Indoor Line Following Drone Project

Team 12

Author(s)

Saranraj Sasikumar (23810317)

Ashintha Akalanka Metaramba Kanatta Gamage (24080233)

Kasuni Gayara Sooriyaarachchi (24119291)

Dhrumil Ghanshyambhai Bhanderi (24047059)

Gopikrishna S Menon Sadiesh Menon (22886126)

Jothis Jacob (23800038)

Final Report

Project Partners: University of Western Australia Aviation Labs (UWAAL)

Australian National Fabrication Facility (ANFF)

UWA Supervisor: Mr. Gregory Crebbin

Team Number: 12

Group Meeting Day and Time: Thursday 11:00

Word Count (Disregarding Appendix): 19,835

Word Limit: 21,600

Rev 2.0

Revision History

Date	Version	Description	Author
02/10/2024	0.0	Coversheet added. Updated all main topics and added the acknowledgement and Introduction of report	Ashintha Akalanka M. K. G.
21/10/2024	1.0	Contribution towards completion of all parts	Team
21/10/2024	2.0	Integration of all parts	Ashintha Akalanka M. K. G.

Executive Summary

This report involves a comprehensive analysis and design of an indoor line-following drone initiated by UWAAL, aimed at being capable of performing remote safety inspections within the ANFF laboratories. These laboratories present unique challenges, such as the absence of GPS signals for navigation and stabilization, thus necessitating a solution that depends on onboard sensors for path tracking, hovering and obstacle avoidance.

The project team has conducted a detailed study of the relevant literature and technology to define the primary design requirements. These requirements are thoroughly outlined in the report with a focus on safety, autonomous line-following and obstacle avoidance, stable altitude control (adjustable between 0.5m and 2m) and a communication system for manual control and emergency landing.

Over the 12-week period, the team conducted a comprehensive design analysis, component selection, system integration, and testing to ensure that the drone meets the specified requirements. The drone's design was mainly divided into two:

- Drone Module: This involved the design of the drone based on selected components and included the selection of the frame, motors, propellers and flight controller
- Line-Following Module: This involved the design of the subsystem which would implement the line following algorithm and included the selection of sensors and microcontroller

The team conducted a weekly budget check and optimization to ensure the design remained within the allocated AUD 350, balancing cost with product performance. The results of the design are discussed in the report, highlighting the ability to meet most of the performance criteria.

Overall, the report provides UWAAL with a practical, cost-effective solution for indoor safety inspections, along with comprehensive design documentation and recommendations for future iterations. The total cost of the project remained within the budget, ensuring that the solution is both affordable and applicable to other indoor applications.

Acknowledgments

The members of Team Number 12 would like to sincerely thank The University of Western Australia, The University of Western Australia Aviation Labs and also to Australian National Fabrication Facility (ANFF) for the opportunity given to undertake such a challenging but educational project.

We would also like to thank Mr. Dilusha Silva and Mr. Greg Crebbin for his support as a supervisor throughout the project towards accomplishing this endeavor successfully. The team also provides gratitude to Mr. Jega Gurusamy for his valuable feedbacks for the Technical Queries and the preliminary design review presentation held on the 29th of August 2024. Additionally, the 3 guest lecturers Mr. Paul Grainger, Prof. Tim Mazzarol and Mr. Doug Hamilton should be thanked for sharing their expertise with potential future engineers of Australia.

Furthermore, the combined effort of the team members should also be commended for contributing every week despite the busy schedules to eventually provide a successful and justifiably demonstratable project design to meet the client demands.

Table of Contents

List of Figures

List of Tables

1. Introduction

The introduction section of report discusses an overview of the undertaken design in-door line following drone project which is inclusive of the project background, purpose and scope of the design project, a summary of the report structure topped along with each team members contributions and responsibilities.

1.1 Project Background

The following project includes the design and implementation of an indoor line following drone, as part of a collaboration between the University of Western Australia Aviation Labs (UWAAL) and the Australian National Fabrication Facility (ANFF) laboratories, for the main purpose of carrying out autonomous remote safety inspections. Due to operations occurring in confined areas coupled with unreliable GPS signal, the following drone is expected to travel along a predefined pathway, approximated to be 200m, with inbuilt methodology for obstacle detection and avoidance [1]. With design demonstration to be carried out in the vicinity of the lycopodium laboratory in the Monadelphous building, the following report discusses findings from each stages of the indoor drone design life cycle.

1.2 Project Purpose

The purpose of the following design project, in summary, was to physically design an indoor line following drone with purposes of autonomously following a predefined pathway (a dark path on a light surface or a light path on a dark surface) utilizing image processing techniques with the option of switching to a remote / manual mode as during the flight period. After a critically tested and functioning verified period of 12-weeks, the following indoor line following drone needs to demonstrate its capabilities of hovering on any location for a 30 second window, follow a path of 200m autonomously at any pre-set altitude between 0.5m to 2m and detect obstacles which appears within 20-50cm of its vicinity, all while ensuring the safety of the users and bystanders with emergency stop and manual control modes hovering on spot and obstacle avoidance [1] [2].

The design was categorized as hardware and software initially with the plan of compatible component selection to be carried out in the hardware section and relevant integration and codes to be generated by the software team to obtain a collaborated output in the process. Based on these classifications, a critically reviewed requirement analysis was carried out to identified a well-established list of 20 major requirements along with 9 additional requirements to enhance the output product developed by the team to which the design was carried out to be full-filled.

In carrying out the following project, as part of the ELEC5552 unit, it also grants the members key learning outcomes including the application of engineering synthesis and design processes, development of real-life problem-solving skills in aspects of both technical and professional skills along with assistance in fostering accountability, responsibility and communications in projects attempted in a group environment.

1.3 Project Scope

The following section details the developed design for the indoor line following drone which was planned to be capable of path following, hovering on spot, obstacle detecting with a feature for switching between autonomous flight and remote-controlled flight. Through literature review, requirement analysis and continuous technical query meetings for generic scope clarifications, the design was adapted accordingly to meet and full-fill the following deliverables in a successful completion scenario [1]:

- Line following drone with both autonomous and remote-control capability and obstacle avoidance feature, developed with the use of open-source software and excluding any DJI drone components. The designed drone is also expected to maintain a stable altitude at 1.2m with capability for adjustments between 0.5m to 2m on request.
- Finalized design files including any simulations, codes (used for sensor programming, image processing programming and overall integration) and any Printed Circuit Board (PCB) layouts printed (if any)
- Bill of materials for components, along with physical bills to be submitted not exceeding

a budget of AUD 350.00

- A final detailed report including component selection, requirement identification and verification and validation results carried out to confirm the requirements achievement
- A demonstration of the drone flight, or all working attributes of the drone, to be carried out at the Lycopodium laboratory, on the 22nd of October 2024, for a 200m pathway and a 30 second hovering test, with the entire model being powered by a 2S battery not exceeding 800mAh capacity with capability of USB chargeability.

1.4 Project Structure

The following report details the entire design process of the indoor line following drone project [3]. Chapter 2 summarizes the output of the requirement analysis re-evaluated and re-prioritized to re-determine the critical requirements of the following design project. This is followed by Chapter 3 compromises of the design architecture providing a high-level overview of the final design in-terms of the software and hardware sections. Chapter 4 highlights the design philosophy which discusses the principles and processes which guided the team throughout the drone designing, with justifications for the rationale behind the design choices and the evolution of the design over the 12 weeks.

Chapters 5-7 delves into a heavily technical discussion related to the design of the drone. Chapter 5 includes a critical review and analysis procedure carried out by the team in regards to the component selection for the drone with its impact on the project requirements. Chapter 6 compromises of the design validation and verification section of the report which provides a detailed analysis of the process, tests and results for requirements fulfillment. Chapter 7 analyzes the integration of the overall system to achieve overall functionality of the drone as a whole including issues faced during the design.

Chapters 8-12 focuses on the project management aspects of the project with Chapter 8 highlighting stakeholder engagements carried out throughout the project including changes carried out after design reviews, Chapter 9 discussing the safety issues identified during the design prototyping and construction process, Chapter 10 identifying ethics and standards related

to the design, Chapter 11 providing a breakdown to the assembling process of the drone construction, Chapter 12 critically analyzing 5 critical risks identified through the risk assessment carried out throughout the project lifecycle and finally Chapter 13 presenting and tabulating the breakdown of the budget allocation of the project.
Chapter 13 highlights the key design outputs including the technical datasheets and pathway to assess any electrical schematics and codes utilized for the project. Finally, in Chapter 14 the report provides a comprehensive review for further development of the project, with recommendations for better optimization and a user manual for the product use by potential users highlighting precautions to be taken included.

1.5 Group member responsibilities and contributions summary

1.5.1 Saranraj Sasikumar contribution

Ultrasonic and Barometric sensor selection, Microcontroller and FC integration, Camera and FC and MC integration

1.5.2 Ashintha Akalanka Metaramba Kanatta Gamage contribution

Introduction (Chapter 1), Evolution of Design (Chapter 4.1), Drone Frame Selection Analysis (Chapter 5.5), Battery and Energy Calculations (Chapter 5.6), Ultrasonic Sensor (Chapter 6.4), Barometric Sensor (Chapter 6.5), Accelerometer (Chapter 6.6), Design Validation and Verification (Chapter 7), Stakeholder Engagement (Chapter 8), Safety Issues (Chapter 9), Ethical Issues (Chapter 10), Risk Management (Chapter 11), Construction Process Explanation (Chapter 12), References and Overall Formatting of the report.

1.5.3 Kasuni Gayara Sooriyaarachchi contribution

Design Validation and Testing (Chapter 7) (Path Following, Image Processing, Accurate Line, Detection, and Mid-Point Calculation- (Chapter 7.3)), Design outputs section (Chapter 13), Design architecture section (Chapter 3), Finalized project requirements section (Chapter 2) Camera selection (Chapter 5.3), Cost of Design (Chapter 14), Path finding and Image Processing

(Chapter 6.3), Image Processing Algorithm (Chapter 6.3.1), Obstacle detection flowchart (Chapter 6.4). Accelerometer Flowchart (Chapter 6.6), Architecture for overall drone integration (Chapter 6.8)

1.5.4 Dhrumil Ghanshyambhai Bhanderi contribution

Microcontroller selection, Accelerometer and Battery Charger selection, User Interface, Testing and Validation results of assistance in sensors, UI and assembling process

1.5.5 Gopikrishna S Menon Sadiesh Menon contribution

Flight Controller, User Manual, Future Scope and Recommendations, Build Tests and Conclusion

1.5.6 Jothis Jacob contribution

Executive Summary, Design Architecture, Design Philosophy, Design Elements (Frame, Motor, Propeller) and Integration Limitations

2. Finalized Project Requirements

In the 4th week of the project, a comprehensive requirements analysis was executed to finalize the requirements. Following an extensive background review, questions, and feedback on technical queries and project partner requirements, 20 high-prioritized requirements and another 10 additional requirements were recognized, each accompanied by a delivery plan at that stage. The requirement analysis document 'ELEC5552 Requirement Analysis Final Report - Group 12' discusses, in detail, the procedure behind the priority selection of each requirement.

However, with the constantly changing environment of technical queries' feedback as the project flows, some of the delivery plans were adjusted, and some requirements were not fully met as desired. Those alternations are described and justified with reasons as the project progresses.

The prioritization for all identified requirements was based on the following matrix.

	Level 1	Level 2		Level 3	Level 4	Level 5
Weighting	Low (0-30%)	Very High Priority	High Priority	Medium Priority	Low Priority	Low Priority
	Medium (30-60%)	Very High Priority	High Priority	High Priority	Medium Priority	Low Priority
	High (60-100%)	Very High Priority	High Priority	Medium Priority	Medium Priority	Low Priority

Figure 1 - Matrix for priority categorization

Based on the above matrix, the priority levels were identified using the definitions below based on the Requirement Analysis report submitted by Team 12 in Week 4 [4]

Table 1 - Finalized Priority Categories

Score	Definition
	Requirements with this score are critical for the project flow and most attention is given to this requirement with most attention and resources. Any compromise

	might introduce a safety, environmental, and legal liability risk.
	Requirements with this score are important for customer satisfaction and standard compliance. Any compromise might result in an impact on the design safety and project flow.
	Requirements contribute to operational efficiency and enhanced performance. Any failure might result in client and customer dissatisfaction, compromised performance, and moderate safety risks.
	Requirements contribute partially to the whole project. Deviation from this project may cause minor inconvenience, but there won't be any major threats to the project.
	Requirements are desirable features but not crucial to the overall project. These requirements can be implemented to add an advantage to the design.

The following table outlines the final requirements in order of the determined priority [4].

Table 2 - Prioritized Final Requirements

S. No	**New ID**	**High Priority Requirement**	**Requirement Description**	**Priority**
1	1	Physical Damage Prevention	The drone's propellers and fast-spinning motors might cause physical injury to the operator or others. To prevent this, propeller guards must be used as a protective mechanism.	
2	2	Flight Safety	The drone requires a manual control option for safety in case of communication failure or software malfunction.	
10	3	The line following Path	The drone must follow a predefined path accurately and smoothly with all the sharp turns and curves.	
7	4	Image	Proper algorithm development and camera	

		Processing	interaction ensure clear path following by distinguishing the path from the floor using contrast detection.	
2 (other)	5	Accurate Line Detection	The drone must separate the path (White color) from the background (Dark/black color floor).	
1 (Other)	6	Midpoint Calculation	The image processing algorithm must be able to find the midpoint between the detected line segments to keep the drone above the path.	
5	7	Hovering Capability	The drone should hover for approximately 30 seconds at a height deviation of no more than 10 cm. Accurate control is needed to maintain the altitude while hovering.	
16	8	Precautions for communication loss	If communication is lost for over 30 seconds, the drone should land safely with a drone should land safely propeller turned off. E-stop needs to be implemented for that.	
3	9	Obstacle Avoidance	The drone should be equipped with an ultrasonic sensor to detect obstacles within a 50cm range. The sensor measures the distance between the object and the drone and gives commands for necessary movements to avoid obstacles.	
12	10	Budget	The project's total cost must be completed within a budget of AUD 350.	
17	11	Proper laboratory	knowing proper practices ensure safety of team members limiting risks during	

		Induction	construction.	
6	12	Travel Distance	The drone must cover 200m with the battery capacity provided. This ensures that energy management is sufficient for full functioning.	
11 (Other)	13	Sensor connection	For ultrasonic and pressure sensor we need to define protocol	
8	14	Altitude Adjustment	The drone must adjust altitude between 0.5m and 2m before initiating autonomous flight.	
14	15	Different speed options	The drone should have different speed options based on the battery capacity.	
19	16	Software and Hardware restriction	Any products used must not be DJI and should be within budget constraints.	
4	17	Battery Charger	The battery should be compatible with an easily charged, commercially available USB charger that is affordable.	
6 (Other)	18	Application interface	Creating a GUI interface for the drone's interaction with all functions.	
18	19	Drone lifespan	The drone components should have a minimum lifespan of 1 year, ensuring durability and cost of replacement.	
13	20	Drone frame and size constraint	The drone must be between 10-30 cm, constrained by battery size and budget.	

3. Design Architecture

The line-following drone is categorized into two major sections to finalize the project requirements. Hardware design discusses the connection between all the hardware components, while software design describes the modules which have been connected to each other through programming platform. Each module's design architecture is discussed in detail below:

3.1 Hardware Design Architecture

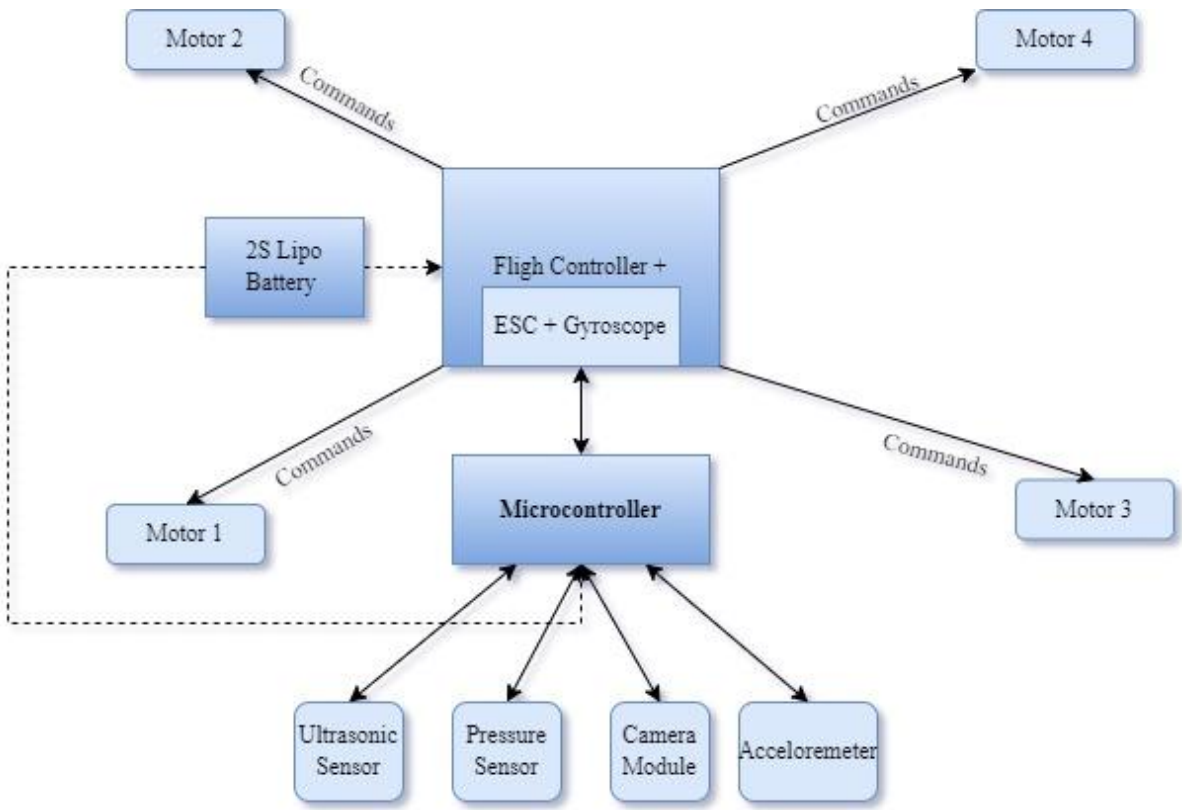


Figure 2 - Hardware Design Architecture

The design architecture of the drone module was developed based on the requirements and constraints introduced by the budget of AUD 350. It involves a flight controller, microcontroller, four motors, a battery, a camera module, an Ultrasonic sensor, a pressure sensor, and an Accelerometer. The block diagram depicts the interaction between the battery and the flight controller, followed by each motor. Likewise, the connection between the three sensors and the camera is also highlighted. The dashed lines show the power flow from the battery to the flight controller and microcontroller.

The flight controller selected involved an inbuilt ESC (electronic speed controller), and a gyroscope to save on separate budget allocation. The propellers and frame are not included in the block diagram as they are structural elements, and there is no flow of information or power between them. However, they play a vital role in the module.

3.2 Software Design Architecture

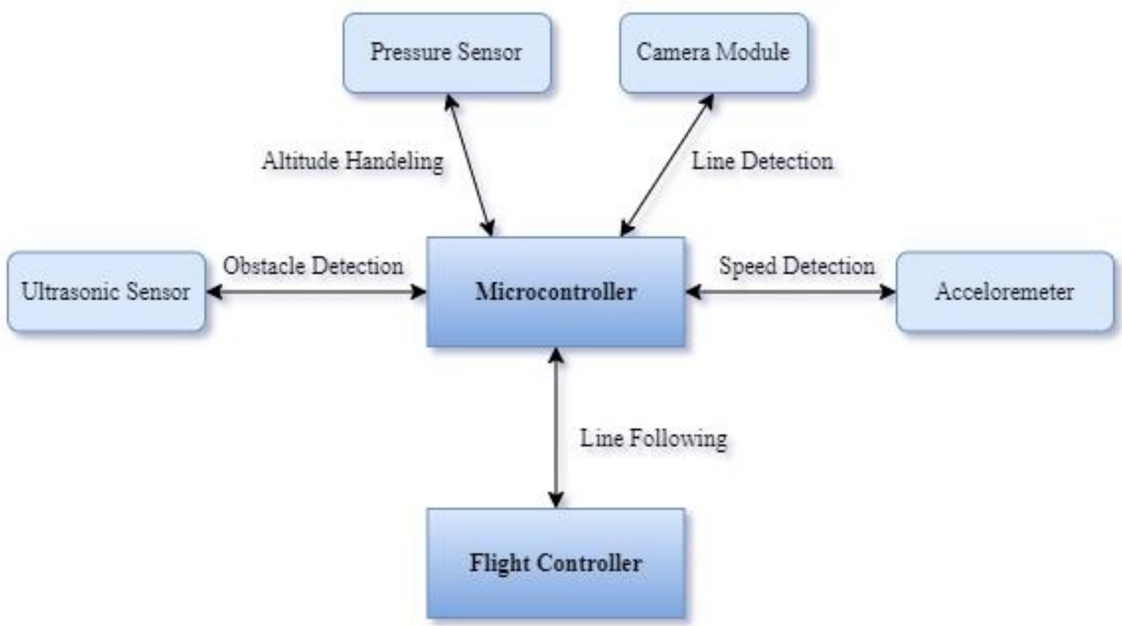


Figure 3 - Software Design Architecture

The design architecture of the software is depicted above in the block diagram. It involves the integration of multiple sensors to ensure smooth and stable navigation. A line-detecting algorithm using image proceeding processes the data captured by the camera to recognize the contrast between the line and the surrounding surface. These signals are processed in the microcontroller, and corresponding signals are sent to the flight controller to follow the line. The system incorporates an ultrasonic sensor for obstacle detection. The pressure sensor is used for altitude control. It is used to determine the drone's height above the ground in order to maintain a stable altitude. This data is fed into the microcontroller, which sends signals to the flight controller, providing corresponding motor thrust to keep the drone at the desired altitude. The accelerometer stabilizes the drone during flight by detecting changes in the drone's orientation and movement.

4. Design Philosophy

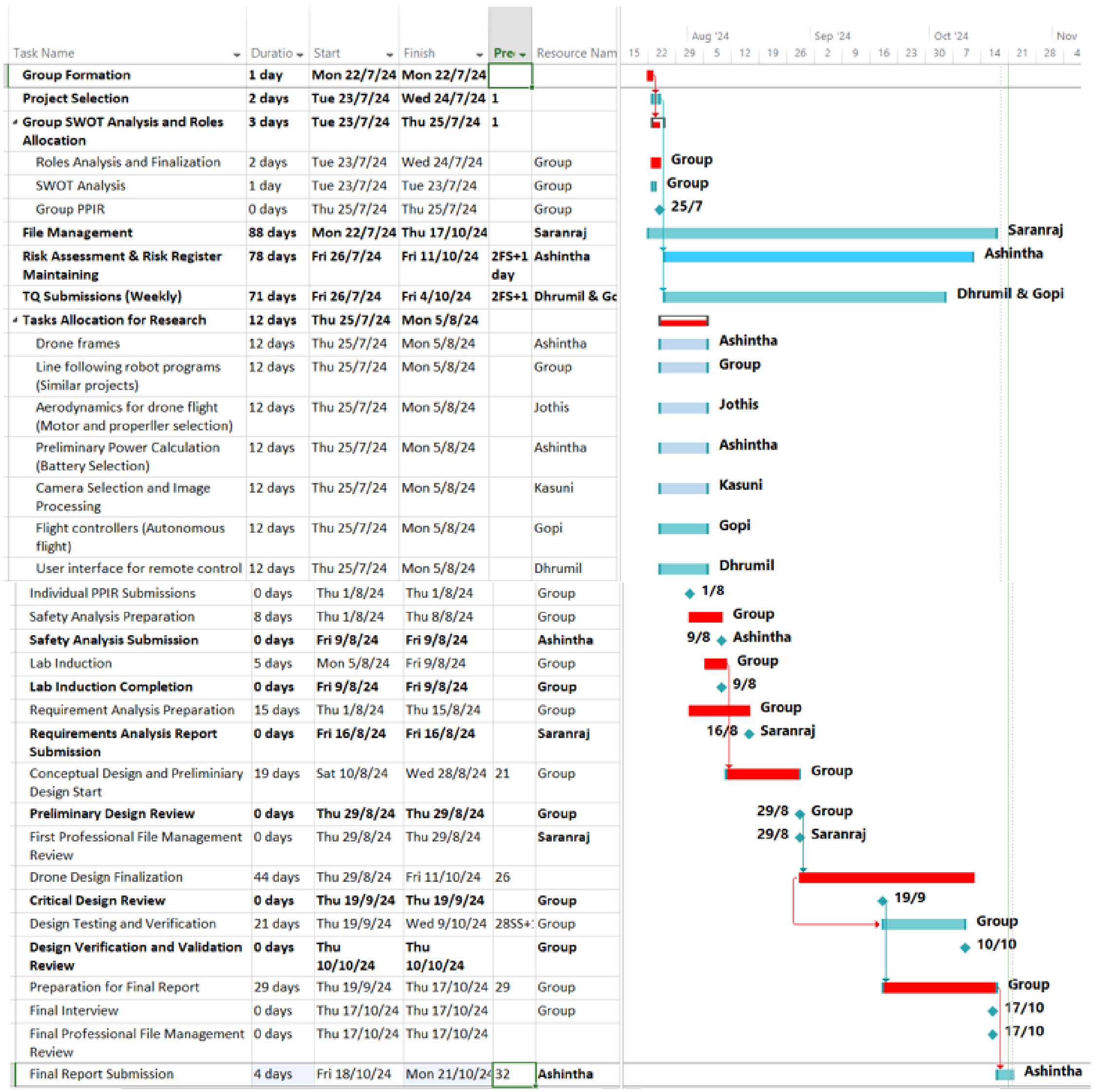
The design philosophy for the indoor line-following drone establishes a set of guiding principles to ensure the system's operational success, safety, and reliability in performing autonomous inspections. These principles help justify the design choices and make sure the drone meets the required performance standards in an indoor environment. The principles were categorized as follows:

- **Safety**: The drone is to operate in confined spaces with potential human interaction, and thus safety was considered as a top priority. The design includes the selection of a frame with propellers guards to protect operators and equipment in case of accidental contact. Additionally, the system was planned to be designed to hover and land automatically if communication is lost, minimizing the risk of uncontrolled flight.
- **Energy Efficiency**: Given the drone's power limitations, the flight time for the drone is limited considering all component functioning. Therefore, energy efficiency was decided to be a critical design factor. The design is optimized for the low power consumption during both autonomous and manual modes through the flight controller powering all the motors so unwanted power consumption from the microcontroller does not occur. A commercially viable USB battery charger is also proposed with the following project to ensure rechargeability for multiple uses.
- **Reliability**: The drone must consistently follow the predefined path and accurately hover at designated locations while maintaining communication with the operator. The design involved the utilization of dependable sensors, which were individually tested and optimized, to implement all the features required by the drone.
- **Operational Simplicity**: The drone was designed to be user-friendly, with an intuitive UI that allows operators to monitor and manage the flight in real-time. The system involves manual override capabilities which provides the user control over the drone to move top, left, right, back, forwards, upwards and a separate E-stop which was planned to ensure that the safe landing of the drone in a battery deficient moment or software malfunction situation.
- **Stabilization and Maneuverability:** With confined spacing available in an indoor environment, it was deemed essential for the design to be stable and easily maneuver. A high

efficient motor was selection during component selection with the flight controller having inbuilt ESC functioning for stable hovering and reliable line following which was an added benefit to ensure a light weight overall design less than 250g which was stated to be the maximum limit for an indoor drone during the preliminary design review.

4.1 Evolution of Design

The design was carried out over a period of 12 weeks from initiation to the closure stage. The project schedule prepared for the life cycle of the project is represented below [5]:



Task Name	Duratio	Start	Finish	Pre	Resource Nam
Group Formation	1 day	Mon 22/7/24	Mon 22/7/24		
Project Selection	2 days	Tue 23/7/24	Wed 24/7/24	1	
Group SWOT Analysis and Roles Allocation	3 days	Tue 23/7/24	Thu 25/7/24	1	
Roles Analysis and Finalization	2 days	Tue 23/7/24	Wed 24/7/24		Group
SWOT Analysis	1 day	Tue 23/7/24	Tue 23/7/24		Group
Group PPIR	0 days	Thu 25/7/24	Thu 25/7/24		Group
File Management	88 days	Mon 22/7/24	Thu 17/10/24		Saranraj
Risk Assessment & Risk Register Maintaining	78 days	Fri 26/7/24	Fri 11/10/24	2FS+1 day	Ashintha
TQ Submissions (Weekly)	71 days	Fri 26/7/24	Fri 4/10/24	2FS+1	Dhrumil & Gc
Tasks Allocation for Research	12 days	Thu 25/7/24	Mon 5/8/24		
Drone frames	12 days	Thu 25/7/24	Mon 5/8/24		Ashintha
Line following robot programs (Similar projects)	12 days	Thu 25/7/24	Mon 5/8/24		Group
Aerodynamics for drone flight (Motor and properller selection)	12 days	Thu 25/7/24	Mon 5/8/24		Jothis
Preliminary Power Calculation (Battery Selection)	12 days	Thu 25/7/24	Mon 5/8/24		Ashintha
Camera Selection and Image Processing	12 days	Thu 25/7/24	Mon 5/8/24		Kasuni
Flight controllers (Autonomous flight)	12 days	Thu 25/7/24	Mon 5/8/24		Gopi
User interface for remote control	12 days	Thu 25/7/24	Mon 5/8/24		Dhrumil
Individual PPIR Submissions	0 days	Thu 1/8/24	Thu 1/8/24		Group
Safety Analysis Preparation	8 days	Thu 1/8/24	Thu 8/8/24		Group
Safety Analysis Submission	0 days	Fri 9/8/24	Fri 9/8/24		Ashintha
Lab Induction	5 days	Mon 5/8/24	Fri 9/8/24		Group
Lab Induction Completion	0 days	Fri 9/8/24	Fri 9/8/24		Group
Requirement Analysis Preparation	15 days	Thu 1/8/24	Thu 15/8/24		Group
Requirements Analysis Report Submission	0 days	Fri 16/8/24	Fri 16/8/24		Saranraj
Conceptual Design and Preliminiary Design Start	19 days	Sat 10/8/24	Wed 28/8/24	21	Group
Preliminary Design Review	0 days	Thu 29/8/24	Thu 29/8/24		Group
First Professional File Management Review	0 days	Thu 29/8/24	Thu 29/8/24		Saranraj
Drone Design Finalization	44 days	Thu 29/8/24	Fri 11/10/24	26	
Critical Design Review	0 days	Thu 19/9/24	Thu 19/9/24		Group
Design Testing and Verification	21 days	Thu 19/9/24	Wed 9/10/24	28SS+	Group
Design Verification and Validation Review	0 days	Thu 10/10/24	Thu 10/10/24		Group
Preparation for Final Report	29 days	Thu 19/9/24	Thu 17/10/24	29	Group
Final Interview	0 days	Thu 17/10/24	Thu 17/10/24		Group
Final Professional File Management Review	0 days	Thu 17/10/24	Thu 17/10/24		
Final Report Submission	4 days	Fri 18/10/24	Mon 21/10/24	32	Ashintha

Figure 4 - Gantt chart for the Indoor Line Following Drone Project - Team 12

The weekly project evolution and progress over the scheduled timeline is represented below:

Week 1 – An overview of the project requirements was discussed with team members along with the Group PPIR and SWOT analysis for role allocation related to project.

Week 2 - The Gantt chart preparation and the literature review division were carried out to separate a hardware and software team separately. The preliminary design overview block diagram was design as follows:

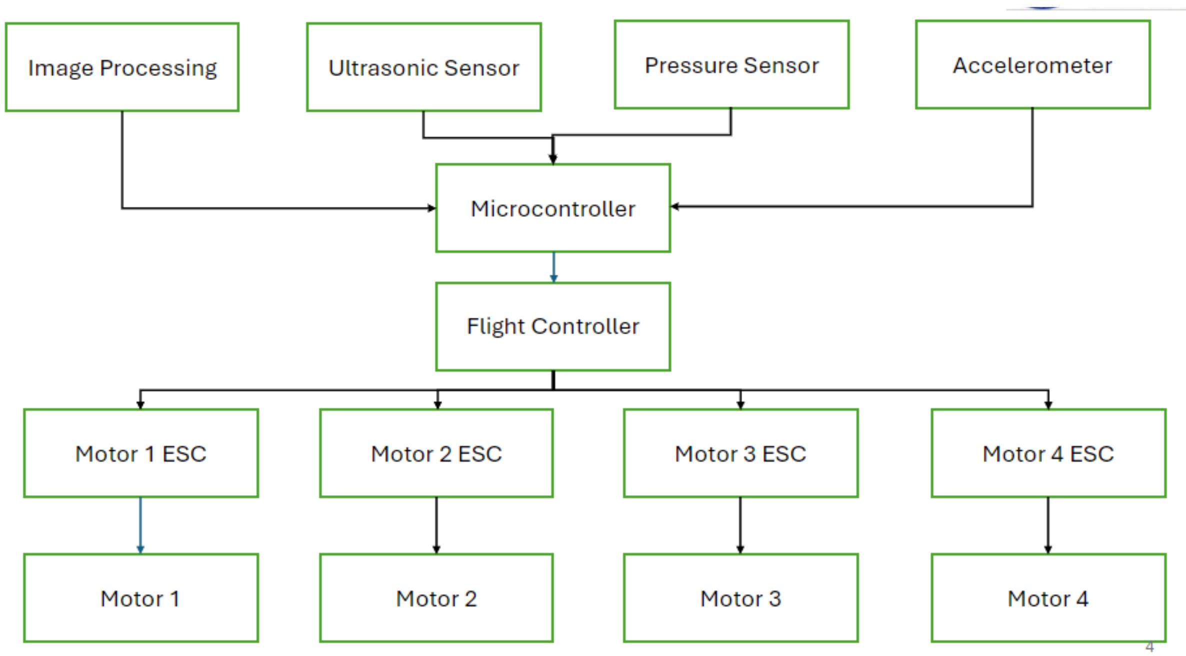


Figure 5 - Preliminary Block diagram created for the design overview

Week 3 – The safety analysis report submission was carried out by the team after completing the lab induction to work at the Lycopodium laboratory. Initial thrust calculations for a 250g drone was carried out for motor selection. Drone frame availability were discussed and left for consideration before finalization

Week 4 - The team proceeded to review more literature on their designated areas. An extensive requirement analysis for the drone project was carried out to select and prioritize 20 critical requirements along with 9 additionally identified requirements. The team finalized the flight controller and preliminary algorithm for path detection was done.

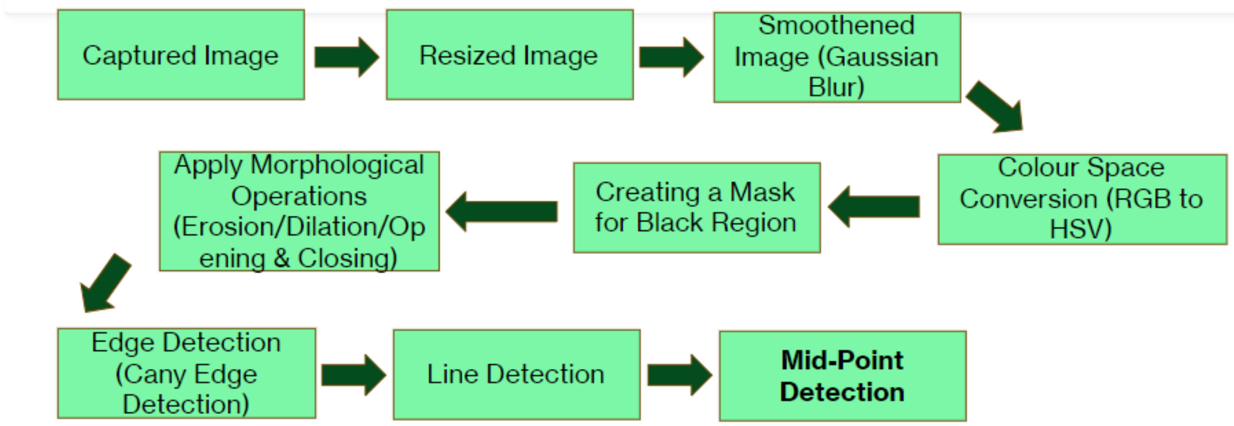


Figure 6 - Preliminary line detection architecture

Week 5 & 6 – The design proceeded with component finalization in-terms of microcontroller, sensors, motors and propellers. Preliminary design review was carried out with insights from the supervisor to implement E-stop button and ensure the reliability of the sensors.

Week 7 – The design was progressed with preparation of a preliminary prototype as represented below after component purchasing. Team proceeded with the use of Beta Flight for flight controller configuration and further improvements for path detection including addition of ranging hue saturations.

Figure 7 - Initial Prototype built for drone testing

Week 8 – Critical design review was carried out. Concerns of prototype drone frame size were discussed to finalize a new selection. Energy calculations were carried out for battery selection and battery charger selection with program developments for the sensors. Image processing codes were developed with the implementation of a 3x3 matrix for flight action identification during path following

Week 9, 10 & 11 – Carrying out modular component testing to ensure individual operation of components. Overall system integration attempted by the team. Path detection tests, image processing tests, obstacle detection tests, altitude measurement tests, drone speed tests carried out and drone hovering tests attempted.

Week 12 – The team proceeded with drone assembling and integration. Further design integration attempted. Final report was prepared for submission by the team.

5. Design Elements

The following section discusses the design elements used and selected for the indoor drone project along with justification for their selection. Proper component selection was carried to adhere with the afore-mentioned design philosophy is Chapter 3 and meet the determine project requirements in a lengthy procedure to ensure that the most suitable components were selected based on energy and budgetary restriction. All electronic devices and their related specific technical parameters such as current, voltage, power, weight and physical size were considered during selection. As per the design architecture, the design elements for component selection includes microcontroller, flight controller, motors, propellers, drone frame, battery, camera, ultrasonic sensor, accelerometer and barometric sensor selection and a critical review of each selection is discussed below.

5.1 Microcontroller Selection

The microcontroller acts as the primary communication entity used for programming the modular components and integration as a whole. The microcontroller comparisons were carried under 6 different choices to ensure the selection of the best suited component. The analysis is represented below [6] [7] [8] [9]:

Table 2 - Microcontroller selection analysis

Name	**Size (mm)**	**Voltage, Current**	**Weight (g)**	**Pins**	**RAM**	**Processor**	**SD card**	**Open CV**	**Price**
Raspberry pi zero W	65x30x0.2	5V DC, 2.5A	9	40	512 MB	Single Core 1GHz	No	Yes	49.95
Raspberry pi 4	85x56x19.5	5V DC, 3A	46	40	2 GB	Quad Core 4 1.5GHz	Yes	Yes	109.19
ESP32-S3	25.5x18x3.1	3.3V, 0.5A	10	34	512 kB	Dual Core 2.4Gz	No	No	42.95

NVIDIA Jetson Nano	69.6x45x45	4.75V, 2.5A	241	40	4 GB	Quad Core 1.43GHz	Yes	Yes	197.99

From immediate observation, NVIDIA Jetson Nano is eliminated due to the project's budgetary and weight limitations, despite it being the most commonly used microcontroller in designing drones [9]. Raspberry pi 4 module was also removed from consideration due to its subjectively higher cost and weight compared to the ESP32 and Raspberry pi zero W modules. Comparison of the remaining 2 microcontrollers with its relation to achieving the requirements are as follows where positive features are highlighted in green, acceptable in orange and inadequate features in red [10]:

Table 3 - Comparison of ESP32 vs. Raspberry Pi MC

Description	ESP32-S3	Raspberry Pi Zero W
Lightweight	Yes	Yes
Small Dimensions	Good.	Acceptable but bigger than ESP32
Low Cost	Similar and acceptable	Similar and acceptable
Sufficient I/O for sensors	Yes	Yes
Good processing speed	Good.	Acceptable but slower than ESP32
RAM	Poor. Concerns with memory for image processing	Good. Suitable for image processing
Supports Open CV	Some features not allowed. Python coding is not available in Open CV (heavy reliance on advanced C programming)	Allows Python coding including required features in Open CV

Therefore, after consideration, the Raspberry Pi Zero W was selected as the microcontroller for the following indoor line following drone project:

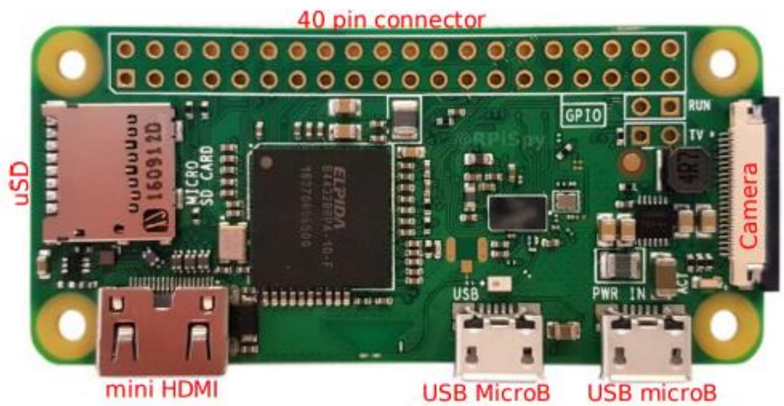
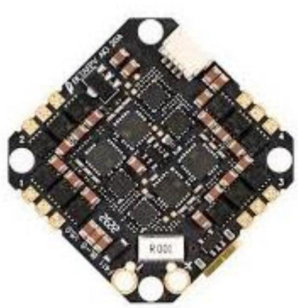
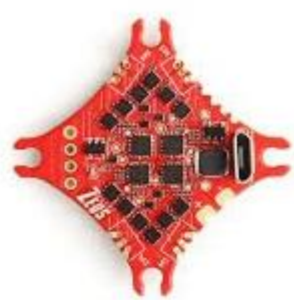


Figure 8 - Selected Raspberry Pi Zero W MC

5.2 Flight Controller Selection

As per flight controller analysis carried out during the literature review of the requirements analysis, ARM based platforms for open CV flight controllers were selected from which the following options were weighed for the flight controller selection [11] [12] [13]:

Table 4 - FC Selection Analysis

Description	F405 4S 20A Toothpick Brushless FC V5	Zeus5 AIO 1-2S F411	Speedy Bee F405 AIO
Image			
Size	25.5mm x 25.5mm	25.5mm x 25.5mm	25.5mm x 25.5mm
Weight	5.2g	4.6g	7.5g

Price	$114.95	$ 72.95	$155.95
Battery Requirement	Supports up to 4S LiPo (14.8V)	Supports 1S or 2S LiPo (4.2V - 7.4V)	Supports up to 6S LiPo (25.2V)
Inbuilt ESC	20A 4-in-1 ESC	12A 4-in-1 ESC	35A 4-in-1 ESC
Processing power	STM32F405 microcontroller (168 MHz)	TM32F411 microcontroller (100 MHz)	STM32F405 microcontroller (168 MHz)
Flash Memory	16MB onboard	16MB onboard	16MB onboard
BEC	5V/1.5A BEC	5V/2A BEC 10V/1A BEC	5V/2A BEC
In – Built Wi-Fi	No	Yes	Yes
Beta Flight Configurator	Supported	Supported	Supported
Gyroscope	MPU6000	MPU6000	MPU6000
UART Ports	4	2	5

Therefore, after careful reviewing and analysis, the HGLRC Zeus5 AIO flight controller was selected due to its lightweight, ESC integrated design and all in-one features which made it ideal for indoor drone applications. The FC weight of 4.4g significantly contributed towards reducing the overall weight of the drone to maintain the safety threshold weight. The selected flight controller was supported by 1S and 2S LiPo batteries with built in Battery elimination circuit to provide constant 5V with 2A current which matched with the ratings of other components used in the system and battery limitations specified by the project sponsors. The FC has the capability of powering the microcontroller, sensors and the camera with these rated values.

The integrated Electronic Speed Controllers (ESC) provides efficient power management for the brushless motors, thereby reducing the weight and related costs in adding a separate component for the ESC. This feature also ensures smooth integration of motors and responsive control during the flight. The BL_S firmware of the FC enhances compatibility with a range of motors allowing for precise and stable control which was crucial for the line-following and hovering requirements. The integrated motor plugs on the board ensures the motor connections are smoother and makes the maintenance easier.

The Zeus5 AIO has built-in Wi-Fi which enables wireless communication for firmware updates, data logging from the microcontroller and real-time communication with the user interface for monitoring effectively. This functionality simplifies drone maintenance, configuration and remote adjustments without needing a physical connection. The compact design also has built in accelerometer and gyroscope for positioning the drone. These sensors shall facilitate safe landings and other navigation tasks using Beta Flight configurator.

Additionally, The Zeus AIO controller is Beta Flight configurator compatible which makes the tuning and calibrations of the system parameters effective with scope to monitor and test the individual components on the flight effectively and is an open CV supported software.

5.3 Motor Selection

The maximum weight allocated for the drone initially was 250g for worst-case scenario calculations. Therefore, thrust calculations performed based on this maximum weight and are depicted below:

$$Thrust\ Required = Maximum\ Weight\ \times 9.8$$

$$Thrust\ Required = 0.25 \times 9.8$$

$$Thrust\ Required = 2.45\ N$$

$$Thrust\ Required\ by\ each\ motor = \frac{2.45}{4} = 0.6125\ N$$

$$Motor\ Peak\ Thrust = 1.225\ N$$

However, in-reality the weight for the drone based on selected components was well below 250g at approximately 100g which was used to re-evaluate the thrust calculations to produce the following result:

$$Thrust\ Required = Maximum\ Weight\ \times 9.8$$

$$Thrust\ Required = 0.1 \times 9.8$$

$$Thrust\ Required = 0.98\ N$$

$$Thrust\ Required\ by\ each\ motor = \frac{0.98}{4} = 0.245\ N$$

Based on compatibility with the selected Zeus5 AIO flight controller, a number of motors were considered and compared. The three main motors that were compared included the Happymodel EX0802 19000KV motor, the BetaFPV 1404 4500KV and BetaFPV 1506 3000KV brushless motors [14] [15] [16].

Table 5 - Motor Selection Analysis

Specification	**Happymodel EX0802 (19000KV)**	**BetaFPV 1404 (4500KV)**	**BetaFPV 1506 (3000KV)**
Weight	1.7g	8.49g	14g
Shaft Diameter	1.0mm	2.0mm	1.5mm
Shaft Length	2mm	3mm	2mm
KV (rpm/V)	19000KV	4500KV	3000KV
Input voltage	2S	3-4S	3-6S
Cable Length	27mm	40mm	40mm
Thrust Potential	High	Moderate	Low
Price	AUD 13.20	AUD 20.81	AUD 25.28

The Happymodel EX0802 brushless motors were selected based on their exceptional performance characteristics and suitability for lightweight applications. They were also selected based on their compatibility with the selected Zeus5 AIO flight controller. The motors itself weigh only 1.7g thus adding insignificant weight to the drone compared to the other two options which weigh 8.48g and 14g per motor.

Figure 9 - Selected Happymodel motor

The 19000kV enables quick response times and agility due to a higher RPM per volt and thus has higher thrust potential at lower heights. The other BetaFPV motors have lower KV ratings which result in slower acceleration and reduced performance. These factors resulted in the selection of the Happymodel EX0802 motors over the other two motors.

The Happymodel motors have impressive efficiency ratings, with thrust-to-power ratios exceeding 2g/W under various operating conditions. The motor is capable of demonstrating a thrust of 0.28714 N with a current draw of 4.09A. Thus, it is able to achieve the necessary thrust required to achieve lift and maneuverability.

5.4 Propellers Selection

The selection of the propellers is crucial for the performance and efficiency of the indoor line-following drone. The Betafpv × Gemfan 40mm 4-Blade Propellers were selected based on their great efficiency while maintaining high thrust output and their compatibility with the selected motors [17]. The propellers' unique blade design minimizes drag and thereby, improve lift.

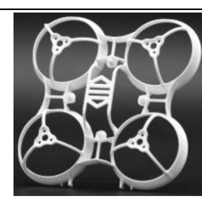
Figure 10 - Selected Propeller

With an inner diameter of 1.0 mm, these propellers are perfectly suited for the Happymodel EX0802 motors, ensuring a secure fit and optimal performance. The propellers' light weight of approximately 2.4g per propeller, contributes to the overall low weight requirement of the drone ensuring a lower burden for the motors. This was deemed essential for maintaining a favorable thrust-to-weight ratio. The selected propellers' build material ensures that it is able to withstand minor impacts and maintain performance over time. Therefore, considering these facts, the Betafpv × Gemfan 40mm 4-Blade Propellers were finalized as the propeller for the drone design.

5.5 Drone Frame Selection

As per the project specifications, the teams were allowed to purchase the drone frames from vendors or consider the option of construction our own. In reviewing 3D-printing possibilities for drone frames, due to its additional expenses for material cost, printing costs and careful requirement for a detailed design (including proper sized holes for assembly) it was decided to eliminate the risk entirely by purchasing pre-design frames and assemble it. The following drone frames were taken into consideration [18] [19] [20] [21]:

Table 6 - Drone Frame Analysis

Description	**Rekkon 3 Nano Long**	**DIATONE Taycan C25 MK2**	**NewBee Drone Cockroach V3**	**Pavo20 Brushless Whoop**
Image				
Dimensions	130mm x 125mm	100mm x 95mm	83mm x 83mm	125.7mm x 125.7mm
Wheelbase	140mm	111.5mm	65mm	90mm
Weight	23g	55g	3.95g	26.7g
FC Hole	25.5mm x 25.5mm	25.5mm x 25.5mm	25.5mm x 25.5mm	26mm x 26mm

Propeller size	3 inches	2.5 inches	1.6 inches	2 inches
Battery	18650-3000mAh	4S (650-850mAh)	1-2S Battery	2S to 4S (450-650 mAh)
Safety Guards	No	Yes	Yes	Yes
Price	AUD 33.87	AUD 49.08	AUD 8.25	AUD 20.89

From battery requirements and weight considerations, Rekkon 3 Nano and DIATONE Taycan drone frames were eliminated as the project requirements were not met. Initially, the NewBee Drone Cockroach V3 was purchased for prototyping purposes as the purchased motors and FC were compatible. However, considering the size of the ultrasonic sensor and microcontroller, during assembling it was found to block the openings which heavily impacted the propeller functions as represented below:

Figure 11 - MC comparison with prototype drone frame

Therefore, the team pivoted and finalized the selection of the Pavo20 frame for the finalized drone design. This selected frame for the indoor line-following drone has a balance of durability, weight and compactness, which is essential for indoor drones due to weight restrictions of 250g for safety reasons which is depicted in the figure below:

Figure 12 - Newly selected drone frame

The Pavo20 frame is constructed from crash-proof PA12 material, which is known for its high strength. This makes it a highly durable frame and the weight of 26.7 grams makes it perfect for the indoor application. The battery strap slot provides a secure fit for various battery sizes from 2S to 4S batteries making it future proof for upgrades, although the drone will primarily operate on 2S batteries, in accordance with project specifications.

Its compact 90mm wheelbase is optimally designed to accommodate all necessary components required for the project such as the sensors, camera, battery, microcontroller, flight controller, while maintaining a lightweight and efficient design [21].

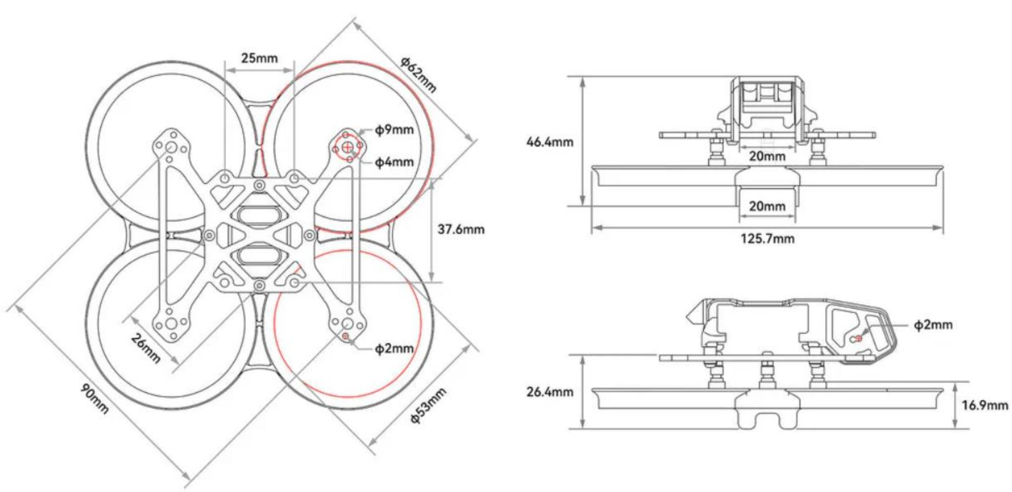


Figure 13 - Dimensions of Pavo20 drone frame

The 26×26 mm FC mount works well to fit the selected Zeus5 AIO flight controller. It also provides a firm and stable base for the core electronics while minimizing vibrations during flight which is critical for sensor accuracy. The sensors can be strategically placed around the drone for obstacle detection and altitude maintenance. Its crash proof propeller guards are integral to ensure safe operation in indoor environments, where the risk of collisions with personnel and other elements is significantly higher which makes the selected Pavo20 frame suitable for the designed drone.

5.6 Battery and Battery Charger Selection

The battery selection was carried out initially considering the maximum rated power consumptions of the components. The power ratings of each component utilized in the project which consumes power is represented below [22] [23] [24] [25] [26] [6] [27]:

Table 7 - Power Calculations for all components

Component	**Total**	**Voltage**		**Current**	**Max Power**	**5% Contingency**	**Total**
		Min	**Max**				
Zeus5_AIO FC	1	3.3V	4.3V	5A	21.5W	1.075W	22.5750W
Happymodel EX0802 Brushless motors	4	-	1.5V	0.3A	0.45W	0.0225W	1.8900W
Ultrasonic - HCSR04	2	-	5V	0.015A	0.075W	0.00375W	0.1575W
Pressure - BMP180	2	1.62V	3.6V	0.001A	0.0036W	0.00018W	0.0076W
Accelerometer - LSM3003	1	2.5V	3.3V	0.001A	0.0033W	0.00017W	0.0069W
Raspberry Pi zero W	1	3.3V	5V	2.5A	12.5W	0.625W	26.2500W
5MP Camera	1	2.7V	3.3V	0.25A	0.825W	0.04125W	1.7325W

For the energy calculations the microcontroller power was only assumed, ignoring the FC power, as either the microcontroller or the flight control will consume energy to power the motors and sensors and not both. Therefore, considering a hovering time of 30 seconds and energy consumption per minute of flight the following values were obtained using:

$$Energy\ for\ one\ minute\ flight = Max\ Power \times \frac{1}{60}\ Wh$$

$$Energy\ for\ 30\ second\ hovering = Max\ Power \times \frac{0.5}{60}\ Wh$$

Table 8 - Energy Calculations for the Final Drone

Component	Total	Max Power	Energy for one-minute flight (Wh)	Energy required for hovering (Wh)
Zeus5_AIO FC	1	22.5750W	N/A	N/A
Happymodel EX0802 Brushless motors	4	1.8900W	0.032	0.016
Ultrasonic - HCSR04	2	0.1575W	0.003	0.001
Pressure - BMP180	2	0.0076W	0.0001260	0.0000630
Accelerometer - LSM3003	1	0.0069W	0.000	0.000
Raspberry Pi zero W	1	26.2500W	0.438	0.219
5MP Camera Module	1	1.7325W	0.029	0.014
TOTAL			0.501 Wh	0.250 Wh

Therefore, maximum allowable battery capacity for the following project is 2S Battery with 800mAh capacity.

$$Total\ allowed\ energy = 2 \times 3.7 \times 0.8 = 5.92Wh$$

Therefore, theoretical possible flight time (without hovering time) is:

$$Max. Flight\ time = \frac{5.92 - 0.250}{0.501} = 11\ minutes\ 19\ seconds$$

With these considerations, the following batteries were evaluated [28] [29]:

Table 9 - Battery Selection Analysis

Description	**BlackZon BZ540037 800mah 7.4v Li-ion Slyder Battery Pack**	**CNHL MiniStar 650mAh 7.4V 2S 70C Lipo Battery**
Image		
Weight	63g	49g
Voltage	2S (7.4V)	2S (7.4V)
Capacity	800mAh	650mAh
Depth of Discharge	80%	70%
Max Power	4.736 Wh	3.367
Price	AUD 32.99	AUD 11.97
Possible Max. Flight Time	8 minutes 57 seconds	6 minutes 13 seconds

Therefore, considering the value for flight time, the CNHL Ministar 2S 650mAh battery was selected for the project.

Due to risks associated with LiPo batteries, typically methods of battery charging is not applicable. As the project requires the use of a commercially available USB battery charger, the following 2 were considered for the chosen battery [30] [31] [32]

Table 10 - Battery Charger Selection Analysis

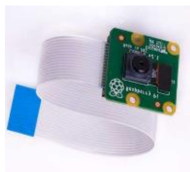
Charger Model	Input voltage	Output voltage	Battery compatibility	Charging current	Main features
B3	240V AC	4.2V DC	2S,3S LiPo	Up to 1A/cell	Compact design and compatible with 2S,3S both
iMax B6AC	100-240V AC	10-20V DC	1-6S LiPo, NiMH	Up to 6A	Lithium battery balancer
2S FMS USB LiPo	USB – 5V	7.4V DC	2S LiPo	Up to 2A	Quick charging and compatible with battery that we selected for our drone

Therefore, for the project the HGLRC Thor 1-2S Charger 4-way 4.35v charging board charger has been selected. The charger is capable of port connections of 4 types including USB

5.7 Camera Module Selection

Selecting an appropriate camera module is essential for the project since it is based on the following line using image processing. According to the project requirements and given conditions, the camera module needs to be lightweight, be compatible with Raspberry Pi, be fit with a budget, and be able to provide clear vision in different lighting conditions with a good field of view.

Table 11 - Camera Selection Analysis

Description	**Raspberry Pi Camera Module V2**	**Arducam Fisheye Lens Camera**	**Raspberry Pi High Quality Camera**	**E-Con Systems e-CAM55**	**Waveshare Fisheye Camera (B)**
Image					
Sensor Type	8 MP Sony IMX219	5 MP OV5647	12.3 MP Sony IMX477	5 MP CMOS	5MP OV5647
Field of View (FOE)	62.2° diagonal	180° diagonal (Fisheye)	Adjustable (Lens Dependent)	70° horizontal	160° diagonal (Fisheye)
Resolution	3280 x 2464	2592 x 1944	4056 x 3040	2592 x 1944	1080p
Lens	Fixed Focus	Fixed Fisheye	C/CS-Mount Lense	Fixed Focus	Fisheye
Compatibility	Raspberry Pi (all modules)	Raspberry Pi (all modules)	Raspberry Pi (with HQ camers)	Raspberry Pi, Jetson	Raspberry Pi (all modules)
Frame Rate	30 fps	30 fps	60 fps	30 fps	30 fps
Weight	3.4g	4g	25g	6g	4g
Dimensions	25 mm x 23 mm	25 mm x 24 mm	38 mm x 38 mm	25 mm x 24 mm	25 mm x 24 mm
Price	AUD 35.00	AUD 29.00	AUD 85.00	AUD 59.00	AUD 28.0

After a comprehensive survey of market available modules, “Arducam Fisheye Lens Camera” was chosen due to its performance in several areas among others. Regarding FOV, fish-eye cameras provide a wide range, but there might be image distortion, especially at the edges, which can cause severe issues in image processing. On the other hand, Raspberry Pi Camera Module V2 provides clear images suitable for line detection. The selected module will be a feasible option Compared to other key project limitations, such as weight and budget. Furthermore, this module is specially designed for the Raspberry Pi system, which brings simplicity than others.

5.8 Accelerometer selection

The table highlights the accelerometers considered for selection [33] [34] [35]

Table 12 - Accelerometer Selection Analysis

Sensor Name	**Price (AUD)**	**Weight**	**Size**	**Measurement Range**	**Voltage**
LSM303DLHC	17.25	0.6 g	13x20x3 mm	±2g, ±4g, ±8g	2.5~3.3 volt
ADXL335	16.95	1.27g	4X4X1.45 mm	±3g	1.8~3.6 volt
MPU-6050	22.95	10 grams	20X16 mm	±2g, ±4g, ±8g, ±16	3~5 volt

Detailed Analysis

ADXL335:

It is a small, low-power accelerometer with analog voltage outputs proportional to acceleration. This makes it perfect for tilt-sensing applications. Measurement of both static and dynamic accelerations is possible. The bandwidth selection can also be done according to the requirement of the application; therefore, this makes them suitable for several low-power applications like mobile devices and gaming systems.

LSM303DLH:

This sensor comes with a 3 dimensions accelerometer. This adds flexibility in applications that may require both kinds of measurements. Full-scale ranges are user-selectable, and the I2C interface provides easy integration into systems. It has an internal self-test capability, which is quite useful in ensuring functionality at final application. The capability for generating interrupts based on events of motion adds to the utility value in motion detection applications.

MPU-6000:

The MPU-6000 is characterized by having a gyro and accelerometer on the same chip in order to minimize cross-axis misalignment effects that would otherwise result from different solutions. It integrates a Digital Motion Processor that can run sophisticated algorithms, and hence targets advanced motion processing. Its small size and great integration make it fit for the space-constrained designs.

Through analysis, the LSM303DLH has a particular advantage over the other two sensors, as this single package offers dual functionality-an accelerometer and a magnetometer for complete motion sensing. Its user-selectable full-scale ranges come with flexibility tailored to the particular needs of the specific application, while the self-test feature ensures dependability in operation. Furthermore, this ability to cause interrupts improves responsiveness in real-time applications, so this could be one of the best selections for modern embedded or drone devices where acceleration and magnetic field data are highly required.

5.9 Ultrasonic Sensor Selection

There were multiple sensor models which were enquired for the project. The following are the models that were cross verified before selecting the ultrasonic sensor [36].

Table 13 - Ultrasonic Sensor selection analysis

Model name	**Operating Voltage**	**Range**	**Interface**	**Compatibility**
HC-SR04	5V	2cm to 400cm	4 pins	Yes
JSN-SR04T	5V	20cm to 600cm	4 pins	Need some

				precaution
Maxsonar MB1013	3.3 to 5V	30cm to 5 meters	UART	Yes

HC-SR04 ultrasonic sensor is one of the most widely used distance sensors due to its reliability and sustainability. It operates by emitting the waves at 40khz and measuring the time it takes for the sound waves to reflect. The sensor then calculates the distance using the speed of the sound. By the table we can see that the range is from 2 cm to 400cm that makes the accuracy of 3mm makes it suitable for short-medium range distance measurement.

JSN-SR04T is a waterproof version of the traditional sensor, where they are used in moisture or liquid ones. It works similarly to the HC-SR04, the only difference is the long range and waterproof transducer.

All these sensors are compatible with the raspberry Pico. The easiest way for communication is the 4-pin connection for the UART. The closer the object is, the easier it is for the drone to make the required changes. Thus, HC-SR04 is chosen as the required sensor for the project due to its short range and the financial budget for the sensor.

5.10 Barometric Sensor Selection

Multiple pressure sensors that are suitable for the raspberry pi zero were analyzed. They are listed in the table below [37]:

Table 14 - Barometric Sensor selection analysis

Model Number	Name	Interface	Voltage	Pressure range
BMP180	Barometric Pressure sensor	I2C/SPI	1.8-3.6V	300hPa – 1100hPa
BME280	Pressure, Humidity sensor	I2C/SPI	1.8-3.6V	300hPa – 1100hPa
MPL3115A2	Precision Altimeter	I2C	1.95V-3.6V	20kPa – 110kPa
MS5611	High resolution Baro sensor	I2C/SPI	1.8-3.6V	10hPa – 1200hPa
MPRLS	Ported Pressure Sensor	I2C	3.3V	0-25 PSI

BMP180 is a low-power, high- precision digital barometric pressure sensor designed specifically for application and wearable devices. It measures the atmospheric pressure and temperature, providing highly accurate readings.

It is lower power consumption which makes it ideal for the battery-operated applications. It is also compact and high in accuracy. The most important thing is it is easier for integration with the raspberry pi zero. Thus, BMP180 has been selected for the project due to its accuracy, ease of integration, low power consumption, and affordability.

6. System Integration

The system integration consists of the implementation and functioning of the design elements in relation to the design requirements and deliverables. Integration of each design element and errors and issues faced are discussed below.

6.1 Microcontroller Integration

6.1.1 Microcontroller communication with Flight Controller

The communication between the Raspberry pi zero and the flight controller is made in 2 primary ways, and they are UART Communication and Wi-Fi communication. Initially, the Raspberry pi zero must be installed with the Desktop environment and micro python which are crucial for operating the microcontroller. The terminal is used for installing the python and the other necessary components like pylearn for image processing. Initially, the UART communication is established by following the flowchart.

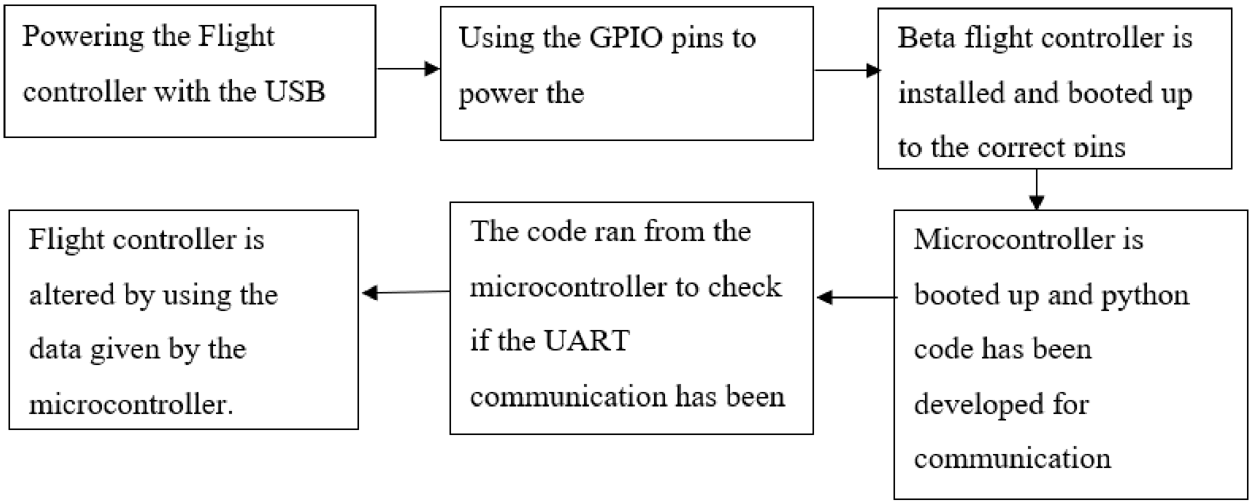


Figure 14 - Communication algorithm between MC and FC

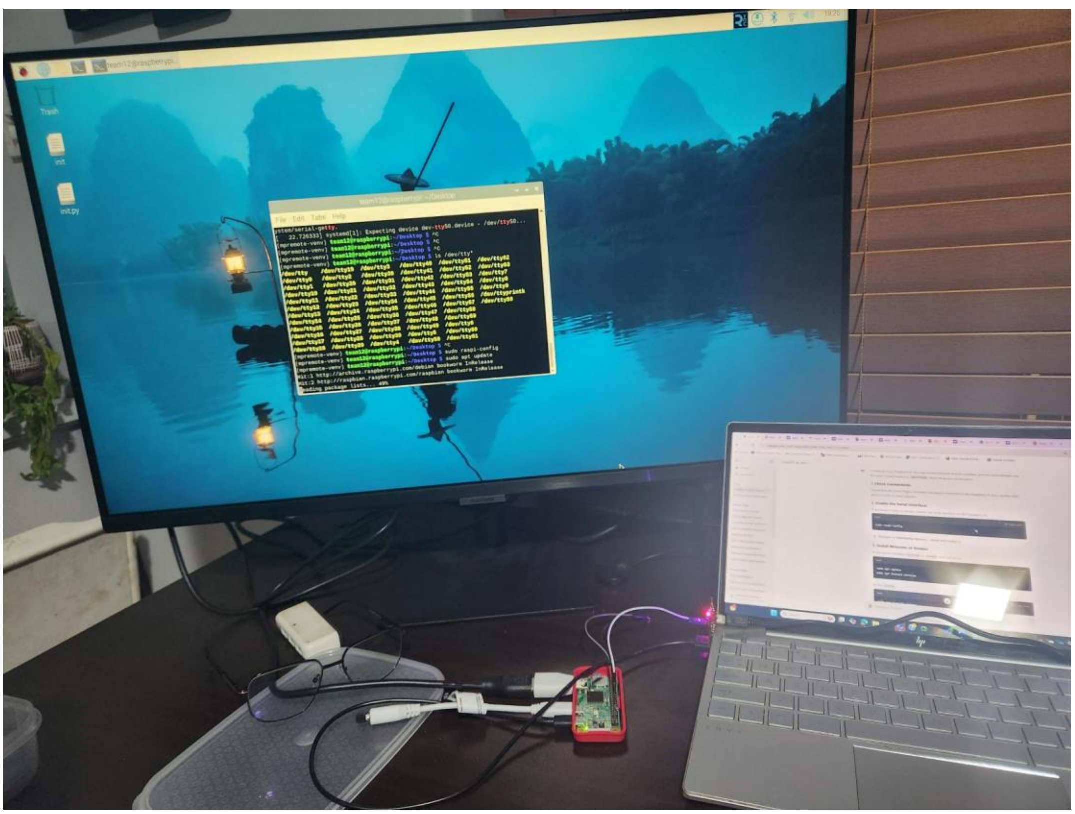


Figure 15 - Communication setup between FC and MC

The communication between the MC and FC explains the UART communication that has been made. The python code that has been used for the testing of communication has been updated in the Appendix A.

6.1.2 Communication from the Camera to the Microcontroller and Flight controller

The communication from the camera is crucial for the image processing since the drone needs to adjust itself towards the line. The camera model that was acquired by the team was Zero Cam. The Zero cam is a fish eyed model with a 5MP camera. Since the camera is an old version for the microcontroller, legacy model library was used to make the camera communicate with the microcontroller. The flowchart of how the communication has been made is given below:

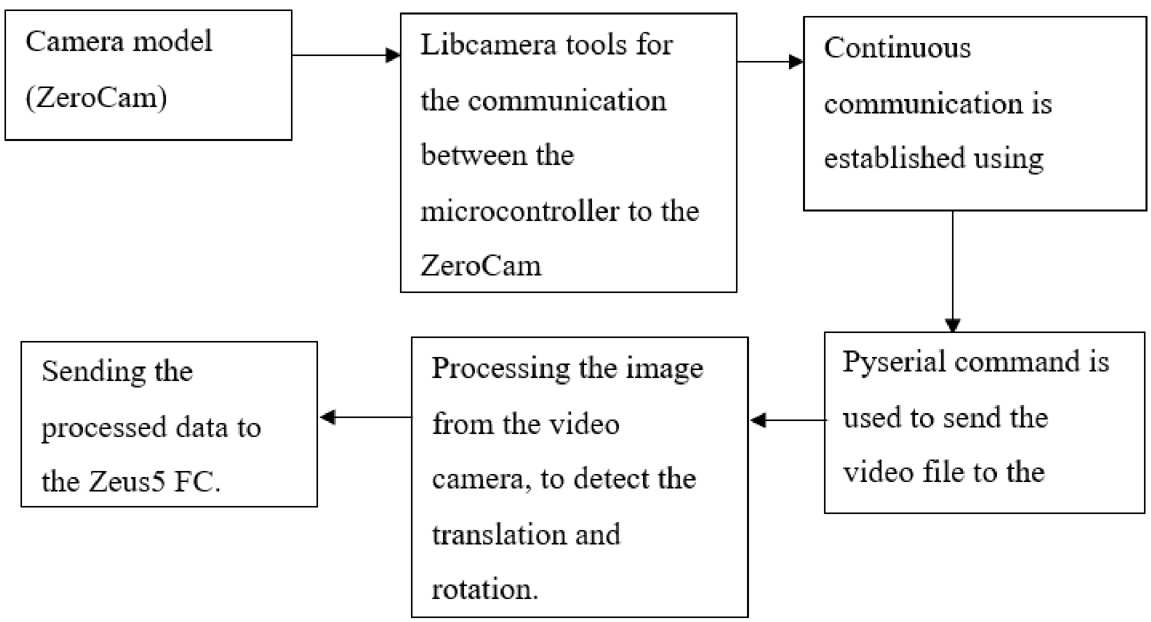


Figure 16 - Communication setup between Camera and FC and MC

The entire communication code between the Camera and the Flight controller is given in the Appendix B. The camera feed is fed into the Image processing python file where the images is converted to the commands like x, y, z co-ordinates where it represents the direction of the drone that needs to be changed. The direction is then fed into the microcontroller where the python file changes the direction to the command using the Mavlink command which is the primary command for communicating between the drone and the Flight controller.

The communication is then established by changing the pitch angle and yaw, that changes the altitude and the speed of the drone. The control is then transferred to the flight controller where it changes the speed of the drone to control the pitch and the yaw.

6.1.3 Issues detected in the microcontroller:

The problems that were discovered when integrating the microcontroller are as follows:

- <u>Noises in the communication</u>:

The communication was so garbled and filled with noises which was so hard to decode. The noises which initially came is given below:

Figure 17 - Garbled Images

The garbled communication was then repaired by soldering the wires between the Flight controller and the Microcontroller.

- <u>The communication going over the limits.</u>

The Beta Flight controller being the main communication center, the pitch angle initially shown was 5 degrees but the communication that was retrieved from the flight controller was 16240 raw units which is then converted to degrees which is totally 25 degrees.

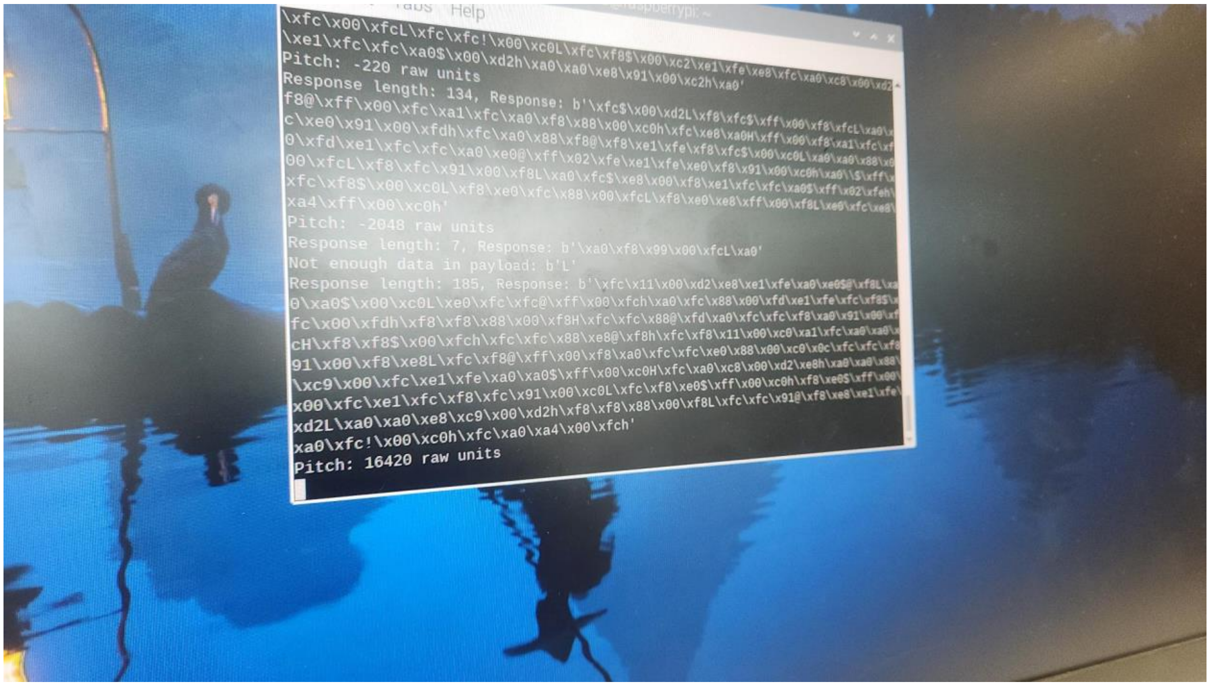


Figure 18 - Noises in Communication

The average mean and the filters were used to change the noises and able to filter the noises and change the garbled noises inside the communication.

- <u>Kernal destroyed</u>

While the Libcamera is used for the camera, initially the camera wasn't supporting the microcontroller. The legacy model was installed using the terminal to install the camera model. While the tools were trying to install, the microcontroller was destroyed due to the terminal error and the desktop environment was destroyed due to the legacy stack which couldn't get installed. The error is given in the figure below.

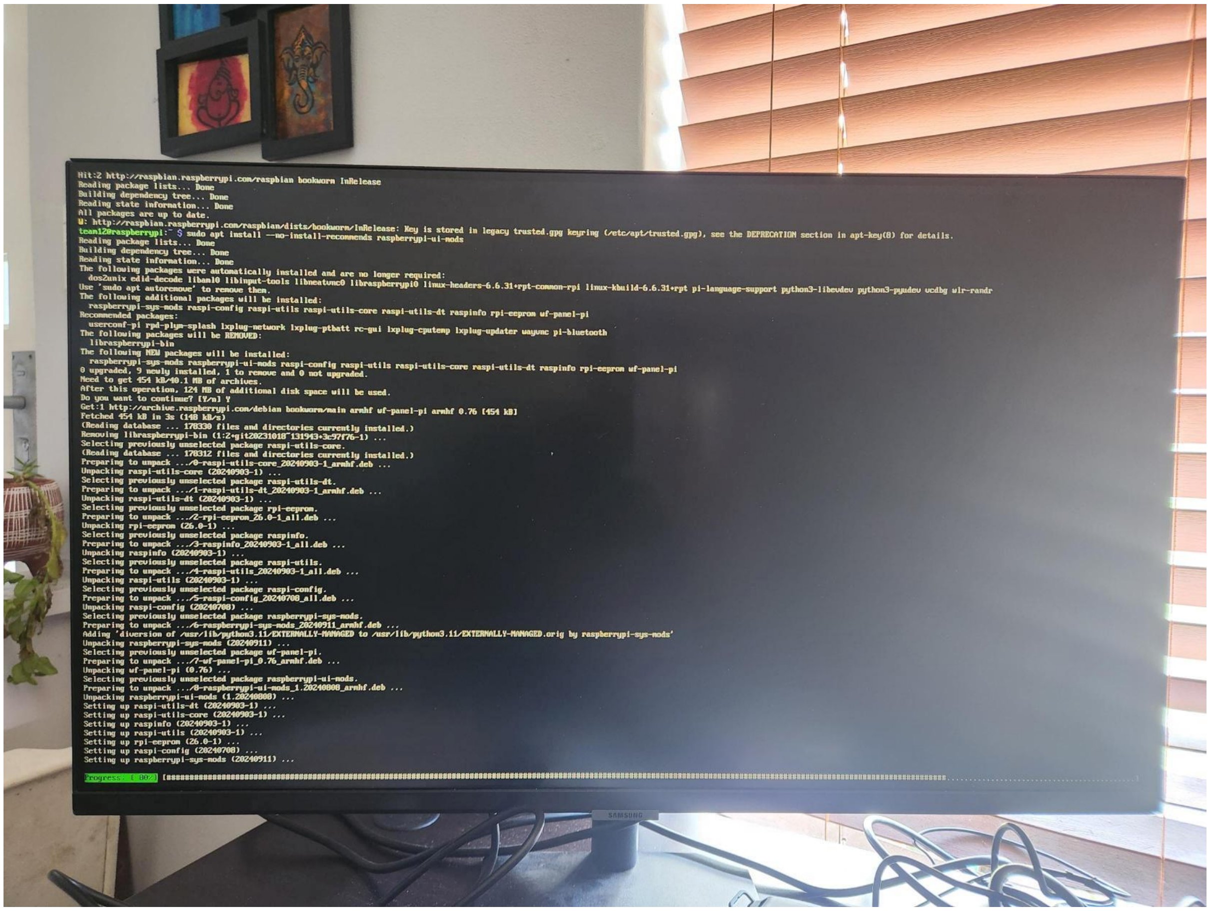


Figure 19: Kernal error.

The debugging was then carried out by installing the new desktop environment using the terminal which solved the kernel error and the Libcamera installation.

6.2 Flight Controller Setup

The initial setup of the drone includes calibrating the sensors and configuring flight parameters using the Beta flight configurator. The drone was connected to the computer via the micro-USB port on the flight controller and the software represented below was opened and following calibration procedures for accelerometer and motor controls were carried out.

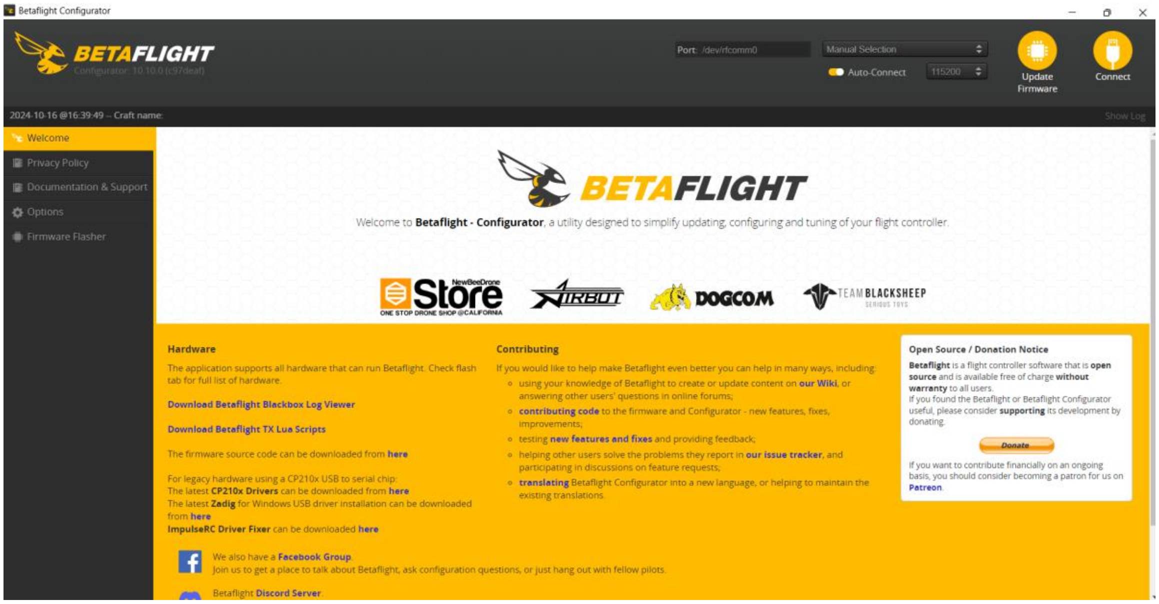


Figure 20 - Beta Flight setup

Initially, the FC was connected to the configurator. The figure below shows the home screen after connecting the FC. Note: It should be ensured that the drone is placed on a flat surface. The accelerometer calibration was pressed and waited for the calibration to be completed as follows.

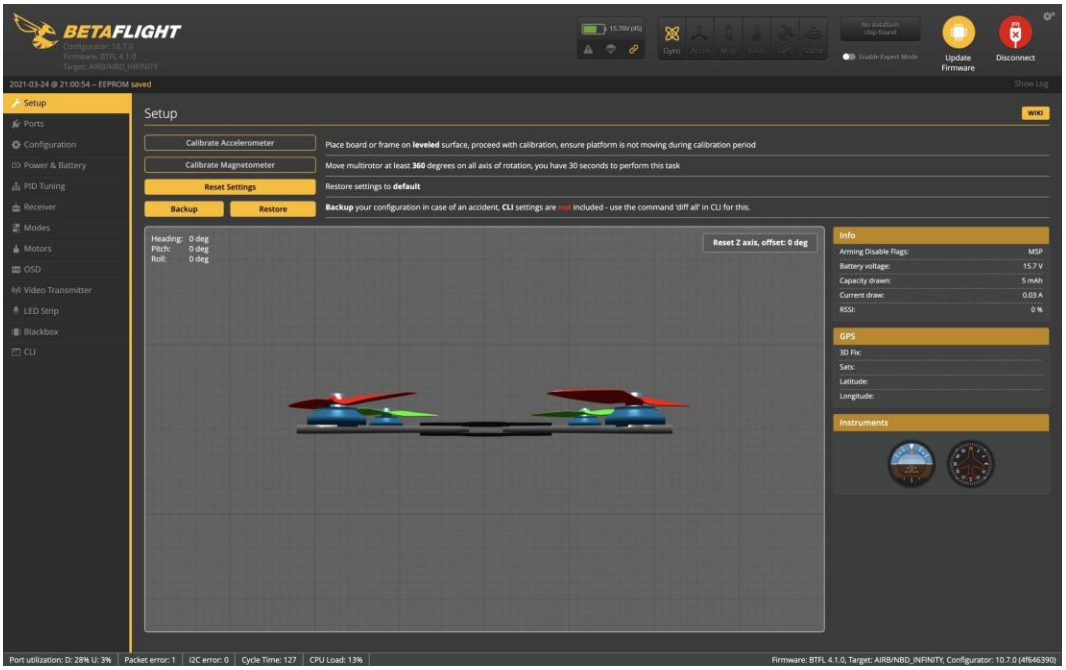


Figure 21 - Beta Flight accelerometer calibration

As shown in the figure below, the working of motors was then tested both individually and as a whole by controlling the individual bars or the Master bar (for all motors).

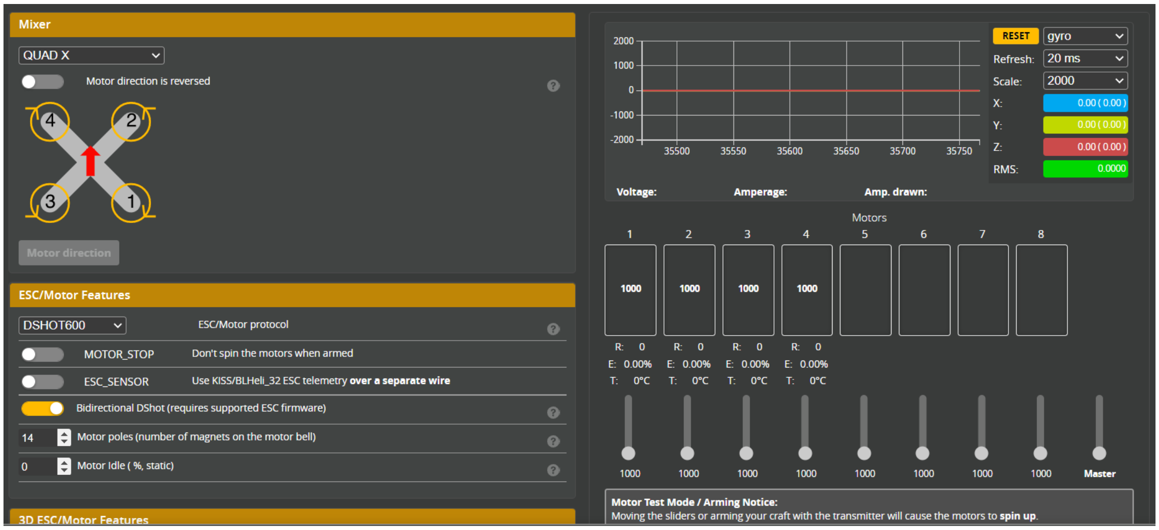


Figure 22 - Motor Calibration and controlling

The above setting could also be done wirelessly by connecting the device to the FC Wi-Fi and connecting to beta flight configurator (for PC/laptop) and using SpeedyBee application (for mobiles).

The drone comes with a preloaded software package in the Raspberry Pi microcontroller for image processing, sensor data collection, FC movement control and UI datalogging. Therefore, no further modifications are required. In case of any issues contact us for further troubleshooting.

6.3 Path finding and Image Processing (Camera Module)

The main project objective of line following was accomplished by developing an algorithm using Python, a programming language. The path contained turns, arks, and different degrees of angles, which were implemented using an arrangement of A4 white papers for testing. With the camera module attached downwards on the drone, the images were initially captured in RGB color mode. The main purpose of implementing a downward camera is to capture images more effectively since the path is directly beneath the camera. The camera captures the images with a 700×400 range in pixels, which was resized into 480×360 to reduce the computational power.

Image resizing decreases the computational load while preserving key features in the image, which helps the system process faster, especially in real-time video streaming.

Then, the resized image was filtered through a Gaussian Blur filter to reduce noise or irrelevant information and small pixel alternations, which can create false edges and result in inaccurate edge detection. The RGB image was converted into HSV (Hue, Saturation, Value) format for easier line segmentation. Then, a mask was applied to separate the black line from the white background or the white line from the black background. These two types of detections were tried during the implementation stage, and the path was planned to be white in color on a black (or dark) background at the final stage to match the actual demonstration location floor color. Furthermore, morphological operations like erosion, dilation, opening, and closing were applied to clean up the mask. Using a Canny edge detection algorithm, the outer boundaries or contours of the path were detected, which resulted in obtaining the path in the original image. Moreover, a line detection algorithm called the Hough transform fitted previously detected lines' paths. The center of the path, also called the mid-point (which can also be known as the center of the detected contours), was calculated. This midpoint is important because if the drone is located away from the path, it uses this mid-point as the reference point to get back to the trajectory.

This image was divided into three equal sections to achieve a 1×3 matrix at the beginning. In the later stages of the project, the image was separated into sections of a 3×3 matrix, where each section was given a value of 1 or 0 based on the dominant color in the section. As a result, 512 combinations were offered ($2^9 = 512$), which seemed a complex analysis. To simplify this, a set of masks was generated; each was 3×3 matrixes. Whenever an image was captured, a mask was applied to multiply it with relevant values in the matrix and add the outcome. Then, the mask that gives the highest total value was chosen, and each mask correlated with a specific angle and speed for the drone to turn and fly accordingly.

Both angles and speed vectors were initialized and planned to be fine-tuned through test runs to achieve optimal performance in the actual environment. The angle vector includes a set of different angles; the desired one was selected depending on the relevant mask. Likewise, the speed vector gives different speed values. Both selections were made to achieve accurate line tracking and smooth and sharp turnings.

The overall code for the path finding, mid-point determination and the 3x3 matrix definition for turning is represented in Appendix C

6.3.1 Image Processing Algorithms and Flowcharts

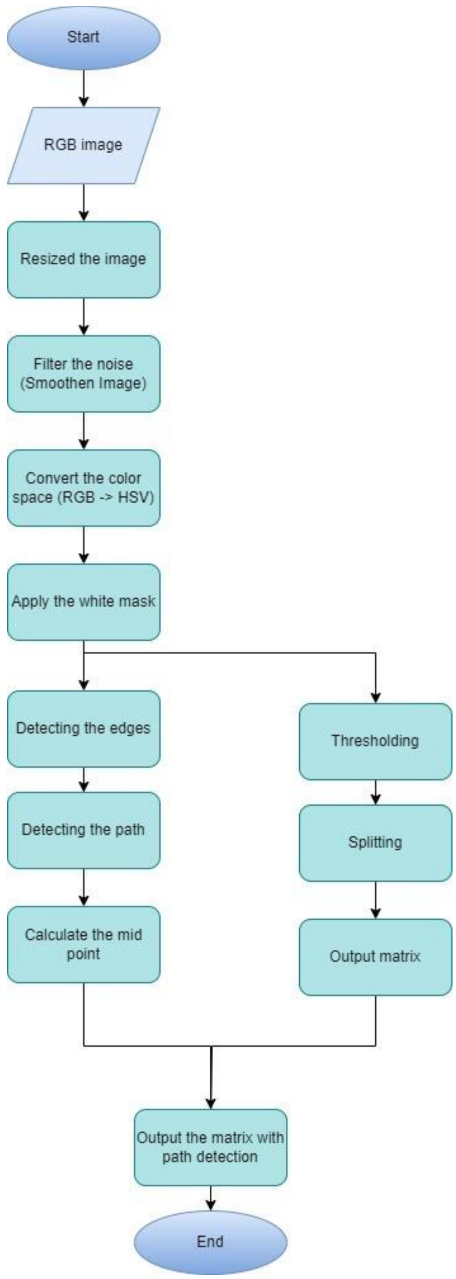


Figure 23 - Image processing detailed architecture

The “Matrix Search” algorithm shown in Figure 23 takes live streaming video as the input, giving back a 3×3 matrix of 1s and 0s as the output. Another significant fact is that a video is a set of images or frames. Therefore, the algorithm follows the same structure for both videos and images, with only minor changes in the programming. The resized image was 480×360 in size, and according to the output result, each section was 160×120.

The output of the “Matrix Search” algorithm was fed into as the input of another algorithm called “Mask Apply,” which applies all the masks one after the other to get the resulting vectors in a list. The flowchart is shown in Figure 23. Similarly, the output of this algorithm; the matrix index, was returned to the main algorithm.

6.4 Ultrasonic Sensor and Obstacle detection

The requirements of the project states that the autonomous drone should be capable of detecting direct approaching obstacles within from a distance of between 20cm to 50cm and communicate a signal to the flight controller to change the drone movement to hover mode. Initially, the ultrasonic sensor, HR-SC04, which is now connected in the final design at the center of the top plate of the frame, was connected at the front of the drone for testing purposes. The code represented in Appendix D, determines the presence of objects every second and responds if there is any object within 50cm with the distance. However, communication of the command to be sent to the flight controller was not done due to failure in the proper integration and communication between the microcontroller and FC which is recommended to be carried out as future work. The flowchart for the ultrasonic sensor operation is represented below

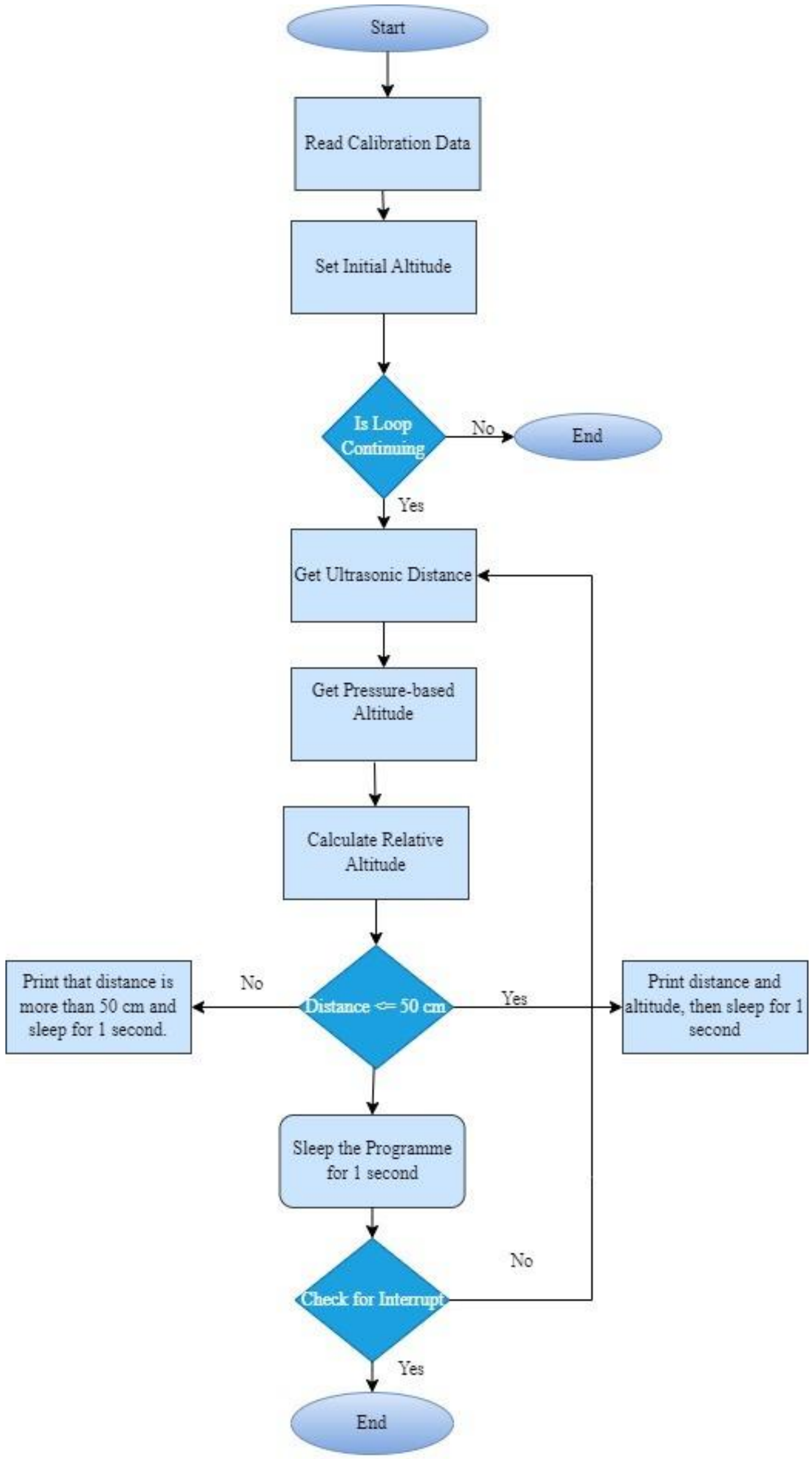


Figure 24 - Architecture for Ultrasonic and Barometric sensor

6.5 Barometric Sensor for altitude

The requirement for altitude control to be stabilized at 1.2m and altered between 0.5m to 2m were decided to be carried out using the barometric sensor BMP180 which was justified in the design elements chapter. The BMP180 sensor measures the altitude by calculating the pressure relative to the sea level (1013.25) as follows:

$$\text{altitude} = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right)$$

The section of the code which calculates this is represented below.

```
def calculate_altitude(pressure, sea_level_pressure=1013.25):
    altitude = 44330.0 * (1.0 - (pressure / sea_level_pressure) ** (1 / 5.255))
    return altitude
```

However, this produces results in 1000+ meters due to changes with the sea level. Therefore, the code was adjusted to consider the start position of the drone to be 0 value and generate a relative altitude as follows

```
current_altitude = get_average_altitude(calibration_data)
relative_altitude = current_altitude - initial_altitude
```

The full code integrated with the ultrasonic code utilized for the altitude measurement is represented in the Appendix D. However, from verification it was found that the resulting altitude in meters from the measured pressures produced a variance of $\pm$ 1 m approximately.

With lag times for sensor delivery the following sensor was used for the final design with the team recommending the use of a different sensor DSP310 which is capable of altitude calculations in the variance of $\pm$ 0.25 m for future work

6.6 Accelerometer

The accelerometer is used in the project to provide the users with forward movement speed that is communicated through to the User Interface (UI). With the battery restrictions from the selected battery after allocating 30 seconds for hovering and a total of 1 minute for obstacle avoidance and starting, the total flight time left is equivalent to:

$$Possible\ Flight\ time = \frac{200\ meter}{313\ seconds} = 0.638\ m/s$$

Therefore, the accelerometer code measures the acceleration of the drone and converts it into the speed which is shown in the UI to ensure that the drone is moving at a pace such that the battery would be sufficient for the distance travelling. The code for the accelerometer is represented in Appendix E and the flowchart is as follows.

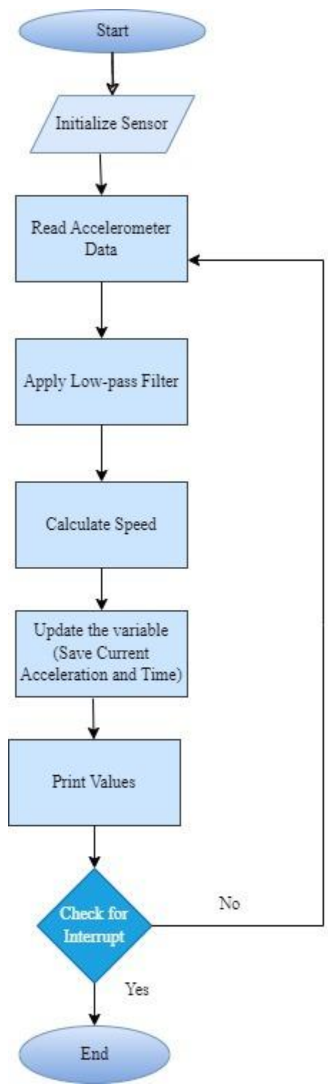


Figure 25 - Architecture for Accelerometer sensor

6.7 Architecture for overall Drone integration

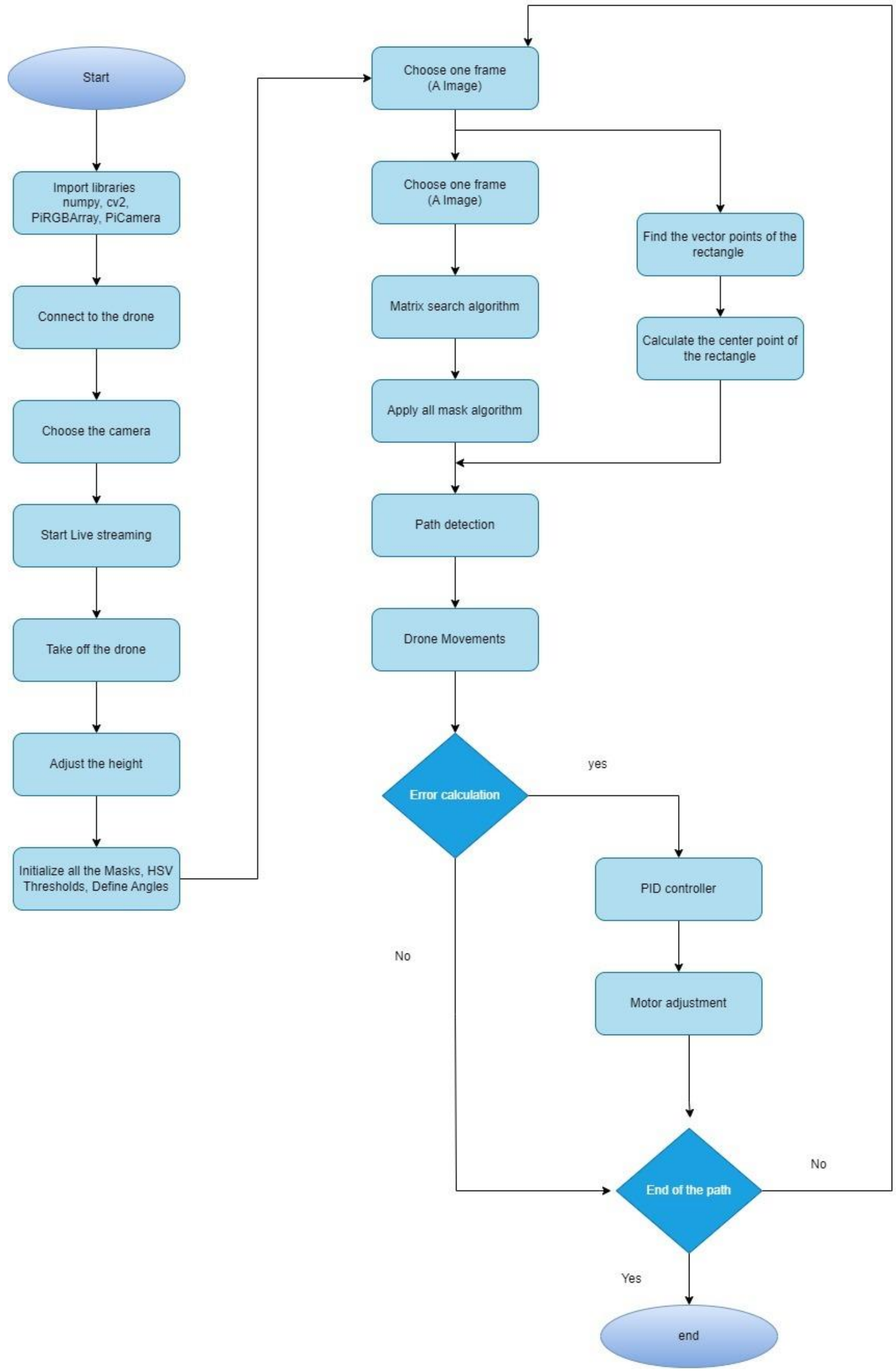


Figure 26 - Overall drone integration architecture

6.8 User Interface Design with HTML, CSS and JavaScript

- HTML: Structure of the UI

Using standard markup language, we can control our drone via Web page. It consists a series of elements such as head, body and paragraphs as represented below.

<!DOCTYPE html>	**HTML5 document**
<html>	Root element for the page
<head>	Includes meta information of respective page
<title>	Specifies title
<body>	Contains main visible contents for instance, headings, paragraph, images, hyperlinks, tables, lists, etc.
<h1>	Large heading
<p>	paragraph

- CSS: Styling the UI

Cascading Style Sheets shows how to display HTML content on screen and how to control colour, fonts and spacing multiple layouts and pages at the same time.

Example:

```
h1 {
        font-size: 36px;
        font-weight: bold;
        margin: 20px 0;
        background-color: blue;
        color: white;
        padding: 10px 20px;
        border-radius: 10px;
        text-align: center;
        width: 300px;
```

```
    position: relative;
  }
```

Example shows CSS content for title “TEAM12 Dashboard” with making prominent font size and noticeable font weight. Color stand out against blue background and border-radius used for round and softer appearance.

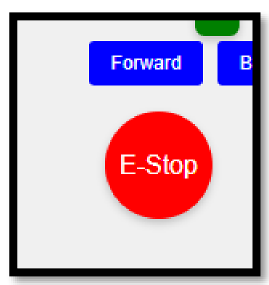
- JavaScript: Interactivity

Behavior of webpage is defined by JavaScript. It makes webpage more dynamic and interactive. It includes response like click, mouse movement and keyboard inputs, which used for forward/backward and E-stop buttons in this project.

Example

```
function sendCommand(command) {
  console.log('Command sent:', command);
  const autoLed = document.querySelector('.led.auto');
  const manualLed = document.querySelector('.led.manual');

  if (command === 'auto') {
    autoLed.style.opacity = '1';
    manualLed.style.opacity = '0.5';
  } else if (command === 'manual') {
    autoLed.style.opacity = '0.5';
    manualLed.style.opacity = '1';
  } else if (command === 'estop') {
    console.log('Emergency Stop activated!');
    // Add logic for emergency stop if needed
  }
}
```



Dynamic Content: when we click on E-stop button it will send "**Emergency Stop activated!**" to the raspberry pi console and microcontroller will stop all the operations and slowly land drone to the safe position.

6.8.1 Finalized UI Webpage

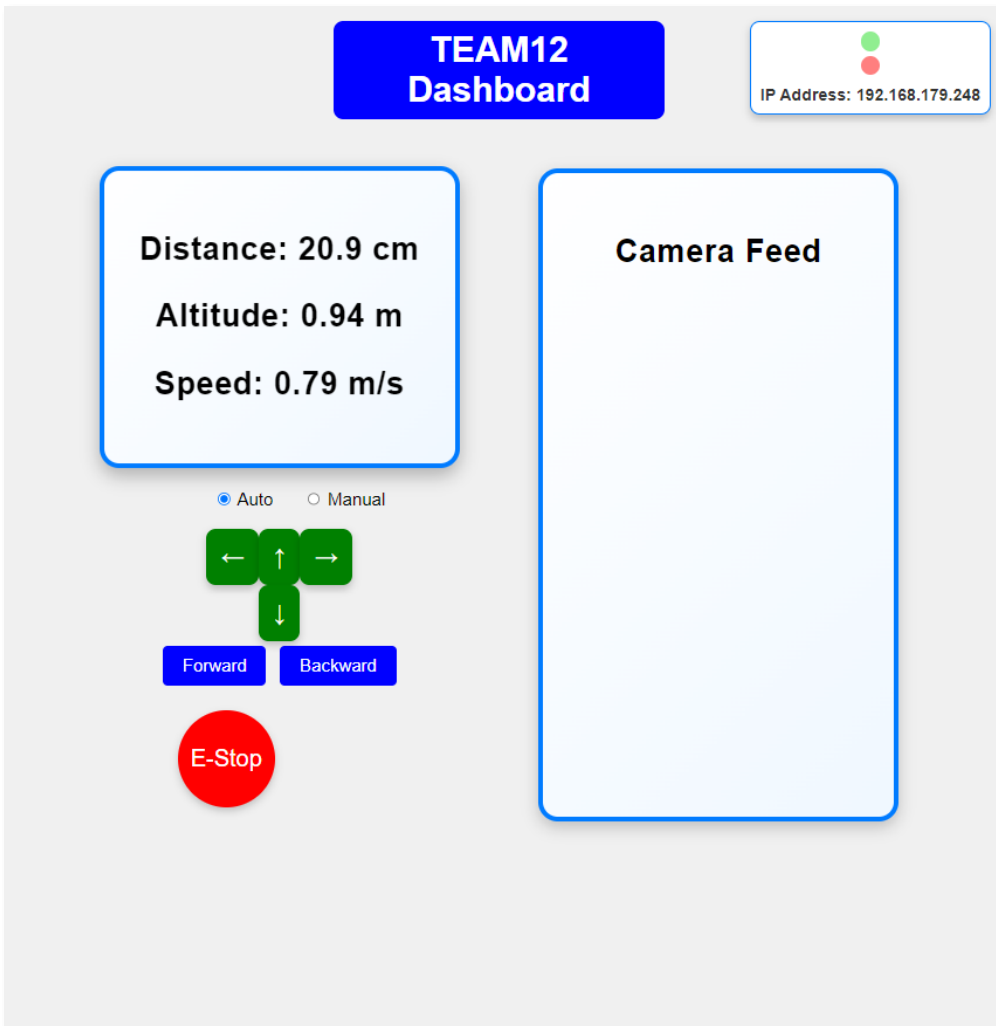


Figure 27 - Final UI

The code given represents a basic dashboard for a software dubbed "TEAM12" that is accessible on the web. HTML is the primary language used for this type of web page, with CSS backing it up for good design and JavaScript providing the necessary functionality. The UI is intended to show results such as how far the end points are, the planes' feasible altitudes, where the planes are, and the speeds attained, as well as accepting instructions on how to use the system and the various operations related to carrying out the drone activities.

- Structure of the HTML Doc

The HTML begins with the <!DOCTYPE html> definition that indicates an HTML5-compliant document. The language attribute of supported languages in the html tag is English. In the header, a UTF-8 character coding is defined, and a 'viewport' meta tag is used in order to adjust the properties of the website's proportions for displaying it on different devices. Finally, the title of the webpage is 'TEAM12 Dashboard'.

The main part of the document, which is the body, consists of many components. At the very top is a title together with the header-tagged <h1> text. Just below the title, the page also displays the status box showing, in this case, the active mode (auto or manual) along with the respective system's IP address. The next part after the status box is a control panel containing buttons and radio buttons for the mode section, which is used to interact with the contained features of the dashboard.

- Those CSS Codes for Styling

All the CSS codes are contained within the target in the <head> that is <style>. The styles are the ones that describe what the dashboard looks like. The user-friendly interface form is well organized such that for the content provided, the body uses the San Serif font, a light gray color with a centered quality, and adjusts well to different screens. The header is highlighted by a blue background and text in glistening white.

Another class is Boxbar, which flashes on the top right side of the screen with a light gray background and deep blue border. For its styling, it has a box shadow that creates a nice hover effect. In the status box on the leds indicating the status of the system, one below shows the signal 'auto'; it is blue in color, symbolizing its status, and the area one below it represents the'manual' system, and it is green in color.

There are other parts of the system, like the camera module and data server, whose interfaces are designed with a rounded corner along with a box-shadow effect, making it appear more presentable and cleaner. There is a mouse-over effect that makes the colors change as well as the elevation of the components a bit.

- JavaScript Functionality

The JavaScript area contains, for instance, methods that are responsible for the modification of data present in the dashboard and the execution of various commands following the articulated action by the user. To describe the functioning of these on-lead functionally can be extended to include the functions <updateValues>, <sendCommand>, and the <_handleClick. Event handler. Adequate functionality is added to the system in order to provide the ability to enable E-stop conditions through additional functions.
For how the content is achieved, the update values function is implemented as an event listener, and when the window load event is triggered, the function will execute and call the function <updateValues> or <getdata, which should display the latest data on the dashboard.

Overall, an elegant synthesis of HTML, CSS, and JavaScript aids in the designing of modern UIs while not sacrificing their functionality or accessibility. It is through the utilization of detailed layout systems like semantic UI and implementation of responsive design, as well as accessibility, that sans-scripts assistance web applications can be created.

6.9 Integration Issues

The proposed design faces certain challenges due to the integration between the flight controller and the microcontroller due to delays in processing data in the microcontroller and synchronization issues. Furthermore, the budget restrictions limit the selection of high-quality components which are readily compatible with each other. Thus, the design had to compromise on optimal performance. Availability of flight controllers that were within the budget also posed a problem, thus limiting the selection of motors and corresponding propellers and batteries. This compromise led to performance issues, but all components were confirmed to be compatible with each other through testing and prototyping.

7. Design Validation and Testing

Validation and Verification tests and results are discussed in the following section in comparison to the newly prioritized Requirements Analysis table.

7.1 Physical Damage Prevention

This is the validation for the new Requirement ID 1 which is ensuring proper safety is assigned preventing any physical damages to bystanders and users.

Test Description

Ensure that the propeller guards of the drone is providing protection to users from any contact with the propellers.

Test Results

The test depended on the arrangement and assembling of the drone and as represented in the below image, the motors and propellers are shielded by the frame ensuring safety is achieved on an acceptable level

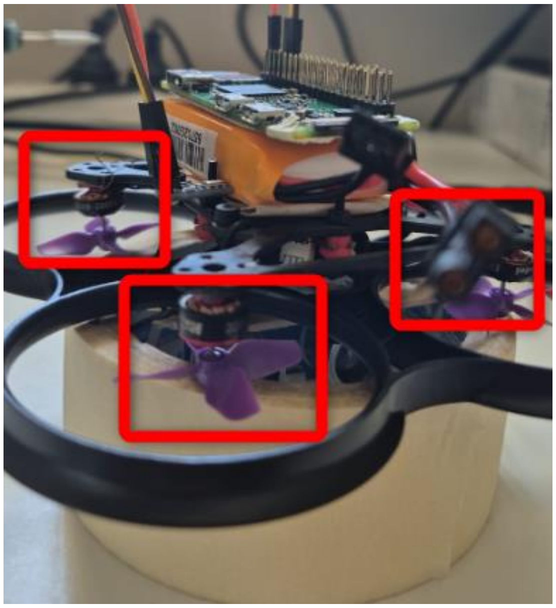


Figure 28 - Propellers and motors safeguarded by the inbuilt propeller guards

7.2 Flight Safety

This is the validation for the new Requirement ID 2 which ensures that the manual mode of the UI is functioning to ensure safety flight in-case of autonomous malfunctions

<u>Test Description</u>

Enable manual mode and provide manual U R L D signals and observe if the communication is shared to the microcontroller which should related the information to the FC

<u>Test Results</u>

```
/home/team12/Desktop/Ultrasonic.py:169: RuntimeWarning: No channels have been set up yet - nothing to clean up!  Try cleaning up at the end of your program instead!
(myenv) team12@raspberrypi:~/Desktop $ python Ultrasonic.py
Serving on port 5000...
192.168.1.101 - - [21/Oct/2024 22:41:54] "GET / HTTP/1.1" 200 -
Auto
192.168.1.101 - - [21/Oct/2024 22:41:56] "GET /auto HTTP/1.1" 200 -
192.168.1.101 - - [21/Oct/2024 22:41:58] "GET /Manual%20U HTTP/1.1" 404 -
192.168.1.101 - - [21/Oct/2024 22:41:58] "GET /Manual%20L HTTP/1.1" 404 -
192.168.1.101 - - [21/Oct/2024 22:41:58] "GET /Manual%20R HTTP/1.1" 404 -
192.168.1.101 - - [21/Oct/2024 22:41:58] "GET /Manual%20D HTTP/1.1" 404 -
```

Figure 29 - Results of manual mode changes

7.3 Path Following, Image Processing, Accurate Line detection and Mid-Point Calculation

The preliminary challenge of the project was designing an efficient image-processing algorithm with the capacity for real-time execution. The final code was developed with several pre-tests under various conditions and considerations. This section carries out the validation for 4 requirements (multiple requirements tested in each test) which are the new Requirement ID 3, 4, 5 and 6 in Chapter 2 of the report

<u>Test 1 Description</u>

The preliminary designing process was carried out with static images captured at home. The path was created using black A4 papers lying on a white surface. At the beginning of the process, experiments were conducted in a controlled environment without real-world complications such as floor textures and lighting variations. The goal during this phase was to develop a basic but reliable line detection algorithm, which was further developed by adjusting parameters according to the requirement.

<u>Test 1 Results</u>

The algorithm used basic thresholding and HSV Color Conversion to isolate the path from the background. Moreover, Canny edge detection and Morphological operations were performed to enhance the detection accuracy by removing unwanted noises and highlighting edges. In the end, the midpoint was successfully detected. The figure represented below shows all the results for each step by verifying the effectiveness of the early stage algorithm.

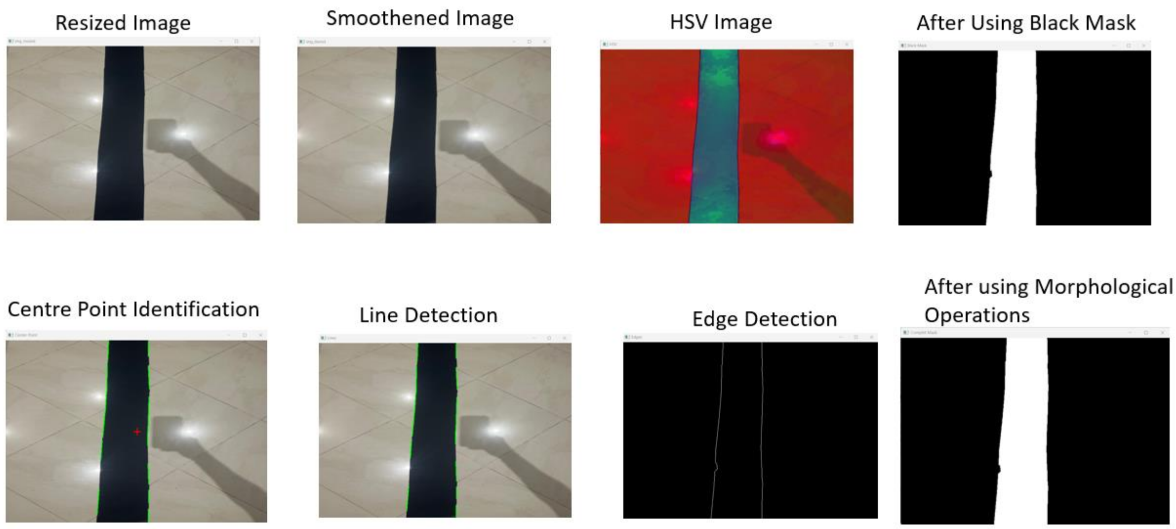


Figure 30 - Results of Line Detection Algorithm with Static Images (Straight Lines)

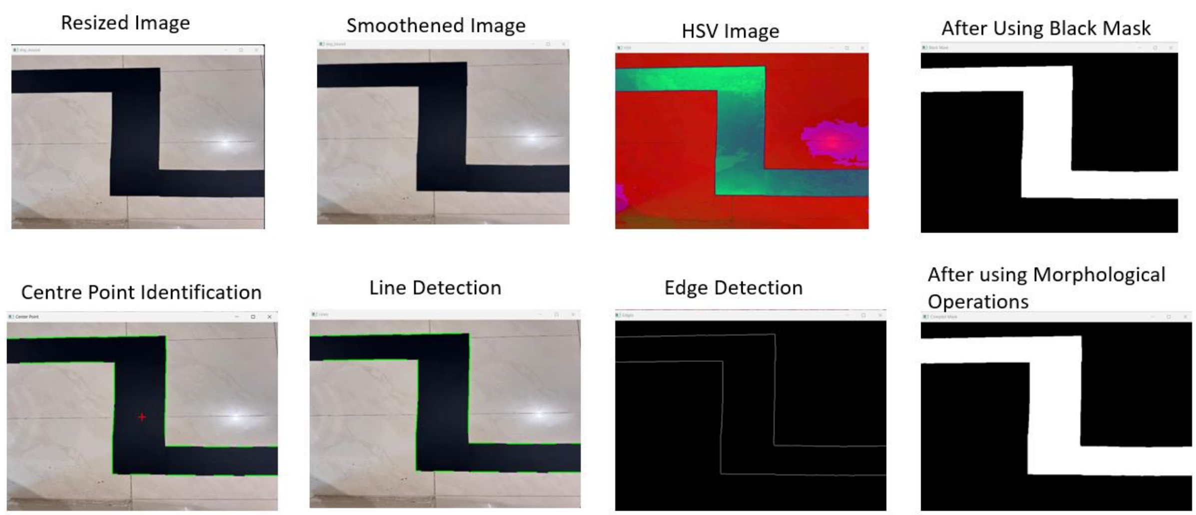


Figure 31 - Results of Line Detection Algorithm with Static Images (With Turns)

Test 2 Description

After successfully implementing the code in a controlled environment, the focus shifted to experimenting with the algorithm with images and videos captured in the actual demonstrating environment (outside of 151 Lycopodium). This transition brought new challenges, such as different floor textures (e.g., light-colored lines on dark-colored floor carpets) and various lighting conditions, including shadows, through tests carried at different HSV values which have a great impact on image processing.

Figure 32 - Pathway for testing outside Lycopodium

Test 2 Results

Parameter tuning, HSV thresholding ([0,0,188,179,33,245]), and contouring were performed to overcome these tasks. Since the real-world floor has a dark background, the path was set using white-colored papers, and HSV values were tuned for that environment. The figures below show the output results from that this code.

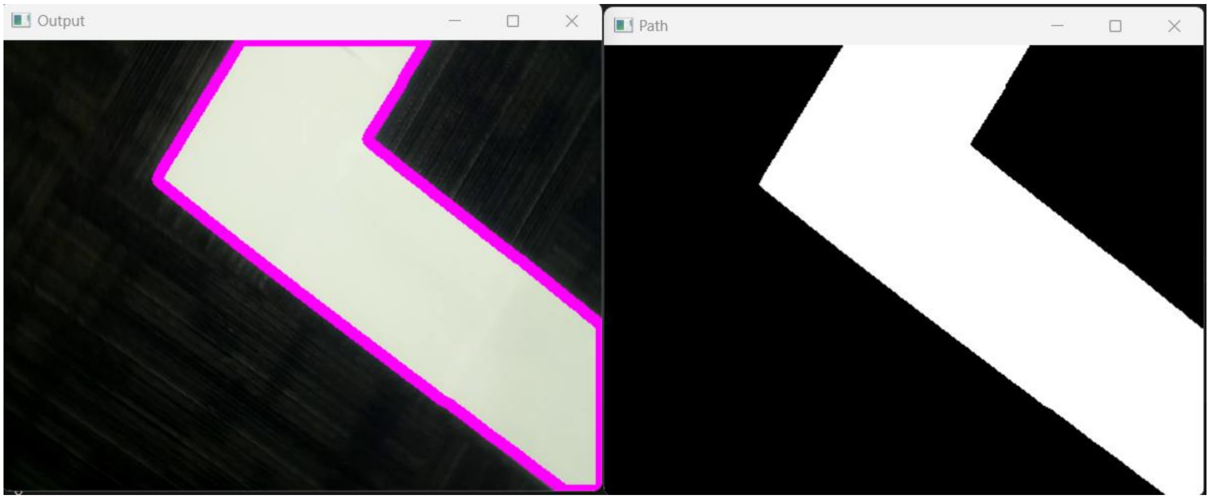


Figure 33 - Results of Line Detection in the 151 Lycopodium Environment (With Arcs)

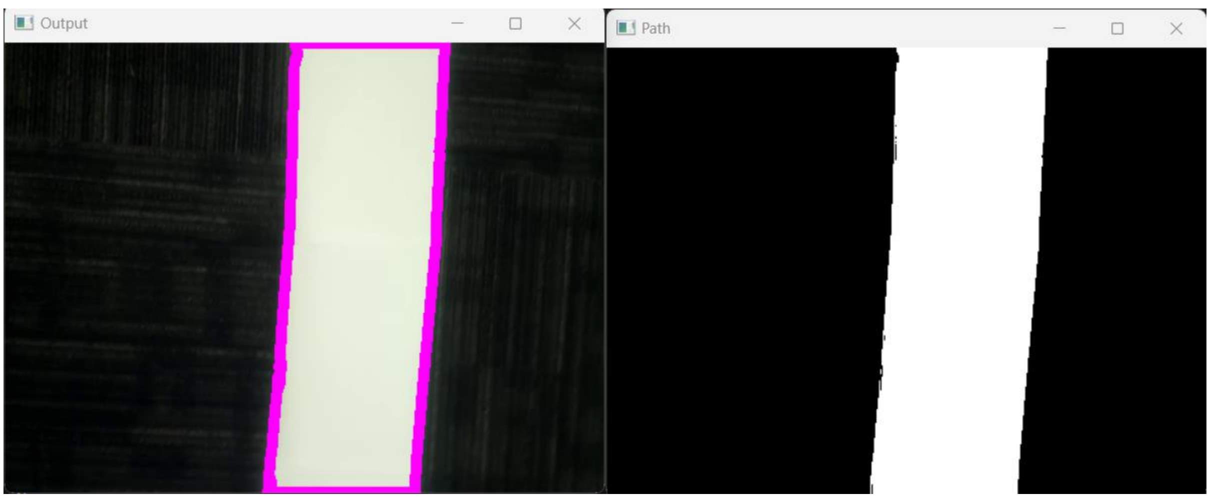


Figure 34 - Results of Line Detection in the 151 Lycopodium Environment (Straight Lines)

A "Color Picker" code was developed as an additional feature of the project, which can be used to find the HSV values in any environment. That function would be really useful when the testing location has been changed or any sudden, unpredictable changes in the testing environment happen. Below figures shows the actual environment image with ideal HSV values, and the effects of low Hue and Value values accordingly.

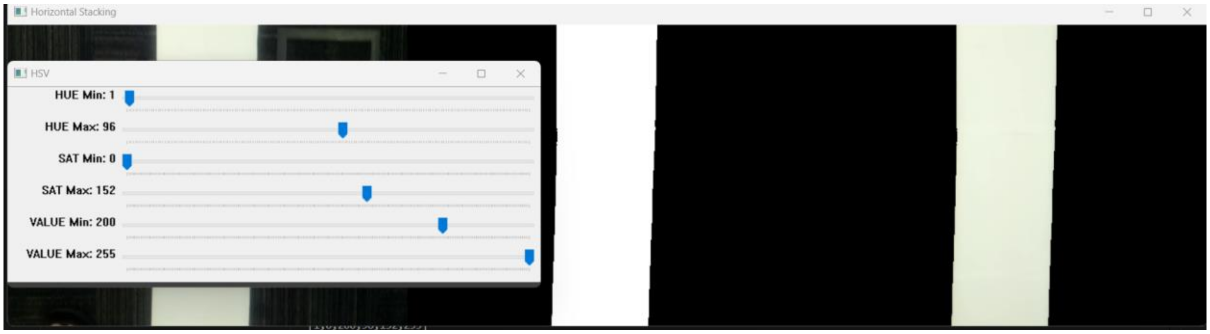


Figure 35 - With Ideal HSV Values for the Location

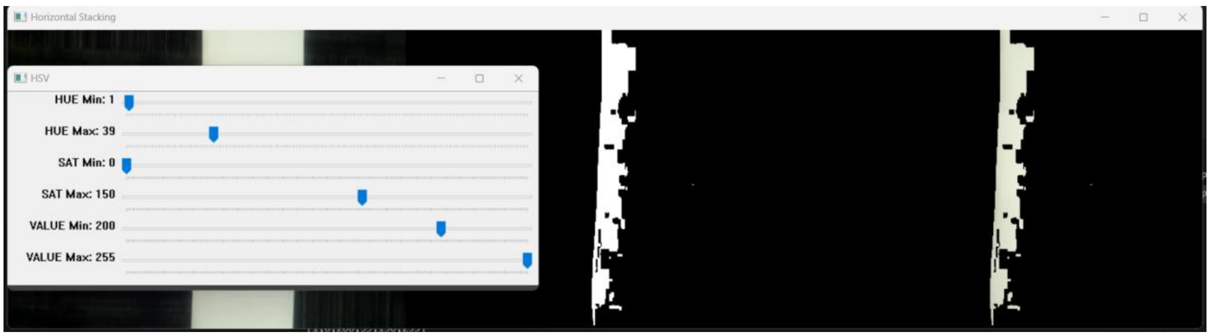


Figure 36 - Image Distortion with Low Hue (H) Values

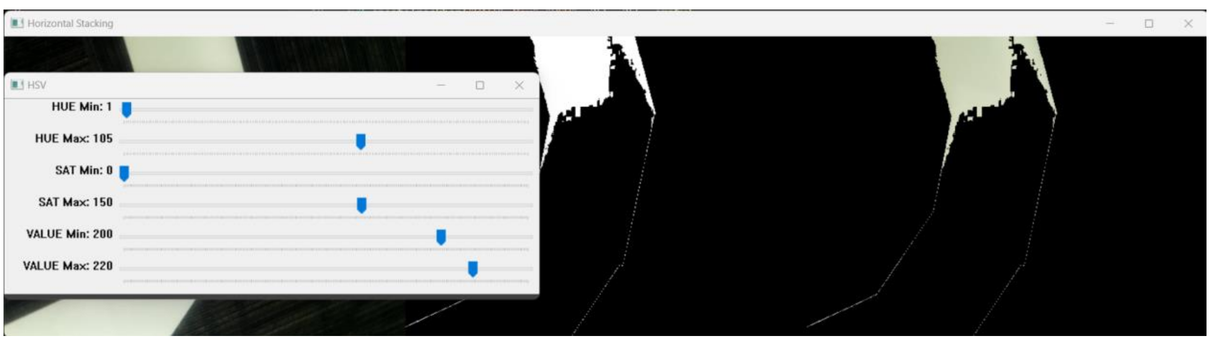


Figure 37 - Image Distortion with Low Value (V) Values

<u>Test 3 Description</u>

This testing of line following was primarily designed using a 1×3 Matrix**,** where the image was divided into three vertically equal parts. Each section was assigned a binary 0 or 1 based on the line appearance (if no line “0”, else “1”). A total of 8 outputs (2^3) were used for the drone’s movements.

<u>Test 3 Results</u>

As an example, a vector of [0, 1, 0] means the drone is in the middle, resulting in forward movement, while a vector like [1, 0, 0] or [1, 0, 1] means the drone needs to turn right and left respectively. Below figures show the algorithm results.

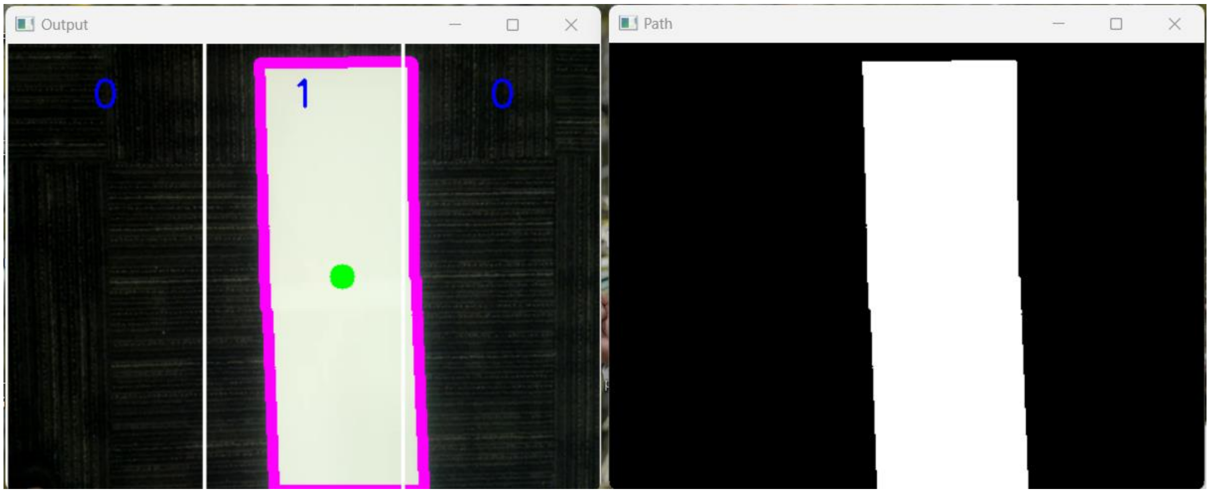


Figure 38 - Results for Forward Movements

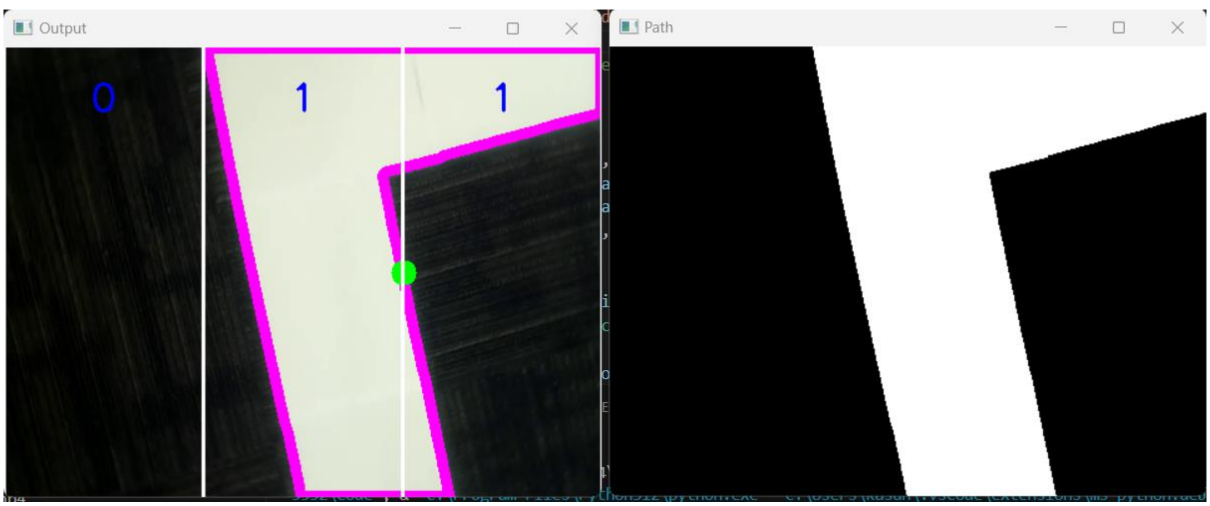


Figure 39 - Results for Slight Right Turn

The figure below gives the complete decisions for matrix outputs.

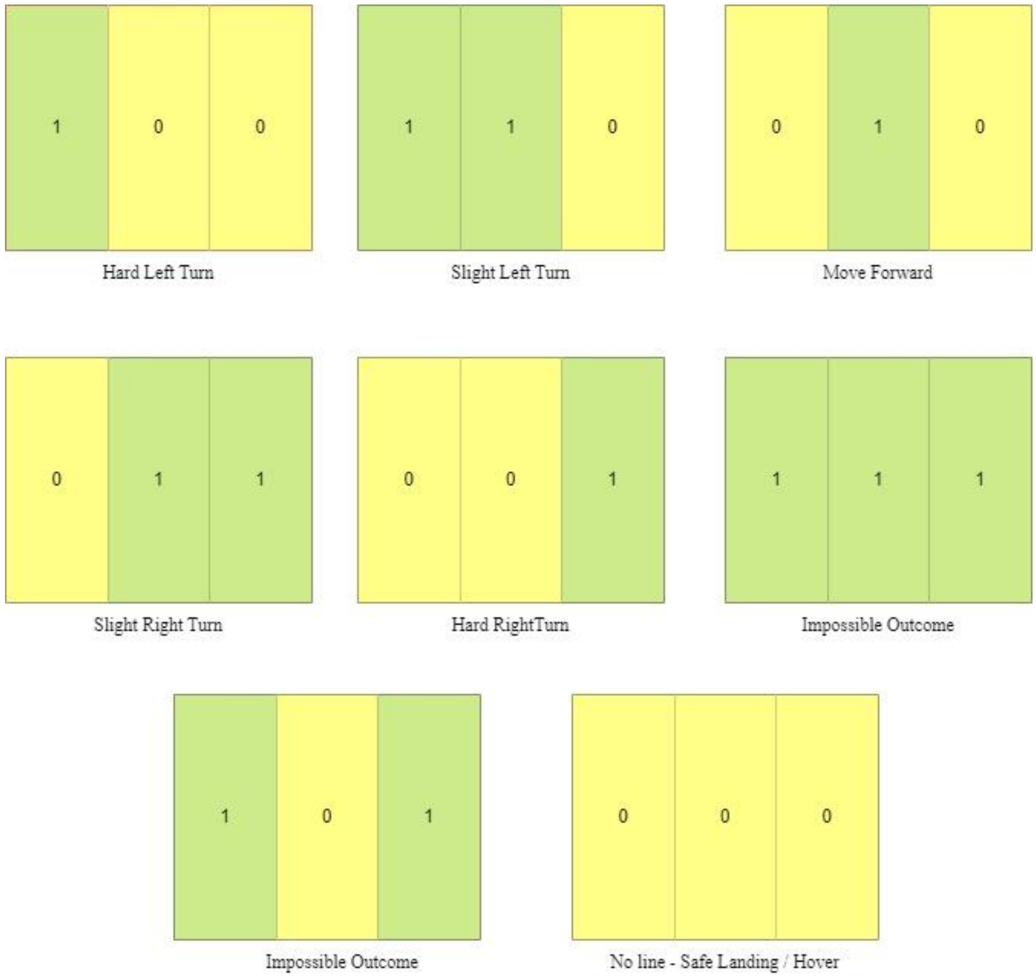


Figure 40 - 1×3 Matrix Output Vector and their Decisions

Test 4 Description

With the progress of the project, the line following code developed with a 3×3 matrix approach in order to achieve accurate and smooth turning, especially in sharp edges. In this stage, the image has been divided into nine sections, each of which assigned either 1 or 0 depending on the line's present. Below figure shows the vector combinations used for decision making in the final code developed.

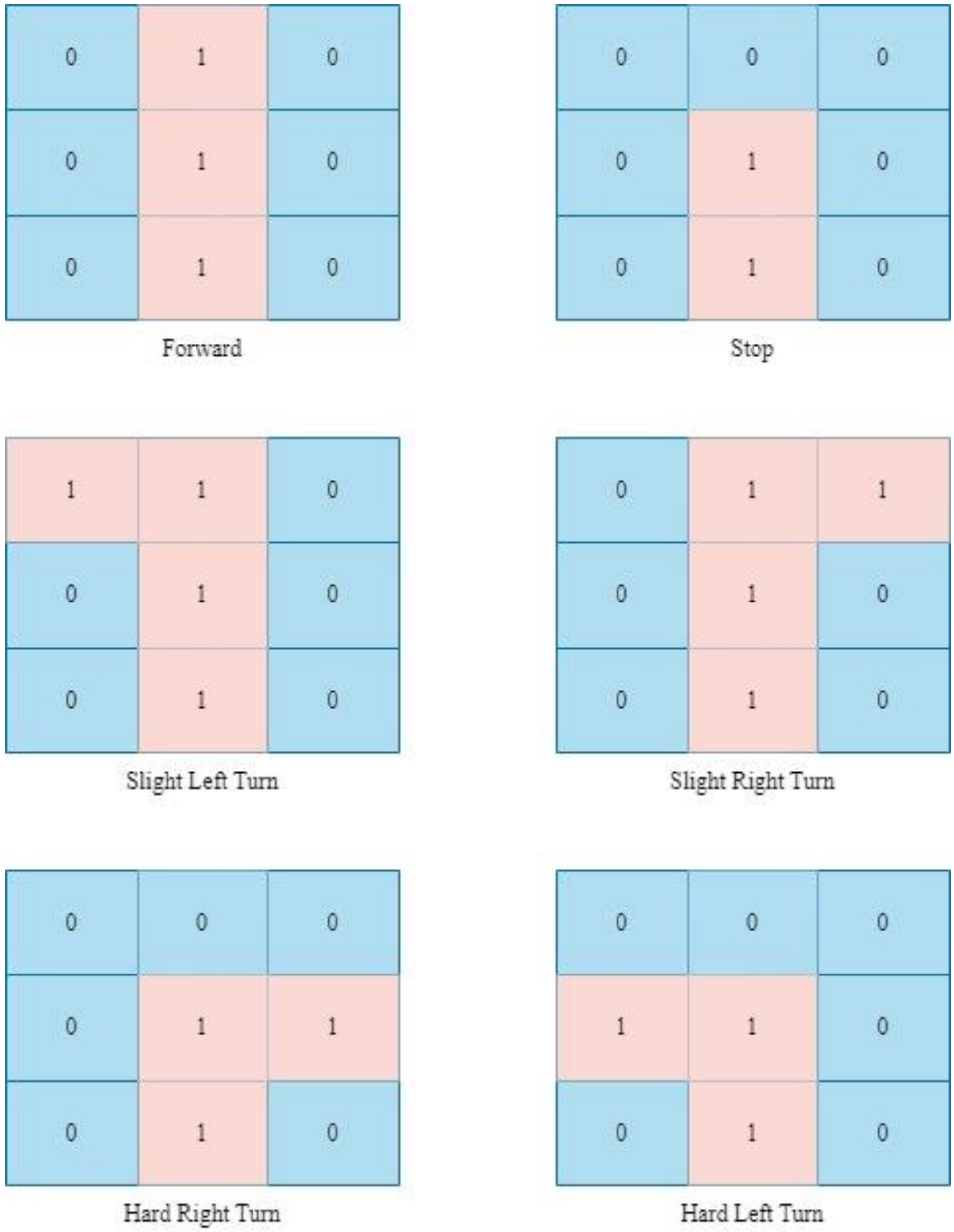
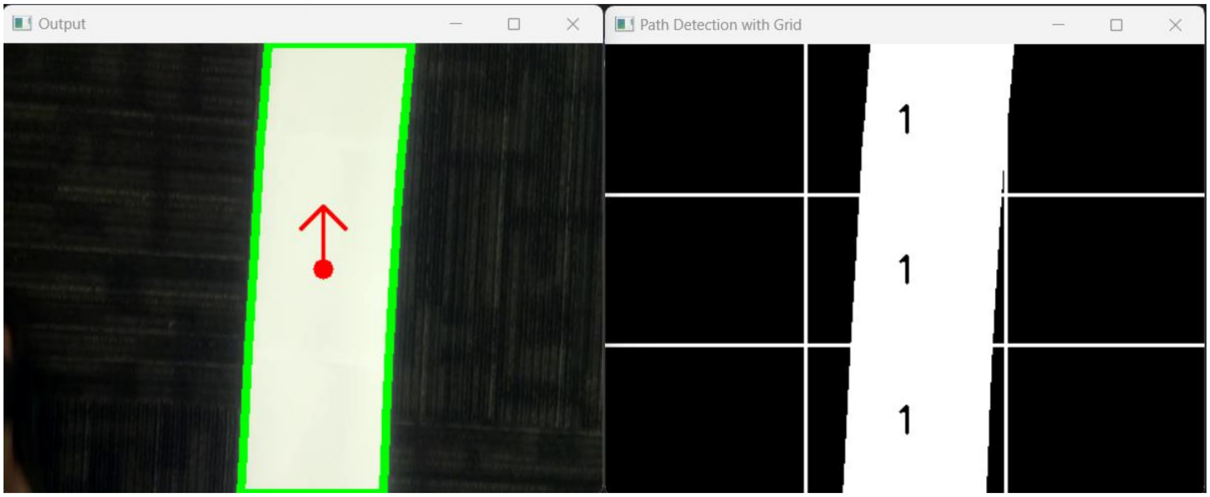


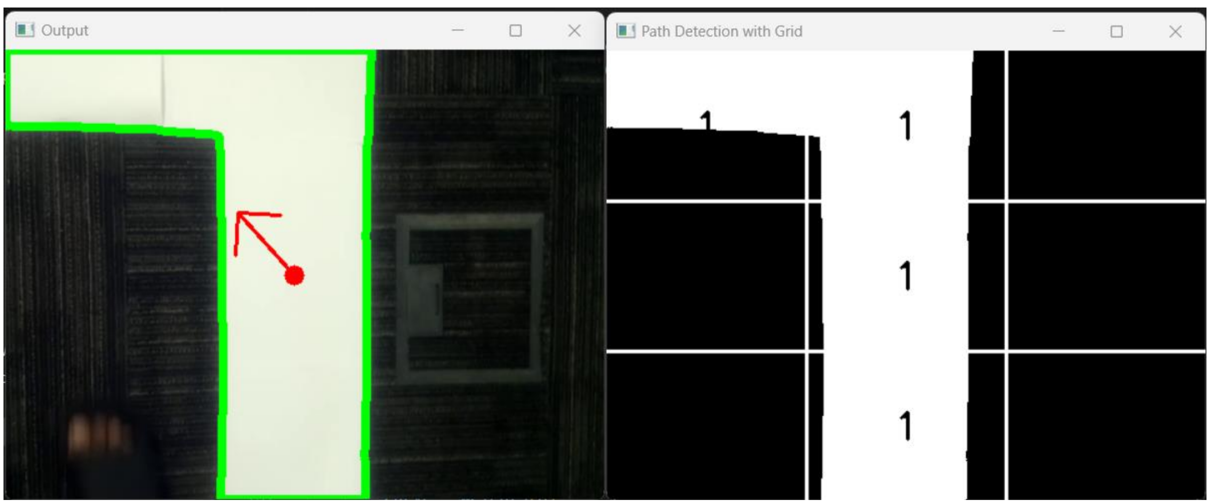
Figure 41 - 3×3 Matrix Used Output Vector and their Decisions

<u>Test 4 Results</u>

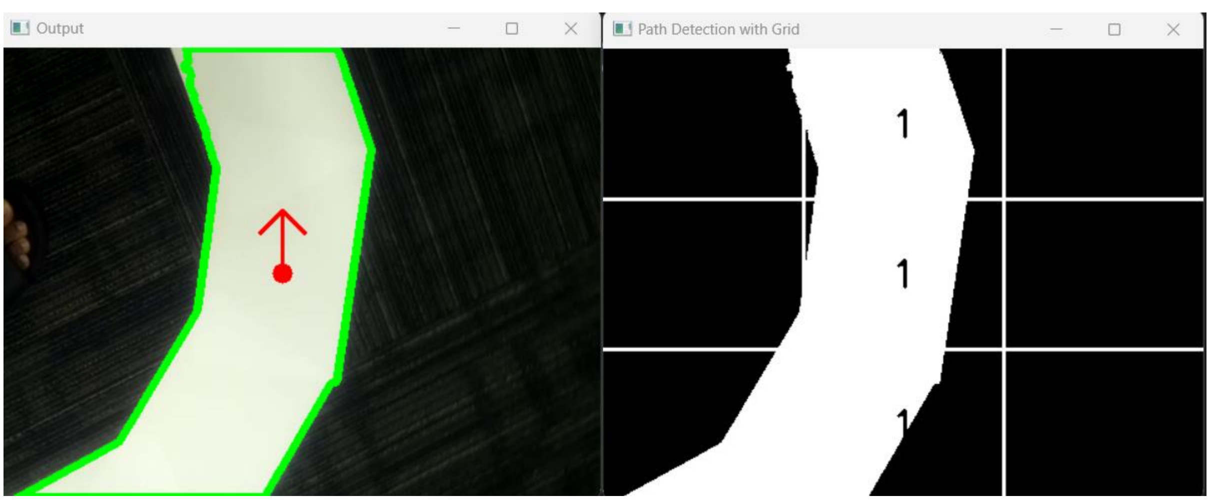
After running the algorithm, the following results were obtained for different turns, and curves. It was clearly seen that; the results were more precise than 1×3 matrix outputs.



Output
Path Detection with Grid
1
1
1



Output
Path Detection with Grid
1
1
1
1



Output
Path Detection with Grid
1
1
1

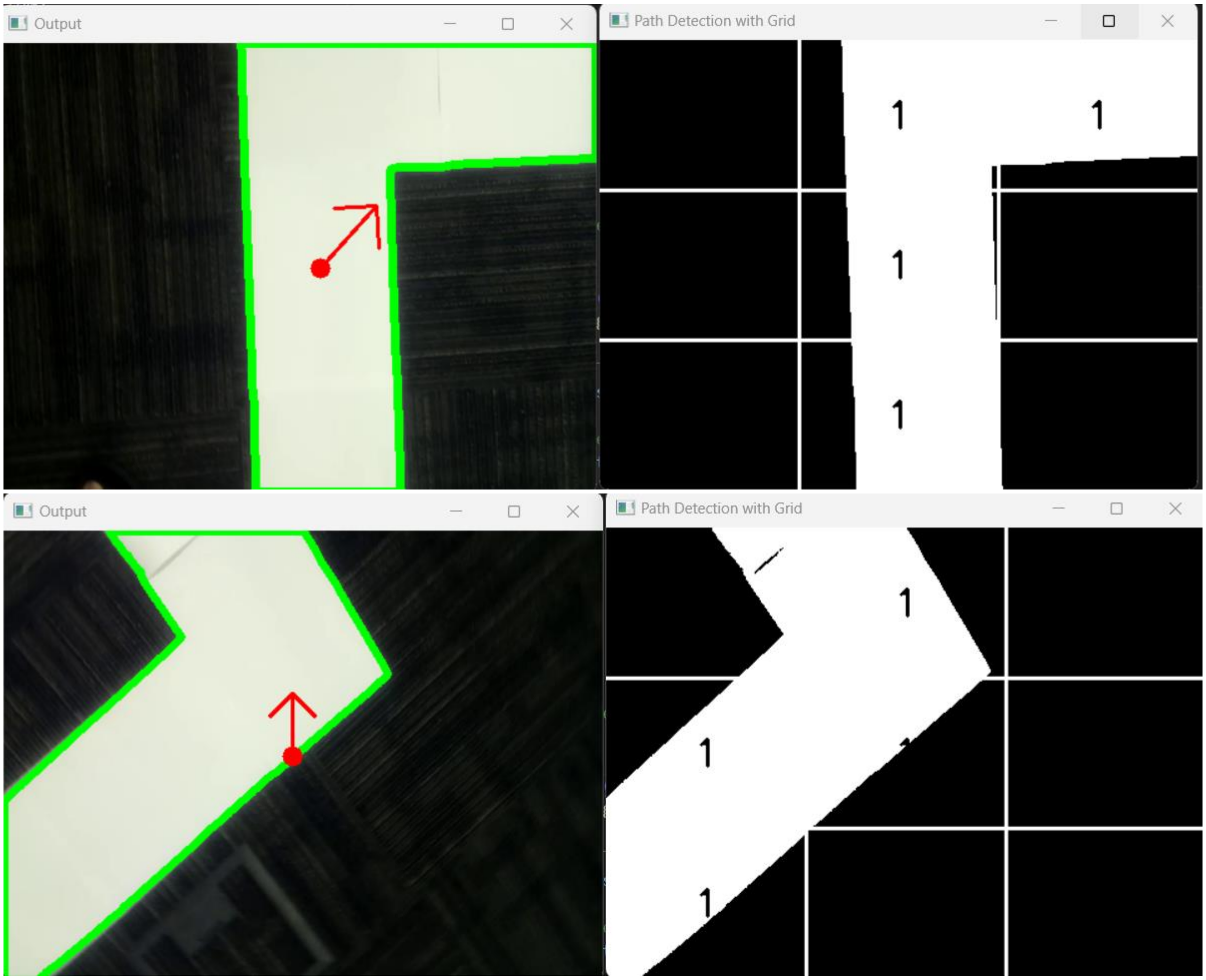


Figure 42 - Verification of 3x3 matrix for turn identification

7.4 Hovering capability

This section carries out the validation for the new Requirement ID 7 in Chapter 2 of the report. However, the team failed to carry out any testing for this feature as the integration between the microcontroller and the flight controller was not carried out successfully.

The planned test was to utilize the ultrasonic sensor and barometric sensor to establish altitude control at 1.2m (measured previously) to see if the variance of drone movement was within an acceptable range. The hovering code is represented in the Appendix F. However, results were not obtained due to failed integration and the detailed test description and expected results are demonstrated in the recommendations for future works

7.5 Precautions for Emergency Stop

This section carries out the validation for the new Requirement ID 8 in Chapter 2 of the report. The E-stop button has been designed in the UI. However, due to failure to integrate communication the testing was not successfully carried out.

The planned test was to simulate a battery failure and communication failure scenario to press the UI which would have been used to determine if the drone immediately safe landed as per the command. Further information on the planned test is demonstrated in the recommendations for future works

7.6 Obstacle Avoidance

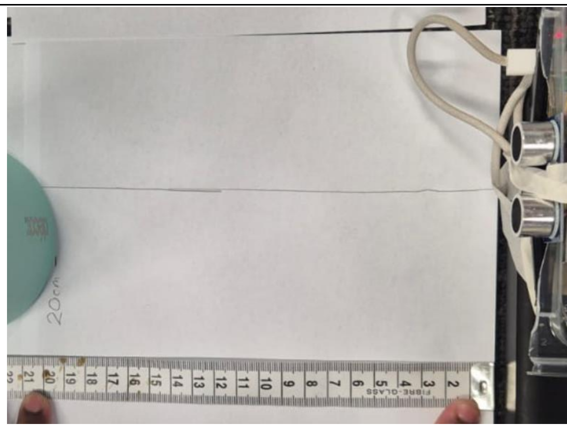
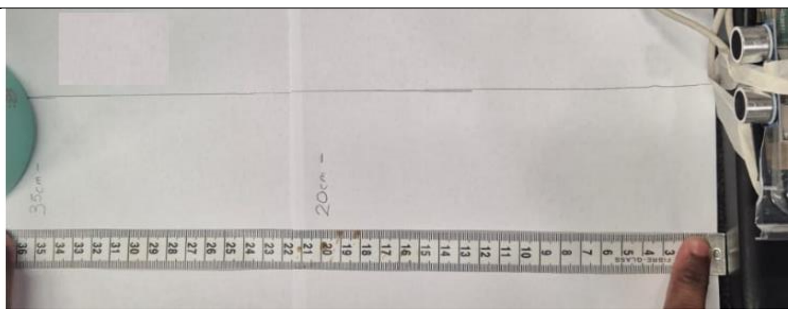
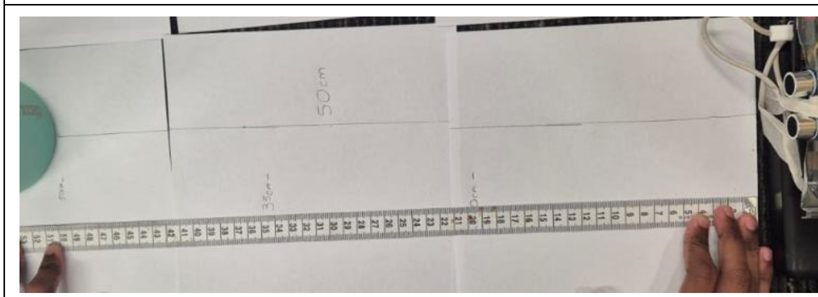
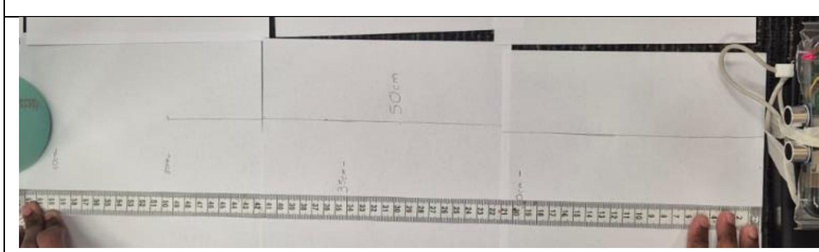
This section carries out the validation for the new Requirement ID 9 in Chapter 2 of the report which was for the drone to detect any object within the vicinity of 50cm and provide the necessary commands to the drone. The command to the drone was not tested due to failed integration but the sensor was tested out individually to produce the following results

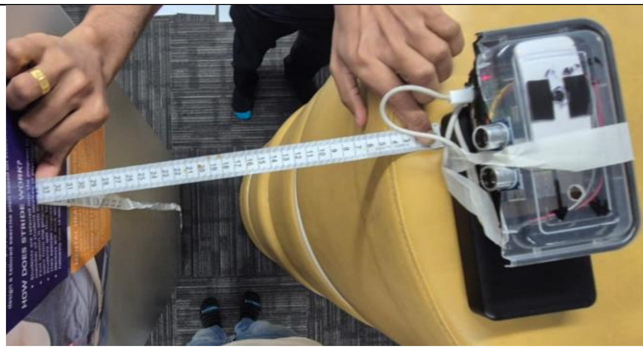
<u>Test 1 Description</u>

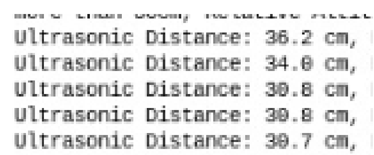
Objects located at different locations (previously measured) were tested against the output resulting from the code. If the object was within range the response should be “Ultrasonic Distance: XX cm”. The test was carried out using a bottle located at 20cm, 35cm, 50cm and 60cm to produce the following results

<u>Test 1 Results</u>

Table 15 - Ultrasonic sensor validation results

Distance	Result of Code
	Ultrasonic Distance: 19.4 cm, Ultrasonic Distance: 19.7 cm, Ultrasonic Distance: 19.1 cm, Ultrasonic Distance: 20.5 cm, Ultrasonic Distance: 18.3 cm, Ultrasonic Distance: 19.5 cm, Ultrasonic Distance: 20.1 cm, Ultrasonic Distance: 20.9 cm, Ultrasonic Distance: 19.4 cm, Ultrasonic Distance: 19.8 cm, Ultrasonic Distance: 19.8 cm, Ultrasonic Distance: 19.1 cm, Ultrasonic Distance: 19.8 cm, Ultrasonic Distance: 20.2 cm,
	Ultrasonic Distance: 34.0 cm, Ultrasonic Distance: 34.3 cm, Ultrasonic Distance: 34.0 cm, Ultrasonic Distance: 34.3 cm, Ultrasonic Distance: 34.0 cm, Ultrasonic Distance: 34.1 cm, Ultrasonic Distance: 34.6 cm, Ultrasonic Distance: 34.4 cm, Ultrasonic Distance: 34.1 cm, Ultrasonic Distance: 34.2 cm, Ultrasonic Distance: 33.9 cm,
	Ultrasonic Distance: 48.0 cm, Ultrasonic Distance: 48.5 cm, Ultrasonic Distance: 48.4 cm, Ultrasonic Distance: 47.8 cm, Ultrasonic Distance: 48.1 cm, Ultrasonic Distance: 48.3 cm, Ultrasonic Distance: 48.0 cm, Ultrasonic Distance: 49.3 cm, Ultrasonic Distance: 48.8 cm, Ultrasonic Distance: 48.4 cm, Ultrasonic Distance: 49.5 cm,
	more than 50cm, more than 50cm, more than 50cm, more than 50cm, more than 50cm, more than 50cm, more than 50cm,





The test results demonstrate that the ultrasonic sensor code clearly identifies the objects and returns a value with is within an acceptable range of variance.

<u>Test 2 Description</u>

The ultrasonic sensor range was also tested out by placing a bottle at different locations of a 50cm line to identify its field of vision

<u>Test 2 Results</u>

The above test was carried out at different locations to determine that the field of vision was 60 degrees from the center of the ultrasonic sensor

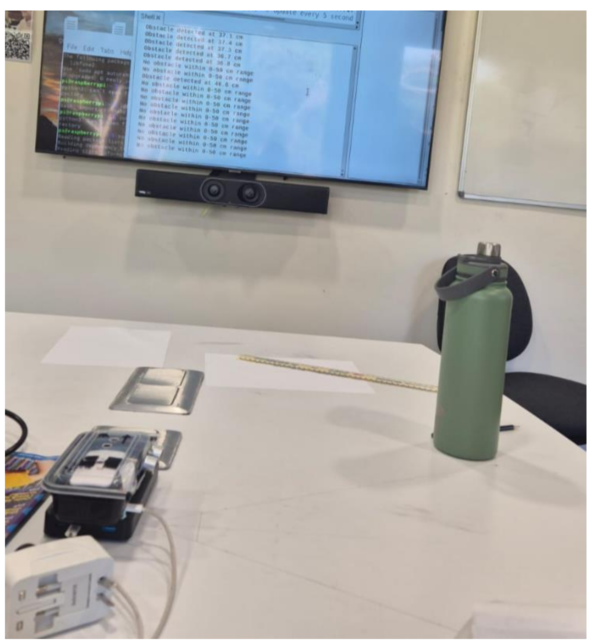


Figure 43 - Ultrasonic Sensor Field of Vision Testing

7.7 Budget

This section carries out the validation for the new Requirement ID 10 in Chapter 2 of the report which tests the maintenance of the project budget

Test Description

Ensure that the budget is well maintained throughout the project, with expenses being shared within the group prior to purchasing, critical evaluation of components in terms of cost considerations in design elements selection and purchasing locally available items to prevent shipping costs

Test Results

The team was able to maintain the budget with the limit of AUD 350. As represented in the table below, the budget calculated for the project was equivalent to AUD 347.23

Table 16 - Project budget overview

Component	Price (AUD)
Ultrasonic Sensor	14.3
Barometric pressure sensor	15.75
Raspberry PI zero W	49.95
3 Axis Compass	15.75
Jumper Leads	4.65
Pin header	2
Header pin 40 way	1.1
5MP camera	36.25
Flight Controller	72.95
Brushless Motor (Anti)	26.4
Brushless Motor (Clock)	26.4
Lipo Battery	11.97
Lipo Battery Charger	35
NewBee Drone	20.89
Propellers	13.87
Total Cost of the Project	**347.23**

7.8 Proper Laboratory Induction

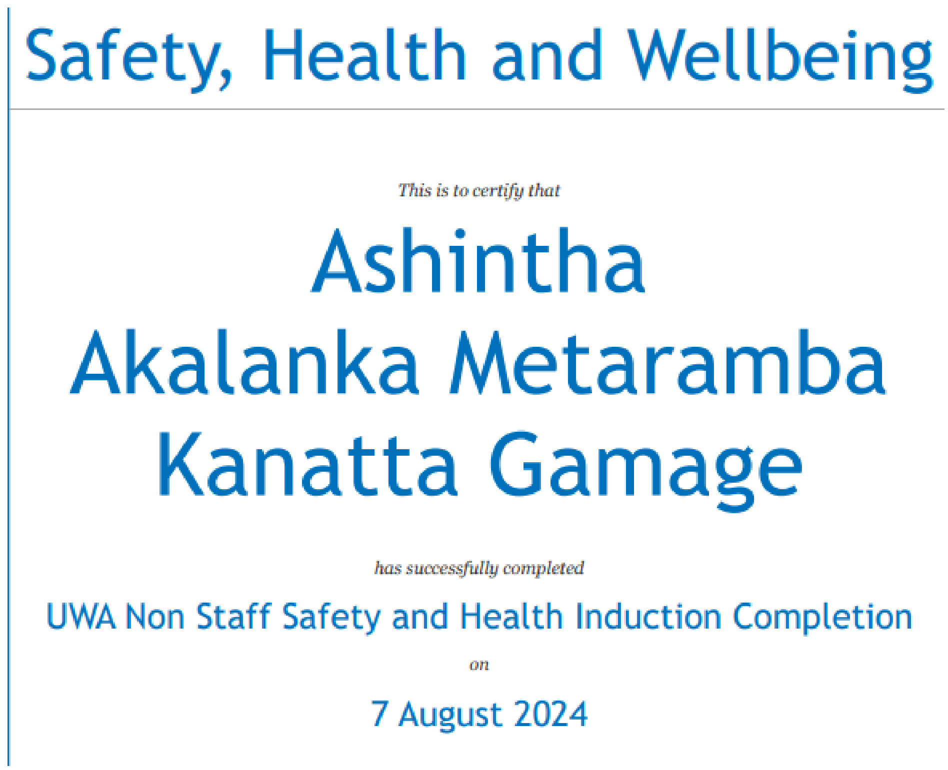
This section carries out the validation for the new Requirement ID 11 in Chapter 2, which ensures that safe practices are carried out in the designing of the drone

<u>Test Description</u>

Carry out the online source materials and quizzes related to the lycopodium laboratory induction (to be carried out by all members)

<u>Test Results</u>

The induction was successfully carried out by the entirety of Team 12 receiving certificates as represented below


Safety, Health and Wellbeing

This is to certify that

Ashintha Akalanka Metaramba Kanatta Gamage

has successfully completed

UWA Non Staff Safety and Health Induction Completion

on

7 August 2024

Figure 44 - Laboratory induction completion example

7.9 Travel Distance

This section carries out the validation for the new Requirement ID 12 in Chapter 2.

<u>Test Description</u>

The planned test for the validation was to carry out a trial-demonstrations in predefined pathway designed by the team to measure the time taken to travel a certain distance to see if the travel speed is sufficient to cover the 200m distance.

Test Result

However, as the team was unable to make the overall system integration for the drone to fly, the testing was unsuccessful. However, theoretical calculations were carried out to determine the time of flight capable for the selected battery as:

Energy required for 30 seconds flight hovering – 0.250 Wh

Energy required for 1 minute of flight – 0.501 Wh

Therefore, with battery purchased for testing 2S 650mAh 70C battery

- Total energy supplied = 3.367 Wh

Therefore, total flight time possible (theoretically)

$$Flight\ time\ (minutes) = \frac{3.367 - 0.250}{0.501} = 6\ mins\ 13\ sec$$

Considering total 1 min time duration for obstacles and speed reductions due to curves:

$$Average\ Drone\ Speed = \frac{200}{313} = 0.64\ m/s$$

7.10 Sensor Connection

This section carries out the validation for the new Requirement ID 13 in Chapter 2, which defines proper connection and operation of the sensors together. The connections of the Ultrasonic sensor were carried out via UART protocol while the accelerometer and barometric sensor was done using I2C protocol.

Test Description

Make the sensor connections with the raspberry pi and run the code to observe the connection

Test Results

The results of the code in Appendix D produced results of the ultrasonic sensor and barometric sensor in an acceptable outcome as represented by the image below, thus validating proper connection has been done.

Table 17 - Sensor connection validation

	more than 50cm, Relative Altitude: 1.57 meters more than 50cm, Relative Altitude: 1.62 meters more than 50cm, Relative Altitude: 1.51 meters more than 50cm, Relative Altitude: 1.27 meters more than 50cm, Relative Altitude: 1.42 meters more than 50cm, Relative Altitude: 1.81 meters more than 50cm, Relative Altitude: 1.97 meters more than 50cm, Relative Altitude: 1.97 meters more than 50cm, Relative Altitude: 1.58 meters more than 50cm, Relative Altitude: 1.39 meters more than 50cm, Relative Altitude: 1.65 meters more than 50cm, Relative Altitude: 1.78 meters more than 50cm, Relative Altitude: 1.77 meters more than 50cm, Relative Altitude: 1.79 meters more than 50cm, Relative Altitude: 1.53 meters more than 50cm, Relative Altitude: 1.43 meters more than 50cm, Relative Altitude: 1.98 meters more than 50cm, Relative Altitude: 1.97 meters

7.11 Altitude Adjustment

This section carries out the validation for the new Requirement ID 14 in Chapter 2, which was supposed to determine and manage altitude adjustment. The code was tested to determine proper altitude determination using the BMP180 barometric sensor.

<u>Test Description</u>

The drone was placed at different measured altitudes to obtain the output from the barometric sensor which are represented below.

<u>Test Results</u>

Table 18 - Accelerometer validation results

Measured Height	Result of Code
	Relative Altitude: 0.96 meters Relative Altitude: 1.00 meters Relative Altitude: 0.86 meters Relative Altitude: 1.52 meters Relative Altitude: 1.38 meters Relative Altitude: 1.19 meters Relative Altitude: 1.41 meters Relative Altitude: 0.92 meters Relative Altitude: 1.53 meters Relative Altitude: 1.57 meters Relative Altitude: 1.62 meters Relative Altitude: 1.51 meters Relative Altitude: 1.27 meters Relative Altitude: 1.42 meters Relative Altitude: 1.81 meters Relative Altitude: 1.97 meters Relative Altitude: 1.97 meters Relative Altitude: 1.58 meters Relative Altitude: 1.39 meters Relative Altitude: 1.05 meters Relative Altitude: 1.78 meters Relative Altitude: 1.77 meters Relative Altitude: 1.79 meters Relative Altitude: 1.53 meters Relative Altitude: 1.43 meters Relative Altitude: 1.98 meters Relative Altitude: 1.97 meters Relative Altitude: 2.07 meters Relative Altitude: 1.69 meters Relative Altitude: 1.74 meters Relative Altitude: 1.75 meters Relative Altitude: 1.94 meters
	Relative Altitude: 1.81 meters Relative Altitude: 1.97 meters Relative Altitude: 1.97 meters Relative Altitude: 1.58 meters Relative Altitude: 1.39 meters Relative Altitude: 1.05 meters Relative Altitude: 1.78 meters Relative Altitude: 1.77 meters Relative Altitude: 1.79 meters Relative Altitude: 1.53 meters Relative Altitude: 1.43 meters Relative Altitude: 1.98 meters Relative Altitude: 1.97 meters Relative Altitude: 2.07 meters Relative Altitude: 1.69 meters Relative Altitude: 1.74 meters Relative Altitude: 1.75 meters Relative Altitude: 1.94 meters Relative Altitude: 1.80 meters Relative Altitude: 1.62 meters Relative Altitude: 1.93 meters Relative Altitude: 1.84 meters Relative Altitude: 1.93 meters Relative Altitude: 2.30 meters Relative Altitude: 2.05 meters Relative Altitude: 2.27 meters Relative Altitude: 1.99 meters Relative Altitude: 1.85 meters

As can be observed, the altitude measured by the barometric sensor varies heavily with an approximate variation of $\pm$ 1m. As recommended previously, the tests are proposed to be carried out using the DSP310 sensor for more accurate results. Furthermore, tests proposed in the recommendation section is to be carried out after successful integration of the drone to test altitude changes from a UI input given.

7.12 Different Speed Options

This section carries out the validation for the new Requirement ID 15 in Chapter 2.

Test Description

The flight controller was held in hand and run to test the response of the accelerometer which are represented below

Test Results

The results were obtained but at a delayed response to the action. Example: during sudden pace changes the speed was increased after a few seconds likely due to the processing speed of the microcontroller

7.13 Software and Hardware Restrictions

This section carried out the validation for the new Requirement ID 16 in Chapter 2

Test Description

Ensuring that only open source software and used for the project and DJI technology is not used for the project completion

Test Results

The following project utilized the following 3 software; Thonny IDE, Visualstudio and Beta Flight all of which are known to be open-source software free to use as represented below:

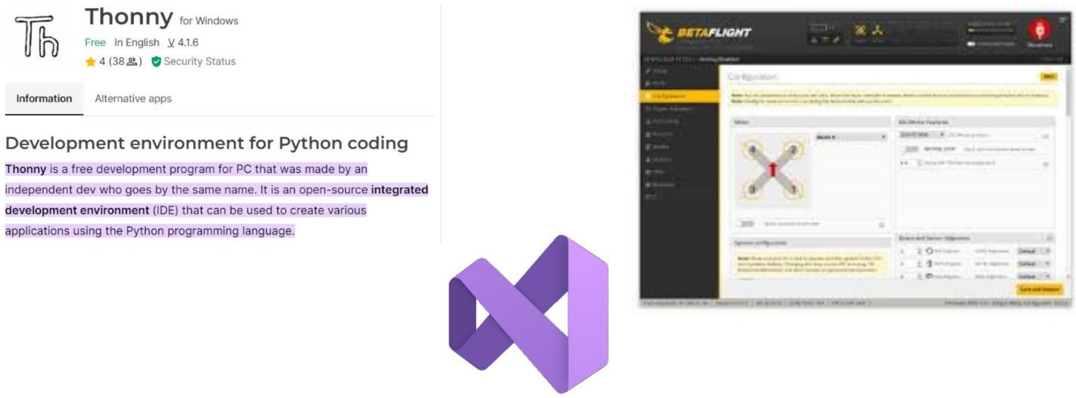


Figure 45 - Representation of used Open CV Software

7.14 Battery Charger

This section carried out the validation for the new Requirement ID 17 in Chapter 2

<u>Test Description</u>

The selected battery charger was used to observe using a multimeter to work and the charger is compatible with the 2S LiPo battery with a universal USB connection

<u>Test Results</u>

The following output from the battery charger after a completed battery charging is as follows

LED Indicator	Charger Status
LED stays green and flashes red.	The charger is ready to charge.
LED glows constant red.	The charger is charging.
LED glows constant green.	The charging process finishes.

Figure 46 - Battery Charger operation validation

7.15 Application Interface

This section involves the designing of a UI to validate the new Requirement ID 18 in Chapter 2

Test Description

Carry out different tasks in the UI to observe code changes. Actual implementation of information was not possible as the drone was not integrated to properly follow the line following

Test Results

The UI created was observed to be very user-friendly and carried out delayed functioning due to lack of microcontroller speed. The UI is as follows:

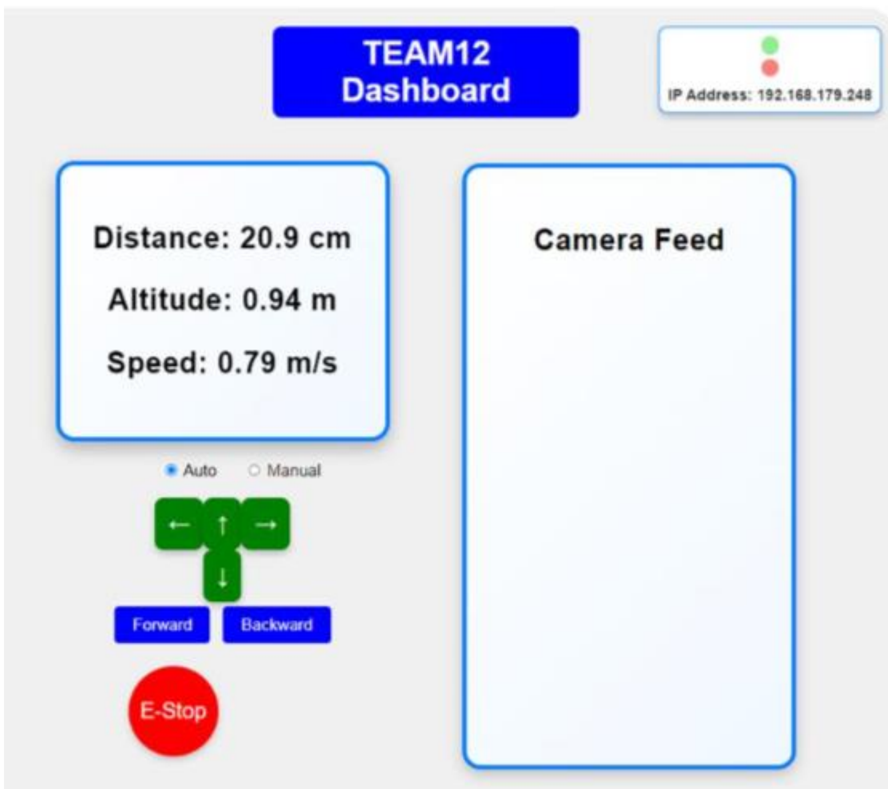


Figure 47 - User Interface operations and options

7.16 Drone Lifespan

This section involves the critical component selection to ensure lasting of the drone for at least 1 year which validates the new Requirement ID 19 in Chapter 2

Test Description

Determining the company warranty provided for each component as represented in the results

Test Results

All key components were observed to have a minimum of 12 months, with their lifetime expectancies ranging up to 10+ years at times. The results are as follows:

Table 19 - Components warranty and lifespan analysis

Component	Warranty	Life Time
Ultrasonic Sensor	12 m	5-7 years
Barometric pressure sensor	12 m	3-5 years
Raspberry PI zero W	12 m	10+ years
3 Axis Compass	12 m	3-5 years
Jumper Leads		
Pin header		
header pin 40 way		
5MP camera	12 m	3-5 years
Flight Controller	12 m	10+ years
Brushless Motor	12 m	3-5 years
Brushless Motor	12 m	3-5 years
Lipo Battery	12 m	10-17 months
Lipo Battery Charger	12 m	10-17 months
NewBee Drone	12 m	7+ years
Propellers	12 m	1+ years

7.17 Drone frame constraint

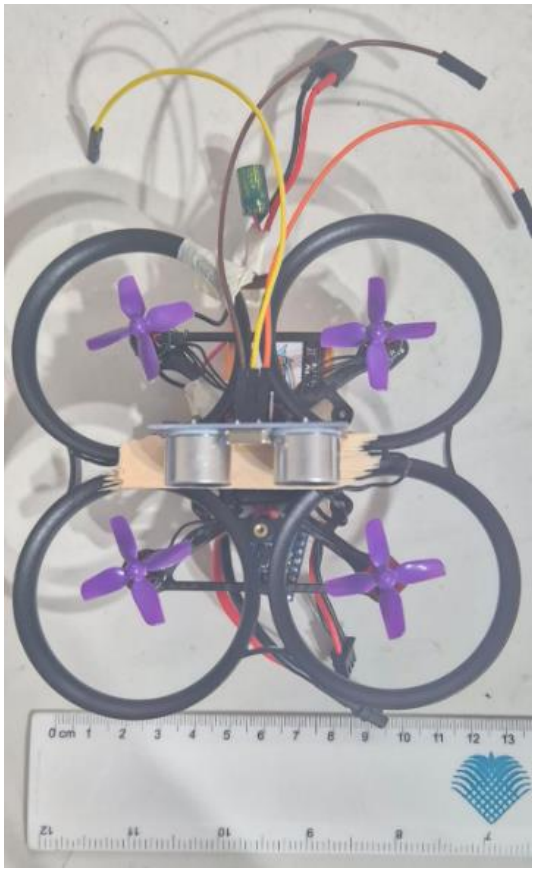
This section carries out the validation for the new Requirement ID 20 in Chapter 2.

Test Description

Measurement of the finally integrated drone size to be between 10 to 30 cm

Test Results

The length and width of the final drone was measured to be 13cm x 13cm with falls well within the range of the required design



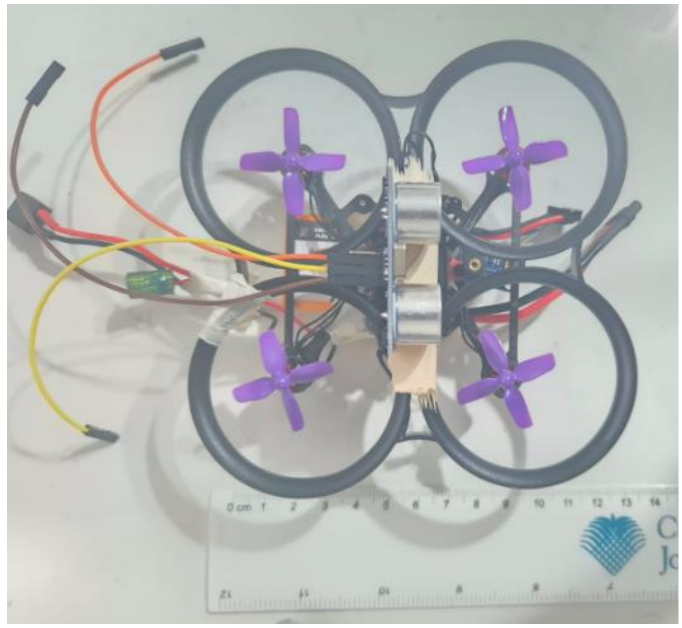


Figure 48 - Drone Frame Size validation

7.18 Summary of Validation and Verification

The table below highlights the summary of validation and verification

Table 20 - Validation and Verification Summary

New Requirement ID	Test Related Chapter	Tested (Y/N)	Remarks
1	7.1	**Y**	**Used propellers for the protection**
2	**7.2**	**Y**	**Provide a manual for safety usage.**
3	**7.3**	**Y**	**Successfully follow the path smoothly, even in sharp turns, and 90, 45, and 180 degrees.**

4	**7.3**	**Y**	**Successfully implemented the image processing code.**
5	**7.3**	**Y**	**Successfully detect lines in different lighting, and floor conditions.**
6	**7.3**	**Y**	**Successfully detected the midpoint.**
7	**7.4**	**N**	**Failed due to the integration issue. Future recommendations suggested**
8	**7.5**	**N**	**Future recommendations suggested**
9	**7.6**	**Y**	**Successfully detected the obstacle**
10	**7.7**	**Y**	**Achieved the project within the budget**
11	**7.8**	**Y**	**Completed laboratory induction**
12	**7.9**	**N**	**Future recommendations suggested**
13	**7.10**	**Y**	**Connection was done**
14	**7.11**	**Y**	**Successfully done.**
15	**7.12**	**N**	**Future recommendations suggested**
16	**7.13**	**Y**	**Used OpenCV software**
17	**7.14**	**Y**	**Obtained a compatible charger**
18	**7.15**	**Y**	**Successfully achieved**
19	**7.16**	**Y**	**Verified lifespan of more than 1 year.**
20	**7.20**	**Y**	**Used a frame within the range.**

8. Stakeholder Engagement

Stakeholder engagement of the following project mainly included the communications between the UWAAL, UWA and the team for generic information clarifications throughout the project lifetime. Client liaising was done in the form of technical questions which were submitted at the end of every week with responses being provided by Mr. Jega Gurusamy. The technical questions raised by Team 12 and the corresponding responses are as follows [30]:

Table 21 - Technical Queries raised by Team 12

ID	Technical Query	Answer
50	Do we need to start our drone behind the starting point or just from where the line starts?	For the line following, start from the line is acceptable. But hovering there will be no line.
70	Could you please provide the total flight time for the drone and confirm whether the final demonstration involves a journey of 200 meters?	Yes, it involves a hovering test and a separate 200m journey. Also, depends on the "safe speed" that you have determined based on your safety assessment.
71	Do we need to consider the cost of the controller? It's understood that the cost of a phone or a laptop when used for manual control will not be considered as part of the budget, so can we use a controller like that and not include it in the budget?	If you are planning to use a proper drone controller, it should be included in the budget as they are not available in the lab. Only laptop or mobile phone should not be included in the budget.
72	As long as battery is below 6Wh (e.g. 2s 800 mAh: 2*3.7*0.8=5.92Wh) that mentioned in previous tqs Can we use 4*3.7*0.4=5.92Wh battery system for our project? It is below 6Wh. Please provide recommendations on this	It may be allowed, but please remember that there is a second requirement. It has to be charged with a simple, commercial USB charger. That means USB charger which you can plug and forget (no adjustments of number of cells, charge

	matter	current etc). I haven't seen these chargers for batteries larger than 3s.
73	In manual mode, is it necessary for the remote button to allow for incremental changes in altitude (e.g., exact increments of 0.5m, 1m, 1.5m, 2m), or is it sufficient to provide gradual changes within ranges (e.g., 0.5m to 1m, 1m to 2m, or 2m to 3m)?	It can be pre-programmed, e.g. we can set 0.8m flight altitude before the flight, and it should keep this level whole time. We should be able to set any value within the range. You don't need to implement the changes during the flight itself.

The TQ sessions were essential as they helped the team keep aware of the requirements and obtain clarification on doubts regarding the specifications, while adhering to any dynamic change requests of the client.

Interactions and communications between potential vendors were also essential in determining compatibility to ensuring careful component selection as well as negotiating the best possible prices for a project completion within the budget including Buzz FPV and Altronics.

Furthermore, valuable interactions for 8 out the 12 weeks with Mr. Greg Crebbin were key to reassure progress as well as obtain incites in relation to similar projects carried out in a professional level. Additionally, valuable input from guest lecturers from their shared experiences and feedback provided by Mr. Jega Gurusamy who acted as the preliminary design review facilitator were essential for the project completion as well as for the future endeavors.

Table 22 - Stakeholders associated with the design project (UWA Staff)

Lecturer	Discussion Area
Mr. Jega Gurusamy	Conducting the laboratory safety induction session for the Lycopodium Laboratory. Furthermore, suggestions provided regarding the addition of the E-stop button and focus on pressure sensors for altitude measurement and hovering were taken as key points for the project
Mr. Greg Crebbin	Supervisor and Facilitator at the critical design review provided us with advise on potentially testing out the individual

	components prior to final integration to ensure that verification and validation is carried out efficiently
Mr. Paul Grainger	Carrying out the guest lecture regarding Professionals Australia which was information for the students, as potential future engineers of Australia
Prof. Tim Mazzarol	Carrying out a guest lecture regarding the commercialization of technology
Mr. Doug Hamilton	Shared insights regarding instrument designing carried out at CSBP
Mr. Nic (Mechnical Engineering UWA)	For the assistance in the induction process for the Maker's lab for soldering acted as an important part of the project while maintaining safety procedures

9. Safety Issues

As an integral part identified in the design philosophy, safety was a major concern while the design project of the indoor drone was being carried out. The laboratory safety inductions carried out regarding safety practices and Makers lab induction for soldering were essential factors in ensure proper safety issues were carried out in the design, However, an in-depth safety analysis carried out in Week 3 which was updated continuously throughout the project, the identified the critical safety issues as follows:

Table 23 - Safety Issues identification and mitigation analysis

Safety Issue	Description	Mitigation Method
Collision with objects or people while operating	Any sudden obstacles such as a running person could come collide with the drone causing damages.	Implementation of the ultrasonic sensor for obstacle detection within 50cm has been carried out for a field of vision facing forwards of 60% approximately. Furthermore, the demonstration is proposed to be carried out in a controlled environment without any dynamic obstacles appears and with clear warning signs to avoid any injuries to bystanders caused by collisions
LiPo battery potential bursts and electric shocks	LiPo batteries are designed such that any short circuits could result in the battery bursting causing injuries to people	Battery is properly encased and a separate connector is used to connect the battery with the flight controller which can be removed if required. The LiPo battery was also stored in a separate bag designed for LiPo batteries to prevent the battery wires from contacting each other to cause short-circuits. All testing was to be carried out with proper PPE including gloves, safety rubber boots and face shield.
Cuts and burns during design	The motors used in the project rotates at speeds	Proper laboratory induction and Maker's laboratory inductions have been successfully

implementation	exceeding 1500 rpm which can cause sharp cuts and injuries during testing. Soldering played a crucial part in battery connection and FC connection. Possible contact the high temperature soldering iron could cause third degree burns	completed by team to ensure proper safety procedures are followed. Wearing proper safety equipment (PPE) such as gloves and face shields can mitigte any damages or injuries caused by cuts or burns.
Loss of drone control while operating	Any errors in image processing or integration could cause the drone to identify a different source of “white” color and proceed out of the line path.	Image processing code has been implemented with a 3x3 matrix to ensure drone positioning is adjusted to the midpoint detected. Furthermore, the UI is developed with an E-stop which is planned to override the normal line following code and safely land on spot. Additionally, a remote mode option is also available in the UI for the user to receive access to manually control the drone to a safe position
Fires	Misplacement of hot equipment such as soldering irons could lead to fires considering the presence of carpeted floors.	The availability of fire extinguishers should be ensured prior to start of soldering, assurance of a clear pathways to an emergency exit should be established. A safety officer can be appointed to ensure any misplaced equipment are removed. Working in minimum pairs would also mitigate any risks of mistaken placement of hot items such as soldering irons.

10. Ethical Issues

In considering ethical aspects related to the project design the following sections needs to be analyzed; environmental impacts, potential dual-usage, meeting data privacy standards and accountability.

10.1 Environmental Impacts

Although the drone developed is small in scale, the manufacturing, operating and eventually disposal of the drone has traces of environmental impacts associated with them. These includes improper disposal of electrical components such as LiPo batteries, sensors, wires and polythene wraps (which components are packed during purchasing). These can cause result in hazardous impacts on the environment which needs to be always taken into consideration during any design project as maintaining a carbon footprint as infinitesimal as possible provides good recognition for both project while also ensuring environmental protection [31].

Methods used for mitigating environmental impacts includes the recycling of waste resulting from all stages of the project to their respective and disposing them in recycling stations. Ensure the selection of long-life time design elements which can be reusable for at least 1 year to reduce the amount of waste produced by the project within its lifetime.

10.2 Concerns of Dual-usage

The indoor line following drone has been developed with the purpose of being utilized for safety inspections of the Australian National Fabrications Facility (ANFF). Therefore, any misuse of the drone for personal or unethical purposes, such as unauthorized surveillance or intrusions of restricted spaces should be prevented .

Therefore, as a measure the access and the use have to be restricted between UWA and ANFF personnel with limitations for unauthorized users. An end-user agreement needs to be signed to ensure the client would not use the technology for any unethical premises protecting and for UWA to not provide the technology to any third party to prevent both parties from exploiting unethical opportunities [32].

10.3 Data Privacy and Prohibited Design Elements

The indoor line following drone utilizes image processing software which could capture images or videos of sensitive or private data. Therefore, proper regulatory laws need to be followed such as the General Data Protection Regulation which requires consent from people nearby to utilize the drone for autonomous indoor activity, the people should have the right to request the removal of data obtained within reasonable justifications to prevent and avoid and release of private data. Additionally, the project prohibits the use of DJI technologies, therefore analysis of DJI frameworks for the proposed design needs to be avoided to prevent any contractual conflicts in design handover [32].

10.4 Accountability

Any potential damages or harms caused after the submission of the finalized drone needs to be addressed based on the negligence of the user and the negligence of the designer. Any malfunctions caused due to technical failures in integration have to be accounted for by the design team, while any harms caused due to misuse or improper guideline following results in the end-user’s accountability [33].

Proper testing and validation have to be conducted by the design team with approvals from the client team’s technical coordinator to ensure all standards were met. The design team should also provide a detailed manual on how to operate the autonomous drone with the do’s and don’ts related to the model.

11. Risk Management

Risk management has been carried throughout the 12 weeks of the design project from which the following 5 risks and mitigation measures (technical and general) are as follows:

Table 24 - Risk Assessment - Top 5 Risks

Risk Identification	**Risk Assessment**	**Risk Mitigation**	**Risk Re-evaluation**
Failure for overall system integration and high expectations for final output	Likelihood – Likely Consequence – Major Risk Rating – Extreme (E1)	Ensure the final product is viable with at least majority of the requirements being full-filled. Careful verification and validation for working components to further optimize them. More time spend for the integration research and testing	Likelihood – Possible Consequence – Moderate Risk Rating – Moderate (Mo2)
Delays in purchased material / supply chain issues	Likelihood – Possible Consequence – Moderate Risk Rating – Moderate (Mo2)	All components selected were obtained from locally available stores (Wangara, Perth City) to ensure there are no delays or damages in component delivery	Likelihood – Rare Consequence – Moderate Risk Rating – Minor (Mi5)
Generic information and dynamic changes resulting in a miss design	Likelihood – Possible Consequence – Major Risk Rating – Major (Maj2)	The technical queries were carefully followed to ensure the design team remains updated and doubtful specifications to be verified through weekly TQs from Mr. Jega Gurusamy	Likelihood – Unlikely Consequence – Minor Risk Rating – Minor (Mi3)

Project Teams clashing schedules and focus required to other assessments and examinations	Likelihood – Almost Certain Consequence – Moderate Risk Rating – Major (Maj1)	Proper Gantt chart was developed in Week 2 itself with proper task and resource allocations. Deadline monitoring was carried to ensure the project progressed smoothly. Mid-weekly meetings and weekly scheduled meetings were essentials in discussing progress, issues and emphatically dealing with them to ensure the team morale and trust remained high	Likelihood – Possible Consequence – Moderate Risk Rating – Moderate (Mo2)
Bias in image processing caused due to lighting conditions, causing the drone to stray from the path causing a collision	Likelihood – Possible Consequence – Major Risk Rating – Major (Maj2)	Improving edge detection algorithm through the implementation of some filters. Carry out tests under different lighting conditions to adjust hue saturation accordingly. Carry out tests with proper warnings in the surroundings to prevent any human injuries due to collisions	Likelihood – Unlikely Consequence – Moderate Risk Rating – Minor (Mi6)

12. Construction / Assembling Process

The construction process of the drone is represented step by step below:

Step 1 – The drone frame components were unboxed and arranged separately with screws belonging to the frame top plate, mounting plate and battery bracket as follows:

Figure 49 - Unboxed components of the drone frame

Step 2 – The propellers were fitted on to the motors and fixed to the mounting plate. Note: the motors should be arranged in a clockwise, anti-clockwise, clockwise and anti-clockwise arrangement (denoted by the red mark in the connection cable).

Figure 50 - Connection of motors to the drone frame

Please note that the initially purchased motors were compatible with the Newbee Drone Cockroach V3. Therefore, the motor holes were not aligned with the drone mounting plate holes. Therefore, the team pivoted the approach by using copper strings for tight motor connection to the drone frame and also utilization of super-glue placed to ensure motor operations are not impacted.

Step 3 – The rubber grommets were inserted onto the 4 holes of the flight controller. The FC was then placed on the center of the mounting plate and tightened using the respective screws. The motor wires were then connected onto the flight controller as mentioned in the arrangement of the Step 2.

Figure 51 - Connection of the FC to the mounting plate

Step 4 – The top plate of the drone frame was placed on the above structure and tightened with the 4 respective screws to produce a result as represented below.

Figure 52 - Connection of the drone top plate to the mounting plate

Step 5 – The battery was placed beneath the center of the mounting plate in a way such that the battery connection with the external wire soldered to the flight control can be made when required for operation

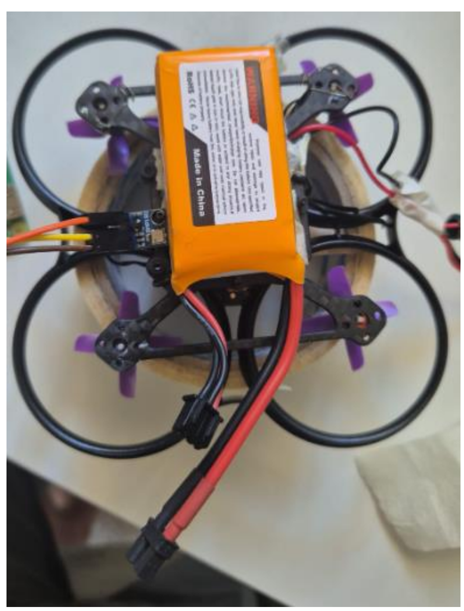


Figure 53 - Battery connection to the drone

Step 6 – The Raspberry Pi Zero W was placed beneath the battery and attached using tape in a sturdy manner.

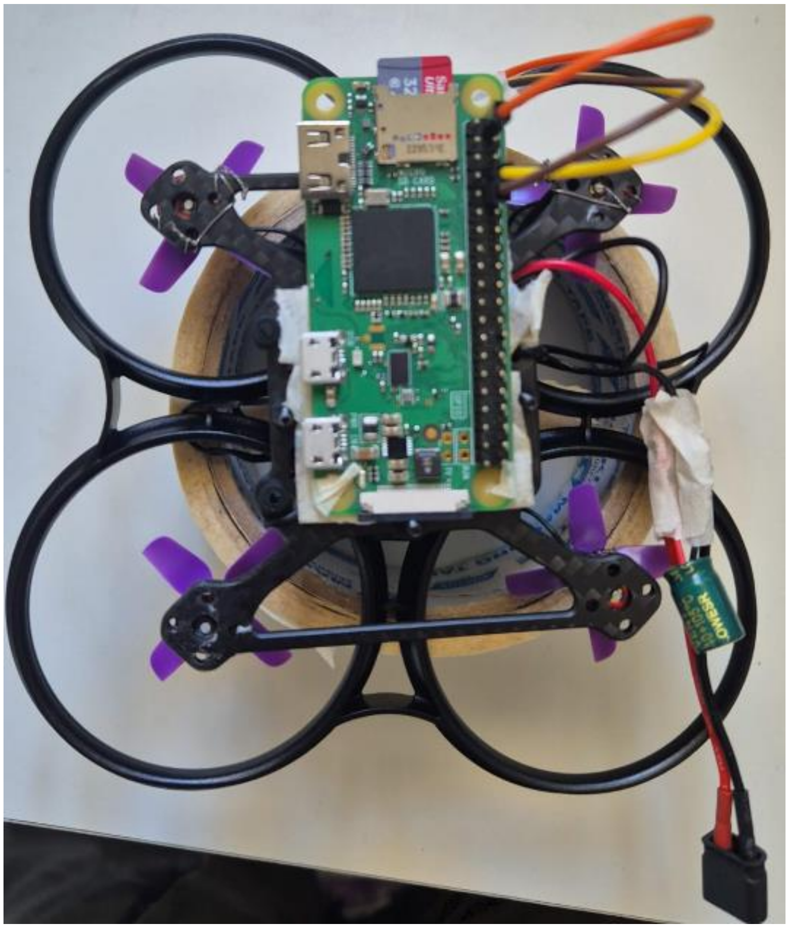


Figure 54 - MC installation to the drone

Step 7 – The accelerometer LSM303 in the front-bottom of the mounting plate with connections made as follows:

Table 25 - Pin Configuration for LSM303

Sensor Pin	Raspberry Pi Pin
Vin	2
SDA	3
SCL	5
Ground	6

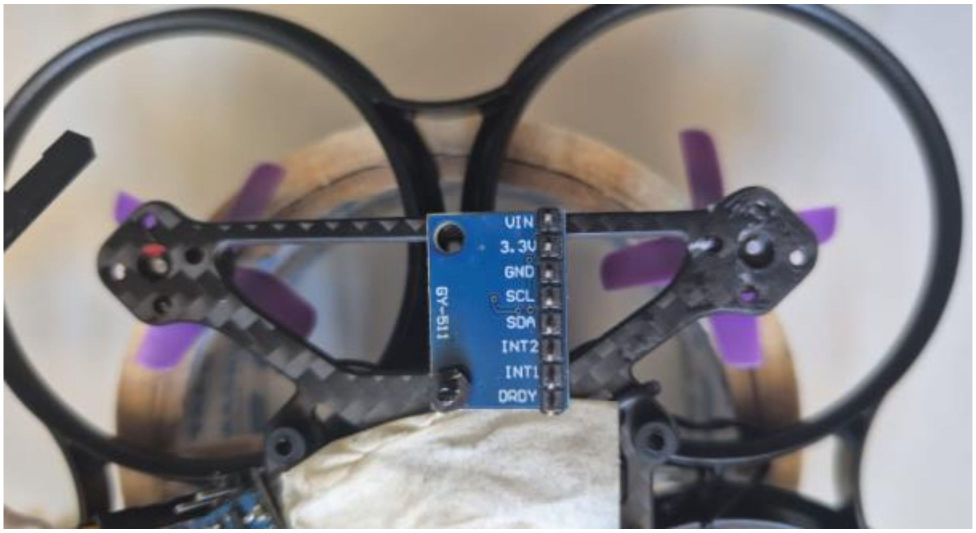


Figure 55 - Accelerometer connection to the drone

Step 8 – The barometric sensor, BMP180, was screwed onto the left side center of the mounting plate with pin connections as follows:

Table 26 - Pin Configuration for BMP180

Sensor Pin	Raspberry Pi Pin
Vin	2
SDA	27
SCL	28
Ground	6

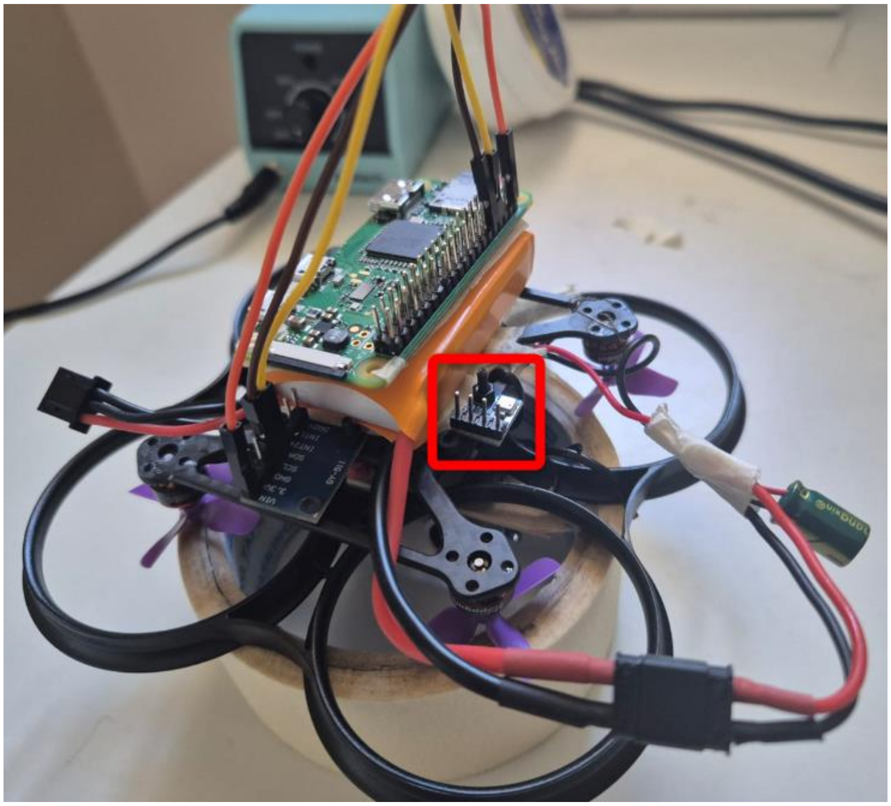


Figure 56 - Barometric sensor connection the drone

Step 9 – The Ultrasonic sensor, HC-SR04, was connected on the center of the top plate with connections made as follows:

Table 27 - Pin Configuration for HC-SR04 sensor

Sensor Pin	Raspberry Pi Pin
Vin	2
TRIG	8
ECHO	10
Ground	6

Figure 57 - Ultrasonic Sensor installation to the drone

Step 10 – The camera module was fixed below the raspberry pi at the center of the drone with video capturing occurring directly towards the ground.

Finally, the constructed and assembled indoor line following drone model is as follows:

Figure 58 - Final Drone design of Team 12

13. Design Outputs

The design output section highlights the team's final design. The image processing algorithms highlight the line detection and line following Python codes with all their functions. Accelerometer, Ultrasonic sensor, and pressure sensor Python codes show the sensor working results.

13.1 Programming Codes

The Python codes used for the system is available in the team channel: Team12 > Final Report and Presentation > Team 12 – Python Code

Table 28 - Pathway for accessing the python codes

Subfolder	Document Description
Line Following	(grid 3.py) Image Processing Code
Ultrasonic and Pressure	(Ultra + pres.py) Ultrasonic and Pressure Sensor Code
Accelerometer	(Accel_Final_2.py) Accelerometer Code
Hover	(Hover.py) Hovering Code
User Interface	(GUI.py) UI Code
Integration	(Integration.py) Integration between microcontroller and flight controller

13.2 Technical Datasheets

The datasheets were collected when sourcing the components. These datasheets were used to evaluate the potential options against the required specifications.

The technical datasheets are available in the project channels: Team12 > Final Report and Presentation > Team 12 – Datasheets

Table 29 - Technical documents related to the project

Document Name	Document Description
GY511 / LSM303	Accelerometer Datasheet
HCSR04	Ultrasonic Sensor Datasheet
BMP180	Pressure Sensor Datasheet
Raspberry Pi	Microcontroller Datasheet
Raspberry Pi V2	Camera Datasheet
MAX481	Drone Frame Datasheet
Zeus aio	Flight Controller Datasheet
LiPo Battery 2S	Battery Datasheet

14. Cost of Design

The following project was carried out with a strict budget of AUD 350. The table below provides a detailed analysis of the total budget spent for the project. Note: A LiPo battery bag priced at AUD 12.9 were purchased as a safety measure for our purposes but is not part of the final design. Therefore, it is not included in the finalized cost of the design

Table 30 - Finalized Budget of the Team 12 Drone Project

Component	Quantity	Price (AUD)	Store
Ultrasonic Sensor	1	14.3	Buzz FPV
Barometric pressure sensor	1	15.75	Altronics
Raspberry PI zero W	1	49.95	Altronics
3 Axis Compass	1	15.75	Altronics
Jumper Leads	1(30pcs)	4.65	Altronics
Pin header	1	2	Altronics
Header pin 40 way	1	1.1	Altronics
5MP camera	1	36.25	Altronics
Flight Controller	1	72.95	Altronics
Brushless Motor (Anti)	2	26.4	Buzz FPV
Brushless Motor (Clock)	2	26.4	Buzz FPV
Lipo Battery	1	11.97	Buzz FPV
Lipo Battery Charger	1	35	Altronics
NewBee Drone	1	20.89	Altronics
Propellers	1	13.87	Buzz FPV
Total Cost of the Project		**347.23**	

15. Recommendations for Future Works

15.1 Processes and Approvals

15.1.1 Drawing Approvals

In future work, detailed schematics for the drone assembly and subsystems (image processing components, microcontrollers, sensors, power) should be submitted for approval. These technical drawings must comply with aviation safety standards and be approved by an authorised engineer or authority. This process ensures that every component aligns with safety protocols and facilitates smoother assembly. Any design revisions should be well-documented during testing and implementation.

15.1.2 Tenders

The next step involves identifying vendors for necessary components through a formal tendering process. High-quality suppliers for components such as motors, camera, microcontroller, flight controllers, sensors, batteries, and communication modules. The tender documentation should specify specification of the components, reliability standards, lead time and compatibility with the existing flight controller Zeus5 AIO. Open tenders will allow for cost-effective procurement while ensuring that the equipment meets project requirements.

15.1.3 Drone Assembly and Testing

After procurement, the assembly line for drone shall be started. Before the starting of this phase, Job Safety Analysis must be carried out with skilled operators for soldering and assembly of the system. The Stage 1 of assembly phase will focus on integrating the FC with the battery leads, UART ports for connecting with the microcontroller (later stage) and motor plugs. Stage 2 shall consist of integrating ultrasonic sensors, camera, barometer and other related modules to the microcontroller. In Stage 3 the FC and microcontroller shall be connected as per the production drawings. Stage 4 shall consist of fixing the motors to the frame base and integrating the FC &

MC to the frame. Stage 5 shall complete the connections between motor and the FC. Battery shall be supplied separately with the packaging. Initial testing will be conducted in a controlled indoor environment to verify that all systems work as intended to. Iterative testing cycles shall be carried out for performance testing and lifecycle testing.

15.1.4 Final Product Assembly and Testing

After the drone assembly, each drone will undergo final elaborate testing phases. At this stage the Factory acceptance Tests (FAT) for the drone's functionalities including navigation, obstacle avoidance, line detection and autonomous operation will be tested based on the FAT documents. Testing shall assess the drone's performance under various lighting conditions, floor surfaces and potential electromagnetic interference from other equipment's.

15.1.5 Regulatory Approvals

The AS/NZS and ISO standards shall be followed as it is crucial getting approval for compliance of the product for commercial usage. This requires the product to follow to legal, safety and quality standards like ISO 12100 for machinery safety, AS/NZS 3000 for electrical safety and AS/NZS 61000 for electromagnetic compatibility. The drone must obtain the Regulatory Compliance Mark (RCM) for commercial sale in Australia. Involvement of a third-party regulatory authority for obtaining RCM shall increase the product's quality and safety assurance for consumers.

15.2 Build Testing Recommendations

Table 31 - Proposed test for flight controller configurations

Test no.	T001
Specification	Flight controller configurations
Description	To check the FC firmware version, to enable communication ports and to calibrate the inbuilt sensors like gyroscope and accelerometer.
Result Analysis	Results shall confirm the firmware version, configuration of communications ports, calibrations of the inbuilt sensors, Wi-Fi module configurations for further drone testings.

Table 32 - Proposed test for propulsion and lift testing

Test no.	T002
Specification	Propulsion and Lift
Description	Measure the drone's ability to achieve stable lift using the selected motors and 2S LiPo battery.
Result Analysis	Results will confirm whether the drone can sustain altitude in a lab environment, lifting capabilities and efficiency in power consumption.

Table 33 - Proposed test for Communication reliability

Test no.	T003
Specification	Communication Reliability
Description	Test the reliability and range of the Wi-Fi module between the drone and controller.
Result Analysis	The test will ensure no signal dropouts during operation especially in environments with high interference offering insight into any adjustments required.

Table 34 - Proposed test for Battery life and power managemetn

Test no.	T004
Specification	Battery Life and Power Management
Description	Test the endurance of the drone with a 7.4V 650mAh 2S LiPo battery under full load conditions including propulsion and sensors.
Result Analysis	The results will provide data on power consumption rates, flight duration, recommendations for battery health and improvements if needed.

Table 35 - Test proposed for Navigation and Stabilization testing

Test no.	T005 - Test Related to Chapter 7.11
Specification	Navigation and Stabilization
Description	Evaluate the drone's hovering, stabilization at different heights and its response to sudden altitude changes.

Result Analysis	The outcome will validate the effectiveness of the altitude sensor and stabilization algorithms under various commands from the controller.

Table 36 - Proposed test for line-following accuracy testing

Test no.	**T006**
Specification	Line-Following Accuracy
Description	Test the camera onboard to determine accuracy in detection of white line paths in low and well-lit areas for automatic line detection.
Result Analysis	Analysis will focus on precision of image processing and the drone's ability to correct its path efficiently in autonomous mode.

Table 37 - Proposed test for Obstacle avoidance

Test no.	**T007**
Specification	Obstacle Avoidance
Description	Simulate a lab environment with obstacles and assess the drone’s response to detect and avoid them.
Result Analysis	The analysis will measure the drone's ultrasonic sensor’s ability to detect obstacles in real-time and microcontrollers processing to calculate the necessary response time for successful avoidance.

Table 38 - Proposed test for safe landing

Test no.	**T008**
Specification	Safe Landing / Fail safe landing
Description	To simulate a battery/ communication fail scenario to test the landing capabilities of the drone.

Result Analysis	The testing results shall determine the fail-safe landing of the drone to prevent damage during failure of battery/communication.

15.3 User Manual

A detailed user-manual for the utilization of the drone to be reviewed by end-users is provided in the Appendix 1.

16. Conclusion

The development of the autonomous indoor line-following drone has been partially completed addressing the project objectives and stakeholder requirements through a structured and comprehensive design process. The project began with a detailed requirement analysis, performing a risk and establishing project requirements from the client. These requirements were prioritized based on the criticality for this project with a focus on essential aspects such as autonomous path detection and navigation, obstacle avoidance, budget constraints and the use of open-source software. Camera integration to the drone for line tracking using image processing algorithm with the help of Raspberry Pi Pico microcontroller using Python script was the selected method of communication establishment for to determine the pathway for the drone to follow. The barometer sensor, coupled with the ultrasonic sensor is used for the stabilization and hovering of the flight. The ultrasonic sensor onboard is programmed for obstacle detection. The Zeus5 AIO 1-2S F411 is used as the Flight Controller in the design. The FC has inbuilt ESC for motor control, Wi-Fi module for wireless communication, Accelerometer for altitude hold and Gyroscope for flight stabilization. The FC is Beta Flight compatible. A 650 mAh 2S LiPo battery was selected to be used to power the drone depending on energy and budget limitations. The design was optimized to meet the project's majority of design requirements but excluding a few key requirements, while complying to the $350 budget constraint.

The systematic design methodology and testing has resulted in a drone prototype that is functional partially meeting the design criteria outlined in the initial project scope. The use of open-source software has facilitated for the development of algorithms for image processing for line detection and obstacle avoidance providing further scope for future improvements for improving the accuracy and sensitivity of the sensors for achieving higher precision in the autonomous mode of operation.

In conclusion, the autonomous indoor line-following drone has demonstrated the potential to meet the design requirements for autonomous indoor line tracking for safety inspections. The project addresses the technical and budgetary constraints while laying the groundwork for further improvements. Future work can build on this foundation by improving the precision of hovering, improving flight time and expanding the drone's ability to operate in complex environments.

17. References

[1] UWA, "ELEC5552 Design Project 2 - Project 1 Brief (Indoor line following drone)," University of Western Australia, Perth, 2024.

[2] D. Silva and J. Gurusamy, "Technical Queries Responses - LMS (ELEC5552) - Project 1 (Drone)," September 2024. [Online]. Available: https://uniwa-my.sharepoint.com/:x:/g/personal/00027162_uwa_edu_au/EYI_HSsUxG1BhB0-2OH9vMMBsHFifxsOKC6wML5iqNf8Jw?e=fVanbD. [Accessed October 2024].

[3] UWA, "ELEC5551/2 Final Report and Manual Guidelines," University of Western Australia, 2023.

[4] Team 12, "Requirement Analysis - Team 12," 2024.

[5] UWA, "ELEC5552-2024 Unit Schedule v1.3," July 2024. [Online]. Available: https://lms.uwa.edu.au/ultra/courses/_91855_1/cl/outline. [Accessed October 2024].

[6] Raspberry Pi Ltd, "Raspberry Pi Zero v1.3," Raspberry Pi Ltd, 2024.

[7] Raspberry Pi Ltd, "Raspberry Pi 4 Model B," Raspberry Pi (Trading) Ltd, 2024.

[8] Espressif, "ESP32 Series Datasheet Version 4.7," Espressif, 2024.

[9] NVIDIA, "NVIDIA Jetson Nano," NVIDIA, 2024. [Online]. Available: https://www.nvidia.com/en-au/autonomous-machines/embedded-systems/jetson-nano/product-development/. [Accessed October 2024].

[10] Elprocus, "Difference between ESP32 vs Raspberry Pi," Electronics | Projects | Focus, 2023. [Online]. Available: https://www.elprocus.com/difference-between-esp32-vs-raspberry-pi/. [Accessed October 2024].

[11] BetaFPV, "F405 4S 20A Toothpick Brushless Flight Controller V5 BLHeli_S (ICM42688)," BetaFPV, 2024. [Online]. Available: https://betafpv.com/products/f405-4s-20a-toothpick-brushless-flight-controller-v5-blheli_s-icm42688?srsltid=AfmBOoqpI8XFcgdVxLsRfmabf4IQaQed5PoYNTfNkeuouG-pkwJAx4WE. [Accessed October 2024].

[12] Buzz FPV, "HGLRC Zeus5 AIO 1-2S F411 Flight Controller 5A BL_S 4in1 ESC with WiFi Function," Buzz FPV, 2024. [Online]. Available: https://buzzfpv.com.au/products/hglrc-zeus5-aio-1-2s-f411-flight-controller-5a-bl_s-4in1-esc-with-wifi-function?srsltid=AfmBOorzHrJhec0gNpUdjnZSINKBuUi4hVglCRChNatQAZf1jYrE_l2c. [Accessed October 2024].

[13] Speedy Bee, "F4 AIO Flight Controller Ver 2.0," Speedy Bee, 2024. [Online]. Available: https://www.speedybee.com/speedy-bee-f4-aio-flight-controller/. [Accessed October 2024].

[14] Buzz FPV, "HappyModel Unibell EX0802 brushless motor 19000KV," Buzz FPV, 2024. [Online]. Available: https://buzzfpv.com.au/products/happymodel-unibell-ex0802-brushless-motor-19000kv?srsltid=AfmBOoqsnFUXt2bi0pQF5Odw-32joxsg8ptEa-rhCJo8Udsncdwj3yhm. [Accessed October 2024].

[15] Beta FPV, "1404 4500KV Brushless Motors," Beta FPV, 2024. [Online]. Available: https://betafpv.com/products/1404-4500kv-brushless-motors?variant=39279977562246. [Accessed October 2024].

[16] Beta FPV, "1506 3000KV Brushless Motors," Beta FPV, 2024. [Online]. Available: https://betafpv.com/products/1506-3000kv-brushless-motors?variant=31550947000454. [Accessed October 2024].

[17] Buzz FPV, "Betafpv X Gemfan 40mm 4-Blade Propellers (1.0mm Shaft)," Buzz FPV, 2024. [Online]. Available: https://buzzfpv.com.au/products/betafpv-x-gemfan-40mm-4-blade-propellers-1-0mm-shaft. [Accessed October 2024].

[18] Buzz FPV, "Rekon 3 Nano Long Range Frame," Buzz FPV, 2024. [Online]. Available: https://buzzfpv.com.au/products/rekon-3-nano-long-range-frame. [Accessed October 2024].

[19] Diatone, "DIATONE Taycan C25 MK2 Cinewhoop FrameKit," Diatone, 2024. [Online]. Available: https://www.diatone.us/products/diatone-taycan-c25-mk2-cinewhoop-framekit?srsltid=AfmBOoqY4ghj1PzsERUO-HkdzKKpEIaQX3IeQVIzoQgleYUCHDrIrgd7. [Accessed October 2024].

[20] Buzz FPV, "NewBeeDrone Cockroach V3 65mm Brushless Ultra Light Frame," Buzz FPV, 2024. [Online]. Available: https://buzzfpv.com.au/products/cockroach-brushless-

v3?variant=43280652337292. [Accessed October 2024].

[21] Buzz FPV, "Pavo20 Brushless Whoop Frame," Buzz FPV, 2024. [Online]. Available: https://buzzfpv.com.au/products/pavo20-brushless-whoop-frame. [Accessed October 2024].

[22] HGLRC, "Zeus5 AIO Flight Controller Manual," HGLRC, 2021.

[23] Happymodel, "Happymodel EX0802 New series brushless motor for Mobula6 Mobula6 HD," Happymodel, May 2020. [Online]. Available: https://www.happymodel.cn/index.php/2020/05/29/happymodel-ex0802-new-series-brushless-motor-for-mobula6-mobula6-hd/. [Accessed October 2024].

[24] SparkFun, "Ultrasonic Ranging Module HC - SR04," Elec Freaks, 2023.

[25] Bosch, "BMP180 Digital Pressue Sensor - Datasheet," Bosch Sensortec, 2013.

[26] STMicroelectronics, "LSM303AGR," STMicroelectronics, 2022.

[27] Altronics, "Raspberry Pi 5MP Fisheye Camera Module," Altronics, 2024. [Online]. Available: https://www.altronics.com.au/p/z6473-raspberry-pi-5mp-fisheye-camera-module/. [Accessed October 2024].

[28] Hobby Co, "BlackZon 540037 Battery Pack for Slayer/Slyder Liion 7 4V 800mAH w/ Dean Plug," Hobby Co, 2024. [Online]. Available: https://www.hobbyco.com.au/products/blackzon-540037-battery-pack-for-slayer-slyder-li-ion-7-4v-800mah-w-dean-plug?srsltid=AfmBOor1rEUGAKg4fw4I1R7ZgHsAn13e5ABxYA-QiXpZbDiAp9tg7Gyi. [Accessed October 2024].

[29] Buzz FPV, "CNHL MiniStar 650mAh 7.4V 2S 70C Lipo Battery with XT30U," Buzz FPV, 2024. [Online]. Available: https://buzzfpv.com.au/products/cnhl-ministar-650mah-7-4v-2s-70c-lipo-battery-with-xt30u?srsltid=AfmBOopAnyKhbc9QtcJBqPtoIUkJnWnFdCgDGXeCPr8qqYEb1WPFLvw8. [Accessed October 2024].

[30] Aus Electronics Direct, "240V AC Li-Po & Li-ion Balance Charger for 2s and 3s Battery packs B3," Aus Electronics Direct, 2024. [Online]. Available:

https://www.auselectronicsdirect.com.au/240v-ac-li-po-li-ion-balance-charger-for-2s-and-3s. [Accessed October 2024].

[31] Battery Mate, "iMax B6AC 80W RC Lipo NiMh Digital Battery Balance Charger Discharger Control," Battery Mate, 2024. [Online]. Available: https://www.batterymate.com.au/products/imax-b6ac-80w-rc-lipo-nimh-digital-battery-balance-charger-discharger-control?currency=AUD&variant=42823096008955&utm_source=google&utm_medium=cpc&utm_campaign=Google%20Shopping&stkn=fc9a6e4e1c3a&tw_source=google&tw. [Accessed October 2024].

[32] HobbiesDirect, "FMS 2S USB LiPo Quick Charger," HobbiesDirect, 2024. [Online]. Available: https://hobbiesdirect.com.au/products/fms-2s-usb-lipo-quick-charger-c2267-49127?gad_source=1&gclid=EAIaIQobChMIlIj15fWeiQMVm6lmAh0iBw4ZEAQYBCABEgKGXvD_BwE. [Accessed October 2024].

[33] Analog Devices, "ADXL335," Analog Devices, 2010.

[34] Sparkfun, "LSM303DLH," Sparkfun, 2009.

[35] Altronics, "MPU-6050 6 Axis Accelerometer Plus Gyro Breakout For Arduino," Altronics, 2024. [Online]. Available: https://www.altronics.com.au/p/z6324-mpu-6050-6-axis-accelerometer-gyro-breakout-for-arduino/. [Accessed October 2024].

[36] Altronics, "Ultrasonic Distance Sensor," Altronics, 2024. [Online]. Available: https://www.altronics.com.au/p/z6322-ultrasonic-distance-sensor-breakout-for-arduino/. [Accessed October 2024].

[37] Altronics, "Barometric Air Pressure Sensor," Altronics, 2024. [Online]. Available: https://www.altronics.com.au/p/z6395-barometric-air-pressure-sensor-breakout-for-arduino/. [Accessed 2024 October].

[38] J. Gurusamy, "Technical Queries Responses - LMS (ELEC5552)," August 2024. [Online]. Available: https://uniwa-my.sharepoint.com/:x:/g/personal/00027162_uwa_edu_au/EYI_HSsUxG1BhB0-2OH9vMMBsHFifxsOKC6wML5iqNf8Jw?e=fVanbD. [Accessed October 2024].

[39] C. M. Costa, J. Barbosa, R. Goncalves and H. Castro, "Recycling and environmental issues of lithium-ion batteries: Advances, challenges and opportunities," Energy Storage Materials, 2021.

[40] S. Sciancalepore, "Privacy and Confidentiality Issues in Drone Operations: Challenges and Road Ahead," IEEE Network, 2024.

[41] T. S. Goetze, "Mind the Gap: Autonomous Systems, the Responsibility Gap, and Moral Entanglement," Seoul, 2022.

[42] P. Biasetti and S. Grigoletto, "Insights toward an ethical assessment of CubeSat technologies," 2023.

[43] ACMA, "Citizen band radio stations class licence," Australian Government, 2024. [Online]. Available: https://www.acma.gov.au/licences/citizen-band-radio-stations-class-licence#:~:text=You%20can%20only%20use%20CB,476.4125%20to%20477.4125%20MHz%20(inclusive). [Accessed 20 May 2024].

18. Appendices

18.1 Appendix 1 – User Manual

INSTALLATION AND USER INSTRUCTIONS

Indoor Line Tracking Drone

Team 12

October 2024
Version 1.0

Table of Contents

Table of Contents

1 PREFACE

1.1 Description of the user

This manual is intended for operators and technicians responsible for the installation, operation, and maintenance of the automated line-following drone. Users should have basic knowledge of indoor drone navigation, electronic circuits and safety procedures related.

1.2 Explanation of safety warnings

⚠**DANGER!** Danger indicates a hazard with a high level of risk which if not avoided will result in death or serious injury.

⚠**WARNING!** Warning indicates a hazard with a medium level of risk which if not avoided could result in death or serious injury.

⚠**CAUTION!** Caution indicates a hazard with a low level of risk which if not avoided could result in minor or moderate injury.

ℹ**NOTICE** Indicates information considered important but not a hazard related.

1.3 Retaining instructions

Read and understand this manual and its safety instructions before using this product. Failure to do so can result in serious injury or death.

Follow all the instructions. This will avoid fire, explosions, electric shocks or other hazards that may result in damage to property and/or severe or fatal injuries.

Keep all safety information and instructions for future reference and pass them on to subsequent users of the product.

2 Description of the product

2.1 Purpose of the product

The automated indoor line tracking drone is designed to autonomously navigate along the paths identifying the line using image processing for safety inspections or other indoor applications. It uses a camera to detect lines and maintain courses without user intervention. The drone uses ultrasonic sensor for obstacle detection and Barometer for altitude correction. The microcontroller inside the drone co-ordinates the sensor data and communicates with the flight controller for flight movements.

2.2 Technical data

Parameter	Unit
Power Source	650mAh 7.4V 2S Lipo Battery
Size	
Weight	150g
Max Speed	1.5 m/s
Micro Controller	Raspberry Pi Pico W
Flight Controller	HGLRC Zeus5 AIO
Range	200m (indoor)
Flight Time	6 – 8 minutes

2.3 Product Compliance

The drone complies with applicable indoor flight safety standards and is designed to operate in confined and controlled environments. It meets the necessary certifications for electromagnetic compatibility and safe indoor use. The drone complies with AS/NZS 60335.1 for electrical safety and AS/NZS 62133 for lithium battery safety. The product ensures compliance with AS/NZS CISPR 32 to avoid electromagnetic interference. Follows AS/NZS 2107 to meet noise level standards for indoor environments. Adhere to UN38.3 for safe battery transportation and AS/NZS 62133-2 for lithium battery safety. All the standards mentioned above have been used from the supplier data sheets.

2.4 Product elements

2.5 Understanding the user interface

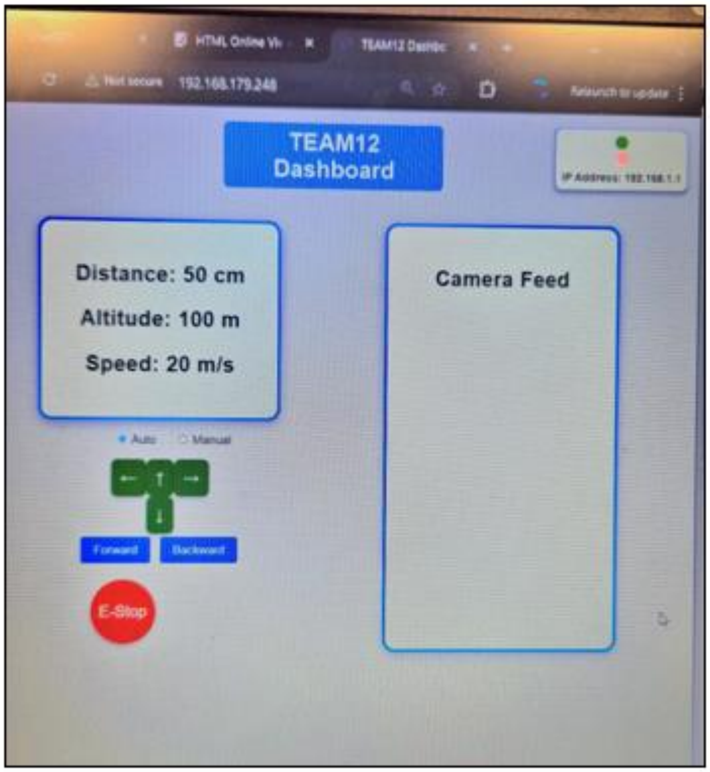


Figure 1: UI Dashboard.

No.	Name	Function
1	WiFi connection	Indicates if the device is connected to a WiFi network - best WiFi signal strength - bad WiFi signal strength - no WiFi connection
2	IP Address	The UI is not connected with the drone. The UI is not connected with the drone.
3	E-Stop	Emergency Stop Button
4	Auto / Manual	For choosing the mode of operation.
5	Arrow keys	For positioning the drone.
6	Forward/ Backward button	For controlling the drone for movement.
7	Camera Feed	For seeing the drone visuals.
8	Data Panel	For viewing the system dynamic data.

3 Safety instructions

⚠ **WARNING! The drone should only be operated by trained personnel. It is not recommended for children or individuals unfamiliar with drone safety.**

3.1 How to use the product safely

3.1.1 Personal safety

- Always wear safety goggles when operating the drone.
- Keep hands, hair and loose clothing away from propellers when the drone is in operation.
- Keep away from the drone when the motors are powered.

3.1.2 Work area safety

- Operate in well-lit areas.
- Ensure there are no people/animals around the flight path.
- Do not operate the drone in outdoor environments.

3.1.3 Electrical safety

- Always carry Lipo battery in the battery pouch provided.
- Only use the provided charger for the 2S LiPo battery.
- Do not expose the drone to moisture or extreme temperatures.

3.2 Graphical symbols

3.2.1 Explanation of graphical symbols in the user manual

Symbol	Meaning
⚠ **Warning**	Signals potential hazards.
ℹ Info	Offers additional operational guidance.

4 Preparation

4.1 How to transport and store the product

⚠ **Warning** Always ensure the drone is not powered on while transporting or storing.

4.1.1 Lifting, handing and transporting the product

To lift the product safely:

The drone shall be lifted by holding the frames between the motors using two fingers. The product can also be lifted by keeping it inside the palm of your hands.

⚠ **Warning** Do not touch the product with wet hands.

To handle the product safely:

Ensure the drone is kept on an even surface before flying or after landing.

To transport the product safely:

The drone should be transported in a secure case to avoid damage. Handle the drone with care especially when moving the delicate sensors and propellers.

4.1.2 Storing the product

Store the drone in a cool and dry location. Avoid exposure to direct sunlight or extreme temperatures that may affect battery life. Store the Lipo battery in the battery pouch provided.

4.2 How to install the product

4.2.1 Removal of the transport and packaging restraints

Carefully remove all packaging materials ensuring no sensors or delicate parts are damaged in the process. Keep the product away from wet areas.

4.2.2 Unpacking the product

Inside the box, you will find:

- The drone (pre-assembled)
- Battery and charger.
- Spare propellers.
- Micro USB cable for software updates.
- Battery pouch.
- User manual.

4.2.3 Conditions for assembling

Ensure the drone is assembled in a clean, well-lit environment free of dust and debris that could interfere with the sensors or flight.

4.2.4 Installation of the product

Step 1: Remove the packaging and arrange all the contents inside the box on a clean dry table.

Step 2: Ensure all the components inside the assembled drone are firm and tight the screws if loose.

Step 3: Attach the propellors to the motor shafts (as shown below) by gently pushing first and turning the propellors in clockwise direction later. Ensure the propellor is firmly attached to the shaft.

Figure 2: Motor shaft and the propellor.

Step 4: Attach the battery to the battery connector coming from the flight controller. Ensure the leads are connected according to the color of the wires. Ensure the same color leads are connected.

⚠ **Warning: Interchanging the leads can damage the battery as well as the onboard controllers.**

Step 5: Attach the battery to the holder provided on the drone.

Step 6: Ensure every component is firmly attached to the drone.

ℹ: **Do not alter the circuit or remove any part of the circuit for individual testing.**

4.3 How to commission the product

1. Connecting the battery shall power up the drone flight controller and microcontroller.
2. Once the flight controller boots up the Led starts blinking in red and blue color.
3. Connect the drone to the Beta flight configurator software and perform initial calibration of accelerometer.
4. The microcontroller onboard (Raspberry Pi Pico W) shall take a few minutes to boot up during initial configuration.
5. Connect to the drone using the User interface provided. Ensure the Wi-Fi connection between the drone and UI is established.
6. In the UI screen all the parameters of the drone shall be seen including the battery percentage, connection status, mode of operation, altitude and controls for manual operation.
7. Once all the above steps have been completed conduct a test flight in a safe and controlled environment in manual mode.

ℹ: **Press the Emergency stop in the UI in case of emergency for stopping the flight.**

5 Operation/Use

5.1 How to use the product

5.1.1 Operational environment

The drone is designed for indoor use. Ensure the space has good lighting and that the line it follows is clearly white in color for the camera to process the image. Avoid reflective floors.

5.1.2 Manual/Automatic operation

For safety purposes, always operate in manual mode initially before switching to autonomous mode. Use the controls in the UI to guide the drone manually. Once put into Auto mode the drone shall fly autonomously detecting the white line path and hovers when an obstacle or stop line is detected. Manual override controls can also be used in emergency situations.

5.1.3 Starting/Stopping the product's operation

To start the drone:

1. Place the drone on the line it will follow.
2. Power on the drone by releasing the Emergency stop button.
3. Verify sensor alignment and that the drone detects the line.
4. Once the drone detects the line, switch from manual mode to auto for autonomous operation.

To stop the drone, press the Emergency stop button again to make the drone to perform a safe landing.

5.1.4 Checks before using the product

Before each flight, perform the following checks:

1. **Battery Level:** Ensure the battery is fully charged and have no bulge on the surface.
2. **Propellers:** Check for any signs of damage.
3. **Camera:** Verify that the camera lens is clean and unobstructed.
4. **Controller Settings:** Check for accelerometer calibration using beta flight configurator.
5. **Line Clarity:** Ensure the path the drone will follow is white and have no reflections.

Warning: Do not use the battery if bulged.

5.1.5 Using the product

Once the drone is in autonomous, it shall follow the white line. Monitor the drone to ensure it is performing as expected. If the drone goes off-course or encounters any undetected obstacles, switch to manual control and make necessary adjustments to prevent any accidents.

5.2 Software documentation

5.2.1 Flight Controller Setting up

The initial setup of the drone includes calibrating the sensors and configuring flight parameters using the Beta flight configurator. Connect the drone to a computer via the micro-USB port on the flight controller and open the software and follow the calibration procedures for accelerometer and to check motor controls.

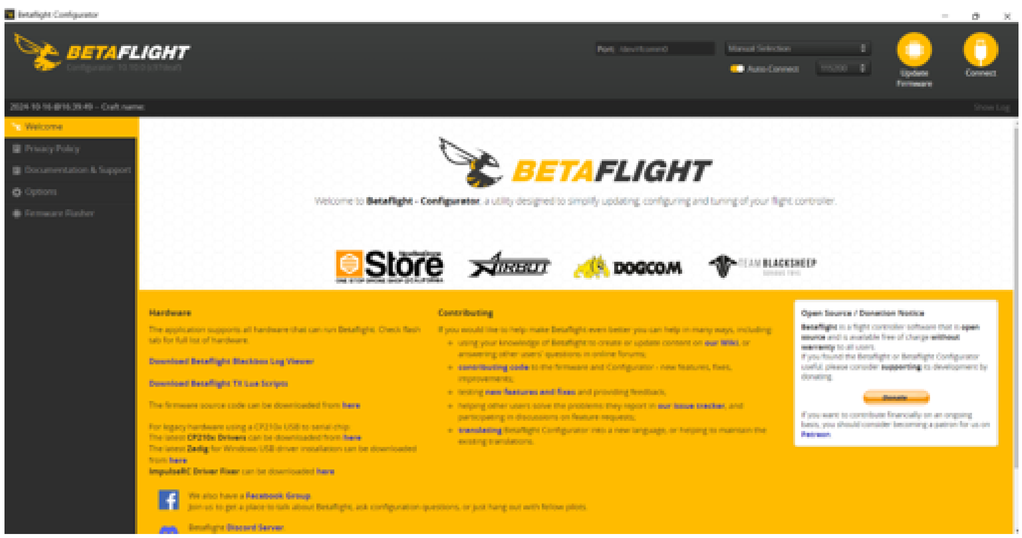


Figure 3: Beta flight configurator home page.

Connect the FC to the configurator. The figure below shows the home screen after connecting the FC. Ensure the drone is placed on a flat surface. Press calibrates the accelerometer and waits for the calibration to be completed. As shown in the second figure below, the working of motors can be tested together or individually by controlling the bars provided in the page.

Warning: Do not increase the speed beyond 1500r rpm during testing. If not followed shall burn the motor bearings.

Figure 4: Setup page for calibrating the accelerometer.

Figure 5: Motor testing in Beta Flight configurator.

⚠ **Warning: Do not alter the firmware or any other settings in the system than mentioned above.**

i : The above setting can be done wirelessly by connecting the device to the FC Wi-Fi and connecting to beta flight configurator (for PC/laptop) and using SpeedyBee application (for mobiles).

5.2.2 Raspberry PI Software settings

The drone comes with a preloaded software package in the Raspberry Pi microcontroller for image processing, sensor data collection, FC movement control and UI datalogging. Therefore, no further modifications are required. In case of any issues contact us for further troubleshooting.

5.3 What to do in emergency and exceptional situations

5.3.1 Emergency

In case of a technical failure or if the drone loses the flight path, switch to manual mode using the UI or stop the drone by pressing the Emergency stop button. If the drone poses a risk to people or property, power it off immediately and remove the battery.

5.3.2 Exceptional situations

- **Loss of Communication**: If the wireless connection is lost for more than 5 sec, then drone will perform a safe landing until communication is restored.
- **Power Failure**: In the event of a battery failure, the drone will land slowly.

: Ensure the battery is regularly maintained to avoid power failures. Lipo batteries with continuous usage below 20% shall reduce the battery life.

6 Maintenance and cleaning

6.1 How to maintain the product

6.1.1 Product maintenance by non-skilled persons

- **Battery:** Regularly charge and store batteries properly.
- **Cleaning:** Wipe the drone's surface and sensors with a dry cloth after each use.

6.1.2 Product maintenance by skilled persons

- **Replacing Propellers:** Use tools to replace damaged propellers.
- **Flight Controller Updates:** Apply firmware updates or recalibrate sensors as mentioned above.

6.1.3 Planned maintenance

Maintenance tasks shall be done according to the following plan:

Task	Frequency
Replacing the battery	Every 6 months
Replacing the propellors	Every 3 months
Cleaning the drone	Every week

i: Replace the parts if required before the mentioned time in case of failures or damages.

6.2 How to clean the product

To clean the drone, use a soft dry brush to remove dust and debris from the body and sensors. Do not use water or cleaning solvents as they could damage the electronic components. Do not use a heating blower as this could melt the circuitry inside.

7 Troubleshooting and repair

7.1 How to Identify and solve problems

7.1.1 Troubleshooting and repair by non-skilled persons

Error	Cause	Action
Drone Not Powering On	Battery may not be properly connected or is fully discharged.	Ensure the battery is securely connected to the drone and fully charged before use.
UI not connecting to Wi-Fi	Network issues.	Try refreshing the connection. If the problem persists, contact customer care.
Short flight time	Poor battery health or battery not fully charged.	Check the battery's health and if it's degraded, replace it with a new one. Ensure the battery is fully charged before the flight.
Images are blurred	Fog or dirt in the lens.	Clean the camera lens using soft dry cloth.
Battery not charging	Faulty charger or battery cell damage.	Replace the charger if faulty or replace the battery if the cell voltages are low after full charging.

7.1.2 Troubleshooting and repair by skilled persons

Error	Cause	Action
Drone firmware malfunction	Corrupted or outdated firmware on the flight controller.	Contact customer care for booting the correct firmware.
Motors not working.	Loose connection or faulty motors.	Check for connections. Use the beta flight configurator to check motor functioning individually. If still failing, contact customer care for replacing.

Drone not responding to line-following commands	Faulty or malfunctioning sensors.	Check for sensor voltages and current. If not in adequate range during work contact customer care for service.
Drone flying with a tilt	Error in accelerometer calibration.	Calibrate the accelerometer using Beta Flight configurator.

Supplied accessories, consumables and spare parts

Image	Name
	Drone body (assembled)
	650mAh LiPo battery
	Spare propellers
	Micro USB cable

Appendix B. Related documentation

Sl no	Document Title	Details	Author
1	Beta Flight User Manual	Provides detailed instructions on how to configure and update the drone's flight controller.	Beta Flight Development Team
2	LiPo Battery Care Guide	Offers best practices for handling and maintaining LiPo batteries for drones.	Turnigy
3	Raspberry Pi Pico User Manual	Provides the details on how to use the microcontroller and troubleshooting it.	Raspberry Pi Foundation

18.2 Appendix A

Communication between the FC and MC controlling over the pitch of the Drone is established using the following:

```
from machine import UART, Pin
import time
uart = UART(0, baudrate=115200, tx=Pin(0), rx=Pin(1))
def calculate_checksum(payload):
    return sum(payload) % 256
def send_pitch_command(pitch_value):
    header = [36, 77, 60]  # MSP header "$M<"
    command_code = 200
    payload = [pitch_value & 0xFF, (pitch_value >> 8) & 0xFF]
    payload size = len(payload)
    checksum = calculate_checksum(payload)
    msp_message = bytearray(header + [payload_size] + [command_code] + payload +
[checksum])
    uart.write(msp_message)
    print("Pitch command sent:", msp_message)
while True:
    pitch_value = 10
    send_pitch_command(pitch_value)
 time.sleep(1)
```

18.3 Appendix B

```
import cv2
import numpy as np
import subprocess
```

```
libcamera_process = subprocess.Popen(
    ['libcamera-vid', '--width', '480', '--height', '360', '-t', '0', '--codec', 'mjpeg', '-o', '-'],
    stdout=subprocess.PIPE
)
def get_frame_from_pipe(pipe):
    raw_image = pipe.stdout.read(480 * 360 * 3)
    image = np.frombuffer(raw_image, dtype='uint8').reshape((360, 480, 3))
    return image
hsvVals = [0, 0, 85, 179, 34, 255]
sensors = 3
threshold = 0.2
width, height = 480, 360
sensitivity = 3  # If number is high less sensitivity
weights = [-25, -15, 0, 15, 25]
curve = 0
def thresholding(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower = np.array([hsvVals[0], hsvVals[1], hsvVals[2]])
    upper = np.array([hsvVals[3], hsvVals[4], hsvVals[5]])
    mask = cv2.inRange(hsv, lower, upper)
    return mask
def getContours(imgThres, img):
    contours, hieracrhy = cv2.findContours(imgThres, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
    biggest = max(contours, key=cv2.contourArea)
    x, y, w, h = cv2.boundingRect(biggest)
    cx = x + w // 2  # Center x
    cy = y + h // 2  # Center y
    cv2.drawContours(img, biggest, -1, (255, 0, 255), 7)
```

```
    cv2.circle(img, (cx, cy), 10, (0, 255, 0), cv2.FILLED)
  return cx

def getSensorOutput(imgThres, sensors):
    imgs = np.hsplit(imgThres, sensors)
    totalPixels = (imgThres.shape[1] // sensors) * imgThres.shape[0]
    senOut = []
    for x, im in enumerate(imgs):
        pixelCount = cv2.countNonZero(im)
        if pixelCount > threshold * totalPixels:
            senOut.append(1)
        else:
            senOut.append(0)
        cv2.imshow(str(x), im)
    print(senOut)
    return senOut

def senCommands(senOut, cx):
    global curve
    lr = (cx - width // 2) // sensitivity
    lr = int(np.clip(lr, -10, 10))
    if lr < 2 and lr > -2:
        lr = 0
    if senOut == [1, 0, 0]:
        curve = weights[0]
    elif senOut == [1, 1, 0]:
        curve = weights[1]
    elif senOut == [0, 1, 0]:
        curve = weights[2]
```

```
    elif senOut == [0, 1, 1]:
        curve = weights[3]
    elif senOut == [0, 0, 1]:
        curve = weights[4]
    elif senOut == [0, 0, 0]:
        curve = weights[2]
    elif senOut == [1, 1, 1]:
        curve = weights[2]
    elif senOut == [1, 0, 1]:
        curve = weights[2]
    print(curve)
    return curve
while True:
    img = get_frame_from_pipe(libcamera_process.stdout)
    imgThres = thresholding(img)
    cx = getContours(imgThres, img)
    senOut = getSensorOutput(imgThres, sensors)
    curve = senCommands(senOut, cx)
    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.putText(img, 'Output Matrix: ' + str(senOut), (50, 50), font, 1, (255, 255, 0), 2,
cv2.LINE_4)
    cv2.imshow('Output', img)
    cv2.imshow('Path', imgThres)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
libcamera_process.terminate()
cv2.destroyAllWindows()
```

18.4 Appendix C

```
import cv2
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
import time

# Initialize the Pi Camera
camera = PiCamera()
camera.resolution = (480, 360)  # Set camera resolution
camera.framerate = 30  # Set the frame rate
rawCapture = PiRGBArray(camera, size=(480, 360))

# Allow the camera to warm up
time.sleep(2)

# Path detection parameters
hsvVals = [0, 0, 193, 179, 238, 255]
sensors = 3
threshold = 0.4
width, height = 480, 360
sensitivity = 3  # If the number is high, less sensitivity
weights = [-90, -45, 0, 45, 90]
curve = 0
cmd = ""

# Function to apply thresholding on the input image
def thresholding(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

```
    lower = np.array([hsvVals[0], hsvVals[1], hsvVals[2]])
    upper = np.array([hsvVals[3], hsvVals[4], hsvVals[5]])
    mask = cv2.inRange(hsv, lower, upper)
    return mask

# Function to draw the grid only on the path detection image
def drawGrid(img, sensors=3):
    cell_width = width // sensors
    cell_height = height // sensors
    for i in range(1, sensors):
        cv2.line(img, (i * cell_width, 0), (i * cell_width, height), (255, 255, 255), 2)
    for j in range(1, sensors):
        cv2.line(img, (0, j * cell_height), (width, j * cell_height), (255, 255, 255), 2)
    return cell_width, cell_height

# Function to detect contours and get the center point
def getContours(imgThres, img, imv):
    # Find contours in both the thresholded image and the smaller image segment
    contours, _ = cv2.findContours(imgThres, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
    contours2, _ = cv2.findContours(imv, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)

    # Ensure contours are not empty
    if len(contours) == 0 or len(contours2) == 0:
        # Return the center of the image if no contours are found
        return img.shape[1] // 2, img.shape[0] // 2

    # Find the biggest contour and bounding rectangle
```

```
    biggest = max(contours, key=cv2.contourArea)
    biggest2 = max(contours2, key=cv2.contourArea)
    x, y, w, h = cv2.boundingRect(biggest2)

    # Getting the center point of the path
    cx = imv.shape[1] + x + w // 2  # Center x
    cy = imv.shape[0] + y + h // 2  # Center y

    # Draw the path contour and center on the main image (no grid)
    cv2.drawContours(img, biggest, -1, (0, 255, 0), 5)
    cv2.circle(img, (cx, cy), 8, (0, 0, 255), cv2.FILLED)

    return cx, cy

# Function to get sensor output values for grid cells
def getSensorOutput(imgThres, sensors):
    imgsh = np.hsplit(imgThres, sensors)
    senOut = []
    cx, cy = None, None

    for x, imh in enumerate(imgsh):
        imgsv = np.vsplit(imh, sensors)
        for y, imv in enumerate(imgsv):
            totalPixels = imv.shape[1] * imv.shape[0]
            pixelCount = cv2.countNonZero(imv)

            if pixelCount > threshold * totalPixels:
                senOut.append(1)
            else:
```

```
                senOut.append(0)

            # Capture the center position for visualization
            if x == 1 and y == 1:
                cx, cy = getContours(imgThres, img, imv)

    senserOut = ((np.array(senOut).reshape(3, 3)).T).tolist()
    print(senserOut)
    return senserOut, cx, cy

# Function to generate movement commands based on sensor output
def senCommands(senOut, cx):
    global curve, cmd
    lr = (cx - width // 2) // sensitivity
    lr = int(np.clip(lr, -10, 10))

    if lr < 2 and lr > -2:
        lr = 0

    if senOut == [[0, 1, 0], [0, 1, 0], [0, 1, 0]]:
        curve = weights[2]
        cmd = "Forward"
    elif senOut == [[1, 1, 0], [0, 1, 0], [0, 1, 0]]:
        curve = weights[1]
        cmd = "Slight Left"
    elif senOut == [[0, 1, 1], [0, 1, 0], [0, 1, 0]]:
        curve = weights[3]
        cmd = "Slight Right"
    elif senOut == [[0, 0, 0], [0, 1, 1], [0, 1, 0]]:
```

```
        curve = weights[4]
        cmd = "Right"
    elif senOut == [[0, 0, 0], [1, 1, 0], [0, 1, 0]]:
        curve = weights[0]
        cmd = "Left"
    elif senOut == [[0, 0, 0], [0, 1, 0], [0, 1, 0]]:
        curve = weights[0]
        cmd = "Stop"
    else:
        curve = weights[2]
        cmd = "Forward"

    return curve, cmd

# Main loop for capturing video frames from the Pi Camera
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    img = frame.array
    img = cv2.resize(img, (width, height))
    imgThres = thresholding(img)

    # Create a copy of imgThres for grid and label drawing
    imgThresCopy = imgThres.copy()
    cell_width, cell_height = drawGrid(imgThresCopy)

    # Sensor values from the thresholded image
    senOut, cx, cy = getSensorOutput(imgThres, sensors)

    # Label each grid cell with 0 or 1 based on `senOut` values in `imgThresCopy`
    for i in range(sensors):
```

```
        for j in range(sensors):
            value = senOut[i][j]
            x_pos = j * cell_width + cell_width // 2
            y_pos = i * cell_height + cell_height // 2
            cv2.putText(imgThresCopy, str(value), (x_pos - 10, y_pos + 10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)

    # Commands based on sensor values
    curve, cmd = senCommands(senOut, cx)

    # Display commands on the original image (no grid)
    #cv2.putText(img, cmd, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2,
cv2.LINE_4)
    cv2.arrowedLine(img, (cx, cy), (cx + curve, cy - 50), (0, 0, 255), 5, tipLength=0.5)

    # Show the images
    cv2.imshow('Output', img)  # Original image without the grid
    cv2.imshow('Path Detection with Grid', imgThresCopy)  # Grid and labels only in
`imgThresCopy`

    # Clear the stream for the next frame
    rawCapture.truncate(0)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cv2.destroyAllWindows()
```

18.5 Appendix D

```
import pigpio
import time
import smbus2
import math

# Pressure sensor (BMP180) configuration
BMP180_I2C_ADDRESS = 0x77
bus = smbus2.SMBus(1)

# BMP180 Registers and commands
BMP180_REG_CONTROL = 0xF4
BMP180_REG_RESULT = 0xF6
BMP180_COMMAND_TEMP = 0x2E
BMP180_COMMAND_PRESSURE = 0x34

# Calibration parameters addresses
CAL_AC1 = 0xAA
CAL_AC2 = 0xAC
CAL_AC3 = 0xAE
CAL_AC4 = 0xB0
CAL_AC5 = 0xB2
CAL_AC6 = 0xB4
CAL_B1 = 0xB6
CAL_B2 = 0xB8
CAL_MB = 0xBA
CAL_MC = 0xBC
CAL_MD = 0xBE
```

```
# Define GPIO pins for ultrasonic sensor
TRIG = 23  # GPIO pin connected to the TRIG pin of HC-SR04
ECHO = 24  # GPIO pin connected to the ECHO pin of HC-SR04

# Initialize pigpio for ultrasonic sensor
pi = pigpio.pi()

# Set TRIG as output and ECHO as input
pi.set_mode(TRIG, pigpio.OUTPUT)
pi.set_mode(ECHO, pigpio.INPUT)

# Constants for ultrasonic sensor
SOUND_SPEED = 34300  # Speed of sound in cm/s
SAMPLES = 5  # Number of samples to average

# Read signed 16-bit value from a register
def read_signed_16bit(register):
    high, low = bus.read_i2c_block_data(BMP180_I2C_ADDRESS, register, 2)
    value = (high << 8) + low
    if value > 32767:
        value -= 65536
    return value

# Read unsigned 16-bit value from a register
def read_unsigned_16bit(register):
    high, low = bus.read_i2c_block_data(BMP180_I2C_ADDRESS, register, 2)
    return (high << 8) + low

# Read calibration data
```

```
def read_calibration_data():
    cal = {}
    cal['AC1'] = read_signed_16bit(CAL_AC1)
    cal['AC2'] = read_signed_16bit(CAL_AC2)
    cal['AC3'] = read_signed_16bit(CAL_AC3)
    cal['AC4'] = read_unsigned_16bit(CAL_AC4)
    cal['AC5'] = read_unsigned_16bit(CAL_AC5)
    cal['AC6'] = read_unsigned_16bit(CAL_AC6)
    cal['B1'] = read_signed_16bit(CAL_B1)
    cal['B2'] = read_signed_16bit(CAL_B2)
    cal['MB'] = read_signed_16bit(CAL_MB)
    cal['MC'] = read_signed_16bit(CAL_MC)
    cal['MD'] = read_signed_16bit(CAL_MD)
    return cal

# Read raw temperature data
def read_raw_temperature():
    bus.write_byte_data(BMP180_I2C_ADDRESS, BMP180_REG_CONTROL,
BMP180_COMMAND_TEMP)
    time.sleep(0.005)  # Wait for conversion to complete
    raw_temp = read_unsigned_16bit(BMP180_REG_RESULT)
    return raw_temp

# Read raw pressure data
def read_raw_pressure():
    bus.write_byte_data(BMP180_I2C_ADDRESS, BMP180_REG_CONTROL,
BMP180_COMMAND_PRESSURE)
    time.sleep(0.04)  # Wait for conversion to complete
    msb = bus.read_byte_data(BMP180_I2C_ADDRESS, BMP180_REG_RESULT)
```

```
    lsb = bus.read_byte_data(BMP180_I2C_ADDRESS, BMP180_REG_RESULT + 1)
    xlsb = bus.read_byte_data(BMP180_I2C_ADDRESS, BMP180_REG_RESULT + 2)
    raw_pressure = ((msb << 16) + (lsb << 8) + xlsb) >> 8
    return raw_pressure

# Calculate temperature and pressure using calibration data
def calculate_temperature_pressure(cal):
    UT = read_raw_temperature()
    UP = read_raw_pressure()

    X1 = ((UT - cal['AC6']) * cal['AC5']) / 32768
    X2 = (cal['MC'] * 2048) / (X1 + cal['MD'])
    B5 = X1 + X2
    temperature = (B5 + 8) / 16.0 / 10.0

    B6 = B5 - 4000
    X1 = (cal['B2'] * (B6 * B6 / 4096.0)) / 2048.0
    X2 = cal['AC2'] * B6 / 2048.0
    X3 = X1 + X2
    B3 = (((cal['AC1'] * 4 + X3) * 2) + 2) / 4.0
    X1 = cal['AC3'] * B6 / 8192.0
    X2 = (cal['B1'] * (B6 * B6 / 4096.0)) / 65536.0
    X3 = ((X1 + X2) + 2) / 4.0
    B4 = cal['AC4'] * (X3 + 32768.0) / 32768.0
    B7 = ((UP - B3) * 50000.0)

    if B7 < 2147483648:
        pressure = (B7 * 2) / B4
    else:
```

```
        pressure = (B7 / B4) * 2

    X1 = (pressure / 256.0) ** 2
    X1 = (X1 * 3038.0) / 65536.0
    X2 = (-7357.0 * pressure) / 65536.0
    pressure = pressure + (X1 + X2 + 3791.0) / 16.0

    return temperature, pressure

# Calculate altitude based on pressure
def calculate_altitude(pressure, sea_level_pressure=1013.25):
    altitude = 44330.0 * (1.0 - (pressure / sea_level_pressure) ** (1 / 5.255))
    return altitude

# Function to calculate distance from ultrasonic sensor
def get_distance():
    distances = []

    for _ in range(SAMPLES):
        pi.write(TRIG, 0)  # Ensure the trigger is low before sending a pulse
        time.sleep(0.05)

        pi.write(TRIG, 1)
        time.sleep(0.00001)  # Trigger pulse for 10 microseconds
        pi.write(TRIG, 0)

        while pi.read(ECHO) == 0:
            pulse_start = time.time()
```

```
        while pi.read(ECHO) == 1:
            pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * SOUND_SPEED / 2
        distances.append(distance)

    return sum(distances) / len(distances)

# Function to average multiple altitude readings to improve sensitivity
def get_average_altitude(calibration_data, num_readings=5):
    total_pressure = 0
    for _ in range(num_readings):
        _, pressure = calculate_temperature_pressure(calibration_data)
        total_pressure += pressure
        time.sleep(0.1)
    average_pressure = total_pressure / num_readings
    altitude = calculate_altitude(average_pressure / 100.0)  # Convert Pa to hPa
    return altitude

# Main loop
calibration_data = read_calibration_data()

try:
    initial_altitude = get_average_altitude(calibration_data)  # Take initial altitude reading
    print("Starting position set to 0 meters.")

    while True:
        # Get ultrasonic distance
```

```
        dist = get_distance()

        # Get pressure-based altitude
        current_altitude = get_average_altitude(calibration_data)
        relative_altitude = current_altitude - initial_altitude

        if dist > 0 and dist <= 50:  # Detect objects within 0 to 50 cm
            output_string = f"Ultrasonic Distance: {dist:.1f} cm, Relative Altitude: 
{relative_altitude:.2f} meters"
            print(output_string)
            time.sleep(1)  # Update every 5 seconds if an obstacle is found
        else:
            output_string = f"more than 50cm, Relative Altitude: {relative_altitude:.2f} meters"
            print(output_string)
            time.sleep(1)  # Update every 5 seconds if an obstacle is found
        #time.sleep(0.1)  # Check frequently but only update if an obstacle is detected
        # Combine the ultrasonic distance and altitude in one string



        #time.sleep(1)  # Update every second

except KeyboardInterrupt:
    print("Program stopped.")
    pi.stop()
```

18.6 Appendix E

```
import time
import smbus2
import math
```

```
# LSM303DLHC I2C address
LSM303_ADDRESS_ACCEL = 0x1e  # Accelerometer I2C address for LSM303DLHC

# Register addresses for LSM303DLHC accelerometer
LSM303_REG_ACCEL_X_LSB = 0x28  # X-axis LSB register
LSM303_REG_ACCEL_X_MSB = 0x29  # X-axis MSB register
LSM303_REG_ACCEL_Y_LSB = 0x2A  # Y-axis LSB register
LSM303_REG_ACCEL_Y_MSB = 0x2B  # Y-axis MSB register
LSM303_REG_ACCEL_Z_LSB = 0x2C  # Z-axis LSB register
LSM303_REG_ACCEL_Z_MSB = 0x2D  # Z-axis MSB register
LSM303_REG_ACCEL_CONFIG_A = 0x20  # Accelerometer configuration register

# Initialize I2CL

# Scale factor
scaleFactor = 0.980665  # mg to cm/s² conversion

# Low-pass filter constant
ALPHA = 0.1



# Initialize sensor
def initialize_sensor():
    # Set the accelerometer to normal mode
    bus.write_byte_data(LSM303_ADDRESS_ACCEL, LSM303_REG_ACCEL_CONFIG_A,
0x57)
```

```
# Read accelerometer data with low-pass filter to smooth the results
def read_acceleration(last_x):
    # Read 6 bytes of accelerometer data starting from the X LSB register
    data = bus.read_i2c_block_data(LSM303_ADDRESS_ACCEL,
LSM303_REG_ACCEL_X_LSB | 0x80, 6)

    # Combine MSB and LSB for X, Y, Z axes
    x_raw = (data[1] << 8) | data[0]

    # Convert to signed 16-bit integers
    if x_raw > 32767: x_raw -= 65536

    # Apply scale factor to convert raw data to cm/s²
    x = (x_raw * scaleFactor)/100

    # Apply low-pass filter to smooth out the noise
    x_filtered = ALPHA * x + (1 - ALPHA) * last_x

    return x_filtered

# Integrate acceleration to compute speed
def calculate_speed(acceleration, dt, current_speed):
    new_speed = current_speed + acceleration * dt
    return new_speed

# Main program
if __name__ == "__main__":
    initialize_sensor()
```

```
current_speed_x = 0.0  # Initial speed along X-axis (in cm/s)

last_accel_x = 0.0  # Last X-axis acceleration (for low-pass filter)

last_time = time.time()  # Start time for integration

try:
    while True:
        current_time = time.time()  # Get current time
        dt = current_time - last_time  # Calculate time step in seconds

        # Read filtered acceleration values (cm/s²)
        accel_x = read_acceleration(last_accel_x)

        # Update speed based on acceleration (numerical integration)
        current_speed_x = calculate_speed(accel_x, dt, current_speed_x)

        # Update the last acceleration values for the next iteration
        last_accel_x = accel_x

        # Update the last time for the next iteration
        last_time = current_time

        # Convert speeds from cm/s to m/s for output
        speed_x_m_s = current_speed_x / 100

        # Print the acceleration and speed values
        print(f"Acceleration X: {accel_x:.2f} cm/s² | Speed X: {speed_x_m_s:.2f} m/s")
```

```
        # Sleep for a short time
        time.sleep(0.1)

    except KeyboardInterrupt:
        print("Program terminated.")
    finally:
        bus.close()
```

18.7 Appendix F

```
import RPi.GPIO as GPIO
import time
# Ultrasonic Sensor Pins
TRIG = 23
ECHO = 24

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

# Measuring distance using the ultrasonic sensor
def get_altitude():
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()
    while GPIO.input(ECHO) == 1:
```

```
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 171.50  # Convert to centimeters (half of speed of sound in m)
    distance = round(distance, 2)

    return distance  # Return altitude in m



class PIDController:
    def __init__(self, Kp, Ki, Kd):
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd
        self.prev_error = 0
        self.integral = 0

    def compute(self, error, dt):
        self.integral += error * dt
        derivative = (error - self.prev_error) / dt
        output = (self.Kp * error) + (self.Ki * self.integral) + (self.Kd * derivative)
        self.prev_error = error
        return output

import time

# PID controller for altitude
pid_altitude = PIDController(Kp=1.0, Ki=0.1, Kd=0.05)
```

```
# Hover parameters
target_altitude = 1.2  # Desired altitude in m
hover_time = 30.0  # Hover duration in seconds
start_time = time.time()

while time.time() - start_time < hover_time:
    # Altitude control
    current_altitude = get_altitude()
    altitude_error = target_altitude - current_altitude
    dt = 0.01  # Time interval between updates
    altitude_output = pid_altitude.compute(altitude_error, dt)

    # Convert altitude_output to throttle value (between 0 and 1)
    throttle = 0.5 + altitude_output  # Adjust base throttle (0.5 is for hovering)

    # Clamp throttle value to valid range (0 to 1)
    throttle = max(0, min(1, throttle))

    # Send throttle command to flight controller
    send_throttle_command(throttle)

    time.sleep(0.01)  # Loop delay
```