Embedded systems design of a low-cost flight controller for use in UAV platforms

Daniel Fraser[1*], *Arnold* Pretorius[1], *James* Hepworth [1], *Natasha* Botha[2]
[1]Mechanical Engineering Department, University of Cape Town, Rondebosch, Cape Town, 7700
[2]Centre for Robotics and Future Production, Council for Scientific and Industrial Research, Pretoria 0184

Abstract: Serving as a critical interface between high-level software and low-level electronic systems, flight controllers are essential for aerial robotic applications. However, their significant cost poses a barrier to widespread adoption. While affordable, off-the-shelf flight controllers exist, they generally do not support the firmware required for research- and industry-level implementation, such as PX4. This paper presents embedded system designs for low-cost flight controllers that are PX4 compliant. The research delves into examining the necessary components, existing open-source firmware/software, and appropriate hardware solutions. The presented designs are based on open standards and capable of operating with various platforms, applications, and components. The final design is shown to be low in cost while also demonstrating substantial potential for integration with existing applications. The flight controllers were verified with a range of tests, including the implementation of a control system that was responsible for governing the roll angle of a quadcopter. The overall flight controller performance is shown to be comparable with the performance and functionality of commercial solutions.

1 Introduction

Unmanned aerial vehicles (UAVs) have become a popular area of research in robotics over the past decade. However, the significant cost and knowledge barrier of UAV hardware remains a challenge for research and commercial entities in developing regions, such as Southern Africa. To address the high cost and to grow in-house knowledge, the Aerial Robotics Capability (ARC) Project, led by the MechatronicSystem.Group in partnership with the CSIR, aims to develop a low-cost quadcopter platform that can be used for a wide range of research and industry applications. One aspect of Project ARC is designing a low-cost extendible flight controller for use on multiple research platforms. The flight controller aims to be both robust and convenient to use, whilst also affordable at scale.

The flight controller is an embedded system responsible for the reliable operation of a UAV. The system's primary objectives are to collect signals from internal/external devices, convert them into meaningful data, and output signals for communication and control. Flight

* Corresponding author: frsdan004@myuct.ac.za

controllers require many components to operate reliably. These typically include: sensors, such as accelerometers, gyroscopes, a magnetometer, and a barometer; microcontrollers; power regulators; storage devices, and interfacing components [1]. Additionally, flight controllers typically support wireless communication through Wi-Fi/Bluetooth via a System on Chip (SoC) or an external radio module. The flight controller components are mounted and connected using a Printed Circuit Board (PCB) for mass reduction and repeatability. Alongside the flight controller hardware, firmware is essential in the design of the embedded system, as it facilitates interaction with the device at a higher level, typically through software. The firmware is responsible for translating high-level commands, such as from a control loop, into low-level signals, enabling the operation of individual components. By bridging the gap between hardware and software, firmware enables developers to control and manipulate the device's functions more effectively.

To address the challenges outlined above and contribute meaningfully to the field of UAV systems development, this research sets out to make the following contributions:

- Conduct a critical literature review on flight controller fundamentals, including embedded systems, communication protocols, and hardware design.
- Evaluate existing flight controller designs to build in-house expertise, with emphasis on open-source hardware and implementation strategies.
- Identify and selected suitable open hardware standards to support prototype development and ensure post-development applicability.
- Design and develop a custom flight controller architecture tailored to the specific requirements of the target UAV platform.
- Ensure compatibility with existing open-source software and firmware for seamless integration.
- Assess system performance, robustness, cost-effectiveness, and limitations through iterative prototyping and testing.

1.1 Open-sourced firmware

Whilst development of custom-use firmware is possible for a flight controller, using community-driven or open-source firmware enables further integration into an ecosystem of existing applications and software.

1.1.1 PX4-Autopilot

PX4-Autopilot (PX4) is an open-source flight control firmware for drones and other unmanned vehicles built on Nuttx, a real-time operating system (RTOS) by Apache [2].
The project provides a flexible set of tools for drone developers to share technologies and create tailored solutions for drone applications. PX4 provides standards to deliver drone hardware and software stack support, allowing for a scalable ecosystem.

PX4, like Pixhawk and MAVLink, is a project under the Dronecode Foundation. The Dronecode Foundation is an umbrella organisation that oversees a suite of open-source projects related to UAV hardware, software, and firmware. One of the key advantages of the Foundation is its role in maintaining interoperability and compatibility across its various projects, in collaboration with the broader developer community [3].

PX4 offers a comprehensive guide for each supported hardware system, providing users with information necessary to develop UAV solutions. PX4 supports a wide array of flight controllers, telemetry modules, Electronic Speed Controllers (ESCs), and computers, making it an ideal choice for developers with diverse specifications. With these supported components, numerous UAV airframe configurations are supported, including: multicopters, fixed-wing aircraft, and vertical take-off and landing (VTOL) systems. Additionally, PX4 firmware can be extended to the following airframes that are considered experimental by the project: helicopters, rovers, and submarines [4].

The basic stack of PX4-Autopilot consists of multiple layers; each serving distinct functions to accommodate various use cases and ensure extensibility. Within the PX4 ecosystem, different applications communicate with one another using internal messages known as uORB topics. These messages facilitate operation and coordination across all firmware system components.

PX4-Autopilot firmware provides supported flight controller hardware with the following features:

- Drivers: Scripts that collect and convert sensor or product signals into PX4-readable data, used for both internal and external sensors and peripherals.
- External Connectivity: PX4 supports protocols like MAVLink and ROS2 for communication with ground control stations and other devices via the uXRCE-DDS PX4-ROS2 bridge.
- Storage: Handlers that organise firmware data in storage devices such as SD cards and flash memory.
- Flight Control: PX4 includes algorithms for autonomous flight, as well as position and attitude control of UAVs [5].

1.1.2 Other notable open-source firmware

ArduPilot: is a trusted, versatile, and open-source autopilot system that supports a wide range of vehicle types, including multi-copters, traditional helicopters, fixed-wing aircraft, boats, submarines, rovers, and more. The source code is developed by a large community of professionals and enthusiasts. ArduPilot hosts Developer Team Forums, which are open and offer daily development insights [6].
Betaflight: is another leading open-source multi-rotor flight control software. The software is highly regarded within the global first-person view (FPV) drone racing and freestyle community for its performance, precision, cutting-edge features, reliability, and extensive hardware support [7].

1.2 Software considerations

Software is of paramount importance in flight controller development, as it provides an abstraction layer enabling the use of high-level commands to design models and set parameters, to achieve a desired system solution. Identifying the desired software aids in development as it helps to avoid compatibility issues with the flight controller firmware. Additionally, using user-friendly software encourages greater development and adoption. The following subsections detail some commonly used software packages for UAVs.

1.2.1 QGroundControl

QGroundControl (QGC) is a ground station designed for UAV systems and is another Dronecode Foundation project. QGC provides full flight control and vehicle setup for PX4-equipped vehicles, and facilitates easy and straightforward usage for beginners, while still delivering high-end feature support for experienced users [8]. Some of these features include:

- Full setup/configuration of ArduPilot and PX4 powered vehicles.
- Flight support for vehicles running PX4 and ArduPilot (or any other autopilot that communicates using the MAVLink protocol).
- Mission planning for autonomous flight.
- Flight map display showing vehicle position, flight track, waypoints and vehicle instruments.
- 3D viewer visualizing the 3D map of the environment (.osm file), the 3D model of the vehicle (only multi-rotors for the moment), and the mission 3D trajectory (including the waypoints).
- Video streaming with instrument display overlays.
- Support for managing multiple vehicles.
- QGC runs on Windows, OS X, Linux platforms, iOS and Android devices.

1.2.2 MATLAB and Simulink

MATLAB and Simulink are popular graphical programming software tools developed by MathWorks. By leveraging various toolboxes within MATLAB and Simulink, rapid development of code and functionality becomes feasible. These tools simplify the development process compared to traditional software methods, as a single block in Simulink can replace multiple lines of code. A designated toolbox seamlessly handles the interface between the host PC and component. These toolboxes allow MATLAB & Simulink the ability to support various hardware platforms, including STM32-based micro-controllers, ESP32, and PX4-based flight controllers.

In the context of flight controllers, MATLAB & Simulink currently only support PX4-based flight controllers through the UAV Toolbox Support Package for PX4-Autopilots. The toolbox enables the design of flight control algorithms, estimators, and navigators in Simulink. These models can then be deployed to PX4-enabled flight controllers in a computationally efficient way. The toolbox integrates the generated code blocks from the Simulink models with the PX4 flight stack. This provides developers with the ability to further customize PX4 features for more specific use cases [9].

Fig. 1 illustrates some examples of Simulink blocks that are used to interface with the PX4 architecture. The blocks shown are based on the functionality the toolbox provides by allowing users to read and write internal uORB messages. This capability supports the development of custom models and applications. The Simukink blocks allow access to various onboard sensor data (such as accelerometer readings), facilitating real-time system status monitoring. Furthermore, uORB blocks are used to enable the generation of PWM signals based on a running Simulink model, such as a flight control algorithm. These signals are then transmitted to the ESCs and motors, allowing for precise control of the UAV's movement.

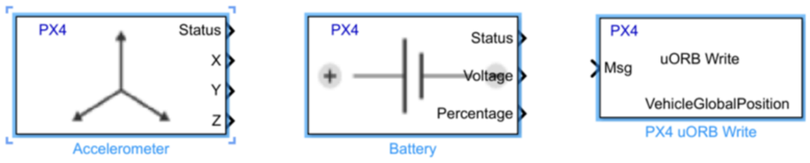

Fig. 1: The MATLAB & Simulink UAV Toolbox Support Package for PX4-Autopilots uORB message-based Simulink blocks. [9].

Other than the ability to read and write uORB messages, some other key features of the UAV Toolbox Support Package for PX4-Autopilots include:

Connected I/O**:**
The I/O interface allows Simulink to interact with hardware prior to deploying the full model. With Connected I/O, the Simulink model runs on the host PC rather than the flight controller, eliminating the need to deploy the entire model for each modification. Using this feature reduces the time required for iterative design adjustments [10].

External mode:
Unlike Connected I/O, External Mode functions by running the Simulink model directly on the PX4-based hardware, rather than on the host PC. The External Modes are further broken up into: *Build, Deploy and Start*, and *Monitor and Tune*:

The *Build, Deploy, and Start* function generates C code from the Simulink model based on the connections and functions of the uORB messages, and then deploys it directly onto PX4-based hardware. Once deployed, the model runs autonomously on the board every time it boots up, with no communication needed with the original Simulink model.

The *Monitor and Tune* action enables the modification and monitoring of specific parameters within a Simulink model running on target hardware. This feature allows for real-time adjustments to model parameters, with changes communicated immediately to the target hardware. The effects of different parameter values can be observed by viewing the output of various uORB messages through the Simulink blocks in real time. Like Connected I/O, *Monitor and Tune* accelerates development, as the entire model does not need to be rerun each time for small parameter changes [11].

1.3 Open-sourced flight controller solutions

To gain in-house knowledge of flight controller systems, solutions with a degree of open-source information are examined. A notable drawback of closed solutions is their limited scope for customisation, as they are designed for general use cases and provide interfaces for all functions, regardless of whether the user requires them. In contrast, open-source solutions offer valuable insights into the inner workings of the systems, enabling a detailed understanding of the various functions used.

1.3.1 Bitcraze

Bitcraze, founded in 2009, is a Swedish company that focuses on open-sourced, small-scale, low-cost UAV quadcopter products. According to the Bitcraze website: “The goal is to enable people to explore, investigate, innovate, research and educate – that’s why all our stuff is open. Go play with it!”. The mission of the company is to encourage community-driven development in the UAV space. Bitcraze does this by providing open-sourced schematics of their products where they allow open sharing and adaptation [12].

Crazyflie 2.1+: Bitcraze’s current flagship product, the Crazyflie 2.1+ is a small-scale, low-cost solution weighing 27 grams. Despite the small size, the Crazyflie 2.1+ is a comprehensive, all-in-one quadcopter solution with all the necessary components needed for flight. A distinctive feature of the Crazyflie 2.1+ is its frame, which is made from the PCB itself. The flight control system schematics are publicly available and allow for the viewing of all the connections and components used. Although schematics are provided the hardware designs are not publicly available. The retail price of a single Crazyflie 2.1+ is currently $240 [13].

The features of the Crazyflie 2.1+ include:

- STM32F405 primary micro-controller (Cortex-M4, 168MHz, 192kb SRAM, 1Mb flash).
- nRF51822 radio and power management SoC (Cortex-M0, 32Mhz, 16kb SRAM,
- 128kb flash).
- Micro-USB connector.
- On-board LiPo charger with 100mA, 500mA and 980mA modes available.
- 8KB EEPROM memory.
- BMI088 6-axis IMU (3-axis gyroscope, 3-axis accelerometer).
- BMP388 barometer.

Crazyflie Bolt 1.1: Bitcraze has also introduced an extendable version of the Crazyflie's flight controller, called the Crazyflie Bolt 1.1, as shown in Fig. 2. This product shares virtually the same components and features with the Crazyflie 2.1+, but also offers the added capability of being extendable to larger quadcopters. The Bolt has built-in connectors to send the PWM signals to external electronic speed controllers and motors, and uses similar pinouts and sensors as the Crazyflie2.1+, ensuring compatibility and ease of integration. The Crazyflie Bolt 1.1 is still in the early access phase with a current retail price of $205 [14].

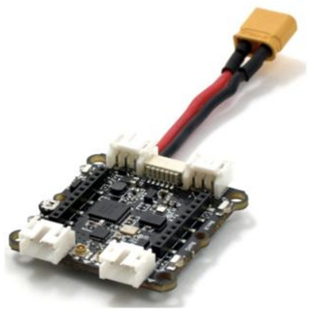
Fig 2: Bitcraze's Crazyflie Bolt 1.1 [14].

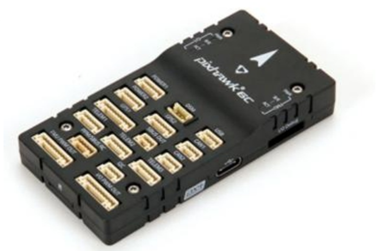
Fig 3: Pixhawk 6C flight controller. [16]

1.3.2 Pixhawk

Pixhawk is an independent, open-hardware project delivering accessible, cost-effective, and high-quality autopilot hardware design standards for academic, hobbyist, and industrial communities. Pixhawk is another project by the DroneCode Foundation and serves as the reference hardware platform for the PX4-Autopilot firmware.

The Pixhawk project outlines many different open hardware reference standards to provide comprehensive hardware specifications and guidelines for UAV system development. The standards cover all aspects of mechanical and electrical specifications necessary for creating Pixhawk-compliant UAV system components [15].

Within the Pixhawk standards, various pin-out and block diagrams serve as essential guidelines for developing Pixhawk-compliant flight controllers. A key advantage of adhering to these standards is the integration into the PX4-Firmware ecosystem, providing access to a wide range of applications and components. The Pixhawk project not only establishes reference standards but also supports existing products in the market, such as those manufactured by Holybro. Manufacturers have utilised open designs to develop various PX4 and Pixhawk-compliant flight controllers with form factors suited for applications ranging from cargo transport to first-person view (FPV) racing [16].

Pixhawk 6C: One product supported by the Pixhawk project and manufactured by Holybro is the Pixhawk 6C. The 6C is the most cost-effective Pixhawk flight controller currently available and is shown in Fig. 3. As with all Pixhawk controllers, the PCB is embedded into a casing, providing convenient connectors for users to interface with the board. The connectors are designed for interfacing with external devices and comply with the Pixhawk connector standard. The flight controller currently retails for $140.99 for a
plastic case and $150.99 for an aluminum case [16].

Similar to the Crazyflie, the hardware designs are not publicly available, however, the Pixhawk 6C design adheres to the FMU-v6C Pixhawk open standard. Key details are illustrated in the block diagram in Fig. 4. Open standards like the FMU-v6C specify the necessary components and provide pin maps that detail internal component connections and external port pins. These standards identify the components used and specify detailed pin functions for each micro-controller pin, equipping manufacturers with essential information to develop compatible hardware [17]. Some key features/components of the Pixhawk 6c are:

- STM32H743 Primary micro-controller/Flight Management Unit (32 Bit Arm, Cortex-M7, 480MHz, 2MB memory, 1MB SRAM).
- STM32F10 Fail-Safe Co-Processor (32 Bit Arm Cortex-M3, 72MHz,64KB SRAM).
- BMI055 Accelerometer/Gyroscope Redundant IMU.
- ICM-42688-P Accelerometer/Gyroscope Redundant IMU.
- MS5611 Barometer.
- IST8310 Magnetometer.

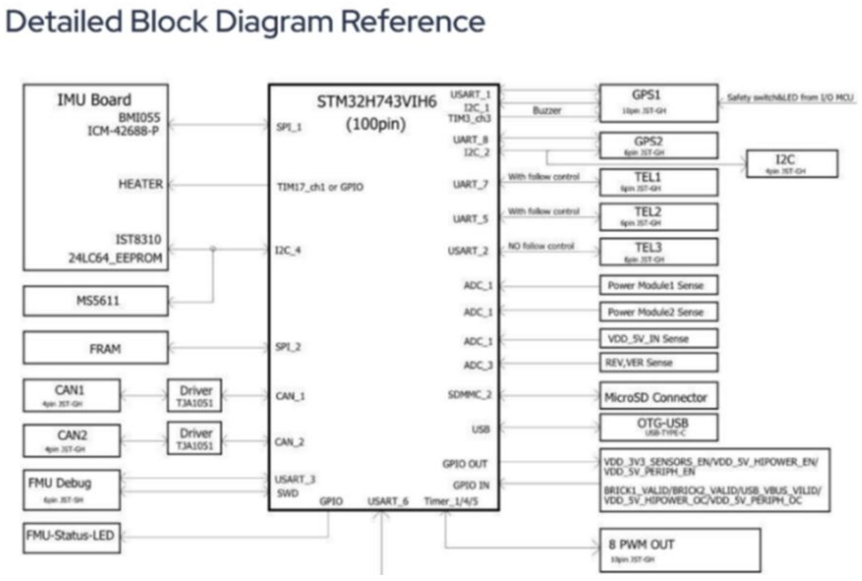

Fig 4: FMU- v6C flight controller open reference block diagram [17].

2 Design framework

A detailed User Needs Analysis, (given in the Appendix) under Project ARC, focused on designing a cost-effective and high-performing flight controller tailored for research and industrial applications. Central to the analysis was the selection of PX4 as the preferred firmware due to its open-source nature, community-driven ecosystem, and direct compatibility with MATLAB & Simulink—tools already established within the Mechatronic Systems Group. PX4's integration with DroneCode initiatives and support for a wide range of additional software and hardware applications further ensures adaptability and extensibility across multiple platforms.

To meet the technical requirements of Project ARC, the hardware design process followed open-source PX4 reference standards to ensure compatibility with sensors, microcontrollers, and essential interfacing components. Emphasis was placed on user-friendly interfacing through standardised connectors and protocols, thereby enabling straightforward integration with target peripherals. The identified Project ARC target peripherals include four PWM signal inputs for ESCs, a UART serial interface for the companion computer, a UART serial interface for the telemetry radio or radio receiver (which can also provide telemetry), and a 5V battery power module input consisting of 5V and GND power lines. Given Project ARC's cost ceiling of R5000 for the entire quadcopter, cost-effective PCB development emerged as a necessity, with manufacturing optimised for low cost while maintaining robustness. JLCPCB was selected as the most cost-effective manufacturer, allowing for full fabrication and assembly services.

2.1 Design requirements

Based on the context above as well as a User Needs Analysis (given in the Appendix), general requirements were established to guide the detailed specifications for flight controller prototyping. The requirements (R) are given in Table 1, covering the critical categories of the system.

Table 1. High level requirements for the flight controller design grouped by critical categories.

Categories	#	Requirement
Firmware and Software	R1	The flight controller must have all the necessary communication interfaces with a host PC for a chosen PX4-Autopilot firmware.
	R2	The flight controller must have the capability to successfully receive a deployed PX4-Autopilot firmware version.
	R3	The flight controller should be fully compatible with PX4 supported applications such as QGroundControl.
	R4	The flight controller should be compatible with MATLAB & Simulink.
Flight Controller Design	R5	The design of the flight controller should be based on an open-sourced standard that has existing PX4-Autopilot firmware support.
Project ARC Interfacing	R6	The flight controller interfaces must support connections to the critical external components of the Project ARC quadcopter platform.
PCB Hardware Design	R7	The hardware of the flight controller should be designed for cost-effective prototyping with the chosen manufacturer — JLCPCB.

To meet these requirements, two prototypes were developed and are shown in Fig. 5. The prototypes are discussed in further detail in the following sections.

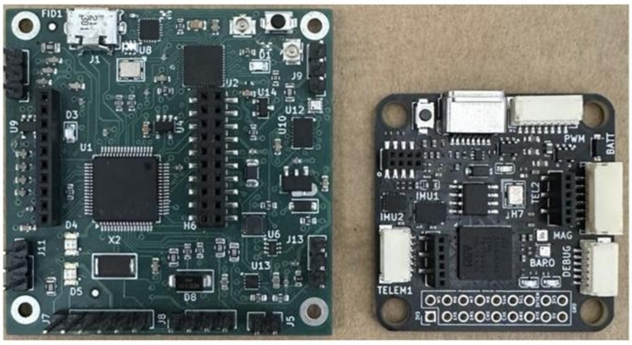
Fig. 5: Prototypes I (left) and II (right) side by side comparison.

3 Prototype I (PW)

The first prototype internally referred to as the Please Work (PW) module, served as the initial solution to evaluate the feasibility of designing a custom embedded system flight controller. To gain hands-on experience with both embedded system design and PX4-Autopilot firmware, Bitcraze products were chosen as the target reference. As outlined in Subsection 1.3.1, Bitcraze provides open-source schematics for their products, including the exact components used, detailed setup circuit configurations, and all the relevant connections. This gives the development important guidelines to create a compliant solution.

Based on the PX4-supported hardware documentation [18], only the Bitcraze Crazyflie 2.1 is categorized as manufacturer-supported. This designation indicates that manufacturers like Bitcraze are responsible for ensuring firmware compatibility when updates or changes are made to different PX4 versions. While the Crazyflie 2.1 can access several PX4-based functions, it is restricted to flying in stabilised mode and lacks access to many modules, due to limited microcontroller computational power and designed all-in-one use case with minimal external components used.

The limited extensibility is a key drawback of the Crazyflie 2.1, as it relies primarily on components manufactured and sold by Bitcraze. The Crazyflie 2.1 employs brushed motors controlled by MOSFETs, which apply a variable voltage across each motor in response to PWM signals.

To meet the design criteria for compatibility across multiple hardware platforms and accommodate ESCs that accept PWM signals, the Crazyflie Bolt 1.1 was selected as the foundation for the design. Since PX4 is configurable, it was assumed that the Crazyflie 2.1-based PX4 firmware could be modified to accommodate these changes.

3.1 Validation

After Prototype I was designed and manufactured, the prototype was subjected to numerous tests to determine the performance limitations of the custom solution. The flight controller was extremely useful in providing insight into the PX4-Autopilot framework as well as gaining experience with configuration and file structure. The flight controller complied with the specifications stemming from the general requirements specified and was able to be operated using QGroundControl as well as MATLAB & Simulink. The initial prototype did, however, have some notable limitations. Some of the key limitations are as follows:

nRF51 Usage and Firmware: Challenges arose during the flashing process of the nRF51 bootloaders and firmware. The nRF51 chip plays a crucial role by putting the microcontroller into DFU mode through the manipulation of the STM32F4's Boot0 pin. Additionally, it oversees the power regulation for the STM32F4. However, in the absence of firmware on the nRF51 chip, the power regulator for the STM32F4 remained inactive. To circumvent this issue, a direct connection was established, giving the STM32's power from the NRF51 regulator.

Partial PX4-Autopilot Support and Limited Extendibility: The Crazyflie 2.1 provides only partial compatibility with PX4, confined to stabilised mode with restricted parameters and onboard modules. Its external radio functionality is similarly limited, as communication depends on the Crazyradio module and the onboard nRF51 chip, facilitated by a Python script called CFBridge. These limitations severely hinder the system's expandability. Additionally, as a manufacturer-supported platform, the Crazyflie does not receive automatic firmware updates for new features. Consequently, when new features are introduced, they may lead to system conflicts if not promptly and properly integrated.

STM32F4 Limitations: A key factor contributing to the limited PX4 application support is the computational limitations of the STM32F4-based microcontroller, particularly when additional modules are deployed. These limitations result in performance constraints, especially when compared to more powerful alternatives.

Given these limitations and the goal of developing a more convenient, PX4-supported, and extendable flight controller, the decision was made to design a new prototype. This prototype aims to address the shortcomings of Prototype I, while enhancing functionality and usability.

4 Prototype II (SWAN)

Prototype II, internally referred to as the Should Work Approximately Now (SWAN) module, represents the second iteration in the development of a flight controller. To address the limitations of the initial prototype, the design was based on the FMU-v6c open standards from Pixhawk. As outlined in Subsection 1.3.2, Pixhawk serves as the reference hardware platform for PX4-Autopilot, with PX4 developers responsible for maintaining compatibility and integrating new features. One notable drawback of adopting these open standards is the lack of detailed documentation beyond pin-outs. Specific information regarding component setup and configuration is not provided, requiring the setup to be determined based on the relevant STM32 micro-controller pin functions and labels. This added complexity necessitated additional effort to acquire the required knowledge from external sources.

Despite these challenges, the decision to utilise Pixhawk FMU-v6c standards rather than Bitcraze was deemed worthwhile, as it offered several significant advantages:

- Provides comprehensive access to all PX4-based functions, enhanced documentation, supported components and numerous configurations.
- Enhances functionality in Simulink, including support for the Connected I/O feature.
- Facilitates off-boarding of communication modules, reducing system size and improving compatibility.
- Incorporates a second redundant IMU and a magnetometer, complementing the existing IMU and barometer from the Prototype I design.
- Utilises a powerful STM32H7-based microcontroller with expanded pins and functionalities for increased versatility.

- Includes a USB 2.0 Type-C connector for improved usability and enhanced durability, minimising the risk of connector damage.

4.1 Validation

Prototype II performed significantly better than the initial solution. PX4 could be flashed using the push button provided and the board was able to be used with QGroundControl and MATLAB & Simulink, with more access to modes and functions.

As a result of the satisfactory performance of the SWAN module and to assess the performance and limitations of the prototype, an additional test was designed. The test involved evaluating a 1-degree-of-freedom (1-DOF) control system. The objective of this test was to compare the performance of the prototype against a well-established commercial flight controller, the Holybro Kakute H7 v1.3 [19]. Additionally, the test seeked to validate the capability to create custom control schemes in Simulink and successfully deploy them onto the hardware.

The stacked design and rig setup are illustrated in Fig. 6. The configuration consists of the ESC as the base layer, followed by the Kakute board, and finally the Prototype II module. Two USB cables facilitate communication, while a modified JST-SH connector links the prototype to the ESC to transmit the relevant PWM signals. The ESC is powered by a 12V power supply through a XT30 connector. A video demonstrating the test and functionality can be found here.

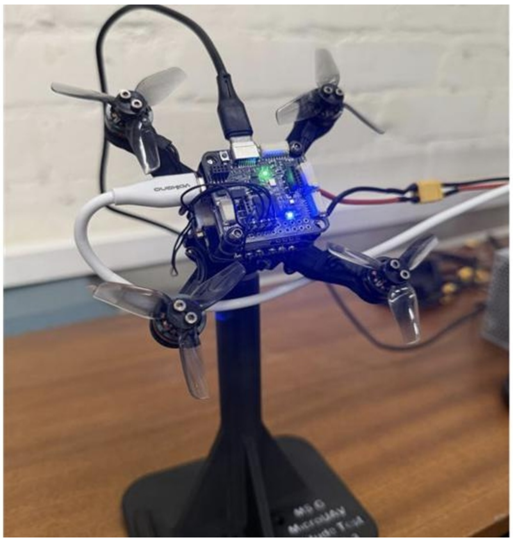

Fig 6: 1-DOF test quadcopter and rig setup, including Prototype II (SWAN) mounted on top of the Kakute H7 and connected to the ESC, with the essential quadcopter components and connectors, placed on the attitude constraining rig.

The test procedure was as follows:

1. A 1-DOF controller was developed in Simulink using the UAV Toolbox Support Package for PX4-Autopilots, using a Proportional-Integral-Derivative (PID) control block.
2. A micro-sized quadcopter was assembled, featuring a custom 3D-printed frame, four TMOTOR F1203 brushless motors [20], a Holybro Tekko32 4-in-1 ESC [21], and a Holybro Kakute H7 flight controller [19].

3. Prototype II was securely mounted on top of the Kakute flight controller and connected to the ESC through a modified JST connector.
4. The quadcopter frame was affixed to a custom 3D-printed rig, designed to constrain rotational motion to either 1 or 3 degrees of freedom whilst preventing translational movement. In this case, the quadcopter was restrained to only permit roll (the rotation of the drone along the defined body x-axis).
5. The flight controllers were calibrated in QGroundControl and thereafter the 1-DOF control system was deployed on both the Kakute and Prototype II controllers.
6. Sensor data and motor responses were collected, overlaid and performance evaluated.

4.1.1 Results

Following the 1-DOF test, using MATLAB the output data was plotted and overlaid for the Kakute and SWAN flight controllers respectively. The control system centres at a target roll angle of 0°, with two step roll angle inputs (10° and -10°) introduced subsecquently. PWM channels 5 through 8 were utilised for motor control, with the control system assigning identical signal values to channels 5 and 6, as well as to channels 7 and 8.

Fig. 7 illustrates the roll angle measured from both flight controllers, revealing a strong similarity between the transient response from the commercial and custom-built systems. This consistency is further evident in Fig. 8, which shows the PWM signal outputs for channels 5 and 6 across both systems. The PWM output is presented using on-time in miliseconds (ms) at a 400Hz frequency. Channels 7 and 8 exhibited similar results. The Root Mean Square Error (RMSE) for the roll angle was calculated to be 0.799°, indicating a relatively low difference as a typical IMU (such as the BMI088 [22]) has orintation errors of approximately 0.5°. Addtionally, there was significant alignment offsets for each controller in the stacked configuration. The PWM signal RMSE was measured at 0.061 ms for channels 5 and 6, and 0.035 ms for channels 7 and 8, respectively. The offset is mainly a result of the small difference in measured roll angle which is amplified by the control system. However, despite the offset the transient response demonstrates significant alignement.

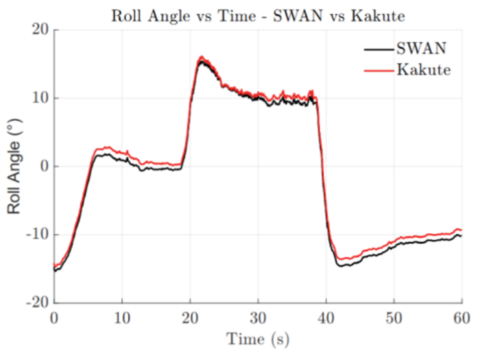

Fig 7: Graph showing roll angle (◦) vs time (s) the dynamic test of the Kakute (red) and SWAN (black) controllers.

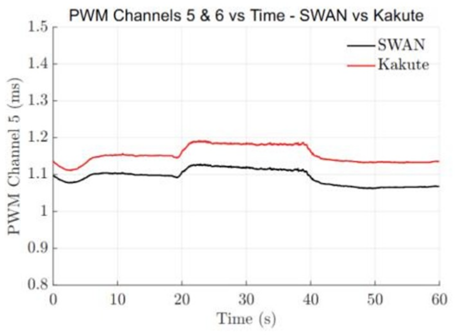

Fig 8: Graph showing the PWM channel 5 & 6 for value(ms) vs time(s) for the dynamic test of the Kakute (red) and SWAN (black) controllers.

5 Infrared heat test

Using a TELOPS M350 TEL-5336 infrared camera ([23]), a thermal test was done to compare Prototype II to the commercial Kakute flight controller. The thermal test (front board images given in Fig 9. & Fig 10.) showed that the SWAN prototype reached maximum surface temperatures of 64.8 °C (front) and 55.2 °C (back). These temperatures are well within the STM32H7's safe operating limits [24]. In comparison, the Kakute H7 operated at lower peak temperatures (50.6 °C and 44.6 °C) with more uniform heat distribution, likely due to its insulating coating and double-sided PCB assembly, indicating superior thermal efficiency.

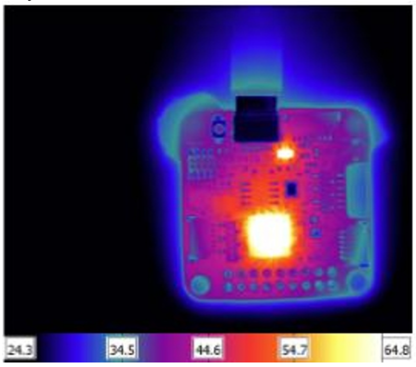

Fig 9: Front infrared image of the SWAN prototype flight controller.

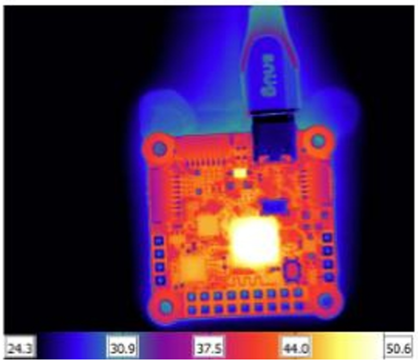

Fig 10: Front infrared image of the Kakute H7 flight controller.

6 Cost analysis

The cost data presented in Table 1 demonstrates the significant cost-saving potential of each developed prototype. Specifically, the manufacturing costs of the two prototypes were compared against their respective commercial counterparts — the Crazyflie Bolt 1.1 ($205) and the Pixhawk 6C ($140.99), respectively. The comparison was initially based on a batch of five manufactured boards without shipping and other custom duties. However, given that a substantial portion of the cost arises from PCB development setup, an additional analysis based on a batch of 25 boards was conducted to further illustrate the potential for cost reduction through scaled production.

Table 2. Cost comparison between Prototype I & II with their respective commercial alternatives.

	Prototype I (PWM)	Cost Saving vs Crazyflie Bolt 1.1	Prototype II (SWAN)	Cost Saving vs Pixhawk 6C
Cost for 5 boards ($)	$242.45	-$782.55	$232.42	-$472.53
Cost per board ($)	$48.49	-$156.51	$46.48	-$94.51
% Cost of Commercial Solution for 5 boards	23.65%	-76.35%	32.97%	-67.03%
% Cost of Commercial Solution for 25 boards	11.60%	-88.40%	10.74%	-89.26%

Table 2 highlights the potential of a custom solution to achieve significant cost savings compared to the open-source commercial platform. Prototypes 1&2 can be manufactured at only 11.6% and 10.74% of the cost of the commercial platform, respectively, when producing 25 units. This substantial reduction in cost aims to support the widespread access of the project by enhancing accessibility to affordable UAV research platforms. Notably, there are significant overheads associated with providing commercially compliant systems which the custom-built controller avoids. The shown reductions therefore aim to highlight the cost advantage of customisable hardware for in-house research purposes and not for providing a commercially compliant system.

7 Conclusion

This research successfully met its objectives by reviewing flight controller fundamentals, evaluating existing designs, and analysing open-source firmware to guide development decisions. By selecting an appropriate open hardware standard, two prototypes were developed: the first provided practical insights into PX4-Autopilot architecture and served as an introduction to flight control system design, while the second demonstrated a fully custom, adaptable flight controller capable of accommodating specific use cases, physical constraints, connector standards, and sensor configurations. Prototype II achieved significant cost savings compared with commercial alternatives without compromising on performance or compatibility. Together, the prototypes illustrate the feasibility, robustness, and scalability of custom UAV systems, highlighting their potential to support accessible innovation and serve as a foundation for future research and development in UAV technologies.

References

1. A. M. Ychpc, Flight controller wiring for drones, LinkedIn (2025). https://www.linkedin.com/pulse/flight-controller-wiring-drones-amuthesh-m-ychpc/
2. Apache Software Foundation, NuttX, Apache (2025). https://nuttx.apache.org/
3. Dronecode Foundation, The Dronecode Foundation – setting standards in the drone industry with open-source, Dronecode (2025). https://dronecode.org/
4. PX4, Open source autopilot for drones, PX4 (2025). https://px4.io/
5. PX4, PX4 architectural overview, PX4 Documentation (2025). https://docs.px4.io/main/en/concept/architecture.html
6. ArduPilot, ArduPilot, ArduPilot (2025). https://ardupilot.org
7. Betaflight, Betaflight – pushing the limits of UAV performance, Betaflight (2025). https://betaflight.com/
8. QGroundControl, QGroundControl user guide, QGC Documentation (2025). https://docs.qgroundcontrol.com/master/en/qgc-user-guide/index.html
9. MathWorks, Integration with general PX4 architecture, MathWorks (2025). https://www.mathworks.com/help/uav/px4/ug/px4-capabilities-integration.html
10. MathWorks, Communicate with hardware using Connected I/O, MathWorks (2025). https://www.mathworks.com/help/uav/px4/ug/px4-connected-io.html
11. MathWorks, Monitor and tune the model running on PX4 autopilots, MathWorks (2025). https://www.mathworks.com/help/uav/px4/ug/monitor-and-tune-px4.html
12. Bitcraze, License, Bitcraze (2025). https://www.bitcraze.io/license/
13. Bitcraze, Crazyflie 2.1+, Bitcraze (2025). https://www.bitcraze.io/products/crazyflie-2-1-plus/

14. Bitcraze, Crazyflie Bolt 1.1, Bitcraze (2025). https://www.bitcraze.io/products/crazyflie-bolt-1-1/
15. Pixhawk Standards Group, Pixhawk/Pixhawk-Standards, GitHub (2025). https://github.com/pixhawk/Pixhawk-Standards
16. Holybro, Pixhawk 6C, Holybro (2025). https://holybro.com/products/pixhawk6c
17. Pixhawk Standards Group, Pixhawk v6C autopilot standard, GitHub (2025). https://github.com/pixhawk/Pixhawk-Standards/blob/master/DS-018%20Pixhawk%20Autopilot%20v6C%20Standard.pdf
18. PX4, Flight controller (autopilot) hardware, PX4 Documentation (2025). https://docs.px4.io/main/en/flight_controller/
19. Holybro, Kakute H7 v1.3 (MPU6000), Holybro (2025). https://holybro.com/products/kakute-h7
20. T-Motor, TMOTOR F1203 brushless motor for 2–3 inch FPV drones, T-Motor (2025). https://shop.tmotor.com/products/f1203-brushless-motor-for-fpv-drones
21. Holybro, Tekko32 F4 4in1 50A ESC (AM32), Holybro (2025). https://holybro.com/products/tekko32-f4-4in1-50a-esc
22. Bosch Sensortec, Inertial measurement unit BMI088, Bosch (2025). https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi088/
23. Exosens, *High-Performance FAST Cameras*, Exosens (2025) https://www.exosens.com/products/high-performance-fast-cameras/
24. STMicroelectronics, *STM32H7 – Arm Cortex-M7 and Cortex-M4 MCUs (480 MHz)*, STMicroelectronics (2025) https://www.st.com/en/microcontrollers-microprocessors/stm32h7-series.html

8 Appendix:

8.1 User needs analysis

8.1.1 Introduction

As outlined in the introduction, the Mechatronic Systems Group, within the UCT Department of Mechanical Engineering, is collaborating with the CSIR Centre for Robotics and Future Production (CRFP) under the Aerial Robotics Capability Project (Project ARC). This initiative focuses on creating an affordable and sustainable in-house quadcopter for research and industrial use. The platform is specifically tailored to target users who currently lack sufficient access to quadcopter components, thereby opening new opportunities for exposure and innovation within the field of UAV research. An important goal of Project ARC is balancing the need for a low-cost quadcopter platform with the ability to provide sufficient performance to support the quadcopters' user requirements.

As a subdivision of Project ARC, the development of the flight controller embodies the project's guiding principles for a balance between performance and affordability. Design decisions for the flight controller aim to be aligned with the overarching goals of the project, ensuring consistency across the entire system's development.

The initial development of the flight controller began with a thorough examination of existing research and findings, presented in the literature review. The next important development stage involves a detailed user needs analysis. The user needs analysis is a study for identifying critical use cases and thereafter deriving requirements for the system. The requirements aim to ensure that the flight controller not only meets all the objectives of Project ARC but also supports important use cases and possible extensibility to multiple platforms. The user needs analysis study is categorized into: key considerations into desired firmware and software functionality, design reference standards, Project ARC interfacing, and hardware design specifications.

8.1.2 Firmware and software compliance

A critical design decision focused on selecting the target firmware, which needed to be determined before considering other design aspects. This choice is fundamental as it dictates the open hardware standards to adopt, ensuring compatibility with the manufacturing process. The goal is to achieve compatibility with a wide range of existing functionalities and applications, encouraging broader user adoption.

Community-driven or open-source firmware provides significant integration potential within an ecosystem of existing applications and software. This accessibility greatly benefits research, as it substantially accelerates the development process. This acceleration is especially evident in the use of graphical programming platforms such as MATLAB & Simulink as the development pathway. These software tools offer extensive advantages for designing complex UAV algorithms, thanks to their robust pre-built toolboxes and function blocks. These functions have the ability to read sensor data and output PWM signals essential for control system design. Furthermore, the MATLAB & Simulink platforms are already well established within the MechatronicSystems.Group, with numerous research outputs utilizing them as the primary development pathway.

Currently, PX4-Autopilot (PX4) is the only firmware directly supported by MATLAB & Simulink through the UAV Toolbox Support Package for PX4-Autopilots. However, the toolbox is not the sole development pathway for PX4, as it also supports alternative approaches, including ROS and MAVLink-based applications, Moreover, as a project under the DroneCode Foundation, PX4 offers extensive compatibility with other DroneCode initiatives. This includes a wide range of compatible hardware and software platforms that enable large-scale comparability for development. The attributes mentioned have led to the decision for PX4 the preferred choice as the target firmware for the flight controller.

8.1.3 Flight controller design reference

With PX4 as the chosen firmware, the flight controller hardware design aims to align with PX4 compatibility requirements, ensuring seamless hardware-firmware integration. Consequently, the reference hardware standards for the flight controller were specifically selected from platforms already supporting PX4. This approach facilitates faster prototyping and minimal interfacing development required to access the firmware.

PX4 supports a variety of commercial closed-source and open-source flight controllers. To gain access to PX4 comparable hardware, the flight controller design shall be based on open-source platforms within the PX4 ecosystem. These standards informed the selection of required components and defined the essential pin connections on the microcontroller. More specifically, the open standards provided guidance on:

- All relevant sensors used, specifying the exact model and manufacturer.
- The microcontroller type and pin package configuration.
- All of the functions for the configurable pins of the microcontroller (e.g., USB pin interfaces, debug ports, communication ports, PWM signal outputs).

- Supporting components include various types of external oscillators, connectors, and USB interface protocols.
- Targeted regulated operating power levels of each of the components.

While open standards offer a broad framework, they do not address the detailed design of PCB hardware. This includes mechanical PCB design, precise component placement, and component set-up configurations, such as those for resistors and capacitors. These design specifics are therefore determined by the interfacing and mounting requirements of Project ARC and on the individual component datasheets for optimal operating conditions.

The primary goal of the hardware of the flight controller prototypes was to closely adhere to the chosen hardware reference standards. Any deviations from these standards were carefully documented, with justifications provided for the adjustments. Component placement and the incorporation of additional or substituted components were optimized to enhance system convenience, facilitate debugging, and improve the overall user experience.

8.1.4 Project ARC interfacing

As a sub-project within the Project ARC quadcopter platform, the PX4 based flight controller must be designed and developed to seamlessly integrate with the intended quadcopter peripherals and components. The integration of these external components shows the need for the flight controller to provide a user-friendly set-up process. This involves using standardized connectors with all the necessary pins required for a reliable connection. The interface requirements for PX4-compatible flight controllers and external components are detailed with specifying the necessary port functionalities and communication protocols for each component's connectivity. Placement and type of interfacing ports must therefore be designed to ensure straightforward and efficient user interaction during the quadcopter set-up. Table 3 is a list of potential target components for the Project ARC platform that requires a flight controller interface:

Table 3: Potential components and interface types for the Project ARC quadcopter platform.

Component	Interface Type
Electronic Speed Controllers (ESC) [61]	4 PWM signal inputs
Companion computer [62]	UART serial interface
Telemetry radio [63] or radio receiver [64] (radio receiver can provide telemetry)	UART serial interface
5V battery power module input [65]	Power: 5V & GND

8.1.5 PCB hardware design

The key hardware specifications of the Project ARC quadcopter platform, crucial to the flight controller design, are as follows:

- The quadcopter platform must have a wingspan of no more than 35 cm, including the propellers.
- The platform should weigh under 500 g, excluding any payload.

Beyond the hardware specifications, a more critical consideration for the flight controller is the overall system cost. Project ARC seeks to promote user adoption by offering a cost-effective platform. The goal is to ensure that the total material and component costs for the fully assembled platform does not exceed R5000 ($265.23). Using an existing PX4 open-source commercial flight controller solution would contribute significantly to overall system costs, with the Crazyflie Bolt accounting for 77.4% and the Pixhawk 6C for 55.2% of the

R5000 limit. This emphasizes the necessity of developing a low-cost flight controller to significantly reduce expenses, enabling Project ARC to achieve the R5000 target budget.

The significant cost-saving potential underscores the importance of designing the flight controller hardware with a primary focus on minimizing manufacturing expenses. Given the complexity of the flight controller, which consists of a network of interconnected components, the hardware was designed as a printed circuit board (PCB) with the necessary components soldered onto the physical board. Due to limited resources and time constraints, it was decided that the chosen manufacturer would ideally handle the entire process—printing the desired PCB boards, sourcing the required components, and assembling them. Research into the different PCB fabrication manufacturers that could facilitate the entire production process providers options revealed that JLCPCB currently offers the most cost-effective solution when compared to competitors such as PCBWay given the same input board characteristics.

As previously mentioned, the PCB hardware for the flight controller was designed with a strong focus on minimizing system costs. The design aimed to remain within the lowest possible cost bracket of the chosen manufacturer, without compromising the mechanical design constraints of the Project ARC quadcopter. However, compact flight controller designs offer significant advantages, including reduced weight and enhanced adaptability for various platform sizes. Consequently, the PCB design was optimized to balance size and cost while ensuring reliable connection with all the essential components. Quotations were assessed using JLCPCB's instant quoting system, which incorporated the cost of board fabrication alongside the availability and pricing of the required component stock.