Worcester Polytechnic Institute

Robotics Engineering Program

Mohamed Bin Zayed International Robotics Challenge at Takemura Lab, Osaka University

A Major Qualifying Project Report
submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science by:

Nolan Greig, RBE
David McHorney, RBE

Project Advisor:
Professor William Michalson

Date: 10/10/2019

Abstract

This project completed a series of base systems for a team of drone to compete in challenges 1 and 3 of the 2020 Mohamed Bin Zayed International Robotics Challenge (MBZIRC). Using an architecture previously selected by the team at Osaka University, the project team prototyped and finalized several of the critical elements required for basic functionality. The sensors, drones, and computers the Osaka University team had collected were also integrated into a complete system that is ready for additional functionality. The result is a complete platform that demonstrates the basic functionality required for the challenges with room for additional features.

Acknowledgments

The authors would like to thank several people without whose contributions this project could not have been completed. Firstly, we would like to thank our project sponsor, Professor Photchara Ratsamee, for providing us with this opportunity and for his guidance during the project. In addition, we would like to thank our teammates on the MBZIRC project for their insight and hard work in ensuring that the system we were all working to build came together as a cohesive whole. We would also like to thank the other students and faculty of Takemura Lab, in particular lab head Professor Takemura Haruo, for the warm welcome we received and for their kindness and patience as hosts. We would like to thank the organizers of the Mohamed Bin Zayed International Robotics Challenge for hosting this competition and for their support of our entry. Finally, we would like to thank our advisor Professor William Michalson for his feedback and advice throughout the project, as well as Professors Jennifer deWinter and Ralph Sutter for their invaluable help preparing us for our three months stay in Japan.

Table of Contents

List of Figures

List of Tables

1. Introduction

For the Japan 2019 Robotics Engineering MQP, we worked with a team of students at the Takemura Lab at Osaka University to modify and program a team of drones for the Mohammed Bin Zayed International Robotics Competition (MBZIRC). Both the competition and previous work by the Osaka U team formed the basic framework upon which we built. Our task was to turn the solutions envisioned by our sponsor into reality.

1.1 MBZIRC

The Mohammed Bin Zayed International Robotics Competition is a robotics competition that takes place in Abu Dhabi. Organized by Mohammed Bin Zayed, the crown prince of Abu Dhabi, the competition intends to showcase the forefront of current robotics technology. The 2020 competition placed a special emphasis on Unmanned Aerial Vehicles (UAVs), colloquially referred to as "drones". The competition is divided into three challenges, with a triathlon-type "grand challenge" combining all three. Organizations were invited to submit proposals for any number of challenges that they would like to participate in; selected teams are given grant money for the competition, while those that do not receive grants may be invited to compete self-financed if they so choose. The challenge requires regular progress reports to ensure continued grant money; in this case the next submission was due on September 30th, providing a hard deadline for technical functionality approximately two weeks before the end of the project. Osaka University's Takemura Lab submitted proposals for challenges 1 and 3, receiving grant money for both.

1.1.1 Challenge 1[1]

Challenge 1 takes place in a 100m by 60m arena, with a flight ceiling of 20m. For this challenge, a team of up to three drones will work together to collect balloons placed randomly around the arena, as well as intercept an "adversary" drone and claim its payload, which in this case is a foam ball. As shown in Figure 1.1, the adversary drone flies in a randomly oriented figure eight pattern somewhere in the arena. In addition to the adversary drone and balloons, the arena will also contain randomly placed obstructions. Points are earned for collecting the balloons as well as claiming the adversary's payload. The written brief states that the drone which captures the adversary's payload

[1] It should be noted that the descriptions of the challenge rules provided here omit several key details or contain apparently contradictory statements. This vagueness, as far as the authors are aware, constitutes a deliberate part of the competition's challenge structure. Key details are withheld by the organizers until predetermined times, with some changes to the rules occurring as late as the day of the competition. Therefore, while the given summary of the challenges is true and correct as far as it goes and at the time of writing, the authors of this paper are aware that it is incomplete, and it is possible that it will no longer be accurate in the future.

should carry it to a ten-meter square target area and land without releasing it. However, an early briefing video shows an interceptor drone depositing the payload in a large box [4]. Twenty minutes are allotted to complete the task. For this challenge, full points will be awarded for a team of drones that acts completely autonomously. Human intervention is allowed but incurs a points penalty. All localization methods are allowed, however the use of Real-Time Kinematic (RTK) or Differential GPS (DGPS) incurs a points penalty [5].

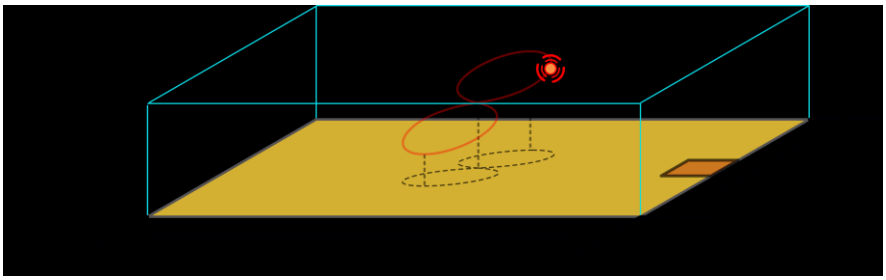

Figure 1.1: The approximate arena dimensions for challenge 1. Note that the projection of the adversary drone's flight path forms a figure-8 pattern on the ground regardless of orientation. The size and location of the 10m square scoring area are speculative.

1.1.2 Challenge 3

Challenge 3, as illustrated in Figure 1.2, takes place in a 60m by 50m arena with a 20m flight ceiling. For this challenge, a team of drones (and optionally 1 Unmanned Ground Vehicle) work together to extinguish "fires" located on the ground and inside of a building in the arena. Details provided on the building are sparse, but it will have a number of windows and be a maximum of 20 m tall. Points are awarded for using any of the permitted methods (sprayed water or chemical fire extinguisher) to "extinguish" the simulated fires, with the number of points being dependent on the amount of fire suppressant accurately dispensed on the target. Twenty minutes are allotted for this challenge. As in challenge 1, full points are available for fully autonomous teams, while human intervention is allowed with a penalty. RTK/DGPS is similarly penalized [5].

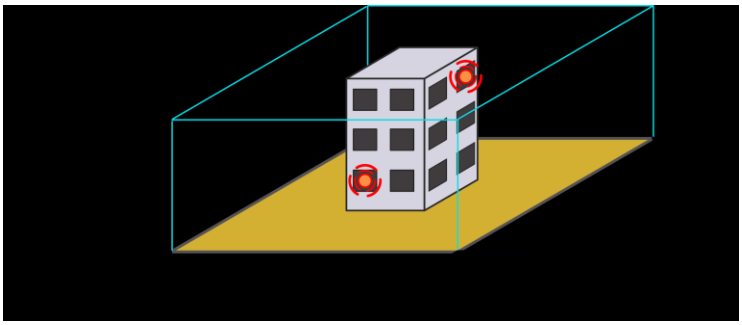

Figure 1.2: The approximate arena dimensions for challenge 3. Note the multiple fire locations throughout the target building. Position and layout of the target building are speculative.

1.1.3 Purpose

The intent of the competition is to showcase the forefront of current technologies and show how they can be applied to difficult-to-solve problems. By focusing on drones, the competition organizers are attempting to showcase a new technology that has captured a great deal of mainstream attention. The challenges are also meant to emulate real-life scenarios. Challenge 1 represents a terrorist attack or similar where a drone carrying a bomb or other dangerous payload must be safely intercepted. Challenge 3 showcases a potential application for drones in fighting fires in high-rise apartment buildings where it is normally difficult for firefighters to reach. By providing grants and holding a competition, the MBZIRC hopes to kickstart innovation in some of these areas.

1.2 Team and Existing Work

1.2.1 Osaka University Team

Takemura Lab is an interdisciplinary graduate research laboratory located at Osaka University. Under the direction of Professor Haruo Takemura, the lab has a focus on human-computer interaction, with research topics including augmented reality, virtual reality, robotics, and computer vision. Professor Photchara Ratsamee, an associate professor at the lab, submitted a proposal to the MBZIRC for challenges 1 and 3. These were both accepted, and awarded grant money. He then assembled a team of students from the lab. While all students working on the project have an interest in robotics, most come from a computer science background, and were unfamiliar with platforms such as ROS.

1.2.2 Proposal

Proposals were submitted for each challenge prior to our attachment to the project. The grant money for the project is provided based on the proposal; as such, our final design must include the core elements indicated in the proposal. This meant that several significant decisions had been made before we were attached to the project, and that we would have to work to incorporate these ideas into our final design.

The challenge 1 proposal would leverage a small drone team (seen in Figure 1.3) to complete the challenge. Two small “scout” drones would use their onboard cameras to collect video of the adversary drone, which would be relayed to a PC running ROS. This would generate a predictive

model of the adversary's flight path[2] and relay it to a third "catcher" drone. This drone would be equipped with a depth camera and a capture claw. It would use the depth camera to track the target and align itself for capture, and the claw to catch and hold the payload. Due to the opacity surrounding the scoring policy this drone would be capable of either depositing the payload into a box, or simply making a safe landing with it still in hand. There was also some confusion regarding the balloons, as the documentation at various times implies that they may be either simple obstacles or scoring targets. In the end it was decided that the adversary's payload would take priority, with any further development to allow balloon capture contingent upon a rules clarification.

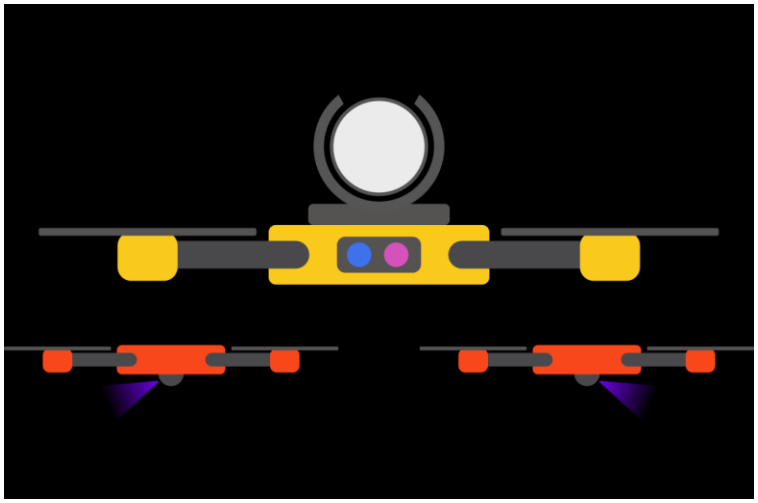

Figure 1.3: Intended team composition for challenge 1, two light scout drones and one heavy catcher drone.

For challenge 3, the most significant element of the proposal included two drones flying in formation fixed together by a physical tether, as seen in Figure 1.4. In this system a larger drone acts as the "parent." This drone carries a water reservoir and pump, as well as sensing equipment for navigation and to detect and track the smaller "child" drone. The child is attached to the parent by silicone tubing or similar material to carry pressurized water and carries a nozzle to control the flow of water and direct the spray. The parent's job is to scan the outer surface of the target building using its LIDAR sensor, identify points of entry, and pump water to the child when requested. The child's job is to enter rooms and locate fires using machine vision or a thermal camera. Upon locating a fire, it would aim the nozzle it carried and signal the parent to begin pumping. Since the parent would carry most of the heavy sensing equipment and could track the child's position, the child could be relatively small and light. This would allow the use of smaller drone frames for the child, enabling it to enter small spaces which might be unsafe for the parent drone.

[2] The sponsor deemed the predictive modeling and relay functions to be outside of the scope of our involvement in the project. Therefore, while we were aware of them and helped implement the foundational platform integration which would allow them to be completed, we had no direct hand in their development.

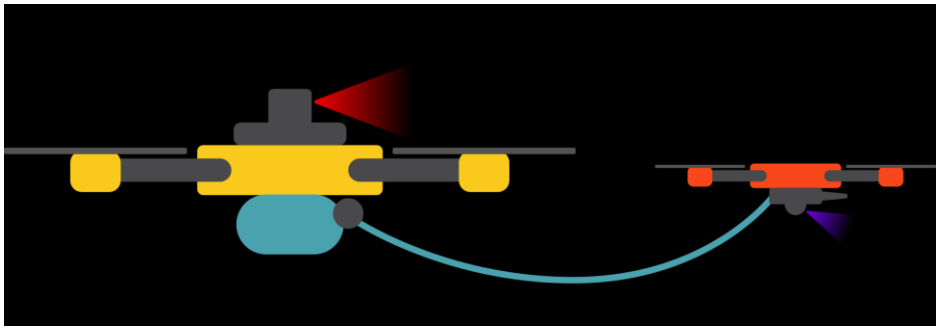

Figure 1.4: The parent/child pairing envisioned for challenge 3.

1.2.3 Existing Work

By the time the project started, the project team at Osaka University had already established a number of design elements based on this proposal. A DJI Matrice 100 (see: section 2.2.1) had already been selected as the catcher for challenge 1 and the parent for challenge 3, along with the sensors to be used. A pair of Parrot Bebop 2s (see section 2.2.2) had also been selected as the scouts for challenge 1, with one also serving as the child for challenge 3. The Osaka University team had also made the decision to use ROS Kinetic to manage the system with all code written in Python 2.7.

The drone and sensors selected were based on a project done by the JSK Laboratory [3] at the University of Tokyo, which used a drone equipped with a camera and laser rangefinder to identify a target tree and fly in a circle around it. By making their drone similar to the JSK drone, the Osaka University team hoped to use the JSK team's code as an example. This code included a modified version of the drone's SDK (reportedly to fix some of the major bugs and improve performance) and a simulation of the drone using the physics simulation software Gazebo. However, the drone had not been flight tested using the JSK code. Furthermore, while all devices for this project were selected for compatibility with the JSK team's code, they had not been used as a complete system.

The Osaka University team had also established the basic concepts for the drone's mechanical systems. Based on the proposal for challenge 1, they developed a basic idea for a ball-catching mechanism. Their requirements stated that it be capable of handling a ball as specified in the challenge documents. It must also be light enough that a drone loaded with it, all of its sensors, and the ball from the challenge be under the drone's maximum takeoff weight.
For challenge 3, the parent drone required a pump system. As envisioned by the team, the pump and reservoir would mount on the underside of the drone creating the need for an extended landing gear. There were no hard requirements for this system determined, except that it carry as much water as possible.

1.3 Project Requirements

The most basic requirement for this project was demonstrable functionality for challenges 1 and 3 for the submission on September 30th of 2019. To meet this requirement, we determined that we must identify some basic tasks related to the challenges and demonstrate completion of those. These basic tasks are detailed below.

1.3.1 Challenge 1

1.3.1.1 Capture the Ball

The most basic functionality possible for challenge 1 is capturing the ball from the adversary drone. While autonomous operation is required for full points, the drone must be capable of handling the ball in some way. With this requirement having been completed, we can show progress towards the completion of challenge 1 and enable further development on more advanced functionality.

1.3.1.2 Criteria for Evaluation

While in flight, the drone must be capable of capturing and handling a suspended ball as described in challenge 1. For demonstration, the ball does not necessarily need to be suspended from a drone. The exact specifications for the adversary drone have not yet been provided, and ball collection in this case would require significant testing and operator practice--neither of which demonstrate actual progress on challenge 1. However, sufficient details on the adversary's payload system were provided that a reasonable test procedure could be assembled. As shown in Figure 1.5, a carbon fiber rod would be connected by flexible wire joints to the drone at one end and a neodymium magnet at the other. A corresponding magnet permanently attached to the ball would be used to hold the ball on the end of the rod. As outlined in the competition rules, a force of no more than 4N would be required to separate the magnets and release the ball.

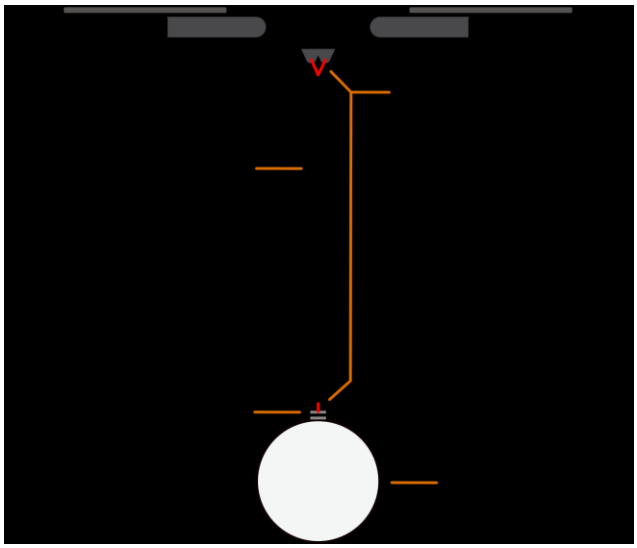

Figure 1.5: Rough diagram of the adversary drone and payload ball, derived from images on the MBZIRC website [5].

For this requirement, ball capture is under manual operation only. This is intended to ensure that the capture system is basically functional before implementation of autonomous ball capture.

1.3.2 Challenge 3

1.3.2.1 Fly the Parent Drone Around the Building Using Sensors

As the size and shape of the building likely will not be known for the competition, the parent drone must be capable of navigating around the building for the challenge using only its sensors (in this case, a scanning laser rangefinder and a depth camera). Furthermore, the location of the fires and layout of the building during the challenge is not known, requiring some strategy to search the building and locate the fires. In this case, the most simple search possible is checking each window to see if there is a fire inside. This can be accomplished simply by flying around the exterior of the building while observing the interior from each window.

Given the above, a simple way to demonstrate the beginning functionality of challenge 3 is to fly the drone around the outline of a building autonomously. This should be done using sensor inputs only; dead reckoning is unacceptable, as the location and dimensions of the building for the challenge will likely be unknown. This requirement also demonstrates full functionality of the drone and its sensors as part of a ROS system. Since none of the components have been tested as part of a complete system, this necessitates their integration and will leave a platform ready for further software development.

While this requirement demonstrates basic functionality for challenge 3, other processes must run concurrently with the navigation during the challenge. This includes future functionality such as

mapping the building, detecting fires, or flying while tethered to a child drone. While not a strictly real-time system, the resource usage by the program is an item of concern so that other critical processes can run without excessive latency. To this end, the resource usage of the navigation software should be minimized.

1.3.2.2 Criteria for Evaluation

Using a laser rangefinder and GPS for navigation, the drone must take off, approach the building, and follow its outline for a complete circuit. Through the duration of the flight, the drone must maintain a designated following distance from the building. The drone must also maintain a constant facing towards the building. Throughout the traversal, the drone should be relatively insensitive to features that cause it to deviate from the true outline of the building, especially windows. While the challenge provides 20 minutes, a significant portion of time must be allotted to actually extinguishing fires. Therefore, this must be demonstrated in 10 minutes as an absolute maximum.

ROS Kinetic does not support real-time tasks. As such it is difficult to determine the execution time of the algorithm and the processor resources required by it. Therefore, observation of the drone and the processes while executing the program is necessary. Success or failure in this case is best determined by the observed responsiveness of the drone to changing conditions. While an objective measurement of performance is desirable, this was not practical in the context of this project.

1.3.2.3 Pump Water from the Parent Drone

Points are scored for this challenge by spraying mock fires with water. As such, the team of drones must be capable of moving water in some way at the absolute minimum. Without a drone capable of extinguishing a fire in some way, it is impossible to score any points in the challenge and therefore impossible to show any real progress towards accomplishing something.

1.3.2.4 Criteria for Evaluation

The drone must be able to take off, maneuver, and land while carrying the pump and a full reservoir of water. In this case, maneuver is defined and a full range of motions (x and y translation, altitude adjustment, and yaw adjustment) at low or medium speed. If possible, the drone shall be capable of 20 minutes of continuous flight low speed flight while fully loaded, including the use of the pump to drain the reservoir. Changes is reservoir level shall have no observable impact on drone dynamics.

2. Background

2.1 Drones

"Drone" conventionally refers to an Unmanned Aerial Vehicle (UAV), and especially to small rotorcraft such as quadcopters in common usage. Because of the simplicity of their controls, small size, and flexibility, quadcopters and other rotorcraft are seeing extensive use by consumers, corporations, researchers, and even the military. Drones are being tested for applications ranging from package delivery to firefighting, while their accessibility has caught the attention of the mainstream. Today, a drone that can be operated using a cell phone can be obtained for less than $100, further sparking public interest in their applications.

2.1.1 Mechanics

Most drones on the market today are quadcopters. Drones of this type use four brushless motors connected to fixed-pitch propellers to generate both lift and thrust. Two rotors spin clockwise while two spin counterclockwise (see Figure 2.1); these are arranged such that no adjacent rotors spin in the same direction. This balances the moments from each rotor and keeps the drone from constantly rotating in the direction of the rotors.

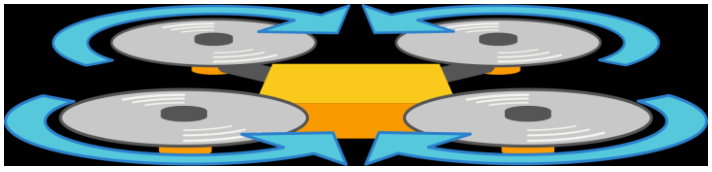

Figure 2.1: The arrangement of counter-rotating propellers on a quadcopter drone.

Because of the multi-rotor configuration, the speed of one or more rotors can be adjusted independently. As shown in Figure 2.2, changing the speeds of particular motors can cause the drone to move in predictable ways. Increasing the speed of two adjacent motors while reducing the speed of the opposing pair will cause the drone to pitch in the direction of the slowed motors, which due to the lifting force applied will cause the drone to shift in that direction (commonly referred to as "strafing"). Increasing the speed of the motors on opposing corners will unbalance the moments they exert on the drone, causing it to yaw. Finally, increasing or decreasing the speed of all motors by the same amount at once changes the lift generated, allowing the drone to adjust its altitude.

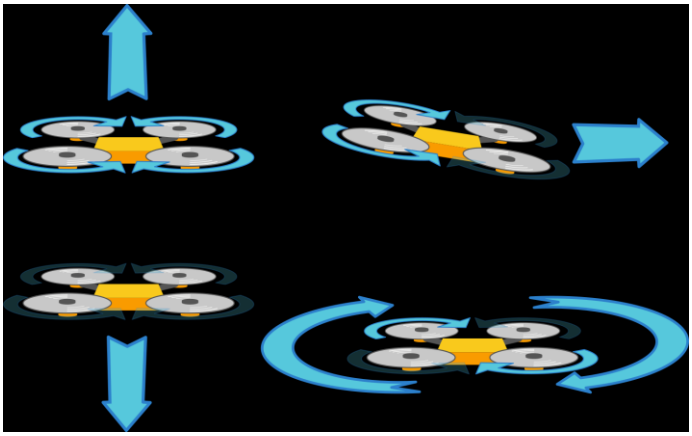

Figure 2.2: The effects of differential motor speeds on a quadcopter.

Other related varieties of rotorcraft exist using different numbers of rotors (ex: six-motored hexcopters). These generally hold to the same design principles as quadcopters, such as using even numbers of counter-rotating propellers to balance moments. Similarly, they use the same principle of varying motor speed to change rotation, altitude, or position. A larger number of rotors tends to increase the stability, lifting strength, and speed of the drone. However, it also increases cost, energy use, and overall size. A larger number of rotors also increases the complexity of any calculations the drone must perform in order to determine the proper motor speeds required to achieve a desired movement. For these reasons quadcopters tend to be the most popular and accessible variety of drone, except in applications where the added lift and stability of hex- or octocopters are absolutely required.

2.1.2 Applications

As their costs have come down, drones have seen increased use in several fields, from recreational to military. Their ability to both hold position steadily and move quickly and smoothly has made them desirable for both photographers and videographers. Various companies have investigated the use of drones as a supplement or replacement for traditional parcel delivery services. The simple fact that they are able to fly while carrying variable payloads makes them able to negotiate delivery routes which land-based couriers would struggle with for a variety of reasons (for example, developing or degrading road networks). Gains in drone speed and improvements in control software have even led to the creation of drone-based sports such as First-Person View (FPV) racing. In this activity pilots take control of high-speed quadcopters and attempt to complete complex obstacle courses in the shortest time possible. As the name implies, a pilot will operate their drone in virtual first-person perspective via a front facing camera, with the feed often delivered by a visor similar to a virtual reality headset.

The same characteristics of drones which make them desirable to hobbyists and commercial interests also spur interest from government bodies. Applications in emergency response, security,

and military support are all being pursued. Carrying complex sensors quickly and effectively while avoiding obstacles allows drones to be used to scout ahead and identify danger, or to tirelessly patrol an area from an elevated vantage point. This can be as valuable to soldiers attempting to locate booby traps as it can be to emergency responders attempting to locate survivors in a disaster. The ability to carry various payloads also means drones can get involved directly in a dangerous situation. While the ethics of weaponized robots are still hotly debated, the use of drones to carry medical supplies or firefighting equipment in times of crisis is much less controversial. And in the event that a drone is used maliciously, another drone is often the best counter. Unlike conventional aircraft they can be deployed rapidly, can operate in tight quarters such as near ground level in a city center, and are agile enough to intercept a hostile drone with minimal potential for collateral damage.

2.2 Items Selected Prior

The following components and systems were selected by the Osaka University team in the early stages of the MBZIRC project and would form the basis of our work.

2.2.1 DJI Matrice 100 (DJI M100)

The Matrice 100, shown in Figure 1.6, is a quadcopter released by the Chinese company DJI in 2016. The drone is marketed as a "drone for developers," featuring a modular construction, UART ports for communication with the flight controller (utilizing an SDK published alongside the drone), and additional ports provided to obtain power from the drone's battery. The drone features expandable payload bays and a large number of unused threaded screw holes, enabling the mounting of first-party or custom hardware upgrades. DJI offers several accessories for the drone, notably a camera gimbal and the Guidance module. The Guidance module consists of an array of cameras and ultrasonic sensors that allow the drone to detect obstacles in any of its four facings. The drone can be configured to brake when an obstacle is detected or to engage in active obstacle avoidance. The Guidance module also allows the drone to measure its distance to the ground, enabling monitored takeoff and landing as well as improving the drone's ability to maintain altitude during maneuvers.

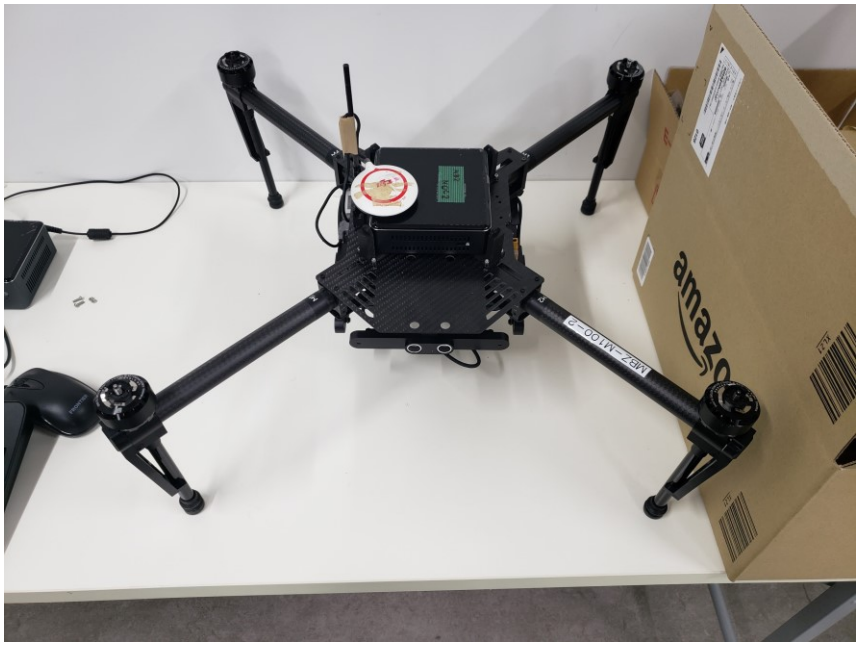

Figure 2.3: One of the two M100 drones owned by Takemura Lab.

The drone itself has a weight of 1.755 kg, while the largest battery weighs 0.676 kg. Its max takeoff weight is listed at 3.6 kg, allowing for a theoretical 1.169 kg payload capacity. At 1 kg of payload, the drone has a listed hovering time of 16 minutes with the largest battery.

The 24V battery used by the drone is a proprietary component referred to as the "DJI Intelligent Battery." It features smart charging and some degree of communication with the drone's operating system. This is essentially a "black box" component, meaning that the battery is only compatible with other DJI products. There are multiple ports on the drone which can relay power from the battery to other devices; however, these terminals all supply the battery's full voltage. There are no specifications about battery voltage or maximum ripple. Therefore, it is assumed that significant voltage drop can be seen on these terminals during maneuvers making these ports only suitable for non-critical systems.

2.2.2 Parrot Bebop 2

The Bebop 2 is a quadcopter produced by the company Parrot, marketed as an iteration on their popular Bebop model which itself is an iteration of their older ARDrone. The Bebop 2 is a relatively low-cost drone aimed at consumers. It is piloted by smartphone app and communicates via a wireless access point it broadcasts. Being a low-cost consumer drone, it has no specified payload capacity and is bereft of features besides its 1080p camera. It has an advertised 26 minutes of flight time.

2.2.3 Hokuyo UST-20LX Laser Rangefinder (Hokuyo LIDAR)

The UST-20LX is a laser rangefinder (LIDAR) produced by the Japanese company Hokuyo. The sensor has a detection angle of 270 degrees and an angular resolution of 0.25 degrees. The detection distance is advertised up to 60 meters but is best within 20 meters. The sensor package itself is quite small and light, weighing just 130 g, making it attractive for drone applications. However, the sensor is not filtered for direct sunlight, which can introduce excessive noise into the scan. The module includes an ethernet connecter to interface with a PC.

2.2.4 Intel Realsense D435I Depth Camera (Intel Realsense)

The Realsense D435I is a depth camera produced by Intel. The unit contains stereo IR cameras for depth measurement up to 10 meters. In addition, the unit contains an RGB camera module and an inertial measurement unit (IMU). This allows for easier visualization of the output and allows the camera to act as a standalone localization and mapping module. The camera communicates through USB. Intel maintains and distributes an SDK for the camera module, which includes a ROS wrapper. The weight of the module is not included in the specification; this was determined experimentally to be approximately 10 grams.

While this sensor was selected for use on the drone, it did not play an essential role in the fulfillment of the requirements. In testing, the unit interfered with the M100's GPS systems. Since it was not strictly necessary to integrate this sensor to meet the project requirements, its functionality was never implemented.

2.2.5 Intel NUC 7 NUC7i7BNKQ (Intel NUC)

The Intel "NUC" is a desktop PC built to an ultra-small form factor. The PC includes an Intel Core i7-7567U, a dual-core 3.5 GHz laptop CPU with integrated graphics. It also features 16 GB of DDR4-2133 RAM, 512 GB of NVMe solid state storage, and integrated Wi-Fi. To save on size and cooling, the unit accepts a 12-19 V DC power input. The specification does not indicate typical current draw or the noise suppression capabilities of the unit; the device itself indicates a maximum current draw of 3.43 A, which is identical to the output of the supplied mains adapter. There is weight given in the specification; this was determined experimentally to be approximately 400 grams.

2.2.6 Water Pump

A centrifugal water pump was acquired by the team prior to our arrival and would form the centerpiece of the physical architecture for challenge 3. While a datasheet was not available for the pump, some characteristics were explicit based on its markings, and we were able to determine

others experimentally. The pump had a mass of 350g, a 10mm inner diameter suction tube, and a 7.5mm ID discharge tube. It was rated for 24V and drew 0.6A of current under normal operating conditions, providing 4m of head. The pump was of a variety which requires a flooded suction in order to operate properly. While this was not a major issue under laboratory conditions, it caused some erratic behavior in the field, especially when an attached reservoir began to run dry.

2.2.7 ROS Kinetic

ROS is a middleware designed to simplify robotics development and prevent duplication of work by providing a set of tools to accomplish tasks commonly necessary in robotics applications. Each ROS release is designed to run over a long-term support distribution of Ubuntu Linux; in the case of ROS Kinetic, it is designed to run on Ubuntu 16.04. Processes in ROS are separated into individual nodes. In ROS, the machine a node runs on does not matter. In this way, multiple computers can be used to run individual parts of a system and can be swapped out or shifted around with no software changes. Nodes communicate via messages through the publisher/subscriber topologies or by request via services. ROS organizes messages into topics. Any node can publish any kind of message to a given topic. Nodes can subscribe to a certain type of message on a certain topic and will be notified when a new message is available on that topic. A node can publish and/or subscribe to any number of topics. Nodes can also advertise services. The service can be called by sending a request with some data for the service (usually arguments) to which the node will send a response. This is useful for computationally expensive processes that do not need to operate continuously, such as path planning.

In addition to this, ROS includes tools for package management, wrappers for popular software libraries, and tools for debugging program execution. While these make ROS seem very attractive, it does have a number of disadvantages. First, ROS and Ubuntu combined have a very large overhead, typically necessitating a PC with at least a laptop-class CPU to run. This means that every drone and sensor used must have a high-bandwidth link of some sort to a relatively powerful computer to handle communication with ROS. Second, ROS is non-deterministic in its execution, and is in many cases at the mercy of wireless networks. This can create latency problems for time-critical tasks and forces the developer to minimize time-critical tasks instead of providing resources to support them.

2.3 System Architecture

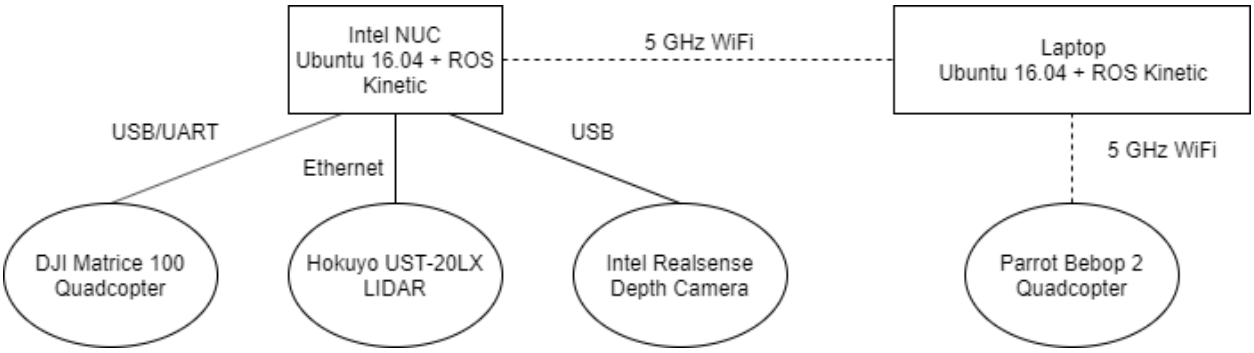

Figure 2.4: Basic system architecture

The system architecture as shown above was not implemented at the start of the project. As envisioned this system, the Intel NUC is located on the M100 and communicates with a laptop base station over 5 GHz Wi-Fi. The NUC runs the driver nodes required to interface with the drone and its associated devices. The laptop base station acts as the master of the system, and runs any computationally expensive processes, specifically navigation in terms of this project. One base station PC is used for the whole system, but more machines could be integrated if necessary. The base station also runs a driver node for the Bebop 2 and communicates with that drone over the same 5 GHz Wi-Fi network.

This basic architecture shown in Figure 2.1 is used for challenges 1 and 3. There are minor differences between challenges; notably, the LIDAR is intended for use in challenge 3 only and is thus not considered in the claw design for challenge 1. Future implementations may also utilize additional Bebop 2 or M100 drones. These will follow the pattern set out in this system architecture.

2.4 Testing

2.4.1 Simulation

The Gazebo simulation created by the JSK team (shown in Figure 2.2) uses a model of the drone equipped with a USB webcam and a laser rangefinder nearly identical to the Hokuyo UST-20LX. To facilitate testing for challenge 3, one of the Osaka University students modified the JSK simulation to include a building similar to the building specified in the challenge.

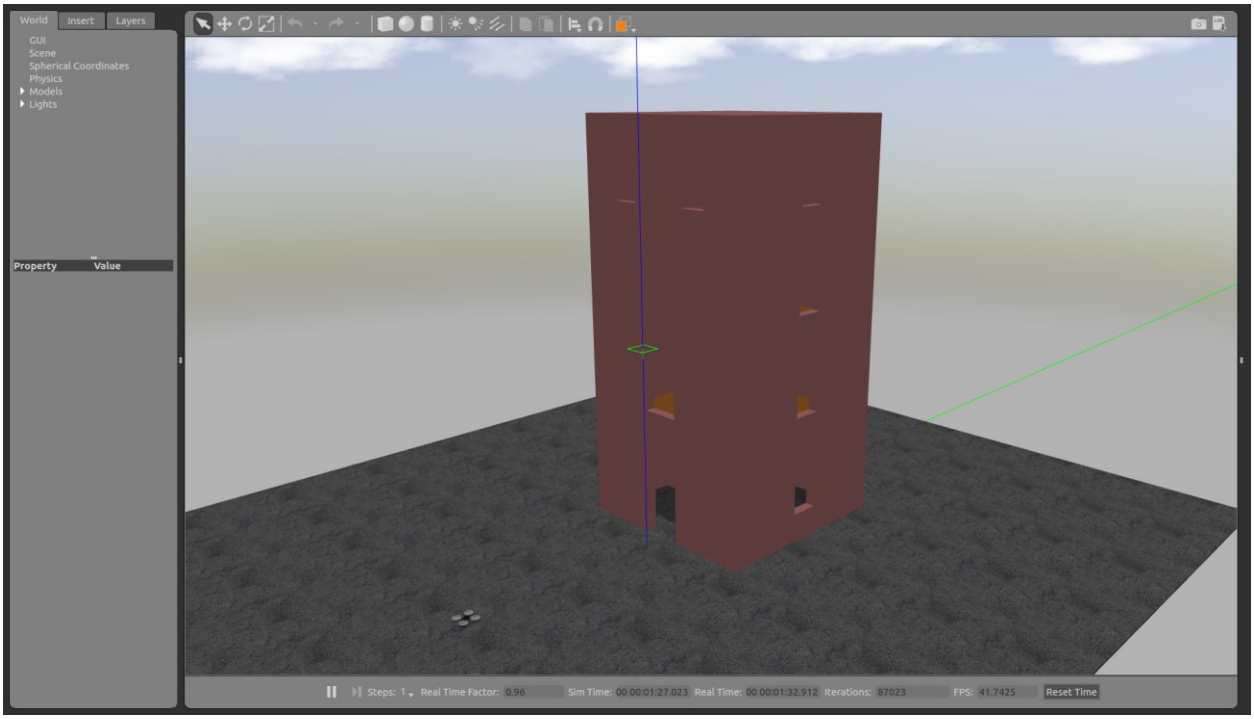

Figure 2.5: Screenshot of the modified Gazebo simulation

2.4.2 Real-World Testing

2.4.2.1 Location

Because the M100's size and weight, real-world testing must be done outdoors. While flying drones outdoors is legal in Japan, there are a number of restrictions on outdoor flight. This includes limiting flights to certain areas and obtaining a permit before each flight. Because of the administrative work required, the Osaka University team secured permission from the Kobe City Fire Academy to test on pre-arranged days. The Fire Academy is not located in a densely populated area and has an arrangement with the local government regarding drone flight on their premises. While this limited the days we could test the drones and necessitated careful scheduling, this location provided plenty of space to engage in safe test flights.

2.4.2.2 Mock Building

For some preliminary tests, a small mock building was constructed using steel tube and a tarp as shown in Figure 2.3. This provided a space for small-scale testing and was used when the large training grounds at the Fire Academy were unavailable.

Figure 2.6: Mock building used for testing

2.4.2.3 Replica Apartment Building

The Kobe Fire Academy maintains a number of replica buildings on the premises for firefighter training. One of these, seen in Figure 2.4, is a three-story replica apartment building complete with windows, a fire escape, and balconies. This building was made available for testing by the Fire Academy. Since specifications on the building for challenge 3 are sparse, this was used as an approximation of the challenge building. The building also adjacent to a large open training ground.

Figure 2.7: Google Earth render of the replica apartment building used for testing

3. Electrical Systems

3.1 System Architecture

In this system, all components communicate through standard ports and protocols as detailed in the background. Electrical considerations were limited to power considerations for the devices mounted on the M100.

3.2 Intel NUC

3.2.1 Design

Since the NUC handles all communication between the drone and the rest of the ROS system, it is a critical component that cannot fail. Therefore, while weight is a concern, more weight can be devoted to the NUC to ensure constant functionality. While the M100 provides several power terminals, the voltage drop at these terminals under maneuvers is unknown. The NUC also does not specify the maximum tolerable voltage sag. This makes providing the NUC with its own power supply more desirable. Finally, the drone is indicated to draw up to 65 W of power (3.43 A at 19 V), which at 20 minutes equates to approximately 22 Wh of energy, or 15% of the drone's battery capacity. This equates to approximately 3 minutes of lost flight time, which exceeds our tight power budget.

3.2.2 Results

Given the NUC's critical status as well as the scarcity of power available on the drone, a 27 Wh lithium-polymer battery was selected to power the NUC. This provides power for the NUC for the duration of the challenge at maximum power draw. The battery itself is fairly light (XX g) and ensures consistent operation of the NUC for at least 20 minutes, which is enough for the duration of the challenge.

3.3 Water Pump

Powering the pump with a dedicated battery was considered. In order to determine if the reduced draw on the main battery would be worth the extra weight the pump's power use had to be established. Lacking a datasheet, a current test was conducted at a known voltage. This was used to estimate the pump's impact on flight time:

-Source Voltage: 24 V
-Measured Current: 0.6 $\pm$ 0.05 A

-M100 battery capacity: 4500 mAh [1]
-Flight time on full charge (from previous experiments): 20 min
-Current draw of drone: $\frac{4500\ mAh}{1} \times \frac{1}{20\ min} \times \frac{60\ min}{1\ h} = 13{,}500\ mA\ =\ 13.5\ A$
$-13.5\ A\ +\ 0.6\ A\ =\ 14.1\ A$
-Modified flight time: $\frac{4{,}500\ mAh}{1} \times \frac{1}{14{,}100\ mA} \times \frac{60\ min}{1\ h} = 19.1\ min$

A drop in flight time of 0.9 minutes was predicted. This prediction was predicated on the drone running the pump constantly throughout its flight, which was not part of the game plan for challenge 3. Furthermore, it was anticipated that if the drone had not found and extinguished the fire after nineteen of the allotted twenty minutes, the remaining one minute was not likely to make a difference. For these reasons a dedicated pump battery was not used.

3.4 Hokuyo LIDAR

Per the specification, the LIDAR requires less than 150 mA of supply current. This equates to approximately 1 Wh of energy over a 20-minute period at 24 V. As this is less than 1% of the total battery capacity, it was decided that the LIDAR would have little to no effect on the flight time of the drone. Furthermore, since the LIDAR accepts 12-24 V, it be run off of full battery voltage. Per the specification, it is actually capable of 10-30 V operation with 10% ripple, making the device insensitive to variations in voltage caused by drone maneuvers. This meant the LIDAR was an excellent candidate to be powered by the drone battery, simplifying wiring requirements.

4. Software

4.1 Overview

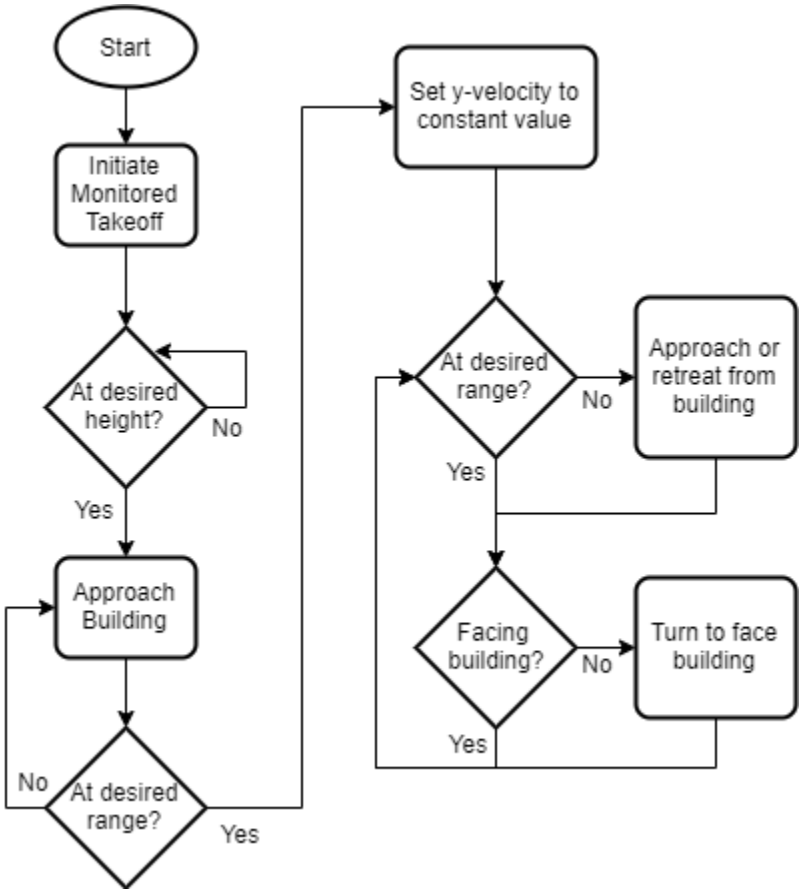

Figure 4.1: Program Flowchart

The above flowchart illustrates the basic structure of the challenge 3 program, which navigates the M100 around a building using GPS and a laser rangefinder. This uses the M100's GPS to monitor height and the LIDAR to measure the distance to the building. The orientation of the drone relative to the building is also determined using the LIDAR. This program uses ROS drivers to interface with the M100 as well as the Hokuyo LIDAR. The monitored takeoff is a feature of the M100's flight controller and is accessed via its ROS driver.

No software was implemented for challenge 1 as part of this project. Challenge 1 required use of the depth camera, which was not integrated. Furthermore, since challenge 1 required the ball capture mechanism for testing, it was assigned a lower priority than the project requirements indicated in this document.

4.2 System Architecture

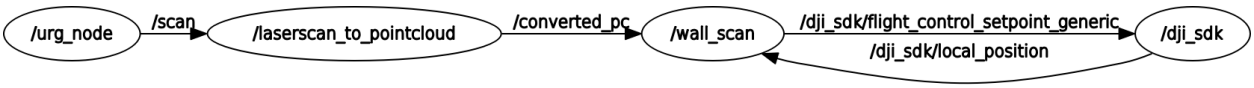

Figure 4.2: ROS rqt_graph Node Plot

The rqt_graph from ROS showing the nodes involved is pictured above. Here /urg_node and /dji_sdk are drivers for the Hokuyo LIDAR and DJI M100, respectively. As indicated above, these nodes are run on the NUC. /urg_node publishes laser rangefinder data as a ROS LaserScan message on the /scan topic. /dji_sdk subscribes to the topic /flight_control_setpoint_generic, which provides velocity control instructions to the robot in the form of a ROS Joy message. In this case, the message data is provided as (x, y, z, yaw, flag) where flag indicates how the flight controller is to interpret the axis messages. The drone publishes the drone's GPS position in the local ground frame to /local_position. The reference point for this frame is set by calling the /set_local_pos_ref service, which is advertised by the /dji_sdk node. The /dji_sdk node advertises two other relevant services. The first is /sdk_control authority, which regulates SDK control of the drone. By calling this service, the SDK can either request or release control of the drone, enabling or disabling autonomous flight. The second is /drone_task_control, which allows the SDK to initiate a monitored takeoff or landing.

The nodes /laserscan_to_pointcloud and /wall_scan both run on the base station. /laserscan_to_pointcloud subscribes to LaserScan data on the scan topic and converts the laser scan data to a 2D point cloud in the laser reference frame. This data is published as a PointCloud2 on the /converted_pc topic. The node /wall_scan functions as the main control node, subscribing to both /local_position and /converted_pc. Based on the program logic, /wall_scan calls the /sdk_control_authority, /set_local_pos_ref and /drone_task_control services while publishing velocity control data to /flight_control_setpoint_generic.

4.3 Program Logic

The program logic is located entirely in the /wall_scan node. Basic control flow is established through a state machine. During initialization, the program first attempts to take control of the drone. Succeeding this, the program then attempts to set the local position reference. If these services complete successfully, the drone moves to the first state of the state machine.

4.3.1 Takeoff

In the first state, the /drone_task_control service is called, initiating a monitored takeoff. Because the program does not have access to the Guidance data or any other indication that the takeoff has resolved, it must monitor the GPS to determine when the drone has reached the correct altitude. Once the drone has risen to the appropriate height and is accepting commands, the program moves

to the next state and initiates a change in altitude through velocity control moving the drone to the final desired height. Height is still monitored using GPS.

4.3.2 Approach

When the drone is at the correct height, it begins to approach the building. This is done using proportional control with a low maximum speed cap. Once the drone has reached the required distance from the building, it changes to its final state, the building scan.

4.3.3 Scan

To keep consistent travel around the building, the program sets the drone's y velocity to a low, constant value. This keeps the drone's movements around the building consistent. The x velocity is set via proportional control and attempts to keep the drone a set distance from the building. The yaw velocity is also set via proportional control and attempts to minimize the angle between the drone and the closest point on the wall. Based on these behaviors, the drone will complete a full circuit of the building while maintaining a consistent distance from the wall and a constant facing towards the building.

4.4 Program Design

4.4.1 Building Detection

Based on the challenge details, we determined the best way to detect the building was by finding the closest point on the laser scan. During the challenge, there will be nothing else in the arena besides the drones and the building; this makes the assumption that the building will be the closest object quite safe. This also greatly reduces the difficulty of identifying the building. Instead of identifying straight lines present in the laser scan and determining which lines represent the building, the program must instead find the minimum value of a list. This greatly reduces the amount of operations required and reduces the difficulty in implementation.

4.4.2 Iteration 1

The first iteration of the program was developed for testing in simulation. This served to test the basic program logic and ensure that the drone would operate safely in a real-world test.

4.4.2.1 Implementation

This iteration of the program differs from the final architecture explained above. In this version, the program operates the laser scan data directly instead of converting it to a point cloud. From

this data, the program would find the minimum measurement in the scan and calculate the angle that the measurement was taken at. Using this angle, the distance from the front of the drone to the wall along the drone body x axis is calculated. This calculation was done to avoid problems with situations as pictured below.

This calculated distance was fed to the x velocity and yaw velocity via proportional control. Since the distance from the front of the drone to the wall is used, an error in distance could indicate improper facing or an incorrect distance to the wall. The drone will therefore attempt both corrective actions, causing the proper action to "win" and bring the drone to the correct position and orientation.

4.4.2.2 Testing

This iteration was tested extensively in simulation to ensure stability. This was to avoid any potential "runaway drone" incidents. With some tuning to the proportional control constants, the drone navigated around the building consistently and accurately.

4.4.3 Iteration 2

The second iteration of the program was a reimplementation of the first for testing on the physical drone. At this point, all of the components had been integrated and tested with ROS; this was to show that they could work together as a complete system.

4.4.3.1 Implementation

This version was based off of a newer release of the DJI SDK and did not utilize any of the JSK team's code. Despite some small changes to add the required service calls and reformatting the odometry and velocity control functions, this implementation was identical to the first iteration.

4.4.3.2 Testing

The second iteration was tested on both the mock building and the replica apartment building. In both cases, the drone was able to take off, approach, and follow the wall. However, on approaching a corner, it would veer off in the wrong direction. This was eventually determined to be the fault of the distance calculation implemented in iteration 1. Because the distance from the front of the drone to the theoretical wall was calculated, corners were functionally invisible to the program and would cause unintended behavior. We were unable to determine why this approach worked in simulation, however.

4.4.4 Iteration 3

4.4.4.1 Implementation

This implementation uses the architecture explained above. When the /wall_scan node receives a point cloud message from the converter, the node converts the point cloud message to a list of points in polar coordinates. The minimum range measurement is fed to the x velocity using proportional control to bring the drone to the desired range. The associated angle is fed to the yaw velocity via proportional control to minimize the angle measurement. This allows the drone to see corners properly and removes any ambiguity from incorrect facing.

4.4.4.2 Testing

This iteration was also tested on the mock building and the replica building. The program performed as expected, bringing the drone on a smooth path around the exterior of the building while maintaining a consistent facing. In practice, the difference between the true difference between the raw distance measurement and the distance as calculated in iteration 1 was very small, making the calculation redundant.

The drone was able to complete a full circuit of the mock apartment building in approximately 7 minutes while travelling at 0.25 m/s. This fits within the time criteria, especially given the large size of the apartment building and the low speed utilized by the drone. At the test site, there was no way to evaluate the latency or resource usage by the program. However, there were no major latency issues observed, as the drone successfully completed the circuit.

5. Mechanical Design

5.1: Platform

For both challenges the M100 would serve as the primary flight platform, filling the role of catcher for challenge 1 and parent for challenge 3. Thanks to its large payload capacity relative to the Bebop 2s it was capable of mounting much more extensive hardware modifications.

5.2: Landing Gear

5.2.1: Rationale

Challenge 3 was predicted to be the most physically demanding for the M100. In order to extinguish a fire, the drone had to carry some quantity of water as well as the hardware necessary to dispense it. The stock landing gear consist primarily of plastic tubes with a wall thickness of 1 mm, and the sponsor anticipated problems if these were required to bear the drone's full takeoff weight under anything other than ideal circumstances. It was also predicted early on that the most efficient way to carry the pump hardware would be underneath the drone, potentially causing ground clearance issues. We were therefore asked to develop supplemental or replacement landing gear to address these issues.

5.2.2: Requirements

The primary requirement for the new landing gear was that they should support the drone's maximum takeoff weight at rest. A minimum factor of safety of 3 was established for this condition. In addition, the sponsor requested some amount of shock absorption in case of a rough landing. Since the stock landing gear included air springs (see Figure 5.1), it was decided that the new landing gear should leverage these if possible.

Figure 5.1: One of the stock air spring modules retrieved from the broken landing gear.

The secondary requirement was ground clearance for the pump system. By default, the drone has a ground clearance of approximately 40mm, as shown in Figure 5.2. Since the landing gear and

pump system were developed in tandem an initial added length of 10cm was established. This was altered for later revisions as the pump layout was finalized.

Finally, the landing gear must have minimal weight. No initial benchmark was established for this condition, although collectively the landing gear and pump system had a maximum mass of about 500g. This requirement was expanded to include minimal changes in the drone's moments of inertia after preliminary field tests with the second revision of the design (see below).

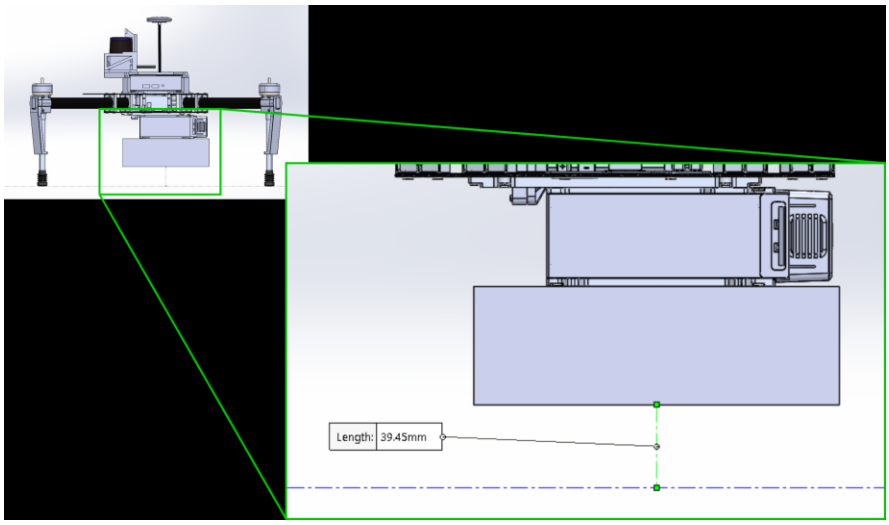

Figure 5.2: Side view of M100 in Solidworks showing stock ground clearance. The large block at the bottom is a stand-in for the Guidance sensor package.

5.2.3: First Revision

The initial concept for the landing gear prioritized strength and shock absorption. This would have consisted of floating stilts which fit over the end of each stock landing gear leg. These would be connected by several long plastic leaf springs to a bracket attached to the drone's bumper mounts. To improve stability and compensate for the loss of the stock bumpers, the corners of each leaf spring on a leg would be joined by a new bumper. This would allow the stock air springs and the new leaf springs to work in tandem to absorb landing forces. These parts were designed to be 3D printed, with the leaf springs using a concentric infill to improve flexibility.

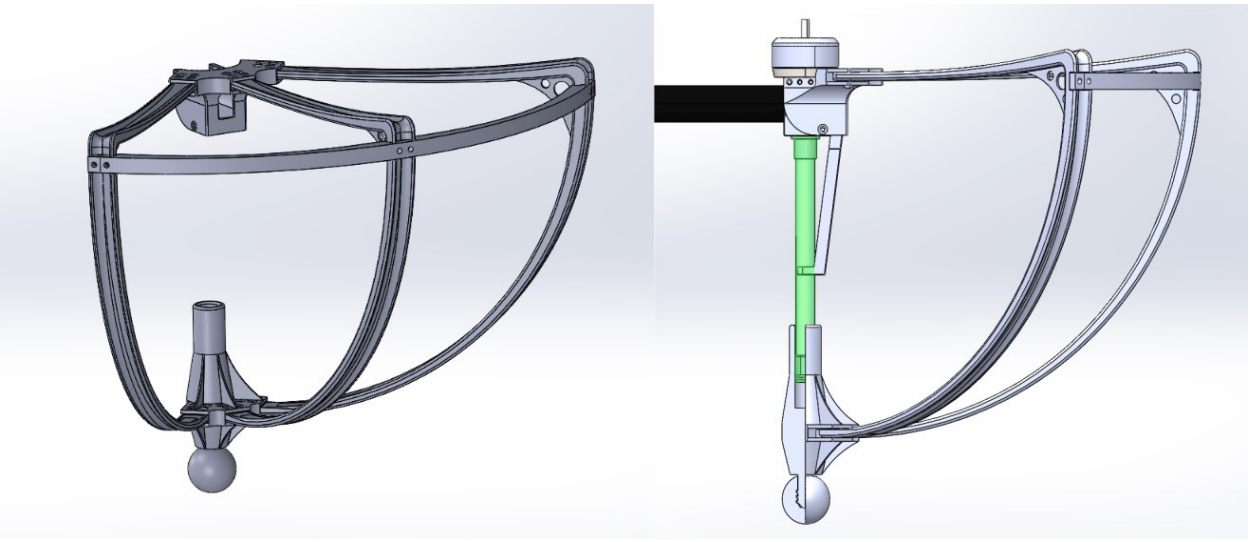

Figure 5.3: Left: The first revision of the landing gear.
Right: The second revision in place on the drone showing the interaction between the stock landing gear (highlighted green) and the additional stilt.

The full assembly was modeled in Solidworks, as shown in Figure 5.3, and a small number of test pieces were printed using an Ultimaker S5 and Polylactic Acid (PLA) filament to check for fitting. However, it quickly became clear that even with an unacceptably low infill this design would use the entire payload capacity of the drone. Attempts to refine it were discontinued in favor of a new design.

5.2.4: Second Revision

The second design revision was much simpler while retaining the floating stilt concept from the first design. This version consisted of stilts which fit over the stock landing gear, as shown in Figure 5.4. Rather than using mounting hardware they were designed to be held on by the drone's antennae. Flexible bumpers were incorporated to prevent damage to the stilts or frame components in the event of collision during stilt movement.

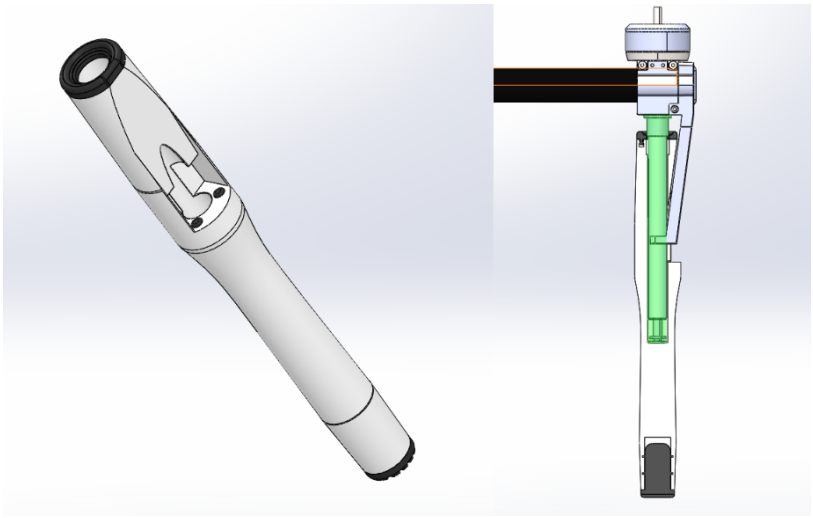

Figure 5.4: Left: The assembly for the second revision of the stilt.
Right: Cutaway view of the second revision in place on the M100.
Stock landing gear highlighted green.

A full set of these stilts were printed on the Ultimaker for testing. Rigid parts were printed in PLA, while the flexible bumpers were printed in Thermoplastic Polyurethane (TPU). In early flight tests pilots noted that the M100 felt sluggish and difficult to control when the stilts were mounted. The likely cause of this was a shift in the drone's moments of inertia due to the extra mass at its extremities. This was deemed unacceptable and a third revision was created.

5.2.5: Final Revision

The final version of the landing gear abandoned the addition paradigm which had guided previous versions in favor of total replacement. These were primarily composed of lightweight aluminum tubing with the same dimensions as the plastic tubes composing the original landing gear, cut to length by hand. Threaded caps were modeled in Solidworks and printed, allowing the tubes to be screwed into the sockets for the stock landing gear. Air springs were harvested from a set of stock landing gear which broke in an early test and inserted into the opposite ends of the tubes.

Figure 5.5: One of the M100 drones fitted with the aluminum legs.

5.2.6: Testing & Results

Final testing consisted of a simulation and live test phase. Static simulation in Solidworks indicated that a single tube of the correct dimensions and the intended length could support the full weight of the drone with a factor of safety of 26, as shown in Figure 5.6. The threaded caps were identified as a point of weakness under these conditions. This was deemed acceptable as they were easy to mass produce. Additionally, in the event of a landing at an angle the screw cap breaking would prevent the torque applied by the longer legs from being applied to the drone's body, reducing further damage. The entire set was found to be very lightweight, representing a weight savings of at least 50% over the previous version. Moments of inertia were also modeled using an official CAD model of the drone [1] with material properties applied. It was found that the Z moment in particular (see Table 5.1), which was cited as the likely cause of the previous sluggishness, was greatly reduced compared to the previous iterations.

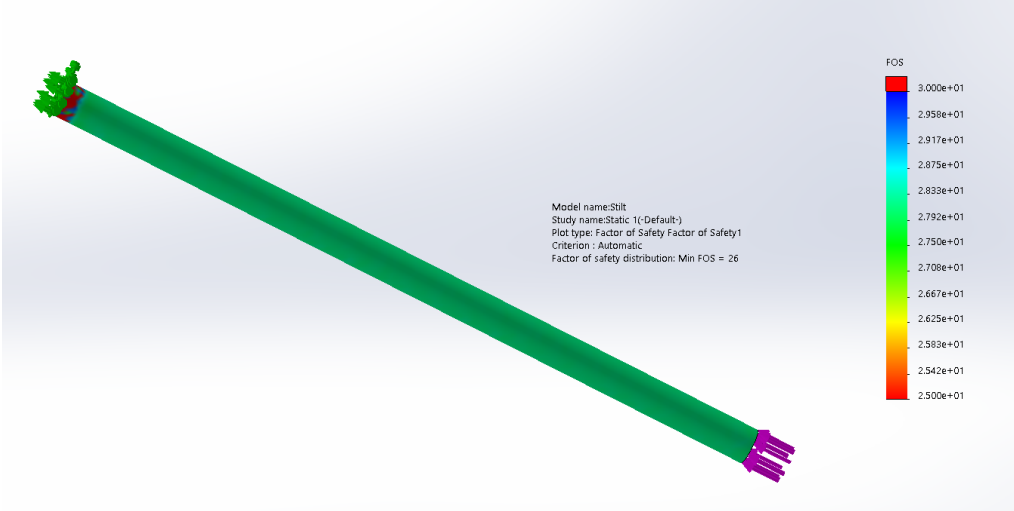

Figure 5.6: Factor of Safety plot from static simulation. The force applied was equal to the full weight of the drone. Note the minimum FoS at the bottom of the text box.

Table 5.1: Mass additions and Moments of Inertia about the Z axis by landing gear revision. Mass values based on experimental data when available, otherwise by Cura mass predictions. Moments of inertia based on Solidworks mass evaluations, with NUC and Hokuyo in place.

Revision	**Added Mass (g)**	**Z Moment of Inertia (g*m^2)**
Base	-	56.28
1	838	208.05
2	248	99.23
3	120	60.46

A full set of the aluminum landing gear was then fabricated, fitted to the drone (see Figure 5.5), and flight tested. The first test was performed with no other peripherals fitted to the M100. No adverse events occurred, and the pilot noted only a minor change in responsiveness. A second test was carried out with the pump system in place and dummy weights to simulate the other components of the intended final assembly. Under these conditions the drone experienced significant control issues consistent with previous tests involving the pump system. This prompted a shift in priorities for the pump system as outlined below, and while the aluminum legs were deemed acceptable on their own merits, they were also deemed redundant and scrapped.

5.3: Pump System

5.3.1: Rationale

Outfitting the M100 to act as the parent in challenge 3 required outfitting it with a pump system. This system would need to be fully controlled by the M100, free of leaks or unexpected pressure losses, and capable of relaying water consistently to the child regardless of its own absolute altitude or the child's relative altitude. It would also need to be securely mounted in such a way that it would change the M100's flight characteristics as little as possible.

5.3.2: Requirements

Our most immediate requirement was that the pump system be able to hold water without spilling and dispense it on command. This meant the robot must be capable of carrying the equipment and also of activating the pump without input from an operator. While the challenge did not specify a maximum amount of water which a drone may carry, it was estimated that due to weight concerns the maximum that could be carried would be 250ml.
As with the landing gear, the pump system must also create minimal disruptions in the drone's flight characteristics. This meant ensuring that its weight was as low as possible. It also had to avoid changes in its own center of gravity, either due to flexing or as the reservoir emptied.

Finally, the pump system had to fit within the grant proposal. This meant that it must be able to consistently send water to the nozzle carried by the child drone rather than engaging a fire on its own. We also had to ensure that the M100's pump system remained compatible with the corresponding nozzle attachment being developed in parallel for the Bebop.

5.3.3: Approach

Before anything else we addressed the question of a reservoir to hold water. Several potential solutions including a 3D printed bottle and commercial coolant reservoirs were investigated. However, all of these proved to be unsuitable, generally due to weight. Eventually we chose to use a commercial soft drink bottle with a relatively regular shape and a volume of 500ml, of the type shown in Figure 5.7. This provided us with an easily modified vessel which was lightweight, watertight, and readily available.

Figure 5.7: Photo of a soft drink bottle of the same brand and size as the one used in the pump system.

The other major component of the system was the pump. As one had previously been acquired by the team (see: Section 2.2.6) design proceeded on the assumption that it would be used for the final system.

Once the basic components were decided upon, they were modeled in Solidworks and placed in an assembly with the CAD model used in the landing gear simulations. Using the built-in mass evaluation tool various arrangements of parts were tried to identify how they altered the drone's center of mass. The lowest absolute disruption tended to occur when the reservoir and pump were stacked vertically below the drone with the three components' centers of mass aligned as in Figure 5.8. This also avoided lateral shifts in the center of mass as the reservoir emptied. While this did result in the center of mass shifting down significantly, it was hypothesized that this would be counteracted by the NUC and other components mounted on the top plate and would have minimal effect on the drone's overall flight characteristics.

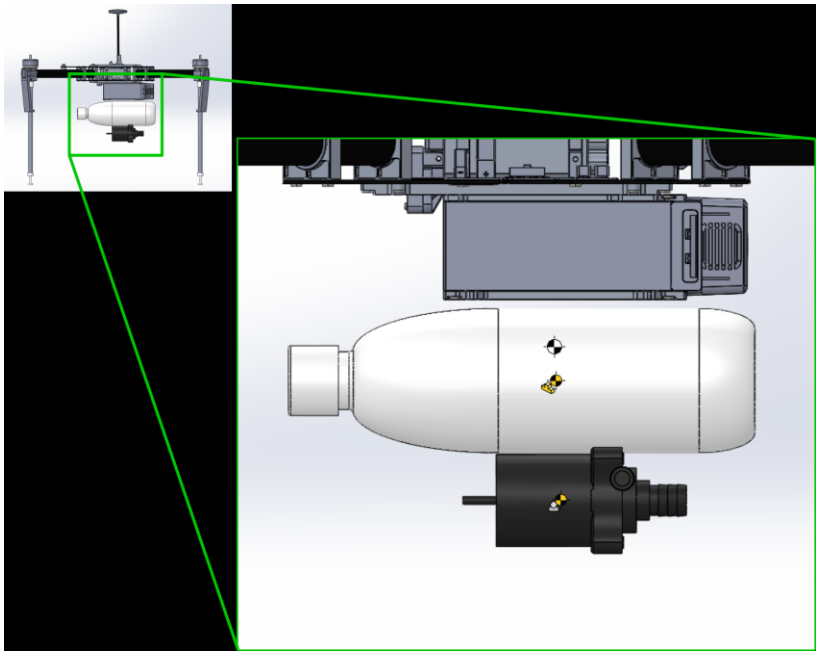

Figure 5.8: Positions of pump and reservoir relative to bottom of M100. View from left.

With the positions of the components established a retaining bracket was designed (see Figure 5.9). Mounting was accomplished using the screw holes for the drone's guidance sensors. In order to allow the guidance sensors to be used identical holes were placed on the bottom of the bracket. A continuously curved shell design was used to minimize weight while maintaining rigidity. This was printed on the Ultimaker in PLA. This included a custom coupling to join the closely spaced reservoir and pump.

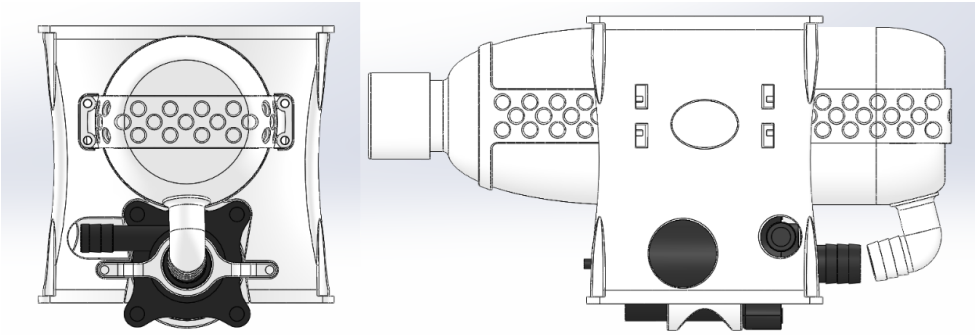

Figure 5.9: Pump system Solidworks assembly viewed from the rear (Left) and the left (Right).

Final integration of the system called for a voltage regulator board linked to the NUC to give the M100 control over the pump. However, this was not completed due to the results of the initial tests of the bracket.

5.3.4: Testing & Results

Once the bracket had been printed the system was fully assembled and put through initial field tests. The first of these consisted of a simple test where the pump was manually activated via an 18v battery while connected to a nozzle carried by the Bebop. This test was very successful, with no leaks or other issues on either end and the Bebop able to reliably place the stream on target.

Further tests focused on the M100's flight characteristics with the bracket mounted. This was done first with the revision 2 landing gear and pump system attached to the drone, and again later with the revision 3 landing gear as outlined above. In every case the drone would perform to expectations for a brief time before beginning to precess. This behavior would progressively worsen until a landing was made or a crash occurred. An exact cause of this behavior was never adequately established. However, it was hypothesized at the time that the drone's software was overcompensating for the lowered center of gravity. This was supported by further tests with weight placed only above the plane of the propellers in which the drone performed a variety of maneuvers without issue.

After these tests it was decided that the pump system should be redesigned so that it could be mounted to the top of the M100. Due to time constraints the sponsor assigned this task to other team members while our focus was shifted to the capture claw for challenge 1.

5.4: Capture Claw

5.4.1: Rationale

Challenge 1 required an M100 to be outfitted for the pursuit and capture of a payload carried by another drone. While other members of the team concentrated on the coding for this task, we concentrated on giving the drone the physical tools necessary.

5.4.2: Requirements

While the exact specs of the target payload were deliberately vague, some concrete information was available through the competition rules. The payload would be spherical, soft, have a mass between 100g and 150g, and be attached to the target drone via a carbon fiber rod and a magnet which would release it under a force of at most 4N. We therefore needed to outfit the drone with a mechanism capable of holding a payload of those sizes and exerting at least that much force without risking a break.

Once the catcher drone had received the target's position from the scouts, it had to be able to autonomously pursue the payload and react appropriately upon capturing it. Therefore, one or more relevant sensors would need to be worked into the design. Relatedly, since the drone would need to intercept the payload under potentially chaotic flight conditions, the mechanism would need to

react quickly upon registering a capture, and transition from a wide initial capture zone to a narrowly controlled grip. Due to time constraints the sponsor indicated that only providing a means to attach the sensors would be sufficient; testing would be carried out manually.

Due to some confusion with the rules (as mentioned in section XX), it was unclear if the catcher drone would be required to simply land with the payload after capturing it or deposit it in a scoring container while still airborne. As clarification was not forthcoming, it was decided that the capture system must enable the drone to accomplish either task.

Finally, as with previous peripherals, the system must disrupt the flight characteristics of the drone as little as possible. The Hokuyo was eliminated from the base sensor suite for this challenge, so some additional mass was freed up, but a maximum of 400g for the system was still set. Additionally, based on the results of the pump system tests, it was decided that any addition which lowered the drone's center of mass would be unacceptable. Upward shifts would be acceptable, provided they had a component in the XY plane of less than 5mm total. The acceptable range is shown in Figure 5.10. Lastly, as little of the system should extend far from the drone's main body as possible in order to minimize shifts in moment of inertia.

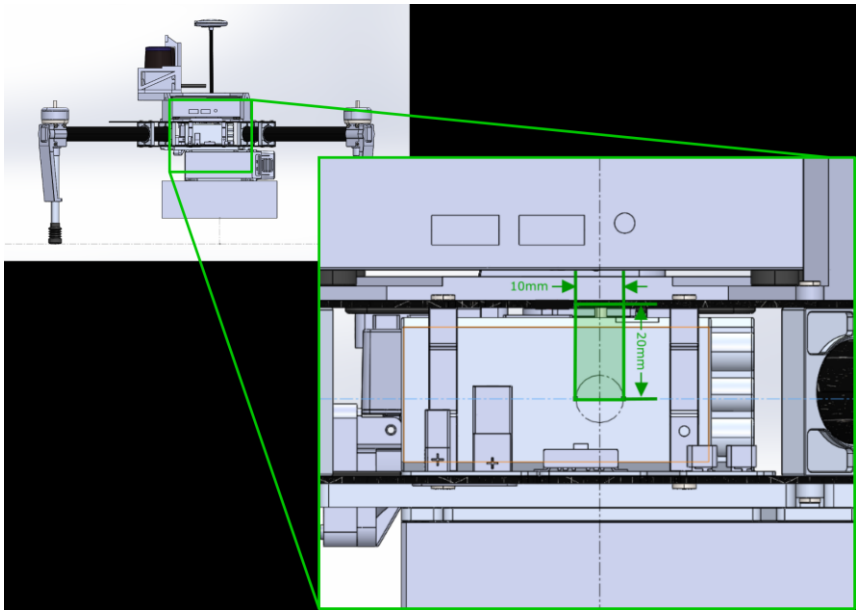

Figure 5.10: Sketch showing acceptable range of positions for adjusted center of mass. View from left. The center of the circle is its original position.

5.4.3: Gearbox

The design of the capture system was split into two major parts, the cage and the gearbox. Design of both began with establishing where the ball would sit once captured. After some deliberation it was decided that over the front of the main body would be the most advantageous spot, as shown

in Figure 5.11. This would keep the ball and most of the machinery close to the center, minimizing changes in moment of inertia, while leaving room behind the ball for sensor mounting. It would also reduce the possibility of the propellers contacting any part of the target.

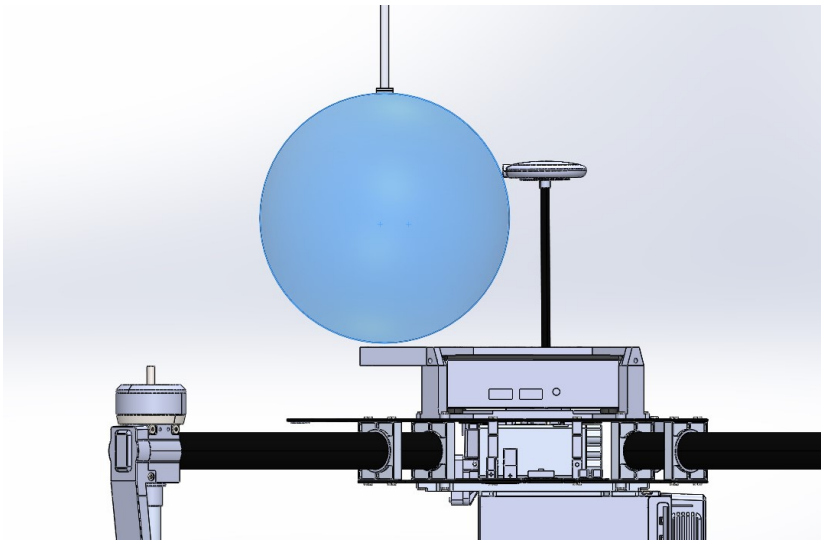

Figure 5.11: Resting ball position. View from left.

With the position of the ball established an actuator was designed to hold it in place against the planned cage. The actuator was to consist of a single claw driven by a servo. A free-body diagram of the claw was created to find the necessary torque at the claw's axle (see Figure 5.12). Since the exact size and orientation of the claw had not yet been established, the FBD was drawn with the assumption that the claw would experience the greatest force if it were to support the ball from below, and that a system capable of keeping its grip under these circumstances would be capable of operating under almost any other circumstances as well.

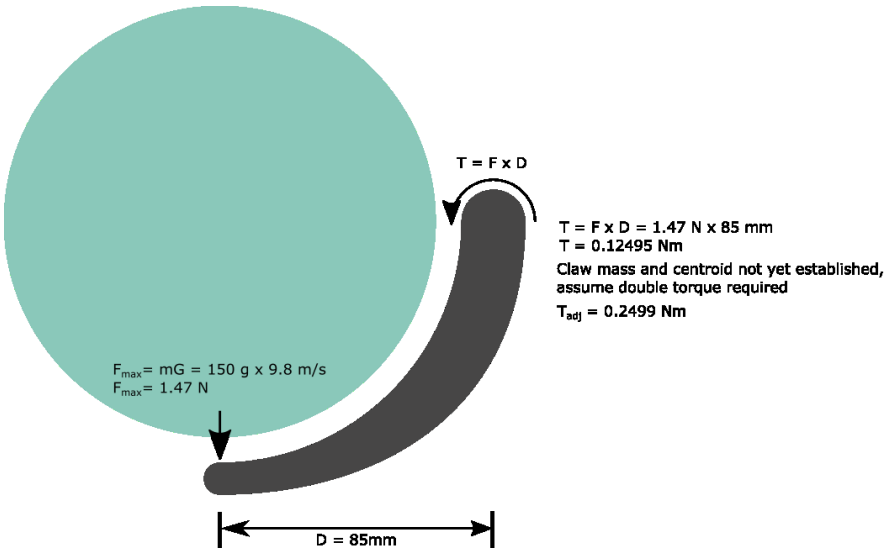

Figure 5.12: Free body diagram of max mass ball acting on claw.

Several servos were readily available from the lab's stores. After consulting their datasheets, a lightweight hobby servo was selected and a gear ratio chosen in order to provide a reasonable factor of safety:

*Required claw input torque: 0.2499 N*m*
Target Factor of Safety: 2
*Tower Hobby 9g servo output torque: 2.5 kg*cm = 0.245 N*m* [6]
*With 2:1 gear ratio: 0.245 N*m * 2 = 0.49 N*m*

$$\frac{T_{max}}{T_{reg}} = FoS = \frac{0.49\ N * m}{0.2499\ N * m} = 1.97 \approx 2$$

To ensure that the claw could close on the target within a reasonable amount of time, the servo's speed was checked in conjunction with the chosen gear ratio against the claw's predicted arc of travel:

Claw travel angle: 66.34°
Gear ratio: 2:1
Servo Speed: $\frac{0.1\ s}{60°}$[6]

$$\theta_{out} \times \frac{Out}{In} = \theta_{in} \rightarrow 66.34° \times \frac{2}{1} = 132.64°$$

$$132.64° \times \frac{0.1\ s}{60°} = 0.22\ s$$

Both the travel time and factor of safety were deemed satisfactory and design proceeded to the modeling stage. The servo and claw were modeled in Solidworks, followed by a gearbox to house them. There was little expectation of finding appropriately sized gears commercially or finding a machine shop to produce them before the deadline, so the gears were modeled and printed on the Ultimaker alongside the other components. Large involute teeth were used in order to improve resilience and minimize the impact of potential printing errors on gear performance. The gearbox and all of its component parts are shown in Figure 5.13.

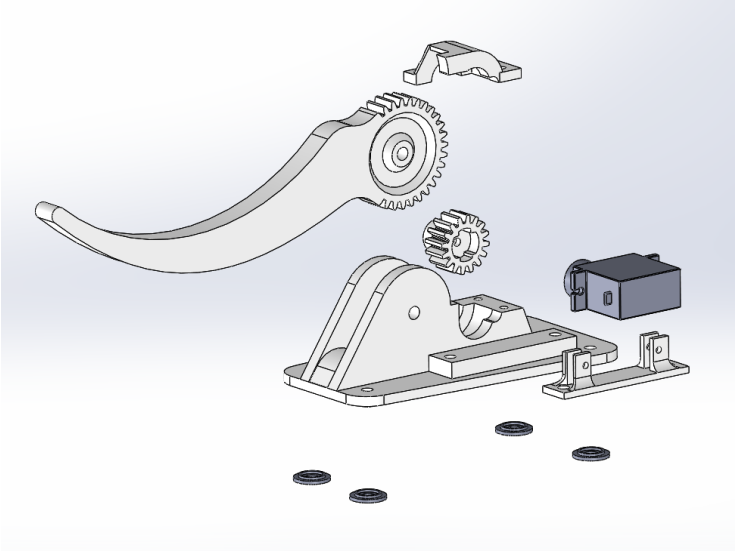

Figure 5.13: Exploded view of the gearbox. View from front left.

5.4.4: Cage

Once the gearbox had been modeled, the corresponding cage was modeled in Solidworks. This was sized so as to contain the largest possible target without leaving room for the smallest to escape. The main load bearing bars and their supports were put through physical simulations, as shown in Figure 5.14, in order to ensure that they could withstand both the initial collision with the ball and subsequent release from the target drone.

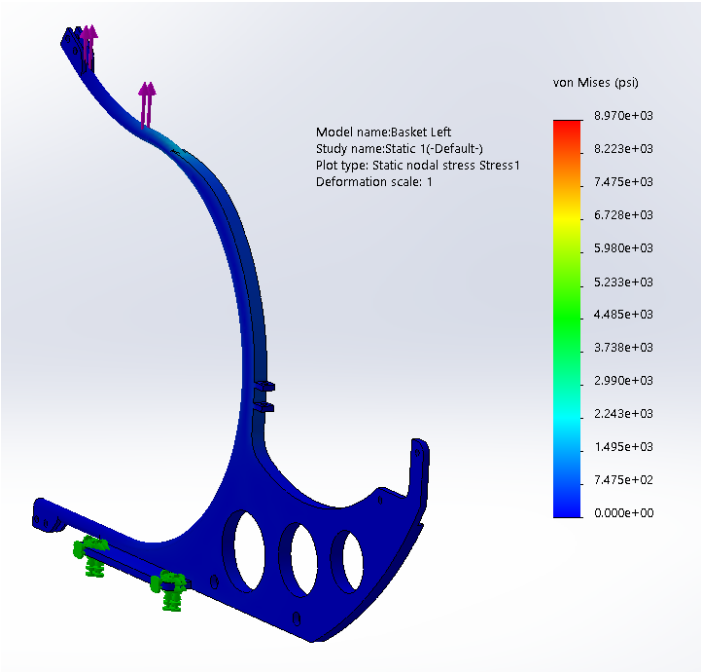

Figure 5.14: Static stress simulation of the cage basket main bars experiencing the force of 4 N necessary to release the payload from the target drone.

The top of the scale has been set equal to the standard flexural strength of PLA.

The front ends of the main cage components were flared outwards in order to create a “funnel” guiding the ball into its intended resting position during capture. Extensions were fitted to the open ends of the bars to further widen this funnel. As shown in Figure 5.15, the connections between the main components and these extensions were made modular both to ease replacement in case of broken parts and to simplify future changes to the geometry of the opening.

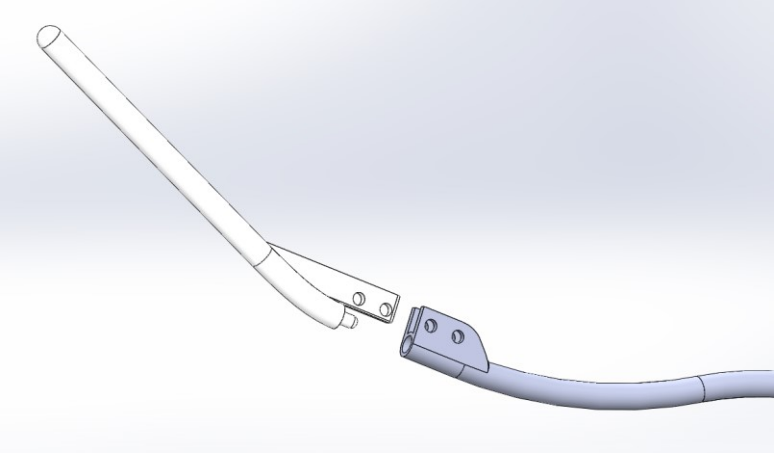

Figure 5.15: Detail of the connection point between the right-side cage riser and one of the funnel extensions.

A ramp composed of two parallel rails was placed in front of the cage as shown in Figure 5.16, extending past the narrowest distance between the propellers. This was intended to allow the drone to easily unload the payload by simply allowing it to roll forward and away. Guards were installed along the sides to prevent the payload from falling into the propellers on unloading. Both the ramp and guards were modeled as continuous curves in order to act as part of the capture funnel.

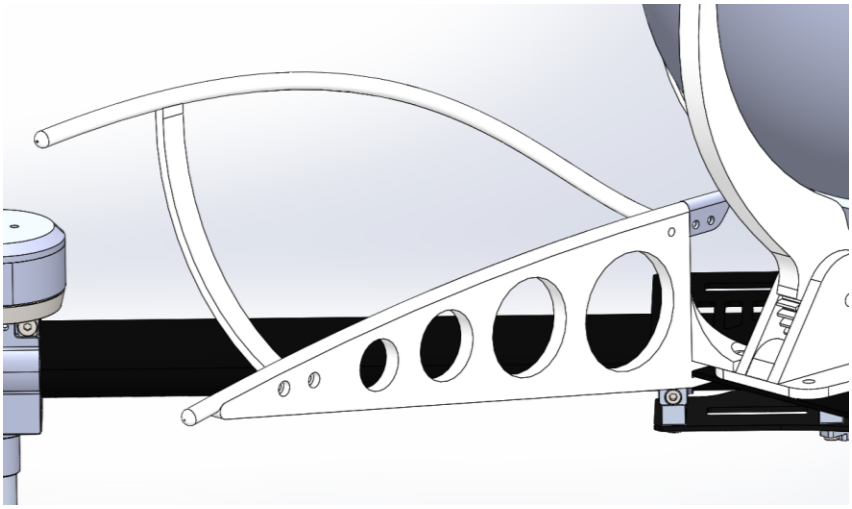

Figure 5.16: Right portion of the ramp and rail assembly, view from front left.

Note the use of the same modular attachment points as the funnel extensions.

On the opposite side of the ball, the cage supports were fitted with a pair of vertical supports joined to a baseplate connected to both support walls. This sensor bay was intended as an expansion for the future addition of a sensor such as a limit switch and included open space and unused screw holes to allow a variety of mounting options. The three pairs of screw holes were arranged such that the two nearest the cage formed a tangent line with the ball position circle, while the third was placed along the corresponding normal line, as shown in Figure 5.17. This was done to assist in alignment of future sensor additions, on the understanding that some might need to be oriented along a tangent or normal line to their target.

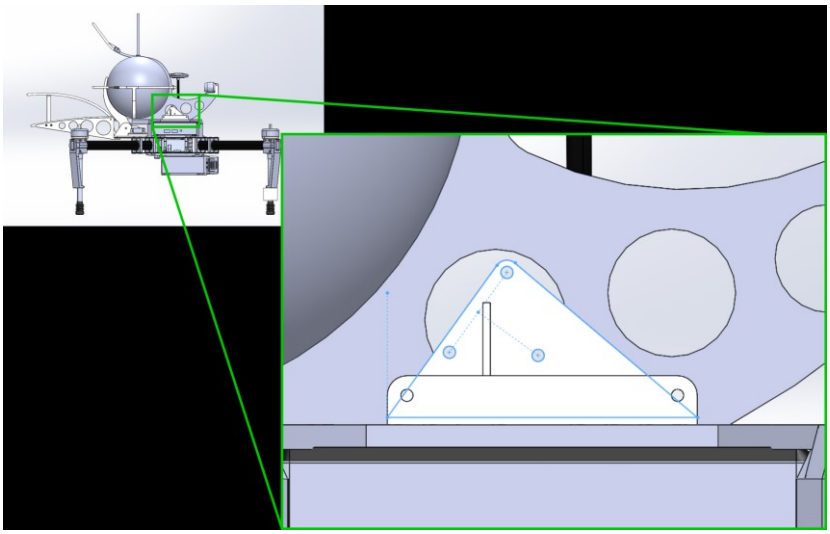

Figure 5.17: Sensor bay with ball in place, view from left. Note the plotting lines for the screw holes.

The RealSense depth camera to be used for target pursuit also had to be mounted. After discussing positioning options with the sponsor, the camera was placed directly behind the ball's captured position in order to ensure that the ball could stay within view during the entire capture attempt. While this would result in the camera being obstructed upon a successful capture, it would allow additional verification of capture, and it would still be possible for the drone to find its drop-off location using its GPS sensor. The camera's field of view was modeled in Solidworks in order to facilitate proper placement as shown in Figure 5.18, with the angles based on those found in the series datasheet [2].

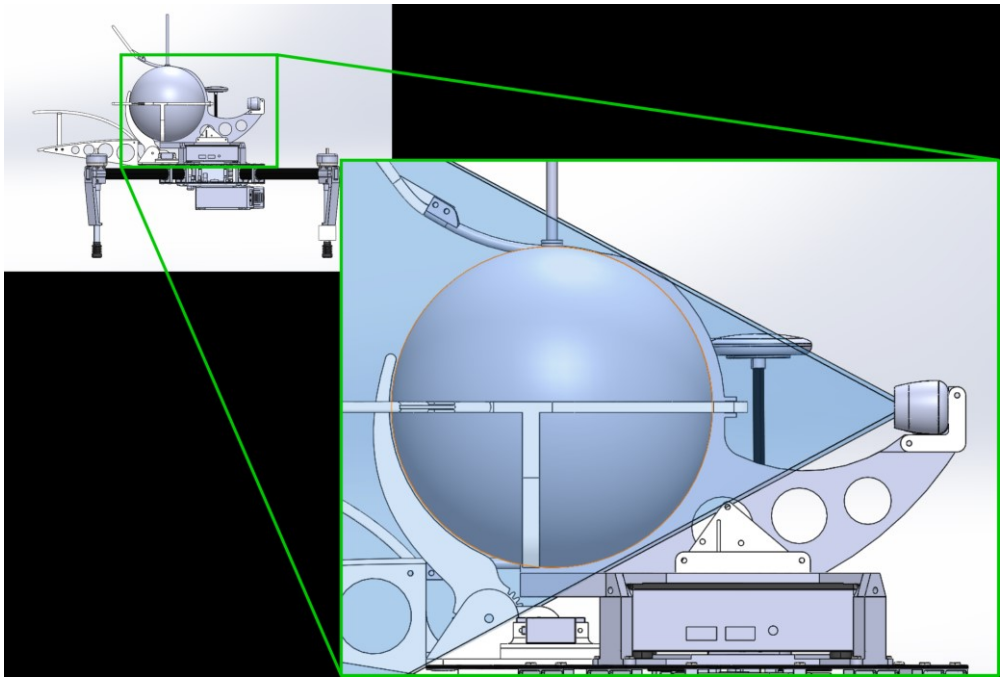

Figure 5.18: Placement of RealSense camera showing FoV cone enveloping the ball. View from left.

The final modeling task was to balance the system. Each part was exported to Cura and its estimated printed mass applied to its Solidworks model. With this applied it was found that the drone's center of mass had shifted up 9.7mm and forward 6.25mm. The vertical shift was deemed acceptable, but the forward shift was slightly outside of parameters. Using solid bodies with varying weights it was determined that attaching 41g weights (equivalent to the mass of an M10x35 bolt and accompanying nut) to each of the rear landing gear would reduce the horizontal shift to 0.34mm and the vertical shift to 6.42mm as shown in Figure 5.19. Fabrication of appropriate weights was left for future work due to time constraints.

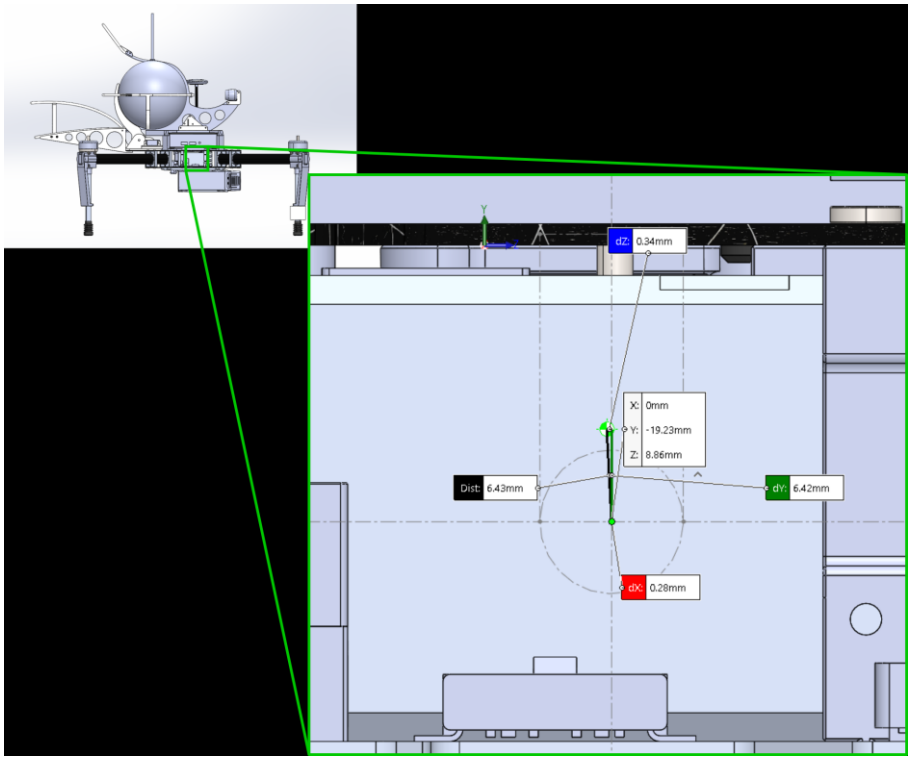

Figure 5.19: Updated balance information after masses attached to rear landing gear.

5.4.5: Testing & Results

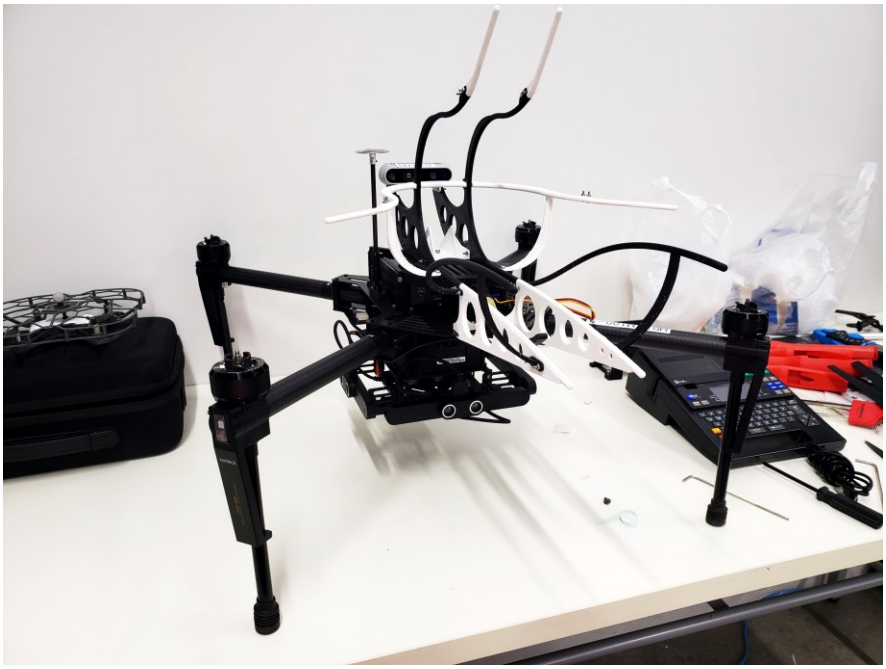
Figure 5.20: One of the M100s fitted with the full capture assembly.

Once all components were printed, they were assembled and fitted to the M100 as shown in Figure 5.20. For testing purposes, a small wi-fi enabled Arduino board was fitted to the drone with a dedicated 5v battery. Another team member added a simple program to this board enabling the servo to be remotely activated. This was successfully used to move the claw to its fully open and closed positions before testing proper began.

Once this was accomplished a static test with an 8cm diameter stand-in ball was performed. For this test the drone with full capture system was placed on a table. A team member then tossed the stand-in ball into the cage funnel while another attempted to close the claw before it could roll out. This test was successfully performed multiple times, with the ball being tossed with varying strength and from various advantageous and disadvantageous angles, and in spite of the stand-in ball's smaller size.

Next a basic flight test was performed to assess the drone's performance with the capture system attached. Despite the lack of the rear balancing weights the drone was easily controlled, performing basic maneuvers safely. A slight tendency to lose altitude when yawing was noted, likely due to the weight of the system, but was not determined to be a significant issue.

Following the testing phase the sponsor declared the capture system satisfactory, and responsibility for further refinement and development was transferred to other team members.

6. Conclusion

6.1 Results

While not all of the targets were met for the project, the overall effort was quite successful. Several important strides have been made in establishing a robust platform both for the MBZIRC challenges and for general drone research at Osaka University.

6.1.1 Complete Tasks

Despite many delays on the hardware, software, and documentation fronts, integration of the overall platform was successful. While some aspects such as Guidance access through ROS and depth camera integration remain incomplete, a strong framework has been created to allow these features to be integrated into the overall system. In the meantime, it is now possible for the M100 and Bebop 2 to be controlled via ROS as a single system. This will enable continued work on the particular MBZ challenges and use of the M100 as a research platform for other tasks.

Likewise, despite a tight schedule, the capture claw assembly was completed to specification with a minimum of tweaking. It has passed initial tests and is ready to move into more rigorous field testing scenarios. And with its modular construction it should be possible for changes to be made quickly and easily should the need arise.

6.1.2 Incomplete Tasks

On the other hand, some large tasks remain incomplete. Chief among these is the pump system. While aspects of this design were successful, it was an overall failure due to its unacceptable effects on the flight characteristics of the M100. This failure can primarily be attributed to assumptions made regarding the stability of the drone while carrying underslung loads. Early testing with dummy weights could have helped to avoid this pitfall. While it is unfortunate that our failure of foresight in this case created a waste of time and resources, the object lesson in planning which it provided served us well in later aspects of the project. Furthermore, some of the particular insights gained from this episode were instrumental in the completion on time of the capture claw system.

Relatedly, the redundancy of the landing gear provides an object lesson in the dangers of developing related parts of a system in tandem. While the final revision met all of the established criteria, the time and materials put into reaching that point were still wasted when the landing gear were made redundant. Again, this could have been avoided with more foresight and preliminary testing of hypotheses in the design of the pump system.

6.2 Recommendations for Future Work

6.2.1 General Upgrades

There were some aspects of the core platform which were not fully integrated by the time our project was concluded. These included ROS control over the Guidance module and use of the RealSense depth camera. Completing the integration of these devices will dramatically improve the versatility of the drone platform and its utility as a research tool.

Similarly, while the integration of ROS control for the Bebop 2 was completed, there was not adequate time for full field testing to ensure that there are no significant bugs left to be worked out. Therefore, we recommend that tests be run on this aspect of the system as soon as possible to confirm its complete status.

Finally, in line with ongoing concerns over payload weight, we recommend that consideration be given to alternate materials for custom peripherals. Most parts created for the drones were fabricated out of PLA filament on a 3D printer. While PLA is cheap and offers a decent strength to weight ratio, its usefulness in creating large, lightweight structural pieces is limited. The results of the landing gear design process demonstrate that for such components other plastics or lightweight metals may be more desirable. Composite solutions, such as continuous fiber reinforced 3D printed parts, may also bring down overall weight without sacrificing strength.

6.2.2 MBZIRC Specific

The first and most obvious item to be addressed is the pump system. Redesigning it to fit on the top side of the drone is a major priority, as this should eliminate the precession problem observed in testing. Furthermore, we recommend identifying a smaller and lighter water pump for use with the system. The existing centrifugal pump, while powerful, uses approximately 30% of the M100's total payload capacity. Using a lighter pump would increase the amount of water the drone can carry into the challenge, which will also increase the team's potential maximum score.

For challenge 1, further integration of the claw system with the M100 and ROS platform is required for successful completion of the challenge. This will entail giving the NUC control over the servo, as well as improving sensor integration. Full integration of the RealSense camera as outlined above will allow the drone to successfully pursue and capture the payload. The addition of a sensor such as a limit switch in the cage's sensor bay will also provide useful feedback, allowing the drone to decide to continue pursuit in case of a failed capture. Live flight tests should also be carried out using the capture system to determine if the cage geometry is suitable. In the event that it is not, the modular construction should make changes relatively easy to make.

7. Bibliography

[1] DJI M100 General Specs & Downloads. https://www.dji.com/matrice100/info

[2] Intel RealSense 400 series datasheet. https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf

[3] JSK Lab jsk_uav_forest Git Repository. https://github.com/tongtybj/jsk_uav_forest

[4] MBZ Briefing Video. https://www.youtube.com/watch?v=l5aPjTNYcpc

[5] MBZIRC Rules. https://www.mbzirc.com/challenge/2020

[6] Tower Hobby 9g servo datasheet. http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf