Institut Superior de l'Aeronatique et de l'Espace

Master of Aerospace Engineering Research Project

S2 Project Report

Autonomous Robotic Aerial Vehicle: ARAV Control in Complex Environments

Author:
Akash SHARMA

Tutors:
Raghuvamsi Deepthimahanthi
Yves Briere

Due Date of report: 29th June, 2022
Actual Submission Date: 29th June, 2022

Starting Date of Project: 29th January, 2022

Contents

List of Figures

List of Tables

Declaration of Authenticity

This assignment is entirely my own work. Quotations from literature are properly indicated with appropriated references in the text. All literature used in this piece of work is indicated in the bibliography placed at the end. I confirm that no sources have been used other than those stated.
I understand that plagiarism (copy without mentioning the reference) is a serious examinations offence that may result in disciplinary action being taken.

Date:
29th June, 2022

Signature:
Akash SHARMA

Abstract

Unmanned Aerial Vehicles (UAVs) and Ground Robots are two pieces of technology that both have their own range of applications in several industries. These applications include surveillance, topography mapping, terrain exploration, videography and photography to name a few. This project which was started four years ago has the goal to merge these two technologies to create a hybrid vehicle capable of both terrestrial and aerial operations. One of the main objectives of this vehicle is to be able to navigate across unknown and dynamic environments completely autonomously, efficiently as well as safely by avoiding obstacles to find the most logical path to the destination. This path may or may not involve the use of both modes of transport available to the vehicle which is a decision that it should make on its own.

Considerable progress has already been made in this project. Several algorithms that implement different aspects of the functionality of the vehicle have been developed and have reached varying degrees of maturity. These algorithms include obstacle avoidance, path planning, hybrid module communication etc. An initial prototype has also been developed and tested with somewhat mixed results. Within the scope of this year's project, the goal is to focus on making the vehicle fully functional as well as optimization of the hybrid module.

This report present the results as well as the methodology used to optimize and test the algorithms and the performance of the vehicle.

Keywords: Hybrid Vehicle, ROS, Pixhawk, Gazebo, Hybrid Module, Aerial Control

1 Introduction

Drones have brought about a technology revolution because of their vast range of applications. The design and kind of technology used to build them depends on these applications such as surveillance, exploration, delivery, topography mapping etc. Ground robots find many uses in space missions for collecting soil samples and analysing the terrain. This is because they are designed to be functionally robust in order to be able to traverse rough and unforgiving terrains. Unmanned Aerial Vehicles are another class of drones that can function without an onboard pilot. They are also used to map topography, collect land survey data for infrastructure projects, fertilizer distribution in agriculture, videography and photography projects etc. Finally, smart robots or autonomous robots that are capable of carrying out complex tasks without the need for human guidance have also emerged as a dominant piece of technology.

This project aims to design, program and test an unconventional design of an Autonomous Robotic Aerial Vehicle that combines the locomotive modes of ground as well as air, hence, making the vehicle more robust in handling mission trajectories in complex environments on its own. In essence, the vehicle demonstrates the technologies implemented in ground robots, UAVs as well as autonomous robots. The vehicle is composed of ground and air modules that activate depending on the decision taken by the vehicle on which path to take. The vehicle uses sensors and a set of algorithms to carry out different challenges during the mission such as obstacle avoidance, safe landing, communication between the air and ground modules etc.

Within the scope of this research, the objectives will be the following:

- This project has already been started by previous students. After the completion of their work, several issues related to aerial performance have persisted. In this context, the communication between the ground and aerial modules will be optimized and tested in simulation as well as on the real vehicle prototype. The flying controller performance will be improved to enable reliable aerial operations.
- A decision-making algorithm will be implemented based on trade-off analysis through which the vehicle is able to make an informed choice about which path to take and what means to use. The vehicle should be smart enough to conserve battery power while avoiding all the obstacles and, at the same time, be able to complete the mission within the time limit.

2 Project Definition

2.1 Context and Key Issues

This project has come a long way from just an idea of the merger of different technologies to being in a stage where we have a partially functional prototype. In order to see this research to its fruition, several key challenges have been identified that are the main roadblocks to be overcome to deliver the final functional vehicle.

- **Master Algorithm Development**: The main challenge in this project is to bring together all the individual algorithms that function as the intelligence of the vehicle. This decision-making algorithm will allow the vehicle in real time to optimise its functional capability of obstacle avoidance, path-planning etc and be able to complete missions more efficiently. Figure 1 shows the desired behavior of the vehicle.

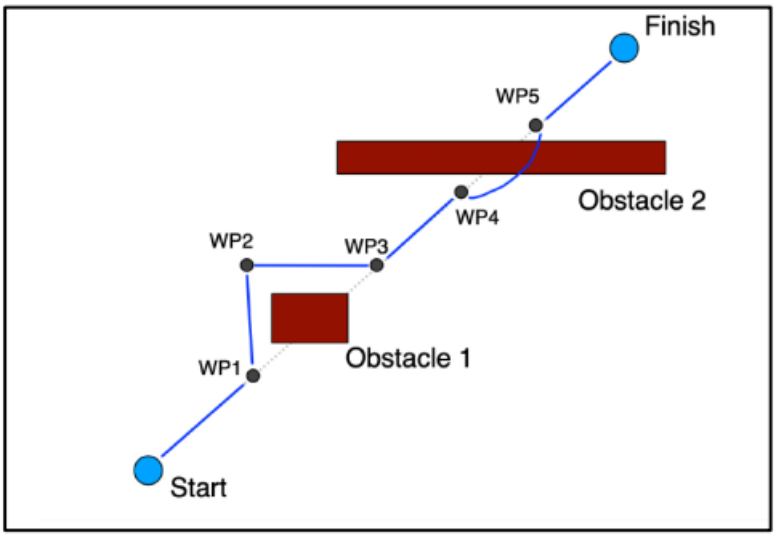


Figure 1: Desired Behavior of the Vehicle. Source [1]

- **Hybrid Module Optimisation**: The choices between the two modules should be made by the vehicle after taking into consideration the optimisation parameters: energy and time. The aerial module would take much less time, but it consumes much more energy, so a trade-off must be made. As of now, the two air and ground modules function properly independently. However, development is needed to be able to switch effectively, seemlessly and safely between them while reducing switching times so the vehicle gets more time to actually navigate through the environment.
- **Weight and PID Tuning**: Another issue to be addressed will be the weight of the vehicle in the context of its aerial performance. The air module is controlled through a PID controller. Currently, the air module has not been tested due to the heavy weight of the vehicle making the PID inadequate in its operation. A thorough analysis must be conducted to find why the vehicle stopped functioning adequately.
- **Real Test Validation**: Due to the heavy weight of the vehicle, none of the aerial tests were performed with the real vehicle. Hence, the previously completed algorithms must be tested in the drone lab along with the new cases to gain confidence of the vehicle's robustness in every scenario.

2.2 State of the art

One of the main objectives of this literature review was to identify decision making algorithms used to optimize the functional capabilities of an autonomous vehicle. These algorithms could be used in the scope of applications other than path selection as well since the idea was to establish their utility.

Po-Lung Yu, in his book, Multiple-Criteria Decision Making [2], talks about non-trivial decision making when several criteria are in play. The best decision is made based on a set of alternatives or choices, the outcomes of each of those choices, the set of criteria that are imposed on the decision to be made and finally, the preferences of some outcomes over others. A decision-making algorithm based on Double Deep Q-Learning with Prioritized Experience Replay has been introduced by Kun Zhang et. al. [3] which is used for autonomous UAV maneuvering and route guidance.

Rovers and drones are used extensively to scout and explore terrain in remote locations or places with hard accessibility. The most popular examples of exploratory drones and rovers are NASA's Mars Helicopter, Ingenuity [4] and the rover, Perseverance [5], as well as the Chinese rovers, Yutu and Yutu-2 [6] which have been exploring the Moon in recent years. Although scarce, some research of unconventional designs of drones similar to our prototype has also been carried out by the Aerospace Robotics and Control Lab at Caltech consisting of a bipedal walking and flying robot [7].

2.2.1 ROS Architecture

The technology revolution and push towards innovation in the field of automation and electric vehicles has created a huge variety of tools, particularly open-source software that have seen a rapid growth in development and usage. This is because these tools offer simple license management and the source code is well designed by communities of developers who also offer abundant support to the industry as well as individual users. The Robot Operating System (ROS) is the most widely used open-source development platform for development of robots and autonomous vehicles [8].

ROS is an open source development kit composed of different libraries and applications for robotics applications. It provides a software platform that can be used for research and prototyping as well as deployment and production. It supports usage of multiple languages and has been designed to be easily integratable with any project.

The underlying principle of ROS is the implementation of **ROS Nodes** which are essentially files of code that are executed when the node is run. These nodes can be written in different languages and still be run simultaneously. Additionally, instead of calling different nodes in order to share data with each other, these nodes communicate via **ROS Topics** that act as channels storing **ROS Messages** that are sent and received by these nodes. In order for a node to participate in communication, it has to either **publish** on a topic or **subscribe** to a topic. This kind of point-to-point architecture builds a network of nodes that works together to carry out different aspects of the robot's functionality. Figure 2 illustrates these concepts through a schematic.

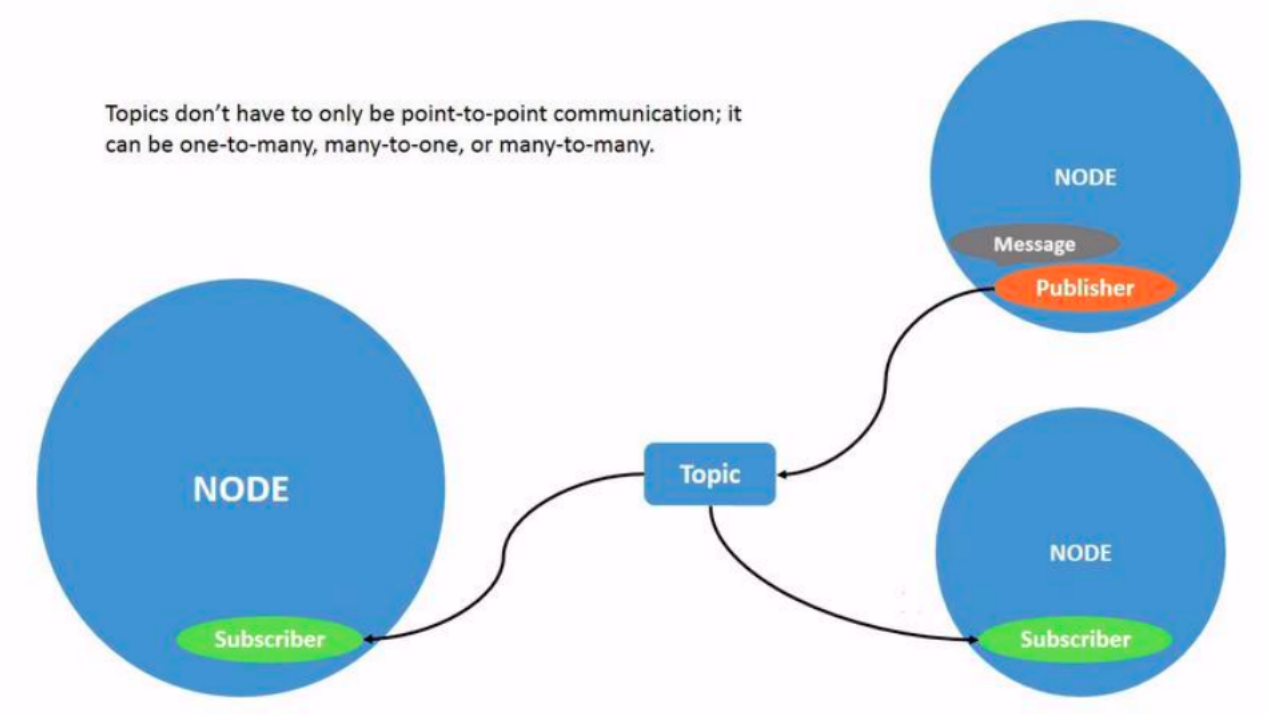


Figure 2: ROS architecture. Source [9]

2.2.2 Current Project Stage

The prototype required for real world testing has been fully assembled comprising of the aerial and ground modules attached together so that during the flight mode, the ground module basically acts as a payload. As of now, this prototype is too heavy to be flown with the current configuration of the Pixhawk PID flight controller. Figure 3 shows the real prototype developed by the previous students [10][11] as well as how the two modules are attached together.

(a) Final Prototype

(b) Air and Ground Module Attachment

Figure 3: Fully Assembled Real Vehicle. Source: Taken by the author

Preliminary research has been conducted on a set of algorithms referred to as modules, which have been simulated and later fully developed by the previous students [11][12][13][1]. However, most of them have only been tested on the ground module. Table 1 gives an idea about the progress made on building the software architecture of the vehicle. Apart from this, a simulation environment with the complete model of the vehicle has been developed using *Gazebo* [14] as well as ROS. Figure 4 shows the developed model by the previous students [12].

Algorithm	Ground Module		Aerial Module	
	Simulation	Testing	Simulation	Testing
Path Planning	Done	Done	Done	Not Done
Environment Perception	Done	Done	Done	Not Done
Safe Landing	NA	NA	Done	Not Done
Decision Making	Not Done	Not Done	To be optimized	Not Done

Table 1: Current Status of the Algorithms. Source [13][1]

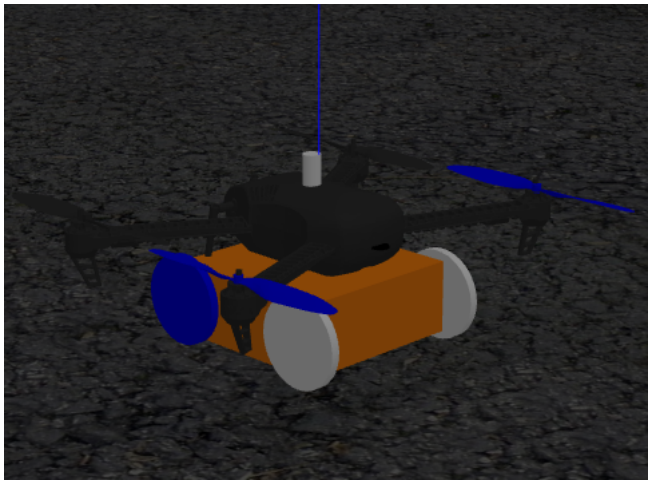


Figure 4: Vehicle Model in Gazebo. Source: [1]

2.3 Potential Degree of Novelty

One of the main factors that has made the technology of drones and rovers so interesting and useful is that they are mechanically simple. These two kinds of robots have been extensively researched and optimized for a variety of applications. Here on Earth, rovers are used for accessing hard to reach or even dangerous places, for example, for diffusing bombs and going into underground mines. At other planets like Mars, they are used to map the terrain or carry out soil sample analysis. Flying drones offer many capabilities of their own such as aerial surveillance and mapping, cargo delivery etc.

This project has the goal to combine these two pieces of technology that can open the doors to a whole new range of applications. With two different means of transport at its disposal, the vehicle can drastically increase its range of area that it can cover. Moreover, the vehicle is also being designed to be fully autonomous which would be particularly useful when carrying out missions that involve uncertainty since the vehicle is equipped to handle such situations. Exploration of other planets with this kind of vehicles will also remove the problem of communication and command signal latency.

During the research and development phase of this project, one aspect of the design of the prototype has been optimized to handle the current weight of the vehicle. The hybrid module has been expanded to take into account energy and time with more effective and realistic switching. Additionally, the incorporation of a safe takeoff and landing (if necessary) protocol has been added to the architecture.

2.4 Objectives

The final objective of this project is to produce a hybrid autonomous vehicle that has the capabilities of a UAV as well as a ground rover and which can navigate through complex environments by assessing its surroundings and making intelligent decisions. It must be able to decide which path and mode of travel is best suited taking into consideration the power consumption, mission time, nature of the environment and the destination coordinates.

The work done in the first semester of this project has yielded some progress as well as allowed a much deeper insight into the various tools and methodologies used to overcome challenges faced in the general design and development of ground and aerial robots. One of the main barriers in testing the aerial module of the actual vehicle for all the previously developed algorithms was that the vehicle was not able to fly. A test analysis to find what exactly caused this problem has been completed. Moreover, the hybrid module has matured further to include more use cases where it has produced good results. Taking this progress into account as well as the work done by the previous students, the following are the list of objectives for the semester.

- **Use case expansion**: The master algorithm which takes into account the power and time constraints in the presence of a complex environment needs to be developed. To do this, firstly, the existing hybrid module needs to be simulated in the presence of an environment of increasing levels of complexity to analyze whether it continues to be able to adhere to the power and time constraints.
- **Latency Reduction**: The master algorithm needs to be developed keeping in mind that the vehicle will function in real time, so signal latency has to be minimal. The master algorithm houses all the control algorithms that are used to coordinate across different platforms like PixHawk, ROS, Raspberry Pi etc, and are, in general, complex algorithms. Therefore, there is always some latency involved between computation and execution which will hinder the vehicle in making fast computations of decisions and hence, the motion of the vehicle will not be smooth and streamlined.
- **Real Vehicle Tests**: All the use cases are going to be tested in real time to validate their performance. It is very likely that the first few tests would reveal new opportunities for improvement that can be worked on.

3 Progress

3.1 Vehicle Specification and Description

The robotic aerial vehicle designed for this project has been upgraded throughout its 4 years of life. This vehicle is a hybrid robot, capable to moving on the ground as well as flying. In order to achieve this, it comprises of 2 different but interconnected modules – the air and the ground module that are managed by a single embedded computer within the vehicle. This section details all the important parts of the vehicle along with their descriptions and functions.

There are several components which are responsible for different functions pertaining to both the modules in the vehicle. These are as follows.

- **Embedded Computer**: The embedded flight computer is responsible for establishing a connection with the external server as well as with the flight controller (Pixhawk [15]) and the ground motor driver. The embedded vehicle computer model is a Raspberry Pi 3 Model B+ [16], which is a single-board computer with a 1.4GHz 64-bit quad-core processor using the arm64 architecture. A 3D printed protective casing has been designed to house the Raspberry Pi on the vehicle.
- **Raspberry Pi Power System**: It is necessary to constantly power the RPi through an external battery with an appropriate capacity and power. A 10000 mAh capacity battery, with a USB-A exit with a 5V-2A performance has been chosen.
- **Environment perception sensors**: For the critical task of understanding the vehicle's environment, it is equipped with different sensors. Two USB stereo cameras based on the OV2659 chipset have been acquired for real time depth perception. Three HC-SR04 ultrasonic sensors are responsible for calculating distances from certain objects that may be obstacles. These sensors have been mounted onto the vehicle frame using 3D printed holders by the previous students [13].
- **Voltage Potentiometer and Adaptor**: Since the voltage of the aerial battery is too high for the ground motors (11.1V compared to the specified 6V) a voltage potentiometer must be used. For this prototype, a DC-DC converter (LM2596 Model) is used which can be regulated allowing the fine-tuning of the voltage amount received by the ground motors. An adaptor component has been designed to install the potentiometer within the vehicle frame.
- **Spherical Markers**: The positioning system present in the ISAE Volière laboratory has been used for the testing of the vehicle, which consists of an OptiTrack [17] motion capture system that uses several cameras around the laboratory environment which detect several spherical markers using a range of cameras located around the environment. These spherical markers on the vehicle provide high accuracy position and orientation of the vehicle.

3.1.1 Aerial Module

The air module takes the form of a quadcopter configuration. The components involved in this module are mentioned below.

- **Drone Frame**: A F450 Drone frame kit with is an X-type frame with a 450mm wheelbase is used for this project.
- **Power Distribution Board (PDB)**: This is included in the F450 Drone frame kit.
- **Motors**: The vehicle has four 2212 920kV brushless DC motors which means that they have a stator width of 22mm and a height of 12mm.
- **Propellers**: Propellers of type 9450 are used (9.4-inch diameter, 5.0-inch pitch) with the motors.
- **Electronic Speed Controller (ESCs)**: 4 of the 30A ESCs are used in this project.
- **Aerial Battery**: A Floureon 3S, 11.1V, 3000mAh, 30C LiPo battery is used to power up the vehicle.
- **Flight Controller**: The vehicle is equipped with Pixhawk which serves as the flight controller.

3.1.2 Ground Module

The ground module design is adopted from that of a tank which enables control on rough terrain and lessens the chances of it getting stuck. The components in this module are as follows.

- **Rover Frame**: The Tamiya universal plate set is used as a frame.
- **Tracks and Wheels**: The ground module is based on the Tamiya 70100 track and wheel set.
- **Gearbox**: The Tamiya 70097 twin-motor gearbox consists of the gearboxes and two independent brushed DC motors, which are used to drive two shafts. The motors run on 3-6V.
- **Power System**: The Floureon 3S, 11.1V, 3000mAh, 30C LiPo battery is used (The same battery powers the drone propellers as well as the rover motors).
- **Motor Driver**: The ground module connections enable the communication between the Raspberry Pi and the TB6612FNG motor driver, as well as the connection of this driver with the ground motors and battery. The TB6612FNG motor driver controls the speed and direction of the two rear motors with a supply range of 2.5V to 13.5V.

3.2 Software Architecture

The final objective of the vehicle is to be completely autonomous. It should be able to navigate through complex environments and avoid obstacles by perceiving the environment around it and making decisions without any human input. In order to make this happen, several algorithms, referred to as modules, have been designed which are responsible for specific aspects of the functionality of the vehicle. Additionally, a strong development platform is needed to build the underlying software for such a hybrid vehicle as well as carry out simulations to test the algorithms. The development tools used for this project are detailed below.

- **ROS – Robot Operating System**: ROS [8] has been used for developing the different modules that manage various functionalities of the vehicle, from path-planning and obstacle avoidance to high level ground and air control. As mentioned in Section 2.2.1, ROS is an open source development environment comprised of libraries, applications and different tools that can be utilized to develop and customize any kind of robot. ROS operates on a point-to-point architecture where several nodes communicate messages with each other through communication channels called topics. This entire network of interconnected nodes provides the base of the vehicle software.
- **Gazebo**: Gazebo [14] is the chosen 3D simulator for testing the vehicle in simulation before the real test. It is another open source real world simulator which can realistically simulate scenarios in indoor as well as outdoor environments. It offers a robust physics engine which makes simulating collisions easier. It can also simulate sensor feedback with noise through its various sensor models. Gazebo is also compatible with ROS which makes its utility much more convenient.
- **Pixhawk (PX4)**: Pixhawk [15] is the flight controller installed on the vehicle. The controller receives sensor data and adequately sends motor and actuator values. A mission using Pixhawk is carried out through a state machine. Each phase of the flight is activated only when the vehicle has successfully gone into that state. For example, the vehicle needs to go into the **Hold Mode** in order to be able to hover at any point in space or go into **Land Mode** to land the vehicle. Since the vehicle is planned to be commanded by an offboard API which is ROS, the vehicle needs to go into **Offboard Mode** before ROS can run its nodes which supply Pixhawk with the destination coordinates.
- **QGroundControl (QGC)**: The QGC station [18] is useful for real time monitoring and mission planning. This tool gives a comprehensive picture about the vehicle state, power and battery consumption which is useful to study the decision-making capabilities of the vehicles when it tries to conserve battery while navigating an optimum path.

It is necessary to simulate the performance of PX4 in the Gazebo simulations as well before carrying out real tests. For this purpose, the **MAVROS** package [19] available in ROS is used to establish communication between ROS and Pixhawk. This package enables the **MAVLink** communication protocol [20] which is the supporting bridge between ROS and PX4. It is a binary telemetry protocol that must be enabled in all drones using Pixhawk. PX4 communicates with Gazebo to receive sensor data from the simulated world and send motor and actuator values. It also communicates with QGC or an Offboard API like ROS to send telemetry from the simulated environment and receive commands. Though commands can also be given through QGC, several ROS nodes containing the algorithms to be simulated are run instead that start by asking the user

directly for the final coordinates as well as the mission time to adhere to. Figure 5 provides a summary of the different elements of the software architecture and how they interact with one another. The ports mentioned in the diagram refer to the connections to be made to integrate Pixhawk to any real vehicle.

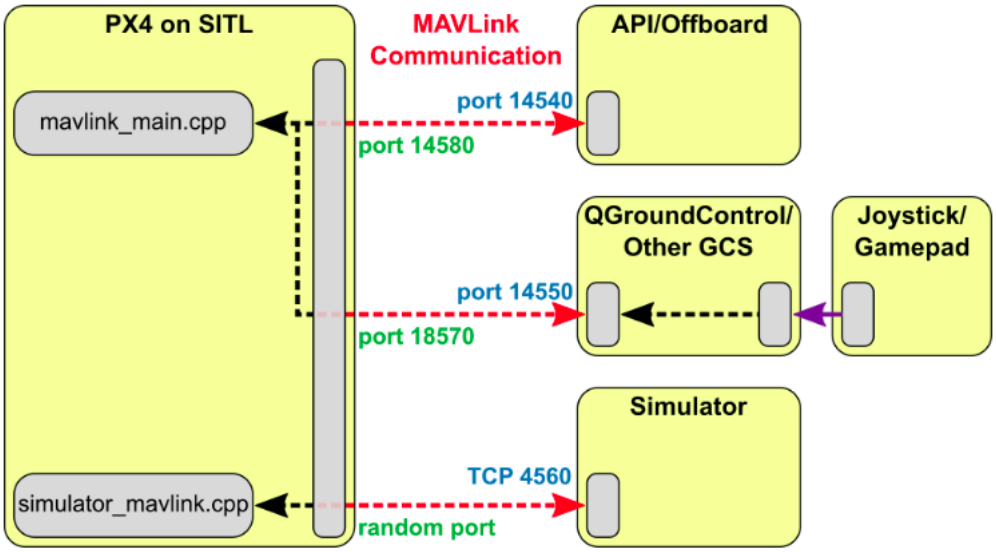


Figure 5: Software Architecture of the vehicle (GCS - Ground Control Station) Source: Pixhawk User Guide [15]

3.3 Functioning Algorithms

This vehicle has the first task to navigate through a complex environment. This means that it needs to understand its environment and identify the optimum path to reach its destination by avoiding different obstacles in the way. The second and equally important task is to do this safely. Several algorithms or modules have been developed that are dedicated to each of these tasks of environment perception, path-planning, safe-landing etc. The manner in which these algorithms run to carry out the tasks of the vehicle is mentioned as follows.

Environment Perception

The onboard ultrasonic and stereo camera sensors capture data of the environment to generate a point cloud map of the environment. This map is then published to a ROS topic that is subscribed by the path-planning module.

Octomap Grid Generation

The sensor data is received by the path planner to generate what is called an Occupancy grid map, or Octomap [21]. A basic occupancy grid represents 3D space as an evenly spaced grid of cubes. Every cube in the space has a binary identifier, which represents whether the space it occupies has an obstacle present or not.

Path-Planning

The path-planner implemented in this project is the Rapidly-exploring Random Trees – star (RRT*) planner. The RRT* algorithm uses Rapidly exploring Random Trees to map the environment and find a suitable path. Once the initial solution is found, it continues to identify more suitable paths, by rewiring the way-points using an optimization logic. In this way, RRT* creates an expanding tree, biased in reaching the solution.

Once the path-planner has the robot position, the destination coordinates and the obstacle map from the sensors, it computes sequentially a ground path and an aerial path and publishes these as a topic. Along with this, it also outputs the length of each path which is then used by the path-selector module.

Path Selection/Module Communication

The two groups of students who have worked on this algorithm have taken two different approaches to select the optimum path based on power and time constraints.

- **First Approach**: The path-selector module is responsible for making a decision between a ground path and an aerial path. It does this by receiving the length data from the path planning module and computes a cost function which is minimized to deduce the most optimum path. Two main criteria used for this decision making are battery consumption and time taken. This method was adopted by Joan Bessa and Alberto Ceballos [13].
- **Second Approach**: The second approach looks at the z coordinate of the final coordinates entered by the user as an input. If this value is above 0.4m, then the algorithm understands that activating the aerial module is necessary, so the vehicle takes off from its initial position and flies to the destination. Otherwise, the ground module is used throughout the journey to reach the destination. This method was adopted by Karthik Mallabadi and Ryan Dsouza [1].

Path Control

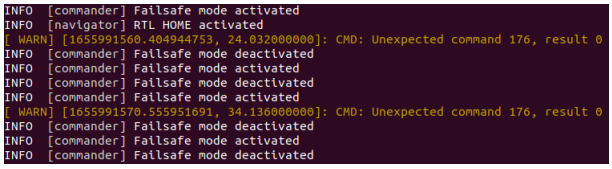
Finally, the path control module makes the vehicle follow the final path. Depending on which path has been chosen, the embedded Raspberry Pi communicates with the Pixhawk flight controller for aerial control and with the Motor Driver for ground control.

Safe Landing

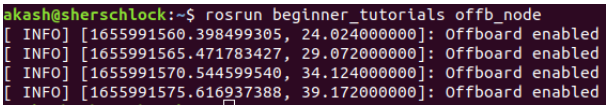
To ensure the safety of the vehicle during the mission, a safe-landing algorithm has also been designed which ensures that the vehicle lands on a flat surface smoothly in case the vehicle needs to switch to the ground module after flying. This algorithm is also activated as a fail-safe mechanism in the event that the battery level goes below 40%.

3.4 Setting up the Offboard Node

Pixhawk offers a tutorial to setup an offboard node through a simple example where a drone is controlled through the node with MAVROS to climb and hover at an altitude of 1m. This node has also been used multiple times in the project itself when a simple test flight is required to tune the PID controller, for example. A ROS publisher is created which publishes the target coordinates to the rostopic mavros/setpoint_position/local at a rate of 20Hz. The target coordinates are entered into the message called geometry_msgs PoseStamped. However, an error had arisen on running the node in the simulation. Figure 6 shows this error.



(a) Failsafe Triggered



(b) Offboard Repeatedly Enabling

Figure 6: Offboard Node Execution Errors. Source: Prepared by the author

Once the node starts, it changes the flight mode to Offboard Mode so that the vehicle can be controlled through an external offboard API (ROS, in this case). This means that the vehicle is no longer connected to any Remote Control(RC) system. The default settings of Pixhawk set a failsafe mechanism in the event that there is an RC loss which forces the vehicle to abandon the mission and land immediately. A number of these failsafe mode are shown in Figure 7. The Return To Land (RTL Home) mode which is activated here refers to this mechanism. Thus, this RTL Home mode and the Offboard mode work against each other and we see that the failsafe mode keeps activating the deactivating while the offboard mode keeps getting enabled repeatedly and the vehicle never takes off.

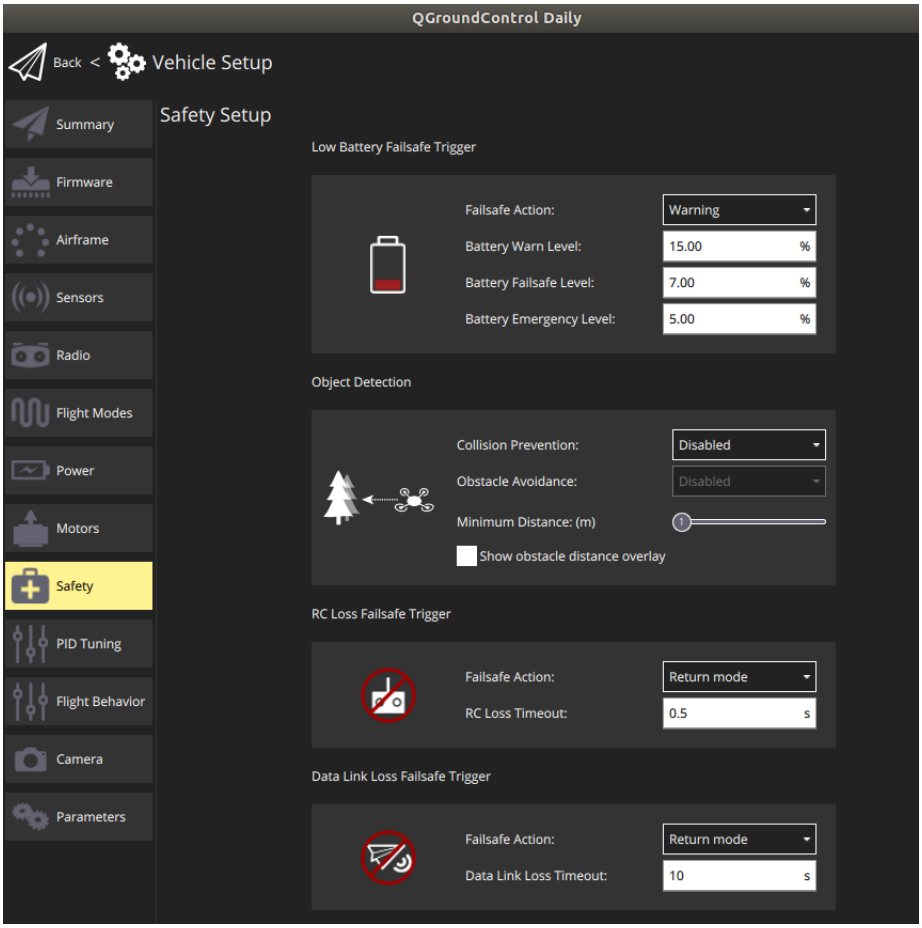


Figure 7: Failsafe Modes in Pixhawk. Source: Prepared by the author

In order to solve this problem, the RC loss exceptions parameter called COM_RCL_EXCEPT [22] was modified which provides exceptions to selected modes where the failsafe is not triggered. In this setting, the offboard mode is ticked as shown in Figure 8. After making this change, the vehicle works as expected and flies to an altitude of 1m.

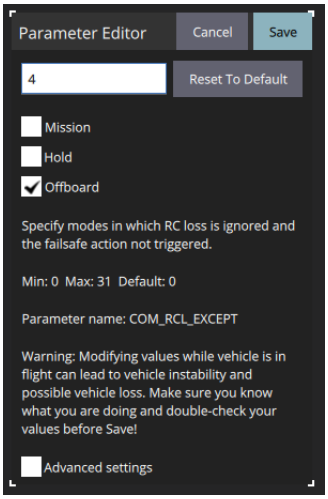


Figure 8: RC Loss Exceptions Settings. Source: Prepared by the author

3.5 Vehicle Test Analysis

The prototype designed for this project has gone through various stages of development and upgrades based on various use case tests conducted by the previous students. As reported by the previous students, none of the

aerial tests were a complete success because of the poor behaviour of the aerial controller. Due to the vehicle not even achieving flight, it was not possible to test the implementation of the path control or obstacle avoidance modules for the aerial path or the safe landing algorithm. Therefore, it is important to understand what went wrong during the course of the development and why it happened. In order to progress further towards the goal, it is also important to understand the insight gained in the last 4 years of the project in order to conduct a thorough test analysis. This section details the tests conducted in the previous years to determine the degree of success as well as identify at what point exactly, the poor behaviour of the aerial controller was observed to identify the potential root cause or causes.

3.5.1 Ground and Aerial Simulations

The simulations conducted were carried out to ensure each module functions properly in a virtual environment before doing the real test. These simulations involve testing the environment perception module to generate an accurate point cloud map and publish it to a topic which is then subscribed by the path-planning module to compute a 3D Octomap and a ground and aerial path, the best one of which is finally implemented by the path-control module.

Environment Perception Module

Two simulated scenarios (a simple scenario and a complex city environment) and a real environment, were used to test the code. A simple scenario was developed in parallel while developing the code consisting of walls and boxes which was then followed by a complex city environment with buildings, roads, cars and other street furniture. Finally, the code was tested in a real environment using a collection of stereo images containing hundreds of images in different environments (rural areas, cities, motorways etc). All the three test cases produced good results – object detection and depth estimation results for the point cloud map were better than expected.

Path Planning Module

Three use cases for the testing the path planning module were carried out. The first use case consisted of a simple environment consisting of a box placed in front of the vehicle. The objective of the vehicle was to compute a path to reach a position behind the box. The second use case creates an environment of higher complexity with a series of obstacles through which an optimal path must be found. Finally, the third use case consists of an environment where only an aerial path is viable, thereby, forcing the path planner to compute an optimal one. In all these tests, the path planner successfully generated the most optimum and logical path. The last part of the aerial path where the vehicle is supposed to land was not computed just yet as the ground path and aerial path could not be calculated simultaneously. Figure 9 shows the simulation result of the simplest first case.

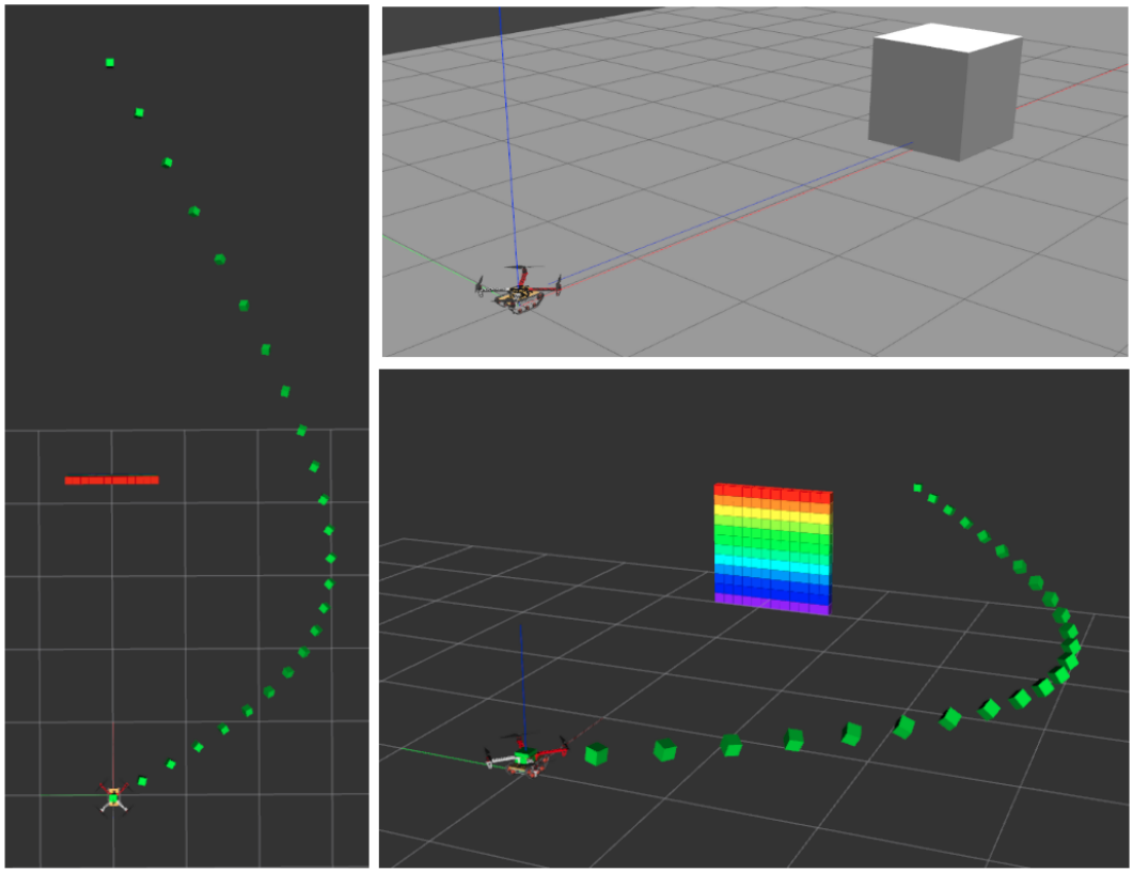


Figure 9: Path Planning in a Simple Environment. Source [13]

3.5.2 Ground and Aerial real tests

The real tests were performed with the aim to replicate everything observed in the simulations. The tests have mainly evaluated the performances of the Environment Perception, Path Planning and Control Modules. Out of these modules, the Control Module's results were mixed, the ground module worked as expected, following the calculated trajectory without any problem, considering the perturbations and dynamics of the prototype. However, the air module was not stable enough to follow a list of way-points safely.

Two flight tests were conducted to test the aerial module. The first test mainly focused on communication between the Raspberry Pi and the Pixhawk flight controller which was working properly. It is important to note at this point that the sensors had not been mounted yet, so the weight of the vehicle was not the final weight. Although considering the objective of the test, which was to establish the connection of Pixhawk and Raspberry Pi, it was a success, the main problematic observations reported were a high level of nominal thrust of 80% along with vertical oscillations about the target altitude. Figure 10 and Figure 11 show the thrust and altitude variation plots of the first test respectively. It can be seen that the thrust sporadically hits the 100% mark and saturates which results in altitude oscillations of the vehicle.

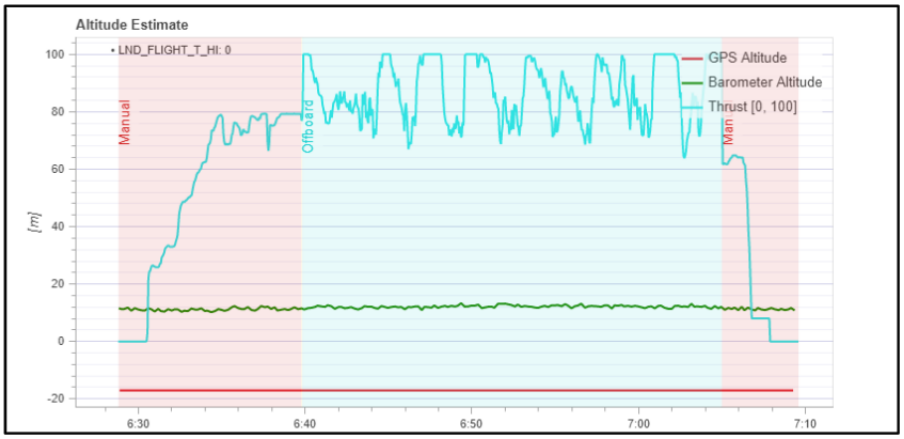


Figure 10: Thrust Variation of the first test. Source: [1]

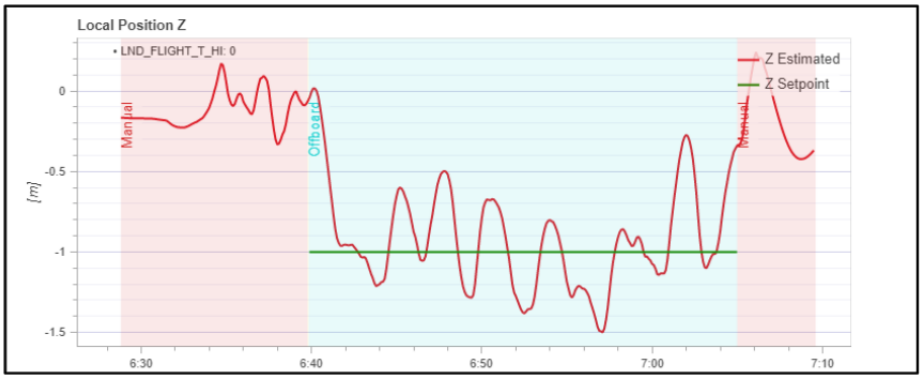


Figure 11: Altitude Plot of the first test. Source [1]

The second test was aimed to rectify these errors and test the other modules. The sensors on the vehicle were added and the Environment Perception and Path Planning Algorithms were tested. They were able to work as expected – the EPM successfully detected the obstacle's shape as well as its distance and the PPM was able to compute a valid path for the vehicle to follow. However, the aerial control module was never tested as the vehicle did not even achieve lift-off. The reported thrust saturated at 100% before reaching the intended altitude.

3.5.3 Hypothesis

After studying the real tests, two things are important to note. Firstly, the first test was conducted at a lower weight (sensors and their holders were not added yet) and secondly, the default gains available in Pixhawk were used for the test. This meant that an untuned vehicle which was not at the final weight was tested. These two factors contributed to the vertical oscillations of the vehicle. Regarding the problem of the near maximum thrust of 80% being used just for hovering, it suggests that the propulsion system needs some kind of improvement. This problem became even more obvious after the second test where the vehicle did not even generate enough thrust to achieve lift-off. As part of an effort to rectify these problems, the vehicle was first tuned at the correct weight of **1.85kg** and simulated. The test case used was the same so as to replicate the test conducted before where the vehicle is supposed to move vertically and hover at an altitude of 1m. The offb_node.cpp node is used for tuning the vehicle. The results of these simulations are explained in Section 4.1.

Power and thrust demands vary significantly during the course of the flight time of a vehicle. These constraints depend not only on the payload weight but also on the distance to fly and type of maneuvering required in order to avoid obstacles or reach specific way-points. For this reason, in general, it is recommended to design your propulsion system consisting of motors, propellers and the battery in such a way that it is able to generate thrust at a safe margin from the maximum thrust capacity. A safe margin of 50-60% thrust during hovering is generally recommended. This range provides confidence that the vehicle would be capable of successfully executing complex and power-expensive requirements. During the transition phase from ground to an altitude

of 1m, the vehicle is supposed to move upwards which means it has to generate thrust more than the weight of the vehicle. In the second test that was conducted, the maximum thrust capability coupled with the vertical velocity demands of the PID increased the required transient thrust to above 100%. This is the reason why the vehicle did not takeoff. Since the thrust margin reported by the previous students is 80% of the maximum thrust capacity [1], two solutions are proposed to increase this margin.

Propeller Optimization

One of the solutions focusses on optimizing the propellers of the aerial module. One advantage of this is that changing the propellers does not require a radical change in the design or any significant overhaul of the vehicle. As mentioned in Section 3.1.1, the aerial module is equipped with four 2212 920kV brushless DC motors, a 3S, 11.1V, 3000mAh, 30C LiPo battery and four propellers of type 9450. In order to generate thrust with the same motor and battery configuration, one would have to install larger propellers. Large propellers have more surface area which generates more lift force at a lower RPM. However, large propellers draw more power due to their increased mass and inertia which acts as a greater resistive load for the brushless DC motors. Equation 1 establishes a basic relation between the thrust(T) generated by a propeller for a given amount of power(P) supplied to it. Thrust also depends on other factors such as the efficiency(η)of the motor, the density(ρ) of the surrounding air and the surface area(S) of the propeller.

$$T = \eta(2\rho S P^2)^{\frac{1}{3}} \tag{1}$$

Most manufacturers share thrust performance data of different brushless DC motors with different configurations of propellers. This allows consumers to select the best design according to their thrust requirements and battery capacity constraints. Figure 12 provides this thrust performance data for the 2212 920kV DC motor used in this project.

ML2212 MOTOR

Item NO.	Volts (V)	Prop	Throttle	Amps (A)	Watts (W)	Thrust (g)	Efficiency (g/W)	Operating temperature(℃)
ML2212 920KV	11.1V	DJI9.4*4.3	50%	1.8	20.0	230	11.5	37℃
			65%	2.8	31.1	310	10.0	
			75%	3.9	43.3	410	9.5	
			85%	5.5	61.1	480	7.9	
			100%	7.6	84.4	610	7.2	
		APC10*4.5	50%	2.6	28.9	290	10.0	55℃
			65%	5.1	56.6	460	8.1	
			75%	7.4	82.1	590	7.2	
			85%	10.1	112.1	730	6.5	
			100%	13.4	148.7	860	5.8	
	14.8V	DJI9.4*4.3	50%	2.7	40.0	350	8.8	52℃
			65%	4.4	65.1	490	7.5	
			75%	6.3	93.2	640	6.9	
			85%	8.3	122.8	790	6.4	
			100%	11.5	170.2	990	5.8	

Notes: The test condition oftemperature is motor surface temperature in 100% throttle while the motor run 10 min.environment temperature 24℃

Figure 12: Thrust Performance Data of the 2212 920kV motor

The total weight of the vehicle is distributed among the 4 propellers. This means that each propeller needs to generate at least a quarter of the weight. This value comes up to be 462g of thrust. At the voltage of 11.1V, which is the voltage supplied by our battery, the data of two propellers is given. The first propeller DJI 9443, which is sufficiently close to the 9450 propeller (pitch is 4.3° instead of 5°), has a maximum thrust capacity of 610g and it can deliver 462g at approximately 80% thrust capacity which agrees with the current findings from the test. For this weight, it also draws a current of approximately 5A.

The second propeller, APC 1045 with larger propeller diameter, is capable of generating a maximum thrust of 860g. This propeller delivers a thrust of 462g at approximately 67% thrust capacity at about 5.2A. This thrust margin is an improvement and should suffice the transient thrust requirements during climbing. One major concern with this propeller is that the operating temperature is much higher at 55°C, however, for a

functioning prototype where all use case tests would involve relatively short flight times, this temperature will not pose a major problem. One can argue that taking even larger propellers would reduce the margin even further but other propellers may not be compatible with the current motor and battery configuration where they may start posing large inertia constraints which may end up increasing power consumption. Moreover, if the tips of propellers are within a certain distance, their airflow dynamics start affecting each other which would reduce efficiency. With the APC 1045 propeller, the distance between the tips of adjacent propellers is 6.97cm which is large enough to not cause this effect. Additionally, the distance between the propeller tip and the body of the model, particularly the spherical markers which are housed on top of the vehicle and protrude out the most, is 2.93cm.

Weight Reduction

Changing an element of the vehicle with a lighter yet effective option is another way to reduce the thrust requirements as this reduces the main payload weight. The heaviest parts of the vehicle are the two batteries. The 3000mAh LiPo battery weighs 400g and powers the Pixhawk, the DC motors as well as the ground motor. Since this battery is connected with and responsible for several components, finding an alternative and connecting it may prove cumbersome. On the other hand, the 10000 mAh battery weighs 175g and has extra battery capacity which is unnecessary considering it powers only the Raspberry Pi which does not draw a lot of amperage. A battery with a high battery capacity is also quite heavy, therefore, removing this battery and finding another alternative which can adequately power the Raspberry Pi would help reduce the thrust requirements.

One alternative is to connect the Raspberry Pi to the 3000mAh aerial battery which would bring all components under a single power source and remove the need for a second battery. This is possible by connecting the Raspberry Pi to the Power Distribution Board that distributes power to the Pixhawk and the Ground Motor. Table 2 shows the different power requirements of all the components.

If the power bank is removed which weighs 175g, it would bring the total weight of the vehicle down to 1.675kg, so each propeller would be required to generate 418.75g of thrust which corresponds to about 74% thrust capacity if the 9450 propellers are used. This thrust would require approximately 43.3W of power. The voltage supplied to the ground motors has been regulated through a potentiometer which brings the voltage down to 6V from the 11.1V supplied by the battery [13]. According to the different power requirements of the components and the battery capacity, we can calculate the battery life which will give us an idea about the average mission time for which we can expect the vehicle to function.

Component	Power Required
Aerial Motors	4*43.3 = 173.2W
Pixhawk	∼2W
Raspberry Pi	5V*2A = 10W
Ground Motors	6V*0.5A*2 = 6W

Table 2: Power Requirement Breakdown. Source: Prepared by the author

$$TotalPowerConsumption = 191.2 \tag{2}$$

$$BatteryCapacity = 3Ah * 11.1V = 33.3Wh \tag{3}$$

$$BatteryLife = BatteryCapacity/PowerConsumption = 33.3Wh/191.2W \approx 10.5minutes \tag{4}$$

Through these calculations, we see that a power distribution system with a singular power source of the 3000mAh, 11.1V LiPo battery would last for about 10.5 minutes before the battery runs out. For the purposes of our flight tests, this time is more than sufficient. The vehicle has been simulated in ROS-Gazebo with the new reduced weight to confirm the results of these calculations which is discussed in Section 4.3.

3.5.4 Simulation Attempts with the 1045 Propellers

It is always important to back research up with precise and accurate simulations before testing your hypothesis through experiments. To accurately model the 1045 propellers in the Gazebo simulation, changes had to be made to the current existing model definition, particularly, the propellers. The model created by the previous

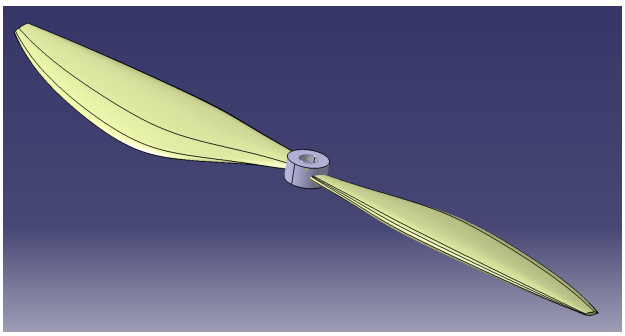
students [12] and used in this project is defined in a single file called dd_bot.sdf which defines the different dimensions of the ground module such as the main body which is approximated with a box and wheels. Mass and inertia properties are defined for each component which corresponds to those of the actual vehicle and each component has to have a link defined so as to integrate it to the dynamics of the entire vehicle [23].

Pixhawk already has existing models of various drones and planes which can be used or modified to fix the custom design requirements and simulated. Therefore, the aerial module is incorporated to the vehicle by taking an existing model of an iris drone and attaching it with a link on top of the main body of the ground module. The iris model defines its propellers using properties of mass and moment of inertia along with the direction of rotation. The code snippet for this definition is shown in Figure 13. The pose Tag refers to the location of the propeller in space.

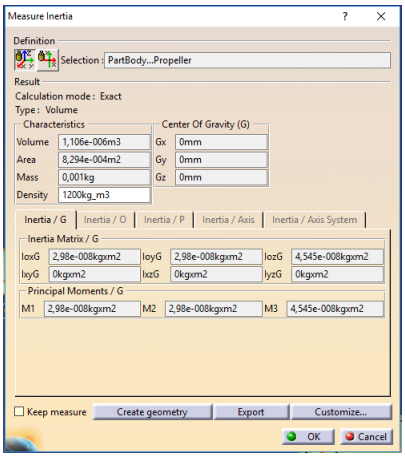
```
<!-- Propeller Definition -->

    <link name='rotor_1'>
      <pose>-0.13 0.2 0.023 0 0 0</pose>
      <inertial>
        <pose>0 0 0 0 0 0</pose>
        <mass>0.005</mass>
        <inertia>
          <ixx>9.75e-07</ixx>
          <ixy>0</ixy>
          <ixz>0</ixz>
          <iyy>0.000273104</iyy>
          <iyz>0</iyz>
          <izz>0.000274004</izz>
        </inertia>
      </inertial>

<!---------------------------->
```

Figure 13: Code Snippet of the Propeller Definition. Source: Prepared by the author

The mass of the APC 1045 propeller is provided by the manufacturer depending upon the material that is used. In order to find the moment of inertia of this propeller, the same was modelled in CATIA which provides the values after selecting the right material. The generated model along with the procured values of moment of inertia are shown in Figure 14.



(a) CATIA Model



(b) Moments of Inertia

Figure 14: CATIA model and MOIs of the 1045 propeller. Source: Prepared by the author

After retrieving the moments of inertia of this propeller, these values were edited into the model definition code. In order to test basic liftoff, the offb_node.cpp node is used which brings the vehicle to an altitude of 1m. Unfortunately, the results are not as expected since the vehicle becomes extremely unstable and fails to takeoff. Since the only change made was in the propeller definitions, it is clear that simulating the propeller dynamics requires further research on the method to integrate it to the model.

3.6 Hybrid Module Path Optimization

As mentioned in Section 3.3, two different approaches have been taken to develop a path selection algorithm for the hybrid module. These approaches work in some use case scenarios but not all. The first approach involving the use of the path lengths and a power consumption factor works well for simple cases where the final coordinates are within the perception range of the sensors so that the sensors can pick up each and every obstacle that might be in the way. However, in reality, final coordinates much further away could be selected in which case calculating the length of the paths would not be possible taking into account all the obstacles since the sensors will not be able to detect them. Moreover, due to the range of the environment perception having limits, these additional obstacles could complicate the path of the vehicle which would require additional power. Thus, predicting and fixing a power consumption factor would not be ideal.

The second approach which involved only looking at the z coordinate of the destination is too simplistic. Essentially, it only provides us with information of whether the aerial module is required or not which would not always provide an optimized path resulting from module switching.

The two parameters to be taken into account for path optimization are energy consumption and time. The two modes that the vehicle employs to navigate through an environment always have one parameter being consumed at the cost of another. The aerial mode consumes less time but at the cost of energy and the ground mode does the opposite. This vehicle is planned to be deployed in unknown environments with uncertain conditions where it will be required to complete missions within a strict timeline. Since it will be working as a real time system where it navigates through a path after destination coordinates have been provided to it, its mission will only be considered to be accomplished if it completes it within this deadline, otherwise it is a failure. The unknown environment of the vehicle makes it difficult to predict how much energy could be consumed to avoid obstacles and follow the most logical path. Therefore, it makes sense to minimize energy consumption while ensuring that the mission is completed within the given deadline. Minimizing energy will allow the vehicle to handle unforeseen situations as well using the energy it has conserved, for example, if wind conditions change or turbulence increases while using the aerial module.

Due to the differences in ROS versions, it was decided to focus on optimizing the code written in ROS melodic [1] first of the hybrid module. The previous students had taken the z coordinate being above or below the value of 0.4 as the criteria to switch to the aerial module. Since this does not take into account energy or time, it is necessary to optimize this condition. The hybrid module has, thus, been developed further in increasing levels.

3.6.1 Adding the Flight Domain

The first level incorporates a logic to minimize power consumption and adhere to the deadline. Since energy is to be minimized, the mission almost always starts with the ground module and switches to the aerial module if there is not enough time to reach the goal through just the ground module. In other words, the vehicle by default uses the ground module unless other criteria force it to switch to the more power expensive aerial module. These criteria can be the time limit or certain obstacles that cannot be avoided by the ground module alone.

The point in the path where it is most optimum to switch to the aerial module depends on a number of things such as the distance between the initial and final coordinates, the speeds of the ground and aerial modules and the deadline imposed on the vehicle to complete the mission. Taking all these factors into account, the concept of a flight domain sphere has been developed. Simply put, a spherical domain is defined at the start of every mission and the vehicle must fly if it finds itself inside this domain while it should use the ground module to traverse the path if it is outside this domain. The only exception would be when the vehicle encounters an obstacle which it cannot avoid without the aerial module, in which case, it will fly over this obstacle and switch back to the ground module after landing and continue on its path. The radius of this domain continuously increases with time. In essence, it represents that as time progresses, it becomes more and more urgent to complete the mission by using the aerial module. The rate of increase of this domain is the parameter that is calculated through the mission time, final coordinates and the air and ground velocities. Figure **??** shows a schematic of the concept of the flight domain. Table 3 shows some variables defined for the calculations of the rate increase.

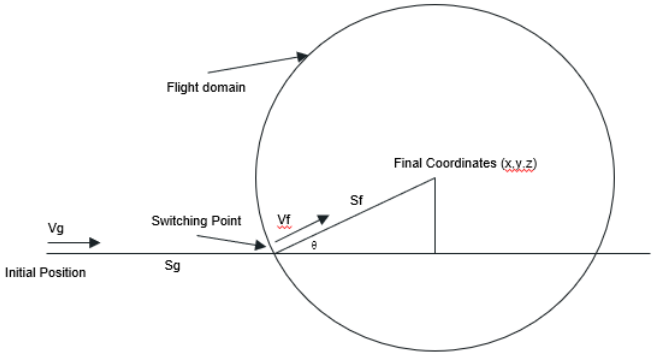


Figure 15: Switching in the Flight Domain. Source: Prepared by the author

Variable	Parameter
S_g	distance covered by the ground module
S_f	distance covered by the aerial module
V_g	ground module velocity
V_f	flight module velocity
T	Mission Time
x	Final x coordinate
y	Final y coordinate
z	Final z coordinate
θ	vehicle angle of climb
r	flight domain growth rate

Table 3: List of Parameters. Source: Prepared by the author

For the first stage of development, the vehicle is assumed to complete the mission at the worst possible time which means it would reach the final coordinates at the mission time T provided. This assumption yields the following equation.

$$GroundTime + FlightTime = MissionTime \tag{5}$$

$$\frac{S_g}{V_g} + \frac{S_f}{V_f} = T \tag{6}$$

Equation 6 is the time constraint that the vehicle must obey. The vehicle will start flying just when it enters the flight domain. Therefore, the time taken by the vehicle to cover a distance S_g should be the same as the time taken by the spherical flight domain to grow to a radius of S_f.

$$\frac{S_g}{V_g} = \frac{S_f}{r} \tag{7}$$

From equations 6 and 7, we have the equation of the growth rate of the flight domain.

$$r = \frac{1}{\frac{T}{S_f} - \frac{1}{V_f}} \tag{8}$$

Here, S_f is the unknown and must be eliminated. We have the geometry relation

$$\sqrt{S_f^2 - z^2} + S_g = \sqrt{x^2 + y^2} \tag{9}$$

Using equations 6 and 9, we can solve for S_f and obtain the long relation.

$$S_f = \frac{V_f\left(TV_g^2 - V_g\sqrt{x^2+y^2} + \sqrt{x^2V_f^2 + y^2V_f^2 + z^2V_f^2 + T^2V_f^2V_g^2 - 2TV_f^2V_g\sqrt{x^2+y^2} - z^2V_g^2}\right)}{V_g^2 - V_f^2} \tag{10}$$

After using equation 10 in equation 8, the final growth rate is calculated. The growth rate of this flight domain is a function of the mission time T and the final coordinates x, y, z. It can be seen that the growth rate would be lower if the mission time is longer, which corresponds to the actual situation that the vehicle would be in, it would have more time to complete the mission, so it should make the decision to switch to the aerial module much later. There can also be cases where a large mission time is provided in which the vehicle may drive all the way to the coordinate directly below the final one and then just fly vertically up. In cases of final coordinates on the ground where a short mission time is provided, the vehicle will be forced to fly over to the coordinates and land. Several use cases have been simulated and their results are discussed in Section 4.

3.6.2 Switching Time

After this initial base logic was coded into the algorithm and tested, it was found that the vehicle takes approximately 18 seconds to actually switch from the ground module to the aerial module. The delay varies slightly depending on the processing speed of the system. This time was measured by carrying out a simple simulation of the vehicle using the **offb_node.cpp** node where the vehicle flies vertically up and hovers at a 1m altitude. The time from when the node runs till the vehicle is armed was measured to be 18 seconds. During this period of time, the ground module is deactivated so the vehicle remains stationary. This time is essentially wasted as the vehicle does not make any progress towards the goal. Therefore, as part of the second level of improvement, this switching time is taken into account so the vehicle makes a more seemless transition from the ground module to the aerial module. In order to save this time, the aerial module is triggered 18 seconds before the vehicle is actually expected to take off. As soon as the vehicle goes into offboard mode and is armed, the ground module is also deactivated which is just when the vehicle is about the takeoff. This removes the idle time completely when the vehicle is not moving. The results of these simulations are discussed in Section 4.5.

3.6.3 Safety Takeoff and Landing Protocol

The Pixhawk which is connected to the vehicle uses a PID controller to control the aerial module. This PID controller takes the final coordinates as the target to reach and takes the vehicle along the shortest possible path, which is a straight line, to it. This may not be the most optimum or safest path to take for the vehicle depending upon the nature of the obstacles around it. If the vehicle stops right next to an obstacle, the PID controller could cause it to crash into the obstacle after takeoff. The vehicle can also experience skidding as it gains altitude. This is especially true when the final coordinates are on the ground since the PID controller sees no reason to increase the altitude.

In order to prevent all these cases, a safety takeoff and landing protocol has been incorporated into the hybrid module as part of the next level of development. Under this protocol, as soon as the aerial module is activated and the vehicle is set to fly, the first target the vehicle reaches is an altitude of 1m from whatever its current position is. In case, the destination coordinates are above the ground, it then simply flies to it. If in case the destination coordinates are on the ground, the next target the vehicle reaches is directly above these coordinates again at an altitude of 1m which is finally followed by landing at the target and the aerial module is deactivated. Therefore, the vehicle must traverse a rectangular path in this case. Several simulations were conducted to test a number of these use cases the results of which are shown in Section 4.6.

4 Results and Analysis

4.1 Simulation 1 - Tuning the Vehicle

As mentioned in Section 3.5.3, it is necessary to tune the vehicle at the right weight. The previous students [1] have reported PID gains that have been simulated with minor thrust saturations. The purpose of this simulation is to verify that these PID gains also produce acceptable performance results at the final weight, that is, 1.85kg. Table 4 shows the PID gains reported by the previous students and Figure 16 shows the tuning setup of Pixhawk in QGC.

Rate Controller	
MC_ROLLRATE_K	2.1
MC_ROLLRATE_I	0.2
MC_ROLLRATE_D	0.01
MC_PITCHRATE_K	2.4
MC_PITCHRATE_I	0.2
MC_PITCHRATE_D	0.01
MC_YAWRATE_K	0.8
MC_YAWRATE_I	0.16

Attitude Controller	
MC_ROLL_P	5.5
MC_PITCH_P	4.5
MC_YAW_P	1.4

Velocity Controller	
MPC_XY_VEL_P_ACC	4
MPC_XY_VEL_I_ACC	3.4
MPC_XY_VEL_D_ACC	0.6
MPC_Z_VEL_P_ACC	6
MPC_Z_VEL_I_ACC	1.05
MPC_Z_VEL_D_ACC	0.45

Position Controller	
MPC_XY_P	1.05
MPC_Z_P	1.15

Table 4: PID gains. Source: [1]

Figure 16: PID tuning setup in QGC. Source: Prepared by the author

The offboard node mentioned in Section 3.4 is used here to test the performance of the vehicle. Figures 17 and 18 show the vehicle performance for the PID gains mentioned in Table 4. These graphs have been extracted through the PX4 Flight Review Tool [24]. It is clear from these graphs that the PID gains are acceptable for this weight as no oscillations are observed in the altitude and there is no thrust saturation. One can also see

that the thrust capacity of 80% is apparent from Figure 18. This high thrust capacity only allows for slow maneuvers, otherwise the thrust will saturate, which can be seen in the relatively slow response time of the altitude curve in Figure 17.

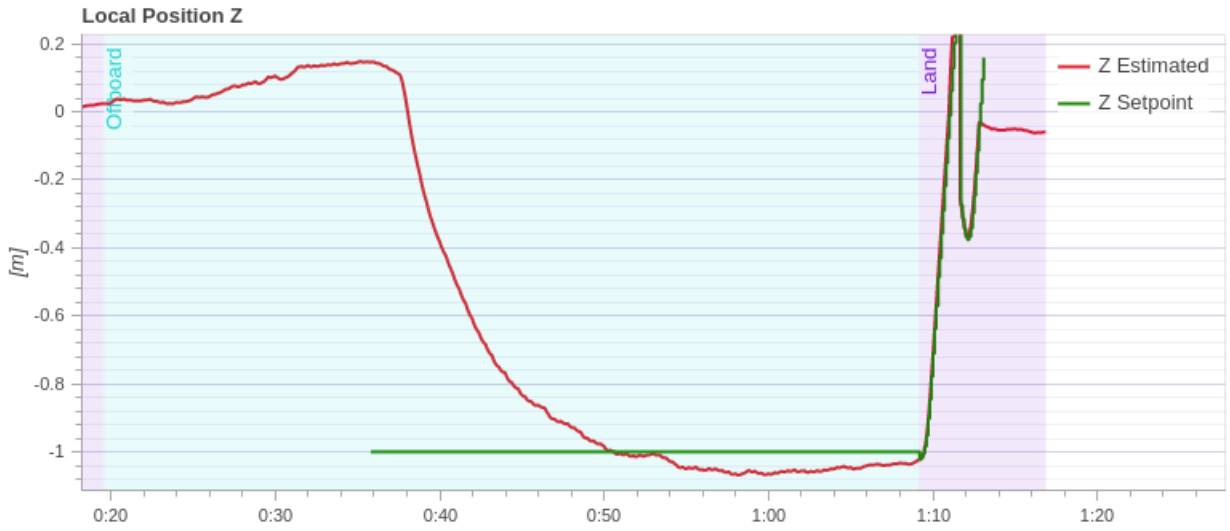


Figure 17: Altitude Plot for the tuned PID gains. Source: Prepared by the author

Figure 18: Thrust Plot at 1.85kg of weight. Source: Prepared by the author

4.2 Simulation 2 - Testing the Propulsion System

The thrust margin even with tuned PID gains and the right weight is not adequate. This thrust allows the vehicle to make only simple and slow maneuvers where the thrust does not saturate. The result is a very slow vehicle which cannot perform if a more complicated trajectory is demanded. In order to visualize this disadvantage, a more demanding trajectory is tested on the vehicle to see the thrust saturating. For this purpose, the node air_ground_hybrid_module.py is used which is developed by the previous students [1]. This node implements the logic of switching between the modules based on the value of the z coordinate as explained in Section 3.3. This node asks the user for the destination coordinates which is then tracked by the vehicle. The following coordinates are input in succession to create the required trajectory - (1,1,1),(-1,-1,2),(1,1,2),(0,0,1),(0,0,0). After the last coordinate, the vehicle lands back to the origin. Figures 19 and 20 show the resulting thrust plot and the battery consumption rate for this case.

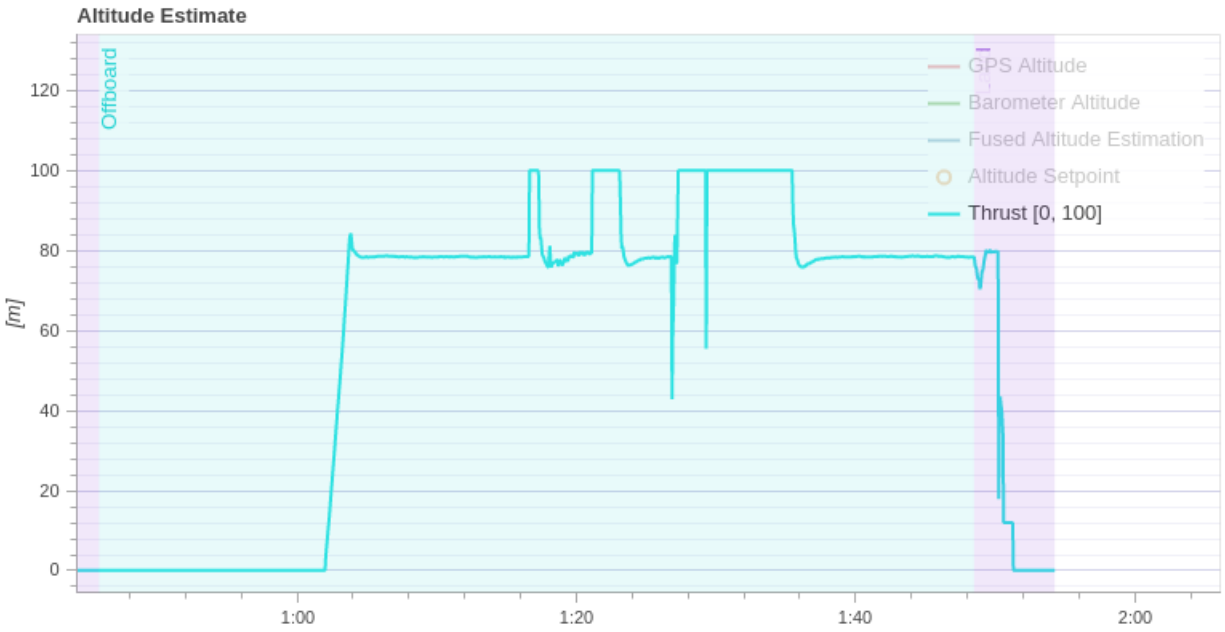


Figure 19: Thrust Plot for a complicated Trajectory. Source: Prepared by the author

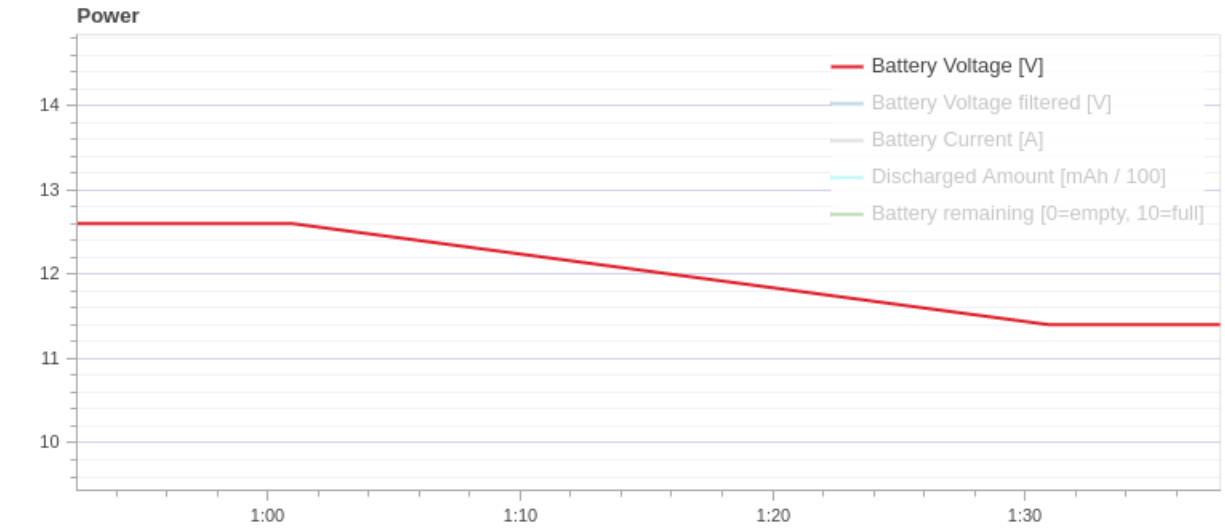


Figure 20: Battery Level for a complicated trajectory. Source: Prepared by the author

According to the graphs, it is obvious that the current propulsion system cannot handle the thrust requirements that can be imposed on the vehicle. The thrust saturates multiple times during the simulation and the battery level drops by around 2V in just 30 seconds. Figure 21 also shows the altitude plot for this trajectory which shows that the vehicle is clearly unable to track the target set-point. This necessitates the need for a higher thrust margin.

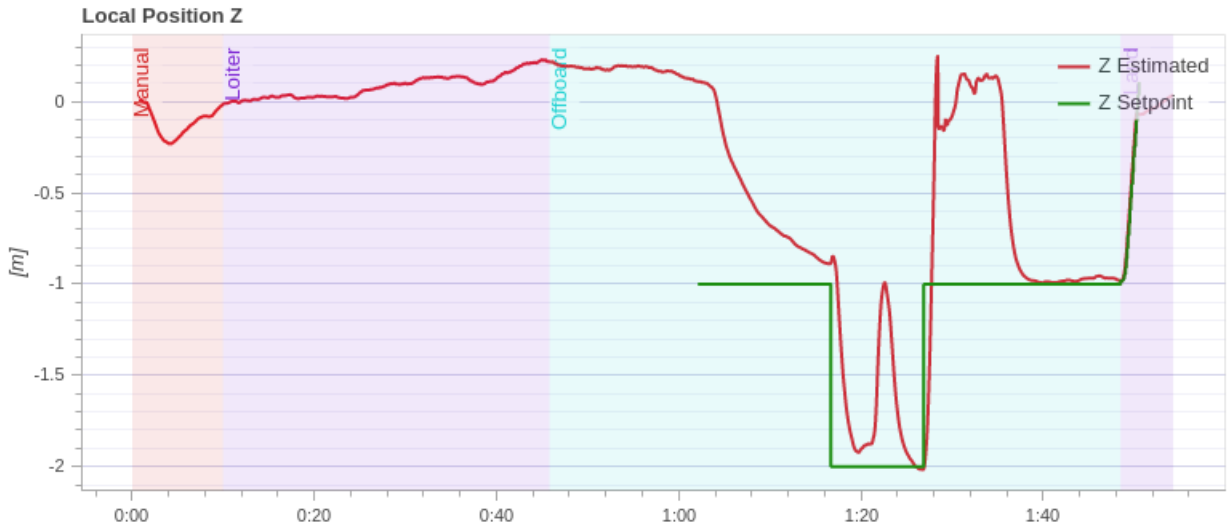


Figure 21: Altitude Plot for a complicated trajectory. Source: Prepared by the author

4.3 Simulation 3 - Weight Reduction

There are two solutions proposed to solve the problem of the weak propulsion system discussed in Section 3.5.3. The first solution which deals with using an alternate propeller could not be simulated since the new propeller could not be coded into the model definition. The second solution, however, was successfully simulated as the weight of the entire model can be changed easily. Therefore, the model weight was reduced to 1.675kg after removing the weight of the power bank which weighs 175 grams. The PID gains mentioned in Table 4 also work with this new dynamic of the model. Figure 22 shows the reduced thrust capacity as the weight is reduced. As seen below, the thrust capacity reduces to around 74% since each propeller now has to handle a quarter of the reduced weight which is about 418.75 grams. Although this percentage is still not in the recommended range, it still provides a higher thrust margin than before. It is also important to note that the propellers used for this weight are still the 9450 type propellers. If we use the APC 1045 propellers with this reduced weight, then according the Thrust Performance Data in Figure 12, the required thrust would be at a capacity of around 60% which is a further improvement. Conducting the required simulation with these propellers to confirm this thrust capacity is planned in the next semester.

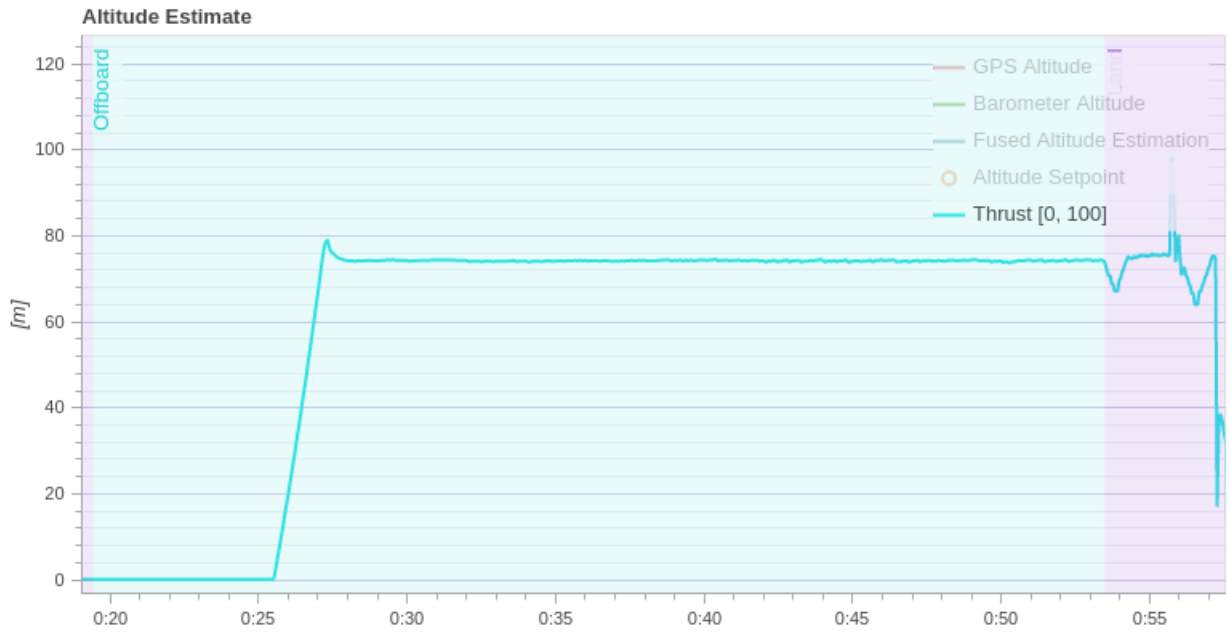


Figure 22: Thrust Plot with Weight Reduction. Source: Prepared by the author

4.4 Simulation 4 - Hybrid Module Optimization

The hybrid module has been explained in Section 3.6 in which the vehicle has to take the appropriate mode of transport in order to save energy and ensure mission completion within the stipulated time frame. The first level of optimization to the existing algorithm developed by the previous students introduces a concept of a flight domain that decides whether the vehicle should fly to the target or continue using the ground module to save energy. Before the mission starts, the user is asked for the final coordinates to be reached by the vehicle as well as the mission deadline in seconds. This part of the algorithm function is shown in Figure 23. The vehicle switches to the air module as soon as it enters the flight domain which means just as its distance to the target coordinates becomes lesser than the radius of the flight sphere. This switch is shown in Figure 24. We can see that when the distance to target becomes less than the flight domain sphere radius, the switch to the air module is made and the offboard mode is triggered.

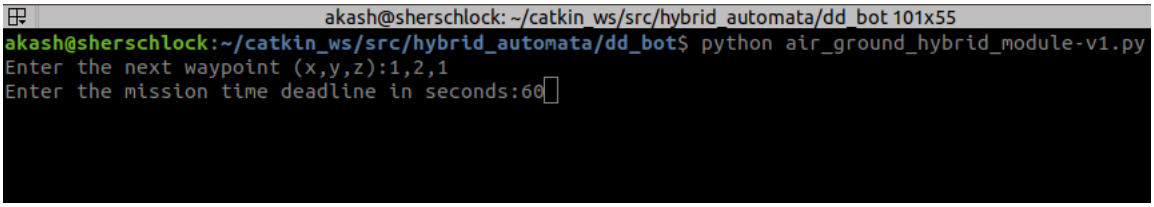


Figure 23: User Input. Source: Prepared by the author

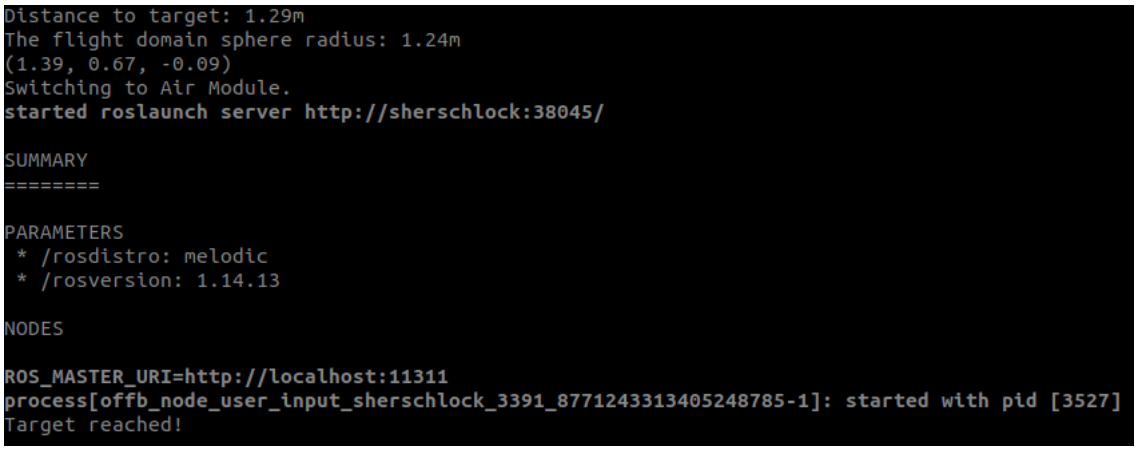
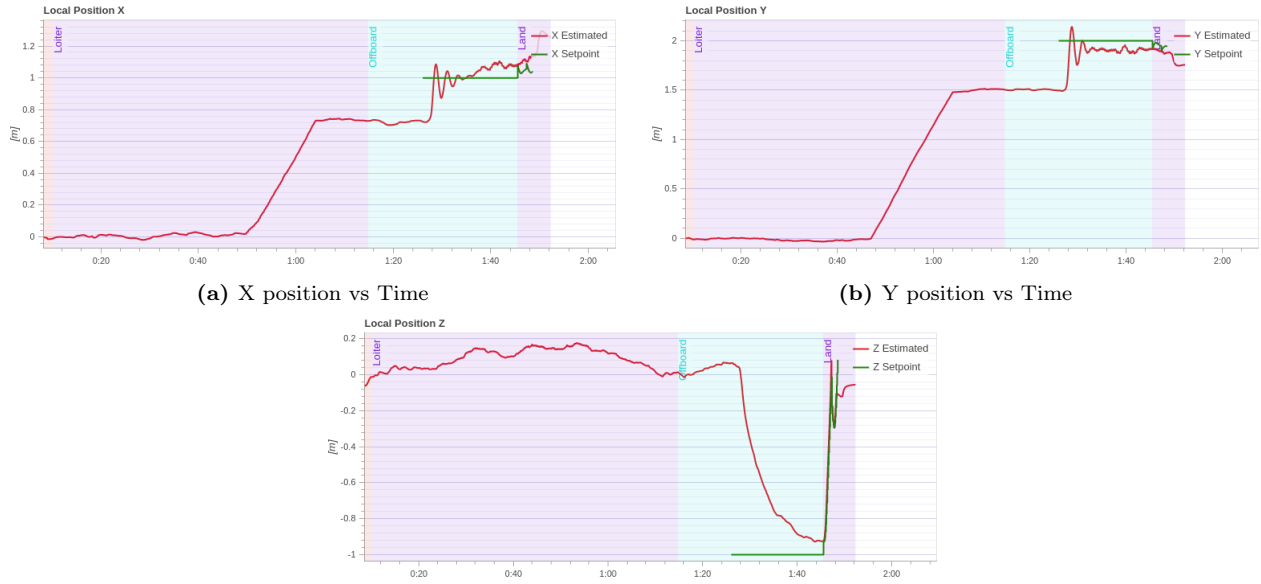


Figure 24: Switching to the Air Module. Source: Prepared by the author

As shown in Figure 23, the final coordinates entered for the simulation are (1,2,1) with a mission time of 60 seconds. The X,Y,Z positions vs time of the vehicle are graphed in Figure 25. The loiter region of the graph represents the part where the vehicle uses the ground module while the offboard region represents the aerial module.



(a) X position vs Time

(b) Y position vs Time

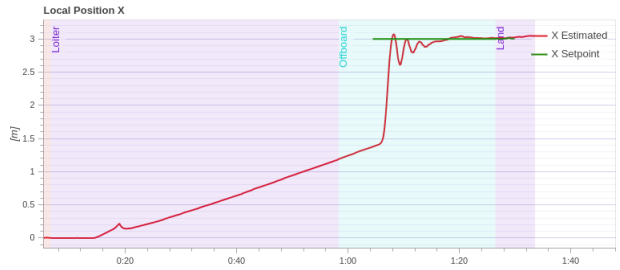
(c) Z position vs Time

Figure 25: Hybrid Module Optimized Trajectory with flight domain. Source: Prepared by the author

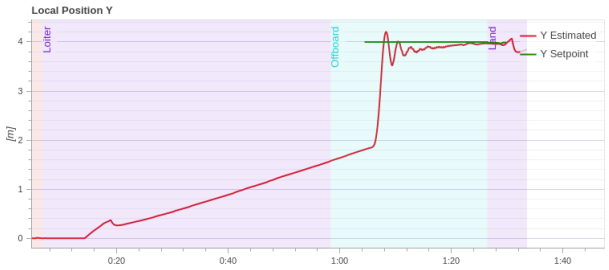
The plots show that even though the final destination is in the air, the ground module is used first so as to prioritize energy conservation until it is time to switch to the air module. It is important to note at this time that in order to increase responsiveness in the horizontal X and Y directions, the gain value of MPC_XY_P was increased to 2 while all order gains are kept the same as in Table 4.

4.5 Simulation 5 - Taking Switching Time into account

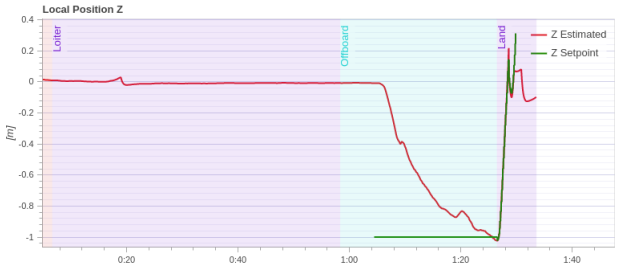
An important observation from the horizontal plots vs time in Figure25 is that both X and Y values plateau at particular values before the vehicle takes off. The X value stays constant at around 0.72 while Y stays at 1.5. The reason the vehicle stays stationary like this is because this is the switching time of approximately 18 seconds that the vehicle takes to carry out the pre-flight checks before enabling the offboard mode and arming the vehicle. This observation is discussed in Section 3.6.2. In order to save this idle time, the aerial module switch is triggered 18 seconds before the actual flight is needed so that the offboard mode can be entered and the vehicle can be armed without stopping while the ground module is still being used. The ground module is deactivated just as the vehicle takes off. The improved trajectory of the vehicle is shown in the X,Y,Z plots vs time in Figure 26. The mission given to the vehicle are the final coordinates of (3,4,1) with a mission time of 60 seconds.



(a) X position vs Time (smooth switch)



(b) Y position vs Time (smooth switch)



(c) Z position vs Time (smooth switch)

Figure 26: Hybrid Module Optimized Trajectory with a smooth switch. Source: Prepared by the author

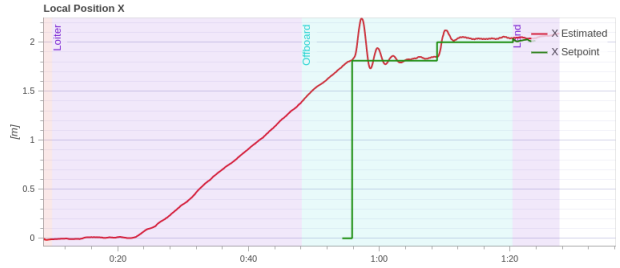
The new plots show that the X and Y positions never stop increasing and finally reach the target after the vehicle starts flying, the vehicle never idles.

4.6 Simulation 6 - Safety Take off & Landing Protocol

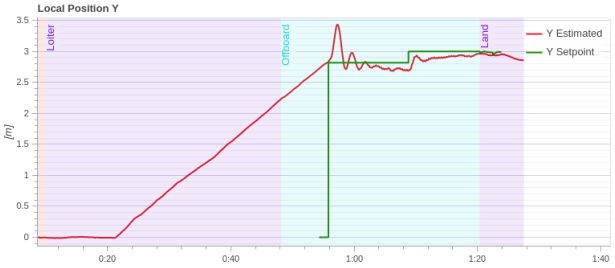
A safety protocol has been added to the optimization algorithm to ensure that the vehicle is safely away from the ground and from other potential obstacles before it starts chasing the target. The protocol works slightly differently depending on whether the destination coordinates are in the air or on ground. Therefore, two different use cases have been simulated to test the effectiveness of the algorithm in both cases.

4.6.1 Aerial Destination Coordinates

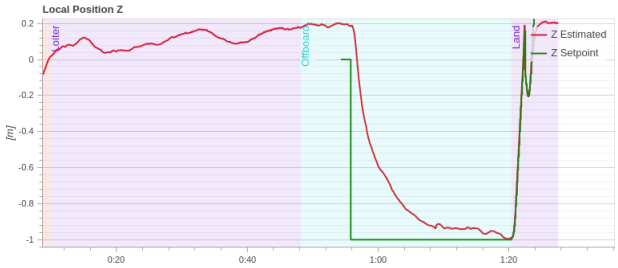
The trajectory to be mapped by the vehicle in this case should be to take off first followed by simply flying to the target. A mission with final of coordinates of (2,3,1) and mission time of 60 seconds has been simulated and the results of X,Y,Z plots vs time are shown in Figure 27.



(a) X position vs Time (SP - aerial coordinate)



(b) Y position vs Time (SP - aerial coordinate)



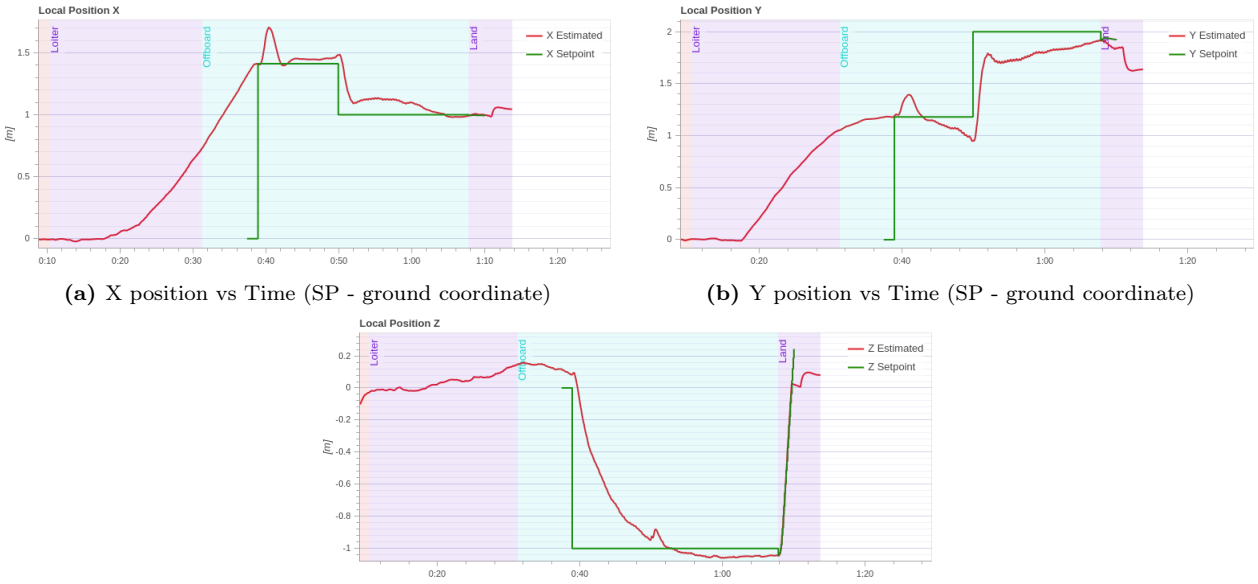
(c) Z position vs Time (SP - aerial coordinate)

Figure 27: Hybrid Module Optimized Trajectory with Safety Protocol with an aerial coordinate. Source: Prepared by the author

The vehicle uses the ground module from the beginning until the vehicle switches to the aerial module. The difference here from the previous cases is that the vehicle first targets an altitude of 1m at the point where the vehicle was armed. This is followed by the vehicle updating its target to the original destination coordinates. In this case, since, the final destination coordinates are also at the same altitude of 1m, we do not see a second step input for the Z plot.

4.6.2 Ground Destination Coordinates

The case when the final coordinates are on the ground is slightly more complicated since the vehicle has to first reach the point directly above the target coordinates (after safely taking off) and then land. This would result in a rectangular path traced by the Z coordinate of the vehicle. The results for this use case are shown in Figure 28. The target coordinates are (1,2,0) with a mission time of 60 seconds.



(a) X position vs Time (SP - ground coordinate)

(b) Y position vs Time (SP - ground coordinate)

(c) Z position vs Time (SP - ground coordinate)

Figure 28: Hybrid Module Optimized Trajectory with Safety Protocol with a ground coordinate. Source: Prepared by the author

We see that the path traced by the vehicle is as expected. Even though the final coordinate is on the ground, the vehicle takes off to a safe altitude and moves horizontally before landing at the final target coordinates.

5 Conclusion and Work to be Done in Semester 3

An unconventional concept of a hybrid autonomous drone certainly brings a piece of novel technology to the industry. Developing this concept has been a challenging task as there are a large number of hurdles to overcome requiring the collaboration of different engineering principles working together. This project has made substantial progress over the past 4 years in which different algorithms were developed and have reached a fair amount of maturity. These algorithms were tested in simulations with a hybrid model as well as in real tests with an actual model. This is where the project was handed over this year where several areas still required further development or optimization.

The first task which was addressed was the problem of the aerial module of the vehicle which was not able to fly. This was, apparently, due to the high weight and thrust saturation and vertical oscillations that were observed in the tests. In order to solve this, a complete test analysis was conducted of all the past tests carried out by the previous students. These tests revealed that the vehicle was not tuned at the right weight which had led to the oscillations. As for the increased thrust capacity, the current propulsion system of the the battery, motors and propellers was investigated and two solutions were proposed. The 2 ways to solve this problem are either to reduce the weight of the vehicle or increase the thrust capacity of the current propulsion system. The weight can be reduced by removing one of the batteries powering the Rasperry Pi and having it receive power from the other battery directly. This reduced the thrust capacity to 74% from the previous 80%. The second solution deals with the type of propellers used. Larger propellers are proposed which can generate the same thrust at a lower RPM. For the weight of the vehicle, APC 1045 propellers were proposed which bring the thrust capacity down to about 67%.

The second task involves the hybrid module responsible for switching between the aerial and ground modules. It is critical for the vehicle to minimize energy consumption while still completing the mission successfully within the stipulated deadline. An algorithm was designed and optimized which allows the vehicle to make such decisions. This algorithm defines a flight domain in which the vehicle must use the aerial module in order to fly quickly to the destination coordinates. Several levels of improvement were also later added to this algorithm where the vehicle is able to save additional idle time that was previously being wasted while the vehicle makes the switch between the modules. Additionally, a safety takeoff and landing protocol was also incorporated for a safer and more realistic real world scenario simulation.

This project utilizes several platforms for simulation purposes. ROS has been implemented which is ideal for custom drone development and Pixhawk has also been used for the PID flight controller for the aerial module of the drone. The results achieved with all these development tools have yielded considerable progress as well as helped lay a much firmer foundation about the different engineering perspectives and principles that are required for the creation of this vehicle. This foundation will be used to continue the progress in the next semester where a new list of objectives will be set and achieved.

- As the algorithms implemented in this vehicle grow in maturity, it becomes more and more cumbersome to handle and understand the logic. Thus, the first objective should be to make the code more organised. To do this, the ROS package, SMACH [25], will be implemented which is capable of building hierarchical state machines. This will not only make the code more organised but also aid in error debugging.
- More complex use cases involving obstacles will be simulated to study how well the hybrid module functions along with the obstacle avoidance algorithm developed by the seniors.
- All the different use cases must also be tested on the real vehicle. This will only be possible after the weight of the vehicle is reduced or the propellers are modified. Hence, the design of the vehicle will be upgraded.

References

[1] K. Mallabadi and R. Dsouza. *Autonomous robotic aerial vehicle: S3 project report*. March, 2022.

[2] Po-Lung Yu. *Multiple-Criteria Decision Making: Concepts, Techniques, and Extensions*. eng. Vol. 30. Mathematical Concepts and Methods in Science and Engineering. Boston, MA: Springer, 1985. ISBN: 9781468483970.

[3] Kun Zhang et al. "A UAV Autonomous Maneuver Decision-Making Algorithm for Route Guidance". eng. In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 17–25. ISBN: 9781728142784.

[4] Karen Northon. *Mars Helicopter to Fly on NASA's Next Red Planet Rover Mission*. Last Updated: Mar 16, 2020. 2018. URL: shorturl.at/bguNZ.

[5] *NASA's Perseverance Rover Begins Its First Science Campaign on Mars*. 2021. URL: shorturl.at/euxBO.

[6] Leah Crane. *First photo of Chinese Yutu-2 rover exploring far side of the moon*. 2019. URL: shorturl.at/cxGJS.

[7] Kyunam Kim et al. "A bipedal walking robot that can fly, slackline, and skateboard". In: *Science Robotics* 6.59 (2021), eabf8136. DOI: 10.1126/scirobotics.abf8136. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.abf8136. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abf8136.

[8] *Documentation - ROS Wiki*. last edited 2022-05-20. URL: http://wiki.ros.org/.

[9] *Understanding nodes*. URL: shorturl.at/oBHTV.

[10] S. CHAUDHARY and N. RAGHUNATHAN. *Autonomous robotic aerial vehicle: S3 project report*. March, 2019.

[11] A. Sambalov. *Autonomous robotic aerial vehicle: S3 project report*. March, 2020.

[12] R. Rodrigues and B. Ismael. *Autonomous robotic aerial vehicle: S3 project report*. March, 2021.

[13] J. Bessa and A. Ceballos. *Autonomous robotic aerial vehicle: S3 project report*. March, 2022.

[14] *Gazebo - Robot Simulation made easy*. URL: https://classic.gazebosim.org/.

[15] *PX4 Autopilot User Guide (master)*. Last Updated: 9/6/2021. URL: https://docs.px4.io/master/en/.

[16] *The Raspberry Pi Foundation*. URL: https://www.raspberrypi.org/.

[17] *OptiTrack - Motion Capture Systems*. URL: https://www.optitrack.com/.

[18] *QGroundControl User Guide*. URL: https://docs.qgroundcontrol.com/master/en/.

[19] *ROS (1) with MAVROS*. URL: https://docs.px4.io/v1.12/en/ros/ros1.html.

[20] *MAVLink Developer Guide*. URL: https://mavlink.io/en/.

[21] *OctoMap 3D Models with ROS/Gazebo*. URL: shorturl.at/dyUVY.

[22] *Parameter Reference*. URL: shorturl.at/qvGHN.

[23] *Gazebo: Make a Model*. URL: shorturl.at/dekAW.

[24] *Pixhawk Flight Review*. URL: https://review.px4.io/.

[25] *SMACH ROS Package*. URL: http://wiki.ros.org/smach.