Cairo University
Faculty of Engineering

Computer Engineering

QuadRaptor

Graduation Project Report Submitted

to

Faculty of Engineering, Cairo University

in Partial Fulfillment of the requirements of the degree

of

Bachelor of Science in Computer Engineering.

Presented by

Abdelrahman Mohamed Salem

Ahmed Fawzy Mohamed

Abdelrahman Mohamed Fathy

Mohab Mohamed Zaghloul

Supervised by

Prof. Amr Wassal

Wednesday 10th JULY 2024

Abstract

Current Drone flight controllers offer poor value for money exceeding the underlying hardware cost which increased the need for democratizing drone technology by designing a cost-effective drone controller.

Exposing the powers of the open-source RISC-V architecture MCUs versus the commonly used Arm architecture MCUs encouraged decreasing the cost of a drone controller.

Designing and testing a drone controller PCB included for example not limited to:

- KiCad (PCB design software)
- Oscilloscope(examining signal integrity)
- logic analyzer(validating signal sequence communication on PCB)
- microscope (soldering)
- Matlab and Simulink (simulation to extract initial PID parameters)
- freeRTOS (open-source Real-time operating system)
- Mrs (useful IDE for debugging WCH MCUs)
- freeCAD (designing utility mechanical 3D parts)
- UltiMaker Cura (printing drone body)
- Multimeter (continuity test, voltage readings)
- Soldering station (hand soldering PCB)
- Vs Code (code editor)

الملخص

توفر وحدات التحكم في الطيران بدون طيار الحالية قيمة سيئة مقابل المال مما تتجاوز تكلفتها الفعلية و اللذى زاد من الحاجة إلى إضفاء الطابع الديمقراطي على تكنولوجيا الطائرات بدون طيار من خلال تصميم وحدة تحكم بدون طيار فعالة من حيث التكلفة.

إن الكشف عن صلاحيات وحدات MCU ذات بنية RISC-V مفتوحة المصدر مقابل وحدات MCU ذات بنية Arm شائعة الاستخدام شجع على تقليل تكلفة وحدة التحكم في الطائرة بدون طيار.

تصميم واختبار PCB لوحدة تحكم الطائرة بدون طيار متضمنة على سبيل المثال لا الحصر:

KiCad (برنامج تصميم للوحات الاكترونية)

راسم الذبذبات (فحص سلامة الإشارة)

محلل منطقي (التحقق من صحة اتصال تسلسل الإشارة على ثنائي الفينيل متعدد الكلور)

المجهر (لحام)

Matlab وSimulink (محاكاة لاستخراج معلمات PID الأولية)

freeRTOS (نظام تشغيل في الوقت الحقيقي مفتوح المصدر)

Mrs (بيئة تطوير متكاملة من اجل وحدات ال WCH)

freeCAD (تصميم الأجزاء الميكانيكية ثلاثية الأبعاد)

UltiMaker Cura (طباعة جسم الطائرة بدون طيار)

جهاز قياس متعدد (اختبار الاستمرارية، قراءات الجهد)

محطة لحام (لحام يدوي للوحات الالكترونية)

Vs Code (محرر الأكواد)

ACKNOWLEDGMENT

- We would like to express our deepest gratitude to our supervisor, Professor Amr Wassal, for his invaluable guidance and support throughout our graduation project, QuadRaptor. His expertise and encouragement have been instrumental in the successful completion of our work. Thank you for your unwavering dedication and for inspiring us to achieve our best.
- **Carbon Aeronautics,** Life long passion for DIY, open-source projects and education. For hobbyist, engineers and drone enthusiasts.
- **Tim Hanewich,** designer of ‘The Scout Flight Controller’.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADC	analog to digital converter.
App	Application.
BEC	Battery-elimination circuit.
BJT	Bipolar junction transistor.
CTS	Clear To Send (the receiver resets this pin to indicate it’s available to receive).
DAC	digital to analog converter.
DOF	Degrees of Freedom.
ESC	Electronic speed control.
GPIO	General purpose input output.
HAL	Hardware abstraction layer.
I2C	Inter-Integrated Circuit.
IC	integrated circuit.
IDE	Integrated development kit.
IMU	Inertial Measurement Unit.
LDO	low Drop-out regulator.
LED	light emitting diode.
LIPO	Lithium polymer battery.
MCAL	Microcontroller abstraction layer.
MCU	Microcontroller unit.
ms	Milli-seconds.
PID	Proportional–Integral–Derivative controller.
RC	Remote Control.
RCC	Reset clock control.
RF	Radio frequency.

RTOS	Real-time operating system.
RTS	Request to Send (the sender reads this pin to check if the receiver is available or not).
SOC	System on Chip.
SOM	System on Module.
SPI	Serial Peripheral Interface.
UART	Universal Asynchronous Receiver/Transmitter.
us	Micro-seconds.

List of Definitions

Decoupling Capacitor	a capacitor inserted between VDD and GND which acts as path short circuit path High frequence noise on the VDD supply.
ESC	a motor driver used to drive btushless dc motors.
F450	a famous open-source body type of drone.
Pitch	angle of rotation around x-axis.
Roll	angle of rotation around y-axis.
thrust	amount of increase of motor speeds.
yaw	angle of rotation around z-axis.

Contacts

Team Members

Name	Email	Phone Number
Abdelrahman Mohamed Salem	abdelrahman.ms.hassan@gmail.com	+2 0112 278 6504
Abdelrahman Mohamed Fathi	fathi791112@gmail.com	+2 0111 858 5574
Ahmed Fawzy Mohamed	afawz6145@gmail.com	+2 0111 488 7199
Mohab Mohamed Zaghloul	mohabmohamedmohamedzaghloul@gmail.com	+2 0102 307 1155

Table 0.1 - team members contact

Supervisor

Name	Email	Number
Prof. Amr Wassal	wassal@eng.cu.edu.eg	+2 0100 871 5559

Table 0.2 - supervisor contact

This page is left intentionally empty

Chapter 1: Introduction

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have rapidly advanced in recent years, becoming pivotal in various sectors including surveillance, agriculture, delivery services, and recreational activities. At the heart of every drone is its flight controller, a critical component responsible for the vehicle's stability, navigation, and overall flight dynamics. The increasing demand for more efficient, reliable, and cost-effective flight controllers has led to significant research and development in this area.

Our project aims to design and implement a custom flight controller for a quadcopter, leveraging the capabilities of the RISC-V architecture through the CH32V203C8T6 microcontroller. This project is motivated by the growing need for open-source and flexible flight control solutions that can be tailored to specific applications and requirements. By utilizing the RISC-V microcontroller, we aim to achieve a high-performance, scalable, and cost-effective flight control system.

1.1. Motivation and Justification

The rapid advancement in drone technology has unlocked numerous opportunities across various industries, including agriculture, surveillance, delivery services, and disaster management. However, the accessibility and affordability of high-quality drone components, particularly flight controllers, remain significant barriers for widespread adoption and innovation, especially in regions with limited technological resources.

Our project addresses the critical need for cost-effective and locally developed flight controllers, driven by the following motivations:

1. **Democratizing Drone Technology**: The current landscape of drone technology is dominated by proprietary systems that are often expensive and inflexible. By developing an open-source flight controller based on the RISC-V microcontroller, we aim to democratize access to advanced drone technology, enabling more individuals and organizations to innovate and utilize drones for diverse applications.
2. **Cost-Effectiveness:** High costs associated with commercial flight controllers can be prohibitive, particularly for educational institutions, startups, and hobbyists. Our project focuses on creating a budget-friendly alternative without compromising on performance or reliability. This approach ensures that more people can afford to experiment with and deploy drones in various fields.
3. **Local Development:** Local Development and Customization: In Egypt, there are currently no factories dedicated to producing drones, which results in a heavy reliance on imported technology. By building and developing flight controllers locally, we can significantly reduce this

dependency. This local production allows for customization to meet specific regional needs and challenges, fostering technological self-reliance. Additionally, it encourages the growth of a skilled workforce capable of designing, maintaining, and advancing these systems, thereby contributing to the country's technological development and innovation.

1.2. Project Objectives and Problem Definition

The primary goal of our project is to develop a cost-effective and high-performance flight controller for quadcopters using the RISC-V microcontroller CH32V203C8T6. This project aims to address the need for more accessible and customizable drone technology, especially in regions like Egypt where there are no local production facilities for drones. Our objectives are designed to tackle the challenges and motivations highlighted in the previous section

1. **Project Objectives:**

    - **Design and Development**: To design a custom flight controller from scratch, leveraging the RISC-V architecture to ensure high performance and flexibility.
    - **Cost-Effectiveness**: To produce a flight controller that offers optimal cost-performance, making advanced drone technology more accessible to a wider audience, including educational institutions, startups, and hobbyists.
    - **Local Development**: To foster local expertise in drone technology by developing and maintaining flight controllers within Egypt, reducing dependency on imported systems and encouraging technological self-reliance.
    - **Library Development**: To develop customized libraries for sensors, transceivers, and motors tailored specifically for the CH32V203 RISC-V microcontroller, ensuring seamless integration and optimal performance.
2. **Problem Definition:**
   Current drone flight controllers often provide poor value for money, with their costs significantly exceeding the underlying hardware expenses. This economic barrier limits the widespread adoption and innovation of drone technology, particularly in regions lacking local production facilities. Our project aims to develop a flight controller that strikes an optimal balance between cost and performance by utilizing customized libraries for sensors, transceivers, and motors. This approach will democratize current drone technology and make it more accessible and customizable for various applications.

1.3. Project Outcomes

The main outcome of our project is the development of a cost-effective, open-source flight controller for quadcopters based on the CH32V203C8T6 RISC-V microcontroller. This achievement addresses the need for affordable and customizable drone technology, particularly in regions like Egypt where local production facilities for drones are absent. The key outcomes of our project include:

1. **Open-Source Flight Controller**: We have successfully designed and implemented a high-performance flight controller tailored for quadcopters. This controller leverages the CH32V203C8T6 microcontroller, providing a robust and flexible solution that can be customized for various applications.
2. **Customized Sensor Libraries**: We developed open-source software packages for several sensors, including the MPU-6050 (accelerometer and gyroscope), HMC5883L (magnetometer), BMP280 (barometric pressure sensor), GY-87 (multi-sensor module), ADXL345 (accelerometer), and an ultrasonic sensor for measuring vertical distance. These libraries are specifically tailored for seamless integration with the CH32V203C8T6 microcontroller.
3. **NRF Transceiver Library**: An open-source package was developed for the NRF transceiver, enabling reliable wireless communication between the application board and the NRF24L01 Arduino RC Remote Radio Transmitter with a pre-programmed SMD ATMEGA328. This facilitates robust remote control and data transmission.
4. **PCB Design**: A printed circuit board (PCB) was designed for the CH32V203C8T6 microcontroller, providing a compact and efficient layout that supports the various components and peripherals required for the flight controller.
5. **Open-Source Availability**: All developed libraries and software packages are made available as open-source resources. This ensures that other developers and enthusiasts can download, use, and integrate these packages into their own applications utilizing the CH32V203C8T6 microcontroller.
6. **Educational and Technological Impact**: By providing a comprehensive, open-source flight controller and associated libraries, our project fosters learning and innovation. It offers educational institutions, startups, and hobbyists the tools needed to explore and develop UAV technology without significant financial barriers.

These outcomes collectively contribute to the democratization of drone technology, making advanced UAV systems more accessible and customizable. They also promote local technological development and expertise, particularly in regions where such resources are limited.

1.4. Document Organization

The document is organized as follows:

- **Chapter 1: Introduction**

  Provides an overview of the project, including the motivation and justification for developing the QuadRaptor flight controller, the project objectives, problem definition, and expected outcomes.

- **Chapter 2: Market Visibility Study**

  Discusses the global drone market, target customers, and conducts a market survey. This chapter also includes a business case and financial analysis to justify the project's viability.

- **Chapter 3: Literature Survey**

  Reviews existing research and developments in drone technology, focusing on RISC-V microcontrollers and drone flight dynamics. It includes a comparative study of previous work and the approach implemented in this project.

- **Chapter 4: System Design and Architecture**

  System Design and Architecture The system design and architecture chapter provides an overview and assumptions, presents the system architecture with a block diagram, and details the modules involved. Each module includes functional descriptions, modular decomposition, design constraints, and other relevant information.

- **Chapter 5: System Testing and Verification**

  Covers the testing and verification process of the flight controller, including testing setup, plan, strategy, and results. It discusses module testing, integration testing, and flight tests.

- **Chapter 6: Conclusions and Future Work**

  Summarizes the project's outcomes, challenges faced, and the experience gained. It also outlines potential future work to enhance the flight control system further.

- **References**

  Lists the references used throughout the document for supporting information and further reading.

- **Appendices**

  Provides additional information and supporting documents such as development platforms and tools, use cases, user guides, code documentation, and feasibility studies.

Chapter 2: Market Visibility Study

The global drone market has experienced rapid growth over the past decade, driven by advancements in technology and expanding applications across various industries such as agriculture, surveillance, military services, logistics, and recreational activities. Despite the proliferation of drone technology, the market for flight controllers remains dominated by a few key players who offer proprietary solutions that are often expensive and inflexible. This chapter surveys the existing market for drone flight controllers, identifying the main competitors and evaluating their offerings. We will discuss the limitations and drawbacks of these existing solutions, which highlight the need for our project—a cost-effective, open-source flight controller based on the CH32V203 RISC-V microcontroller.

2.1. Targeted Customers

The development of our cost-effective, open-source flight controller targets a diverse range of customers, each benefiting uniquely from the affordability, flexibility, and customizability of our solution. The key intended customers include:

- **Educational Institutions**: Schools, colleges, and universities can integrate our flight controller into their curricula. The open-source nature of the project allows educators to tailor the flight controller for specific educational goals, fostering a deeper understanding of UAV systems.
- **Startups and Entrepreneurs**: Small businesses and startups in the drone industry can benefit from a cost-effective solution that reduces initial investment costs. Our flight controller enables entrepreneurs to develop and deploy UAV applications across various sectors, including agriculture, logistics, and surveillance, without the burden of high expenses associated with commercial flight controllers.
- **Research and Development Labs**: Research institutions and labs focusing on UAV technology can leverage our flight controller for experimental projects and innovative applications. The ability to customize and modify the controller to meet specific research requirements enhances the potential for breakthroughs in various fields.
- **Hobbyists and DIY Enthusiasts**: Drone hobbyists and DIY enthusiasts often seek affordable and customizable components to build and enhance their UAVs. Our open-source flight controller provides a flexible platform for experimentation and personal projects, encouraging creativity and innovation within the maker community.
- **Agricultural Sector**: Farmers and agricultural service providers can utilize our flight controller to develop drones for precision agriculture, including crop monitoring, irrigation management, and pesticide application.

- **Surveillance and Security Firms**: Companies specializing in surveillance and security can integrate our flight controller into their UAV systems for monitoring and reconnaissance purposes. The reliability and performance of our controller ensure effective and efficient operations, enhancing security measures.

2.2. Market Survey

This section analyzes competitive flight controllers, examining their features, advantages, and limitations to highlight the unique value proposition of our project, QuadRaptor.

2.2.1. Competitive Project 1: AeroFly

Overview: AeroFly is a commercial flight controller designed for professional and hobbyist drone applications. It is based on the STM32 microcontroller architecture, known for its robustness and reliability. AeroFly offers extensive support for various sensors and communication protocols, making it a popular choice for diverse UAV projects.

- **Pros:**
    1. High Reliability: Known for its stable performance in diverse conditions.
    2. Extensive Documentation: Provides detailed manuals and support, making it easier for users to implement and troubleshoot.
    3. Wide Compatibility: Supports a broad range of sensors and peripheral devices.
- **Cons**:
    1. Cost: AeroFly is relatively expensive, which can be a significant barrier for startups, educational institutions, and hobbyists.
    2. Proprietary System: Being a proprietary solution, it lacks the flexibility for customization and adaptation to specific needs.
    3. Complexity: The advanced features and settings can be overwhelming for beginners and may require a steep learning curve.

2.2.2. Competitive Project 2: FlySafe

Overview: FlySafe is a user-friendly flight controller designed for consumer drones. It is based on the AVR microcontroller architecture, which is known for its simplicity and ease of use. FlySafe targets the recreational and semi-professional market, offering integrated features such as GPS and obstacle avoidance, making it an attractive option for casual users.

- **Pros**:
    1. User-Friendly Interface: Designed with an intuitive interface that simplifies setup and operation.
    2. Integrated Features: Includes built-in GPS, return-to-home function, and obstacle avoidance, enhancing user experience.
    3. Affordability: Priced competitively, making it accessible for a broader audience.
- **Cons**:
    1. Limited Customization: The closed-source nature of FlySafe limits its adaptability and customization for specific applications.
    2. Performance Constraints: While suitable for recreational use, it may not meet the performance requirements for professional or research applications.
    3. Support and Updates: Dependence on manufacturer updates and support, which can be a bottleneck for rapid innovation and troubleshooting.

2.2.3. Competitive Project 3: OpenPilot

Overview: OpenPilot is an open-source flight controller project that emphasizes flexibility and community-driven development. It is based on the STM32 microcontroller architecture, which provides a powerful and flexible platform for various applications. OpenPilot is popular among developers and hobbyists for its customizable nature, allowing extensive modifications to suit specific needs.

- **Pros**:
    1. Open-Source: Allows users to modify and customize the software and hardware according to their needs.
    2. Community Support: A vibrant community provides ongoing support, updates, and enhancements.
    3. Cost-Effective: Generally more affordable due to its open-source nature.
- **Cons**:
    1. Documentation: While community-driven, the documentation may not be as comprehensive or consistent as commercial products.
    2. Variable Quality: The open-source nature can lead to variations in quality and performance, depending on the specific implementation.
    3. Learning Curve: Requires a good understanding of hardware and software for effective customization and use.

2.3. Business Case and Financial Analysis

In this section we will describe the potential success of establishing a company to sell our product, QuadRaptor, a cost-effective, open-source flight control system for drones.

1. **Business Case:**

Based on the market survey above, we anticipate significant demand for a cost-effective, customizable flight controller. Our target customers include educational institutions, research labs, startups, hobbyists, and various industries such as agriculture and surveillance. By providing a high-performance, affordable alternative to existing proprietary systems, we expect to capture a substantial market share over the next five years.

To counter competition and attract our target audience, we will set our pricing strategy based on a value proposition that emphasizes affordability, flexibility, and local customization. We will offer competitive prices while ensuring sustainable margins by leveraging cost-effective production and open-source development. We suggest selling the flight controller in the first 6 months for $11.5.

2. **Financial Analysis:**

Based on the business case, we must anticipate the following financial aspects:

A. **Capex (Capital Expenditure)**:
- Initial development costs for the flight controller hardware and software.
- Costs for setting up manufacturing facilities or outsourcing production.
- Purchasing equipment and tools required for production and testing.
- Marketing and promotional costs for the initial product launch.

B. **Opex (Operational Expenses)**:
- Salaries for development, production, and administrative staff.
- Recurring costs for raw materials and components.
- Marketing, advertising, and customer support expenses.
- Maintenance and operational costs for production facilities.

We will create a cash flow table to project our financial performance over the next five years. This table will include monthly capex and opex on separate rows and our revenues from product sales on another set of rows. The difference between the revenues and expenses will represent our profit before tax.

In the initial stages, it is likely that the difference will be negative until our sales volume increases and offsets the expenses. From this cash flow analysis, we will determine the break-even point—the date at which our cumulative revenues equal our cumulative expenses. Achieving the break-even point is a critical milestone, as it marks the transition to generating true profit.

Financial Projections table:

Year	Capex	Opex	Revenue	Profit	Cumulative Profit
1	$50K	$100K	$80K	$-70K	$-70K
2	$20K	$120K	$150K	$10K	$-60K
3	$10K	$130K	$220K	$80K	$20K
4	$10K	$140K	$300K	$150K	$170K
5	$10K	$150K	$400K	$240K	$410K

Table 2.1 - Financial Projections table.

Chapter 3: Literature Survey

This chapter presents the essential engineering and non-engineering background knowledge required for a comprehensive understanding of the cost-effective flight controller project. The chapter begins with an overview of RISC-V microcontrollers, followed by a discussion of the principles of drone flight dynamics and control. A comparative study of previous work on drone flight controllers is then presented, highlighting the key features and limitations of existing solutions. Finally, the implemented approach for the flight controller design is outlined, with a justification for the chosen approach and modifications made to optimize cost-effectiveness.

3.1. Background on RISC-V Microcontrollers

RISC-V is an open-source instruction set architecture (ISA) based on reduced instruction set computing (RISC) principles. This architecture offers several advantages over traditional proprietary ISAs, such as flexibility, modularity, and cost-effectiveness. RISC-V microcontrollers, like the CH32V203 series, leverage these advantages to provide a powerful and affordable platform for embedded systems development.

The CH32V203 microcontroller used in this project is based on the 32-bit RISC-V core, featuring a high-performance pipeline and a wide range of peripherals. Its low power consumption and small footprint make it an ideal choice for drone flight controller applications, where size and weight constraints are critical.

3.2. Background on Drone Flight Dynamics and Control

Drone flight dynamics are governed by the complex interplay of aerodynamic forces, inertial effects, and control inputs. Understanding these dynamics is crucial for designing a stable and responsive flight controller.

The primary control inputs for a drone are throttle, roll, pitch, and yaw. These inputs are translated into motor commands by the flight controller, which in turn generate the necessary forces and moments to manipulate the drone's orientation and position.

Control algorithms, such as PID (Proportional-Integral-Derivative), are employed to maintain stability and achieve desired flight characteristics. These algorithms continuously monitor the drone's state and adjust the control inputs to compensate for disturbances and maintain a stable flight path.

3.3 Comparative Study of Previous Work

The field of drone flight controllers has seen significant advancements over the years, with various approaches and technologies being employed to enhance performance, stability, and cost-effectiveness. This section presents a comparative study of some notable previous works in the domain of drone flight controllers, highlighting their key features, advantages, and limitations.

3.3.1 DJI Naza-M V2

The DJI Naza-M V2 is a highly regarded flight controller known for its robust performance and user-friendly setup. It features a built-in Inertial Measurement Unit (IMU) and barometer, providing accurate flight stabilization. The controller supports multiple flight modes, including GPS Atti mode, Manual mode, and Return-to-Home (RTH) function. However, the Naza-M V2 is relatively expensive, which can be a limiting factor for cost-sensitive applications.

Advantages:

- High stability and precision.
- Multiple flight modes and safety features.
- Comprehensive software support.

Limitations:

- High cost.
- Proprietary hardware and software, limiting customization.

3.3.2 ArduPilot APM 2.8

ArduPilot APM 2.8 is an open-source flight controller widely used in both hobbyist and professional drone applications. It offers extensive customization options and supports various sensors and peripherals. The controller uses a combination of accelerometers, gyroscopes, magnetometers, and barometers for precise flight control. While highly flexible, the APM 2.8 can be complex to configure and may require significant tuning for optimal performance.

Advantages:

- Open-source and highly customizable.
- Supports a wide range of sensors and peripherals.
- Active community and extensive documentation.

Limitations:

- Can be complex to set up and configure.
- Moderate cost.

3.3.3 Pixhawk

Pixhawk is another open-source flight controller that offers advanced features and high performance. It is designed for both fixed-wing and multi-rotor aircraft and includes an IMU, barometer, and magnetometer. Pixhawk supports sophisticated flight control algorithms and is compatible with various ground control software. However, its advanced features come with a higher price tag, making it less accessible for budget-conscious projects.

Advantages:

- Advanced features and high performance.
- Open-source with extensive customization.
- Wide compatibility with sensors and software.

Limitations:

- High cost.
- Complexity in setup and configuration.

3.4 Implemented Approach

The implemented approach for the flight controller design focuses on achieving a balance between performance and cost-effectiveness, tailored to meet the needs of a wide range of drone applications. This section outlines the chosen hardware and software strategies, providing a justification for each decision.

Hardware Selection

The CH32V203 microcontroller was selected as the core of the flight controller due to its combination of low cost, high performance, and versatile peripheral support. The 32-bit RISC-V architecture provides a powerful computational platform, while its low power consumption and small footprint are ideal for drone applications where size and weight are critical constraints.

Justification:

- **Cost-Effectiveness:** The CH32V203 offers a significant cost advantage over other microcontrollers with similar capabilities, making it an economical choice for the project.
- **Performance:** The high-performance pipeline and extensive peripheral set of the CH32V203 ensure that it can handle the computational demands of flight control algorithms.
- **Power Efficiency:** The low power consumption of the microcontroller extends the drone's flight time, which is crucial for practical applications.

Sensor Fusion

A Kalman filter was implemented to fuse data from the gyroscope and accelerometer, providing an accurate estimation of the drone's roll and pitch. The magnetometer is used separately to estimate yaw directly. This approach balances computational simplicity with accuracy for stable flight control.

Justification:

1. **Enhanced Accuracy**: By using the magnetometer to estimate yaw directly, along with the Kalman filter for roll and pitch, the overall orientation estimation is improved.
2. **Simplicity and Performance**: The Kalman filter offers a good trade-off between computational load and accuracy for roll and pitch estimation, leveraging the microcontroller's capabilities efficiently.

Control Algorithm

A simplified PID control algorithm was chosen to maintain the drone's stability and responsiveness. The algorithm continuously adjusts the control inputs based on the drone's state, ensuring smooth and stable flight.

Justification:

- **Computational Efficiency:** The PID algorithm is computationally efficient and well-suited for real-time control applications.
- **Proven Effectiveness:** PID control is a widely used and proven method for maintaining stability in dynamic systems like drones.

Custom Firmware

Custom firmware was developed to optimize the performance of the CH32V203 microcontroller, minimizing power consumption and maximizing responsiveness. This firmware is tailored to the specific needs of the flight controller, ensuring that all resources are utilized effectively.

Justification:

- **Performance Optimization:** Custom firmware allows for fine-tuned optimization of the microcontroller's resources, ensuring peak performance.
- **Power Management:** Tailoring the firmware helps in implementing power-saving features, extending the drone's operational time.

By integrating these hardware and software optimizations, the implemented approach achieves a cost-effective solution for the flight controller while maintaining high performance and reliability. This balance makes the project accessible for a broader range of applications, from hobbyist drones to more advanced UAV systems.

Chapter 4: System Design and Architecture

The system consists of 2 PCB boards, one's responsibility is the stability of the drone (this board is called a 'drone board'), and the other is an application board (called 'App board') whose responsibility depends on the application.

The drone board requires a companion computer whether it is a low-power MCU, SOC, SOM, etc…

Communication between the App board and the Drone is done via UART with hardware flow control (RTS and CTS) at a **115200 baud rate.** The drone board will send a packet every second containing some info.

That way our system is built to make it possible to add a high-power companion computer for various aerial uses which carries off the load of drone stability from the application engineer. The app drone will command the drone board which way to move in and the drone takes full responsibility for drone stability. A dummy App board PCB was made to illustrate our idea in the demo and test the system.

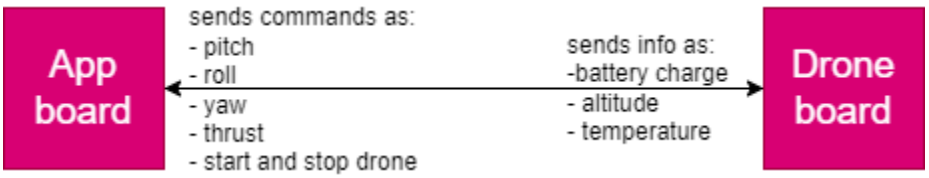


Figure 4.1 - interface between both boards.

The body of the drone was supposed to be made of carbon fiber for its strength and lightweight. But instead, we printed the body using a 3D printer with PLA+ material. The design files were taken from the following URLs:

1- https://www.thingiverse.com/thing:437561

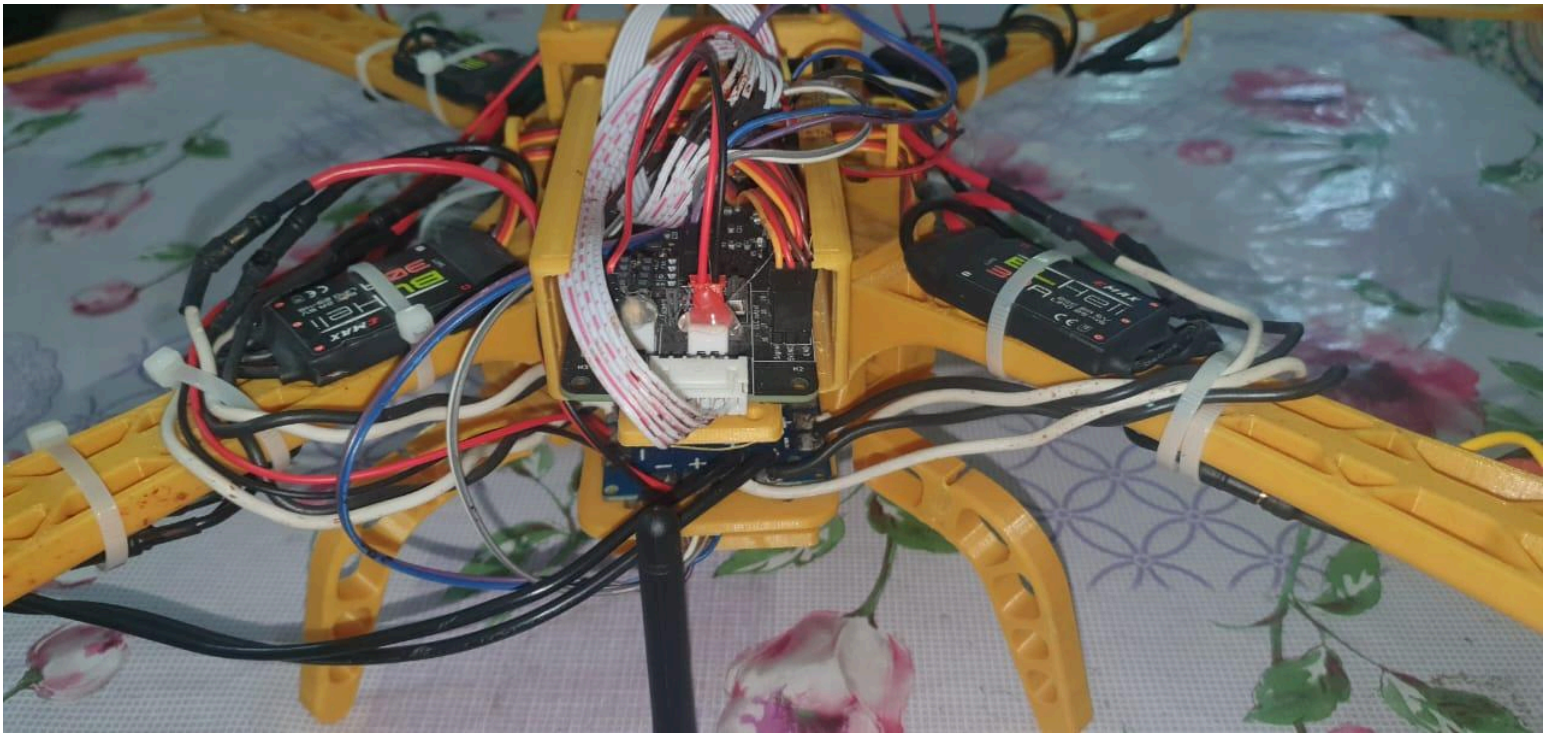
2- https://www.thingiverse.com/thing:1206960

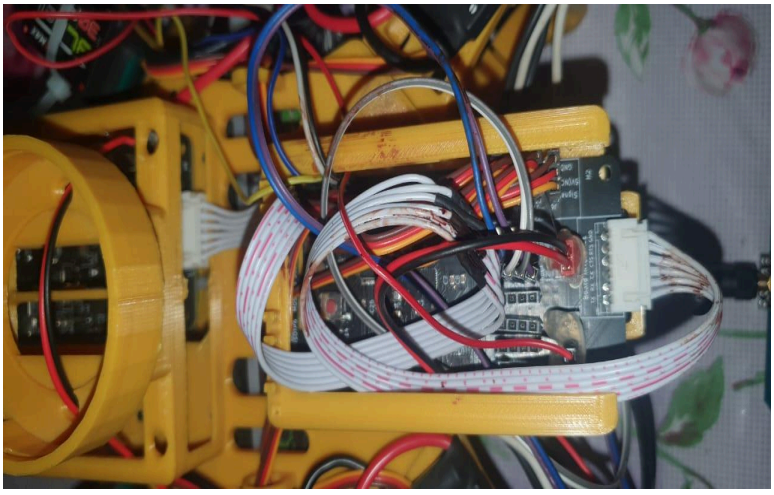
3- https://www.thingiverse.com/thing:5196881

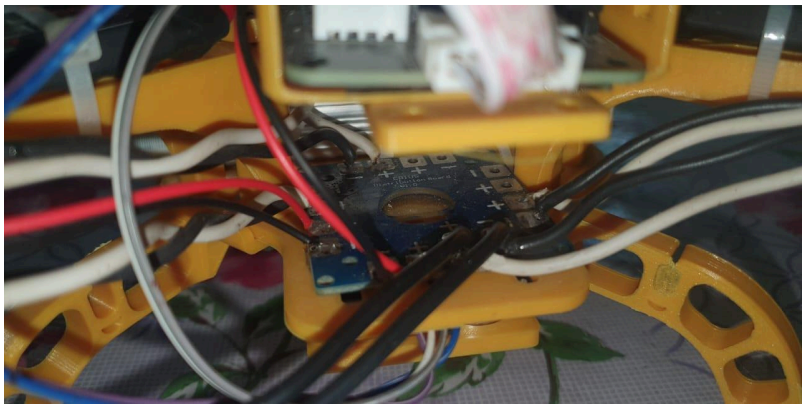
4- https://cults3d.com/en/3d-model/game/f450-flamewheel-pixhawk-fpv-electronics-top-plate-canopy

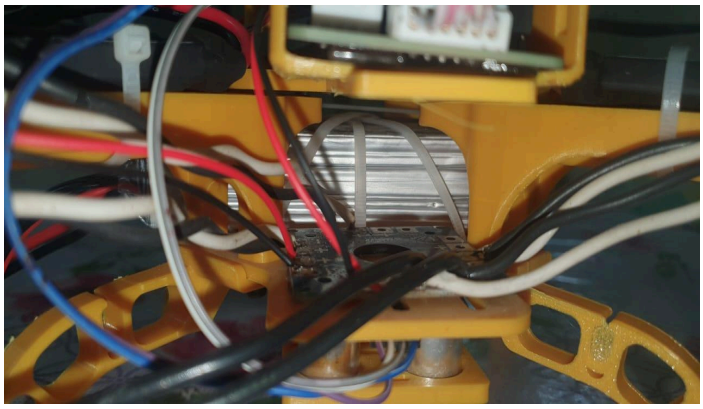
5- https://www.thingiverse.com/thing:2043499

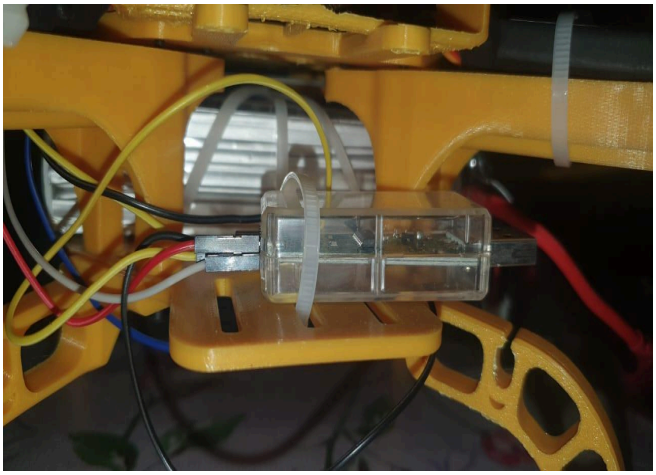
Images of the final drone body can be seen on the next pages.











4.1. Overview and Assumptions

The file structure is as follows:

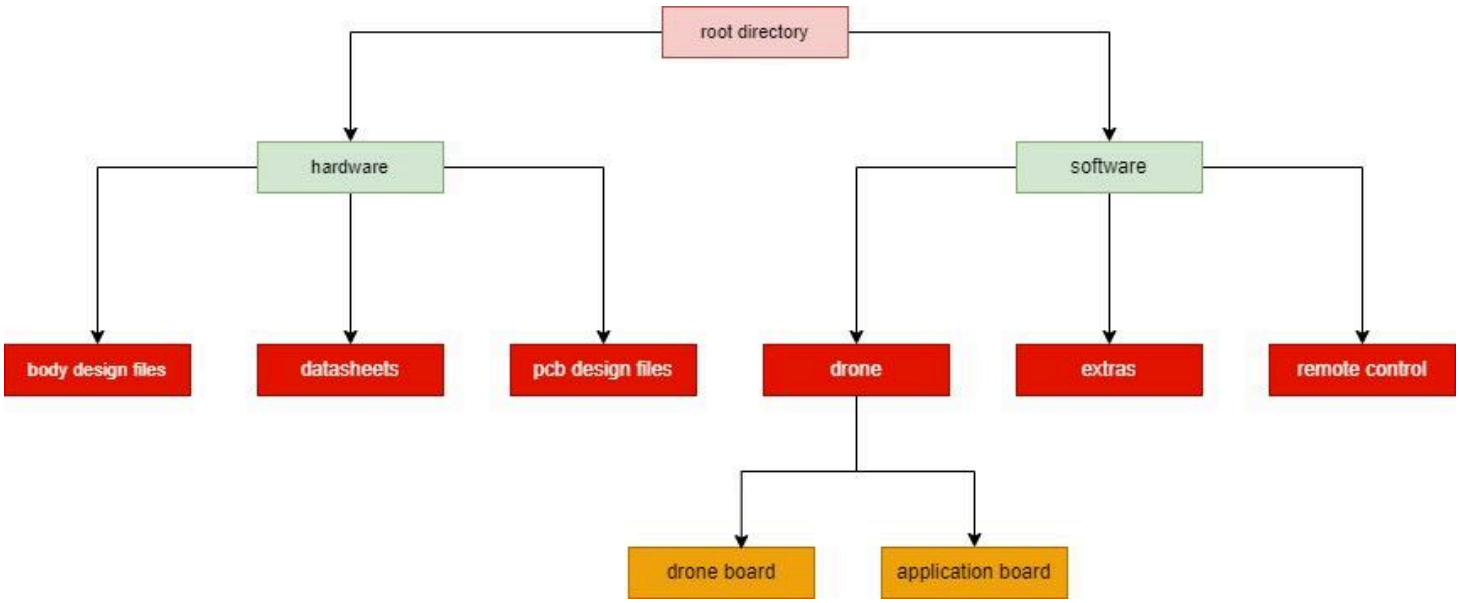


Figure 4.2 - file structure.

Where the importance of each subdirectory is as follows:

root directory	The main directory of the project files.
hardware	Contains all the important files regarding the hardware of things.
software	Contains all the important files regarding the software of things.
body design files	Contains ‘.stl’ files for the drone to be printed (F450 drone).
datasheets	Contains the datasheets of ICs used in both boards.
pcb design files	Contains KiCad PCB design files (schematics & routing) for both boards.
drone	Contains all software that’s running on both example App and Drone boards.
extras	Contains extra tunning scripts, motors equations, analysis of the code, etc…
remote control	Contains Code running on the remote control.
drone board	Contains Code running on the drone board.
application board	Contains Code running on the dummy App board

Table 4.1 - file structure.

4.2. System Architecture

The main 2 subsystems are the app board and drone board the drone board is connected to all kinds of sensors and actuators while the app board system is connected to application-specified hardware where the interface between both systems is via a UART where the responsibility of each system is nearly independent of each other.

4.2.1. Block Diagram

The system architecture is as follows:

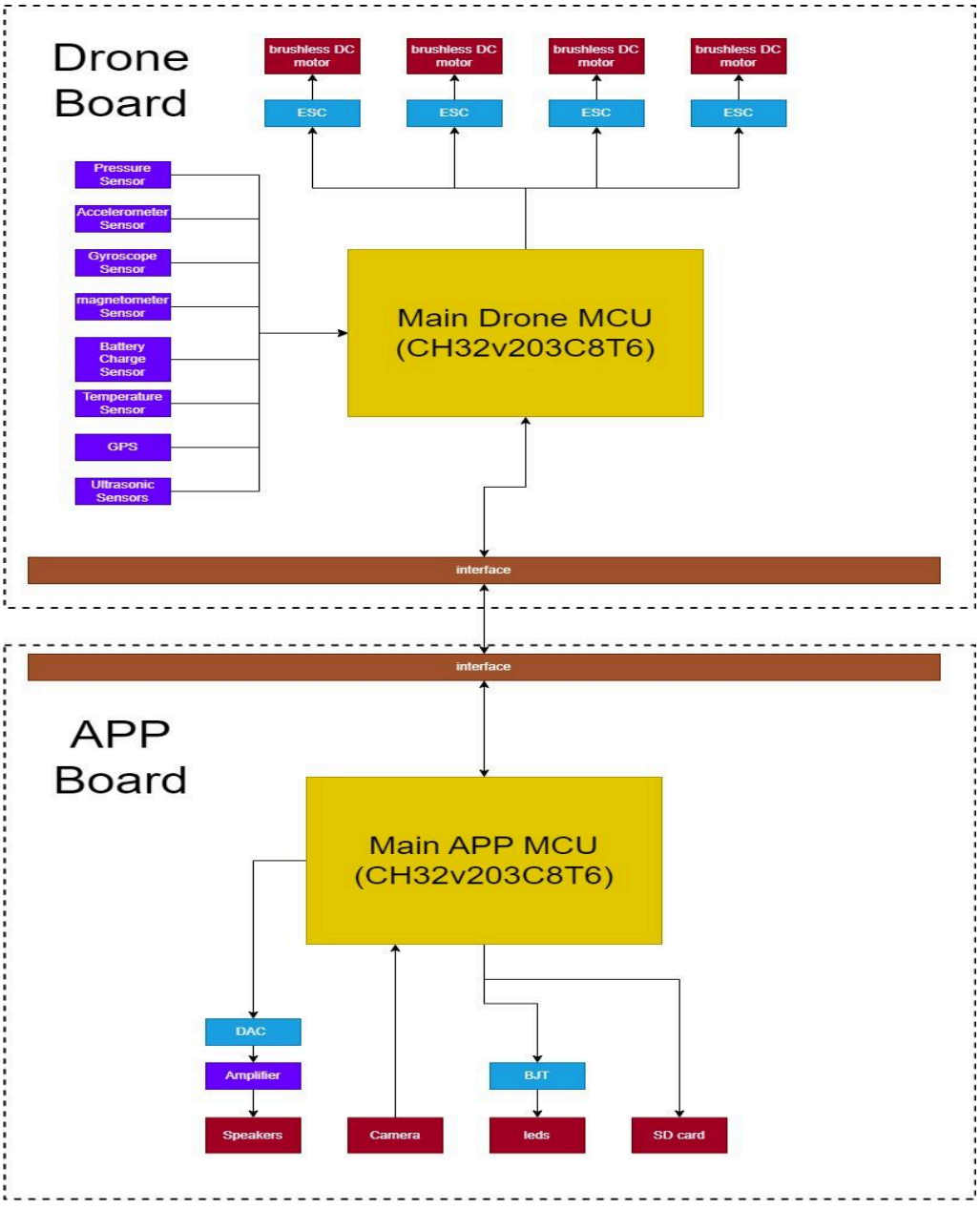


Figure 4.3 - system architecture.

Where the responsibility for each block is as follows

Brushless DC Motor	The main motors in the drone that control the movement of the drone and consume a current of a maximum of ~25A.
ESC	Control motor speed and have a maximum continuous current of around ~30A.
Pressure Sensor	Reads pressure to compute the altitude of the drone which in turn fused with an accelerometer sensor to get vertical velocity.
Accelerometer Sensor	Reads the accelerometer along each axis to be fused with other sensor readings to compute roll, pitch, and yaw angles.
Gyroscope Sensor	Reads rate of change of angles around each axis to be fused with other sensor readings to compute pitch, roll, and yaw angles.
Magnetometer	Reads change in magnetic field to be fused with other sensors to compute yaw angle.
Battery Charge Sensor	Computes approximate battery charge to determine if the drone is able to fly or not.
Temperature Sensor	Compute dummy data which is temperature.
GPS	It gets its current location in the world as a magnetic field will change from one place to another on Earth.
Ultrasonic Sensor	It can be fused with a pressure sensor to get a better altitude reading.
Main Drone MCU	Responsible for taking feedback from Sesnors and applying actions on Motors
DAC, amplifier and Speakers	Application specified and accountable for outputting a Guidable audio.
Camera	Taking videos and storing them on the SD card.
BJT with leds	Flashing LEDs for visual output.
SD card	Store videos/images and for debugging.
Main App MCU	Responsible for application specified and communication with both drone board and remote control.

Table 4.2 - system architecture

Software architecture for the code written on each MCU is as follows:

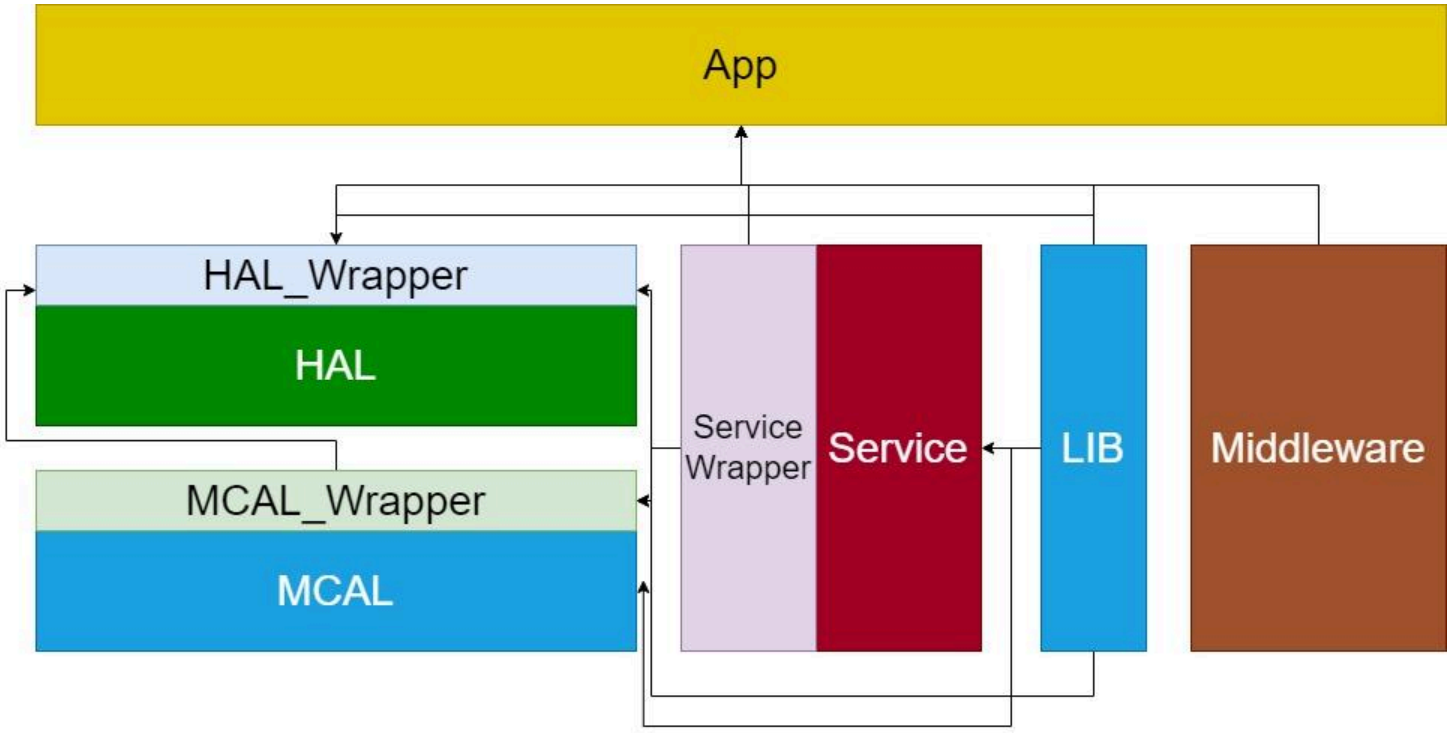


Figure 4.4 - software architecture

Where the functionality of each block is as follows:

HAL_Wrapper	Wrapper interface for the HAL layer so that if we wanted to change the HAL layer, we could do it simply by just changing the implementation of HAL_Wrapper. To overcome the overhead of changing all function calls in the APP layer.
MCAL_Wrapper	Wrapper interface for the MCAL layer so that if we wanted to change the MCAL layer, we could do it simply by just changing the implementation of MCAL_Wrapper. To overcome the overhead of changing all function calls in the HAL layer.
App	The actual code is written in the main file.
HAL	The hardware abstraction layer contains all code related to external hardware in an abstract way.
MCAL	The microcontroller abstraction layer contains all MCU main functionality abstractly.
Service	Contains all service functionality as a Real-time operating system, etc…
Service_Wrapper	Wrapper interface for the Service layer to easily change RTOS.
Lib	Contains common utility functions.
Middleware	It contains common algorithms, PID blocks, Sensor Fusion, etc…

Table 4.3 - software architecture

Each APP layer on each MCU has the RTOS tasks running on it where tasks running on the drone board are as follow:

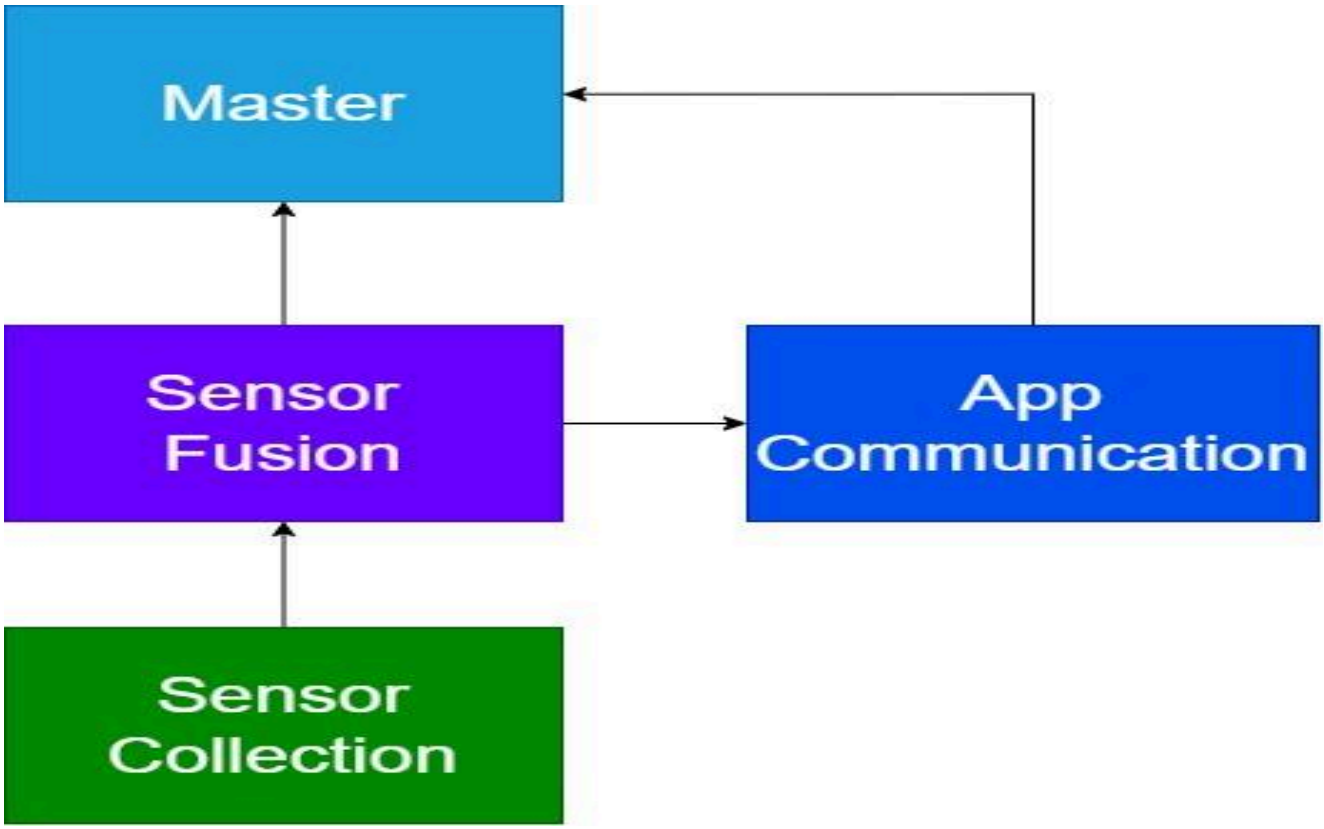


Figure 4.5 - drone board RTOS tasks

Where the responsibility of each task is as follow:

Sensor Collection	Collects raw data measurements from different sensors on the drone board where these readings would be sent later to Sensor Fusion through Queue.
Sensor Fusion	Takes data from Sensor Collection task to fuse it and get a good estimate for the current state. It then sends some infor to App communication task and sends the current state to Master task.
App Communication	Responsible for sending and receiving data from and to the app board like sending info and receiving commands.
Master	The task responsible for taking actions where it take commands from App communication task and estimated states from Sensor Fusion task to fed the error into PID block which later being fed into motor speeds.

Table 4.4 - drone board RTOS tasks

And tasks running on the App board are as follow:

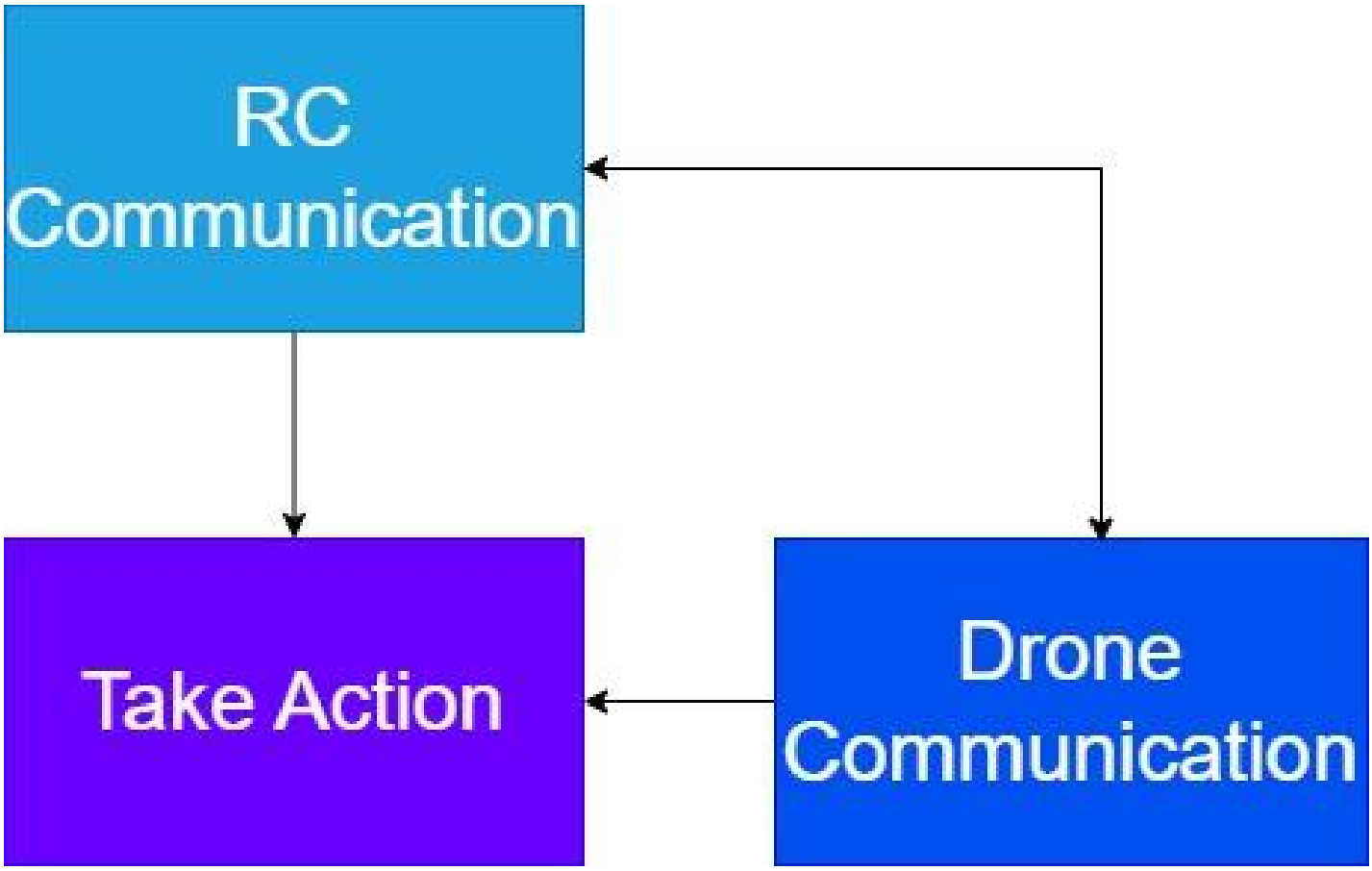


Figure 4.6 - app board RTOS tasks

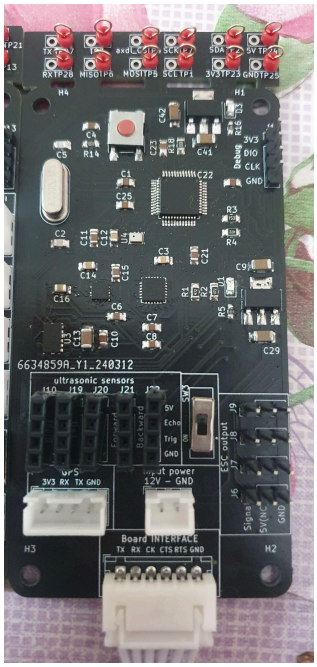
Where the responsibility of each task is as follow:

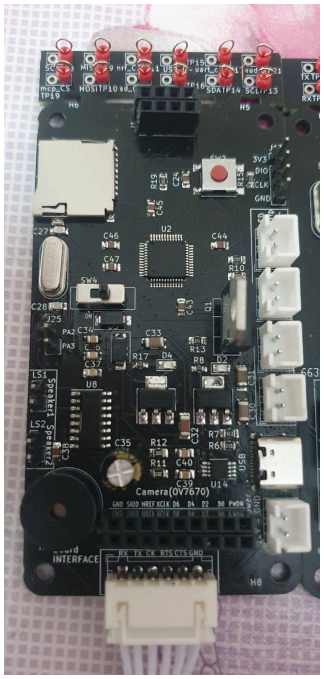
RC Communication	Responsible for communicating with remote control by sending info and taking actions.
Take Action	Not implemented task yet but it was supposed to take some application defined tasks.
Drone Communication	Responsible for sending and receiving data from and to the drone board like sending commands and receiving info.

Table 4.5 - app board RTOS tasks

4.3. PCB design Module

We had to keep in mind the mechanical dimensions of the drone body. Regarding IC selection, we choose RISC-V MCU from WCH for its cost-effective price with comprehensive documentation. For other sensors, we chose the most common ICs in the Egyptian market for their popularity. Adding Test points to the PCB was critical to debug the board. JLCPCB manufactured the PCB where the design was done by JLCPCB and soldering was done by us using a microscope from Amscope and a soldering station from Kada.





4.3.1. Functional Description

Designing of both PCBs was done on KiCad where the schematic of the whole design is as follows:

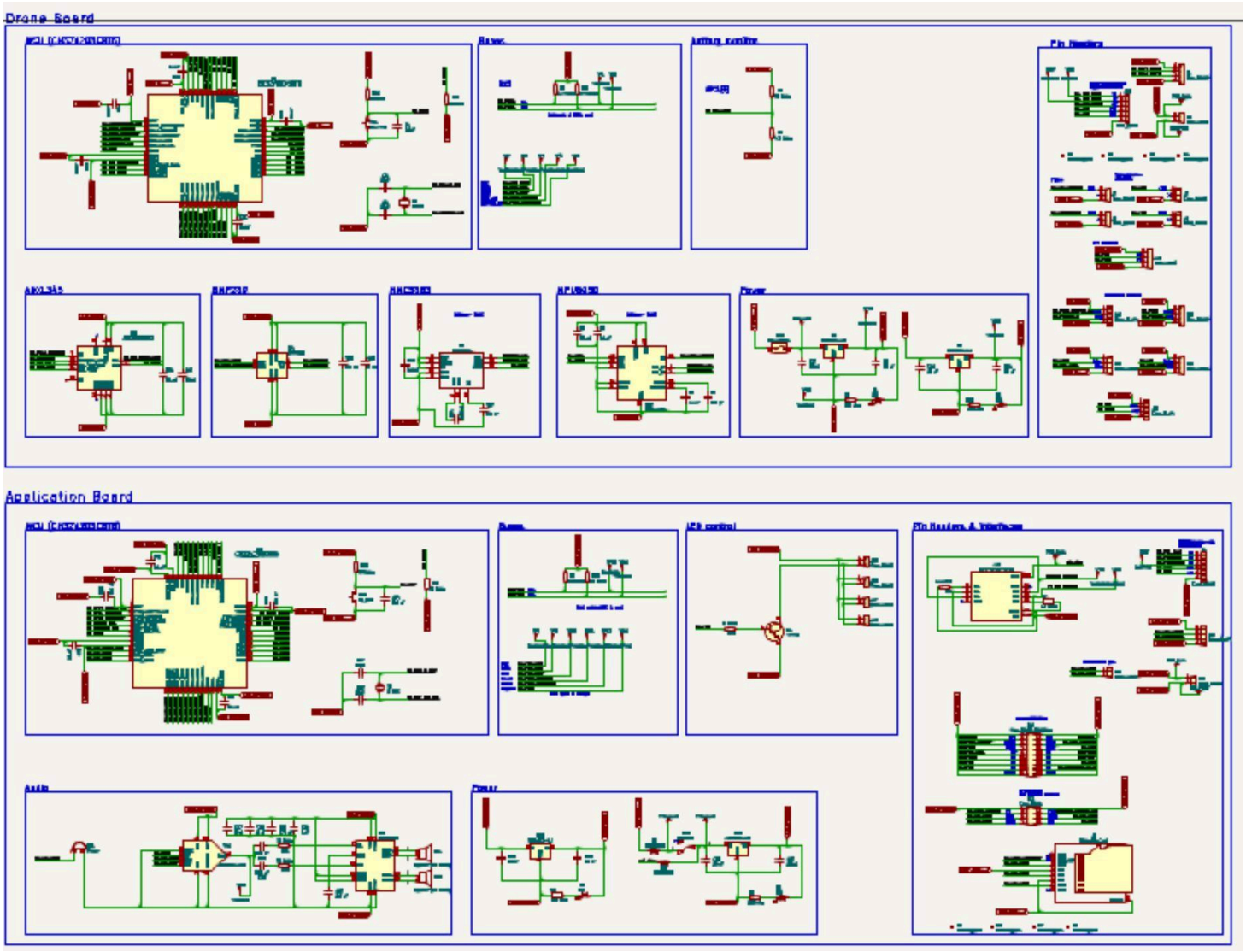


Figure 4.7 - whole PCB schematic

Where the design is split into 2 main parts: Drone board and app board. Each part of the schematic will be detailed in the Modular Decomposition. You will notice alot of component designators called ‘TP<NUM>’ where these designators are just test points to prope signals running on the board for testing integrity, correctness of the signal and some debugging.

The routing is as follows:

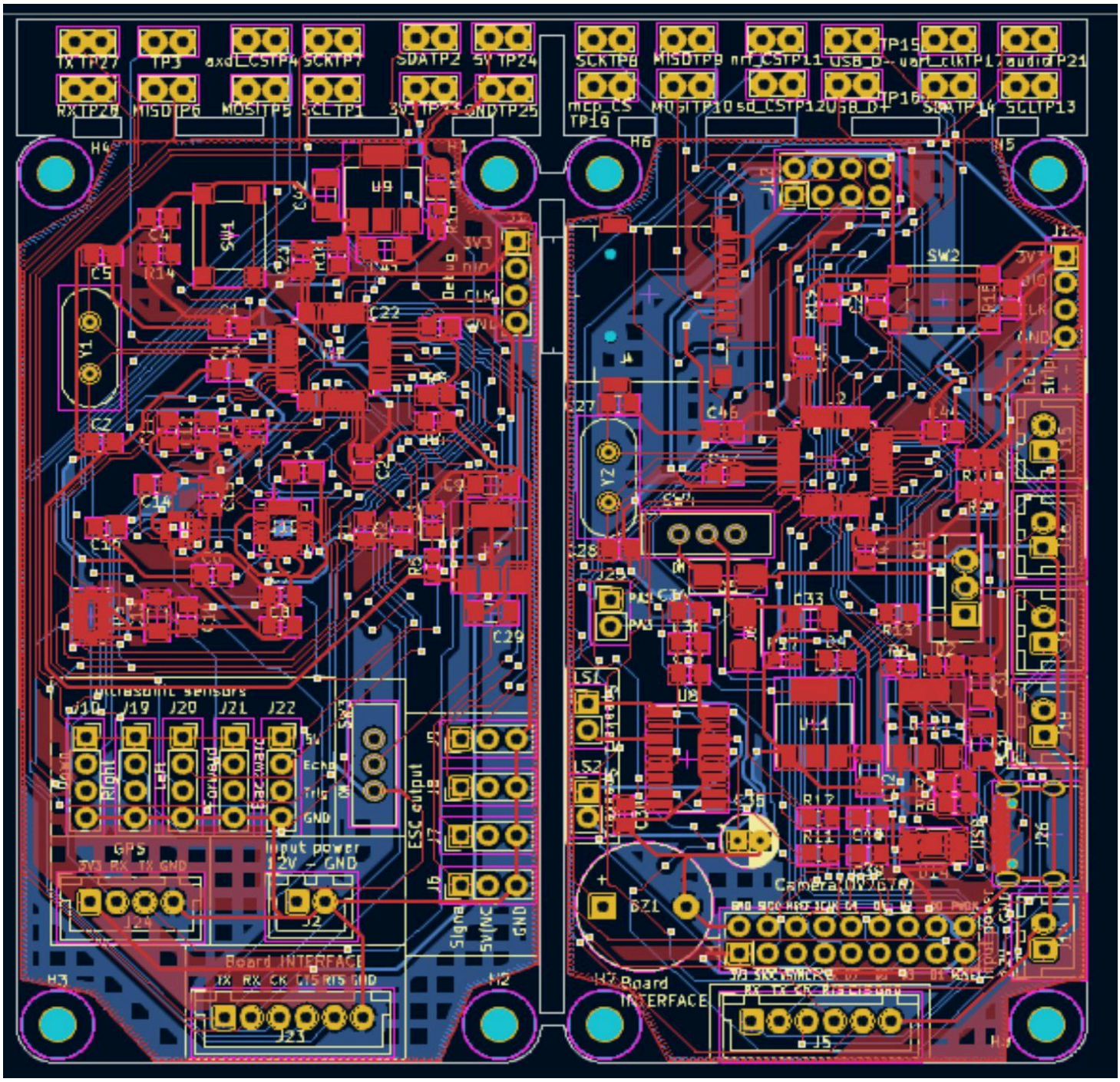
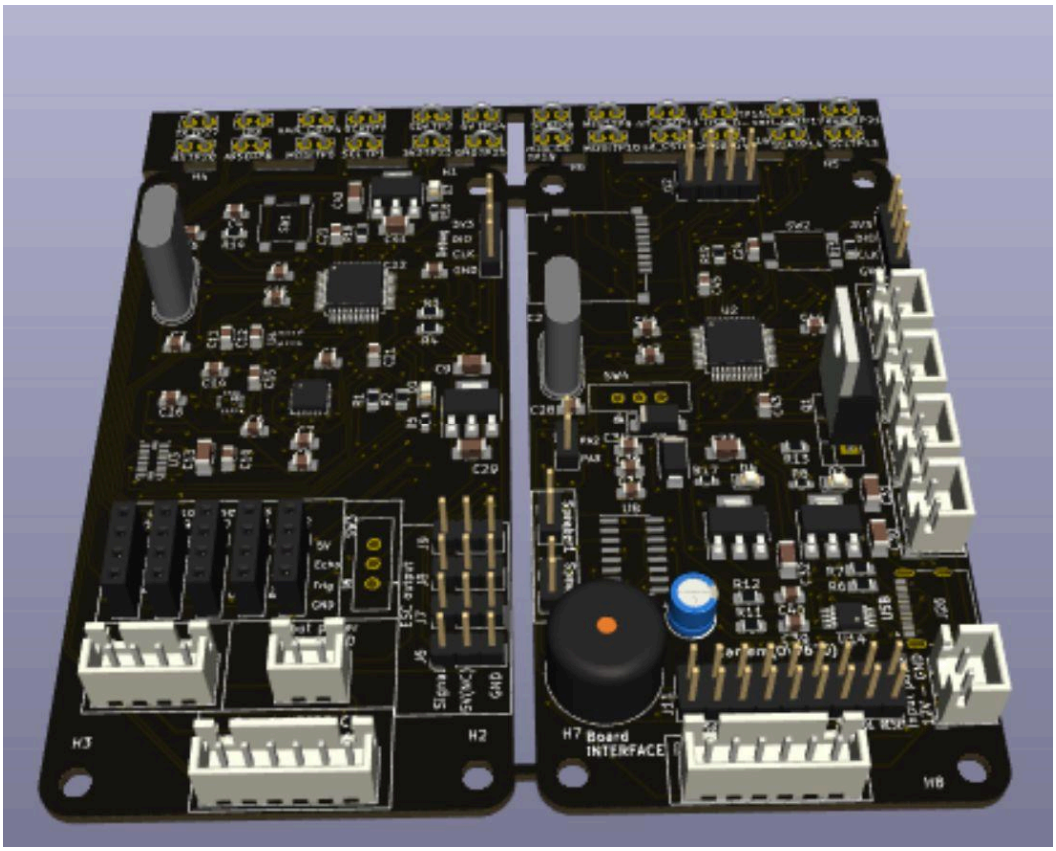
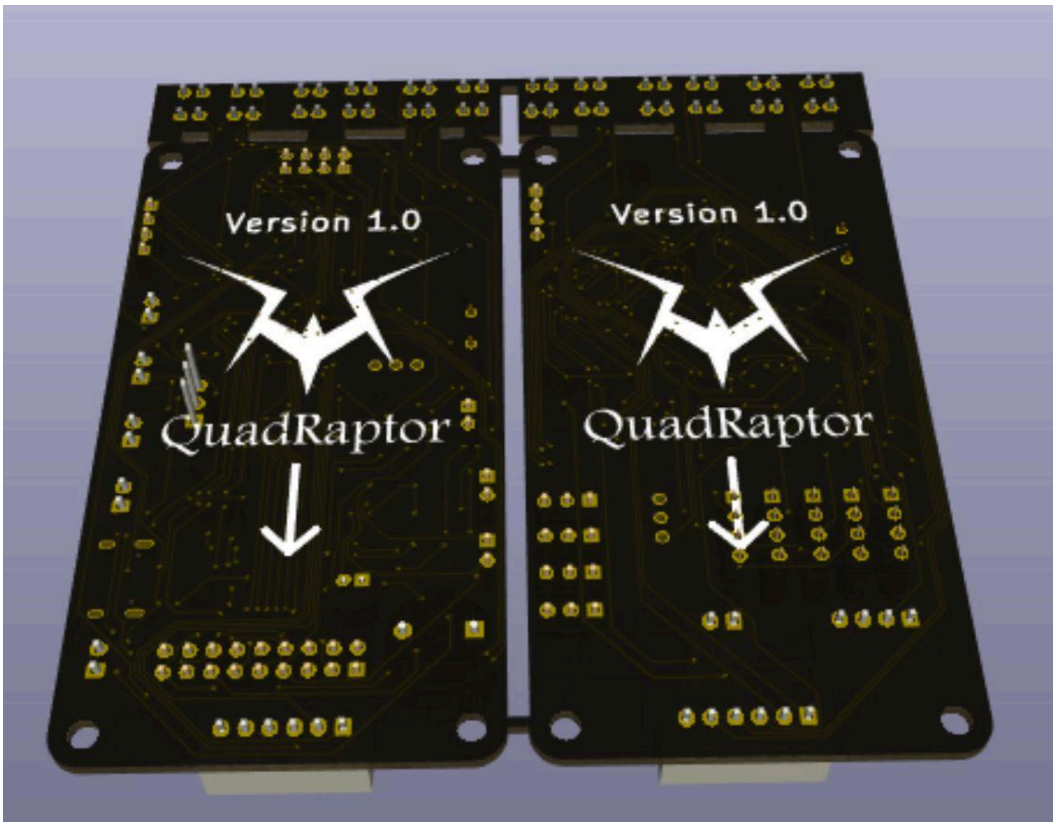


Figure 4.8 - whole PCB layout

Note: the routing of this PCB violates almost every common PCB routing rule of thumb, routing didn't take into consideration: 'skew', 'cross-talk', 'impedance matching', 'cuts in ground plane', 'etc.…'.

The intended PCB in 3D viewer:





4.3.2. Modular Decomposition

4.3.2.1 Drone board

4.3.2.1.1 MCU

We used a cheap but cost-effective RISC-V based MCU from WCH which is CH32V203C8T6 where a 0.1uF decoupling capacitor is added to every VDD pin. An external high speed 8MHZ crystal is used to drive clock of the MCU where the MCU runs at 144 MHZ. the MCU is jammed with so many peripherals for a 0.5$ MCU. NRST pin is pull-up high through 10Kohm resistor while C4 responsibility is prevent noise from reseting the MCU randomly. Boot0 is pulled low through 10Kohm pull-down resistor so the MCU will always boot from the flash memory

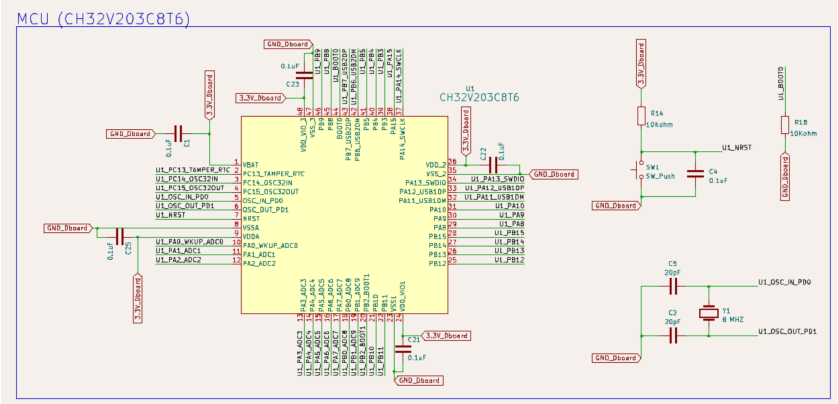


Figure 4.9 - drone board main MCU schematic

4.3.2.1.2 Buses

We have only 2 main buses in the system:

1- I2C2 bus running at fast mode (400 Kbps) connects the following ICs: (CH32v203C8T6, HMC5883L, MPU6050). Pull-up resistors for I2C bus are calculated from I2C specification.

2- SPI bus running at around 500 Kbps connects the following ICs: (CH32v203C8T6, ADXL345, BMP280).

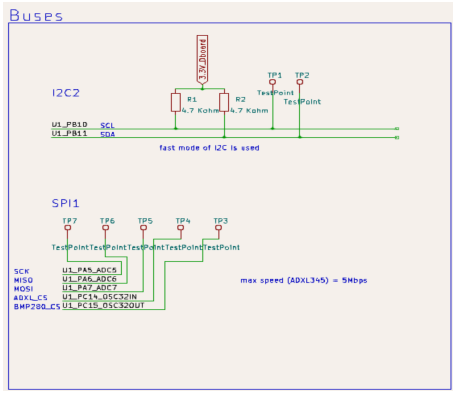


Figure 4.10 - drone board main bus schematic

4.3.2.1.3 Battery Monitor

Determining the charge of the LIPO battery was done in a dummy way by assuming linear discharge rate where maximum voltage of 1 Lithium cell is around 4.2V and minimum is 3V and so 3S lipo battery maximum voltage would be 12.6V and minimum is around 9V. Where the reading of voltage is done via ADC where input voltage is fed into voltage divider as max input volt for ADC is 3.3V.

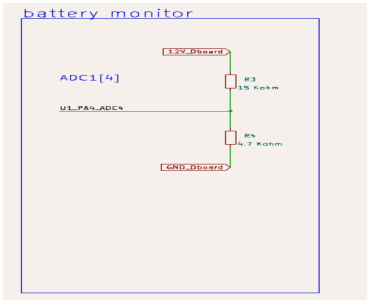


Figure 4.11 - drone board battery indicator schematic

4.3.2.1.4 ADXL345

ADXL345 is an accelerometer chip (reads linear acceleration along 3 axes) that was supposed to be fused with MPU6050 accelerometer readings to get a more accurate reading. ADXL345 is connected to the main MCU via the SPI interface with 2 configurable interrupt pins. We didn't use this chip in our project as it got a fake chip from aliexpress. So, make sure to buy chips from reputable distributors. Decoupling capacitors are added between VDD pins and GND.

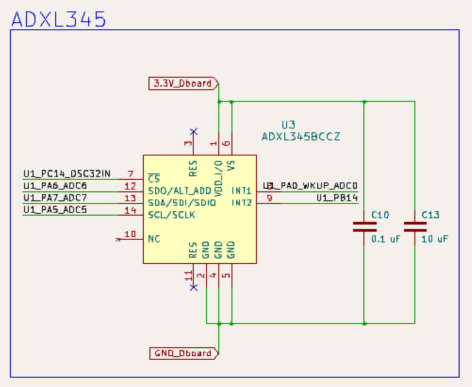


Figure 4.12 - drone board ADXL345 schematic

4.3.2.1.5 BMP280

Barometric pressure sensor whose readings are to be fused to get a more precious reading of altitude. This sensor encapsulates a temperature sensor also. Decoupling capacitors are added between VDD and GND pins. This pins is connected to the main MCU via SPI bus.

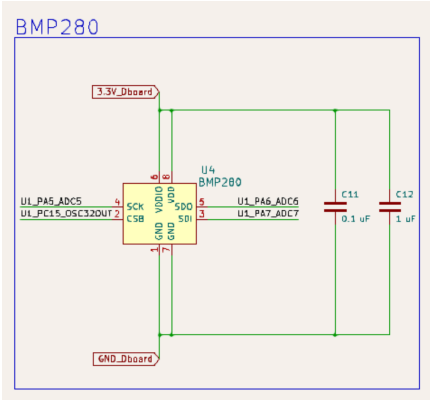


Figure 4.13 - drone board BMP280 schematic

4.3.2.1.6 HMC5883

Magnetometer IC is used to measure changes in magnetic whose readings are fused to get a correct angle for yaw. The schematics were taken from the datasheet. The IC is connected via an auxiliary I2C bus to MPU6050 which is a feature offered by the chip. The address of the chip is 0x1E.

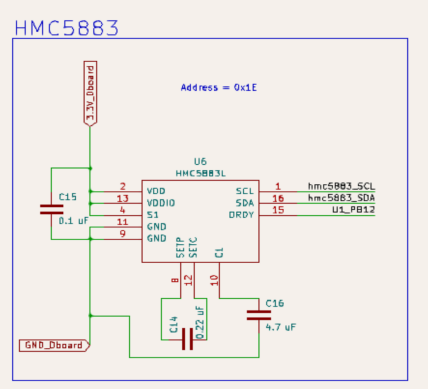


Figure 4.14 - drone board HMC5883L schematic

4.3.2.1.7 MPU6050

IMU which has an accelerometer (measures linear acceleration along each axis) and gyroscope (measures rotational acceleration around each axis). The chip connected to the main MCU via I2C bus where the address of the MPU6050 is 0x68. Decoupling capacitors and other capacitors were taken from the datasheet.

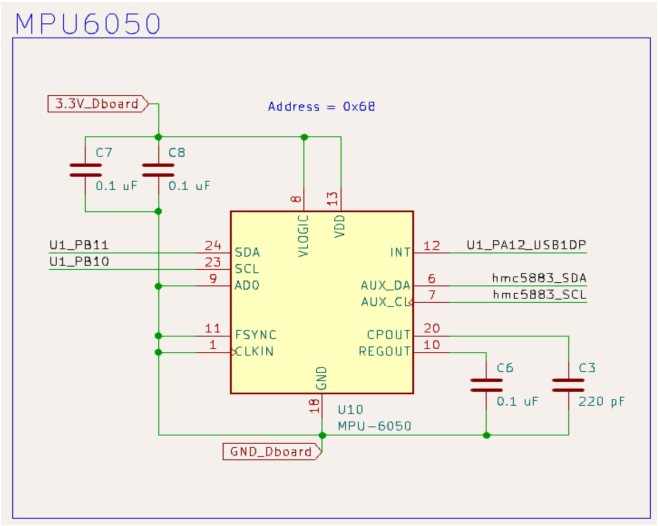


Figure 4.15 - drone board MPU6050 schematic

4.3.2.1.8 power delivery

We used LDO voltage regulators that have low voltage drops in the form of heat. We needed a 5V power supply for the ultrasonic sensor supply while the remaining circuit only needed a 3V3 supply.

An LED with a series resistor is added to the output of each regulator as an indicator that the circuit is working.

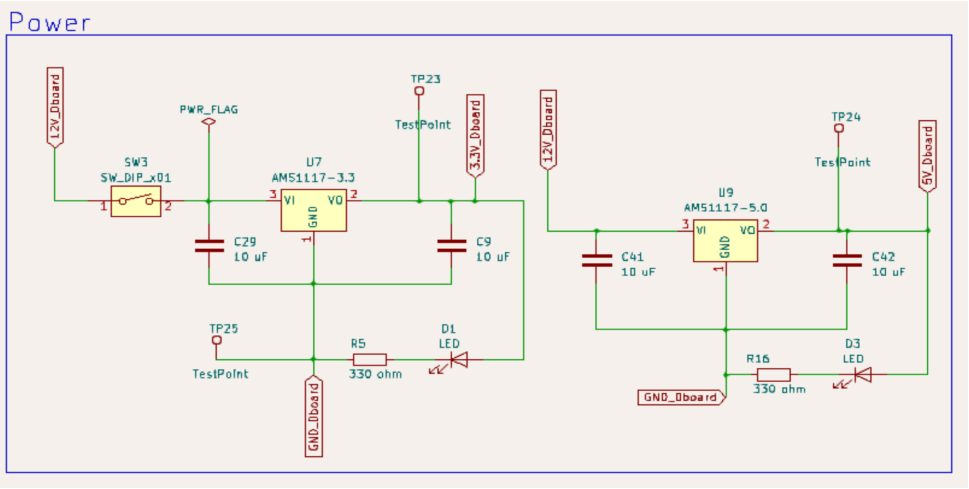


Figure 4.16 - drone board power schematic

A decoupling capacitor is added to both the input and the outputs of each regulator to smooth the power supply and remove any noise where the values of these capacitors are taken from the datasheet. A switch is added to enable/disable the power supply to the main MCU.

4.3.2.1.9 Pin headers

We have multiple pin headers divided into the following subsections:

1) 1 Debugging interface (involves PA13 and PA14 for SWDIO and SWCLK)
2) 1 Board interface (UART interface with hardware flow control for a reliable connection), note that CLK pin isn't used in UART because CLK only is used in 1 one direction as there has to be only 1 master who drives the clock
3) 4 Mounting holes (M3 screws for easily mounting the board)
4) 1 External power (typical 12V, but any number from 7V to 35V can be handled)
5) 4 ESC interface. ESC has 3 pins (GND, Signal, 5V). The 5V pin coming from the ESC isn't used in our design (it's a 5V supply coming from ESC BEC). the signal is just a PWM signal whose period time is around 20 ms. where a 5% duty cycle is zero speed and 10% duty cycle is the highest speed.
6) 1 GPS interface. Where the using of the GPS can lead to better results in estimating current position.
7) 5 ultrasonic interfaces. We only used 1 one of them to get a better estimate for height where its readings are fused with barometer readings.

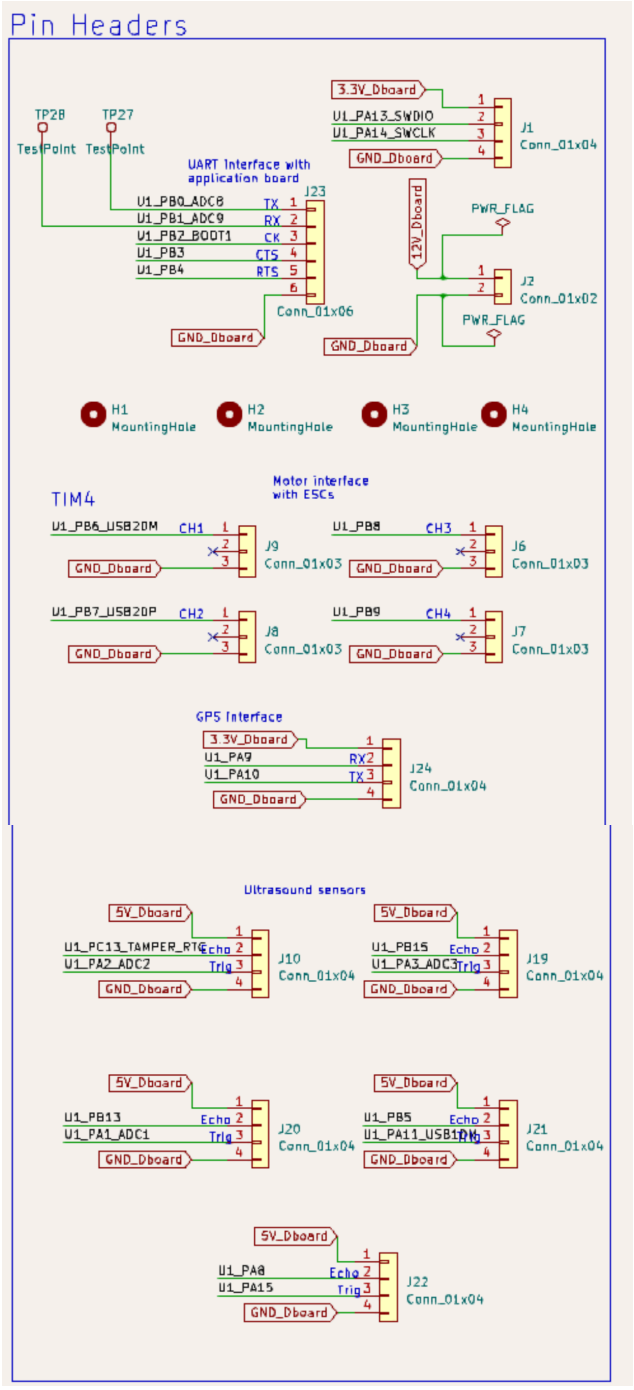


Figure 4.17 - drone board headers schematic

4.3.2.2 App board

4.3.2.2.1 MCU

We used a cheap but cost-effective RISC-V-based MCU from WCH which is CH32V203C8T6 where a 0.1uF decoupling capacitor is added to every VDD pin. An external high-speed 8MHZ crystal is used to drive the clock of the MCU where the MCU runs at 144 MHZ. The MCU is jammed with so many peripherals for a 0.5$ MCU. NRST pin is pulled up high through a 10Kohm resistor while C4 responsibility is to prevent noise from resetting the MCU randomly. Boot0 is pulled low through a 10Kohm pull-down resistor so the MCU will always boot from the flash memory.

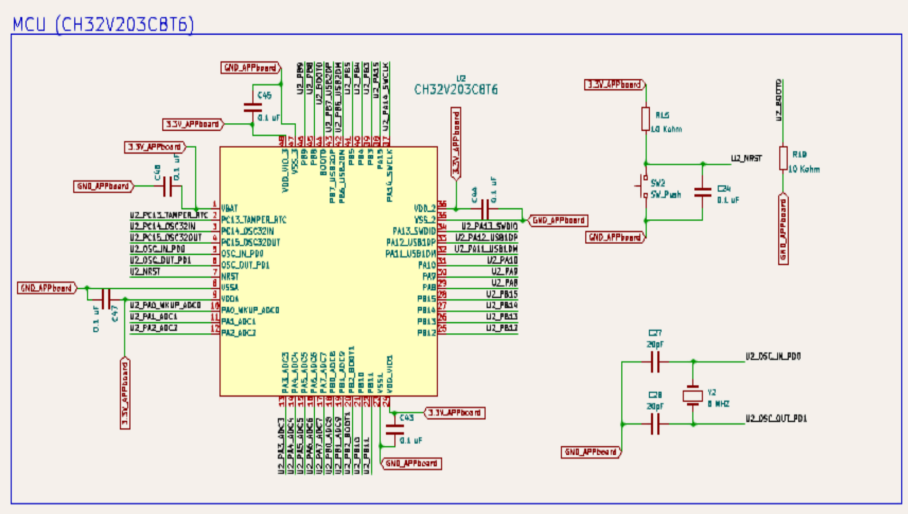


Figure 4.18 - app board main MCU schematic

4.3.2.2.2 Buses

We have only 2 main buses in the system:

1- I2C2 bus running at fast mode (400 Kbps) connects the following ICs: (CH32v203C8T6, MCP4921). Pull-up resistors for the I2C bus are calculated from the I2C specifications.

2- SPI bus running at around 500 Kbps connects the following ICs: (CH32v203C8T6, NRF24L01, SDcard).

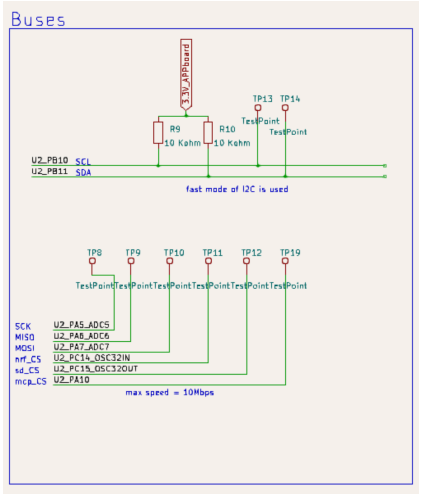


Figure 4.19 - app board buses schematic

4.3.2.2.3 Led control

TIP121 transistor (BJT transistor) is used to drive 4 led strips where a base 1 kohm resistor is added to the base of the transistor to prevent short current at the base of the transistor.

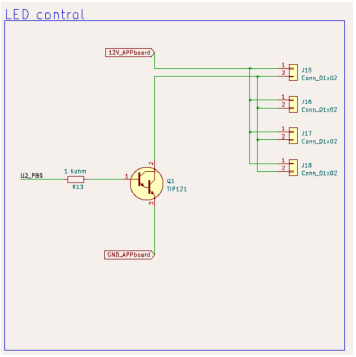


Figure 4.20 - app board led control schematic

4.3.2.2.4 Audio

A buzzer is connected to one of Timers output channels. MCP4921 DAC communicates with main MCU through SPI connection and the output of the DAC is then fed to the input channel of the amplifier which then is fed into speakers. The schematic were taken from the datasheet.

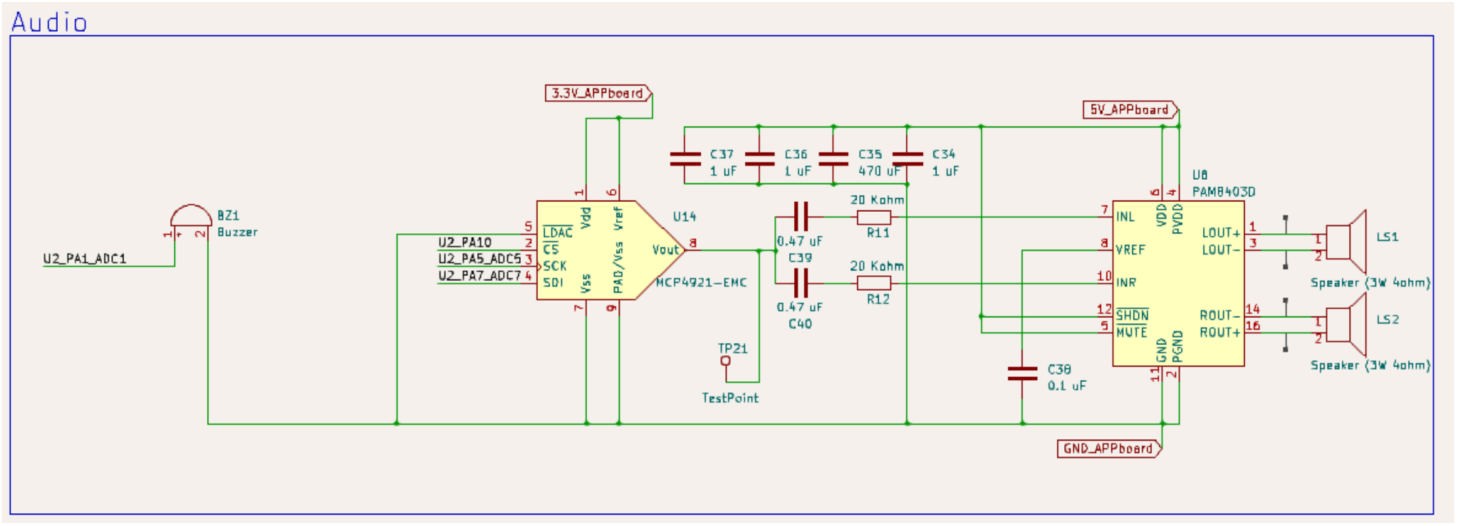


Figure 4.21 - app board audio schematic

4.3.2.2.5 power delivery

We used LDO voltage regulators that have low voltage drops in the form of heat. We needed a 5V power supply for the amplifier while the remaining circuit only needed a 3V3 supply.

An LED with a series resistor is added to the output of each regulator as an indicator that the circuit is working.

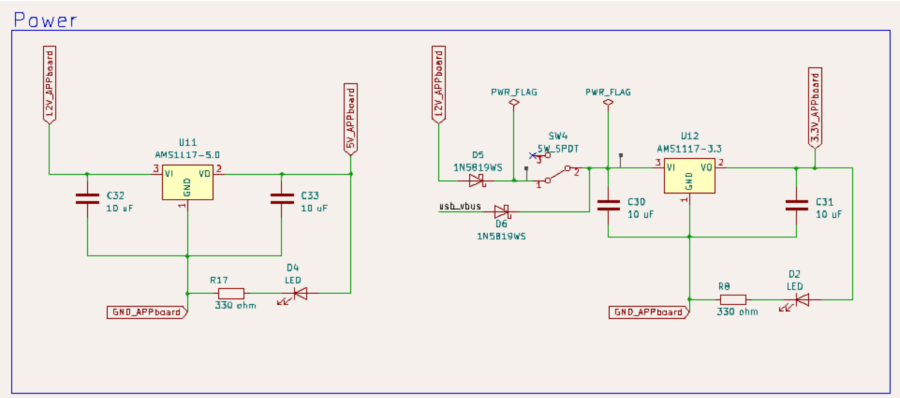


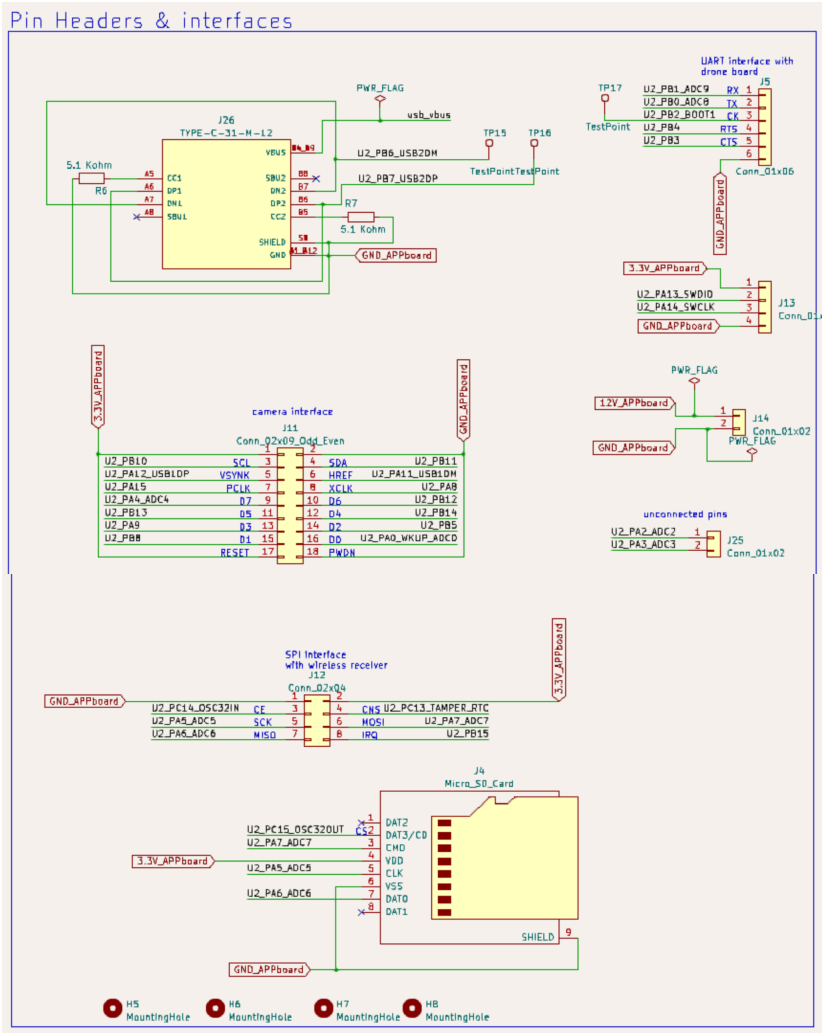
Figure 4.22 - app board audio schematic

A decoupling capacitor is added to both the input and the outputs of each regulator to smooth the power supply and remove any noise where the values of these capacitors are taken from the datasheet. A switch is added to enable/disable the power supply to the main MCU.

Since we can have 2 sources of power (USB or from LIPO battery). We used schottky diodes to OR power sources.

4.3.2.1.9 Pin headers

Figure 4.23 - app board interface schematic



We have multiple pin headers divided into the following subsections:

1) 1 USB-type C. we made some mistakes here as grounding the shield. USB-C was intended to be interface between laptop and the drone for updating firmware, calibration, etc…
2) 1 Debugging interface (involves PA13 and PA14 for SWDIO and SWCLK)
3) 1 Board interface (UART interface with hardware flow control for a reliable connection), note that CLK pin isn't used in UART because CLK only is used in 1 one direction as there has to be only 1 master who drives the clock.
4) 4 Mounting holes (M3 screws for easily mounting the board)
5) 1 External power (typical 12V, but any number from 7V to 35V can be handled)
6) 2 unconnected pins,. Extra pins not needed but maybe helpful in debugging.
7) 1 NRF24L01 pin header interface. A nrf24L01 is connected to these pin headers for wireless communication with the remote control.
8) 1 SD card interface. Useful for storing captured images, debugging info, audio, etc…
9) 1 OV7670 interface. A low quality camera interface for taking pictures.

4.3.3. Design Constraints

Each board dimension wasn't supposed to be more than 10 cm x 5 cm.

4.3.4. Other Description of PCB design Module

Routing and layout of this PCB is really really bad. Make sure to route it on 4 layers not 2!.

4.4. Sensors and Estimators

The first step to balance and control the quadcopter is to know its current state. This is done by fusing 10 DOF sensors. This module is responsible for monitoring the altitude and the flight angles of the drone.

4.4.1. Functional Description

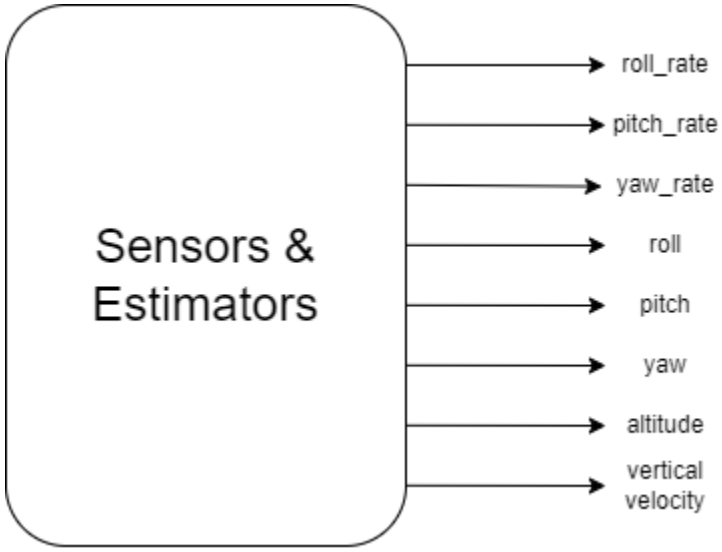


Figure 4.24 - Sensors and estimators.

The sensing and estimation module is a critical component of the quadcopter's flight controller, responsible for acquiring, processing, and integrating data from various sensors to estimate the vehicle's state in real-time. The state includes parameters such as orientation (roll, pitch, and yaw), altitude, and angular velocities. These estimates are essential for the flight controller to maintain stability and control the quadcopter's movements.

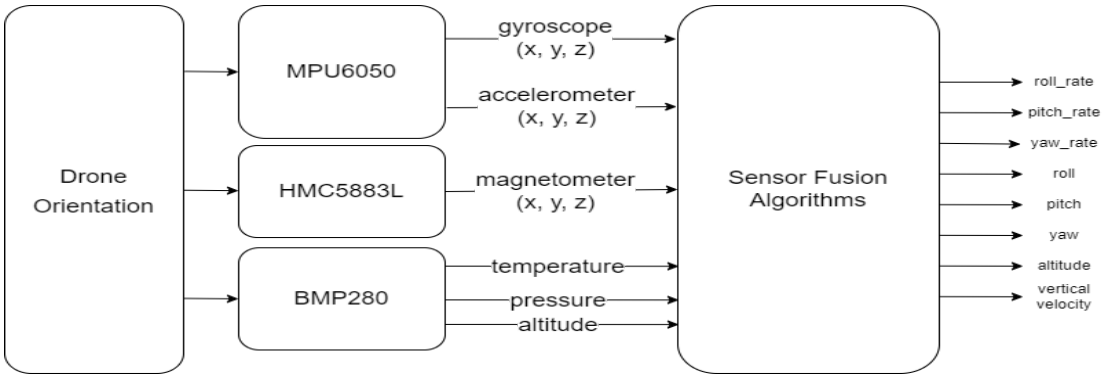


Figure 4.25 - block diagram of sensor feedback

The module integrates data from several sensors: the MPU6050 (a combined gyroscope and accelerometer), the HMC5883L (a magnetometer), and the BMP280 (a barometer). Each sensor provides specific types of data:

MPU6050: Provides angular velocity (from the gyroscope) and linear acceleration (from the accelerometer).

HMC5883L: Measures the Earth's magnetic field, used to determine the quadcopter's heading (yaw).

BMP280: Measures atmospheric pressure, used to estimate the quadcopter's altitude.

To obtain accurate state estimates, the sensing and estimation module employs sensor fusion techniques, including 1D and 2D Kalman filters. The 2D Kalman filter combines readings from the barometer and accelerometer to estimate altitude and vertical velocity, while the 1D Kalman filter is used for pitch and roll estimation. These filters integrate data from the various sensors, considering the uncertainties in each measurement and the system model, to produce more accurate and reliable estimates than any single sensor could provide.

4.4.2. Modular Decomposition

The sensing and estimation module is decomposed into several smaller sub-modules, each with distinct responsibilities:

- Sensor Interface Module: This module handles the hardware interfaces for the MPU6050, HMC5883L, and BMP280 sensors. It includes routines for initializing the sensors, performing necessary calibrations, and acquiring raw data. Calibration is critical for compensating for sensor biases and errors, ensuring that the data is as accurate as possible before further processing.
- Preprocessing Module: The raw data from the sensors is often noisy and may require preprocessing to improve its quality. This module applies filtering techniques, such as low-pass filters, to remove high-frequency noise from the accelerometer and gyroscope data. Additionally, it applies transformations to convert sensor readings into a common frame of reference.
- Sensor Fusion Module: This module is the core of the sensing and estimation system, where data from multiple sensors is combined using Kalman filters. The 1D Kalman filter is used for estimating roll and pitch angles by integrating gyroscope data with accelerometer data to correct for drift and noise. The 2D Kalman filter is used for altitude and vertical velocity estimation, combining barometric pressure readings with vertical acceleration data from the accelerometer. The Kalman filters account for measurement noise and process noise, dynamically adjusting their estimates based on new sensor data and the model's predictions.
- State Estimation Module: Based on the fused sensor data, this module computes the quadcopter's orientation (roll, pitch, and yaw), altitude, and angular velocities. Orientation is determined by integrating accelerometer and gyroscope data, with corrections from the magnetometer to prevent drift. Altitude and vertical velocity are estimated from the barometric pressure readings and vertical acceleration data, processed through the 2D Kalman filter.
- Data Communication Module: Once the state estimates are computed, this module handles their transmission to other parts of the flight controller. This ensures that the control algorithms have access to the most recent and accurate state information, which is crucial for maintaining stability and executing control commands.

4.4.3. Design Constraints

Several constraints influenced the design of the sensing and estimation module:

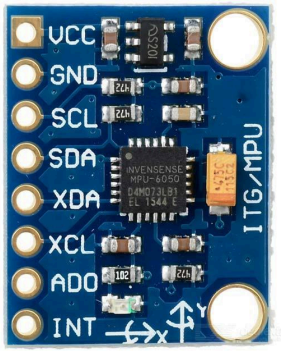
- Real-time Performance: The module must process sensor data and update state estimates with minimal latency to ensure timely responses from the flight controller. Real-time performance is critical for maintaining stability, as any delays can result in sluggish or incorrect control responses. The algorithms and data processing routines are optimized to run efficiently within the time constraints.
- Computational Efficiency: The quadcopter's onboard processing capabilities are limited, necessitating efficient algorithms that do not overburden the processor. The use of Kalman filters, which are computationally efficient for real-time applications, helps balance the need for accurate state estimates with the available computational resources. The 1D Kalman filter for pitch and roll estimation simplifies the computations required for these orientations, while the 2D Kalman filter effectively manages the more complex integration of barometer and accelerometer data for altitude and vertical velocity.
- Accuracy and Precision: Accurate state estimates are essential for reliable flight control. This requires precise calibration of sensors to reduce biases and errors, as well as carefully tuned Kalman filter parameters to ensure optimal performance. High accuracy is particularly important for maintaining stable flight and executing precise maneuvers.
- Robustness: The module must handle sensor noise, inaccuracies, and potential failures gracefully. Robustness is enhanced by using multiple sensors and sensor fusion techniques, which provide redundant information that can compensate for the failure or inaccuracy of any single sensor. The Kalman filters are particularly effective in this regard, as they are capable of adjusting their estimates dynamically in response to changing sensor conditions and noise characteristics.
- Power Consumption: Power efficiency is a critical consideration for extending flight time. The module is designed to operate within the power constraints of the quadcopter, minimizing energy usage while still providing accurate state estimates. Efficient data acquisition and processing help reduce overall power consumption, contributing to longer flight durations. The choice of sensors, such as the low-power MPU6050 and BMP280, further supports this goal by minimizing the energy required for continuous sensing operations.

By addressing these constraints and employing advanced sensor fusion techniques, the sensing and estimation module ensures that the quadcopter can maintain stable and reliable flight, even in challenging conditions. The careful integration of sensor data, use of transformations for orientation estimation, and dynamic adjustment capabilities of the Kalman filters collectively contribute to the module's effectiveness and reliability. This detailed approach enables the flight controller to accurately monitor and control the quadcopter's state, ensuring safe and efficient operation during flight.

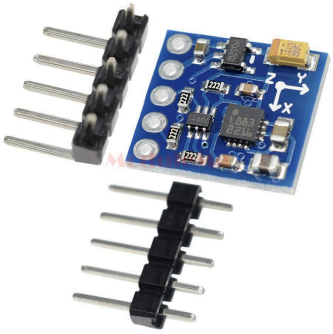
4.4.4. Other Description of Sensors and Estimators

Now that we have an overview of how the feedback of our system works. Let's explore in detail the sensors that we used. Before we start, I need to elaborate that these sensors were mainly chosen based on the **availability in the local market**. The chosen sensors were the only ones of their kind.

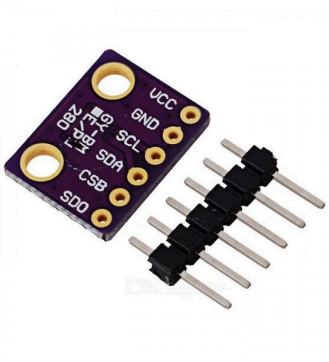
Each of the sensors has its own module that is available in almost every electronics store. We chose a module that combines them together to ease the development and testing process. This module is called GY-87. It is a 10-DOF IMU. Its only limitation is that it contains BMP180 and not BMP280. Both sensors are produced by the same manufacturer. They serve the same function with different accuracy and performance. It was not a big deal since the difference between their registers architecture is not huge.



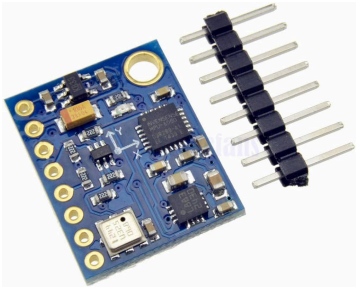
MPU-6050



HMC5883L



BMP280



GY-87

Let's explore the sensors one by one

MPU6050

This is the most crucial sensor among all of them. Not only that it offers 6 DOF in one chip, but also that it contains the gyroscope. Gyroscope provides robust feedback of the angle rates. It is the cornerstone of the rate mode controller that helps the drone maintain its balance.

Fortunately, this is one of the most commonly used sensors in the development of drones and self-balancing robots. You can easily find open-source implementations of interfacing with this sensor. Most of them are Arduino-based. They were helpful in validating that the sensor works correctly.

Here we can see the results of the first check:

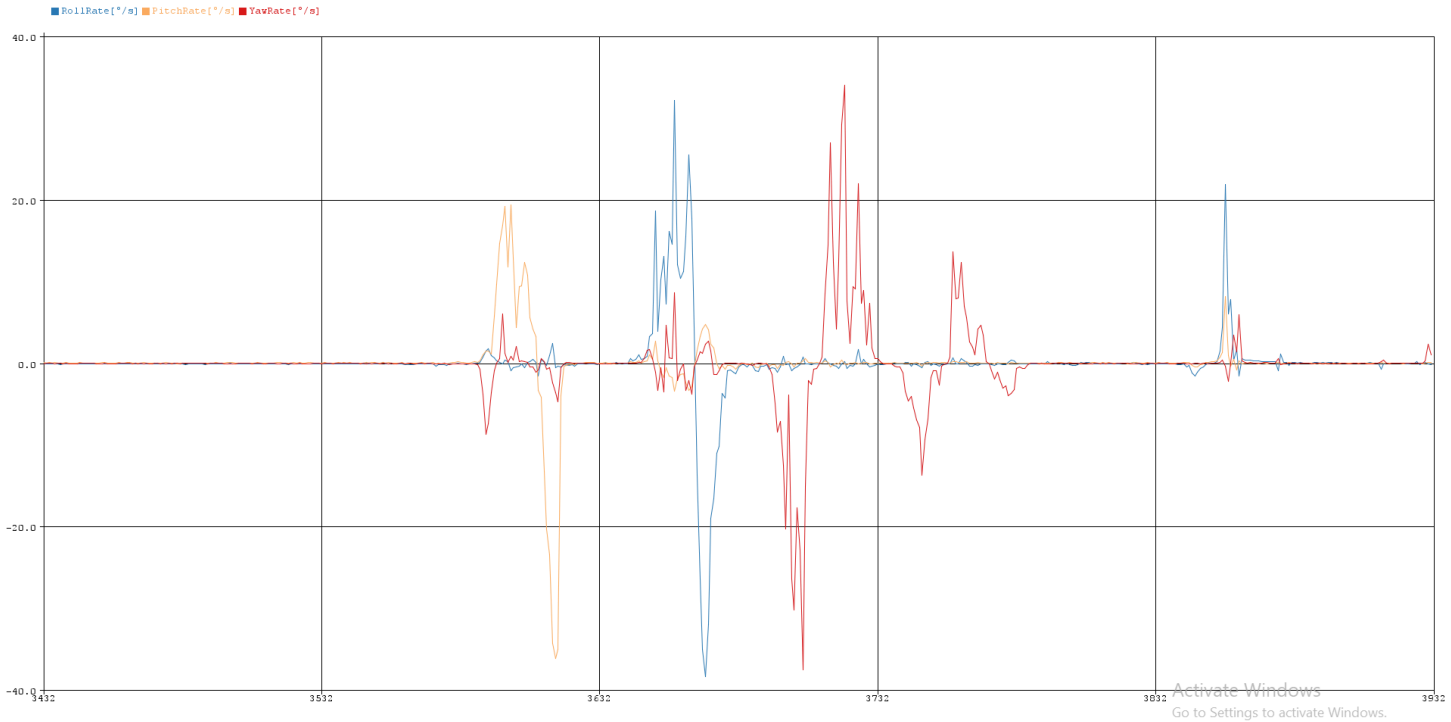


Figure 4.26 - Gyroscope readings

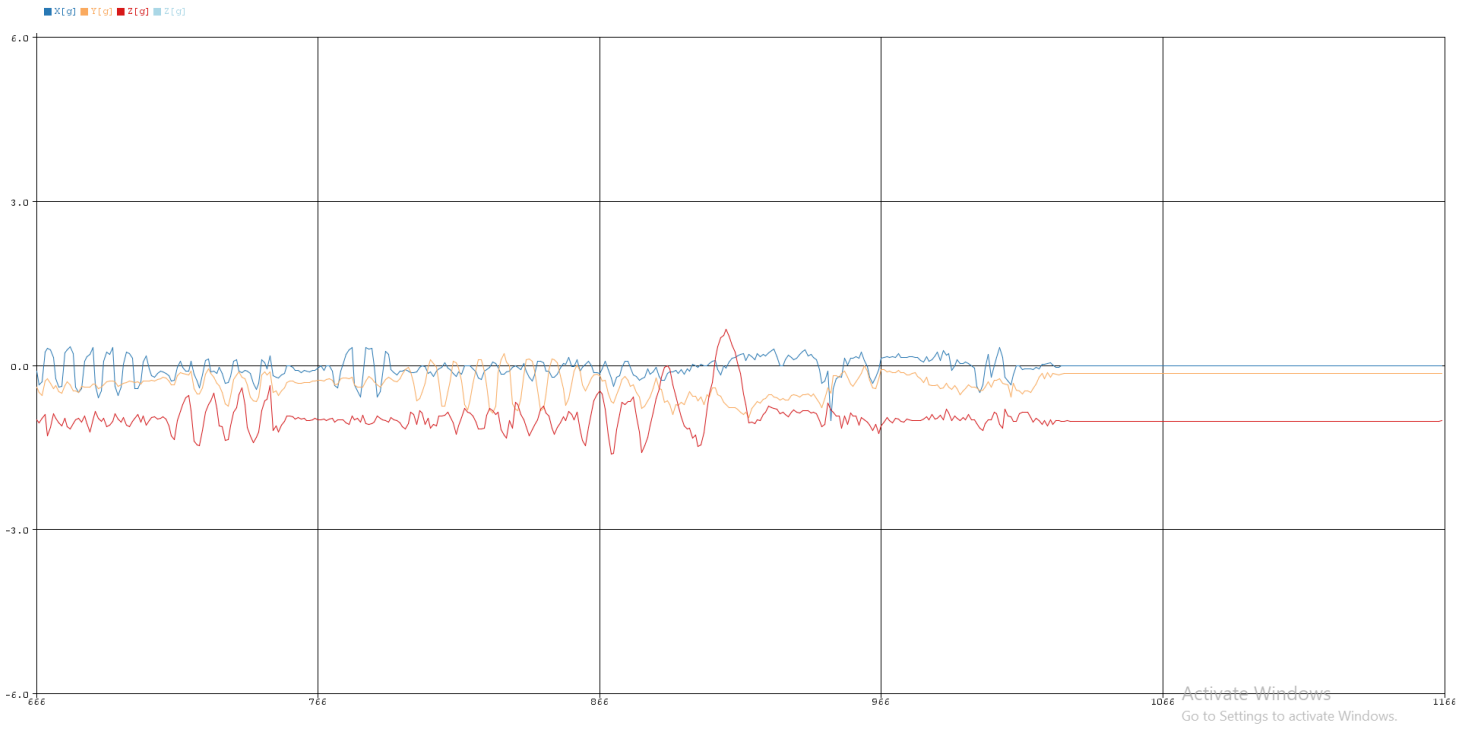


Figure 4.27 - Accelerometer readings

Note: These results were generated using Arduino Uno and GY-87 module

HMC5883L

The HMC5883L is a three-axis digital magnetometer, commonly referred to as a digital compass, manufactured by Honeywell. It is designed to measure the strength and direction of the Earth's magnetic field, which can be used to determine orientation in space.

It operates based on the principle of magnetoresistance, where changes in the magnetic field cause variations in the electrical resistance of the material inside the sensor. These variations are then converted into digital signals representing the strength and direction of the magnetic field.

To interface the HMC5883L with a microcontroller, the I2C communication protocol is used. The sensor's registers can be accessed to configure measurement modes, read data, and perform self-tests. In our board, the HMC5883L is connected to the auxiliary bus of the MPU6050. We chose to match the GY-87 module for stable and easy integration.

This sensor is a commonly used sensor in open-source projects (just like the MPU6050). So many Arduino sketches were found that helped with validating that the sensor works. It is very important to check whether the sensor you bought is HMC5883L or QMC5883L. They are both sold with the same name and the same package. The two sensors are identical, except for their slave addresses.

For accurate measurements, the HMC5883L needs to be calibrated to account for hard iron and soft iron distortions. Hard iron distortions are caused by permanent magnetic objects near the sensor, while soft iron distortions are due to ferromagnetic materials affecting the magnetic field. Calibration typically involves rotating the sensor in all directions and using algorithms to correct the distortions. We used a simple normalization technique to account for those errors.

The calibration process is done over several steps. Logging sensor readings while rotating it. Feeding the output logs into a simple python script that calculates the offset and scale along each axis. Note that we are mainly concerned with calibrating readings along x-axis and y-axis.

In our domain, we are using the magnetometer to determine the heading angle. It is also called the azimuth angle. This is the angle between the sensor and the north pole. It is very useful for navigation purposes. The computation of heading angle will be further explained in the Sensor Fusion section.

BMP280

The BMP280 is a high-precision, low-power barometric pressure sensor developed by Bosch Sensortec. It is designed to measure atmospheric pressure and temperature, making it suitable for a variety of applications, including weather forecasting, altitude tracking in drones and smartphones, and indoor navigation. The BMP280 is known for its accuracy, versatility, and ease of integration into microcontroller-based systems.

The BMP280 natively measures pressure and temperature. We used those readings to get an estimate of the altitude using a formula that will be explained later.

Sensor Fusion Algorithms

Until now, the previously mentioned sensors can only provide raw measurements. These measurements do not fit the main requirements to track the state of the drone in a reliable way. Our main goal is to track the drone's Euler angles [Roll(ϕ), Pitch(θ), Yaw(ψ)].

Let's tackle the calculation of those angles one by one.

Yaw (ψ)

Yaw angle is not necessarily important for the drone's control. It's much easier to control the yaw angle using a rate mode controller (measured by the gyroscope). As we previously mentioned, we need the heading angle for navigation purposes. Let's start with the simplest method to estimate the heading angle.

$$\psi = tan^{-1}(\frac{y}{x}),\ where\ x\ and\ y\ are\ the\ magnetometer\ measurements$$

What if the drone pitched or rolled during its flight? The equation is no longer valid since the body frame is now different from the inertial frame. Let's get the inertial y-axis of the magnetometer measurements.

$$y_{inertial} = y * cos\,\theta + z * sin\,\theta$$

$$x_{inertial} = x * cos\,\phi + z * sin\,\phi$$

The second problem is a natural phenomenon called magnetic declination. Also known as the declination angle, is the angle between true north (geographic north) and magnetic north (the direction a compass needle points). This angle varies depending on geographic location and changes over time due to variations in the Earth's magnetic field.

Using NOAA's (National Oceanic and Atmospheric Administration) Magnetic Field Calculator, we were able to determine the magnetic declination. The last step for a correct calculation of the heading angle is adding the declination angle (δ).

$$\psi = tan^{-1}\left(\frac{y^{*}cos\,\theta + z^{*}sin\,\theta}{x^{*}cos\,\phi + z^{*}sin\,\phi}\right) + \delta$$

Note that tan^{-1} can be used in the code using the atan2 function from the math library.

Note that the order of the parameters passed to the atan2 function depends on the orientation of the sensor on the drone board.

Roll (ϕ)

The simplest method to estimate roll angle is using the gyroscope's reading. Since gyroscope measures angular rates, we can use the following equation.

$$\phi = \int_0^{k*Ts} x\,dt \;,\; where\ x\ is\ the\ angular\ velocity\ around\ x\ axis$$

Discrete form:

$$\phi(k) = \phi(k-1) + x * Ts$$

This method has a great downfall. The small noise in the gyroscope measurements will accumulate as we integrate its readings. This will produce a constantly increasing/decreasing roll angle.

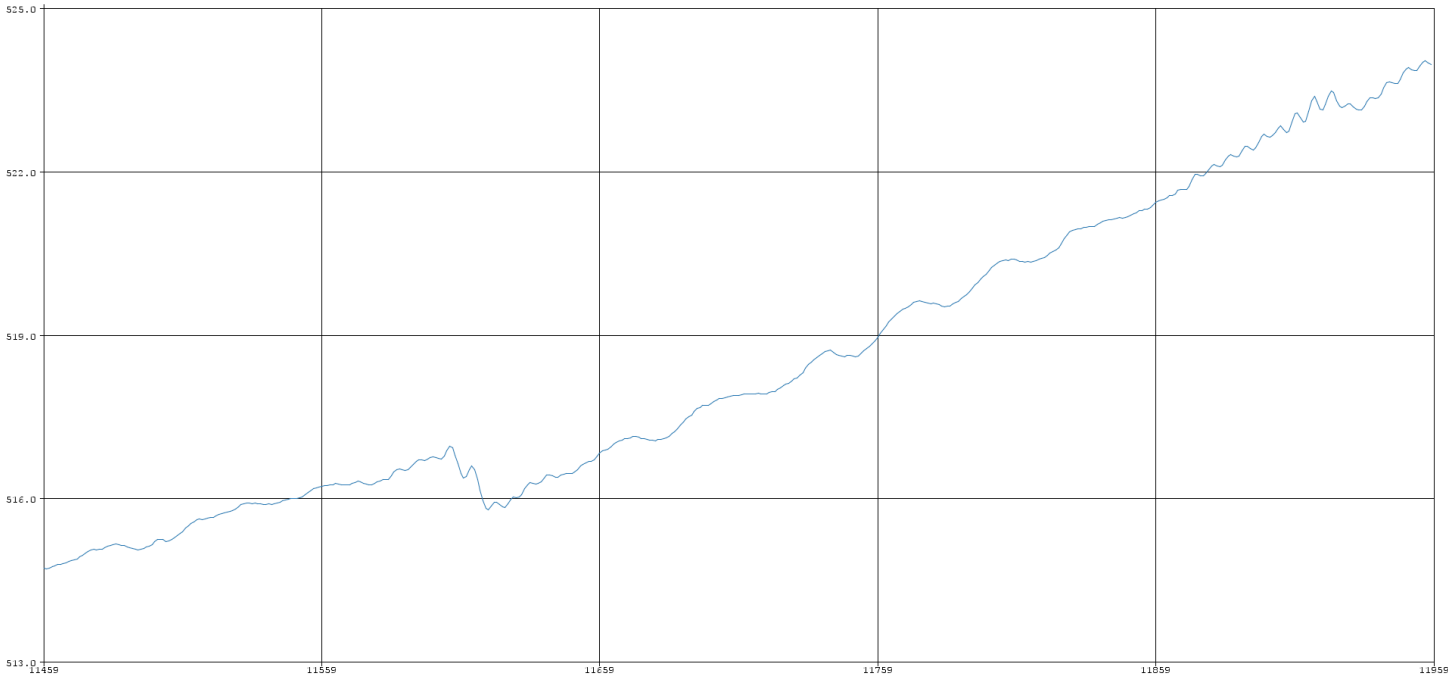
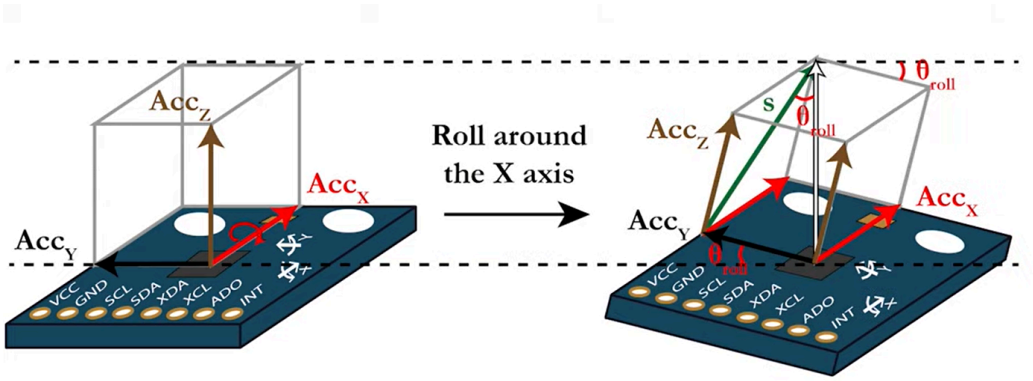


Figure 4.28 - gyroscope readings

Let’s try another estimate of the roll angle using the accelerometer.



From the previous figure we can find that the roll angle can be estimated as follows

$$\phi = tan^{-1}\frac{y}{\sqrt{x^2+z^2}}$$

Let's observe the results of this approach.

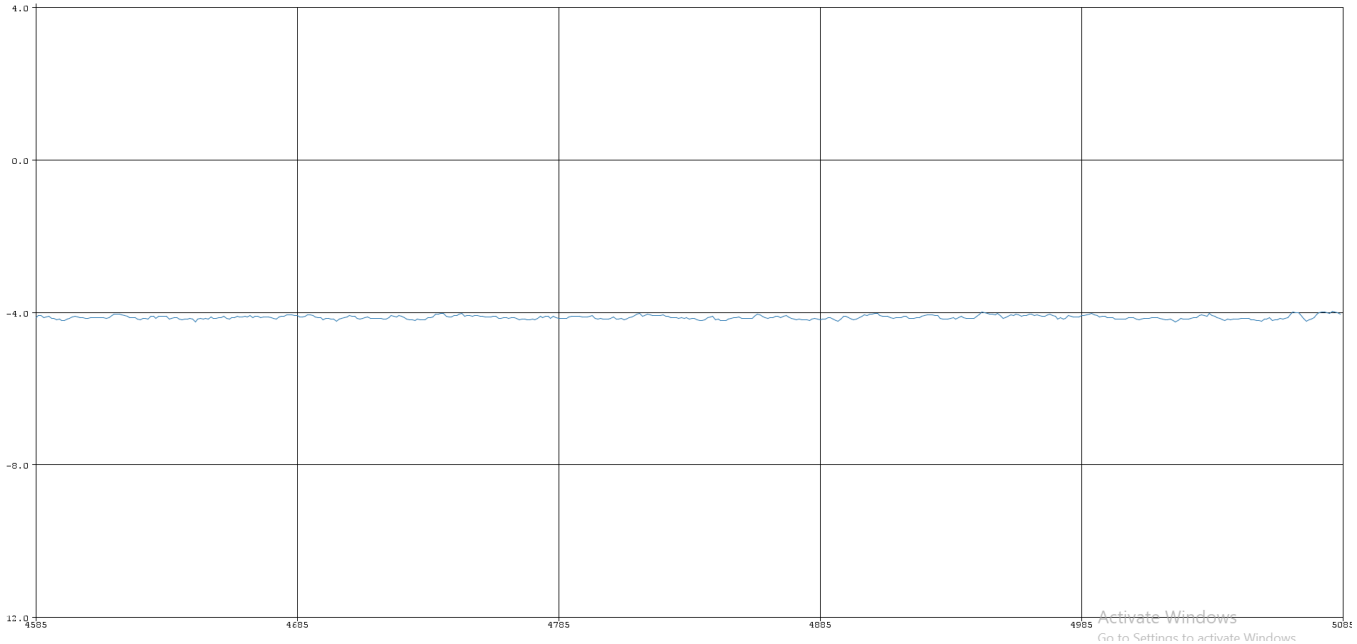


Figure 4.29 - fused gyroscope readings

As we can see, the previous problem is solved. But the measurements are a bit noisy. Let's try to simulate the vibrations of the drone and see what happens.

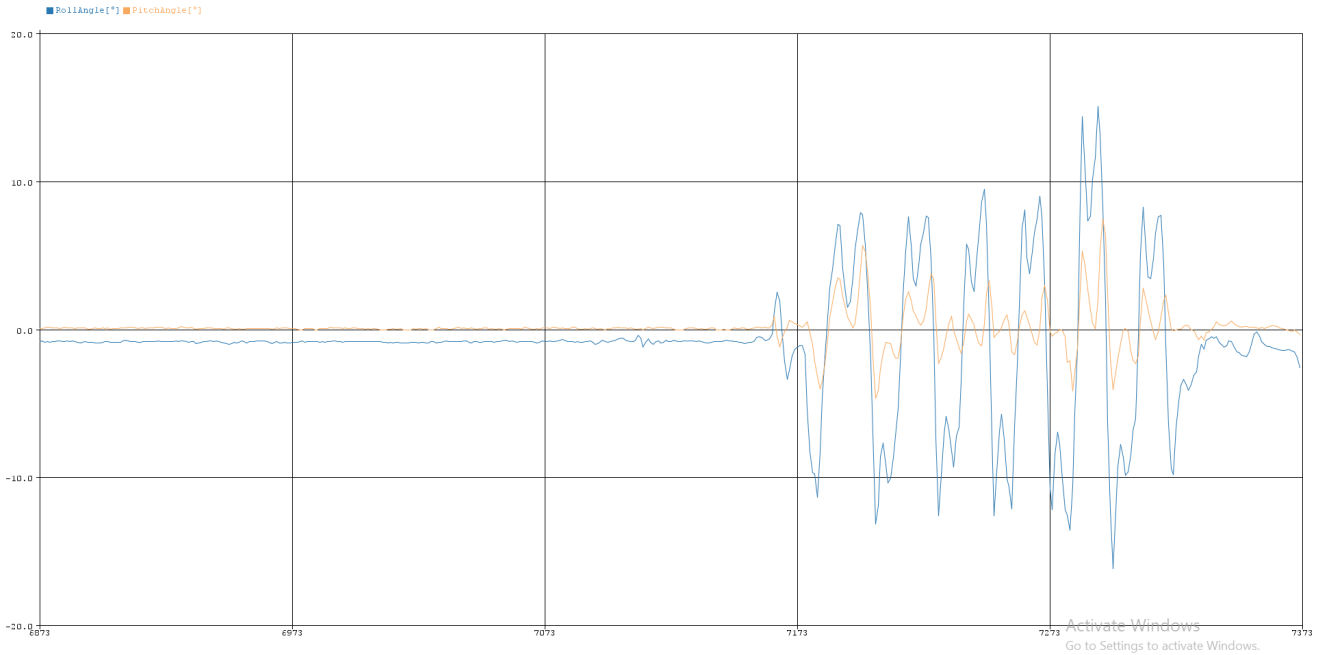


Figure 4.30 - noisy fused gyroscope readings

Both of the previous images represent the sensor reading while being at zero roll angle.

To account for the errors of the gyroscope and accelerometer, we had to use a sensor fusion algorithm. We decided to use Kalman filter for its promising results. Let’s show its steps

Kalman filter

1. Predict the current state of the system:

$$S(k) = F \cdot S(k-1) + G \cdot U(k)$$

S=state vector ($Angle_{kalman}$)
F=state transition matrix (1)
G=control matrix (0.004)
U=input variable (Rate)

This equation can be translated into the following

$$\phi = \phi + gyro_x * Ts$$

2. Calculate the uncertainty of the prediction:

$$P(k) = F \cdot P(k-1) \cdot F^T + Q$$

P=prediction uncertainty vector ($Uncertainty_{angle}$)
Q=process uncertainty ($T_s^2 . 4^2$)

The prediction uncertainty vector of the roll angle will be a 1D vector.
Note that $Q = \sigma^2_{gyroscope} * Ts^2$

3. Calculate the Kalman gain from the uncertainties on the predictions and measurements:

$$L(k) = H \cdot P(k) \cdot H^T + R$$

$$K = P(k) \cdot \frac{H^T}{L(k)} = P(k) \cdot H^T \cdot L(k)^{-1}$$

L= Intermediate matrix
K=Kalman gain
H=Observation matrix (=1)
R=Measurement uncertainty ($T_s^2 . 3^2$)

For simplicity it can be written as follows

$$K = \frac{Uncertainty_{roll}}{Uncertainty_{roll} + \sigma^2_{accelerometer} * Ts^2}$$

4. Update the predicted state of the system with the measurement of the state through the Kalman gain:

$$S(k) = S(k) + K \cdot (M(k) - H \cdot S(k))$$ M=measurement vector (Angle)

In our case: $$\phi = \phi + K * (\phi_{acc} - \phi)$$

5. Update the uncertainty of the predicted state:

$$P(k) = (I - K \cdot F) \cdot P(k)$$ I=unity matrix (=1)

$$Uncertainty_{roll} = (1 - K) * Uncertainty_{roll}$$

Results

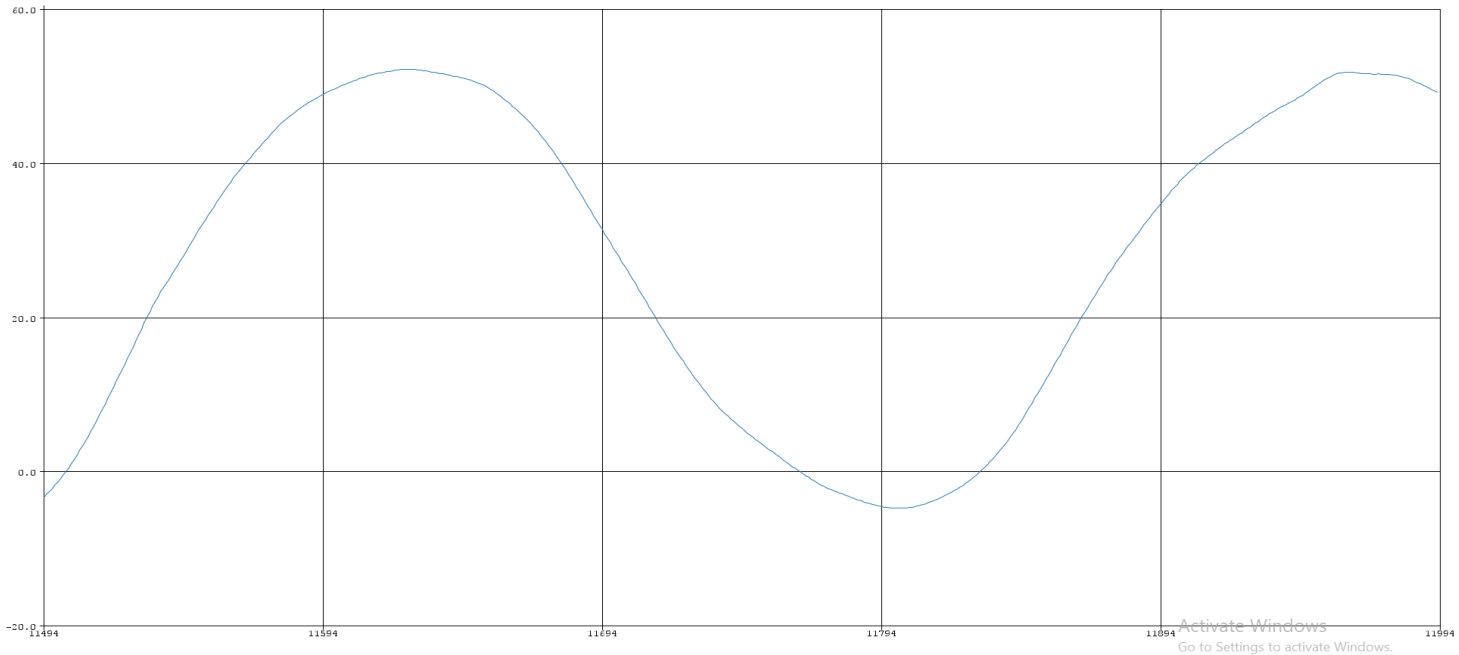


Figure 4.31 - kalman filtered gyroscope readings

As we can see, the angle is changing smoothly without any noise

Pitch (θ)

We used the exact same approach of estimating the roll angle to estimate the pitch angle. Here are the related equations

$$\theta = \int_{0}^{k*Ts} y\,dt \text{ , where } y \text{ is the angular velocity around } y \text{ axis}$$

$$\theta(k) = \theta(k-1) + y * Ts$$

$$\theta = tan^{-1}\frac{x}{\sqrt{y^2+z^2}}$$

Kalman filter steps are

$$\theta = \theta + gyro_y * Ts$$

$$Uncertainty_{pitch} = Uncertainty_{pitch} + \sigma^2_{gyroscope} * Ts^2$$

$$K = \frac{Uncertainty_{pitch}}{Uncertainty_{pitch} + \sigma^2_{accelerometer} * Ts^2}$$

$$\theta = \theta + K * (\theta_{acc} - \theta)$$

$$Uncertainty_{pitch} = (1 - K) * Uncertainty_{pitch}$$

Results

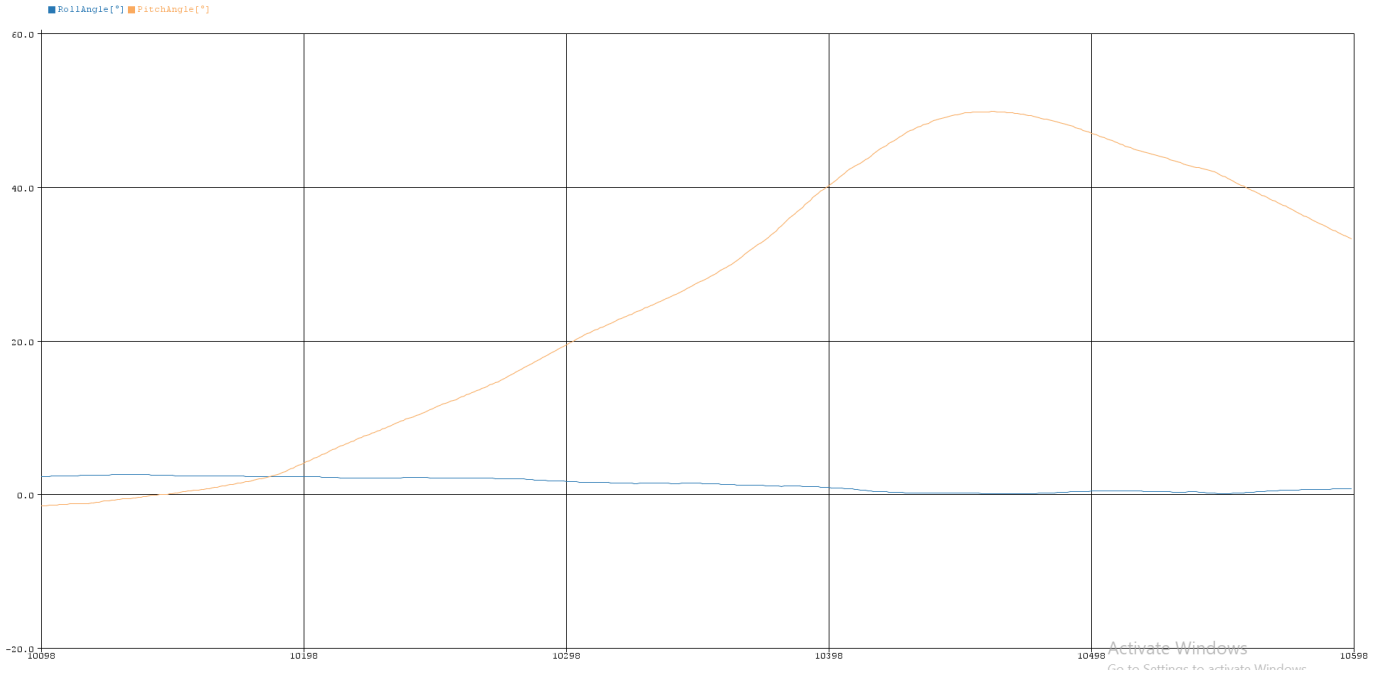


Figure 4.32 - kalman filtered pitch readings

Altitude and Vertical velocity

As mentioned before, we can use the barometer readings of atmospheric pressure to estimate the altitude. Here is the formula

$$altitude = 44330 * [1 - (\frac{pressure}{reference\ pressure})^{0.1903}]$$

The formula assumes (T = 15 ℃)

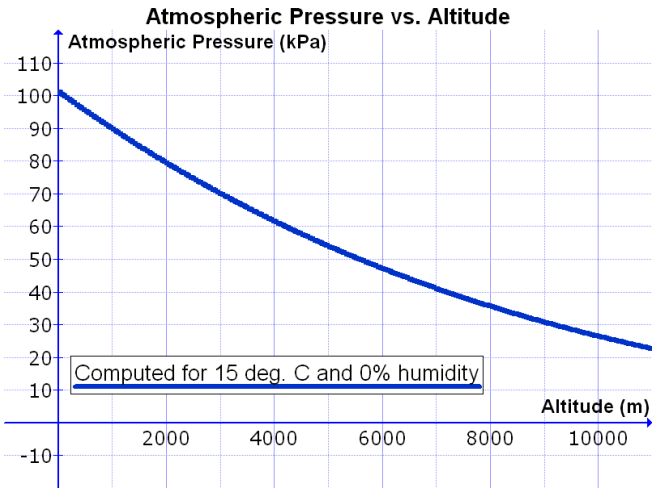


Figure 4.33 - altitude readings

Results

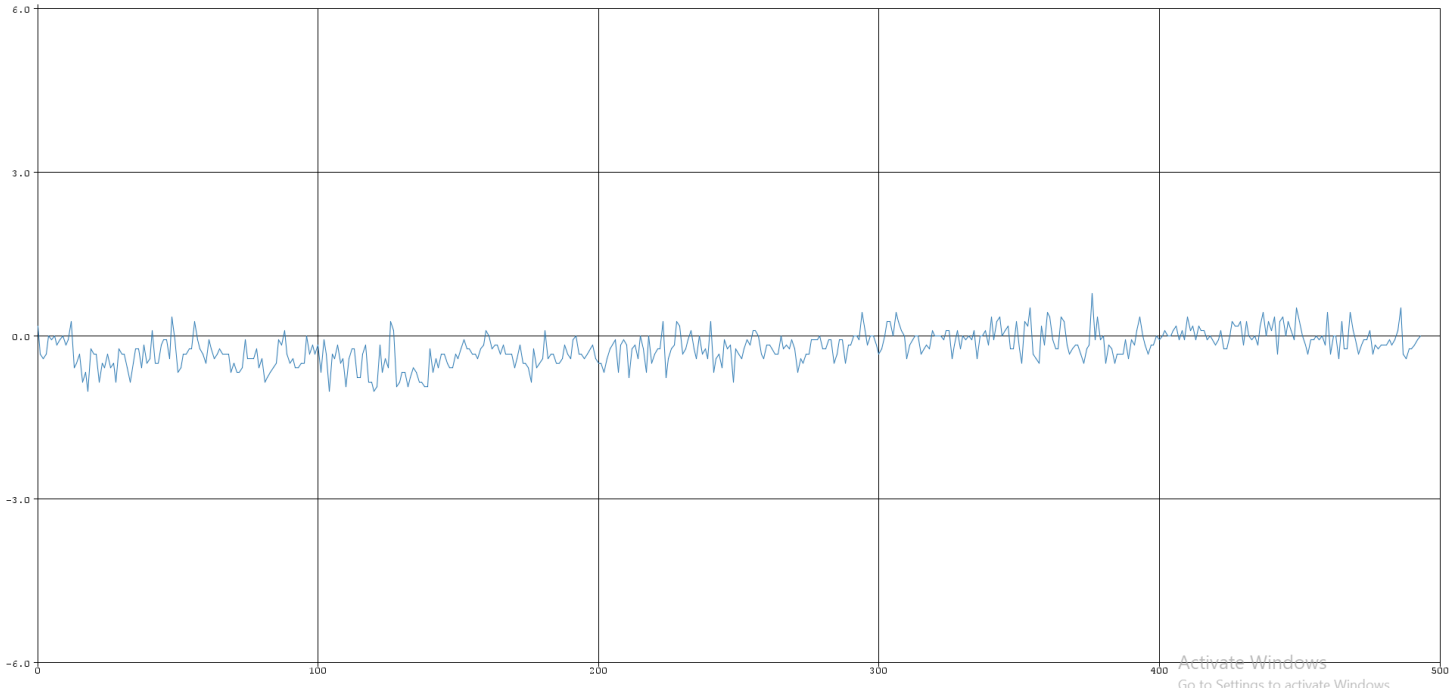


Figure 4.34 - Barometer readings with no motion

As we can see, the barometer estimate of the altitude is very noisy. Thus it cannot be used as a reliable telemetry option on its own. Note that the previous readings are in meters.

To reduce this noise, we need another sensor that can help with measuring the vertical velocity in the inertial frame. If you get the actual rate of change of altitude, you can get a more stable estimate of it.

Let's get back to the accelerometer that can measure the acceleration along the z-axis in the inertial frame using the following transformation.

$$z_{inertial} = z * cos\,\phi * cos\,\theta + y * sin\,\phi * cos\,\theta - x * sin\,\theta$$

If we used this value in a simple integration formula, we will face the same problem as before. The small noise of the inertial-z acceleration will accumulate. It will result in an inaccurate measurement.

You guessed it right, we will use the Kalman filter again. This time the state vector will not be 1D like the Euler angles. We will have a 2D state vector.

State vector: [altitude vertical_velocity]

1. Predict the current state of the system:

$$S(k) = F \cdot S(k-1) + G \cdot U(k)$$

S=state vector $\begin{bmatrix} Altitude_{kalman} \\ Velocity_{kalman} \end{bmatrix}$

F=state transition matrix $\begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$

G=control matrix $\begin{bmatrix} 0.5 \cdot T_s^2 \\ T_s \end{bmatrix}$

U=input variable ($Acc_{Z,inertial}$)

2. Calculate the uncertainty of the prediction:

$$P(k) = F \cdot P(k-1) \cdot F^T + Q$$

P=prediction uncertainty vector

$$P(k=0) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Q=process uncertainty $G \cdot G^T \cdot 10^2$

3. Calculate the Kalman gain from the uncertainties on the predictions and measurements:

$$L(k) = H \cdot P(k) \cdot H^T + R$$

$$K = P(k) \cdot \frac{H^T}{L(k)} = P(k) \cdot H^T \cdot L(k)^{-1}$$

L= Intermediate matrix
K=Kalman gain
H=Observation matrix $\begin{bmatrix} 1 & 0 \end{bmatrix}$
R=Measurement uncertainty (30^2)

4. Update the predicted state of the system with the measurement of the state through the Kalman gain:

$$S(k) = S(k) + K \cdot (M(k) - H \cdot S(k))$$

M=measurement vector
($\text{Altitude}_{\text{barometer}}$)

5. Update the uncertainty of the predicted state:

$$P(k) = (I - K \cdot F) \cdot P(k)$$

I=unity matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Result

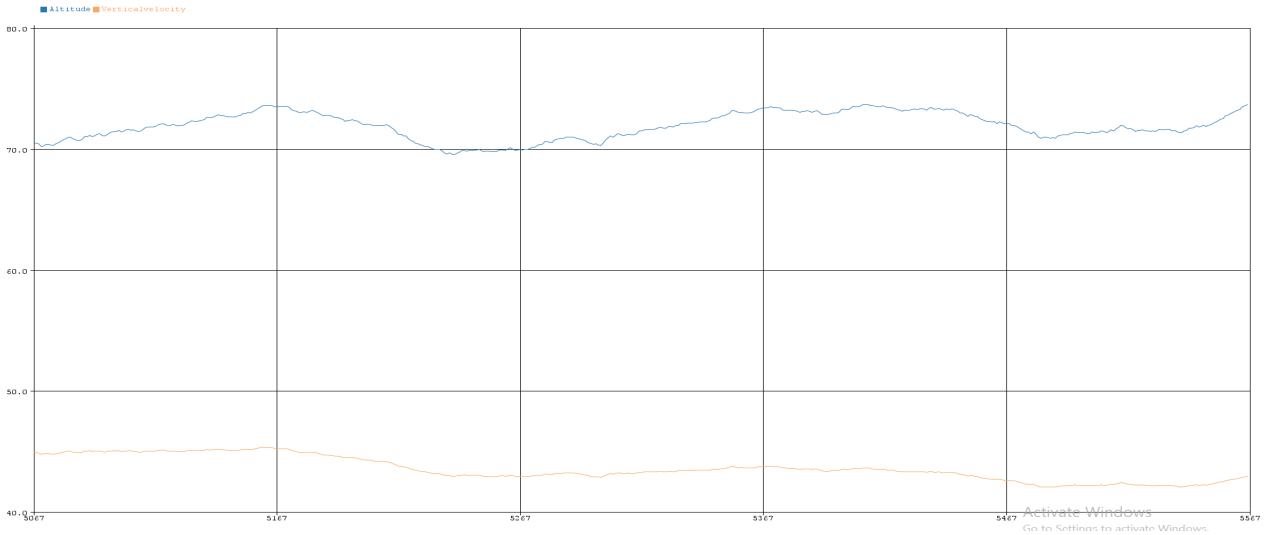


Figure 4.35 - Altitude using 2D-Kalman filter

As we can see, there is a small deviation around +-0.7 m. This is acceptable since the altitude is used for telemetry purposes and not fed into a control algorithm.

4.5. MCAL (Microcontroller Abstraction Layer)

The Microcontroller Abstraction Layer (MCAL) is a critical component of our flight controller system. It provides a standardized interface to various hardware peripherals on the CH32V203 microcontroller, facilitating their integration into the higher-level application code. This abstraction ensures that the application software can interact with the microcontroller's hardware without needing to manage the low-level details directly. The MCAL module includes drivers for essential peripherals such as GPIO, RCC, SPI, UART, I2C, and Timer, each encapsulated with wrappers to offer a consistent and simplified interface.

4.5.1. Functional Description

The MCAL module serves as an interface between the hardware peripherals of the CH32V203 microcontroller and the higher-level application software. Its primary function is to abstract the hardware details and provide standardized APIs for accessing and controlling the peripherals. This abstraction simplifies the development process and ensures portability and scalability of the application code. In the MCAL module, we have utilized wrappers to encapsulate the peripheral drivers. **Wrappers** are intermediary software layers that encapsulate the functionality of peripheral drivers, providing a unified and simplified interface for higher-level software components. These wrappers play a crucial role in the following ways:

1. **Modularity:** Wrappers enhance the modularity of the code by isolating changes in the hardware layer from affecting the application layer. This separation of concerns allows for easier maintenance and updates**.**
2. **Simple Integration:** By using wrappers, the complexity of directly interfacing with the hardware registers is hidden, making it easier for developers to integrate and use various peripherals.
3. **Readability and Maintainability**: The use of wrappers improves code readability. This, in turn, enhances the maintainability of the codebase, facilitating future development and debugging efforts.

The MCAL module includes drivers for essential peripherals such as GPIO, RCC, SPI, UART, I2C, and Timer, each encapsulated with wrappers to provide a consistent and simplified interface.

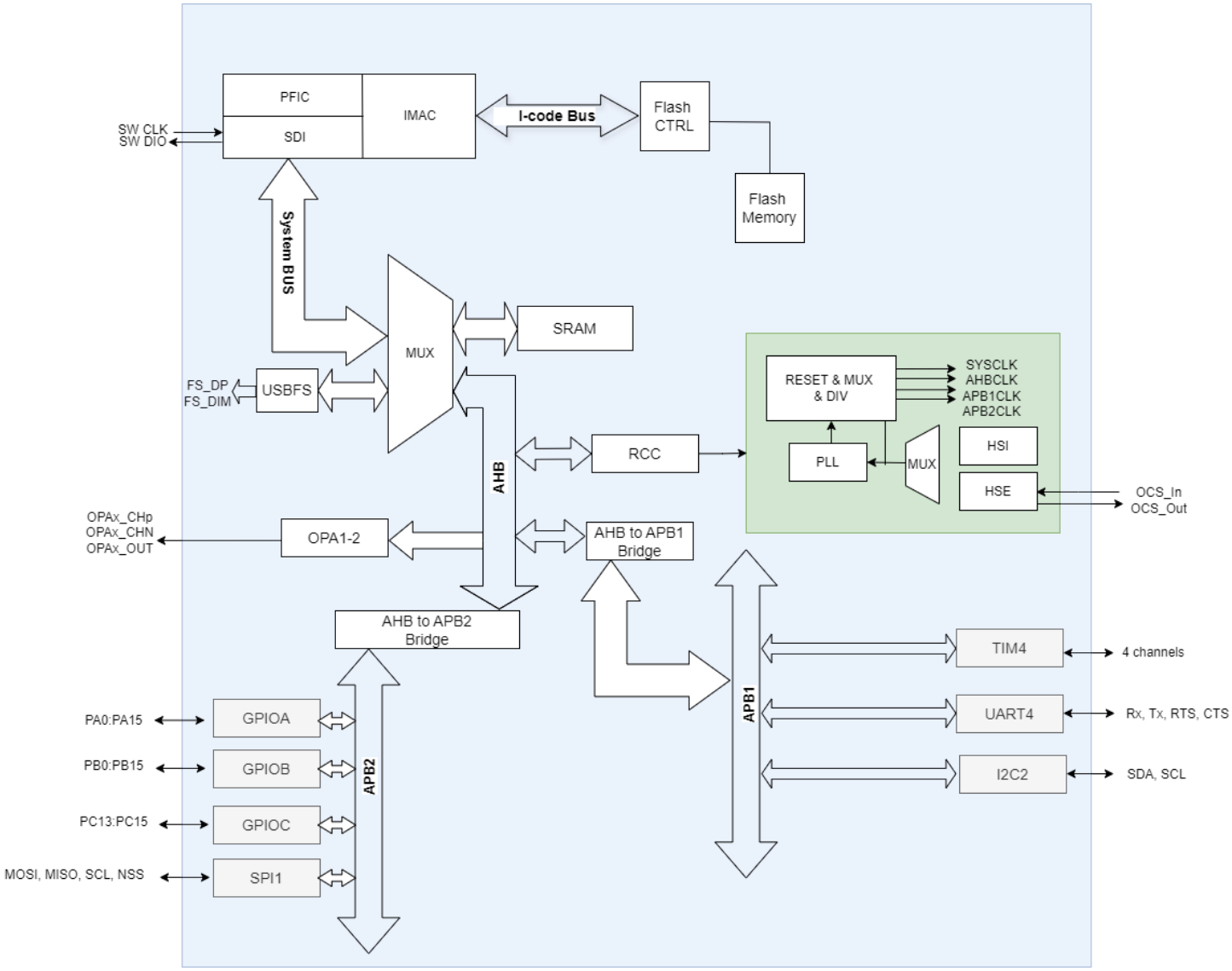


Figure 4.36 - MCU internal architecture

4.5.2. Modular Decomposition

The MCAL module is composed of several sub-modules, each responsible for interfacing with a specific peripheral device of the CH32V203 board.

4.5.2.1. GPIO

- The GPIO (General-Purpose Input/Output) driver is responsible for the configuration and control of the microcontroller's GPIO pins. GPIO pins are versatile and can be configured as inputs or outputs, allowing them to read signals from sensors or drive signals to actuators. This flexibility is fundamental to the operation of the flight controller, enabling it to interact with various hardware components.

- **Features**
    1. Configurable pin modes (input, output, alternate function)
    2. Configurable output types (push-pull, open-drain)
    3. Internal pull-up and pull-down resistors
    4. Adjustable drive strength (speed)
    5. Input data reading capability
    6. Bit set/reset for output control

- **Pin Modes**
    1. **Input Mode:**
        - **Digital Input:** The pin can read digital signals.
        - **Analog Input:** The pin can read analog signals (used with on-chip ADC).
        - **Alternate Function Input:** The pin can be used as an input for on-chip peripherals such as UART, SPI, I2C, etc.
    2. **Output Mode:**
        - **Digital Output:** The pin can drive digital signals.
        - **Alternate Function Output:** The pin can output signals generated by on-chip peripherals.
    3. **Alternate Function Mode:**
        - This mode allows the pin to function as an interface for on-chip peripherals (e.g., UART, SPI).

- **Output Types**
    1. **Push-Pull:**
        - **Description**: The pin can actively drive both high and low states.
        - **Usage**: Suitable for standard digital outputs where the pin needs to drive a load to both logic levels.
    2. **Open-Drain:**
        - **Description**: The pin can only drive a low state; an external pull-up resistor is required to achieve a high state.
        - **Usage**: Ideal for bus systems like I2C where multiple devices can pull the line low but require an external pull-up to achieve high state.
- **Internal Resistors**
    1. **Pull up resistors:**
        - **Description**: An internal resistor connected between the pin and Vcc, ensuring the pin reads high when not driven.
        - **Usage**: Prevents floating states and unintended low readings.
    2. **Pull-Down Resistor:**
        - **Description**: An internal resistor connected between the pin and GND, ensuring the pin reads low when not driven.
        - **Usage**: Prevents floating states and unintended high readings.
- **Drive Strength (Speed):**
    - **Low Speed(2MHz):** Suitable for low-frequency applications where power consumption is a concern.
    - **Medium Speed(10MHz):** Balances power consumption and speed for general-purpose applications.
    - **High Speed(50MHz):** Required for high-frequency applications where fast signal transitions are necessary.

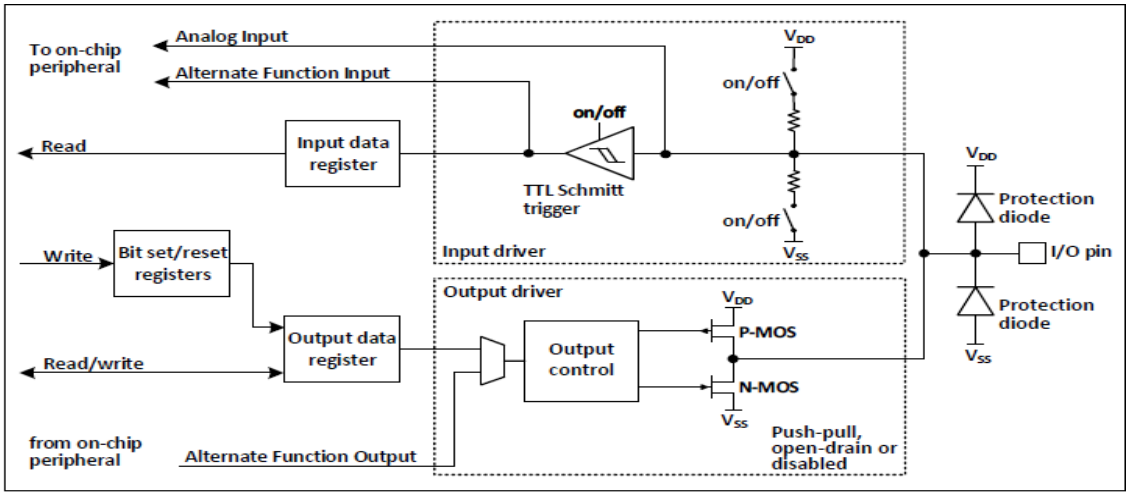


Figure 4.37 - MCU internal GPIO pin schematic

1. **Input Path:**
    - Analog Input: The pin can be used to read analog signals if configured as an analog input.
    - Alternate Function Input: The pin can serve as an input for on-chip peripherals (e.g., SPI, UART) when configured in alternate function mode.
    - Input Data Register: Stores the current state of the GPIO pin, which can be read by software.
2. **Output Path:**
    - Bit Set/Reset Registers: Allows software to set or reset the pin state.
    - Output Data Register: Holds the state of the GPIO pin when configured as an output.
    - Output Driver: Manages the pin's drive capability based on the configured mode (push-pull or open-drain).
    - Output Control: Switches the pin state based on the output data register.

3. **Control and Configuration:**
    - Read/Write Access: Software can read from or write to the input and output data registers, respectively.
    - Alternate Function Output: The pin can output signals for on-chip peripherals when configured in alternate function mode.

4.5.2.2. RCC

The RCC (Reset and Clock Control) module in the CH32V203C8T6 board is responsible for managing the clock and reset functionalities of the device. It offers a wide range of features, including:

1. Clock source selection and configuration
2. Clock prescaling and division
3. Peripheral clock enable/disable control
4. System reset control
5. Power management through clock gating

1. **Clock Source:**
   - **Internal Oscillator (HSI)**: Provides a stable clock source from an internal RC oscillator. Suitable for applications where an external clock source is not necessary.
   - **External Oscillator (HSE):** Utilizes an external crystal or resonator for higher accuracy and stability. Ideal for applications requiring precise timing.
2. **Clock Domains:**
   - **AHB (Advanced High-performance Bus)**: The main system bus, which connects the CPU to high-speed peripherals and memory.
   - **APB1 (Advanced Peripheral Bus 1)**: A lower-speed bus for connecting slower peripherals.
   - **APB2 (Advanced Peripheral Bus 2)**: Another lower-speed bus for connecting slower peripherals but generally faster than APB1.
3. **Reset Control:**

   Controls system reset sources and flags, including software reset, watchdog timer reset and power-on reset.

4.5.2.3. SPI

The Serial Peripheral Interface (SPI) module in the CH32V203C8T6 microcontroller is designed for high-speed, synchronous, serial communication between the microcontroller and external devices. The SPI module is used to interface between the microcontroller and the NRF transceiver as well as the BMP280 barometer. The SPI is configured to operate in 2-line full-duplex mode, with the microcontroller acting as the master and the external hardware devices (NRF transceiver and BMP280) as slaves.

- **Features**
    1. Full-duplex and half-duplex communication
    2. Master and slave mode operation
    3. Configurable clock polarity (CPOL) and clock phase (CPHA)
    4. Configurable data frame size (8-bit, 16-bit, etc.)
    5. Multiple baud rate prescaler options such as 2, 4, 16, 32, etc.
    6. Hardware-based chip select management

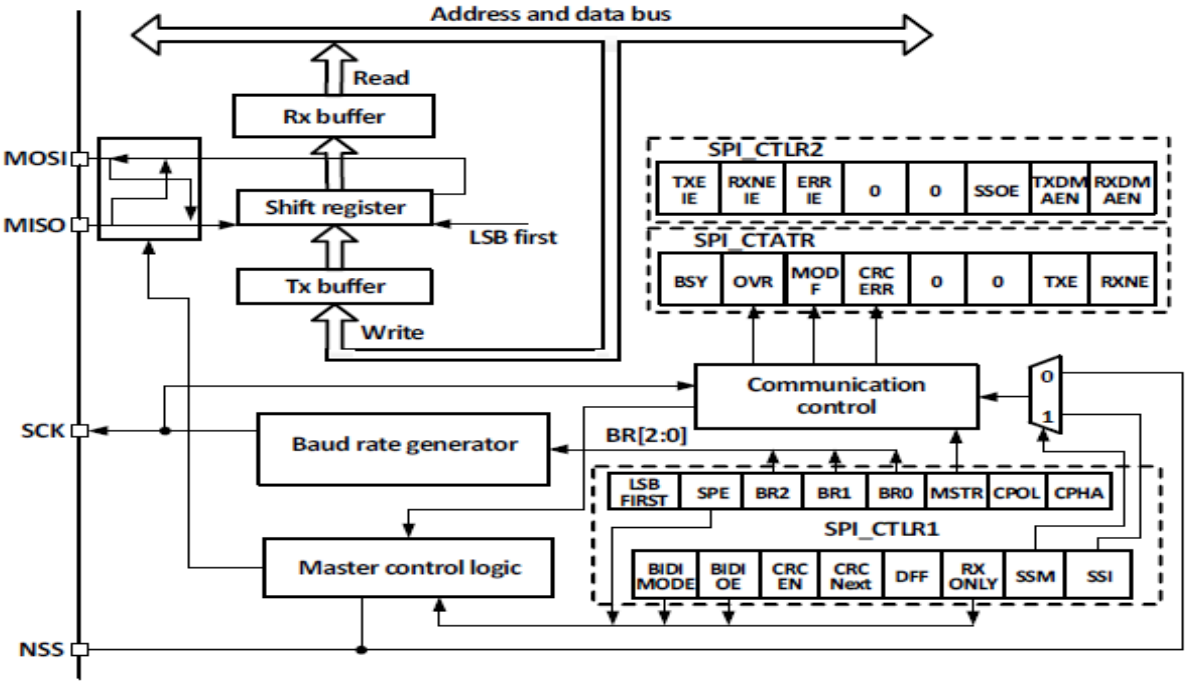


Figure 4.38 - MCU internal SPI pin schematic

- At the core of the SPI module is the master control logic, which determines the operation mode (master or slave) and manages overall functionality. The baud rate generator, influenced by settings in the SPI_CTLR1 register, generates the SPI clock (SCK) essential for synchronizing data transfer. Data flow is handled by the shift register, which performs serial-to-parallel and parallel-to-serial conversions, interfacing with the Tx (Transmit) and Rx (Receive) buffers for data storage. These buffers work in tandem with the MOSI (Master Out Slave In) and MISO (Master In Slave Out) lines to facilitate full-duplex communication. Control registers SPI_CTLR1 and SPI_CTLR2 configure various parameters such as clock polarity (CPOL), clock phase (CPHA), data frame size, and baud rate, while also enabling interrupts and DMA. The communication control block ensures proper data flow management and synchronization, with status flags in the SPI_STATR (Status Register) indicating conditions like transmit buffer empty (TXE) and receive buffer not empty (RXNE). The master control logic interacts with the NSS (Slave Select) pin to select the appropriate slave device, ensuring correct data routing.

4.5.2.4. UART

The Universal Asynchronous Receiver-Transmitter (UART) facilitates data transfer between the microcontroller and other devices. We used UART4 for communication between the application board and the drone board, employing hardware flow control using CTS (Clear to Send) and RTS (Request to Send) signals to ensure reliable data transmission.

- **Features**
    1. Full-duplex communication
    2. Configurable baud rate such as 9600 and 115200
    3. 8 or 9 data bits
    4. Parity bit generation and detection
    5. Hardware flow control using CTS and RTS

1. **Clear to Send (CTS):**

    The CTS signal is an input to the UART module, indicating whether the external device (e.g., drone board) is ready to receive data. When the CTS signal is asserted (low), the UART can proceed with data transmission. If CTS is deasserted (high), the UART will pause transmission until CTS is asserted again.

2. **Request to Send (RTS):**

    The RTS signal is an output from the UART module, indicating to the external device that the UART is ready to receive data. When the UART's receive buffer is not full, RTS is asserted (low). If the receive buffer becomes full, RTS is deasserted (high), signaling the external device to pause data transmission until the buffer is ready to accept more data.

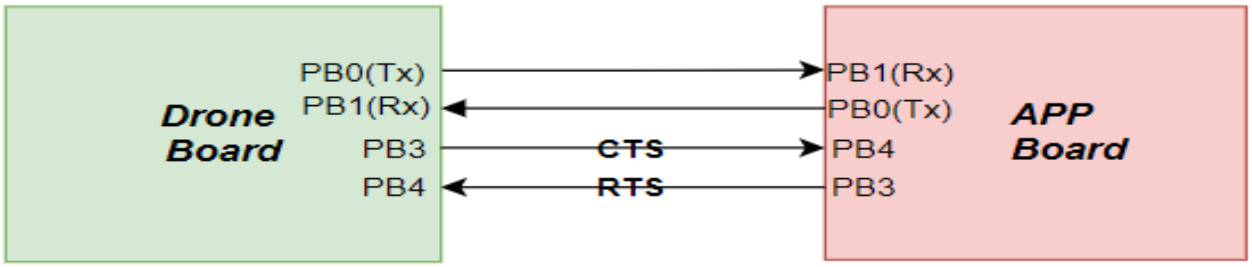


Figure 4.39 - MCU interface via UART schematic

4.5.2.5. I2C

The Inter-Integrated Circuit (I2C) module facilitates the serial communication with multiple hardware devices using just two wires SDA (Serial Data Line) and SCL (Serial Clock Line). The I2C module supports multi-master and multi-slave communication, making it suitable for connecting multiple sensors and devices on the same bus.

The I2C interface is used to communicate with sensors such as the MPU-6050 (an accelerometer and gyroscope) and the HMC5883L (a magnetometer).

- **Features:**
    1. Multi-master and multi-slave configuration
    2. Standard mode (100 kHz) and fast mode (400 kHz) communication
    3. 7-bit and 10-bit addressing modes
    4. Clock stretching for flow control

In I2C mode, data exchange begins with the master (the drone board) generating a start condition by pulling SDA low while SCL is high. The master then sends the 7-bit or 10-bit address of the slave device, followed by a read/write bit. The slave device with the matching address acknowledges (ACK) by pulling SDA low. During a write operation, the master sends data bytes to the slave, each followed by an ACK from the slave, continuing until all data bytes are transmitted. In a read operation, the master receives data bytes from the slave, sending an ACK after each byte except for the last byte, where the master sends a NACK control signal to indicate the end of reception. The communication ends by the master generating a stop condition by releasing SDA high while SCL is high.

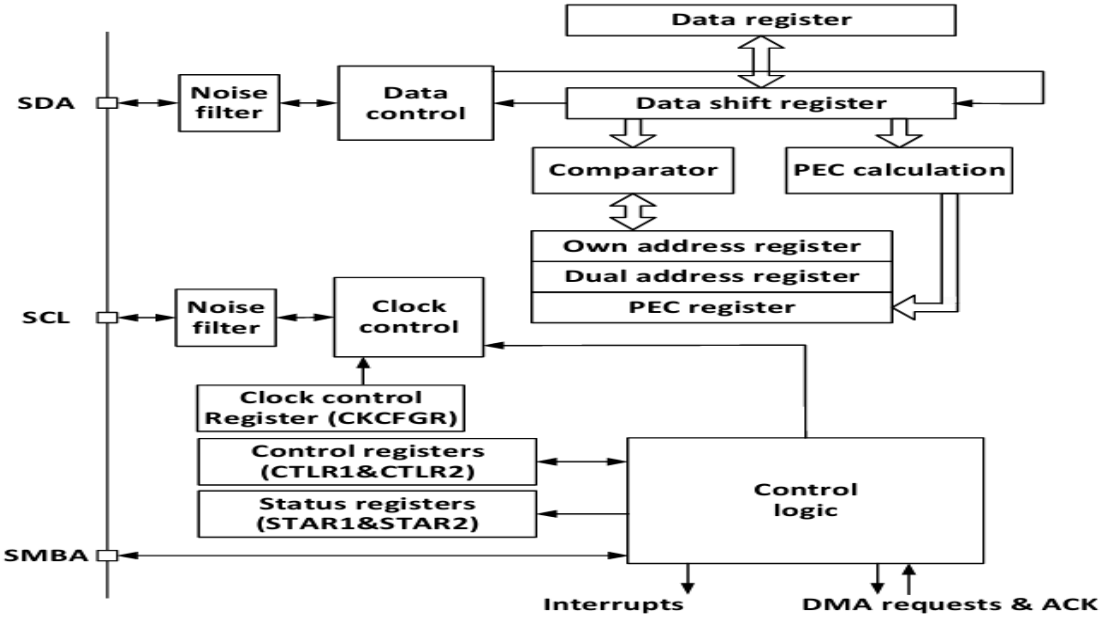


Figure 4.40 - MCU internal SPI schematic

4.5.2.6. Timer

The timers module is used for generating precise time delays and producing Pulse Width Modulation (PWM) signals. We used Timer 4 in center-aligned mode to generate PWM signals for driving the Electronic Speed Controllers (ESCs) that control the four motors of the drone.

- **Features:**
    1. Up, down, and center-aligned counting modes
    2. Configurable prescaler and auto-reload values
    3. Multiple channels for PWM generation
    4. Input capture and output compare functionalities

We used Timer 4 in center-aligned mode in motor control for the following two reasons:

1. The center-aligned mode ensures that the PWM is symmetrical and helps in reducing the potential noise that affects the motor performance.
2. The precise and stable nature of center-aligned PWM provides smoother and more accurate control of motor speed and torque, improving the overall stability and performance of the drone.

4.5.3. Design Constraints

4.5.3.1. GPIO

When designing a GPIO (General-Purpose Input/Output) driver. These constraints cover electrical characteristics, timing requirements and system-Level Integration.

1. Electrical constraints:
    - The GPIO pins must operate within the specified voltage levels. Exceeding these levels can damage the microcontroller or connected devices. Typically, GPIO pins support 3.3V or 5V logic levels.
2. Timing Requirements:
    - Implementing software debouncing is necessary to ensure reliable input readings because mechanical switches connected to GPIO pins can cause bouncing
3. System-Level integration:
    - Many GPIO pins can serve multiple functions (e.g., alternate functions for peripherals like UART, SPI). Careful planning is required to avoid conflicts and ensure that all required functionalities are available.

4.5.3.2. RCC

- Different applications require different levels of clock accuracy and stability. Internal oscillators (HSI) offer convenience but may not be as accurate as external crystal oscillators (HSE). Critical applications may require high-stability clocks like external crystals.

- The system clock frequency must be chosen to balance performance and power consumption. Higher frequencies improve performance but increase power consumption and heat generation.
- Suitable prescaler values must be selected to achieve the desired peripheral clock frequencies.

4.5.3.3. SPI

- The SPI clock frequency must be within the range supported by both the master and slave devices. Exceeding the maximum frequency of any device can lead to communication failures.
- Configuration of CPOL and CPHA is essential to ensure data is correctly sampled and shifted. Mismatched settings between master and slave can cause data corruption.
- Determining whether full-duplex or half-duplex communication is required for the application. Full-duplex allows simultaneous transmission and reception, while half-duplex does not.
- Management of the NSS (chip select) signal is crucial for selecting the correct slave device. In multi-slave systems, ensuring that only one slave is active at a time is necessary to prevent bus contention.

4.5.3.4. UART

- The selected baud rate must be supported by both the application board and the drone board.
- The data size matches between the two boards.The UART driver must support the required data frame size (e.g., 8-bit, 9-bit) for the application. Incorrect data frame size can lead to misaligned data and miss some data
- Configuring the GPIO pin carefully as many GPIO pins can serve multiple functions. So we must ensure that the selected UART pins don’t conflict with the other functions.

4.5.3.5. I2C

- The I2C clock frequency must be compatible with both the CH32V203C8T6 microcontroller and the sensors. Standard mode (100 kHz) and fast mode (400 kHz) are commonly used. Ensure the chosen frequency does not exceed the maximum supported by any device on the bus.
- Ensuring that the timing of start, stop, and repeated start conditions is accurate.
- Ensure that the I2C driver supports the addressing mode used by the sensors. Most sensors, including the MPU-6050 and HMC5883L, use 7-bit addressing.
- Ensure that the selected I2C pins do not conflict with other necessary functions or peripherals as the GPIO pin may contain more that one function

4.5.3.6. Timer

- The PWM frequency must be within the acceptable range for the ESCs and motors. Typical frequencies range from 1 kHz to 32 kHz. Too low a frequency can cause audible noise and inefficient motor control.
- The resolution of the PWM signal affects the granularity of motor speed control. Higher resolution provides finer control but requires a higher timer clock frequency.

4.5.4. Other Description of MCAL

The Microcontroller Abstraction Layer (MCAL) module provides a standardized interface between the hardware peripherals of the CH32V203C8T6 microcontroller and the higher-level application software, ensuring portability and scalability. This module includes drivers for GPIO, RCC, SPI, UART, I2C, and Timer 4, each encapsulated with wrappers to enhance modularity and maintainability. Wrappers serve as intermediary layers that hide the complexity of the underlying hardware, offering a simplified and consistent interface for peripheral interaction. This approach improves code readability, eases integration, and allows for efficient updates and maintenance, making the MCAL module a crucial component for the reliable operation of the flight controller system.

4.6. System Modeling

4.6.1 Description

System modeling is a critical component of developing an effective flight control system for drones. It involves creating mathematical and computational representations of the drone's dynamics and behavior, which are essential for understanding and predicting how the drone will respond to various inputs and environmental conditions. This comprehensive approach allows us to dissect and analyze the complex interactions between the drone's numerous subsystems, including aerodynamic, mechanical, and electronic components.

At the core of system modeling lies the development of accurate mathematical representations of the drone's physical properties. This includes defining the equations of motion that govern the drone's translational and rotational dynamics. These equations typically take the form of differential equations that describe how forces and torques act on the drone's body, leading to changes in its velocity and orientation over time. Additionally, state-space representations are often employed to encapsulate the system's state variables and their evolution, providing a structured framework for analysis and control design.

A crucial aspect of system modeling is the characterization of the drone's propulsion system. This encompasses the dynamics of the motors and propellers, which convert electrical power into thrust and torque. Understanding the aerodynamic principles behind propeller performance, including factors such as thrust coefficient and efficiency, is vital for accurately modeling the propulsion system's response to control inputs. Furthermore, the interaction between the drone's structure and aerodynamic forces, including lift, drag, and moments, must be thoroughly examined to predict the overall behavior of the vehicle.

Environmental interactions also play a significant role in system modeling. External forces such as wind, turbulence, and gusts can significantly impact the drone's stability and performance. Modeling these environmental influences requires sophisticated techniques to simulate their stochastic nature and incorporate them into the overall system dynamics. By doing so, we can design control algorithms that are robust to such disturbances, ensuring reliable operation in real-world conditions.

Parameter identification and validation are indispensable steps in the modeling process. Experimental data obtained from test flights and controlled experiments are used to estimate the values of key parameters within the mathematical models. This data-driven approach enhances the fidelity of the models, making them more representative of the actual system. Validation involves comparing the model's predictions with observed behavior to ensure its accuracy and reliability. Through iterative refinement, the models are honed to provide a precise representation of the drone's dynamics.

In this section, we will explore the various methodologies and tools employed in system modeling. We will delve into the mathematical formulations, simulation environments, and experimental techniques

that collectively contribute to the development of robust models. By establishing a comprehensive system model, we lay the groundwork for designing advanced control strategies. These strategies will enable our drone to perform complex tasks with high precision and efficiency, ultimately contributing to the success of our flight control system.

4.6.2. Mathematical Formulation and Modeling Techniques

4.6.2.1. Equations of Motion

Description: Equations of motion, such as the Newton-Euler equations, are fundamental in describing the physical dynamics of the drone. They articulate how forces and torques interact with the drone's mass and geometry to produce translational and rotational motion.

Advantages:

- **Fundamental Representation:** Provides a clear physical basis for understanding how forces and torques influence motion.
- **Versatility:** Can be applied to both linear and nonlinear systems.
- **Initial Design:** Useful for preliminary design and static stability analysis.

Limitations:

- **Complexity:** Becomes challenging for drones with multiple degrees of freedom or complex geometries.
- **Simplifying Assumptions:** Relies on assumptions such as rigid body dynamics and negligible aerodynamic effects at low speeds.

Applications:

- Ideal for initial conceptual design and understanding static equilibrium.
- Used in early stages of design for estimating forces and moments acting on the drone.

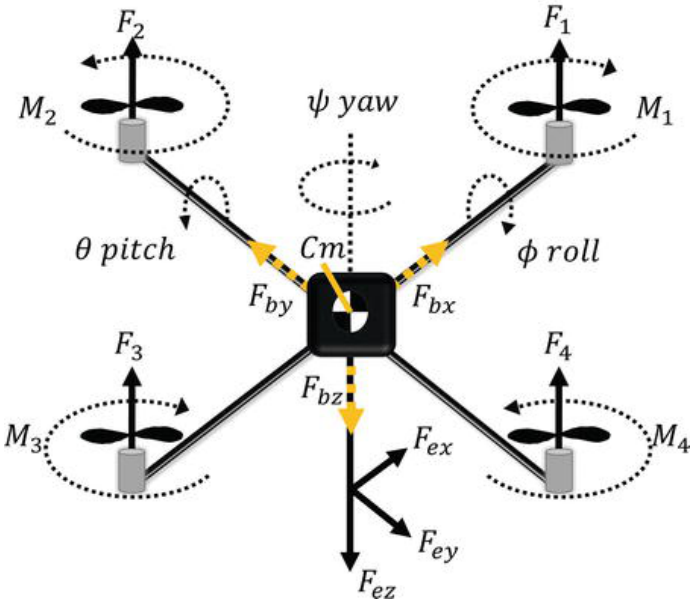


Figure 4.41 - equation of motion

4.6.2.2. State-Space Representation

Description: State-space models represent the drone's dynamics in terms of state variables, input variables, and their relationships. These models provide a systematic framework for describing both linear and nonlinear system behaviors over time.

Advantages:

- **Compact Representation:** Offers a concise representation of complex dynamics.
- **Control System Design:** Facilitates the design of robust control systems using modern control theory.
- **Real-Time Applications:** Suitable for real-time simulation and control implementation.

Limitations:

- **Complexity:** Requires careful selection and approximation of state variables and dynamics.
- **Sensitivity to Parameters:** Sensitive to uncertainties in initial conditions and parameter estimation.

Applications:

- Used extensively in control system design and analysis.
- Suitable for dynamic simulations and real-time implementation of feedback control algorithms.

$$\dot{\mathbf{q}} = \mathbf{A}\mathbf{q} + \mathbf{B}u \qquad \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -a_4 & -a_3 & -a_2 & -a_1 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ b_0 \end{bmatrix} x$$

$$y = \mathbf{C}\mathbf{q} + Du \qquad y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

4.6.2.3. Propulsion System Modeling

Description: Propulsion system modeling focuses on understanding and predicting the behavior of motors and propellers. This involves modeling the electrical, mechanical, and aerodynamic characteristics that determine thrust and torque generation.

Advantages:

- **Accuracy:** Models motor and propeller dynamics precisely, crucial for optimizing flight performance.
- **Efficiency:** Allows optimization of power consumption and flight endurance.
- **Customization:** Can be tailored to different motor and propeller configurations.

Limitations:

- **Complexity:** Requires detailed knowledge of motor and propeller characteristics.
- **Data Dependency:** Highly dependent on accurate experimental data for parameter estimation.

Applications:

- Essential for designing efficient propulsion systems and predicting flight performance.
- Used in developing control algorithms that regulate thrust and stabilize the drone during flight.

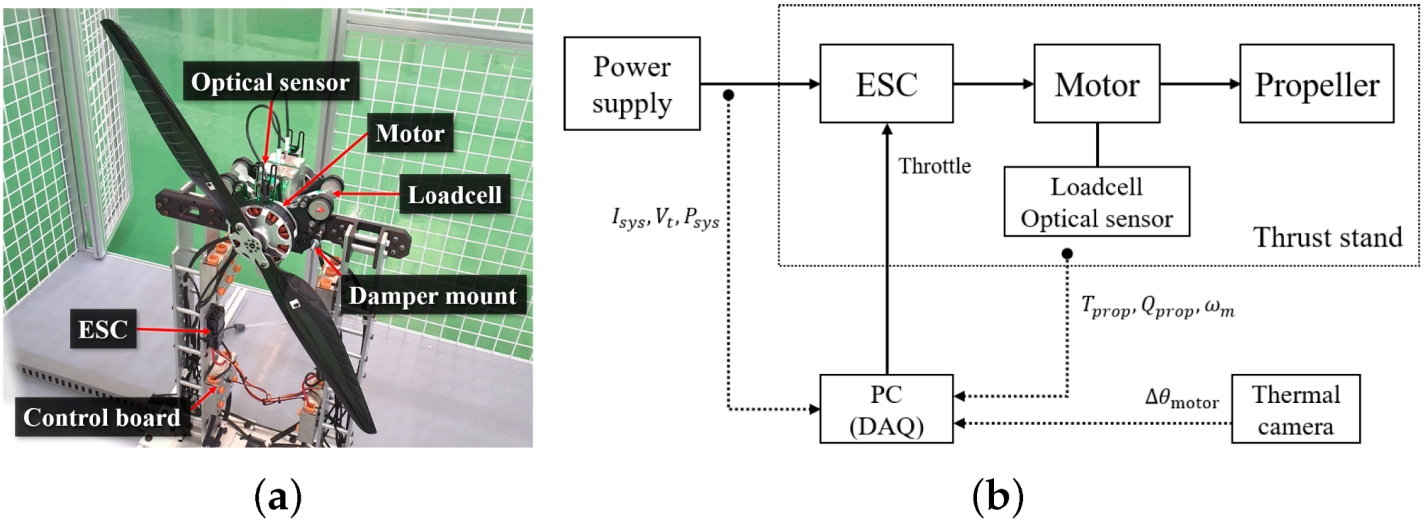


Figure 4.42 - thrust modeling

4.6.2.4. Aerodynamic Forces and Moments

Description: Aerodynamic modeling quantifies the forces (lift and drag) and moments (pitch, roll, and yaw) acting on the drone due to airflow. These models are based on aerodynamic principles and empirical data.

Advantages:

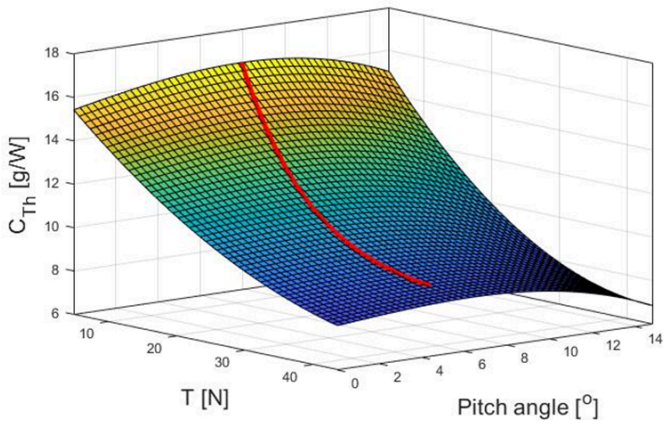
- **Realism:** Provides a realistic representation of aerodynamic interactions.
- **Stability Analysis:** Essential for assessing stability and maneuverability.
- **Design Optimization:** Guides airframe design and control surface placement.

Limitations:

- **Complexity:** Requires understanding of aerodynamic theory and empirical data for accurate modeling.
- **Computational Intensity:** Simulation of aerodynamic effects can be computationally demanding.

Applications:

- Crucial for designing drones optimized for specific flight conditions (e.g., high-speed, maneuverability).
- Used in developing control algorithms that account for aerodynamic forces to maintain stability and control.



The red curve on the graph was drawn to show the maximal efficiency which can be obtained for a specific lift force with the best pitch angle setting.

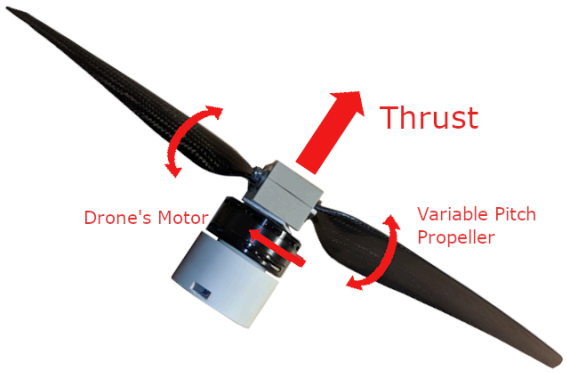


Figure 4.43 - forces in aerodynamics modeling

4.6.2.5. Environmental Interactions

Description: Environmental interaction modeling involves predicting the effects of external factors like wind, turbulence, and gusts on drone flight. These models simulate stochastic disturbances that impact flight stability and performance.

Advantages:

- **Realism:** Models environmental disturbances realistically.
- **Robust Control:** Enables the design of robust control strategies that mitigate disturbances.
- **Safety Enhancement:** Improves safety by predicting and adapting to adverse conditions.

Limitations:

- **Uncertainty:** Environmental models may introduce uncertainties that are challenging to quantify.
- **Validation Requirements:** Requires extensive validation against real-world data.

Applications:

- Essential for designing drones capable of operating safely in varied environmental conditions.
- Used in autonomous navigation systems to ensure stability and performance reliability.

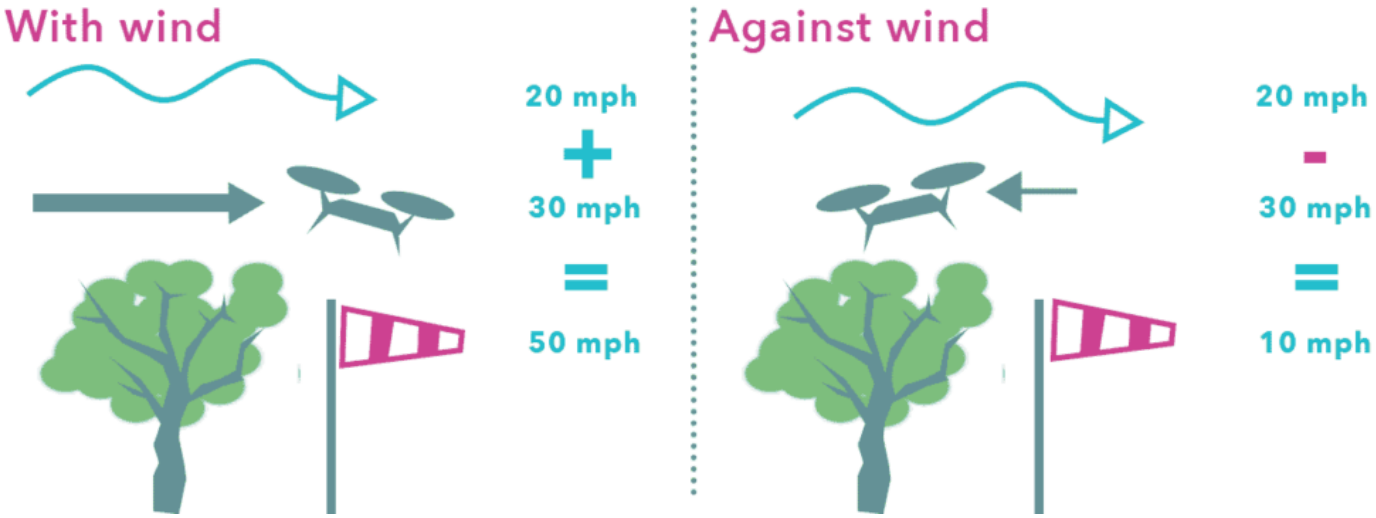


Figure 4.44 - wind forces in aerodynamics modeling

4.6.2.6. Parameter Identification and Validation

Description: Parameter identification involves estimating model parameters based on experimental data, while validation assesses how well the model predictions match real-world observations.

Advantages:

- **Accuracy Improvement:** Enhances model accuracy by fitting parameters to empirical data.
- **Reliability:** Validates model predictions against actual flight performance.
- **Iterative Refinement:** Allows iterative improvement of models based on empirical findings.

Limitations:

- **Data Dependency:** Relies on accurate experimental data, which can be costly and time-consuming to collect.
- **Complexity:** Parameter estimation techniques may require expertise in system identification.

Applications:

- Crucial for refining and validating mathematical models of drone dynamics.
- Used in developing models that accurately represent the drone's behavior under operational conditions.

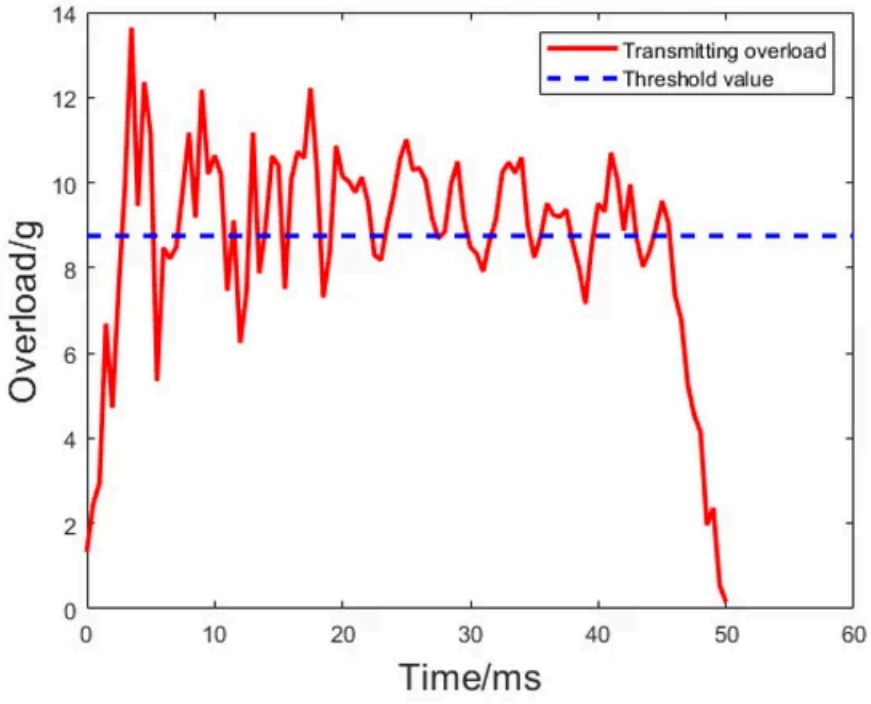


Figure 4.45 - overload with respect to time

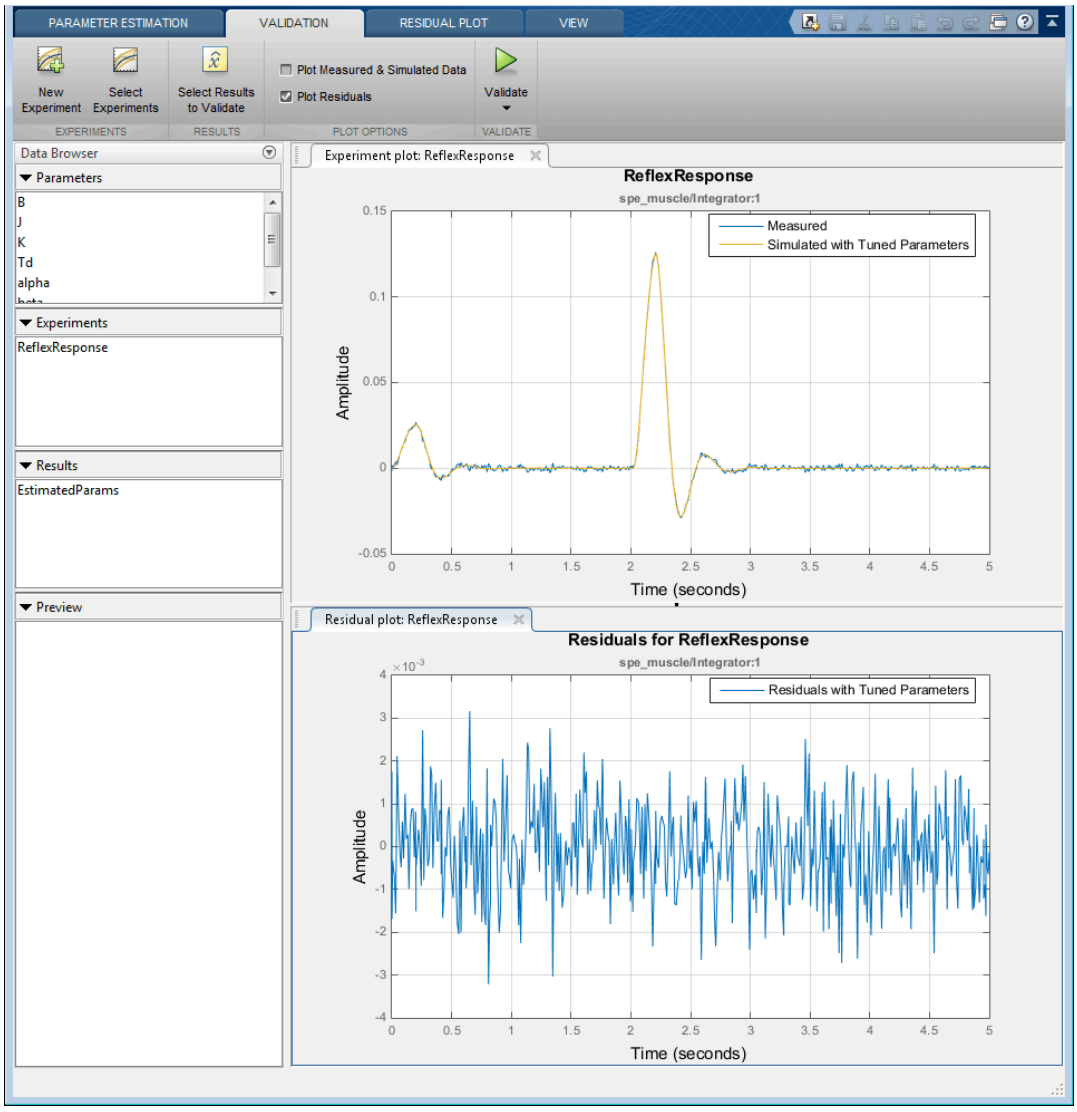
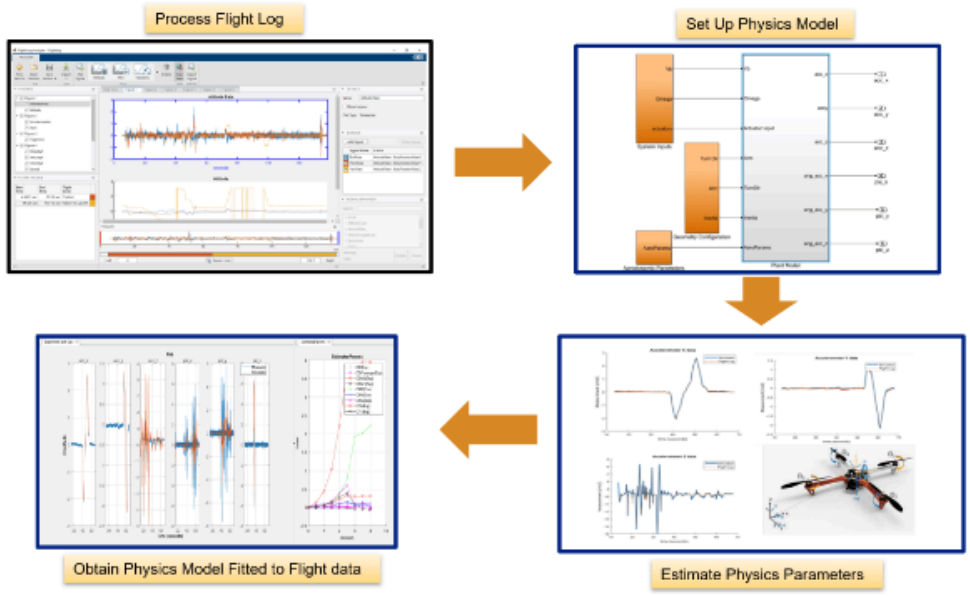


Figure 4.46 - system response



4.6.3. Simulation and Analysis

4.6.3.1. Simulation Environments

Description: Simulation environments are software platforms that replicate the real-world conditions in which the drone will operate. These environments can range from simple desktop-based simulators to complex, high-fidelity virtual environments that closely mimic physical reality. Popular simulation tools for drone modeling include MATLAB/Simulink, Gazebo, and ROS (Robot Operating System).

Pros:

- **Safety:** Testing in a virtual environment eliminates the risk of physical damage to the drone, making it safer to test extreme scenarios.
- **Cost-Effective:** Virtual testing reduces the need for multiple physical prototypes and extensive flight testing, saving time and resources.
- **Versatility:** Simulations can easily be adjusted to test a wide range of conditions, from different weather patterns to varying payload weights and mission profiles.

Cons:

- **Accuracy:** Simulations may not capture all real-world complexities, such as unpredictable environmental conditions or unmodeled interactions.
- **Resource Intensive:** High-fidelity simulations, especially those involving detailed aerodynamic modeling or real-time physics, can require significant computational power.

When to Use:

- Use simulation environments during the initial design phase to test basic functionality and feasibility.
- Ideal for iterative testing and refinement of control algorithms before deploying them on actual hardware.

4.6.3.2. Scenario Analysis

Description: Scenario analysis involves subjecting the drone to various operational scenarios within the simulation environment to evaluate its performance and robustness. These scenarios can include different weather conditions, obstacle avoidance tasks, varying payloads, and mission-specific activities such as search and rescue operations or delivery missions.

Pros:

- **Comprehensive Testing:** Ensures the drone is tested under a wide range of conditions, providing a thorough evaluation of its capabilities.
- **Performance Insights:** Provides valuable data on the drone's strengths and weaknesses, guiding further development and optimization.
- **Improved Design:** Helps identify potential issues and design flaws early in the development process, allowing for timely corrections.

Cons:

- **Time-Consuming:** Running multiple scenarios can be time-intensive, particularly if the simulations are complex.
- **Data Overload:** Can generate large amounts of data that need to be carefully analyzed and interpreted.

When to Use:

- Use scenario analysis to validate the drone's performance across different operational conditions, ensuring it meets design specifications and user requirements.
- Essential for mission-specific drones that need to perform reliably in particular environments.

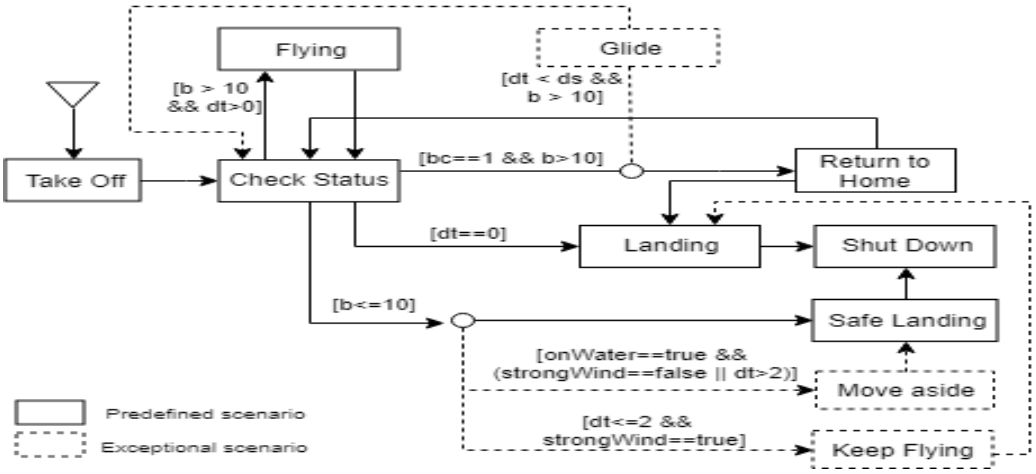


Figure 4.47 - Scenario Analysis

4.6.3.3. Validation of Simulation Results

Description: Validation involves comparing the results from simulations with experimental or real-world data to ensure the accuracy and reliability of the models. This process is critical for building confidence in the simulation outcomes and for making necessary adjustments to improve model fidelity.

Pros:

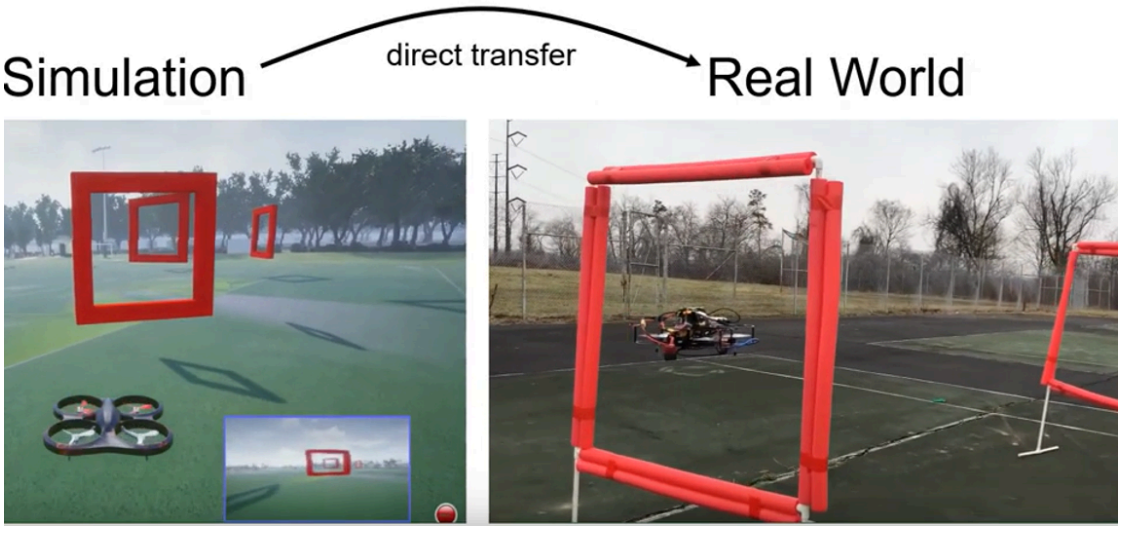
- **Reliability:** Increases trust in the simulation models and their predictions by ensuring they accurately reflect real-world behavior.
- **Model Improvement:** Helps refine models by identifying and correcting discrepancies between simulated and actual data.
- **Regulatory Compliance:** Can be necessary for meeting regulatory and safety standards, which often require proof of model validation.

Cons:

- **Data Requirement:** Requires extensive experimental data for validation, which can be costly and time-consuming to collect.
- **Complexity:** Can be complex and require significant effort to match simulation and real-world data accurately.

When to Use:

- Use validation during and after the development of simulation models to ensure they accurately represent the drone's behavior.
- Critical for ensuring the simulation models are reliable for predicting real-world performance and for regulatory compliance.



4.6.3.4. Performance Metrics and Analysis

Description: Defining and analyzing performance metrics helps quantify the drone's effectiveness and identify areas for improvement. Common metrics include stability, control precision, power efficiency, response time, and overall flight performance.

Pros:

- **Quantitative Assessment:** Provides clear, measurable indicators of performance, allowing for objective evaluation.
- **Benchmarking:** Allows comparison with other systems or previous versions to track progress and improvements.
- **Design Optimization:** Guides design improvements based on performance data, helping to achieve desired specifications.

Cons:

- **Complexity:** Some metrics can be difficult to measure accurately, requiring sophisticated instrumentation and analysis techniques.
- **Trade-offs:** Focusing on certain metrics might lead to trade-offs in other areas, requiring careful balancing of priorities.

When to Use:

- Use performance metrics to evaluate the drone's overall effectiveness and identify areas for improvement throughout the development process.
- Essential for iterative design processes and continuous performance enhancement.

No	Title	Symbol	Unit of measurement
P1	Range	L	km
P2	Flight duration	T	hour
P3	Maximum speed	V_{max}	km/h
P4	Cruising speed	V_{crs}	km/h
P5	Static ceiling	s_{cl}	m
P6	The inverse of take-off weight	$1/W_{take\text{-}off}$	1/kg
P7	Fuel weight	W_{fuel}	kg
P8	Power plant take-off parameter	$N_{take\text{-}off}$	horse power

Table 4.6 - Performance Metrics and Analysis

4.6.3.5. Sensitivity Analysis

Description: Sensitivity analysis examines how changes in model parameters affect the drone's performance. This helps identify critical parameters that significantly influence behavior and stability, and assess the robustness of the system to parameter variations.

Pros:

- **Insightful:** Provides insights into which parameters are most critical for performance, guiding focused improvements.
- **Robustness:** Helps in designing robust systems that can tolerate variations in parameters, enhancing reliability.
- **Risk Mitigation:** Identifies potential risks associated with parameter uncertainties, allowing for proactive mitigation.

Cons:

- **Resource Intensive:** Can require numerous simulations to explore parameter variations comprehensively.
- **Complex Interpretation:** Results can be complex and require careful interpretation to draw meaningful conclusions.

When to Use:

- Use sensitivity analysis during model development to identify critical parameters and improve model robustness.
- Useful for understanding the impact of parameter changes and for optimizing system design.

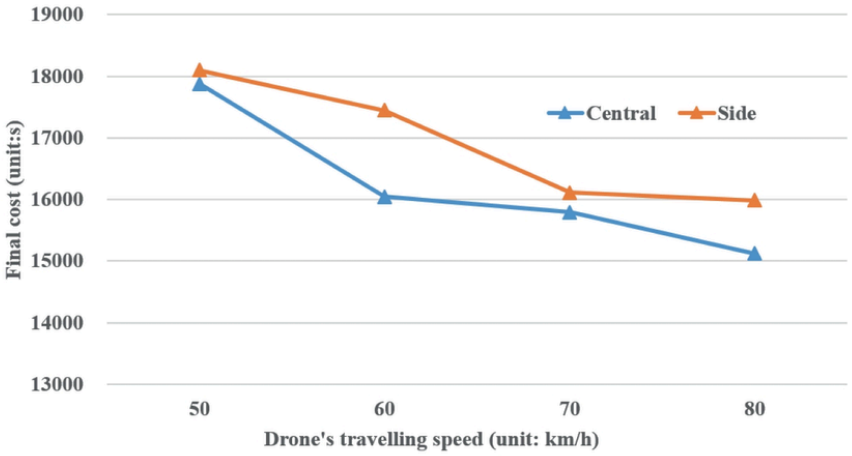


Figure 4.48 - Sensitivity Analysis

4.6.4. Control System Design

Designing a robust control system for a drone is crucial for ensuring stable, efficient, and reliable operation. This section will explore various control strategies, detailing their principles, advantages, limitations, and appropriate use cases.

4.6.4.1. Control Objectives and Requirements

Description: The first step in control system design is to define the primary objectives and requirements. These typically include:

- **Stability:** The drone must maintain stable flight under various conditions.
- **Responsiveness:** The drone should respond promptly to control inputs and environmental changes.
- **Precision:** The control system should enable accurate positioning and orientation.
- **Robustness:** The system should tolerate disturbances and uncertainties.

Understanding these objectives helps in selecting and designing appropriate control strategies.

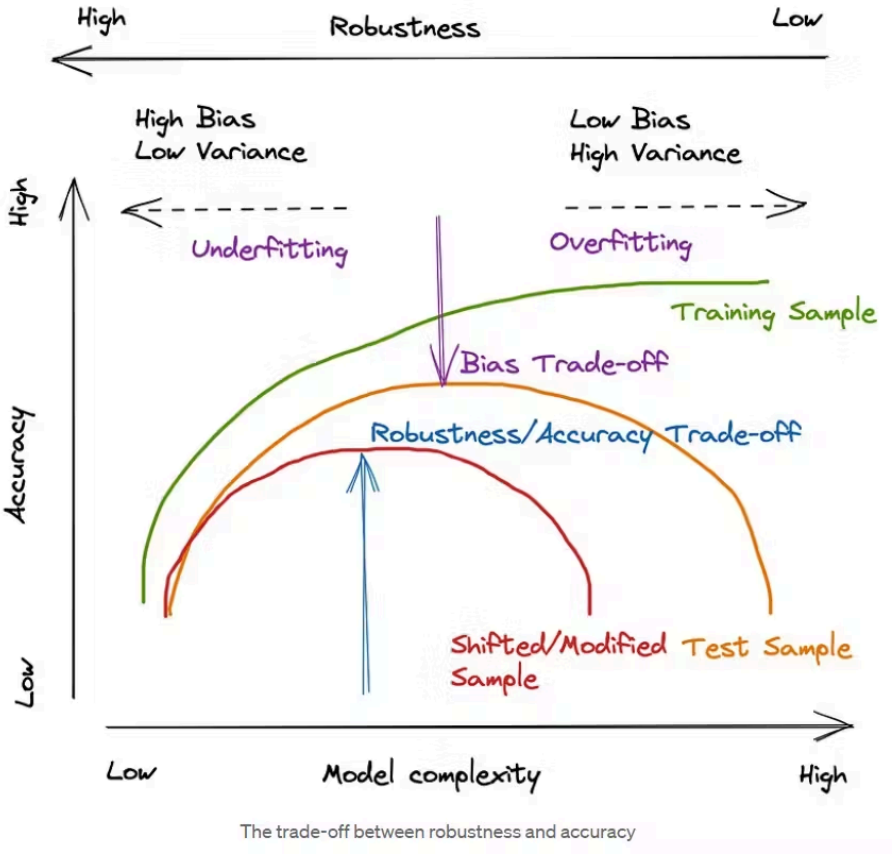


Figure 4.49 - Trade-off between robustness and accuracy

4.6.4.2. Control Strategies

Description: Various control strategies can be employed to achieve the desired objectives. Each has its principles, advantages, limitations, and suitable applications.

4.6.4.2.1. Proportional-Integral-Derivative (PID) Control

Principle: PID control is a widely used feedback control strategy that combines three terms:

- **Proportional (P):** Responds to the current error.
- **Integral (I):** Responds to the accumulation of past errors.
- **Derivative (D):** Responds to the rate of change of the error.

Advantages:

- **Simplicity:** Easy to understand and implement.
- **Effectiveness:** Works well for a wide range of control problems.
- **Tuning:** PID parameters can be tuned to achieve desired performance.

Limitations:

- **Tuning Complexity:** Finding the optimal PID parameters can be challenging.
- **Performance:** May not handle highly nonlinear or time-varying systems effectively.

How to Use PID Control:

1. **Understand the Dynamics:**
    - Identify the key states to control (e.g., position, velocity, orientation).
    - Develop a basic model of the drone's dynamics to understand how control inputs affect these states.

2. **Design the PID Controller:**
    - **Proportional (P):** Choose a gain Kp to reduce the error between the desired setpoint and the current state. This term helps the system respond to the error.
    - **Integral (I):** Choose a gain Kii to eliminate steady-state errors by accumulating past errors. This term corrects any residual errors that persist over time.
    - **Derivative (D):** Choose a gain Kd to counteract the rate of change of the error, providing a damping effect to reduce overshoot and oscillations.

3. **Tune the PID Gains:**
    - Start with Kp to get the system responding to the error.
    - Add Ki to address any steady-state error.
    - Finally, adjust Kd to smooth out the response.
    - Use methods like Ziegler-Nichols, trial-and-error, or software tools to optimize the gains.

4. **Implementation:**
    - Implement the PID controller in the drone's flight control software.
    - Continuously measure the error (difference between the desired and current state) and compute the control input using the PID formula:

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

    - Apply the computed control input to the drone's actuators (motors, control surfaces).

5. **Testing and Refinement:**
    - Conduct test flights to observe the drone's response.
    - Adjust the PID gains based on the performance observed during the tests.
    - Iterate the tuning process until satisfactory performance is achieved.

Applications:

- Suitable for basic stabilization and position control tasks.
- Commonly used in simple to moderately complex drones.

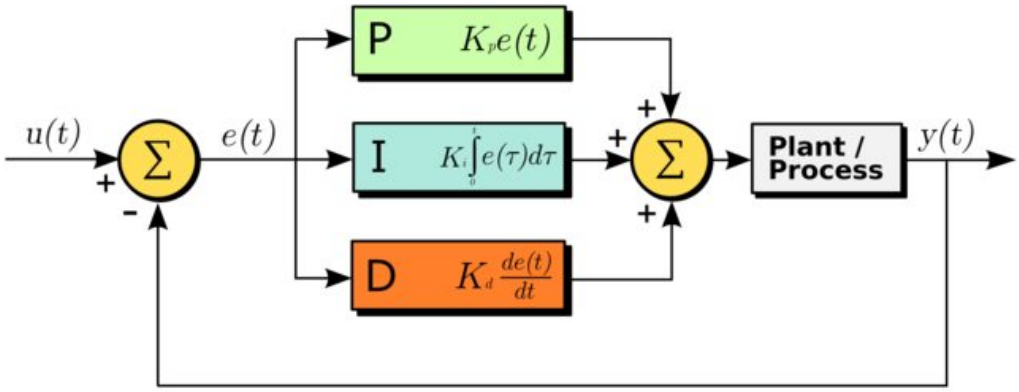
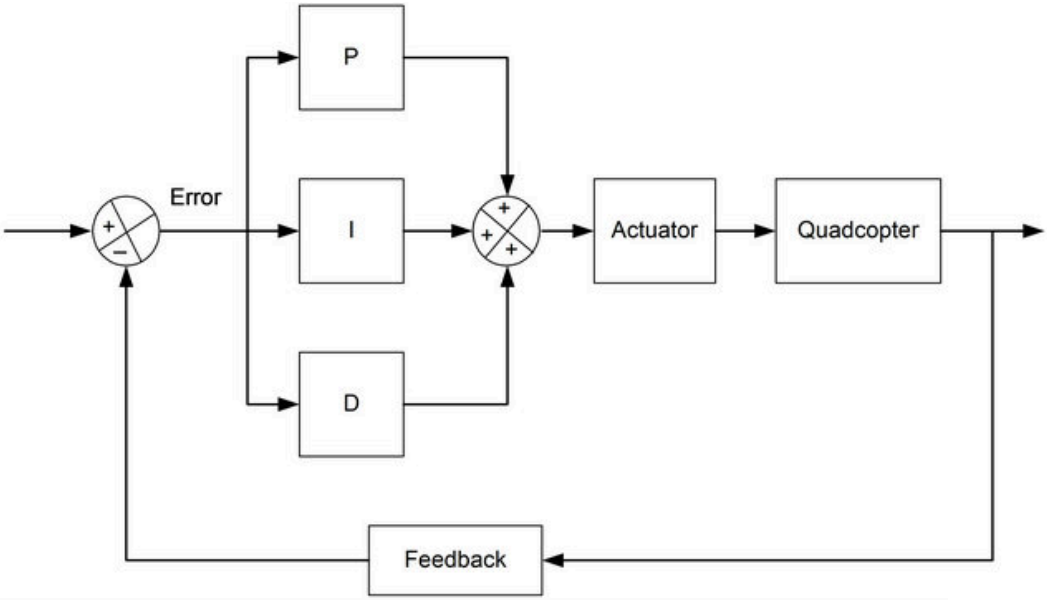


Figure 4.50 - PID block



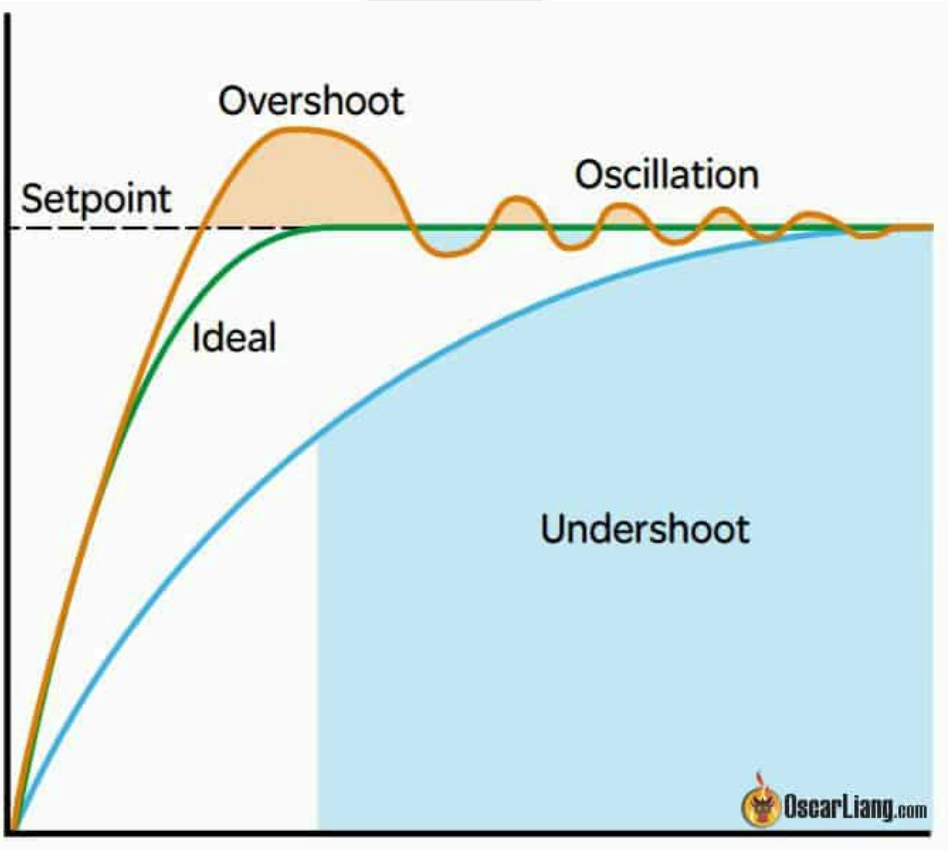


Figure 4.51 - PID loop

4.6.4.2.2. Linear Quadratic Regulator (LQR)

Principle: LQR is an optimal control strategy that minimizes a quadratic cost function, balancing control effort and system performance. It uses state-space representation to derive control laws.

Advantages:

- **Optimality:** Provides an optimal control solution based on the defined cost function.
- **State Feedback:** Uses full state feedback for precise control.

Limitations:

- **Model Dependency:** Requires an accurate state-space model of the system.
- **Complexity:** More complex to design and implement compared to PID control.

How to Use LQR:

1. **State-Space Model:**
    - Develop a state-space model of the drone dynamics:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

    - Here, **x(t)** represents the state vector, **u(t)** is the control input vector, **A** is the system matrix, **B** is the input matrix, **C** is the output matrix, and **D** is the feedthrough matrix.

2. **Define Cost Function:**
    - Choose the cost function to minimize:

$$J = \int_0^\infty (x(t)^T Q x(t) + u(t)^T R u(t)) dt$$

    - Here, **Q** is the state cost matrix and **R** is the control input cost matrix. These matrices define the trade-off between state regulation and control effort.

3. **Solve Riccati Equation:**
    - Solve the Continuous Algebraic Riccati Equation (CARE) to find the optimal gain matrix **K**:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

    - Compute the optimal control law:

$$u(t) = -Kx(t)$$

    - Where:

$$K = R^{-1}B^T P$$

4. **Implementation:**
    - Implement the LQR controller in the drone's flight control software.
    - Continuously measure the state **x(t)** and compute the control input **u(t)** using the optimal control law.
    - Apply the computed control input to the drone's actuators.

5. **Testing and Validation:**
    - Conduct test flights to validate the controller's performance.
    - Adjust the **Q** and **R** matrices to fine-tune the performance, balancing control effort and state regulation.

Applications:

- Suitable for systems where optimal performance is critical.
- Used in advanced drones requiring precise control and efficiency.

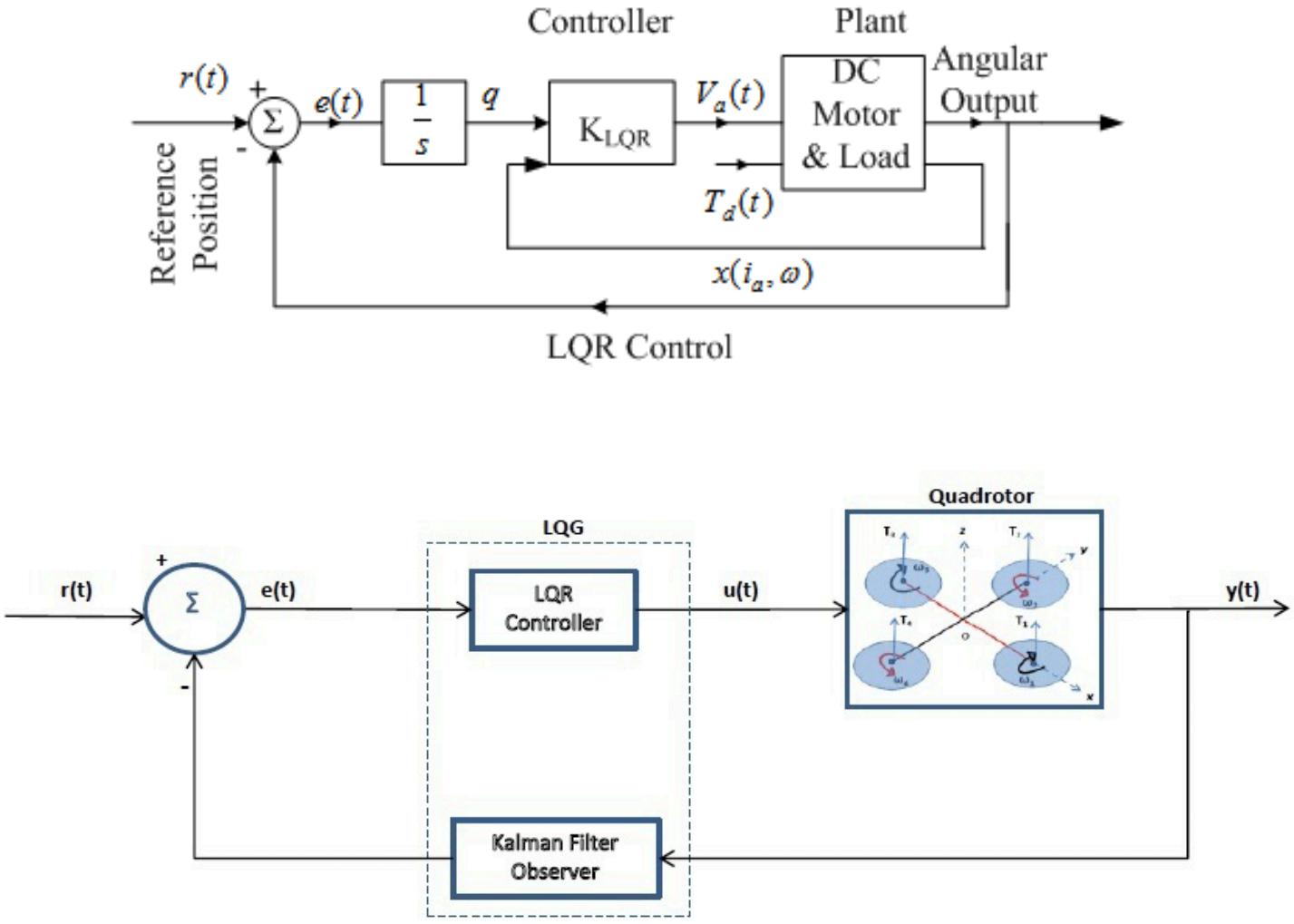


Figure 4.52 - Quadrotor loop

4.6.4.2.3. Model Predictive Control (MPC)

Principle: MPC uses a model of the system to predict future behavior and optimize control inputs over a finite time horizon. It updates control actions based on the predicted future states.

Advantages:

- **Predictive Capability:** Anticipates future system behavior for optimal control.
- **Constraint Handling:** Can handle constraints on control inputs and states.

Limitations:

- **Computational Demand:** Requires significant computational resources.
- **Model Accuracy:** Relies on accurate models for effective prediction and control.

How to Use MPC:

1. **Develop System Model:**
   - Create a model of the drone's dynamics that predicts future states based on current state and control inputs:

$$x(t+1) = Ax(t) + Bu(t)$$

   - Here, **x(t)** is the state vector, **u(t)** is the control input vector, and **A**, **B** are the system matrices.

2. **Define Optimization Problem:**
   - Formulate an optimization problem to minimize a cost function over a finite prediction horizon **N**:

$$J = \sum_{k=0}^{N-1} (x(k)^T Q x(k) + u(k)^T R u(k)) + x(N)^T Q_f x(N)$$

   - Here, **Q** and **R** are weighting matrices for states and control inputs, and **Qf** is the terminal state cost matrix.

3. **Incorporate Constraints:**
    - Include constraints on states and control inputs to ensure feasible operation:

$$x_{min} \leq x(k) \leq x_{max}$$

$$u_{min} \leq u(k) \leq u_{max}$$

4. **Solve Optimization Problem:**
    - At each time step, solve the optimization problem to find the optimal sequence of control inputs **u(k)**.
    - Apply only the first control input **u(0)** to the drone, then repeat the process at the next time step with updated state measurements.

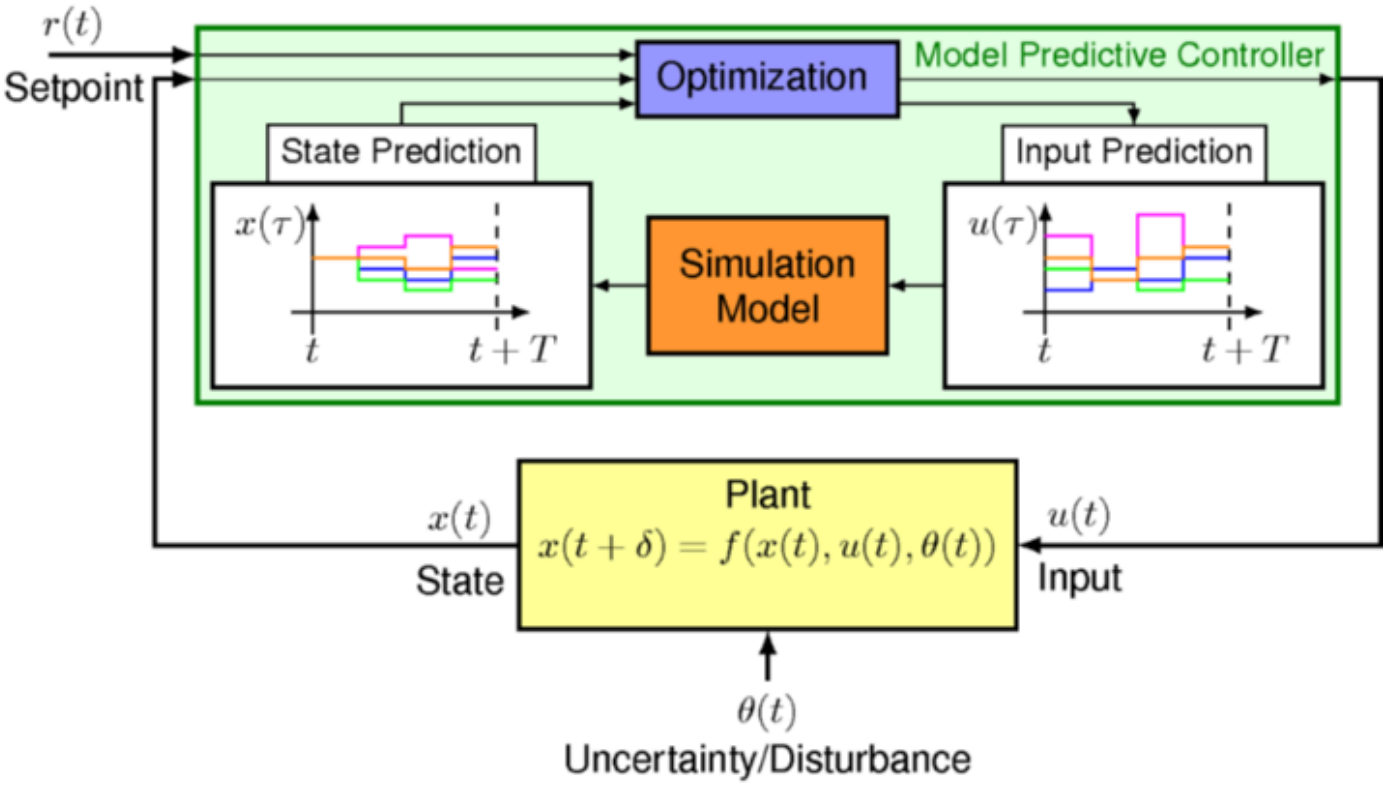
5. **Implementation:**
    - Implement the MPC algorithm in the drone's flight control software.
    - Continuously solve the optimization problem and update control inputs in real-time.

6. **Testing and Fine-Tuning:**
    - Conduct test flights to validate the MPC controller's performance.
    - Adjust the prediction horizon **N**, and weighting matrices **Q**, **R**, and **Qf** to optimize performance.

Applications:

- Ideal for complex, constrained, and multivariable control problems.
- Used in high-performance drones requiring advanced control strategies.

Figure 4.53 - Model Predictive controller loop



4.6.4.2.4. Adaptive Control

Principle: Adaptive control adjusts control parameters in real-time to cope with changes in system dynamics or uncertainties. It modifies its behavior based on observed performance.

Advantages:

- **Flexibility:** Adapts to changing conditions and system uncertainties.
- **Performance:** Maintains control performance in the presence of parameter variations.

Limitations:

- **Complexity:** More complex to design and implement.
- **Convergence:** Requires careful design to ensure convergence and stability.

How to Use Adaptive Control:

1. **Identify Variable Parameters:**
    - Identify the parameters in the drone model that are uncertain or vary with time (e.g., payload weight, battery voltage).

2. **Choose Adaptive Control Law:**
    - Select an adaptive control law that adjusts the control parameters in real-time based on system performance. Common approaches include Model Reference Adaptive Control (**MRAC**) and Self-Tuning Regulators (**STR**).

3. **Design Reference Model:**
    - For **MRAC**, design a reference model that defines the desired closed-loop performance:

$$\dot{x}_m(t) = A_m x_m(t) + B_m r(t)$$

    - Where Xm(t) is the reference model state, **Am** and **Bm** are the reference model matrices, and **r(t)** is the reference input.

4. **Update Control Parameters:**
    - Use an adaptation mechanism to update control parameters in real-time. For MRAC, the control input is:

$$u(t) = \theta(t)^T \phi(x(t))$$

    - Where **θ(t)** are the adaptive parameters and **ϕ(x(t))** is a vector of known functions of the state.

5. **Implement Adaptation Law:**
    - Implement an adaptation law to update θ(t)\theta(t)θ(t) based on the tracking error:

      e(t)=X(t)−Xm(t)

    - A typical adaptation law is:

$$\dot{\theta}(t) = -\gamma \phi(x(t)) e(t)^T PB$$

    - Where **γ** is a positive adaptation gain and **P** is a positive definite matrix solving the Lyapunov equation.

6. **Implementation:**
    - Implement the adaptive control algorithm in the drone's flight control software.
    - Continuously measure the state and adjust control parameters based on the adaptation law.

7. **Testing and Validation:**
    - Conduct test flights to validate the adaptive controller's performance.
    - Ensure the adaptation mechanism converges and the system remains stable.

Applications:

- Suitable for environments with significant uncertainties or varying dynamics.
- Used in drones operating in unpredictable or changing conditions.

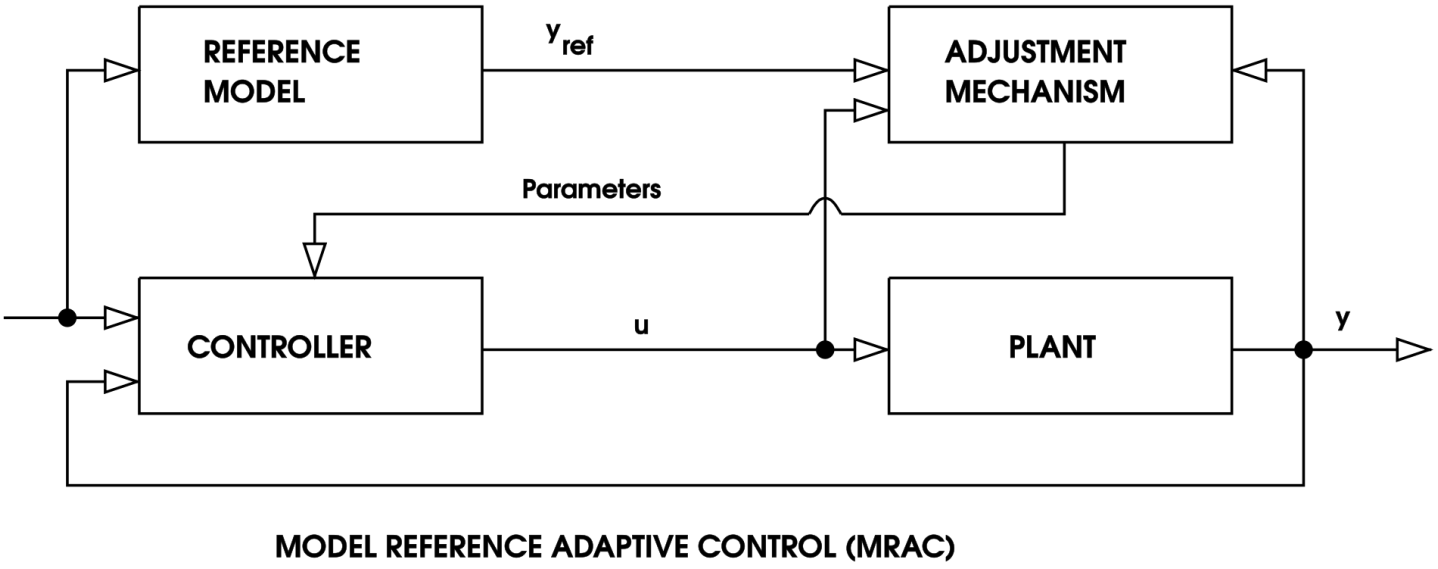


Figure 4.54 - Model Reference Adaptive Control

4.6.4.2.5. Robust Control

Principle: Robust control aims to maintain performance despite uncertainties and disturbances. Techniques like H-infinity (H∞) control design controllers that optimize worst-case scenarios.

Advantages:

- **Uncertainty Handling:** Provides robust performance under model uncertainties and external disturbances.
- **Stability:** Ensures system stability even in worst-case conditions.

Limitations:

- **Conservatism:** May result in conservative control actions.
- **Complexity:** Typically more complex to design and analyze.

How to Use Robust Control:

1. **Model Uncertainties:**
   - Identify and model the uncertainties in the drone's dynamics (e.g., parameter variations, external disturbances).

2. **Design Robust Controller:**
   - Choose a robust control design method such as H-infinity (H∞) control. Formulate the control problem to minimize the worst-case effect of uncertainties on system performance.

3. **H∞ Control Design:**
   - Formulate the H∞ control problem as an optimization problem:

$$\min \|T_{zw}\|_\infty$$

   - Where **Tzw** is the transfer function from disturbance **w** to controlled output **z**, and **||.||∞** denotes the H∞ norm.

4. **Solve Riccati Equations:**
    - Solve the associated Riccati equations to find the optimal controller gain matrices.

5. **Implement Robust Controller:**
    - Implement the robust controller in the drone's flight control software.
    - Continuously measure the state and apply the control input designed to handle worst-case scenarios.

6. **Testing and Validation:**
    - Conduct test flights to validate the robust controller's performance.
    - Ensure the controller maintains performance and stability under various conditions and disturbances.

Applications:

- Ideal for critical applications requiring guaranteed performance and stability.
- Used in drones where reliability and robustness are paramount.

4.6.4.3. Controller Design and Tuning

Description: Once a control strategy is selected, the next step involves designing and tuning the controller to meet the specified objectives. This process includes:

- **Parameter Selection:** Choosing appropriate parameters (e.g., PID gains, LQR weights).
- **Optimization:** Using techniques like trial-and-error, heuristic methods, or optimization algorithms to fine-tune parameters.
- **Simulation Testing:** Validating controller performance through simulations before real-world implementation.

Techniques:

- **Manual Tuning:** Adjusting parameters based on experience and trial-and-error.
- **Automated Tuning:** Using optimization algorithms to find optimal parameters.
- **Simulation-Based Tuning:** Testing and refining parameters in a simulated environment.

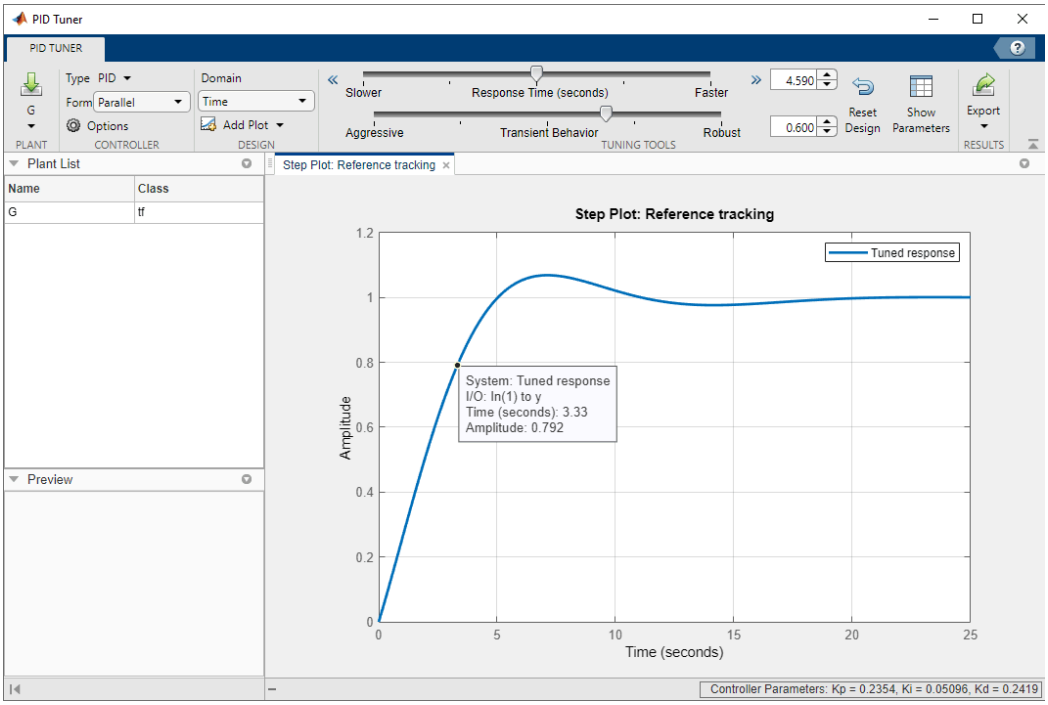


Figure 4.55 - Ex of simulation based Tuning

4.6.4.4. Implementation and Testing

Description: The final step involves implementing the designed controller on the drone's hardware and testing its performance in real-world conditions.

Steps:

- Hardware Integration: Integrating the controller with the drone's onboard computer and sensors.
- Real-World Testing: Conducting flight tests to evaluate the controller's performance under actual operating conditions.
- Iterative Refinement: Refining the controller based on test results to address any observed issues or improve performance.

Considerations:

- Safety: Ensuring safe testing procedures to avoid damage or accidents.
- Data Collection: Gathering data during tests to analyze performance and guide further improvements.
- Regulatory Compliance: Adhering to relevant regulations and standards for drone operation.

4.6.5. Practical Approach

This section outlines the practical approach we undertook to design, implement, and test the flight control system for our drone. It covers the step-by-step process, the challenges encountered, and the solutions we developed to achieve our objectives.

4.6.5.1. Project Planning and Requirements

Description:

The first step in our project was to establish clear goals and requirements. This involved:

- **Defining Objectives:** We aimed to develop a robust flight control system capable of stable, autonomous flight with effective obstacle avoidance.
- **Setting Milestones:** We broke down the project into manageable phases, each with specific deliverables and deadlines.
- **Resource Allocation:** We identified the necessary resources, including hardware (drones, sensors, computing units) and software (MATLAB for simulation).

Key Decisions:

- Selection of drone platforms based on payload capacity, flight time, and compatibility with sensors.
- Choice of sensors for navigation and obstacle detection, including GPS, IMU, and cameras.
- Selection of software development tools and simulation environments, specifically MATLAB for control system design and simulation.

4.6.5.2. System Integration and Hardware Setup

Description:

The next step was integrating the chosen hardware components and setting up the system. This involved:

- **Drone Assembly:** Assembling the drone and mounting the selected sensors and computing unit.
- **Sensor Calibration:** Calibrating sensors to ensure accurate data collection. This included GPS, IMU, and Lidar calibration procedures.
- **Communication Setup:** Establishing communication links between the drone's onboard computer, sensors, and ground control station.

Challenges and Solutions:

- **Challenge:** Ensuring reliable communication between components.

- **Solution:** We used robust communication protocols (e.g., MAVLink) and tested the setup under various conditions to ensure stability.

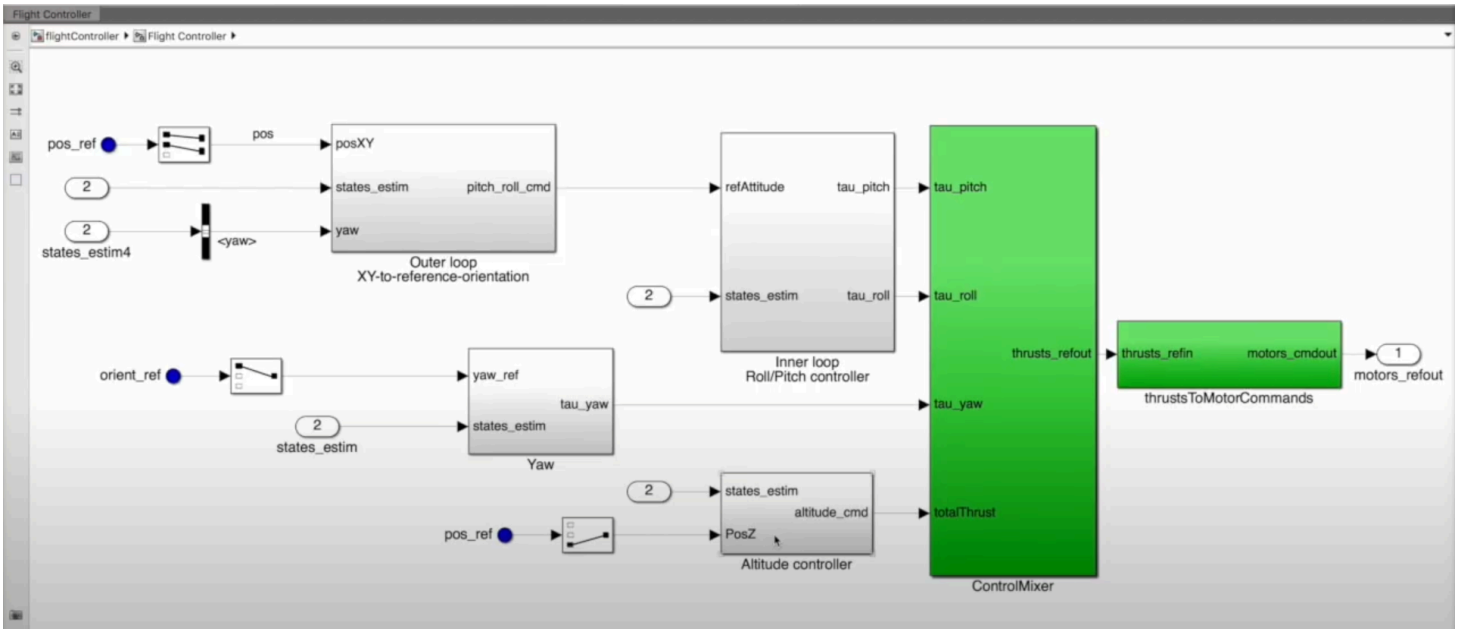
- **Challenge:** Calibrating sensors accurately.
  - **Solution:** Followed manufacturer guidelines and used calibration tools and software to achieve precise calibration.

4.6.5.3. Software Development and Simulation

Description:

We developed and tested the control algorithms using MATLAB Simulink. This phase included:

- **Algorithm Development:** Implementing the PID controller using MATLAB's PID tuner and the Root Locus method.
- **Simulation Environment:** Creating a MATLAB simulation environment to test and refine the PID controllers.
- **Software Testing:** Running simulations to test the stability and performance of the PID controllers.



Implementation Steps:

1. **PID Controller Design:**
    - **Motor Dynamics:**
        - **Calculate Thrust Against Throttle and Plot the Graph:**
            - Measure the motor's positive or negative weight while operating to calculate the thrust.
            - Plot thrust against throttle percentage to visualize the relationship.
            - Derive a linear equation from the graph to model the thrust.
        - **Motor Efficiency:**
            - Plot the efficiency of the motor over various throttle levels.
            - Use the sound property of the motor to assess its power output.
            - Record and plot the sound amplitude against time.
        - **Step Response Analysis:**
            - Operate the motor at 15% throttle, then immediately increase to 50% throttle.
            - Transform the data from the time domain to the frequency domain.
            - Identify the frequencies at which the motor operates at 15% and 50% throttle.
            - Use a spectrogram to observe the frequencies over time.
        - **Determining Tau (τ):**
            - Identify the time at which the frequency shifts from 15% to 50% throttle and is almost settled (95%).

- Calculate τ by dividing this time by 3.

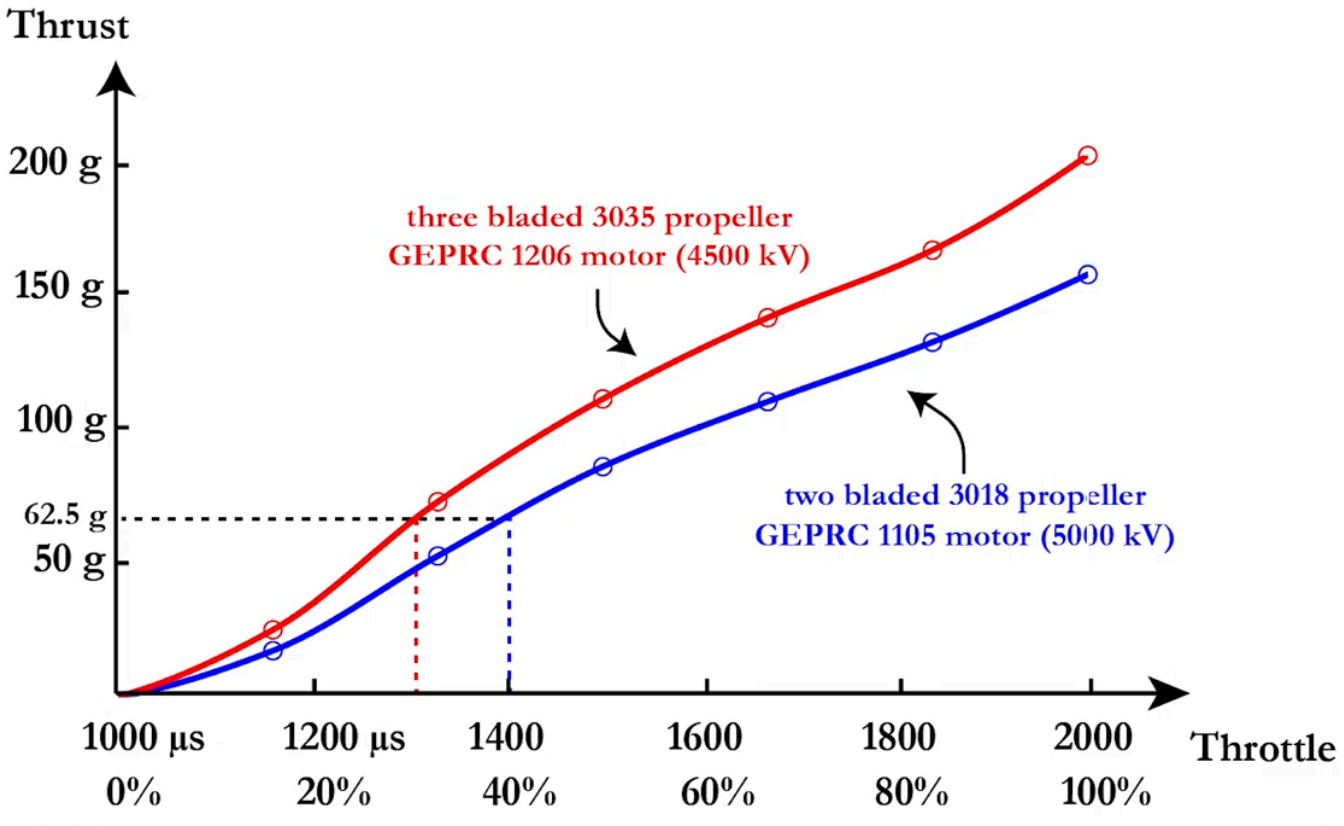
- **Motor Dynamics Model:**
  - Model the motor dynamics using the equation

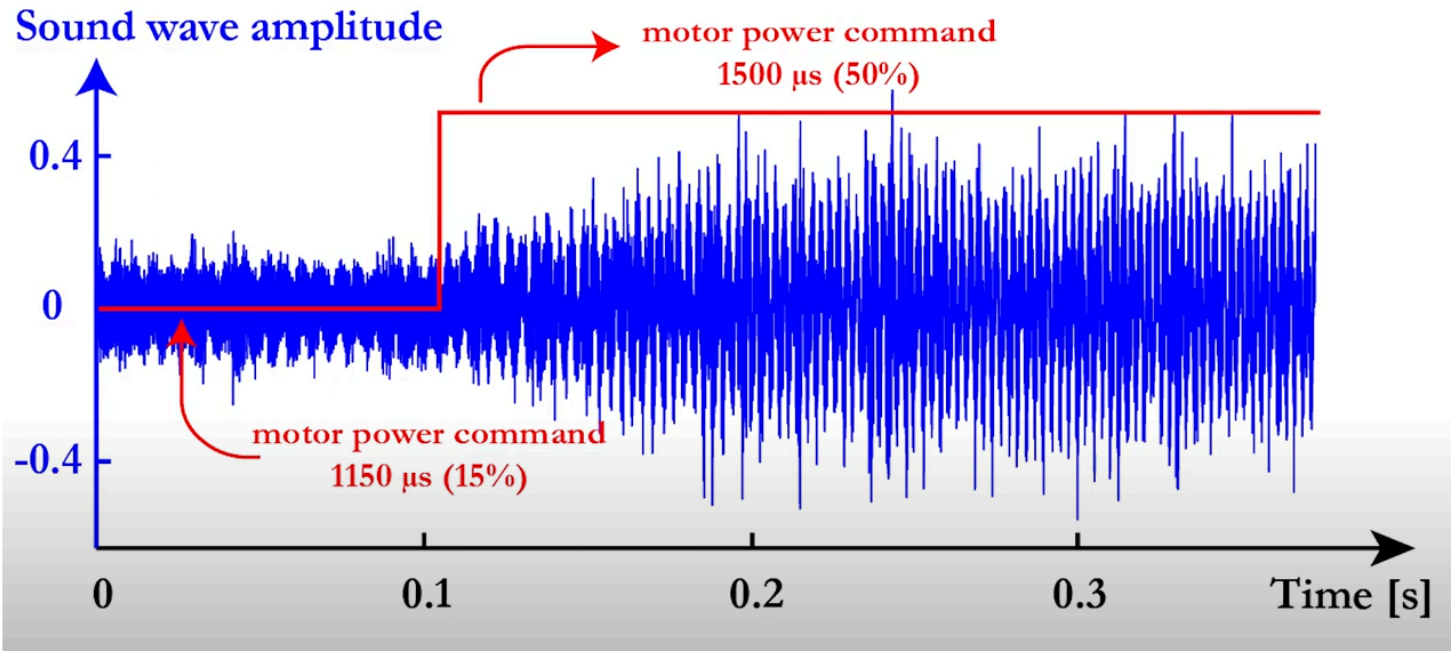
$$Y(t) = (1 - e^{-t/\tau})X(t).$$

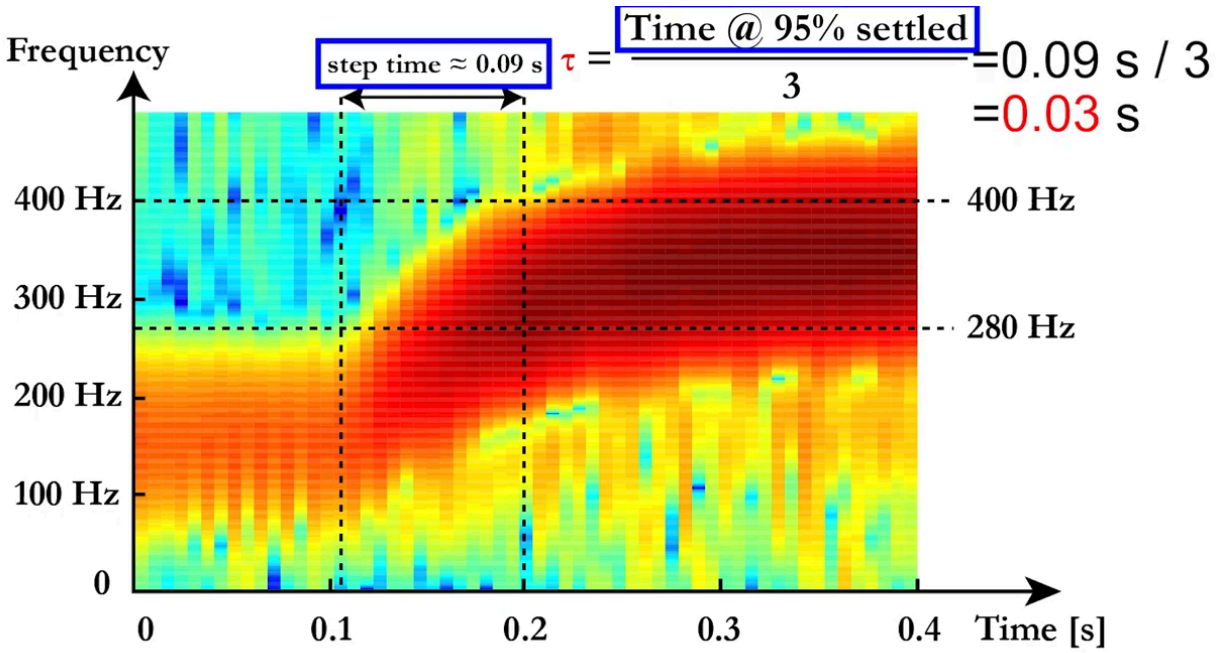
  - Apply the Laplace transform to get

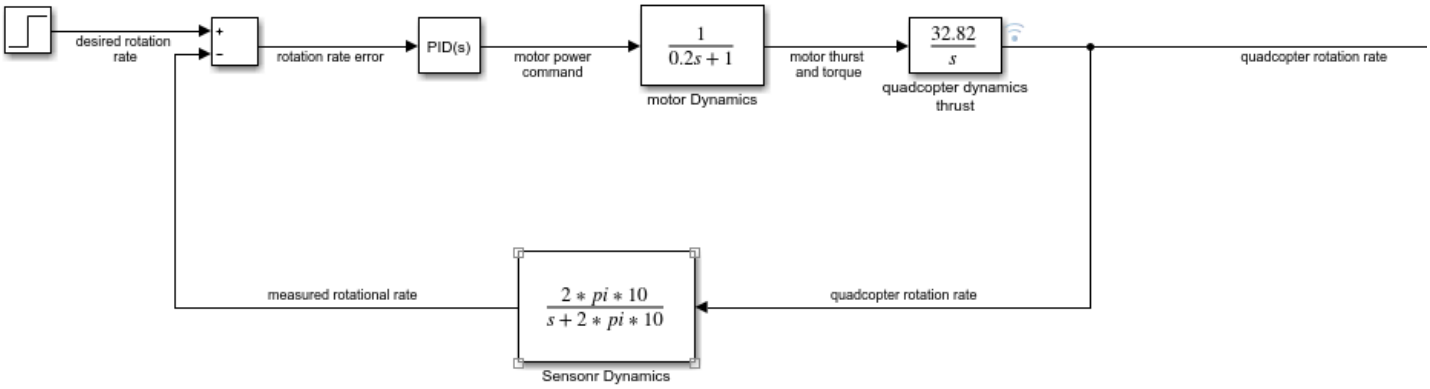
$$Y(s) = \frac{1}{\tau s+1}X(s).$$

  - Substitute the calculated τ value to finalize the motor dynamics block for the thrust control loop.


Thrust
200 g
150 g
100 g
62.5 g
50 g
three bladed 3035 propeller
GEPRC 1206 motor (4500 kV)
two bladed 3018 propeller
GEPRC 1105 motor (5000 kV)
1000 µs
0%
1200 µs
20%
1400
40%
1600
60%
1800
80%
2000
100%
Throttle

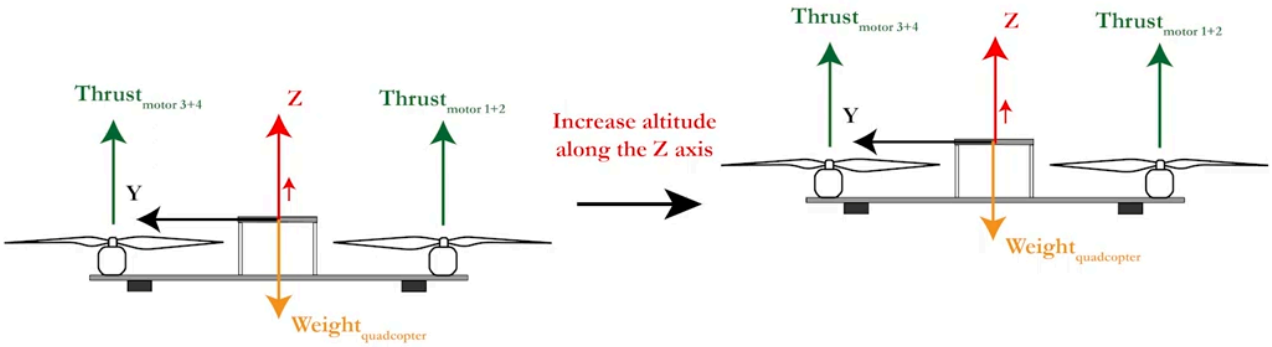

Sound wave amplitude
motor power command
1500 µs (50%)
0.4
0
-0.4
motor power command
1150 µs (15%)
0
0.1
0.2
0.3
Time [s]

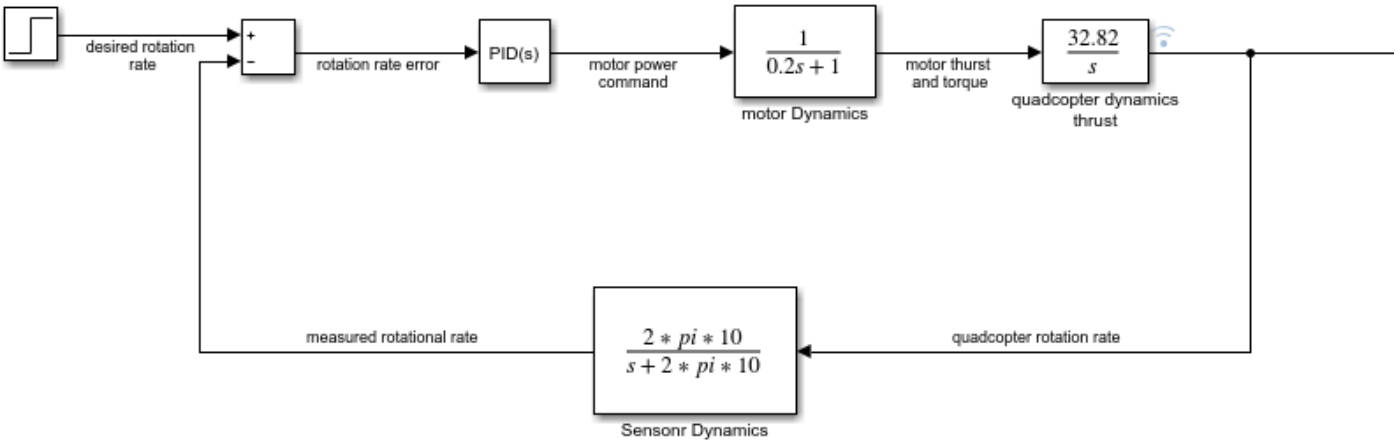




- **Sensor Dynamics:**
  - **Determine Cut-Off Frequency:**
    - Identify the cut-off sample frequency at which each sensor operates.
    - Model the sensor dynamics using the cut-off frequency ωc for our IMU the cut-off Frequency was 10HZ.
    - Represent the sensor dynamics by a first-order transfer function $H(s) = \frac{\omega_c}{s+\omega_c}$.
  - **Implementation:**
    - Integrate the sensor dynamics model into the overall system model.

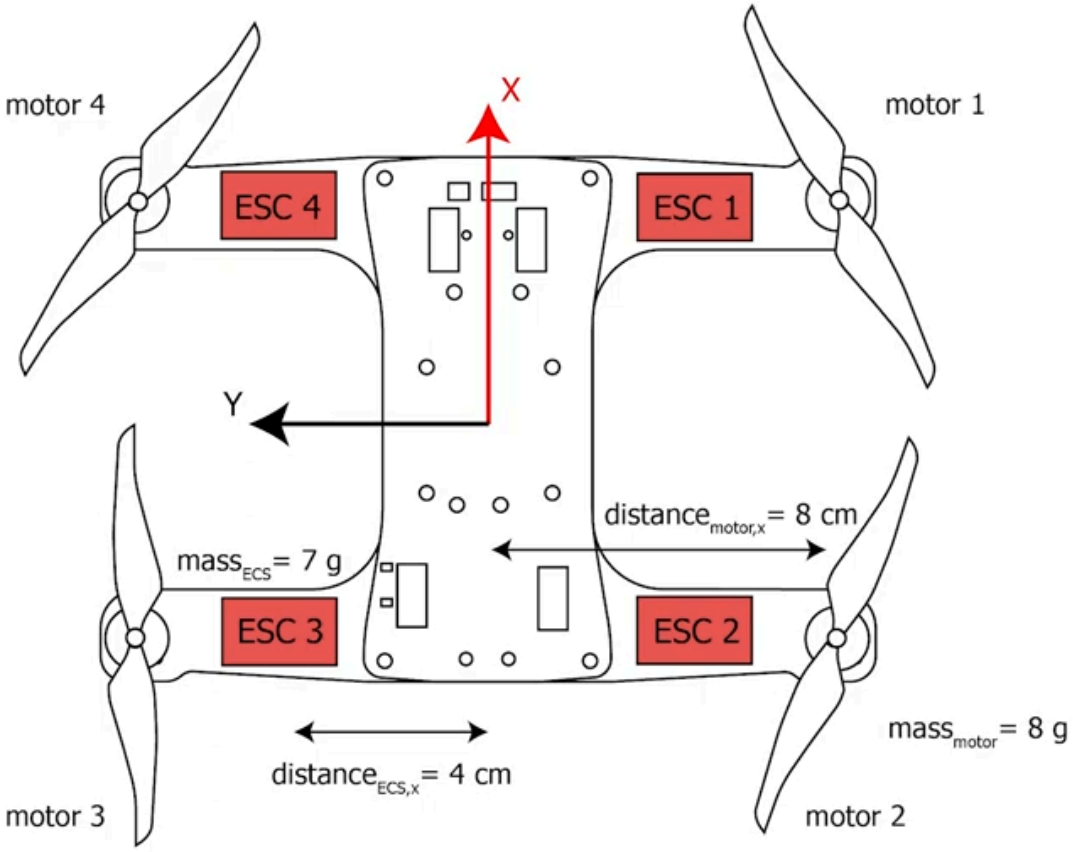
- **Quadcopter Dynamics:**
  - **Thrust Control:**
    - Design a PID controller block for maintaining altitude using the derived motor dynamics model.
    - Get the thrust from the previous equation and divide it by the mass of the whole drone to get the acceleration.
    - Get the Laplace transform of the acceleration for the quadcopter dynamics block in the thrust PID control loop.

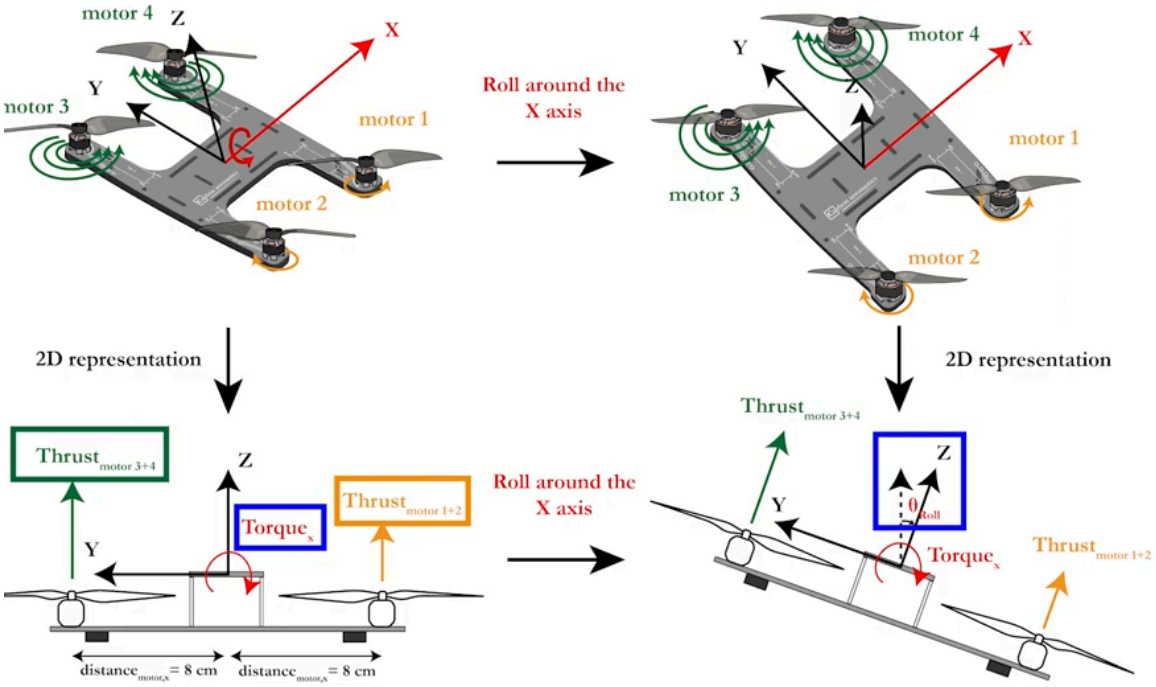


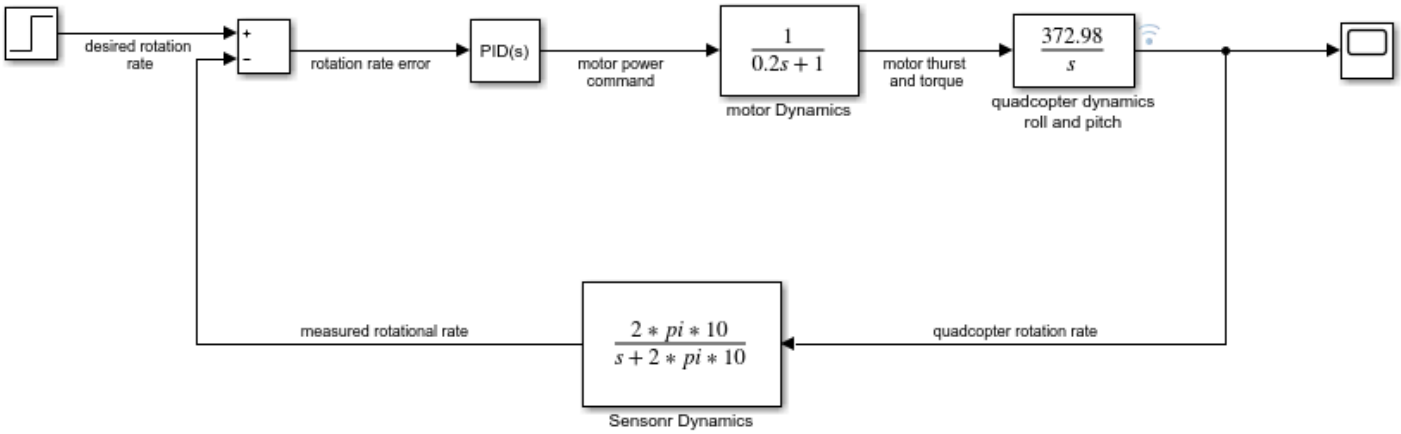


  - **Pitch/Roll Control:**
    - Implement separate PID controllers for pitch and roll to stabilize and maneuver the drone.

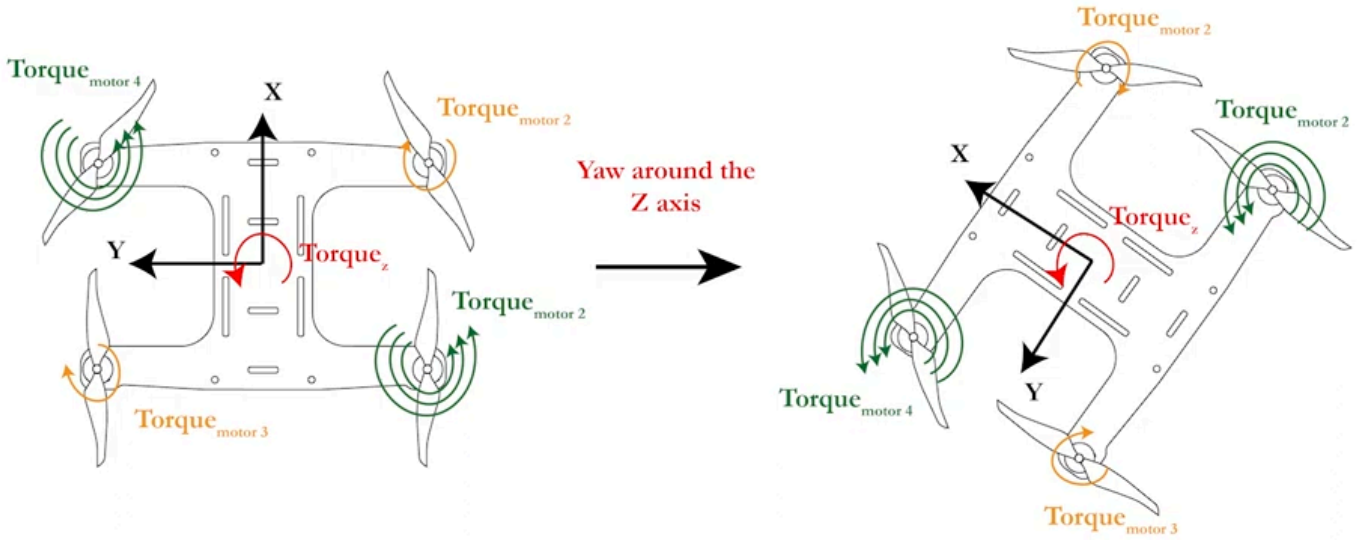
- Calculate the torque done by each motor.
- Calculate the moment of inertia at the x direction.
- Divide the torque by the inertia to get the angular acceleration in the roll direction.
- Get the Laplace transform for the quadcopter dynamics block in the roll PID controller.
- Assume symmetry for pitch and roll control blocks.

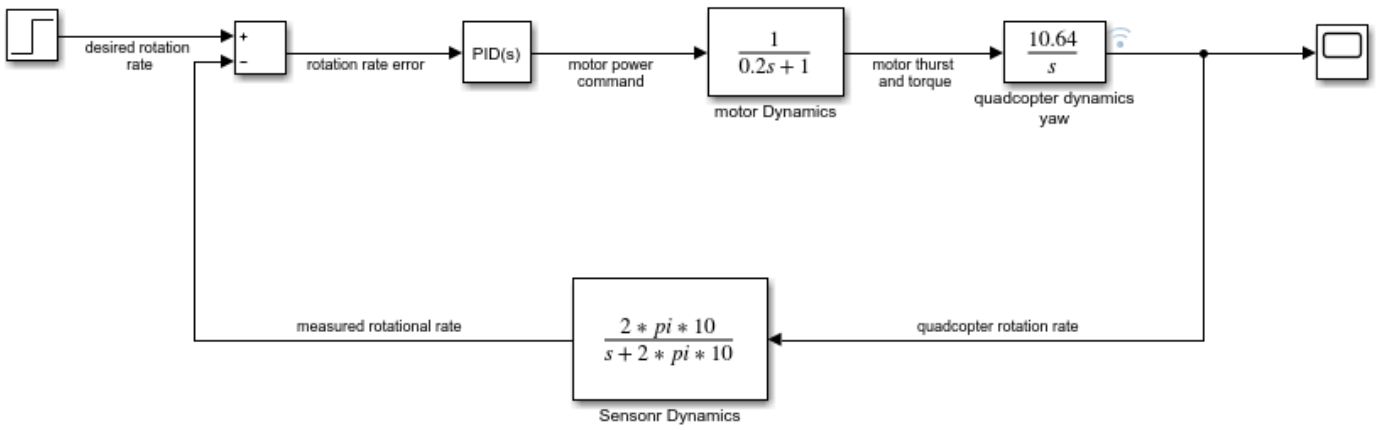






- **Yaw Control:**
  - Develop a PID controller for controlling the yaw movement.
  - Similar steps as pitch/roll control but with inertia calculated in both x and y directions.





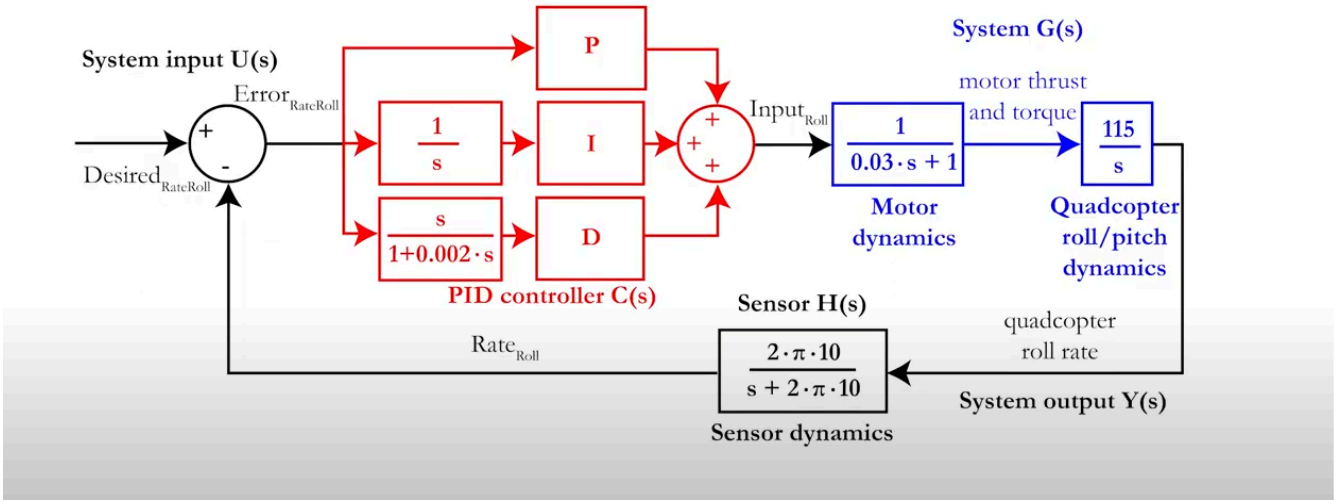
Control System Design Methods:

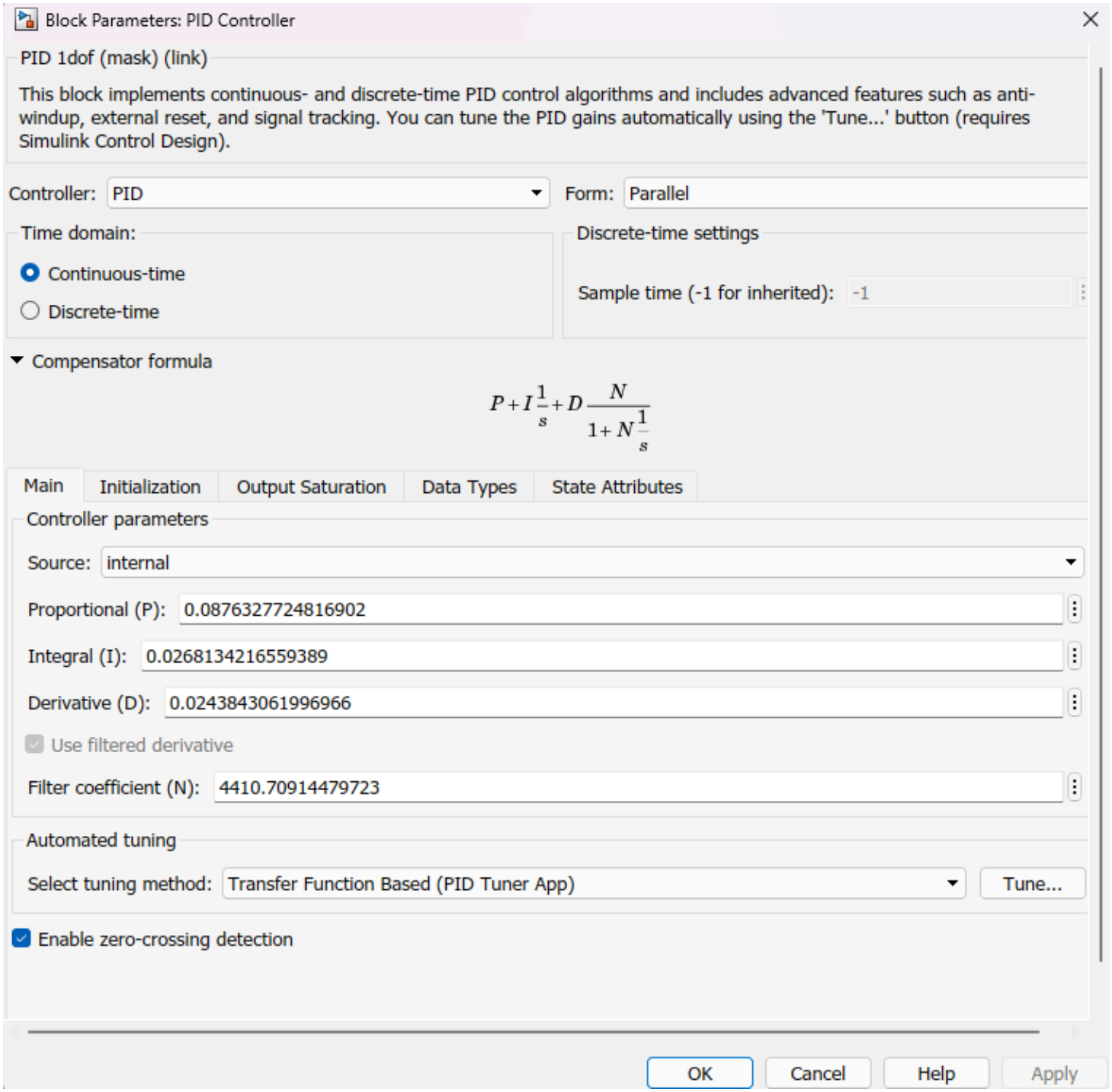
We Used 3 ways to get the PID values for each Block

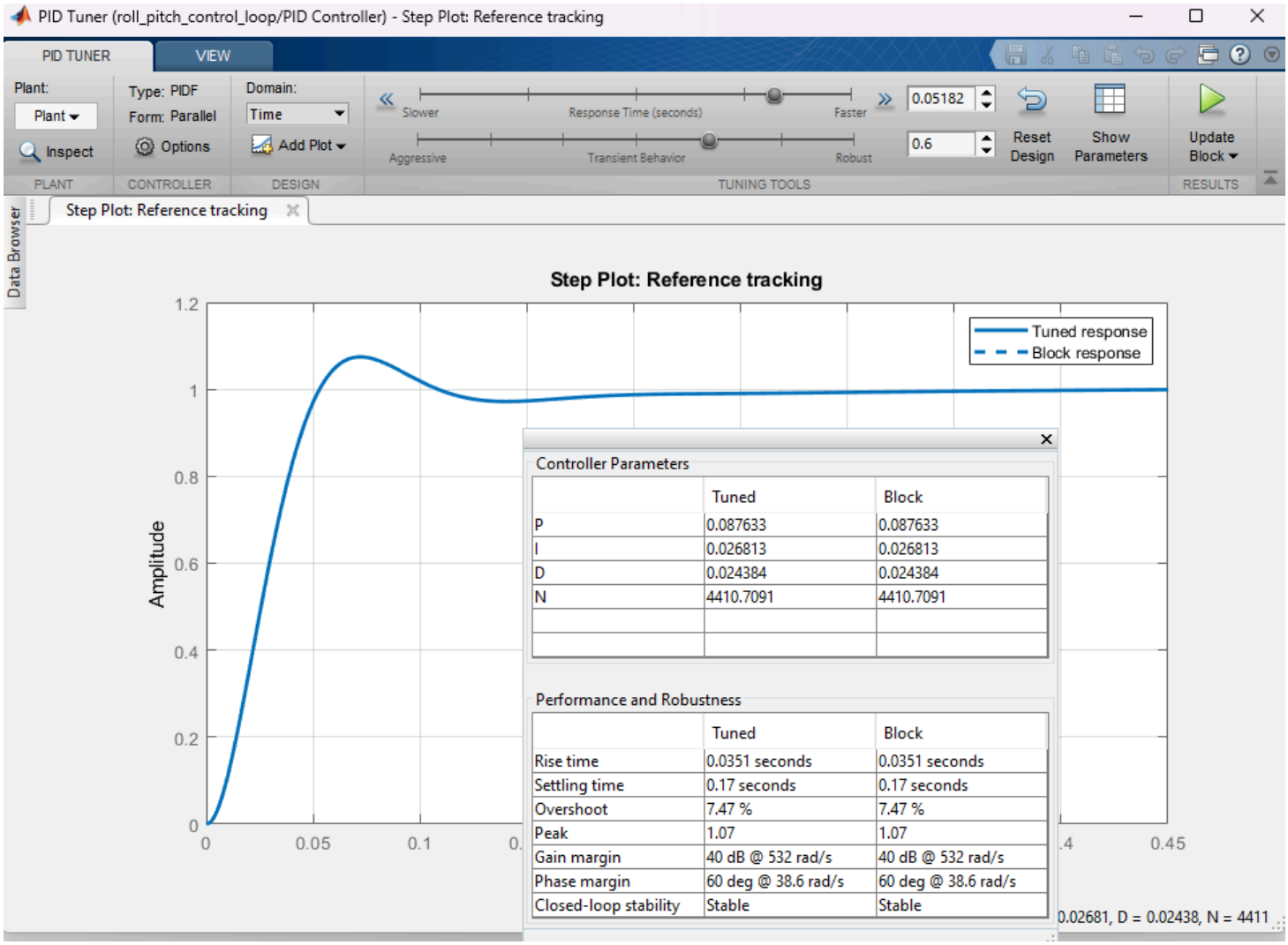
1. **Root Locus Method:**

- **Description:** A graphical approach to analyze and determine system stability by varying the gain parameter (K).
- **Application:**
  - Model the system and generate the root locus plot.
  - Analyze stability and adjust gains based on desired pole locations.
- **Pros:**
  - Provides a clear visual understanding of pole movement with varying gains.
  - Helps ensure system stability and desired transient response.
- **Cons:**
  - Complex for high-order systems.
  - Requires initial mathematical modeling of the system.

2. **MATLAB PID Tuner:**
   - **Description:** An interactive tool that automatically tunes PID controller gains to achieve a desired response.
   - **Application:**
     - Launch PID Tuner and import the plant model.
     - Use automatic tuning and refine gains using the interactive interface.
   - **Pros:**
     - Easy to use with an intuitive graphical interface.
     - Quickly provides a good starting point for PID parameters.
   - **Cons:**
     - May not always achieve the most optimal tuning for specific requirements.
     - Limited control over the underlying tuning algorithm.




Block Parameters: PID Controller
PID 1dof (mask) (link)
This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).
Controller: PID
Form: Parallel
Time domain:
Continuous-time
Discrete-time
Discrete-time settings
Sample time (-1 for inherited): -1
Compensator formula
$P+I\frac{1}{s}+D\frac{N}{1+N\frac{1}{s}}$
Main
Initialization
Output Saturation
Data Types
State Attributes
Controller parameters
Source: internal
Proportional (P): 0.0876327724816902
Integral (I): 0.0268134216559389
Derivative (D): 0.0243843061996966
Use filtered derivative
Filter coefficient (N): 4410.70914479723
Automated tuning
Select tuning method: Transfer Function Based (PID Tuner App)
Tune...
Enable zero-crossing detection
OK
Cancel
Help
Apply



3. **Manual Tuning:**
    - **Description:** Adjusting PID controller gains based on empirical observations of the system's response.
    - **Application:**
        - Begin with initial guesses for PID gains.
        - Adjust proportional, integral, and derivative gains iteratively.
        - Test the system response and refine gains to achieve desired performance.
        - After the previous PID unfortunately we had to change the PID values we got from the simulation as for sure we won't be able to model everything accurately.
    - **Pros:**
        - Provides deep understanding and control over the tuning process.
        - Can achieve highly optimized performance for specific scenarios.
    - **Cons:**
        - Time-consuming and requires significant trial and error.
        - Dependent on the engineer's experience and intuition.

4.6.5.4. Implementation and Field Testing

Description:

After validating the PID controllers in the simulation environment, we deployed them on the actual drone and conducted field tests. This phase included:

- **Software Deployment:** Transferring the tested PID controllers to the drone's onboard computer.
- **Initial Testing:** Conducting initial test flights to verify basic functionality and stability.
- **Iterative Refinement:** Continuously refining the PID controllers based on test flight data and feedback.

Challenges and Solutions:

- **Challenge:** Dealing with unexpected flight behaviors.
  - **Solution:** Conducted thorough pre-flight checks and made incremental adjustments to the PID parameters based on observed behavior.
- **Challenge:** Ensuring safety during test flights.
  - **Solution:** Established safety protocols, including predefined no-fly zones and emergency landing procedures.

4.6.5.5. Performance Evaluation and Results

Description:

We evaluated the performance of our PID-controlled flight system through a series of structured tests and metrics. This involved:

- **Flight Stability:** Assessing the drone's stability under various conditions (e.g., wind, different payloads).
- **Navigation Accuracy:** Measuring the accuracy of the drone's navigation and path-following capabilities.
- **Obstacle Avoidance:** Testing the effectiveness of the obstacle detection and avoidance algorithms.

Results:

- **Flight Stability:** The drone maintained stable flight under various conditions, demonstrating the effectiveness of the PID controllers.
- **Navigation Accuracy:** The drone consistently followed planned paths with high accuracy, meeting our navigation requirements.

- **Obstacle Avoidance:** The obstacle detection and avoidance system successfully prevented collisions in all test scenarios.

Key Learnings:

- Importance of thorough testing in both simulated and real-world environments.
- Need for continuous monitoring and refinement to achieve optimal performance.
- Value of robust safety protocols to ensure safe operation during testing.

4.6.5.6. Conclusion and Future Work

This section summarizes the outcomes of our project, reflecting on the successes and challenges, and outlines potential future work to enhance the flight control system.

Summary:

- Successfully developed and implemented a robust flight control system for a drone, capable of autonomous navigation and obstacle avoidance.
- Achieved high levels of stability, accuracy, and safety through rigorous testing and refinement.

Future Work:

- Enhancing the PID controllers to further improve performance under extreme conditions.
- Integrating advanced AI techniques for more sophisticated autonomous behaviors.
- Expanding the system to support multi-drone coordination and collaboration.

4.7. HAL Module

Defines External hardware connected to the main MCU of both boards. Currently implemented the drivers for the following hardware:

1) NRF24L01 (RF IC)
2) Remote control (the user interface)
3) ADXL345 (accelerometer IC)
4) MPU6050 (IMU IC, has build-in accelerometer and gyroscope)
5) BMP280 (barometer IC)
6) ESC (drivers for brushless dc motors)
7) HC SR04 (ultrasonic sensor)
8) HMC5883L (magnetometer IC)

4.7.1. Functional Description

4.7.1.1 NRF24L01

Responsible for communication between the remote control and application board.

4.7.1.2 Remote control

Responsible for sending commands from the remote control to the application board and receive info from the application board.

4.7.1.3 ADXL345

An accelerometer IC, the drivers code are working correctly but the chips itself is fake. How did we know it was fake? It didn't behave as described in the datasheet. It behaved more likely to be a random number generator. So make sure to buy ICs from authorized distributors.

4.7.1.4 MPU6050

IMU IC. contains both accelerometer and gyroscope IC. it's mainly responsible for yaw, pitch, roll angles estimation after fusion with other sensor readings.

4.7.1.5 BMP280

barometer IC. measures pressure which gives an approximation for the altitude in order to compute vertical velocity which in turns acts as a feedback for the thrust.

4.7.1.6 ESC

Driver ICs for the brushless DC motor where every DC motor can consume up to 30A. Controlling the ESC is done via PWM signal.

4.7.1.7 HC SR04

Ultrasonic sensor. Can detect the correct altitude estimate up to 4 meters but wasn't used because it took so much time to give back a reading.

4.7.1.8 HMC5883L

Magnetometer IC. useful to get a yaw angle which is considered feedback to the system.

4.7.2. Modular Decomposition

4.7.2.1. NRF24L01

The NRF24L01 is a popular highly efficient wireless transceiver module that operates in the 2.4 GHz ISM band. It is widely used for wireless communication applications due to its robust performance, low power consumption, and ease of integration. In our QuadRaptor project, the NRF24L01 module is utilized for wireless communication between the remote control and the drone board, enabling us to transmit the desired orientation parameters such as roll, pitch, yaw, and thrust.

Characteristics of NRF24L01:

- **Frequency Range**: Operates in the 2.4 GHz ISM band.
- **Data Rate**: Supports data rates of 250 kbps, 1 Mbps, and 2 Mbps.
- **Low Power Consumption**: Designed for low power consumption, making it suitable for battery-powered devices.
- **Range**: Effective communication range up to 100 meters in open space, extendable with additional antenna configurations.
- **Enhanced ShockBurst™**: Ensures high-speed data transmission with reduced interference and enhanced reliability.
- **SPI Interface**: Simple Serial Peripheral Interface (SPI) for easy integration with microcontrollers.
- **MultiCeiver™ Feature**: Supports multiple data pipes, allowing the module to receive data from up to six different transmitters.

Customized Package Development:

For the QuadRaptor project, we developed a customized software package for the NRF24L01 transceiver module specifically tailored to our CH32V203 microcontroller. This package ensures seamless integration and optimized communication between the remote control and the drone board. Key functionalities of the customized package include:

- **Initialization and Configuration**: Efficiently sets up the NRF24L01 module with the desired parameters for communication.
- **Data Transmission and Reception**: Handles the real-time transmission and reception of control signals for roll, pitch, yaw, and thrust.
- **Additional Telemetry Data**: Sends critical telemetry data to the remote control, including temperature, altitude, distance from origin, battery storage and status, and flying time (time elapsed since the drone was flown).

4.7.1.2 Remote control

The remote control is just a hobbyist remote control which can be programmed via arduino IDE. where the schematic of the remote control is as follow:

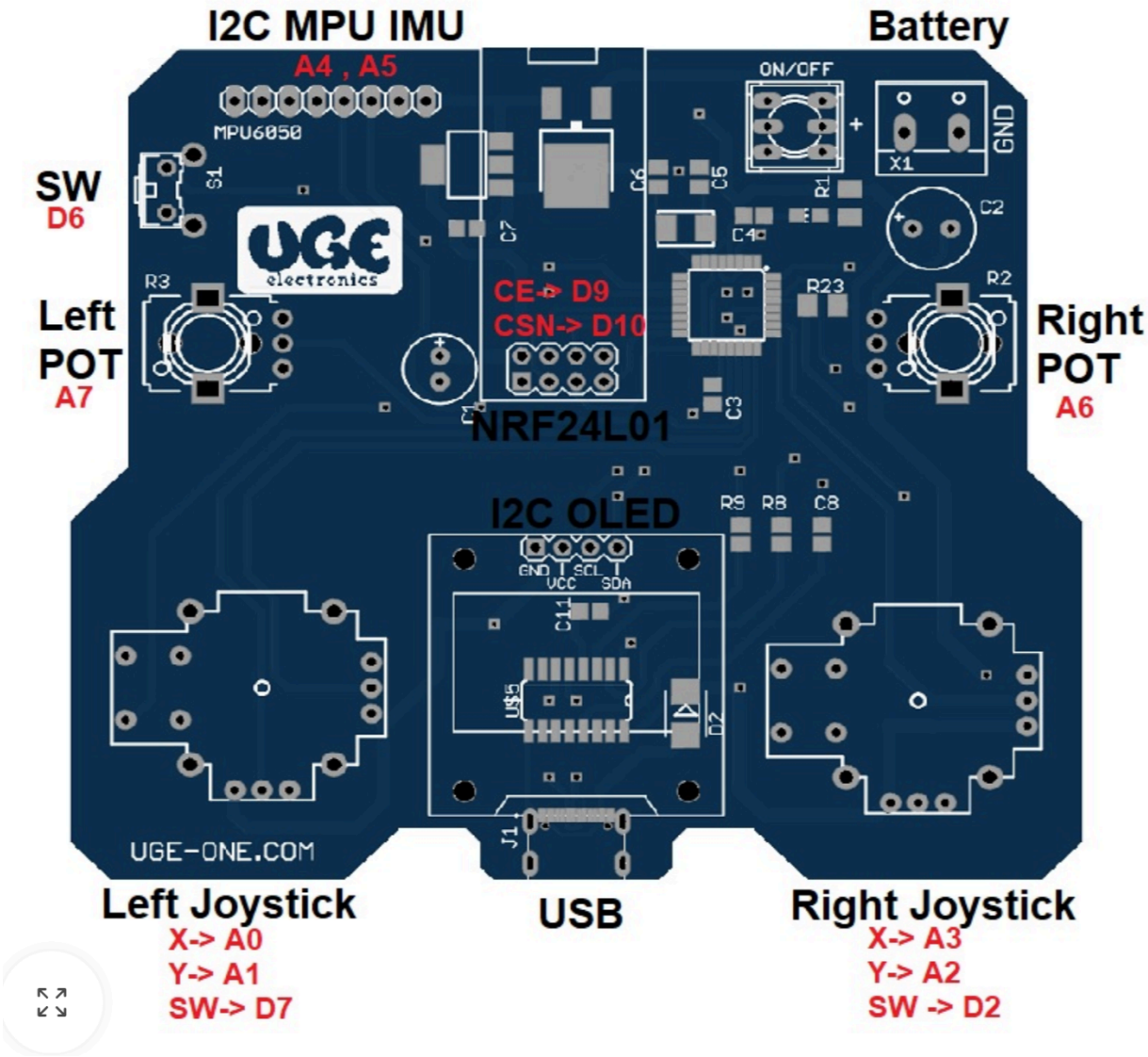
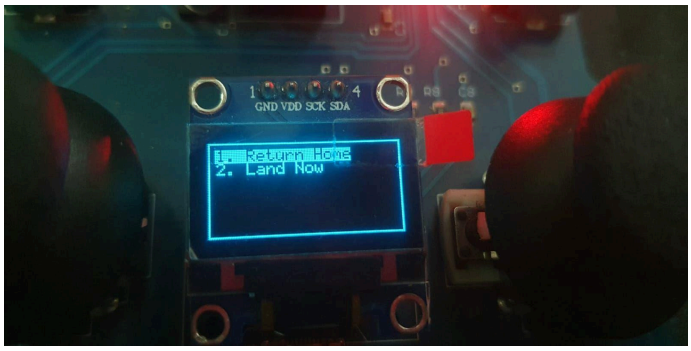
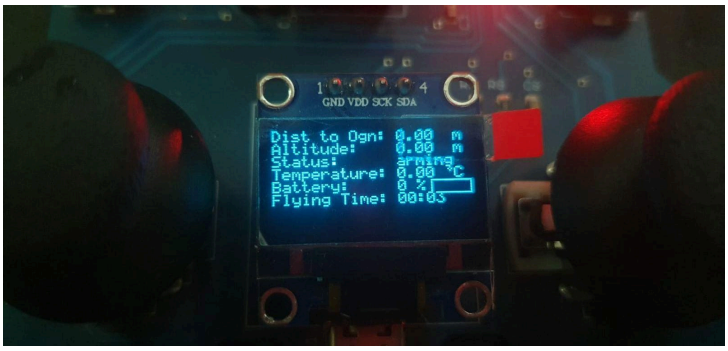


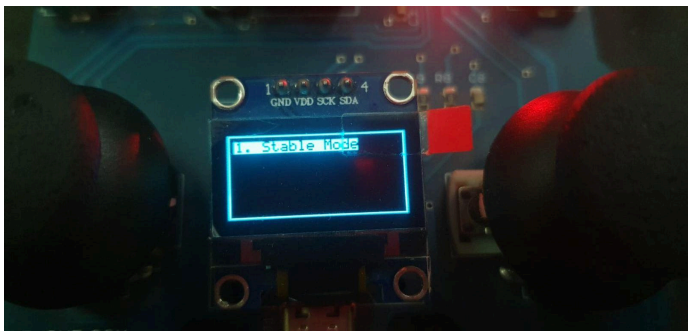
Figure 4.57 - Remote Control Schematic

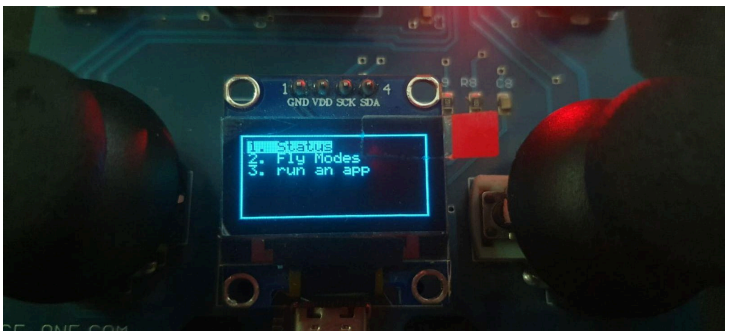
The remote control code is written by us but note that it seems that whoever designed this remote control pcb didn't take into consideration the amount of current it can draw. As sometimes the remote control just powers off indicating the failure of regulators to maintain the intended current draw from the USB-C port.

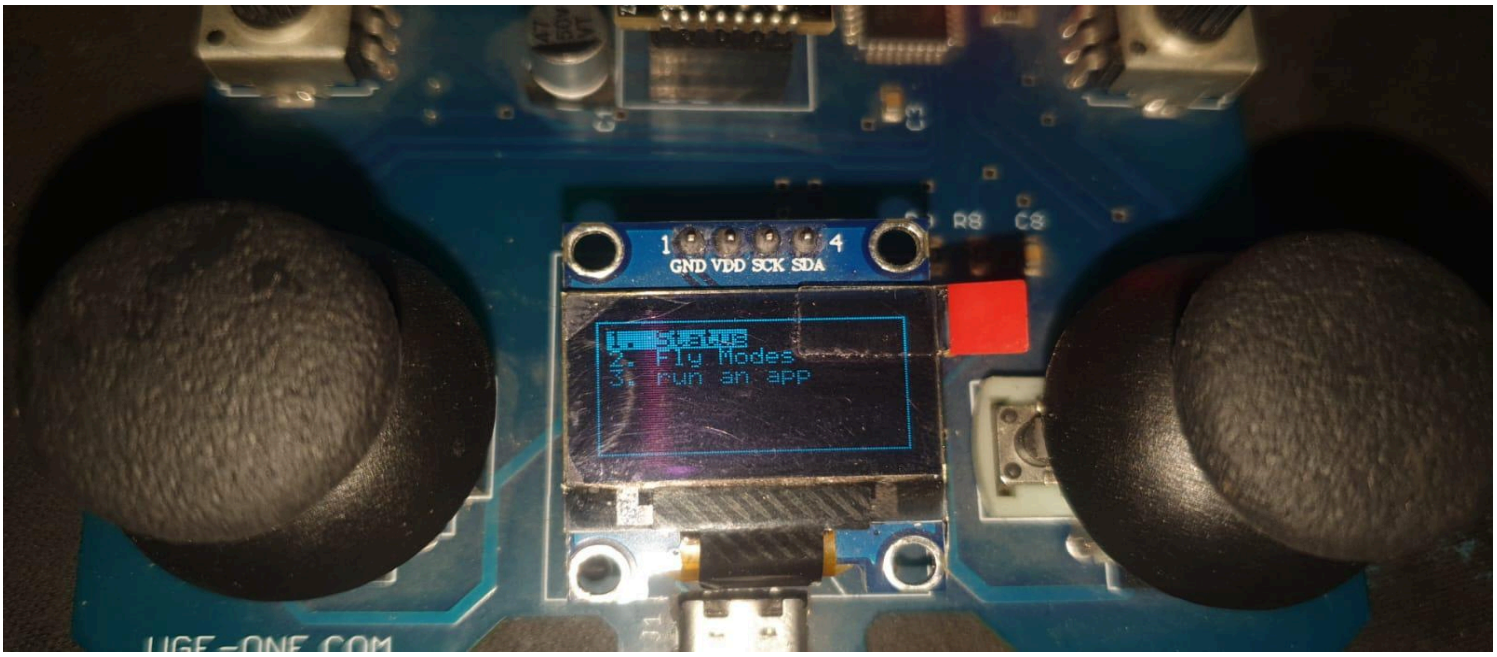
The remote control had many windows for different applications. So at the main menu, you can see this where scrolling is done via the right joystick and selecting is via clicking right joystick switch. And to return to the main menu just click on S1 switch.











4.7.1.3 ADXL345

The ADXL345 sensor we had was a fake sensor. But for the concept of working, we can make use of datasheet. Where ADXL345 will do the ADC conversion to you then store the accelerometer values along each axis in specific registers which later can be read via SPI/I2C interface.

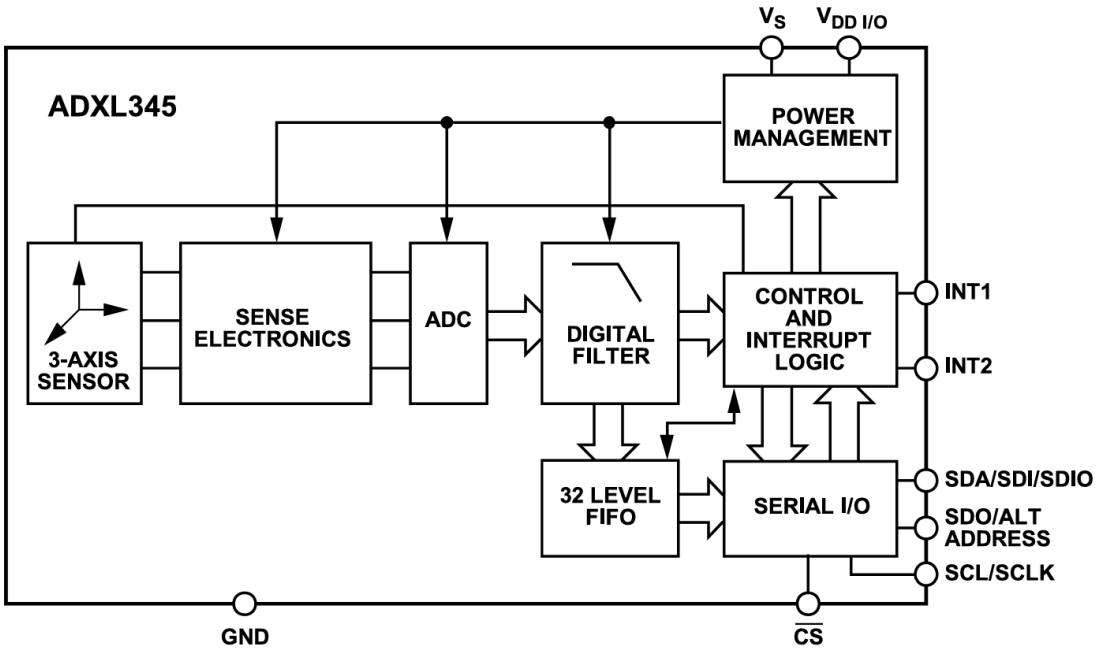


Figure 4.58 - ADXL345 internal block diagram

Where if the sensor was layed down stand still, it shall give around a constant **1g** reading at one axis and nearly 0 at the other 2 axes depending on the orientation of the sensor.

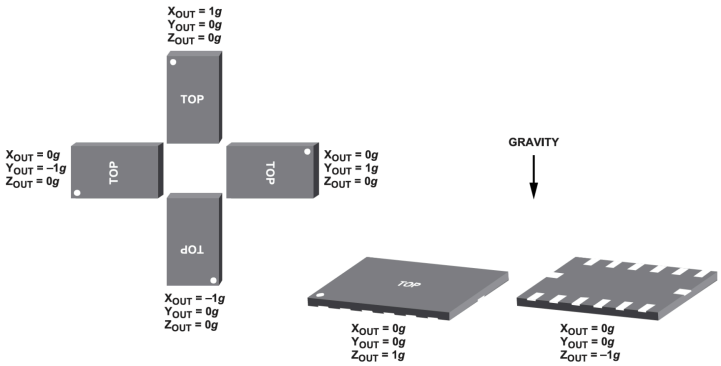


Figure 4.59 - ADXL345 stand still readings

4.7.1.4 MPU6050

The MPU6050 is a widely used and highly integrated sensor module that combines a 3-axis gyroscope and a 3-axis accelerometer in a single package. This module is designed for various applications, including motion sensing, stabilization, and control in devices such as quadcopters, robotics, smartphones, and gaming devices.

Key Features of MPU6050:

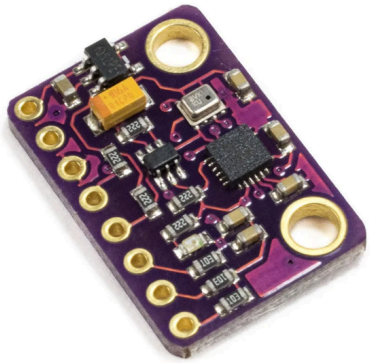
1. **6-Axis Motion Tracking:**
    - **3-Axis Gyroscope:** Measures the angular velocity around the X, Y, and Z axes, providing rotational motion data.
    - **3-Axis Accelerometer:** Measures acceleration along the X, Y, and Z axes, which can be used to determine the orientation and movement of the device.
2. **Digital Motion Processor (DMP):**
    - The MPU6050 includes an integrated DMP, which offloads complex motion processing tasks from the main microcontroller. The DMP can process the raw sensor data and provide higher-level motion data, such as pitch, roll, and yaw.
3. **I2C Communication:**
    - The MPU6050 communicates with a host microcontroller via the I2C (Inter-Integrated Circuit) bus, making it easy to integrate with various microcontroller platforms.
4. **Configurable Full-Scale Range:**
    - The accelerometer and gyroscope have configurable full-scale ranges, allowing for flexibility in different applications:
        - Accelerometer: ±2g, ±4g, ±8g, ±16g
        - Gyroscope: ±250°/s, ±500°/s, ±1000°/s, ±2000°/s
5. **Low Power Consumption:**
    - Designed for battery-powered applications, the MPU6050 features low power consumption modes and an integrated power management system.
6. **High Sensitivity and Precision:**
    - The MPU6050 provides high sensitivity and precision, enabling accurate motion detection and control.

4.7.1.5 BMP280

The BMP280 is a high-precision barometer sensor designed for measuring atmospheric pressure and altitude. It is widely used in various applications, including weather monitoring, altimetry, and drone navigation. In our QuadRaptor project, the BMP280 barometer is utilized to provide real-time altitude data, which is crucial for maintaining stable drone altitude after altitude fusion using 2D Kalman filter.

Characteristics of BMP280

High Precision: Provides accurate measurements of atmospheric pressure and altitude.
Low Power Consumption: Designed for low power consumption, making it suitable for battery-powered devices.
Small Form Factor: Compact design allows for easy integration into various systems.
Fast Response Time: Capable of quickly responding to changes in atmospheric pressure, providing real-time data.
I2C and SPI Interfaces: Supports both I2C and SPI communication protocols for easy integration with microcontrollers.



Customized Package Development:

For the QuadRaptor project, we developed a customized software package for the BMP280 barometer specifically tailored to our CH32V203 microcontroller. This package ensures seamless integration and optimized data acquisition from the BMP280 sensor. Key functionalities of the customized package include:

- **Initialization and Configuration:** Efficiently sets up the BMP280 sensor with the desired parameters for pressure and altitude measurement.
- **Data Acquisition:** Continuously measures atmospheric pressure and calculates altitude based on the collected data.
- **Additional Telemetry Data**: Provides critical telemetry data such as temperature, which is essential for various calculations and environmental monitoring.
- **Transmission to Remote Control:** Sends altitude, temperature, and other telemetry data to the remote control, allowing real-time monitoring of the drone's status.

4.7.1.6 ESC

ESC has 3 pin interface as follows:

Where +5V is an output from ESC through BEC circuit. While Signal pin is an input PWM signal to the ESC.

The Signal pin concept is same as servo motor where the driver for Servo motor can be used for the ESC. as the PWM signal Period should be around 20 ms where 5% duty cycle is minimum speed while 10% is maximum speed.

You can program ESC so that instead of having 5% minimum speed to be like 6% and same for the maximum.

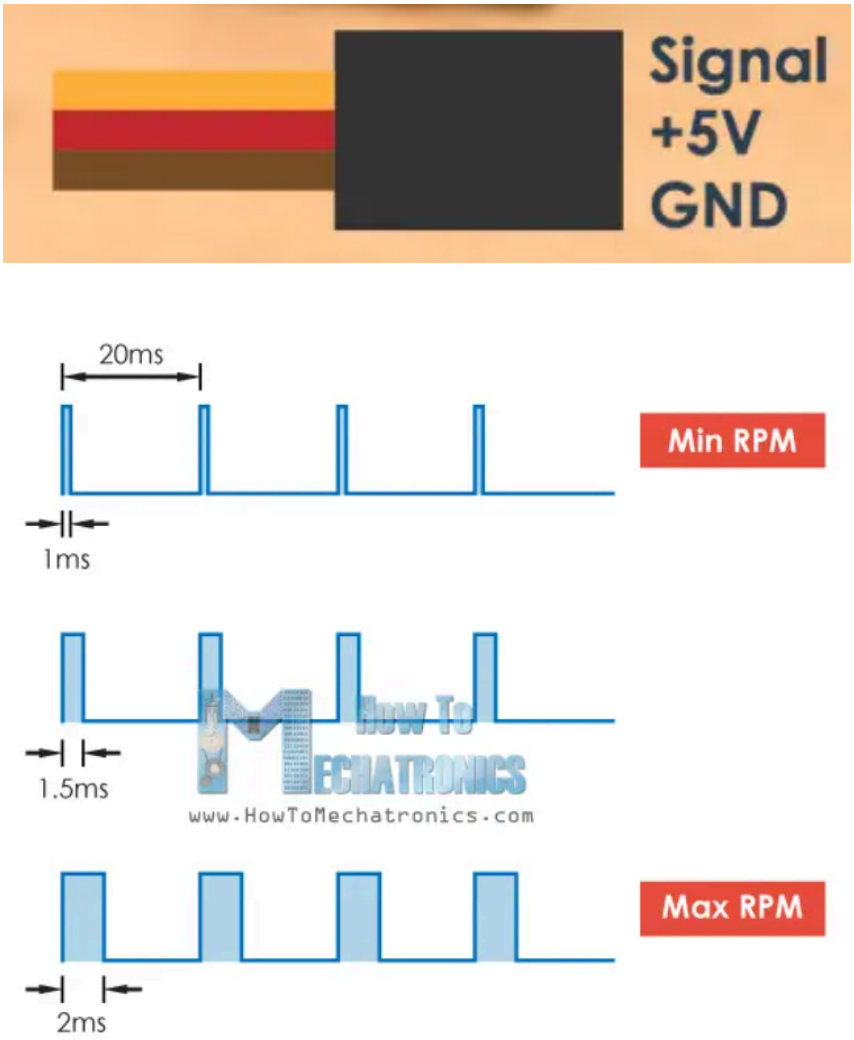


Figure 4.60 - ESC PWM signal

4.7.1.7 HC SR04

HC SR04 ultrasonic sensor is 5V Sensor where it has 4 pins (VCC, GND, Trig, echo) where Trig pin is input pin while echo pin is output pin. Since output pin ‘echo’ will be a 5V signal then we need the pin on MCU connected to that 5V signal to be 5V tolerant.

That module driver was written but wasn’t use because it took so much time to get distance and the maximum range is 400 cm.

Regarding the theory of operation, you should output a pulse on Trig pin of width more than 10 us. Then after small period of time, HC SR04 will output a pulse on echo pin whose width will correspond to the distance detected.

You can use input capture for pulse of timers to get the width of echo pulse where the width time of that pulse will correspond to the total time of sending and receiving ultrasonic signal back. The equations for transferring time into distance is self-explanatory in the next image.

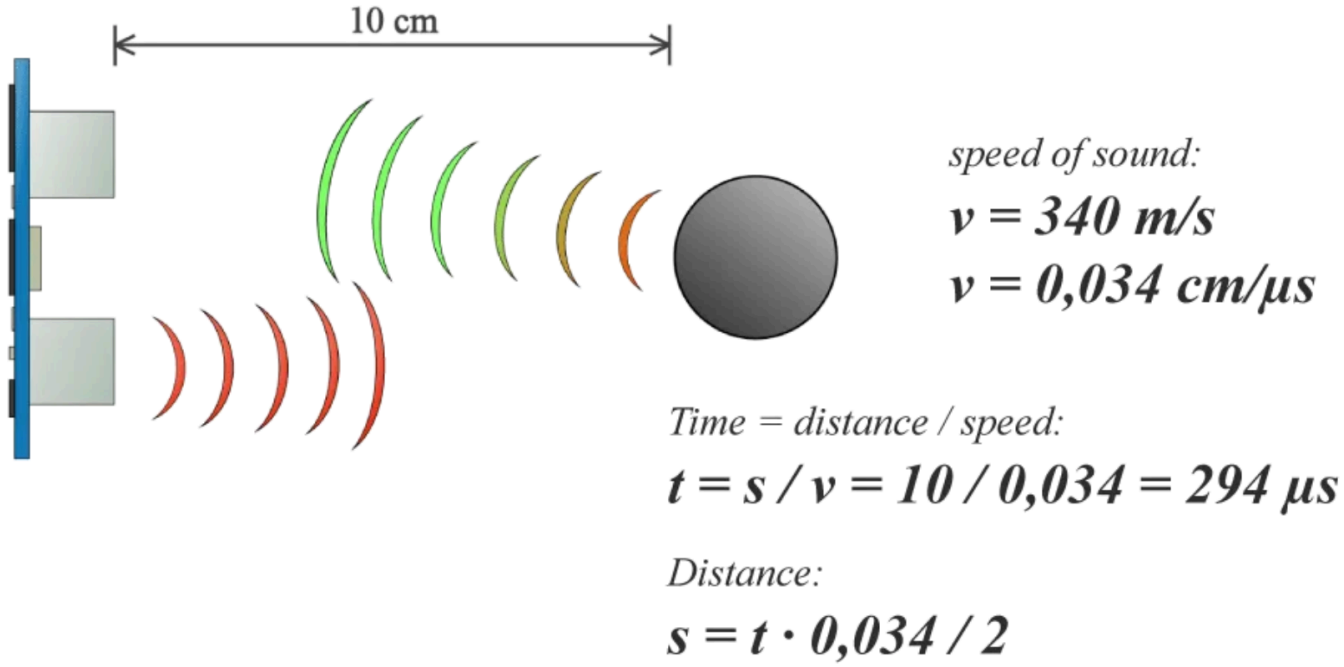


Figure 4.61 - HC SR04 equations

4.7.1.8 HMC5883L

The HMC5883L is a highly integrated 3-axis digital magnetometer designed for low-field magnetic sensing. It is widely used in various applications such as electronic compasses, GPS navigation systems, and motion tracking devices. This module measures the strength and direction of the Earth's magnetic field, allowing it to determine precise heading information.

Key Features of HMC5883L:

1. **3-Axis Magnetic Field Sensing:**
    - The HMC5883L can measure magnetic fields along the X, Y, and Z axes, providing comprehensive data on the direction and magnitude of the magnetic field.
2. **Digital Output:**
    - The sensor provides digital output data through an I2C (Inter-Integrated Circuit) interface, facilitating easy integration with microcontrollers and other digital systems.
3. **High Resolution and Accuracy:**
    - The HMC5883L offers high-resolution measurements, with a sensitivity of 5 milli-gauss per least significant bit (LSB). This high resolution allows for precise heading determination.
4. **Low Power Consumption:**
    - Designed with low power consumption in mind, the HMC5883L is suitable for battery-powered applications. It features power-saving modes to reduce energy usage when the sensor is not actively measuring.
5. **Wide Range of Magnetic Field Measurement:**
    - The sensor can measure magnetic fields in the range of ±8 gauss, making it suitable for various applications that require different sensitivity levels.
6. **Built-in Self-Test:**
    - The HMC5883L includes a built-in self-test function, which allows for verification of the sensor's proper operation and accuracy.
7. **Small Form Factor:**
    - The HMC5883L is available in a compact package, making it easy to integrate into small and portable devices.

Chapter 5: System Testing and Verification

Our system is a flight controller. Its main function is receiving commands from the flight computer and executing those commands while maintaining stable and safe flight. As a result, we need to get a quadcopter frame to mount our system on. Then, give it a simple command to lift up and hover in the air without losing balance.

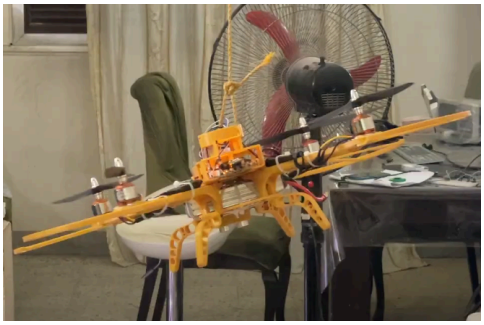
5.1. Testing Setup

For more safe testing, we had to do PID tuning first. The manual tuning of the system had to be done in a safe environment. Usually this is done using a gimbal. Due to importing constraints, we had to make our own test bed.

We made 3 main test beds:

1- Fixed-axis setup

This setup was done to be equivalent to a test rig that allows only one degree of freedom. The main goal of this setup is to tune pitch and roll PID controllers.
The body should be normally unbalanced. When the controller starts, it balanced the body successfully.



2- Pendulum setup

The main goal of this setup is to allow 3 degrees of freedom (rotational motion). This setup was used to tune the PID parameters of the roll, pitch and yaw controllers all together.

3- Free flight setup

This setup allows the whole six degrees of freedom. It is very dangerous due to space constraints. It is done for fine tuning and as a proof that the controller works perfectly.

5.2. Testing Plan and Strategy

The testing setup involved mounting our flight controller on a quadcopter F450 frame. The F450 frame was chosen due to its robust build and compatibility with various components. The frame was equipped with four brushless motors, Electronic Speed Controllers (ESCs), and a power distribution board. The sensors (MPU6050, HMC5883L, and BMP280) were calibrated and connected to the flight controller. The quadcopter was placed in an indoor space with minimal wind conditions to ensure safety and accuracy during testing.

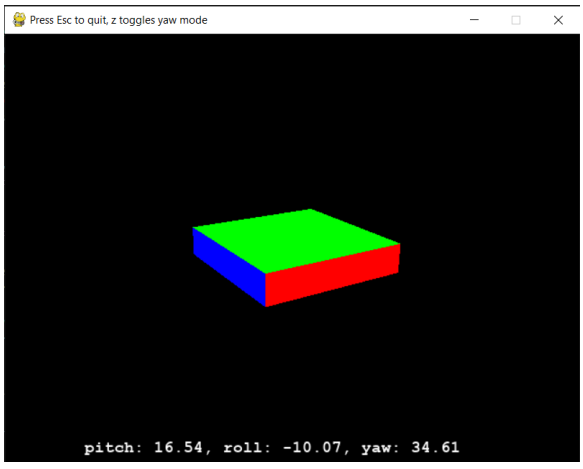
The testing strategy was divided into module testing, integration testing, and final flight tests. Each module was tested individually to ensure proper functionality before integrating them into the full system.

5.2.1. Module Testing

1. **Sensor Calibration and Testing:**
    - **Gyroscope and Accelerometer (MPU6050):** Calibrated for offset and scale factors. Static and dynamic tests were performed to verify accuracy.
    - **Magnetometer (HMC5883L):** Calibrated to remove hard and soft iron distortions. Verified against known magnetic field directions.
    - **Barometer (BMP280):** Calibrated for accurate pressure readings. Verified by comparing altitude readings with a known reference.
2. **Sensor Fusion Algorithm:**
    - Tested the 2D Kalman filter for altitude and vertical velocity estimation.
    - Tested the 1D Kalman filter for pitch and roll estimation.
    - Simulated sensor inputs and compared the fused output with expected results.
3. **Motor Control:**
    - Tested ESCs and motors individually to ensure proper response to throttle commands.
    - Verified PWM signal generation from the flight controller.

Results:

- The sensors showed accurate readings within acceptable error margins.
- The sensor fusion algorithm provided smooth and reliable state estimations.
- Motors responded accurately to control inputs, indicating correct ESC configuration and signal integrity.



Note: An open-source script was used for 3D rendering of the feedback after sensor fusion.

5.2.2. Integration Testing

Steps:

1. **Integration of Modules:**
    - Connected all sensors to the flight controller.
    - Integrated the sensor fusion algorithm with motor control logic.
2. **Ground Tests:**
    - Conducted ground tests to verify the interaction between sensors, flight controller, and motors.
    - Simulated flight commands and observed the response of the quadcopter while it was restrained to the ground.
3. **Flight Tests:**
    - Performed initial low-altitude hovers to fine-tune PID controller parameters.
    - Gradually increased the complexity of maneuvers, including take-off, hover, and landing.

Results:

- Ground tests confirmed that all modules communicated correctly and responded as expected.
- Flight tests demonstrated stable hovering and controlled flight, with the quadcopter maintaining balance and responding accurately to commands.
- Adjustments to PID controller parameters were made based on observed behavior, resulting in improved stability and responsiveness.

Chapter 6: Conclusions and Future Work

This chapter summarizes the entire project, highlighting its features and limitations, and provides directions for future work. Through this project, we aimed to develop a cost-effective flight controller using the CH32V203 microcontroller, optimizing both hardware and software components. Here, we discuss the challenges we faced, the experience we gained, the conclusions we reached, and potential future enhancements.

6.1. Faced Challenges

During the development of the QuadRaptor flight controller, we encountered several challenges:

- **Manual Tuning of PID Controller:** One of the major challenges was manually tuning the PID controller to achieve stable flight. This required numerous trial-and-error adjustments to find the optimal parameters that balance responsiveness and stability.
- **Safety Issues:** Ensuring safety during test flights was crucial. We had to wear protective glasses and use a gimbal test setup to prevent injuries from the drone's propellers. Additionally, we recognized the importance of implementing proper precautions, such as setting up a safe test environment and using propeller guards to protect ourselves from potential harm.

6.2. Gained Experience

Throughout the project, we gained valuable experience and skills, including:

- **PID Control and Control Loop:** We developed a deep understanding of PID control and the control loop of flight controller systems. This involved learning how to adjust PID parameters and how these adjustments affect the drone's flight characteristics.
- **System Dynamics of Quadcopter:** We acquired a thorough understanding of the system dynamics of a quadcopter, including how different inputs (throttle, roll, pitch, yaw) affect its movement and stability. This knowledge is crucial for designing effective flight control algorithms.
- **Embedded Systems Programming:** Working with the CH32V203 microcontroller enhanced our skills in embedded systems programming, particularly in optimizing firmware for performance and power efficiency.

6.3. Conclusions

In conclusion, the QuadRaptor project successfully demonstrated the feasibility of creating a cost-effective flight controller using the CH32V203 microcontroller. The main features and limitations are as follows:

Features:

- **Cost-Effective:** By carefully selecting components and optimizing both hardware and software, we achieved a cost-effective solution without compromising performance.
- **Stable Flight Control:** The implemented PID control algorithm provided stable and responsive flight control, making the drone suitable for various applications.
- **Custom Firmware:** The custom firmware optimized the microcontroller's performance, ensuring efficient use of resources and extending flight time.

Limitations:

- **Manual PID Tuning:** The process of manually tuning the PID controller was time-consuming and required a significant amount of trial and error.
- **Safety Concerns:** Ensuring safety during test flights was challenging, highlighting the need for improved safety measures in future iterations.

6.4. Future Work

To further enhance the our project, several potential extensions and improvements can be explored:

- Update the firmware of the flight controller board to be supported in one of the large open-source projects (ex: PX4, Ardupilot)
- Make the Drone Autonomous: Implementing autonomous flight capabilities would enable the drone to perform tasks without human intervention, increasing its utility for various applications.
- Using Drone Camera in AI Applications: Integrating a camera and using AI algorithms for tasks such as object recognition, tracking, and navigation can significantly expand the drone's capabilities.
- Change BLDC Motors for Heavy Lifting: Upgrading to more powerful BLDC motors would allow the drone to lift heavier objects, making it suitable for logistics and delivery applications.
- Perform 3D Mapping: Implementing 3D mapping capabilities would enable the drone to create detailed maps of areas, useful for surveying, inspection, and environmental monitoring.
- Alternative to GPS for Localization: Developing an alternative method for drone localization, such as visual odometry or using radio signals, would provide reliable positioning in GPS-denied environments.

Appendix A: Development Platforms and Tools

Multiple testing tools and platforms were used and we will mention some of them as follows.

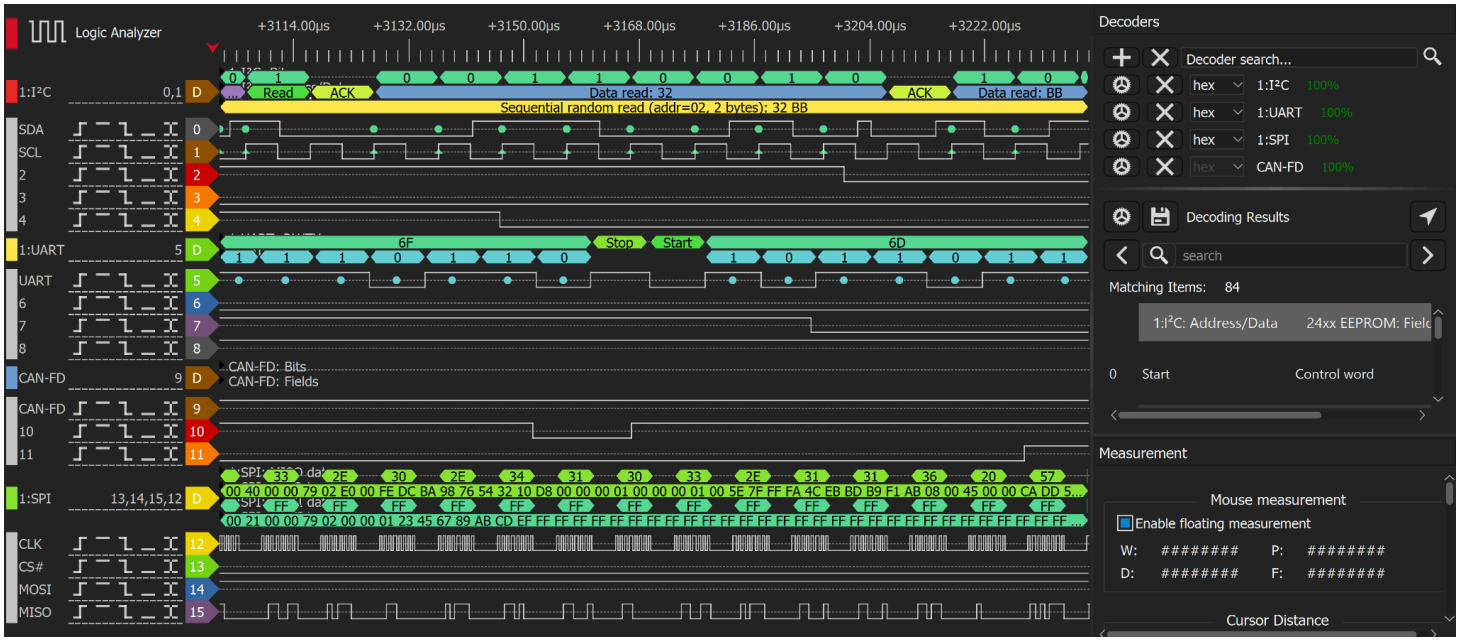
A.1. Hardware Platforms

A.1.1 Oscilloscope

The oscilloscope was used to examine the real-time behaviour of signal and to examine rise and fall time of different buses like I2C to verify it follows the standard Specification of I2C standards and verify signal integrity. We used sds1104x-e oscilloscope.

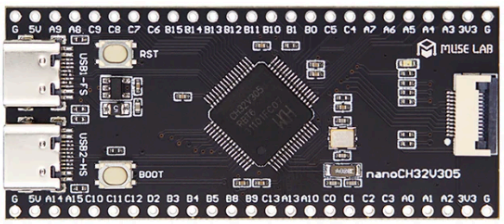
A.1.2 logic analyzer

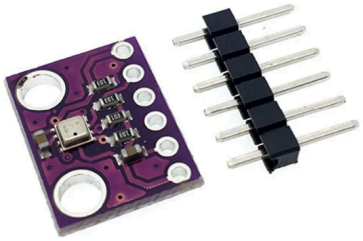
The logic analyzer is used to examine whether the correct data was written on bus or not and usually it has more channels than the oscilloscope. We used DSlogic U16pro.

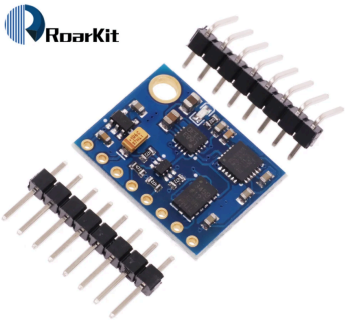


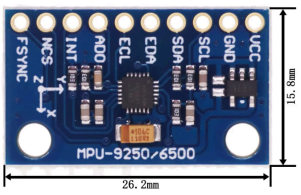
A.1.3 modules

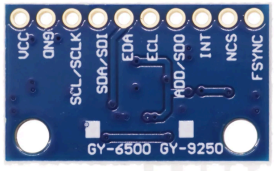
IC modules were used for fast software development until the final PCB was made.











A.1.4 debugger

Since Ch32V203 support serial debugging then we used WCH link debugging which was very helpful in determining MCU faults, correct initialization sequence, etc…

A.1.5 microscope and soldering station

Microscope and soldering station were both useful for SMD soldering components on the PCB board. We used Amscope SM745NTP and KADA852+.

A.1.6 power supply.

A variable power supply was used to test the how much the system needs power and for system modeling of the brushless dc motors.

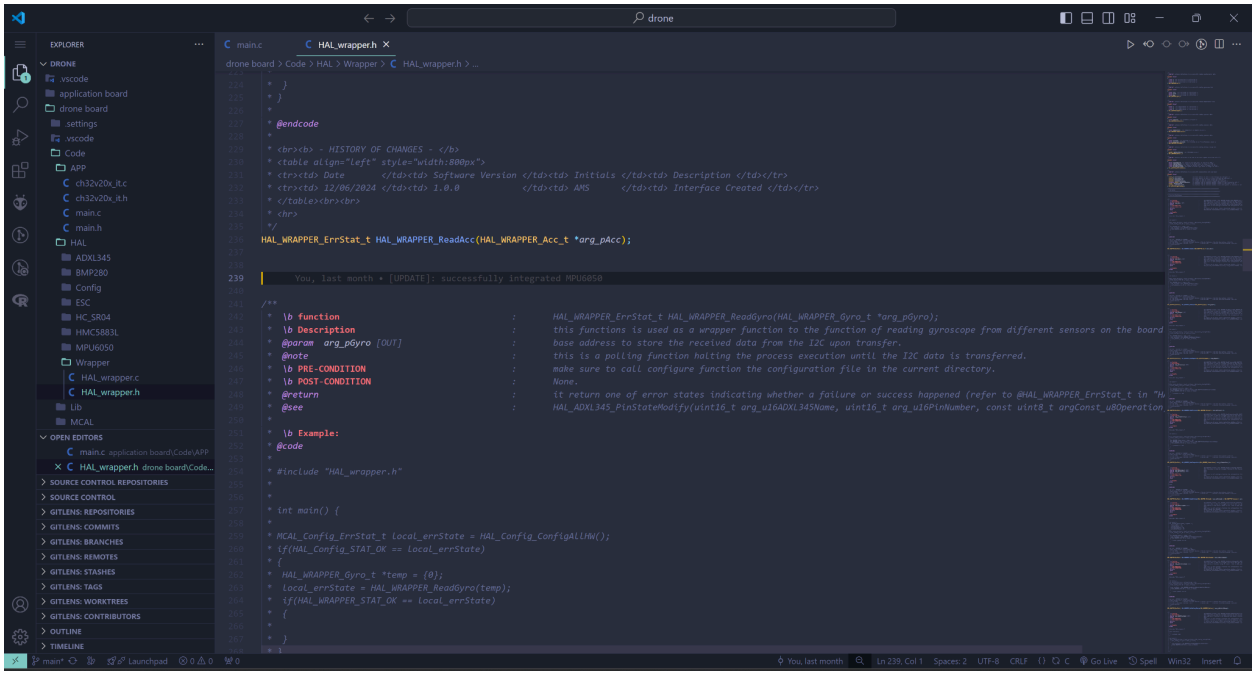
A.1.7 Weight Scale.

For measuring how much thrust does each motor can give at different speeds for system modeling.

A.2. Software Tools

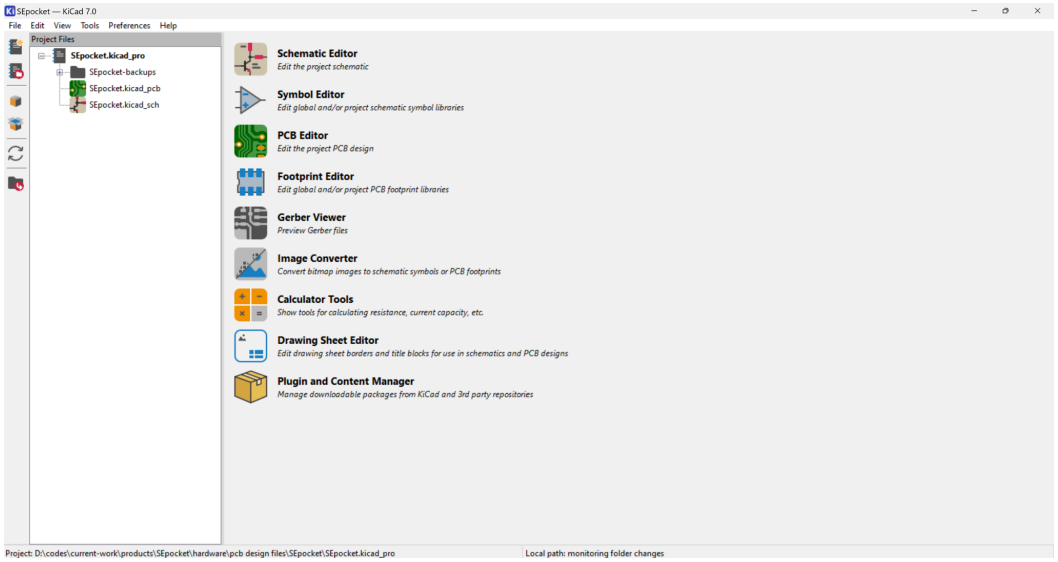
A.2.1 Vs Code.

For ease of editing code and navigating file.



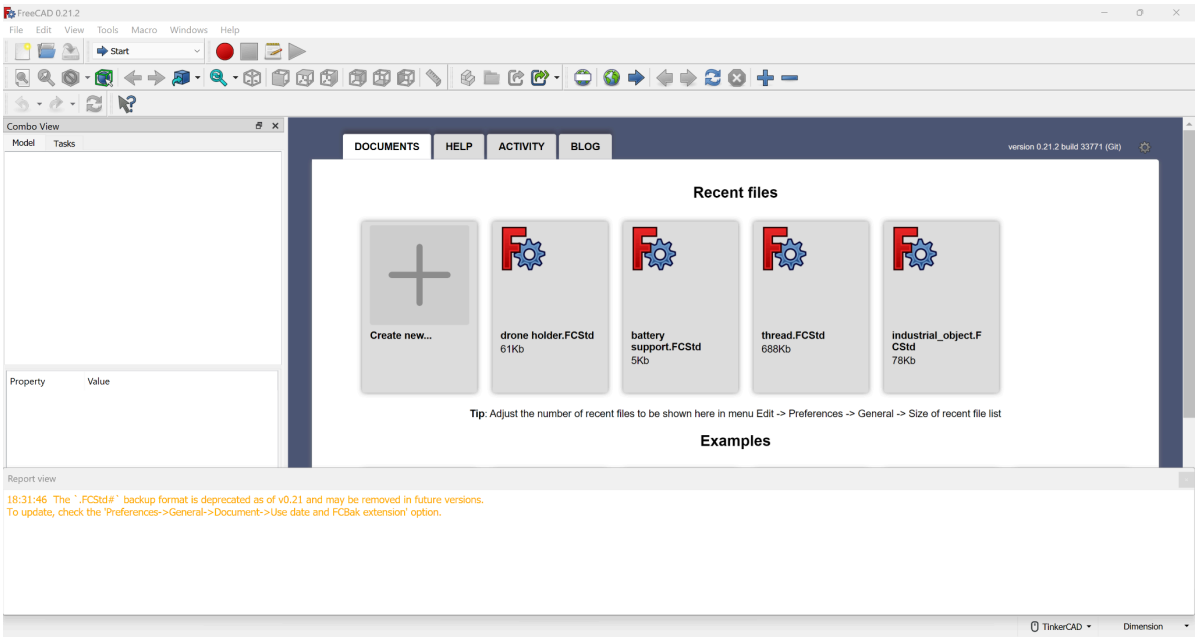
A.2.2 KiCad.

For desiging PCB, routing tracks and laying ICs on the PCB.



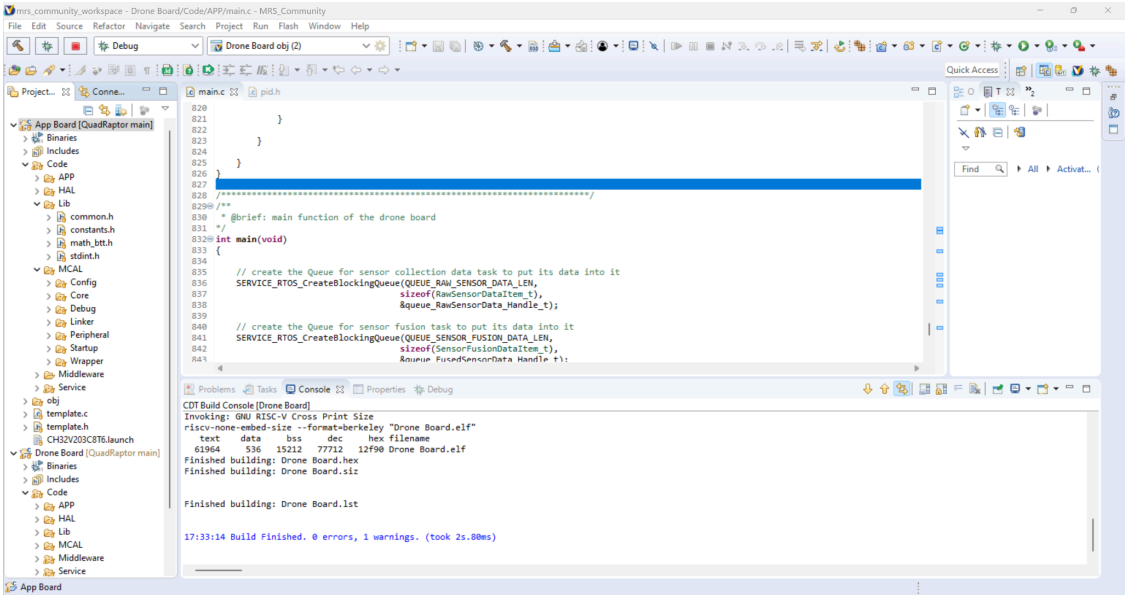
A.2.3 freeCAD.

For desiging 3D models to be printed.



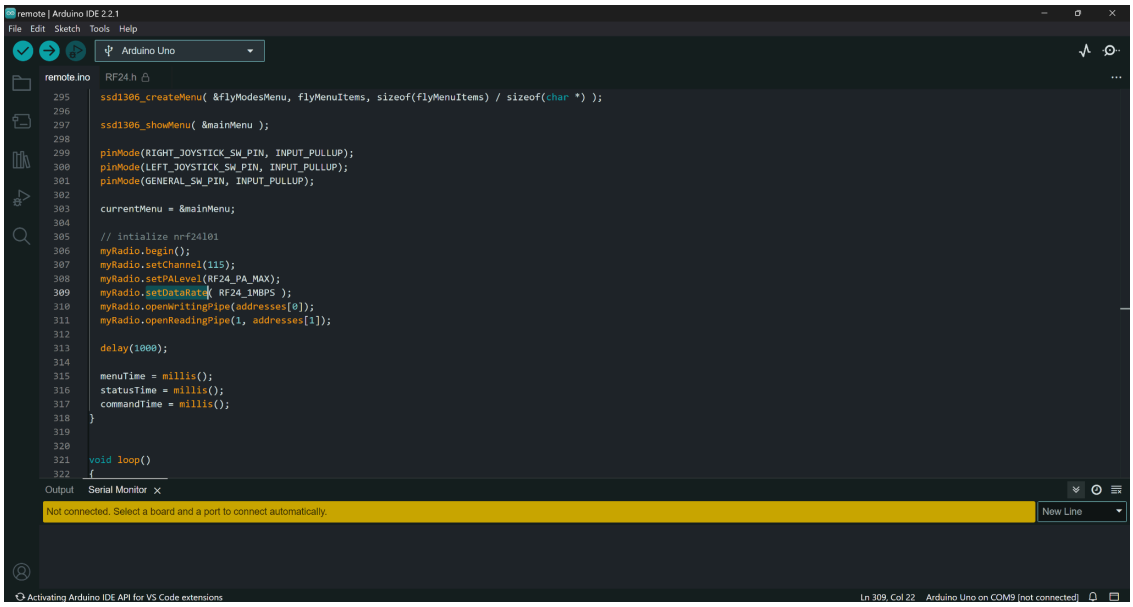
A.2.4 MRS community.

For ease of debugging CH32 MCU and downloading code.



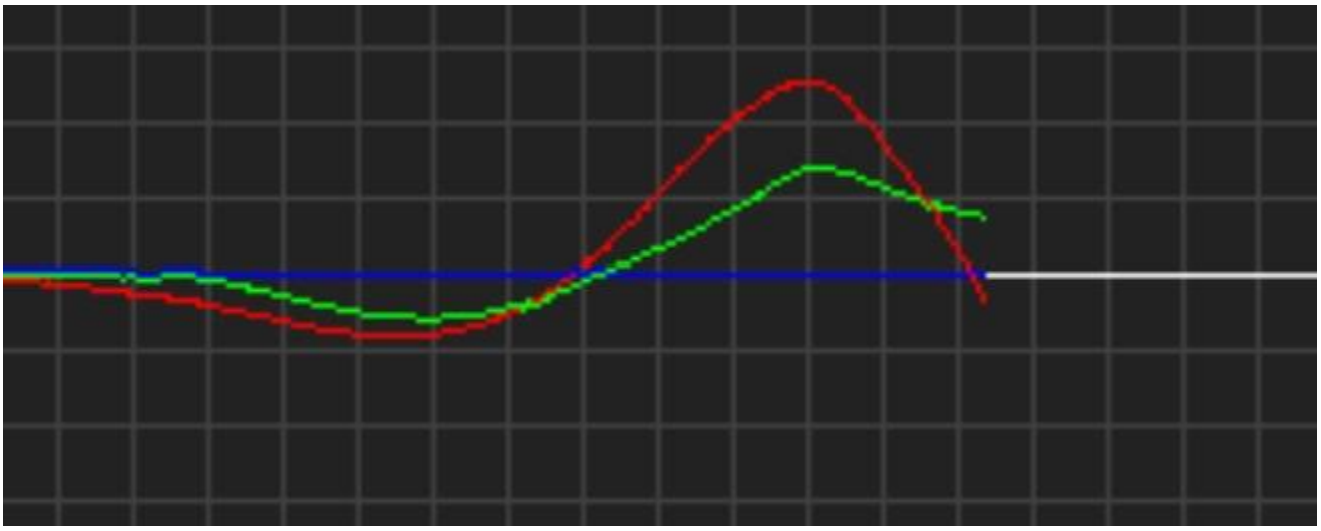
A.2.5 arduino IDE.

For writing code for the remote control.



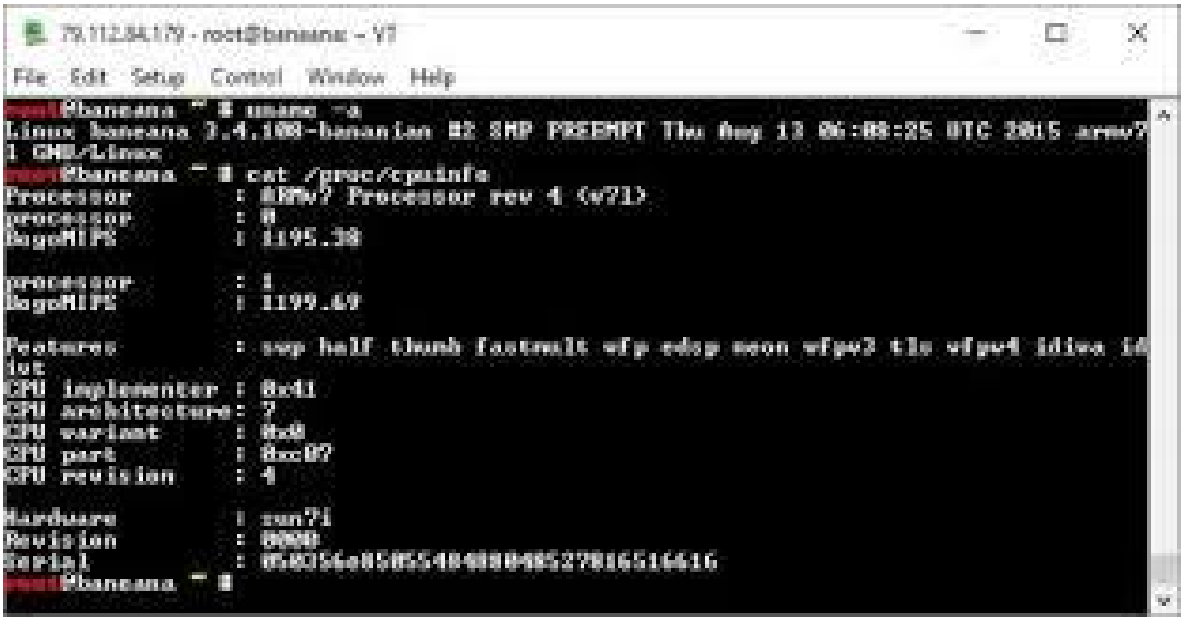
A.2.6 Serial Oscilloscope.

For plotting serial data and examining system response.



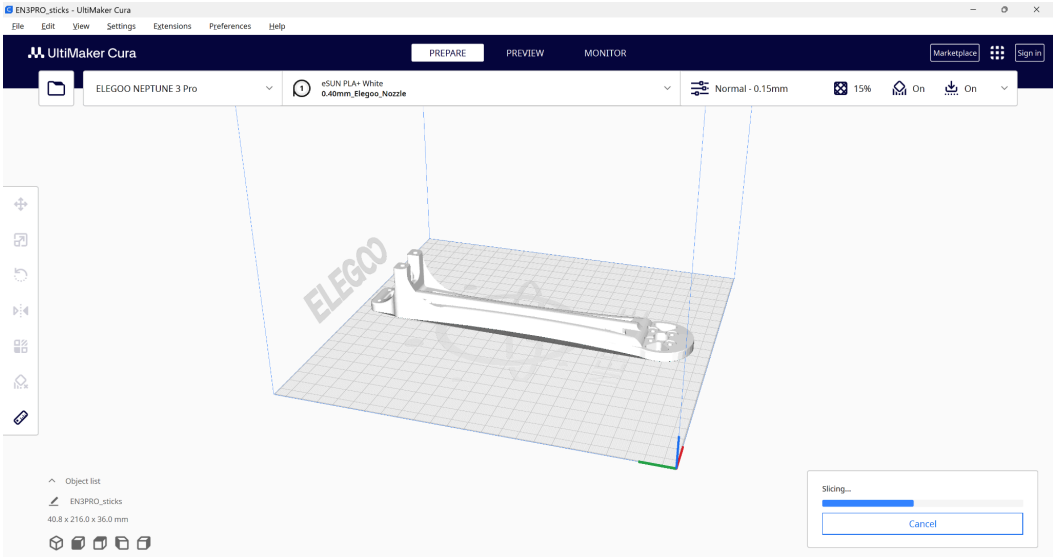
A.2.7 Tera Term.

For printing serial messages.



A.2.8 ultimaker cura.

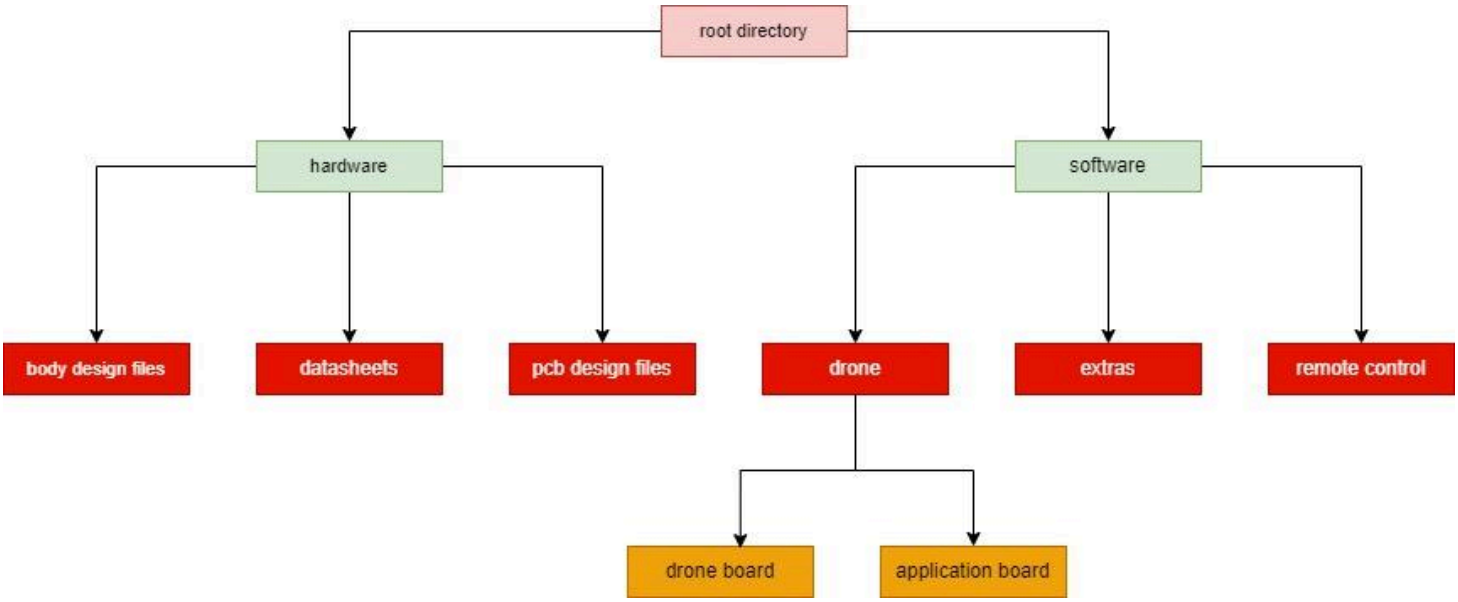
For slicing 3D models to be printed.



Appendix B: Use Cases

- **Recreational Drone Piloting**: Hobbyists and enthusiasts use drone controllers to pilot and maneuver their personal drones for fun, photography, and videography. The controller allows them to precisely control the drone's movements, camera angles, and various flight modes.
- **Commercial Aerial Photography/Videography**: Photographers and videographers employ drone controllers to capture aerial footage and photographs for commercial purposes, such as real estate, construction, events, and filmmaking.
- **Agricultural Monitoring and Spraying**: Farmers and agricultural businesses use drone controllers to monitor crop health, detect pests, and apply pesticides or fertilizers precisely from the air.
- **Search and Rescue Operations**: Emergency response teams utilize drone controllers to pilot drones during search and rescue missions, providing a birds-eye view of the search area and locating missing persons or assessing disaster zones.
- **Infrastructure Inspection**: Inspectors and maintenance crews use drone controllers to remotely inspect and monitor infrastructure, such as power lines, bridges, and cell towers, without the need for costly and dangerous manual inspections.
- **Delivery and Logistics**: Drone controllers are used to pilot delivery drones that transport small packages and goods over short distances, improving efficiency and accessibility.
- **Surveying and Mapping**: Surveyors and geospatial professionals employ drone controllers to capture high-resolution aerial imagery and data for mapping, land surveying, and 3D modeling applications.
- **Public Safety and Law Enforcement**: Police, firefighters, and other public safety agencies utilize drone controllers to deploy drones for surveillance, incident response, and hazardous environment monitoring.
- **Environmental Monitoring**: Scientists and environmental agencies use drone controllers to pilot drones for tasks like wildlife monitoring, forest management, and pollution tracking.
- **Military and Defense Applications**: The military and defense sectors employ sophisticated drone controllers for the operation of unmanned aerial vehicles (UAVs) in various strategic and tactical scenarios.

Appendix C: User Guide



open either drone board or application board directory in MRS community then connect the debugger to the intended board where PA14 -> SWCLK and PA13 -> SWDIO. Navigate then to MRS community where you can upload and debug the code.

Appendix D: Feasibility Study

The cost of controller of drone board is as follows:

CH32V203	0.5$
HMC5883L	2$
MPU6050	4.4$
BMP280	1.7$
Extra switches, SMD resistors, SMD capacitors	2$
PCB fabrication	2$
Total	12.6$

Table 4.7 - components pricing

Which makes our controller board 4x cheaper than the nearest competitor.