELRS

ELRS (ExpressLRS) is an open-source Radio Link for Radio Control applications that was officially released in 2021 and has been growing at an incredibly fast rate.

Key features of the ELRS are low latency and long range. It was originally designed to be the best FPV Racing link, and by the year 2024, it is.

The protocol uses lightweight, highly optimized over-the-air (OTA) protocol to provide superior performance compared to legacy RC links.

ELRS was designed to be capable of extremely high packet rates (up to 1000Hz) and extreme sensitivity, this protocol was able to achieve ranges well over 100+ km with 2.4 GHz hardware.

While primarily designed for use with 2.4 GHz radio systems, ExpressLRS is also compatible with other frequency bands, such as 900 MHz (915/868). This allows users to choose the frequency band that best suits their needs and regulatory requirements.

Also, one of the significant features of ExpressLRS is adaptive frequency hopping. This is a technique to minimize interference and maintain a stable connection, even in crowded radio frequency environments. This helps ensure reliable communication between the transmitter and receiver.

However, due to a number of features, ELRS is more technically complex than other RC systems, which could result in issues while updating.

Along with that, this link protocol has the other disadvantage of lack of encryption. ELRS communication isn't secure and is not jam-resistant. Data over the air isn't encrypted, and there are no special security measures provided within the software.

D. Proportional Integral Derivative controller

A PID controller, also known as Proportional Integral Derivative controller, is one of the most used feedback control mechanisms in automation. The idea is to regulate a process variable by adjusting a manipulated variable based on the error between the desired point and the actual process variable.

Itself PID controller is a combination of proportional, integral, and derivative action to regulate a variable by adjusting a manipulated variable.

The PID algorithm is a crucial part of the drone's control system. There are three terms in a PID controller: Proportional (P), Integral (I), and Derivative (D).

- P (Proportional): relation to the present error or proportional error. The larger the error the bigger correction value.
- D (Derivative): future error prediction. It predicts how quickly the set point is approached and counteracts P to smooth overshoot when nearing the target, it is the derivative of the error.
- I (Integral): accumulation of all past errors. It is used to counteract external forces that occur over time, such as a drone drifting away from a set-point due to wind or an off-centred weight.

The PID controller output(co) is the sum of the proportional, Integral, and derivative.

$co(t)$ – controller output at time t.

$e(t)$ – error at a time t (desired position-actual position).

K_p, K_i, K_d – tunning constants (proportional, integral, and derivative terms).

So, the PID controller formula looks like:

$$co(t) = K_p * e(t) + K_i * \int_0^t e(\tau)d\tau + K_d * \frac{de(t)}{dt}$$

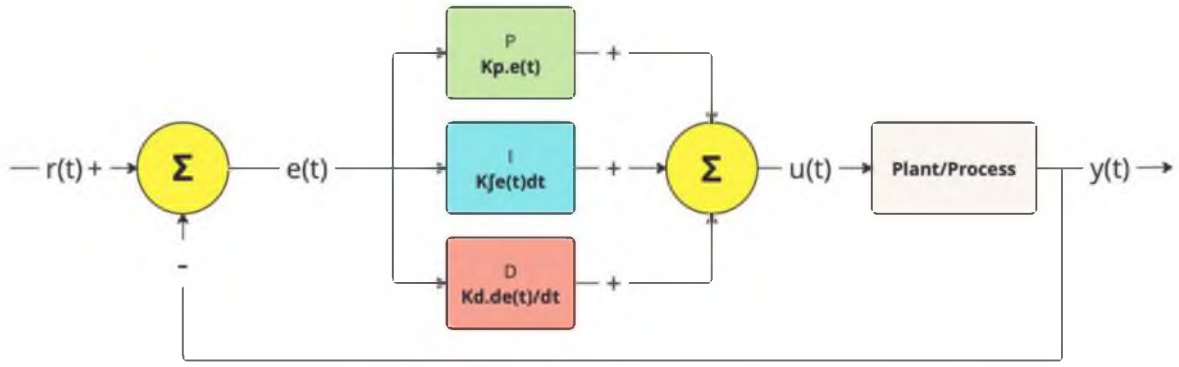

Fig. 1. Block diagram of PID controller

VI. Development of the MVP

A. General system design

The PicoFlight system consists of three parts: Motors, Rx, and IMU. Each part is responsible for a specific task like mutating raw values or reading external data.

The program has two cores (core0 and core1). Core1 is responsible for reading data from the MPU board and writing necessary information to static variables, so the other core will be able to get these values and continue the main loop without issues, while core0 is responsible for getting values from Rx, processing them, blinking the led on the board, depending on was arm or disarm command was sent, computing correction with PID and sending them to the ESC.

But using the same variables could cause so-called “race condition” or “race hazard” when core0 is trying to read data from variables and at the same time core1 is trying to write new values to these variables. To avoid this type of issue the mutex has been used.

Mutex is a kind of application-level lock, the idea of mutex is to secure the program execution and protect data that are used in multiple threads.

Mutex works in a way, that it could have an owner (a thread) and if another thread wants to take authority of this mutex and become its owner, it should wait for the other thread to free it.

```
static mutex_t mx;
...
void core1_entry() {
...
    mutex_enter_blocking(&mx);
        g_gyroRateX = mpu.getGyroRateX();
        g_gyroRateY = mpu.getGyroRateY();
        g_dt = (float)mpu.getDeltaTime();
```

```
    mutex_exit(&mx);
}
```

- *mutex_enter_blocking(mutex_t* mx)* – function to take ownership of a mutex.
- *mutex_exit(mutex_t* mx)* – function to release ownership of the mutex.

Core0, or main core, calls the *setup()* function to initialize all of the Pi Pico SDK's stdio types that are linked for the binary, Led on the Pi Pico board, mutex and launches core1.

Then all necessary classes are initialized: for motors and stabilization, for reading data from Rx and parsing CRSF packets. After that main loop starts.

In the main loop, there are 6 logic blocks:

1. Controlling led: if the drone is disarmed – blink led each 1 000 000 us (1 second), if armed – light without blinking.
2. CRSF parser: it reads bytes from Rx to uint_8 array, then sends this data to the *read_packet_from_bytes()* function, in a result, the program gets RcChannels structure, which contains 16 uint16_t values, representing data from each switch or joystick of the remote control.
3. Motor controller: responsible for the transformation of raw values from channels to Motor commands (throttle, yaw, roll, pitch, arm).
4. Getting data from MPU: simply copy values that core1 wrote to static variables to local ones.
5. Compute correction: sending desired angular velocity and actual angular velocity with a delta time to PID function, it is only needed to calculate correction for roll and pitch.
6. Mutate motors commands: calculation a value that should be sent to each motor depending on correction and previously got values.

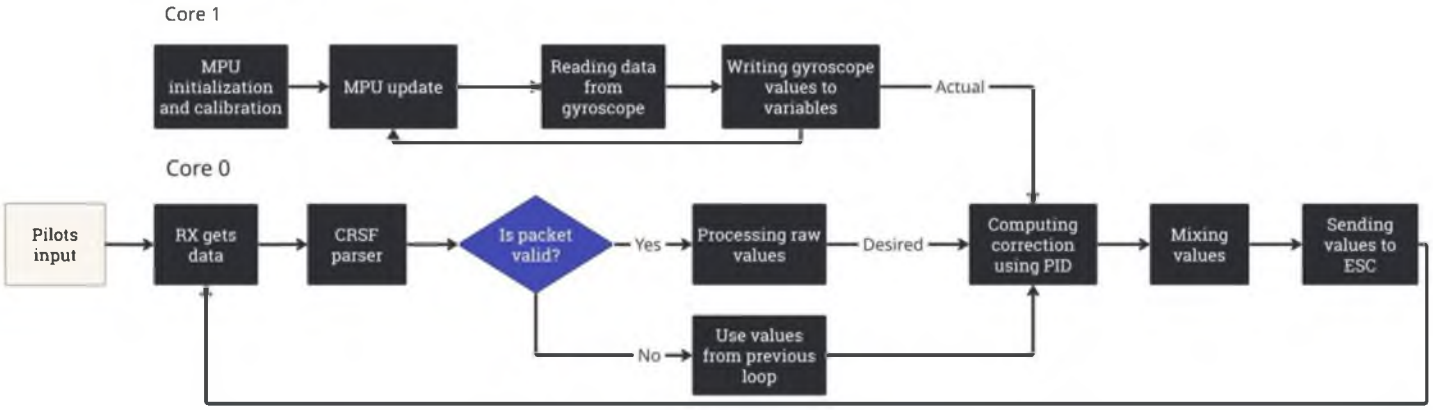

Fig. 2. Block diagram of the PicoFlight loop

B. Motors

1. DShot implementation

To implement the DShot protocol on the Pi Pico board there are a few significant things to use:

- PIO: programable input-output, it allows to write small, simple programs for PIO state machine. With this, it is possible to write and simulate any protocol that is not present by default.
- State machine: on the board, there are two of them, and it is required to choose which PIO program will run.
- Offset: offset where the program is loaded in PIO's 5-bit program address space.
- Pin: GPIO number on which the PIO program will operate.

On the initialization, the program should claim a free state machine on the chosen PIO and save its number. If no free state machine presents initialization functions will return false and the DShot instance will not do a thing.

Then program searches for the place in the instruction memory where there is enough space for the PIO program and saves this position to a variable.

After this program initializes with the provided PIO instance, state machine number, program's offset in instruction memory, GPIO number and baud rate. After all this program will be considered initialized and work appropriately.

In PicoFlight DShot 600 is used by default (but also supports 150, 300 and 1200).

DShot class is responsible for building packets that will be sent on ESC and sending these packets. Due to the protocol's sensitivity to time packets will be sent no more than every 30 us. This number was chosen because the timeframe of the DShot 600 packet is nearly 27 us and it is good practice to have a pause of one packet between each.

If there are no packets present zeros will be sent, what the disarm command will mean.

2. Motors class

The class is responsible for initializing four DShot instances on four pins and sending throttle on each of them.

The class also has two structures MotorsCommands with throttle, roll, pitch, yaw and arm values and MotorCommandsRaw with throttles that should be sent on each motor.

The function for sending throttle takes MotorCommandsRaw structure and calculates the coefficient which will be multiplied by the max throttle value. It was done to easily manage code and more efficiently change values during the tests.

3. MotorController class

This class is responsible for initializing the default MotorsCommands structure and updating it with the provided RcChannels structure.

MotorsCommands structure contains five values:

a. Throttle: value between 0 and 1000, responsible for all motor thrust.
b. Yaw: value between -500 and 500, represents the drone's angular velocity (degrees per second) in the yaw axis.
c. Roll: same as yaw, but in roll axis.
d. Pitch: same as yaw, but in pitch axis.
e. Arm: value could be 1000 (disarmed) or 2000 (armed).

Also, motor controller has failsafe features such as:

- To arm the drone throttle should be placed all the way down to zero. This prevents unexpected departures due to human error and inattention.
- To disarm the drone there should be at least five packets with disarm command. Sometimes the receiver could get the wrong values due to physical or some other reason and to prevent in-air disarming there should be more than one packet. For a person, a delay of 5 packets will be absolutely imperceptible, but it protects the safety of the drone many times over.
- If Rx did not get any valid packet – use last got values.

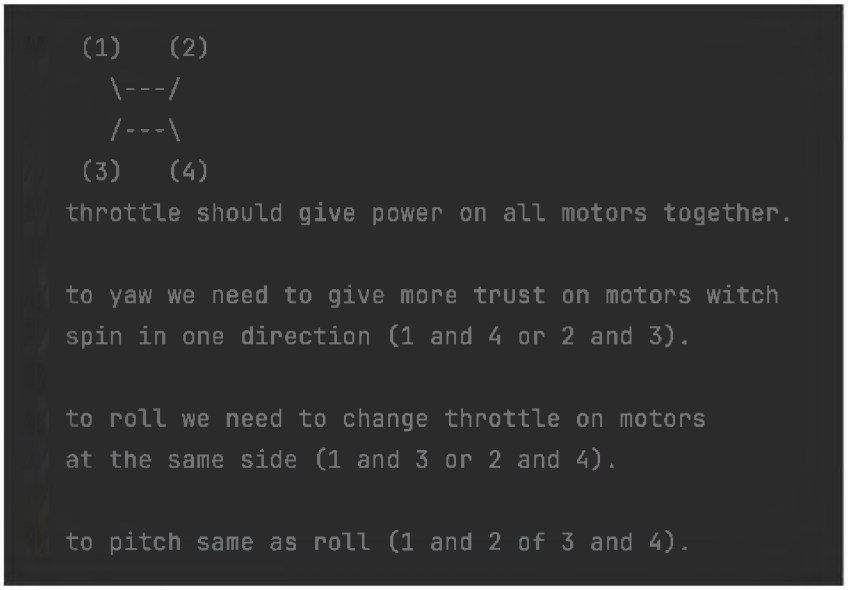

Fig. 3. Drone movements logic description

C. Receiver

1. Reading data from RX and Parsing CRSF packets

For reading data from Rx program initializes the UART instance with the provided baud rate, and GPIO numbers for Rx and Tx pins.

It is possible to read one byte (char) at a time, so in the main program loop, it collects 64 characters from UART and then transfers this array to the *read_packet_from_bytes()* function. Of course, there is a check that is UART readable when each type of program tries to get char, this prevents reading random data or the program's fail because there is nothing to read.

read_packet_from_bytes() four values:

- CrsfPacketParser instance, which is declared at Rust library and included in the project with *extern "C"*:

```
extern "C" {
    CrsfPacketParser create_parser();
    ...
}
```

- RcChannels structure, in which all channels' values would be written.
- uint8_t array with bytes, that was previously red from Rx.
- Size of the array.

D. IMU

1. Initialization and Data reading

For IMU the program initializes i2c for provided SDA and SCL pins with a baud rate of 400 000. Then reset the MPU by sending 0x6B (01101011) on MPU (address 0x68).

Then calibration begins, the program reads 500 values from MPU, then reads 500 more but with adding these values to each other to get the middle aromatic. It is needed because the MPU board could have a small drift in one of the axes, which would affect the drone's stabilization.

In MPU's update function program reads raw values for the accelerometer (address 0x3B) and gyroscope (address 0x43), calculates delta time (dt), and calculates gyroscope rates with previously defined sensitivity scale.

The sensitivity scale simply affects values' diapason:

Diapason (°/s)	Sensitivity (LSB/°/s)
± 250	131
± 500	65.5
± 1000	32.8
± 2000	16.4

2. Motors throttle coefficient Stabilization and Mixing

To achieve stabilization program uses a PID controller with predefined constants. There are four constants used in the controller:

- G: for "gravitation" force, it is used to multiply correction by 0.01.
- K_p: proportional value.
- K_i: integral value.
- K_d: derivative value.

To calculate the correction program finds an error - the difference between the desired angular velocity and the actual angular velocity. Then to calculate an integrator program simply adds current error to it (integrator is an accumulation of all previous errors). In the end, the program finds the correction value with the PID controller formula and returns it. The program has stearate PID instances for roll and pitch values.

Resulted values with yaw were then multiplied with a mixer to reduce motor power and used to calculate each motor throttle individually.

The value of each motor is calculated with this mixing table:

Motor	Throttle	Roll	Pitch	Yaw
M1	1	-1	1	-1
M2	1	-1	-1	1
M3	1	1	1	1
M4	1	1	-1	-1

VII. Further work and Conclusions

A. Further work possibilities

Although the system is working in its current form it has the potential for further improvements and expanding the functionality. There are at least two more flight modes that could be implemented: angle and altitude hold. This would allow for less skilled pilots to use this firmware.

The addition of new sensors could also further enhance the system with new ways of usage. However, such improvements would require more hardware which would increase the cost of development and product.

B. Conclusions

The research that was conducted in this study has resulted in the development of small, but still useful and fast FPV firmware. The system is developed as MVP for the Raspberry Pi Pico microcontroller using their C/C++ SDK and Rust library. The firmware successfully implemented acro mode for the drone, along with all necessary systems.

The main focus of the system was to research the ability to create custom firmware for FPV drone. Each module of the system could be used separately for one's needs and could work independently. The use of Rust with its great known reliability, safety and speed contributed to the firmware's efficiency.

In conclusion, this research resulted in the development of a new working drone firmware and scientific material that can be used to create another new firmware with other components. While there are many options for improvement, the firmware in its state is still a great step in the development of the new drone firmware and could inspire other people to try steps in this field.

VIII. References

1. Betaflight. Betaflight Wiki [Electronic resource] / Betaflight. – Access mode: https://betaflight.com/docs/wiki

2. ArduPilot. History of ArduPilot [Electronic resource] / ArduPilot. – Access mode: https://ardupilot.org/copter/docs/common-history-of-ardupilot.html

3. PX4. PX4 User Documentation [Electronic resource] / PX4. – Access mode: https://docs.px4.io

4. PX4. Software Overview [Electronic resource] / PX4. – Access mode: https://px4.io/software/software-overview/

5. Raspberry Pi. C/C++ SDK for RP2040 microcontrollers [Electronic resource] / Raspberry Pi. – Access mode: https://www.raspberrypi.com/documentation/microcontrollers/c_sdk.html

6. Raspberry Pi. Raspberry Pi Pico C/C++ SDK [Electronic resource] / Raspberry Pi. – Access mode: https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf

7. Raspberry Pi. Getting started with Pico [Electronic resource] / Raspberry Pi. – Access mode: https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf

8. ExpressLRS. Home Page [Electronic resource] / ExpressLRS. – Access mode: https://www.expresslrs.org

9. AllDatasheet. MPU-6050 Datasheet [Electronic resource] / TDK. – Access mode: https://pdf1.alldatasheet.com/datasheet-pdf/view/1132807/TDK/MPU-6050.html

10. BrushlessWhoop. DShot and Bidirectional DShot [Electronic resource] / BrushlessWhoop. – Access mode: https://brushlesswhoop.com/dshot-and-bidirectional-dshot/

11. DJI. DJI Global Home Page [Electronic resource] / DJI. – Access mode: https://www.dji.com/global

12. Autel Robotics. Autel Robotics Home Page [Electronic resource] / Autel Robotics. – Access mode: https://www.autelrobotics.com

IX. Glossary

CRC (Cyclic Redundancy Check) - An error-detecting code used to detect accidental changes to raw data, particularly in digital networks and storage devices.

ESC (Electronic Speed Controller) - A device that regulates the speed of an electric motor, commonly used in drones to control propeller speed.

FC (Flight Controller) - The main processing unit of a drone that controls its stabilization and flight operations.

F3 - Refers to a type of flight controller with an STM32F3 processor, known for good performance in flight control tasks.

FPV (First Person View) - A method used to control a drone from the pilot's viewpoint, typically using an onboard camera and remote video stream.

GPIO (General Purpose Input/Output) - Pins on a microcontroller board like the Pi Pico that can be programmed to perform either input or output operations.

IMU (Inertial Measurement Unit) - A sensor device that measures and reports force, angular rate, and sometimes magnetic field surrounding the drone.

MPU (Microprocessor Unit) - In this context, often refers to the motion processing units used in drones for processing information from motion sensors.

OSD (On-Screen Display) - A system used to overlay flight data (like altitude, speed, and battery status) onto the video feed from an FPV drone.

PID (Proportional Integral Derivative) - A control loop feedback mechanism widely used in industrial control systems and drones for stable flight.

PIO (Programmable Input/Output) - A feature of the Pi Pico microcontrollers, allowing the programming of pins for various input/output.

RC (Remote Control) - A system that is used to control a drone or other device remotely.

Rx (Receiver) - The device on the drone that receives signals from the transmitter.

SCL (Serial Clock Line) - Used in I2C communication, this is the line that carries the clock signal.

SDA (Serial Data Line) - Used in I2C communication, it is the line through which data is sent and received.

Tx (Transmitter) - The device used to send commands from the pilot to the drone's receiver.

UART (Universal Asynchronous Receiver/Transmitter) - A hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable.

UAV (Unmanned Aerial Vehicle) - A drone or aircraft piloted remotely or autonomously, without a human onboard.

VTOLs (Vertical Take-Off and Landing) - Aircraft that can take off, hover, and land vertically.