RVC-2202

Low-Cost Quadrotor Micro-Aerial Vehicle

A Major Qualifying Project Report
submitted to the Faculty of the
WORCESTER POLYTECHNIC INSTITUTE
in Partial Fulfilment of the Requirements of the
Bachelor of Science Degree
in Aerospace Engineering
by

Akul Agarwal

Antonio Calcagni

Martin Runquist

Nicolas Hesel

March 24, 2022
Approved by:

Raghvendra V. Cowlagi, Advisor
Associate Professor, Aerospace Engineering Department, WPI.

Abstract

The objective of this project was to design and construct a low-cost quadrotor micro-aerial vehicle (MAV) with a multi-sensor payload capable of indoor flight using localization by a motion-capture system. Design considerations included low overall weight, ease of construction, and ease of replacing the sensor payload. To meet these objectives, the literature was thoroughly reviewed. An existing MAV frame design was adapted and modified. Motors and propellers were chosen to accommodate the desired sensor payload. An autopilot was implemented in conjunction with a high-level microcontroller for future implementation of intelligent control algorithms. A series of unit tests and bench tests were conducted within the motion-capture environment to demonstrate feasibility of the design.

Acknowledgements

Our team would like to personally thank a few people who have been very helpful throughout the process of our MQP project. Professor Cowlagi, our advisor, has been a tremendous help by always leading our team in the right direction and always pushing us to think about our project objectives in a different way. Many others were instrumental in helping us succeed with the Vicon motion capture system. We would like to thank Chase St Laurent, a WPI PhD student, for his help with the physical set up of the system, the Vicon support team for answering our countless troubleshooting questions, and Ed Burnham, a WPI Research Data Scientist for lending IT services. It was a long process to get all the pieces together and make the program function properly for our project and we appreciate their patience with us and the system.

Thank you.

Table of Authorship

Section	Project Work	Report Writing
1. Project Overview	All	AA
1.1. Project Objectives	All	AA
1.2. Literature Review	All	AA
1.3 Design Restraints and Other Considerations	All	AA, AC
1.4 Project Management	All	MR, AA, NH
1.5 Relevant Engineering Standards	All	AA
1.6 Methods	All	MR, AA
1.7 Broader Impacts	AC	AC
2 System Design	All	
2.1 Flat Frame Components	All	NH
2.2 Propellers	All	MR
2.3 Microcontroller	All	AA, NH
2.4 Flight Controller	All	AA
2.5 Battery	All	NH
2.6 Power Distribution Module	All	NH
2.7 Power Distribution Board	All	NH
2.8 Motors	All	NH
2.9 Electronic Speed Controllers (ESCs)	All	NH
2.10 Motion Capture Software	AC, MR	MR
2.11 Ground Station Software	AC, MR	AA
2.12 ODROID XU4	All	AA
2.13 Universal Sensor Clamp/Holder	All	NH
3.1 Electrical System Iterations	All	NH
3.2 SolidWorks	NH	NH
3.3 Thrust Stand Testing Using RC Benchmark	AC	AC
3.4 Vicon	MR, AC	AC
3.5 Ground Station (Mission Planner)	AC	AC
4 Results	All	MR, AC
4.1 Vicon Experimental Results	AC, MR	AC

4.2 MAV Experimental Results	AC, MR, AA	MR
5 Conclusions	All	MR
5.1 Recommendations for Future Work	AC, MR	AC, MR
6 References	All	All
7 Vicon Motion Capture System Manual	All	MR
7.1 Hardware and Physical Setup	AC, MR	MR
7.2 Software Setup and Troubleshooting	AC, MR	MR
7.3 MATLAB SDK Interrogation	AC, MR	MR
7.4 Troubleshooting	AC, MR	AC, MR

1 Project Overview

The goal of this project is to design a MAV that will utilize a Vicon motion capture system as its navigational means and carry a sensory payload from Point A to Point B. The essential electronics the MAV will be carrying will be a single-board computer (SBC), a flight controller, sensors, electronic speed controllers, a power distribution board, and motors with propellors. The single-board computer will interface with the flight controller, the sensory packages, and a ground station computer. This iteration of the MQP seeks to merely ensure that a connection between the SBC and the sensory packages exists and that the two can be accommodated on a single MAV. It is up to future iterations of the MQP to make use of the data collected from the sensory packages towards some larger goal such as using the volume level from the microphone to fly towards noisier or quieter areas. A crucial aspect of this iteration of the MQP is ensuring the design is low-cost and easily reproducible by future users. To this end, the main "body" of the MAV can be completely 3D printed and requires simply screws and standoffs to hold together. Making the body 3D printable also makes the design flexible as per future users' requirements and allows for changes to easily be made.

Unmanned Aerial Vehicles (UAVs), in modern times, are used for a variety of civilian and military applications. Most UAVs as thought of, by the general public, are aircraft that are large, capable of carrying massive payloads and remaining airborne for long endurances. These are inevitably military devices, used by the various defense forces of the world for a variety of roles. Such roles include surveillance and intelligence gathering of targets, border security, coast-line defense and combat (1). UAVs specifically designed for combat are termed as Unmanned Combat Aerial Vehicles (UCAVS) and they are a rapidly growing subsection of UAVs. Micro Aerial Vehicles (MAVs) are another subset of UAVs that have become popular in recent years due to their versatility as well as portability. They are also the considered superior in situations that require an inexpensive solution or where there is a chance of harm occurring to a human pilot (2). MAVs are regularly used by several defense forces around the world for a variety of roles such as surveillance of targets or operations, gaining intelligence of an area of interest and by a few, even for combat. The Black Hornet PRS is one such drone (3). Developed by Proxy Dynamics, it was designed for use by non-specialist soldiers to give them immediate situational awareness, without causing hinderances. The 10cm by 2.5cm drone can fit in a pocket and boasts a flight time of 25 minutes. It transmits live video as well as captures images in infrared, night vision and visible spectrum.

Navigation in these scenarios is done with the aid of GPS and sensors onboard the MAV that record the aircraft's attitude, accelerations in the three directions and, magnetic readings (4). All these quantities are then fed to an Extended Kalman Filter (EKF) that outputs the UAV's position, heading, speed and whether it is flying in the requisite direction. The measurements that the UAV makes of the accelerations, attitude and magnetic are done by the Inertial Measurement Unit (IMU) and are not fit to be used as the sole means of navigation, which is where the GPS and EKF come in. Without the GPS, the aircraft's localization, or idea of where it is relative to the surroundings, will deteriorate and continue to do so as the duration of the mission increases. This is known as drift and is a quantity that is minimized as much as possible.

There are environments where the optimal solution would be to use a UAV, but the GPS signal may be too weak for the UAV to use or denied entirely. In such cases, the easiest method of control is simple line-of-sight operation of the UAV. When line-of-sight is not an option, vision-based methods are used. These are software that make use of the incoming visual odometry data from the camera to gather captured images and conduct analysis on them. Visual odometry packages are reliable, allow for real-time analysis and localization of the aircraft. They face issues in that if the UAV happens to remain stationary for too long in a fixed spot, the drift will increase. Simultaneous Localization and Mapping (SLAM) methods make use of the same visual odometry data, for UAVs to "simultaneously" deconstruct the environment around it to make a map, as well as orient themselves within that map to localize themselves. A third way that is used for navigation in indoor GPS-denied environments is via motion capture. By attaching markers to the body of the aircraft and tracking their movements as they occur, the navigation of the UAV can be handled autonomously. The goal of this project is to use such a motion capture system to control an aircraft bearing sensor packages with the aim of recording data pertaining to the environment of deployment.

Background

An unmanned aircraft system (UAS) constitutes of 5 coordinated disparate elements: A UAV, a control station, a launch and recovery system, a maintenance and support system and an operating payload (4). Colloquially, UAS and UAV are used interchangeably along with terms such as "drone", "quadcopter" or "aerial robot." To better understand what each of these terms means and clear away any misconceptions of exactly what the whole Unmanned Aerial scene entails, it is necessary to start with what exactly an UAS is. After a brief discussion of the history of global UASs and the events that led to their emergence, the paper will expand on the term UAV and the air vehicles that define it before proceeding to MAVs, an offshoot of UAVs.

The first generally accepted use of an UAS was by the British Royal Navy (5), which operated a remotely operated airplane for aerial target practice as part of training sessions. The plane is question was the de Havilland DH.82 Tiger Moth and a total of 380 were used. They could attain an altitude of 17,000 feet and had an operating range of 300 miles. Interestingly it is from this very plane that the term "drone" arose. A drone is a male bee that makes a singular flight for the queen bee before perishing (6).

The success of the British in utilizing RC technology spurred the US on and the US Navy began its own program to develop radio-controlled aerial targets. Major advances in UAV technology next arose in the Vietnam War. During this period, planes had been modified to act as decoys, conduct surveillance and reconnaissance behind enemy lines, gather intelligence through photographs and drop propaganda leaflets (5). The UAVs in question were the AQM-34 Ryan Firebee and they were launched from a "mother" plane, Lockheed DC-130, which would simply drop the UAV from its underwing pylons to begin the mission. The Firebee was a very reliable tool for the USAF and prevented human losses while providing crucial information. The recovery system for the Firebee was a parachute that would deploy upon mission completion and allow for the UAV to be caught by a helicopter.

From the Vietnam War onwards, the next country making leaps and bounds in the UAV field would be Israel. In 1982, Israel launched Operation Mole Cricket 19 against Syria. What makes this conflict a pivotal moment in history is that it was the first recorded and legitimate use of UAVs on the battlefield to carry out critical aspects of a combat mission in a successful fashion. Israel also developed the Scout and Pioneer aircraft; they are both small and light aircrafts that could be tactically deployed at short-notice and provide excellent surveillance (5).

In the 1990s and 2000s were when UAVs became a dominant tool used in the various defense forces throughout the world. Their unmanned nature and plethora of information-seeking abilities allowed for militaries to keep a real-time track of events on a battlefield halfway across the world and not lose troops to the pitfalls of manned aviation (7). Some famous UAVs that made an emergence in these times were the American RQ-1 Predator, the joint American Israeli RQ-2 Pioneer and the Israeli Firebird. Fast forward 10 years and UAVs have started making appearances in consumer and civilian spaces. Companies like Parrot and DJI response to consumer commercial interests by producing UAV technology tailored for aerial photography, industrial work, data collection or mapping. It is becoming increasingly popular for all sorts of events to occur with a specialized video UAV overhead, taking gorgeous captures of vistas, controlled via a smartphone or remote control (8).

A micro-aerial vehicle is a specific type of UAV, one with a weight between 200 grams to 2 kilograms. Using weight as a means of classification is just one of many ways. Other methods include using wingspan, flight range or even intended use. Regardless of classification method, the defining characteristic of MAVs is their small size. Military applications of MAVs are surveillance, reconnaissance and search and rescue operations. Civilian and scientific applications of MAVs include traffic monitoring, urban mapping, cloud and coast monitoring and rainforest observation (9).

1.1 Project Objectives

The project objective is to design a MAV that is capable of carrying sensor modules along with a micro-computer. The proposed design should be light, easily assembled by one person, and scalable to allow for a swarm to be manufactured if necessitated. The MAV must utilize a Vicon Motion Capture system for its navigation rather than the Global Positioning System (GPS). The MAV will use the Vicon Motion Capture to orient itself within an indoor environment and be controlled using a ground control station running a ground control software. The flight controller of the MAV must communicate with the onboard micro-computer to allow for flight autonomy. The final iteration of the MAV project is aimed to be able to take off from an arbitrary point, utilize the Vicon Motion Capture System to fly to another arbitrary point, collecting sensor data along the way and then land in a smooth and safe manner. Requirements that the MAV must meet along with the objective are being light ($\approx$ 500 grams) and compact to store, capable of being constructed singlehandedly and have a flight time of 15 minutes.

1.2 Literature Review

Literature review for the MAV's construction consisted of an examination of modern quadcopter design theory, basic quadcopter frames used by other research projects as well as recreational projects and popular commercial drones such as DJI (8) and Parrot (10). Previous MQP designs for competitions were also examined in a hands-on fashion at Professor Cowlagi's laboratory. From these three sources of information, the team narrowed down frame layouts, component choices, component placements and frame materials.

A quadcopter has four motors attached to the ends of its four arms. The motors have propellors attached to them, oriented vertically. These propellors when spun are what generate the thrust allowing the craft to fly. The arms extend from the central frame, where the components are housed. The four motors evenly carry a quarter of the weight of the quadcopter if the components are arranged correctly. Two of the motors spin in the clockwise direction and two spin in the anti-clockwise. Motors spinning in the same direction cannot be placed in adjacent quadrants of the quadcopter. Additionally, the motors must all be on the same vertical plane. Both clockwise and anticlockwise rotation is necessitated because otherwise the quadcopter would simply spin in place. For example, consider a quadcopter with the X design. Common design types will be discussed in the following subsection. If the front right motor of this hypothetical quadcopter spins clockwise, the other clockwise motor must be the back left one.

The thrust generated by the propellors is measured in force units, metric and imperial. Propellor acceleration generates thrust, moving the system up and down on the Z-axis. Rotation of the quadcopter about the Z-axis is dictated by the net torque about the Z-axis. By default, this quantity is zero, balanced by the equal and opposite clockwise and anticlockwise motion of the motors. Shifting the net torque from zero rotates the quadcopter. When the clockwise torque is greater than the anticlockwise torque, the quadcopter will rotate in a clockwise direction and when the anticlockwise torque is greater than the clockwise torque, the quadcopter will rotate in an anticlockwise direction. The torque in a direction can be changed by changing the rotation speed of the motors for that direction (11). An important quantity is the ratio of the thrust generated to the weight of the quadcopter. In general, the maximum possible thrust that can be generated by the motors should be twice the weight of the quadcopter. This is done to ensure that flight is possible without risking motor failure by constantly having the motors at maximum thrust. Brushless DC motors are the most commonly used type of motors for quadcopters (12). This is due to their numerous advantages over brushed types (13). The first comes from the name itself. Brushless motors have no brushes and so periodic replacement of components in not required, reducing overall maintenance. Brushless motors are also more electricity efficient than their brushed counterparts due to the lack of brushes. The brushes of a brushed motor are actually the commutator, a rotary switch that reverses the direction of current between the rotor and the external circuit. Brushless motors do away with the physical commutator and handle the commutation electronically. By doing so, friction from the brushes as well as the electrical resistance of the brushes themselves is eliminated, improving electrical and torque efficiency. The third main advantage that brushless motors offer is a higher torque to weight ratio due to their lower weight.

Propellers can be two-bladed, three-bladed, or four-bladed. The number of blades affects the control of the aircraft and thrust generation (11). Fewer blades offer greater speed and efficiency at the cost of control. Propellor size and material are also important factors to consider. Longer propellors generate more thrust at the same speed than shorter ones, however they require more torque to get them spinning. Propellor size also influences propeller pitch. Pitch is the distance travelled per revolution of the propeller. A low pitch offers more torque and higher stability than a lower pitch. This is because the propeller is reaching the operation revolutions per minute at a slower speed. Propellor material affects how stiff the propellor will be as well as its durability. Materials with a lower density will need less torque to acquire the same revolutions per minute but may be more fragile. They may also be more susceptible to damage due to vibration and can flex if the propellor spins at a particular speed, which can introduce undesirable effects.

Modern quadcopter design theory dictates that the final application of the MAV is the most influential factor on the design (1). An MAV meant for combat applications will be more likely to have an assortment of cameras, sensors and weapon systems, than a civilian MAV. A recreational racing MAV will likely have as little weight as possible, to maximize flying speed and ensure smooth handling. To this end, the team’s final application was very clear and that greatly simplified the design process. The autonomous nature of the project meant that we had to include a micro-controller. The MAV needed to be operated in an indoor environment, so a global position system (GPS) was not required. For the same reason, the battery did not need to be able to provide charge for an extended duration. In an outdoor environment, the user may navigate the craft too far away for it to be able to return to their vicinity, but the same cannot be said for an indoor environment. If the battery level becomes too low, the user can simply land the vehicle and navigate to it to retrieve it.

1.2.1 MAV Basic Designs

MAV quadcopter frame shapes are dictated by the placement of the arms relative to the central body. There are four commonly used frame styles: True X, Hybrid X, Stretched X and H (14). These are all pictured below in Figure 1. True X is as the name implies, when the arms are all evenly spaced from the central body and the ends of the arms form a geometric X. This frame shape is very stable. H frames resemble a I shaped beam. The arms are attached perpendicularly to the body. These frames can necessitate long arms if the propellors are long, to ensure there is no contact between the body and the tips of the propellors. Hybrid X frames blend the H and true X frames. The angle of the arms is adopted from the true X layout and combined with the longer central body of the H frame. These are the most common type of quadcopter frames. Stretched X frames are similar to the hybrid X frames, with the difference being that they have a greater distance between the front and rear arms. The greater distance between the front and back means more material is used, making the stretched X frames slightly heavier than the other X frames. Another frame choice, not pictured, is where the arms of the quadcopter are integrated to the central body. This makes the entire frame one piece, which comes with its own set of advantages and disadvantages. Fusing the arms to the body reduces the number of linkage points, reducing the chance of mechanical failure. It also makes the frame stiffer than if the arms had been separated. The joining does mean that if there is a crack somewhere or if an arm breaks that the entire frame must be replaced rather than just one arm.

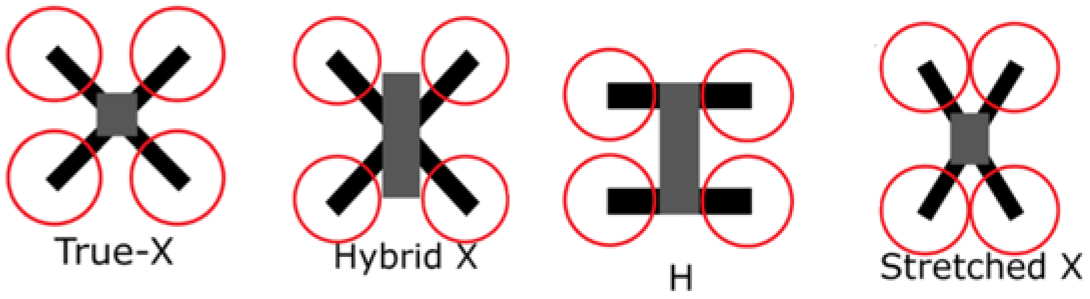


Figure 1: Quadrotor MAV Configurations (Image taken from DroneNodes.com)

1.2.2 MAV Control

MAV control has two main aspects to it (1). There is the remote control for the actual flying and movement of the craft and then there is attitude control of the craft. Attitude control is handled by the flight controller and motion control is handled by a remote controller and partially by the flight controller as well. The flight controller is the “brain” of the quadcopter. It contains sensors such as accelerometers, magnetometers and barometers. These sensors measure acceleration, magnetic field strength and pressure respectively. The flight controller also makes use of a model of the quadcopter. This model’s behavior is determined by the number of degrees of freedom it was designed with. A model with a greater number of degrees of freedom will paint a more detailed picture of how the craft will respond to stimulus. The flight controller combines this model with the measurements it makes to calculate the “attitude” of the craft. The attitude is essentially information on how the craft is oriented at an instant of time. Once the attitude is calculated, the flight controller uses it along with user input to alter the speed of the motors to move the quadcopter as per the user's desire.

1.3 Design Requirements, Constraints, and Other Considerations

The following were the design requirements the project was completed with in mind. A few of the requirements were provided at the inception of the project and some were decided by the team while undertaking the project, in order to make the project our own and simplify some choices.

The MAV:

- Must be under 500 grams total.
- Must be under 400 grams without the sensors.
- Must use Vicon Motion Capture.
- Must be able to autonomously travel from a point A to an arbitrary point B within reasonable distance.
- Must be low cost (around $100 to assemble given instructions).
- Must be easy to set up (given instructions a person could make one in a day).
- Include a controller.

Some of the considerations we had to make while working on our project were related to the manner we worked in the laboratory. We had to always be careful to disconnect the battery before altering the electrical components in any manner as doing so with a connection would have shorted out some of the more sensitive components. The Vicon Motion Capture system is extremely sensitive to perturbations and the slightest movement while it is operating throws the system out of calibration, so we had to keep that in mind while carrying out the test flights. Additionally, the MAV had to be properly armed and disarmed every time we carried out the flight tests. Failing to disarm and accidentally triggering the controller could have caused damage to both the team and the MAV. Appropriate conduct with the MAV also extended to the way it armed before the test flights. The Pixhawk carries out a burst of tests before arming the MAV and if voltage is too low, or the throttle is being activated before the craft is ready, the procedure to arm will stop. Mission Planner notifies the user and asks if the MAV should be force armed rather than regularly armed. Force arming is not something this group carried out, as the risks are far too great.

1.4 Project Management

At the onset of the project, the team divided into sub-teams to allow for a deeper focus and understanding of different aspects of our project. Antonio and Martin worked together to learn the Vicon motion capture system. Nicolas focused on the design and modeling of the frame, while Akul worked with the microcontrollers. Antonio and Akul also collaborated on Pixhawk familiarization.

Aside from these major subdivisions, team members were responsible for other, smaller tasks. Martin submitted purchase requests as well as performing 3D printing related tasks. Antonio and Nicolas often focused on the electrical assembly and propulsion subsystems of the MAV, including soldering, and thrust testing of the engines. Additionally, Akul worked on the QGroundControl software. Later in the project, Akul, Antonio, and Martin collaborated when working with Mission Planner software to communicate with and control the MAV.

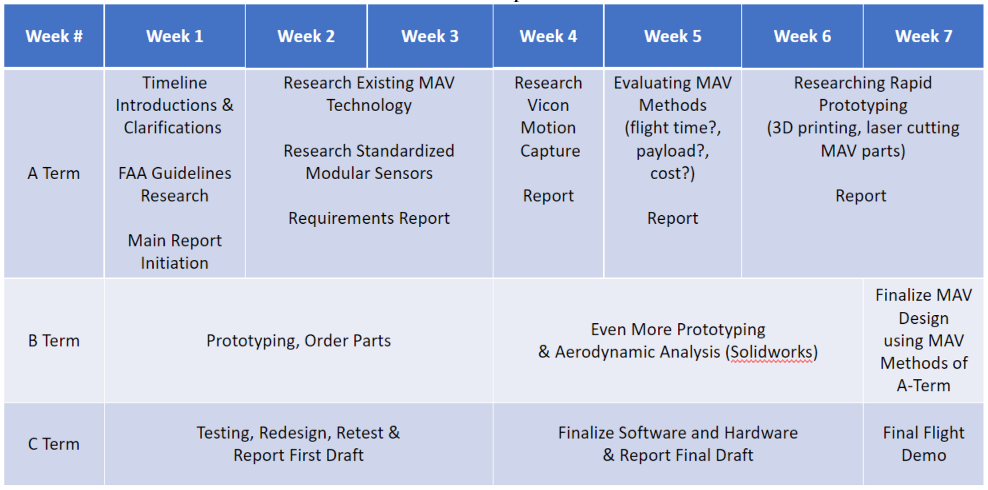
For the first several months of the project, the team and advisor met weekly to discuss progress, upcoming goals, and any questions or adjustments to the project. Later, these meetings occurred less frequently on an as-needed basis. Our team of four held three independent meetings weekly, on Monday, Wednesday, and Friday. These meetings were scheduled in advance to work with the varied academic schedules of the team members. Additional meetings were scheduled as necessary to complete specific time-sensitive tasks or meeting with outside persons, such as Vicon support staff.

As an engineering team we also made sure to uphold the Code of Ethics for Engineers as outlined by the National Society of Professional Engineers. The duties include the following listed below:

1. Hold paramount the safety, health, and welfare of the public.
2. Perform services only in areas of their competence.
3. Issue public statements only in an objective and truthful manner.
4. Act for each employer or client as faithful agents or trustees.
5. Avoid deceptive acts.
6. Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.

1.4.1 Tasks and Timetable

Table 1: A-term Proposed Timetable



Week #	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
A Term	Timeline Introductions & Clarifications FAA Guidelines Research Main Report Initiation	Research Existing MAV Technology Research Standardized Modular Sensors Requirements Report		Research Vicon Motion Capture Report	Evaluating MAV Methods (flight time?, payload?, cost?) Report	Researching Rapid Prototyping (3D printing, laser cutting MAV parts) Report	
B Term	Prototyping, Order Parts			Even More Prototyping & Aerodynamic Analysis (Solidworks)			Finalize MAV Design using MAV Methods of A-Term
C Term	Testing, Redesign, Retest & Report First Draft			Finalize Software and Hardware & Report Final Draft			Final Flight Demo

Table 2: B-Term Proposed Timetable

Week #	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
B Term	Detailed B Term Schedule Order More Components Continue Learning Vicon Setup Test XU4, Jetson, & Pixhawk		Order any last components 3D Modeling Frame & Sensor Mount Continue learning processor and Pixhawk communication	Finalize Frame and Begin Printing Motor Thrust & Battery Power Testing Continue processor and Pixhawk communication	3D Print Frame Before start of Week 5 Assembly of Drone Preliminary Testing *Reprint if needed*	<u>Testing of Quadrotor:</u> Are the thrust of motors sufficient? Is the battery power enough? Are the Pixhawk and Processor communicating properly with each other and the ground station? Are the sensors communicating/outputting as expected?		
C Term	Testing, Redesign, Retest & Report First Draft			Finalize Software and Hardware & Report Final Draft			Final Flight Demo	

Table 3: C-Term Proposed Timetable

Week	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
C Term	Set plan for C Term Get back into Vicon** Order Last Parts Finish Modeling and Printing MAV Odroid with Microphone Research		QGroundcontrol and Vicon Connection Test Motors Begin Assembling MAV Develop Test Flight Procedure and Evaluation	Fully Assembled MAV Test Sensor Holder	Integration of Software and TESTING Report Writing	TESTING Report Writing	Complete Testing

The above tables document the manner in which the team divided their work up across the three terms. During the weekly presentations, the week of the term that it was at the time was highlighted green and the progress made towards the listed objectives was presented to the advisors. For the most part, the team stuck to the proposed timetables and completed tasks at the speeds we had initially thought we would. There were a few setbacks, namely with the Odroid initial testing and towards the end of the project with the test flights. One of the ESCs for the motors was incorrectly plugged in to the Pixhawk 4 Mini and that caused it to short out. Obtaining another set of ESCs quickly proved to be difficult and in the subsequent delay, less testing of the actual completed MAV was done than hoped. Additionally, a team member contracted COVID-19 and to adhere to university standards, the entire team remained in isolation and conducted meetings virtually rather than in-person. This also took away valuable testing time.

1.5 Relevant Engineering Standards

For on board autonomy, the Robot Operating System (ROS) standard was used. The specific version of ROS was mavROS Kinetic. mavROS is a communication node used for communication between the ground control station through the Odroid and the autopilot running within the PixHawk 4 mini (15). For the hardware, namely the flight controller, the Pixhawk open standards were used (16).

1.6 Methods

Table 4: Various methods applied to subsystems.

Method	Subsystem
Additive Manufacturing	Fused filament fabrication (FFF) of polylactic acid (PLA) to construct frame
Vicon Tracker	Motion capture software for live position and attitude tracking
Mission Planner	Flight control software to communicate with flight controller
MATLAB	Receive and store data from Vicon Tracker for motion capture analysis
RCBenchmark	Software for testing thrust of motors

These are the methods used to carry out tasks for the various subsystems comprising the MAV. The frame was constructed using FFF of PLA. The instrument we used was a Creality 3D printer from Professor Cowlagi's lab. The Vicon Tracker is software companion to Vicon Motion Capture systems and is used as a control station for it. Our use of it was to make the MAV an object within the program's database that the system would recognize and track the position and altitude of. Mission Planner and QGroundControl are also methods that we used. They were the control stations for the actual MAV and were used to control it though a PC. They were also used for the calibration of the flight controller as well as the electronic speed controller tuning. MATLAB was used to process the captured data from the Vicon Motion Capture system and produce quantitative results.

1.6.1 ODROID Setup:

To get the ODROID functional, an SD card with an operating system has to be inserted into it. The ODROID reads the contents of the SD card and boots accordingly. Inserting an operating system on an SD card is called flashing the SD card. Our choices of operating systems to flash were either an Android based system or a Linux based system. We chose the Linux based system as it is the system recommended by HardKernel, the company that sells the ODROID. The HardKernel website contains links to images of different versions of the Linux operating system. An image is a compressed file that captures an "*image*" of an operating system and can be used to move data between devices. Our initial thought was to use the most recent version, Ubuntu 18.04, however our XU4 could not handle the processing associated with that version and would frequently crash. We chose a version that was one generation earlier, Ubuntu 16.04, and these problems stopped.

The image must first be downloaded from the HardKernel website and then decompressed using either Windows extract or through software like 7-Zip (17). Once the image has been decompressed, it is ready for use. The SD card can be inserted into the computer containing the contents of the image and then flashed with them using another software like Balena Etcher. Once Balena Etcher is done flashing the SD card, it should be ejected from the computer and inserted into the ODROID. The operator must make sure that the boot switch for the ODROID is set to MicroSD and not eMMC. Figure 2 below shows the layout of the ports and pins for the XU4. The boot mode selector is on the bottom left of the board. Once the SD card has been inserted, a keyboard and a mouse must also be plugged into the USB ports of the

ODROID. They will allow the user to use the graphical interface of the Linux OS. Initial setup of the Linux OS is things such as device name, username, user password, time zone and Wi-Fi connections. These must be completed to the user's preference.

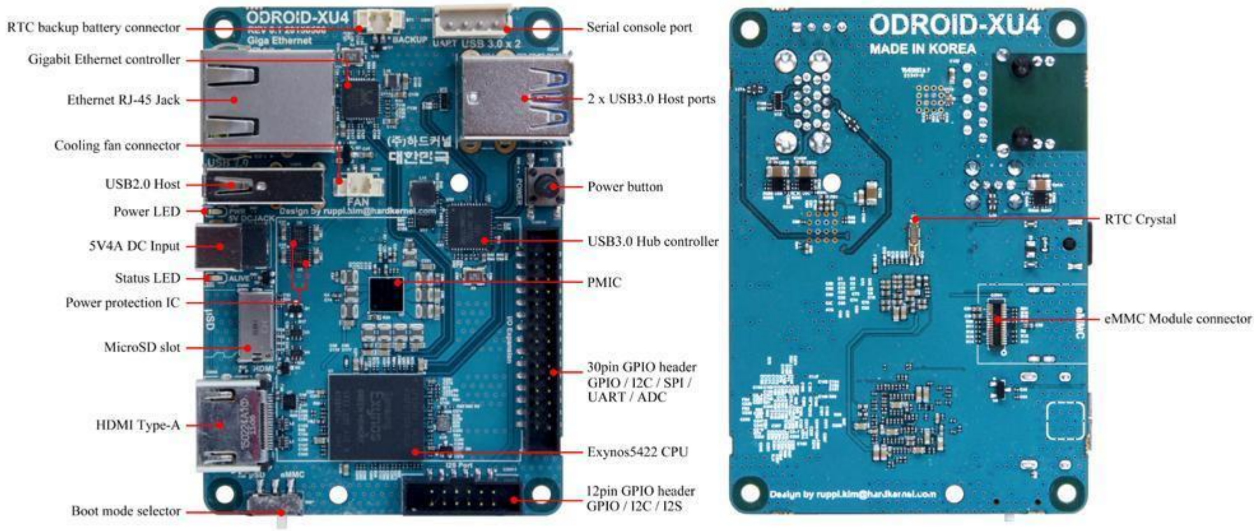


Figure 2: ODROID XU4 Component Layout

1.6.2 Pixhawk 4 Mini Setup:

Setting up the PIXHAWK 4 MINI requires downloading a ground station software on a ground control station. The ground control station for the project was a desktop running Windows. The two most popular pieces of ground station software are QGroundControl and Mission Planner. The PIXHAWK 4 MINI comes with a USB-A to USB – Micro B cable. The USB – A plug is to be connected to a port on the computer and the USB – Micro B plug is to be connected to the PIXHAWK 4 MINI. Once the two are connected, the ground station software should be launched.

For Mission Planner:

If using Mission Planner, the connection to the PIXHAWK 4 MINI can be established in the top right corner of the software as shown in Figure 3. There are two drop-down fields in the top right corner. The left field indicates the port of the computer the Pixhawk is connecting to (COMX), and the right field displays the baud rate. The baud rate is the frequency at which the communication between the two occurs. The user should use Device Manager (Windows Software) to determine the port to which the PIXHAWK 4 MINI has been connected and then select the same port in Mission Planner. Alternatively, simply choosing the AUTO option will select the software-determined port and baud rate to carry out the communication. When using the AUTO feature, keep in mind that the chosen baud rate is the minimum required baud rate and that higher baud rates will speed up communication. The baud rate should be set to 921600.

Flashing firmware OS onto the PIXHAWK 4 MINI is a simple task with Mission Planner. Click the SETUP tab, pictured with a gear, on the top left and then click on Install Firmware. A display pops up showing the various types of vehicles that can be configured (Figure 4). Select the type of remote-controlled craft being used. If you wish to change any part of the firmware, select "All Options". There are a few drop down fields from which the version type, platform and version can be chosen as desired. The difference between version type and version is that version type describes whether the version being flashed is “Stable”, “Beta”, or “Dev”. “Stable” versions are the most-used as they are mostly bug-free and have dedicated forums that answer most trouble-shooting questions users may have. “Beta” versions are usually new versions which have bugs that are being worked on the developers. They also have greater capabilities than “Stable” versions in terms of available features. “Dev” versions are the versions where the developers of the software are actively working and creating new features. They are to be used only by those well versed in RC software.

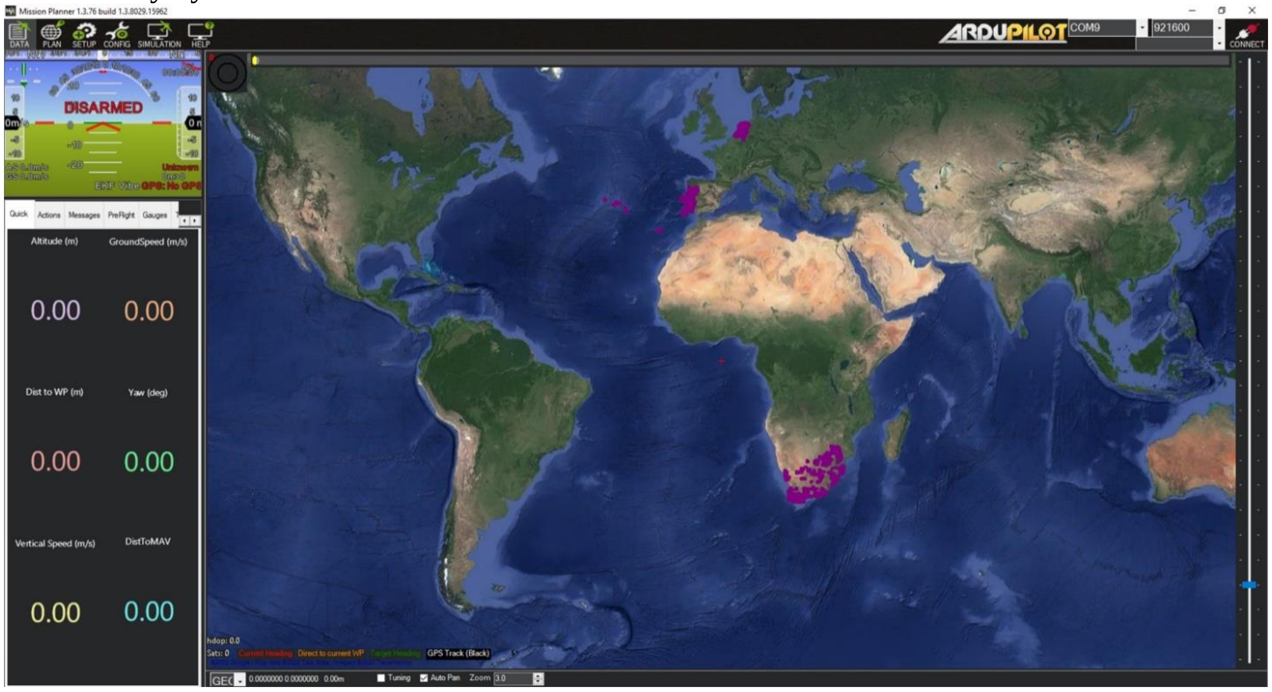


Figure 3: Mission Planner Main View (Mission Planner)

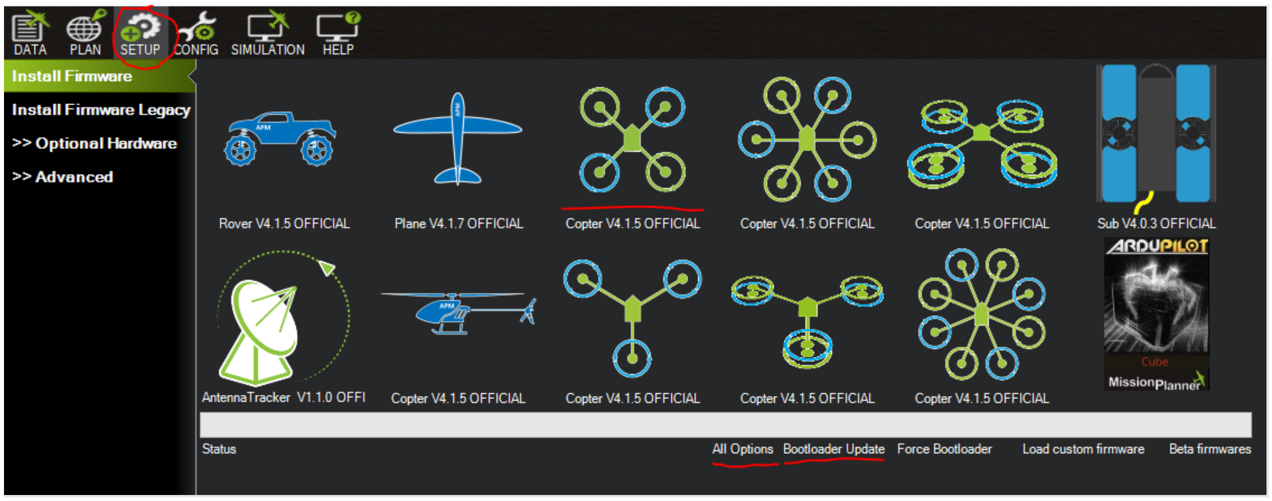


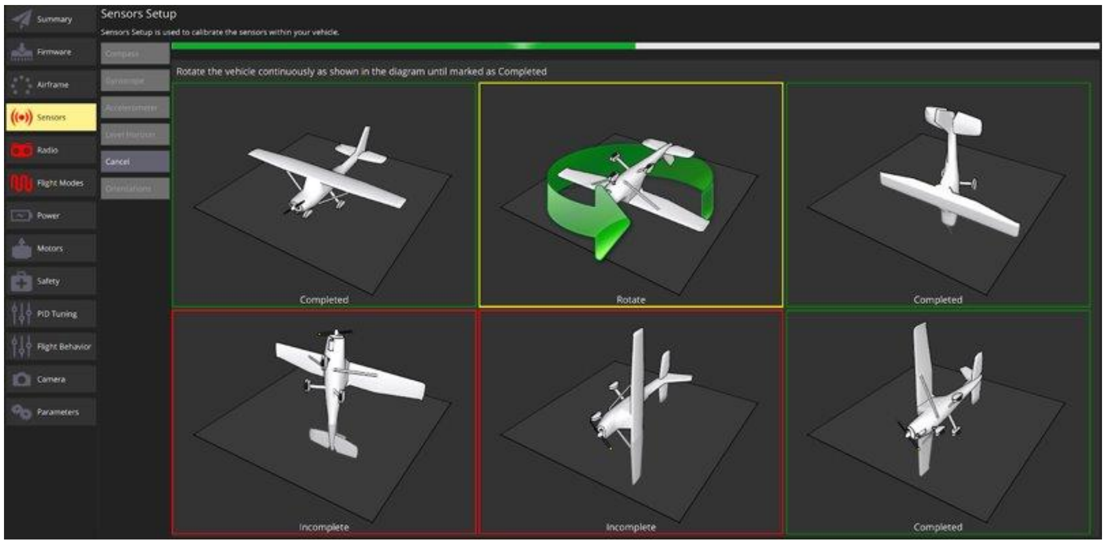
Figure 4: Mission Planner Vehicle Configuration Menu (Mission Planer)

For QGroundControl:

Plug the Micro USB B end into the PIXHAWK 4 MINI and the USB B end into the computer and launch QGroundCotrol. The software should detect the PIXHAWK 4 MINI and change status in the top left from Not Connected to Connected. Click the second icon of the top left bar of the gears. A small screen title “Firmware Setup” should be showing on the right side of the screen. Choose either PIXHAWK 4 MINI Pilot Flight Stack or ArduPilot Flight Stack.

1.6.3 PIXHAWK 4 MINI Calibration:

Before the PIXHAWK 4 MINI can successfully be used with a UAV frame, its internal sensors such as the accelerometers and rate gyros must be calibrated. Mission Planner simplifies the process. The PIXHAWK 4 MINI was connected to a computer running the software and the setup tab was navigated to. From there, the calibration process began. It involved placing the PIXHAWK 4 MINI on a flat surface and rotating about the axis perpendicular to the side it was resting on. This was completed for all 6 faces of the PIXHAWK 4 MINI. The rotation is shown below in Figure 5.


Sensors Setup

Sensors Setup is used to calibrate the sensors within your vehicle.

Rotate the vehicle continuously as shown in the diagram until marked as Completed

Figure 5: PIXHAWK 4 MINI sensor calibration.

1.6.4 Autonomous software setup instructions:

On the XU4, open a new terminal and begin the installation of mavROS. mavROS is a ground control station software that allows for the SBC to act as a ground station and control the Pixhawk autonomously. mavROS makes use of MAVLink, a messaging protocol for UAV as well as onboard component communication. The mavROS installation procedure is clearly outlined on the MAVLINK GitHub page. The GitHub page is referenced (18).

1.6.5 Serial communication between XU4 and PIXHAWK 4 MINI:

Connecting the XU4 and the PIXHAWK 4 MINI required male to female wires. The male ends of the wires plugged into the PIXHAWK 4 MINI, and the female ends to the XU4. The PIXHAWK 4 MINI port was the UART1 port, shown in Figure 6 below. The connection on the XU4 end was the expansion connector board. Figure 9 shows the pin configuration for the XU4. The relevant pinouts from the PIXHAWK 4 MINI, initially, to get a start on serial communication are the TX and RX ones. The TX (out) from the PIXHAWK 4 MINI connects to the RX (in) on the XU4’s expansion board and the RX (in) from the PIXHAWK 4 MINI connects to the TX (out) of the XU4. Figure 7 below is a diagram intended to simplify where the first pin on each of the expansion boards is. There is also a set of wires responsible for the powering of the ODROID, VCC and GND, but these can be connected once communication is initiated.

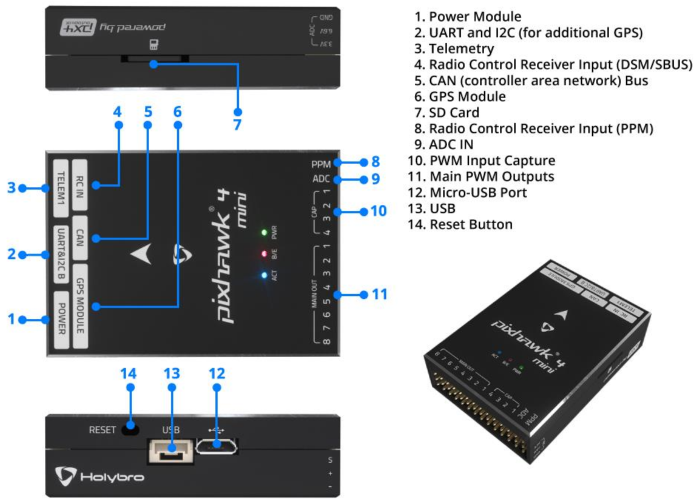


Figure 6: The PIXHAWK 4 MINI port layout.

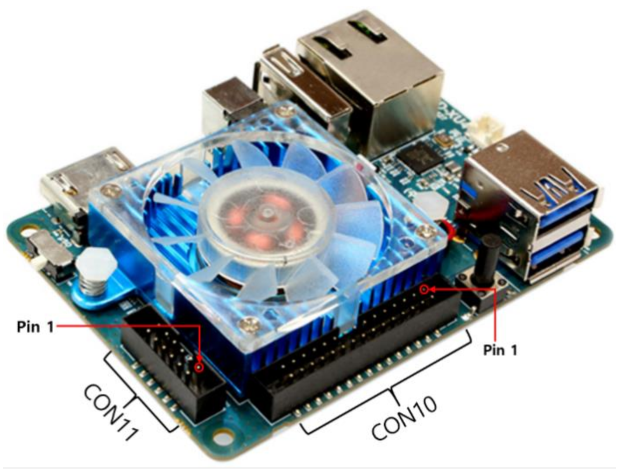
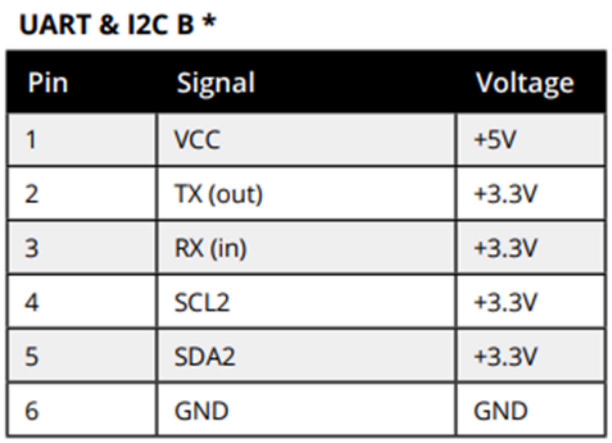


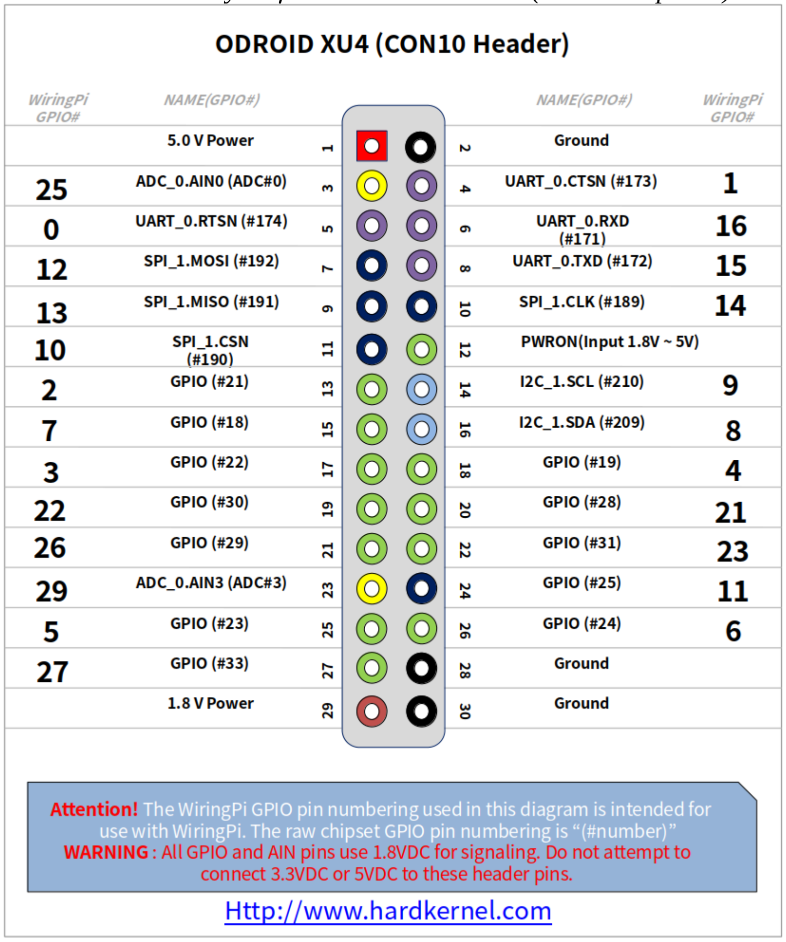
Figure 7:ODROID-XU4 Expansion Pin Locations (www.hardkernel.com)



UART & I2C B *

Pin	Signal	Voltage
1	VCC	+5V
2	TX (out)	+3.3V
3	RX (in)	+3.3V
4	SCL2	+3.3V
5	SDA2	+3.3V
6	GND	GND

Figure 8: UART1 Pin Layout for PIXHAWK 4 MINI. (www.docs.px4.io)



ODROID XU4 (CON10 Header)

WiringPi GPIO#	NAME(GPIO#)			NAME(GPIO#)	WiringPi GPIO#
	5.0 V Power	1	2	Ground	
25	ADC_0.AIN0 (ADC#0)	3	4	UART_0.CTSN (#173)	1
0	UART_0.RTSN (#174)	5	6	UART_0.RXD (#171)	16
12	SPI_1.MOSI (#192)	7	8	UART_0.TXD (#172)	15
13	SPI_1.MISO (#191)	9	10	SPI_1.CLK (#189)	14
10	SPI_1.CSN (#190)	11	12	PWRON(Input 1.8V ~ 5V)	
2	GPIO (#21)	13	14	I2C_1.SCL (#210)	9
7	GPIO (#18)	15	16	I2C_1.SDA (#209)	8
3	GPIO (#22)	17	18	GPIO (#19)	4
22	GPIO (#30)	19	20	GPIO (#28)	21
26	GPIO (#29)	21	22	GPIO (#31)	23
29	ADC_0.AIN3 (ADC#3)	23	24	GPIO (#25)	11
5	GPIO (#23)	25	26	GPIO (#24)	6
27	GPIO (#33)	27	28	Ground	
	1.8 V Power	29	30	Ground	

Attention! The WiringPi GPIO pin numbering used in this diagram is intended for use with WiringPi. The raw chipset GPIO pin numbering is "(#number)"
WARNING : All GPIO and AIN pins use 1.8VDC for signaling. Do not attempt to connect 3.3VDC or 5VDC to these header pins.

Http://www.hardkernel.com

Figure 9: ODROID XU-4 Pin Wiring Diagram (www.hardkernel.com)

1.6.6 Manufacturing Processes:

In this project, fused filament fabrication (FFF) of polylactic acid (PLA) plastic was used to create initial models of the frame of the MAV. Cura, an open-source 3D printing software, was used to slice 3D models into printable instruction files, and all parts were printed on a Creality CR-10S Pro V2 3D printer. While PLA was not an appropriate material for the final use of the MAV due to poor stiffness properties, it allowed rapid prototyping of different frame designs and a “sanity-check” that the components were sized correctly before sending files to be manufactured in carbon fiber elsewhere.

- Prototype-
    - Creality CR-10S Pro V2 3D printer
    - Construction of prototype by connecting wires
        - Soldering electrical components

1.7 Broader Impacts

The MAV that resulted from this project was designed for indoor flight. The quadrotor will be controlled using a motion capture system that feeds a ground control station the position of the quadrotor in real time. Unlike MAVs that are designed to fly outdoors there are fewer mainstream applications, such as using an MAV to run quality control tests on large construction sites that would typically be too difficult for people to test themselves. An example of this is described in a study by Jun Wang which discusses integrating building information modeling (BIM) and Light Detection and Ranging (LiDAR) for real-time construction quality control. Another more common outdoor application is the use of UAVs to deliver packages instead of delivery trucks, which is still in the developmental phase. It is clear to see how these applications of MAV can be useful and beneficial to society, however the type of UAV we have created is different in several ways. Primarily, our MAV is guided by a motion capture system in lieu of GPS.

The overall wiring and frame of our UAV would require minimal changes to be suited to fly outdoors, the most significant addition would be connecting a commercially available GPS module to the Pixhawk 4 mini. A GPS is a reliable navigation tool for many applications, especially those which require a large area of operation; however, it is far less precise than the Vicon motion capture system that is being used with this MAV in a small capture volume. This is an important factor when discussing how our project differs from existing MAVs. Using Vicon motion capture we are able to have real time data and far more accurate than GPS (accurate to tenths of millimeters) the Euler angles, coordinates, velocity and much more send to the ground control station which works with the Pixhawk 4 mini to control the MAV. This results in us being able to program the MAV to perform more precise maneuvers than is possible with GPS. An example of another group of people who have worked with Vicon motion capture to achieve this result is Mark Cutler and his team at MIT, who successfully navigated their MAV through obstacles and performed aggressive maneuvers. The MIT team could not have flown their UAV properly without the more precise data provided by the motion capture system. The next step for society is to master this technology and determine practical applications in the world, people did not just know to use outdoor UAVs on construction sites overnight. To master this technology, we need to make it easier to teach and easier for students to have access to and learn how to work these systems. That is why we are creating a low-cost quadrotor MAV in this project alongside learning how the Vicon hardware and software works and writing a manual for future students to follow. As stated above, the goal of this project is to design an MAV that will utilize a Vicon motion capture system as its navigational means and carry a sensory payload. This MAV needs to have easily accessible parts and instructions so that future students can be given the parts list and instructions for the MAV and Vicon and continue learning and researching this topic without the need to learn it from scratch like we have in this project. The broader impacts of this project are yet to be known because we are laying the building blocks for future students to use this technology as a tool to use instead of a subject to learn. Most modern aircraft are made with autonomous features or are completely autonomous, so learning how to process and deal with the large amounts of data that the Vicon motion capture system provides extends well past quadrotor MAV.

2 System Design

Table 5: Component List

Component	Final Selection
Flat Frame Components	3D Printed PLA
Frame Standoffs	35 mm M3 Female Threaded Hex Standoffs
Mechanical Fasteners	M3 Machine Screws
Propellors	5 inch diameter 3 inch pitch 2 blade propellers
Microcontroller	ODROID-XU4 Single Board Controller
Flight Controller	Pixhawk 4 Mini
Flight Controller Power Module	Pixhawk Power Module 5.3V BEC XT60
Battery	HOOVO 4500 mAh 3s (11.3 V) Lithium Polymer Battery
Power Distribution Module	Matek PDB-XT60
Motors	T-MOTOR MN2206 KV2000 Brushless Electric Motors
Electronic Speed Controllers (ESCs)	ARRIS Swift Series BLHeli 20A 2-4S BEC 5V/1A Brushless ESC
Motion Capture Software	Vicon Tracker 3.0
Ground Station Software	Mission Planner
Microcontroller Software	Ubuntu 16.04
Pixhawk 4 Mini Firmware	Latest Firmware available from Mission Planner

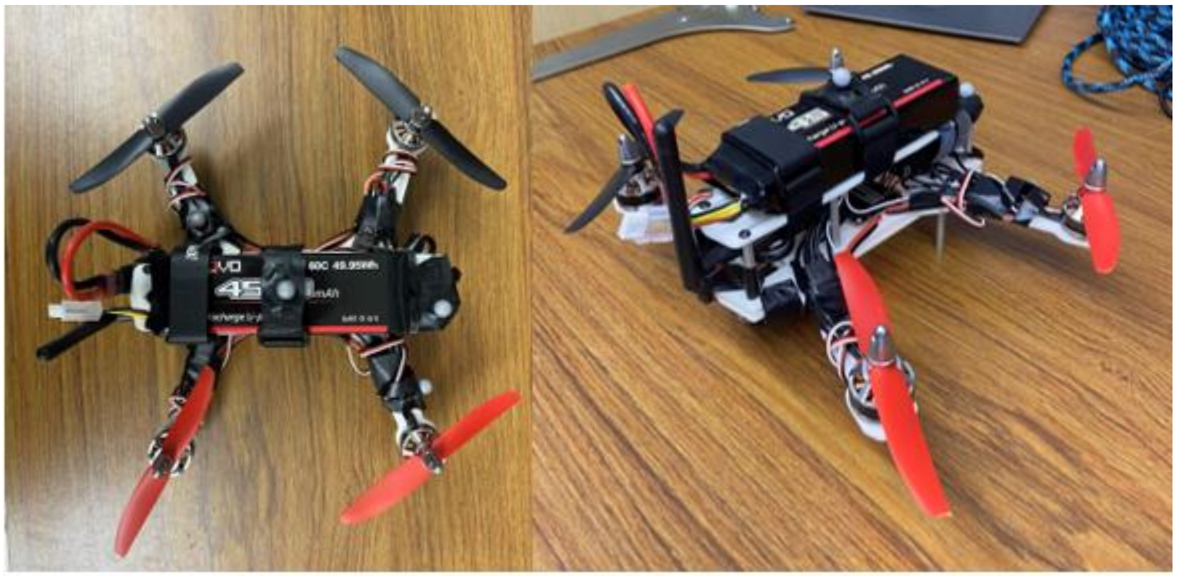


Figure 10: Fully Assembled MAV (without ODROID microcontroller)

The UAS is made up of the UAV, the Vicon motion capture system, and the ground control unit. The frame of the UAV is made of three decks, holding the XU4, the Pixhawk 4 mini, the power distribution board, and the sensor payload (Figure 10).

2.1 Flat Frame Components

Before deciding on the specifications of what the frame of the quadrotor will look like, we made a list of all the components. Table 5 above shows the list of our initial components. The list in the figure above also includes a column with the dimensions of all the components, to make sure the frame we construct will be able to enclose them all. The frame is based on a design from Holybro.

While researching existing quadrotor frame designs, we found many to have their builds made from carbon fiber. Rather than using carbon fiber for the whole MAV, we decided to use the 3D printed Polylactic Acid (PLA) for the main frame and carbon fiber for the arms. This decision was made since 3D printing material is much cheaper than carbon fiber. Also using carbon fiber for the arms will allow them to be much stronger and stiffer and support the motors torque and vibrations.

To construct the frame and attach all components to it, M3 machine screws and nuts were used. To provide vertical structure and function as landing gear, 35mm female-threaded standoffs were used. Initially, the MAV was proposed to land on the flat bottom of the frame without landing gear. However, upon construction, part of the power module extended beneath the frame, requiring landing gear.

The addition of motor guards to the MAV was discussed to protect anything that may be near the MAV during flight from the spinning propellors. However, as the MAV was never constructed past the initial prototype stage, including flexible PLA instead carbon fiber motor arms the motor guards were not included. With further iteration, motor guards would likely be added as flight patterns became more elaborate.

2.2 Propellers

The propellers that we selected for our project are 5 inches in length with a 3-inch pitch (5030) (Figure 11). Our propellers were decided by several factors including compatibility with both the frame and motors. Diameter and pitch are the two main factors determining propellor performance. We decided to determine the diameter first, as the diameter of the propellers is restricted by the overall frame geometry, while the pitch is not restricted by other components. The frame design created by Holybro Holybro (19) that we modified for this project was originally designed for five-inch propellers, and with the expected mass and dimensions of our components retaining this size was appropriate. For the pitch of the propellers, we referenced the motor specification sheet published by T-MOTOR. For an 11.1V battery and five-inch propellers, only data for a three-inch pitch is available. With the thrust generated at the throttle ranges provided, 5030 propellors were decided on for compatibility with frame size, motors, and thrust requirements.

Figure 11: Example of 5030 2 Blade Propellers

2.3 Microcontroller

A microcontroller is a small computer. For UAVs, microcontrollers allow autonomous control. To give our MAV autonomous capabilities, we had to choose between using an NVIDIA Jetson Nano or a HardKernel ODROID XU4 as the microcontroller. The microcontroller that we decided on using was the ODROID XU4. There were a few factors that made the XU4 a better choice for the MAV overall. It is significantly lighter than the Jetson Nano. The XU4 weighs 60 grams, while the Nano weighs 210 grams, nearly four times as much. With a mass of 210 grams, about half of the weight of the entire MAV would be just the microcontroller, which seemed poor designing strategy. Additionally, the Jetson Nano requires a heatsink to operate. NVIDIA recommends that this heat sink not be removed under any circumstances as it could cause the Nano to overheat. With the heat sink, the only way to accommodate the Jetson on the MAV would be on the topmost deck. In the event of a crash, this would likely significantly damage the Jetson, which also seemed poor design strategy. In terms of technical specifications, the Jetson offers much greater processing power than the XU4 due to its combination of quad-core CPU and 128-core GPU. The XU4 has only an octa-core CPU. The Jetson's RAM is 4GB and the XU4's is 2GB. For sensory data capture and transmission to an external device, the processing power required is not extremely high. Additionally, since navigation was going to be done using the Vicon system, there wasn't a need to run a Kalman Filter or other such taxing processes. Keeping the above factors, as well as the costs in mind, it made much more sense to use the XU4 rather than the Jetson. Since the XU4 is still a little wider than our design for the top of our frame, we have designed an attachment piece for the XU4 to help mount the minicomputer to the aircraft. For any person replicating our model, they will have the ability to exchange the XU4 for a microprocessor of their choice by simply reprinting the attachment piece that is designed to fit on top of our aircraft but move the location of the attachment holes to match those of their processor.

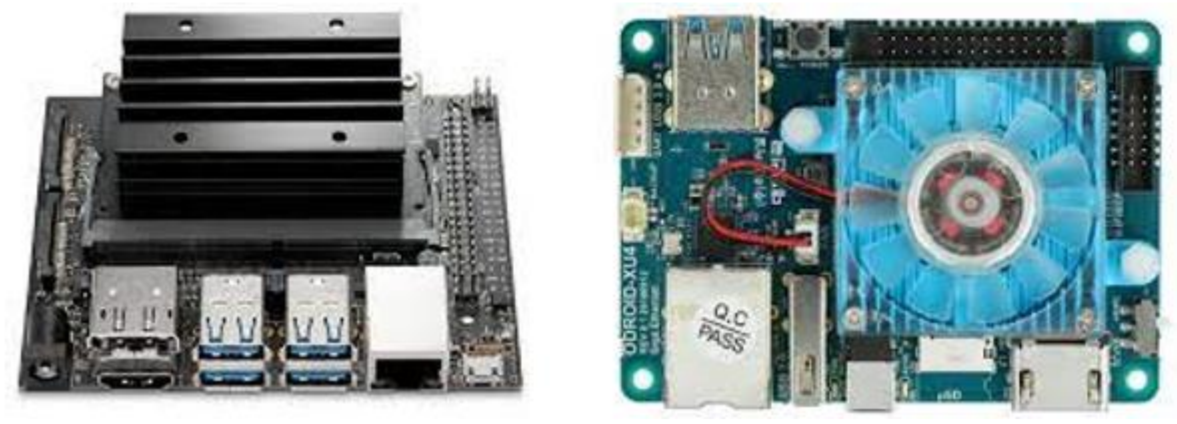


Figure 12: NVIDIA Jetson (left) and ODROID XU-4 (right)

2.4 Flight Controller

A Pixhawk 4 Mini flight controller was chosen primarily for open-source, platform neutral software options, versatility, and physical size (Figure 13). The Pixhawk 4 Mini runs PIXHAWK 4 MINI autopilot software, an open-source, industry standard ecosystem for unmanned vehicles (20). PIXHAWK 4 MINI allows for extensive customization, allowing us to modify the autopilot for this specific project. The Pixhawk 4 Mini was chosen over the Pixhawk 4 solely due to physical size. The Mini version possesses the exact same processing capability, but in smaller dimensions and with fewer additional data ports, which were not necessary for this application. With the small size of the frame, space saving was a priority.

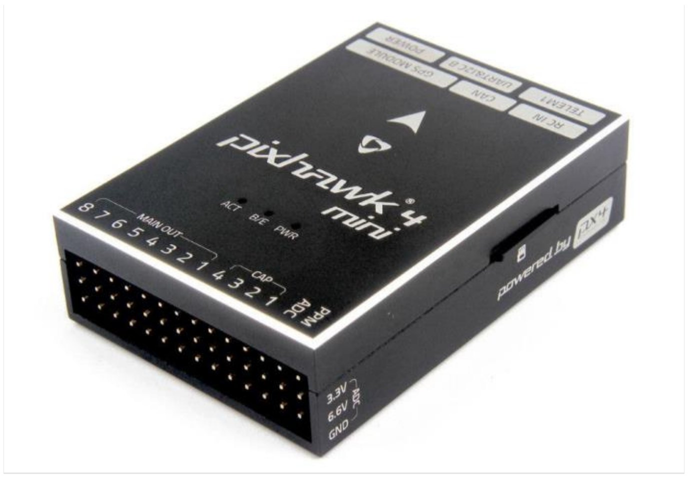


Figure 13: Pixhawk 4 Mini

2.5 Battery

For the final design a 4500 mAh 3S (11.1 V) lithium-ion polymer (LiPo) battery manufactured by HOOVO was chosen (Figure 14). This battery was chosen out of other batteries with the same electrical properties due to mechanical dimensions and minimal weight. The dimensions of this battery better matched the dimensions of the frame to allows for better component positioning. The electrical properties of the battery were determined as part of the iterative electrical system design process described in Section 3. To support the proper voltage of the motors and provide a flight time of at least 15 minutes running all proposed components, 11.1 V and 4500 mAh were required. Before deciding on the 4500 mAh battery by HOOVO, our initial consideration was the Turnigy Bolt 1800 mAh 3S (11.1 V) LiHV battery. The Turnigy Bolt battery had less milli ampere-hour but was much lighter than the HOOVO battery. We eventually found and decided on the HOOVO battery since the weight of our aircraft increased and the motors we choose required more power to operate.

Figure 14: 4500 mAh 3S (11.1V) HOOVO LiPo Battery

2.6 Power Distribution Module

Our project team decided on the Pixhawk power module 5.3V (Figure 15). This component gives us the ability to power our flight controller while also reporting the battery voltage and current data to the controller to log. The connectors on the module output a minimum of about 5.3V at a maximum of 2.25 Amps and can be used up to a maximum of 28V at 90 amps. The connectors are called XT60, which are the same as the PDB below. This decision was made because it was created by the same company as our flight controller, Pixhawk, and its specs included everything that we needed. It was also a highly recommended product by many reviewers custom building quadrotors.

Figure 15: Pixhawk 5.3V Power Module

2.7 Power Distribution Board

Our project team chose the LinsyRC Power Distribution Board (PDB) (Figure 16). This PDB comes with an XT60 connector which connects directly to the power module we decided on. It also provides our UAV with up to six positive and negative ESC outputs. When using four ESCs it relays 25 amps through each one with a peak current of 30 amps. The decision to purchase this PDB over others was relatively simple. It provided us with a direct connection to the power module we selected as mentioned earlier and can handle a 3S LiPo battery. The board also provides an extra 5V and 12V BEC output which can be used for any sensors that are added to the aircraft.

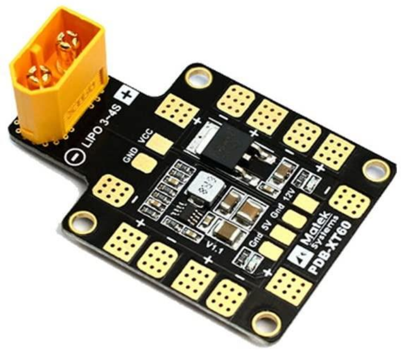
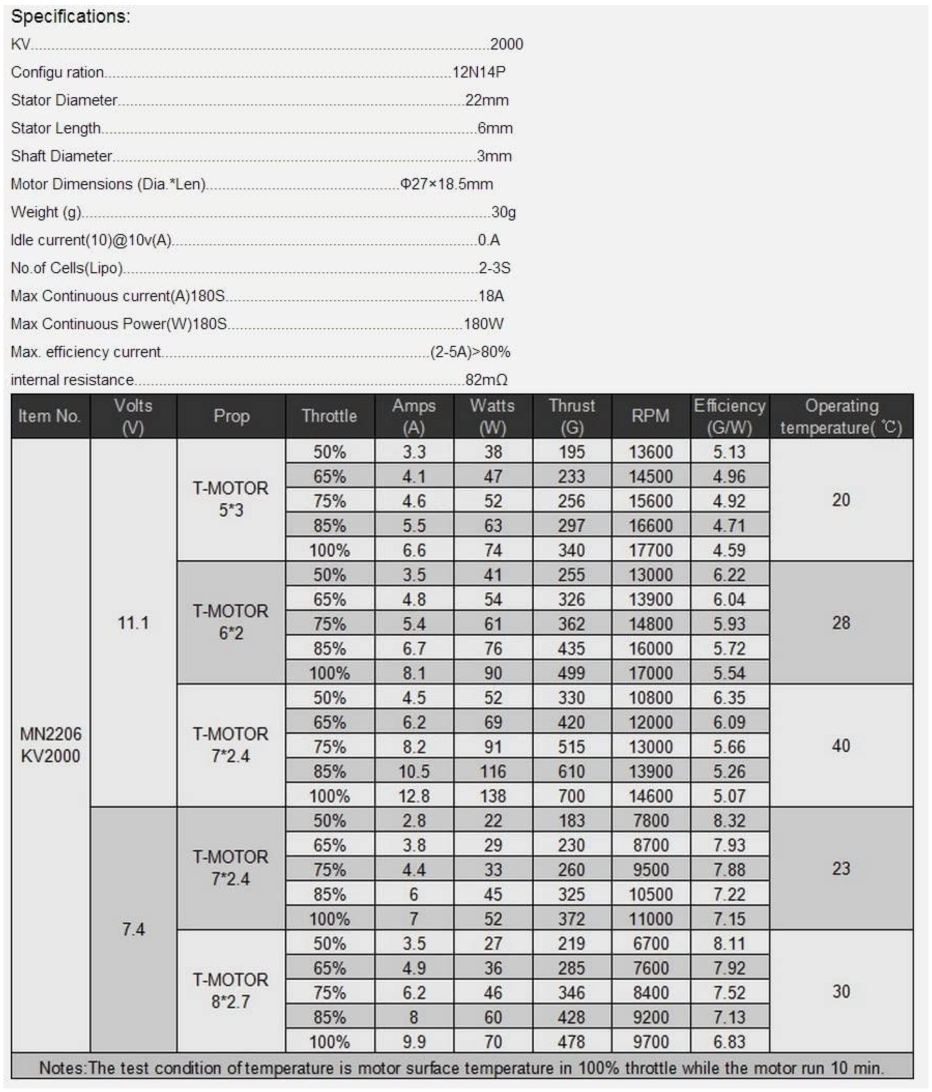


Figure 16: LinsyRC Power Distribution Board

2.8 **Motors**

The quadrotor motor that was decided on was the T-Motor MN2206 KV2000 (Figure 17). This motor is a high-performance brushless electric motor specifically made for multi-rotor aircrafts. A picture of the motor and its specifications are shown in the figures below.

Figure 17: T-Motor MN2206 KV2000 Brushless Motor



Specifications:

KV	2000
Configu ration	12N14P
Stator Diameter	22mm
Stator Length	6mm
Shaft Diameter	3mm
Motor Dimensions (Dia.*Len)	Φ27×18.5mm
Weight (g)	30g
Idle current(10)@10v(A)	0.A
No.of Cells(Lipo)	2-3S
Max Continuous current(A)180S	18A
Max Continuous Power(W)180S	180W
Max. efficiency current	(2-5A)>80%
internal resistance	82mΩ

Item No.	Volts (V)	Prop	Throttle	Amps (A)	Watts (W)	Thrust (G)	RPM	Efficiency (G/W)	Operating temperature(℃)
MN2206 KV2000	11.1	T-MOTOR 5*3	50%	3.3	38	195	13600	5.13	20
			65%	4.1	47	233	14500	4.96	
			75%	4.6	52	256	15600	4.92	
			85%	5.5	63	297	16600	4.71	
			100%	6.6	74	340	17700	4.59	
		T-MOTOR 6*2	50%	3.5	41	255	13000	6.22	28
			65%	4.8	54	326	13900	6.04	
			75%	5.4	61	362	14800	5.93	
			85%	6.7	76	435	16000	5.72	
			100%	8.1	90	499	17000	5.54	
		T-MOTOR 7*2.4	50%	4.5	52	330	10800	6.35	40
			65%	6.2	69	420	12000	6.09	
			75%	8.2	91	515	13000	5.66	
			85%	10.5	116	610	13900	5.26	
			100%	12.8	138	700	14600	5.07	
	7.4	T-MOTOR 7*2.4	50%	2.8	22	183	7800	8.32	23
			65%	3.8	29	230	8700	7.93	
			75%	4.4	33	260	9500	7.88	
			85%	6	45	325	10500	7.22	
			100%	7	52	372	11000	7.15	
		T-MOTOR 8*2.7	50%	3.5	27	219	6700	8.11	30
			65%	4.9	36	285	7600	7.92	
			75%	6.2	46	346	8400	7.52	
			85%	8	60	428	9200	7.13	
			100%	9.9	70	478	9700	6.83	

Notes:The test condition of temperature is motor surface temperature in 100% throttle while the motor run 10 min.

Figure 18: Manufacturer's Specifications for T-Motor MN2206 KV2000

This motor when using a 5in diameter by 3in pitch propellor provides a thrust of 195G at 50% throttle (Figure 18). The thrust provided at 50% throttle must be approximately equal to the weight of the UAV. This is called the hovering thrust, which will allow the motors to ascend and descend the UAV by rotating above or below 50% thrust. This motor was also decided on because it came from a highly reviewed company and was available on amazon providing quick and easy shipping. The price was also matched up against other motors before the decision was made. The price of these MN2206 motors matched the quality of motor our team was looking for. Table 6 below shows some of the motors that we compared before deciding on the T-Motor MN2206 KV2000.

Table 6: Motor Comparison Chart

Name	Mass (g)	Power Consumption(V)	KV	RPM	Thrust	Idle Current
EMAXMT1806	18	2-3S	2280	Depends on Battery	Depends on Prop	Depends on Prop
T - MotorF1303	6.1	2-3S	5000	22735 @ 50%throttle	66.24g @ 50%throttle	0.32 A @ 4V
T - MotorF1204	6.9	2-3S	5000 /6500	19881 @ 50%throttle	58.85g @ 50%throttle	0.7A @ 10V
T - MotorF1103	5.3	2-3S	8000 /11000	29663 @ 50%throttle	48.3g @ 50%throttle	0.58A @ 3V
T-MotorF1404	7.75	3-4S	3800	22759 @ 50%throttle	158g @ 50%throttle	0.5A @ 10V

Need to change the table, not all the correct motors we compared

2.9 Electronic Speed Controllers (ESCs)

The ESCs we decided on were the ARRIS Swift 20A 2-4S BLHeli Brushless ESCs (Figure 19). They were designed to for a 20A continuous current and up to 30A burst current. Batteries of 2-4S (7.4-14.8 V) are supported. These were decided based on the amount of current and voltage our motors required as well as the voltage output by the battery.

Figure 19: ARRIS Swift 20A 2-4S BLHeli Brushless ESC

2.10 Motion Capture Software

Vicon Tracker is the software developed by Vicon for engineering applications. At the time of this project Vicon Tracker 3.9 was the most recent version. Designed to maximize accuracy and precision, Tracker is capable of dynamic accuracy of 0.017 mm and allows for real-time data transfer and modeling via Mathworks Simulink (21). With Tracker designed specifically for use with the Vicon Vero cameras in engineering applications, there were no comparable software alternatives.

2.11 Ground Station Software

Mission Planner is a ground control station software for remote-controlled vehicles. It can be used to simply configure the RC vehicles or to actively control them if they are autonomous. QGroundControl is another ground control station software and offers many of the same features as Mission Planner, however not as many. QGroundControl has a simplified user interface compared to Mission Planner and streamlines many of the initial procedures to get the MAV armed. The reason the team opted for Mission Planner over QGroundControl is that the arming procedure using QGroundControl did not work. The software could not recognize the ESCs, which was not an issue that could be solved by the team. Opting to maintain the schedule, we pivoted to Mission Planner.

2.12 Ubuntu 16.04

For the initial flashing of the OS for the ODROID, the team had to choose between using an Android based system or a Linux based one. The team chose a Linux-based OS because of two main reasons. The first being that it was the OS the team was more comfortable using and it has a lot of support online if troubles were encountered. The second is that the Android OS is primarily designed to be used via touchscreens on mobile devices, not utilizing a controller to command a MAV. Linux is a system that has been used for millions of embedded devices across the globe, while Android is a more recent entry in that space. It was simply a question of the amount of existing information pertinent to our task between the two OS, and the answer was Ubuntu.

2.13 Universal Sensor Clamp/Holder

Our project team never officially decided on a sensor clamp or sensors to use on the MAV. With the Odroid XU4 microcontroller we initially thought to include a clamp that could hold multiple different sensors. Some sensors that we had in mind include microphones, cameras, and LIDAR. The universal holder would consist of a wall of holes being held facing the front of the aircraft. These holes can be used to attach a clamp of the user’s choice. This will allow the user to decide exactly what sensor they would like to use on the MAV. Since the XU4 has a solid processing power, there are a wide variety of sensors that the user can choose to attach on the MAV.

3 Design Process and Analysis

3.1 Electrical System Iterations

In Figure 20 below we have our iteration process expressed within a diagram. It shows our thinking and how we arrived at our selected motors, propellers, and battery combination.

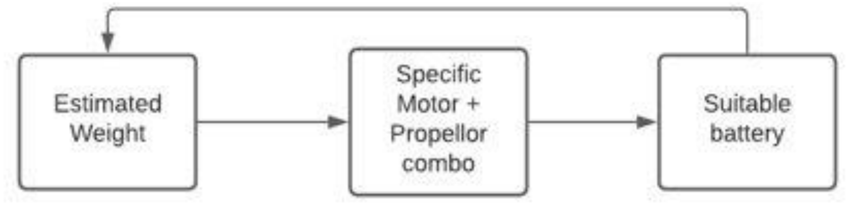


Figure 20: Propulsion System Iterative Process

3.2 SolidWorks

Our design process began by forming a complete parts list. Once our group knew all the components that will be included in the aircraft, we were able to start modeling our frame. Our initial idea was to have a quadrotor that has 3 levels as shown in the figure below. After some calculations and research on quadrotor designs that already exist, we established 2 levels would have enough space to hold all necessary components. From a website named HolyBro (19) we were able to find a schematic and 3D model of a quadrotor that had been tested in flight. Using this model as shown in Figure 21 below, we were able to adjust the mounts and the size of the bases to fit our needs.

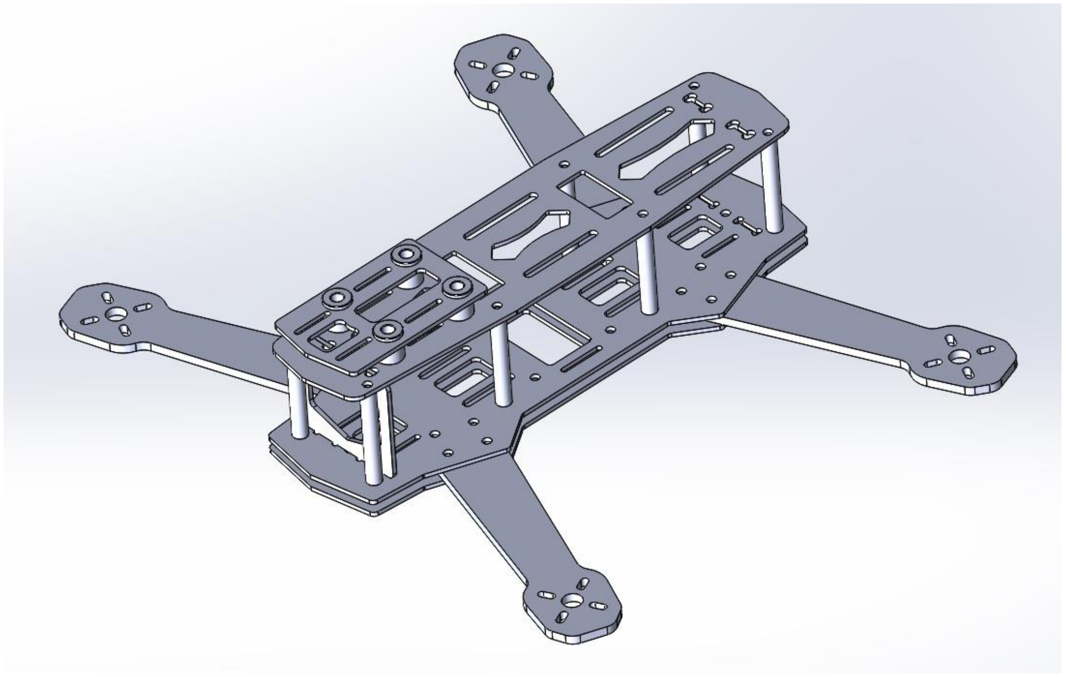


Figure 21: HolyBro Frame Design

We began by modeling our components in SolidWorks in order to make sure that all our components fit onto the designed frame before we printed the 3D model. Using the SolidWorks software made it easy to visual all the parts attached to the frame virtually to make sure enough space was provided for wiring and in between components. Some of the 3D modeled parts were found on a similar website named GrabCad. (22) Once a place was found for all the components on the frame, we began printing starting with the arms (Figure 22).

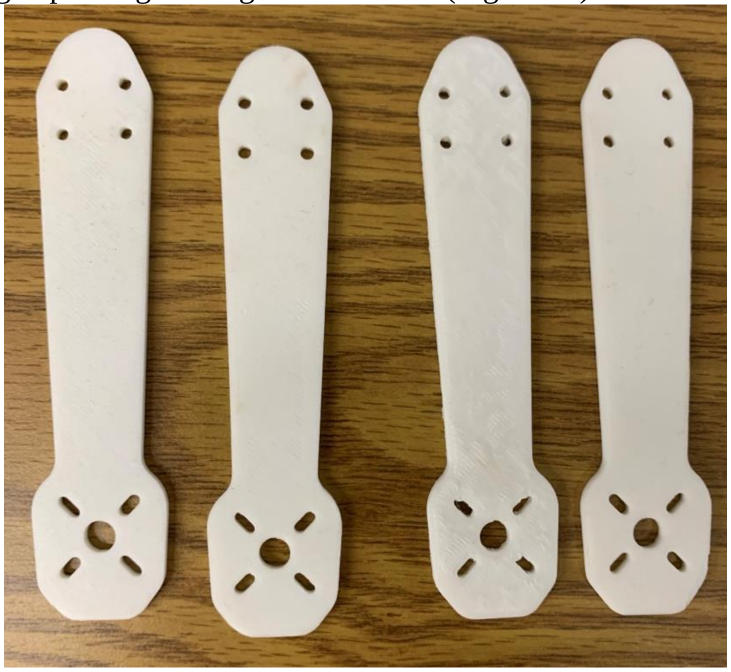


Figure 22: 3D Printed Motor Arms

The first test was to see whether the motors could be connected to the arms properly. This test was our top priority for we needed to have our motors attached to an arm to test it on the thrust stand. We immediately recognized that the 3D printing process caused the holes to shrink. The screws we ordered did not fit through the arms. Therefore, we reprinted the arms after increasing the 3mm diameter holes and slots to 3.2mm. This adjustment was made to all frame parts that required the use of the screws.

Next in the design process came the main base. It was redesigned to fit both the PDB and have a space for the Pixhawk to be strapped in place to maintain its orientation for flight. Two main base plates were printing out to hold the arms together. The arms attached perfectly to the base and did not need to be reprinted. The PDB also fit nicely with the wire for the battery running along the bottom of the drone to the top of the drone where the battery will be held. The frame was a two-floor design where the arms are not directly connected to the main frame as mentioned before. This way in the case of an arm breaking during flight, only a new arm needs to be printed instead of the whole main base. The top plate of the drone was redesigned next to have holes for the battery straps to flow through as well as new holes for the microcontroller holder. An extra platform was designed for the microcontroller to be held perpendicular to the battery. This was done to prevent the battery and microcontroller from sticking too far over the edge. The microcontroller does stick over the sides of the drone above the propellers, which could affect their performance. Figure 23 below is an assembly of all the components planned for our structure of the MAV.

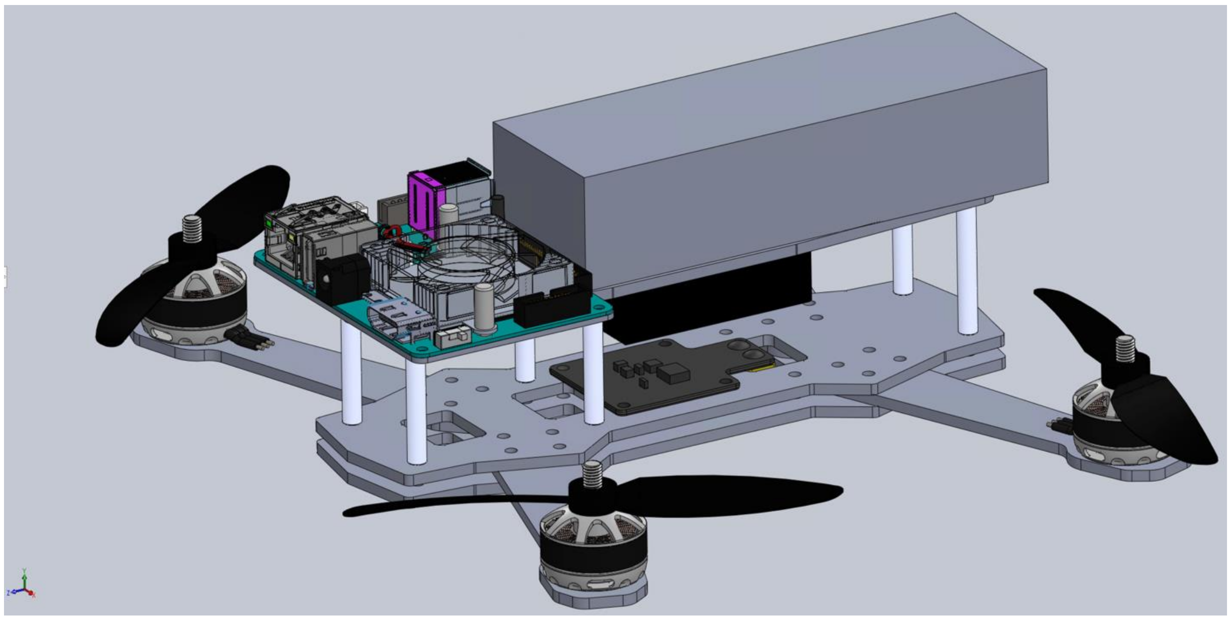


Figure 23: Solidworks CAD Model of MAV

Once all the pieces were 3D printed for the MAV, the assembly began. Standoffs were used to attach the top plate to the base of the MAV. The holes aligned as predicted through SolidWorks. The battery fit well on the top plate, but the extra microcontroller platform has some issues when attached. A piece coming off the bottom of the microcontroller was preventing it from sitting flat on the platform. When all screws were tightened the platform bent in a slight U shape.

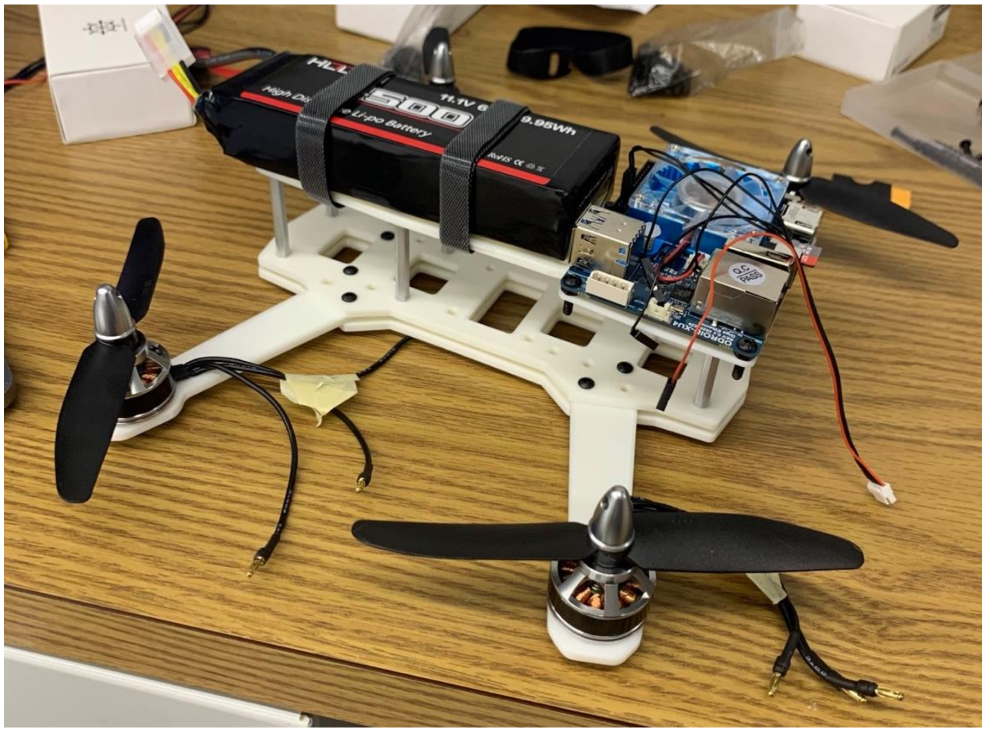


Figure 24: Frame with Motors, Battery, and Microcontroller Attached

Holding the frame and during flight tests it was noticed that the arms are very unstable. They tend to bend during flight. This was why we initially projected to use carbon fiber arms but did not have enough time to order them.

3.3 Thrust Stand Testing using RC Benchmark

The main purpose of conducting these tests of the motors on a thrust stand was to verify that our motors were performing to listed specifications. The most important specification being the thrust generated at half power. Generally, for a quadrotor, the motors should allow the MAV to hover at approximately half thrust (around a 2:1 thrust to weight ratio is widely used in MAV so that the MAV can hover at half thrust) so it is important to make sure the motors provide the necessary thrust. The thrust stand test was the first time that many components were used together, so it served as a check of component compatibility, specifically for the PDB, ESCs, and motors

For this test we collaborated with students from other MQPs who had experience using the thrust stand correctly and taught us how to use it, specifically Chris Davenport taught us how to work the hardware and software properly in a timely manner. The steps we took in setting up the hardware of the thrust stand are as follows:

1. Plug the three signal wires of the esc into the back three holes on the top of the stand and use the screws on the top to tighten them, such as in Figure 25 below.
2. The two power wires of the esc are plugged into the power holes on the board, seen in Figure 25 beneath the three esc signal wires as two gold circles which can be accessed from the bottom and screwed tightly from the side.
3. The wire on the esc which receives signal from the flight controller (the RC benchmark software in this test) is the yellow and black wire seen in the pictures and is plugged into the three-prong set labeled esc on the left side of the board.
4. There is a micro-USB cable that can be seen underneath the three signal esc wires that are plugged into the inner side of the board, the other end of that wire is a USB cable that plugs into the computer running the RC Benchmark software.

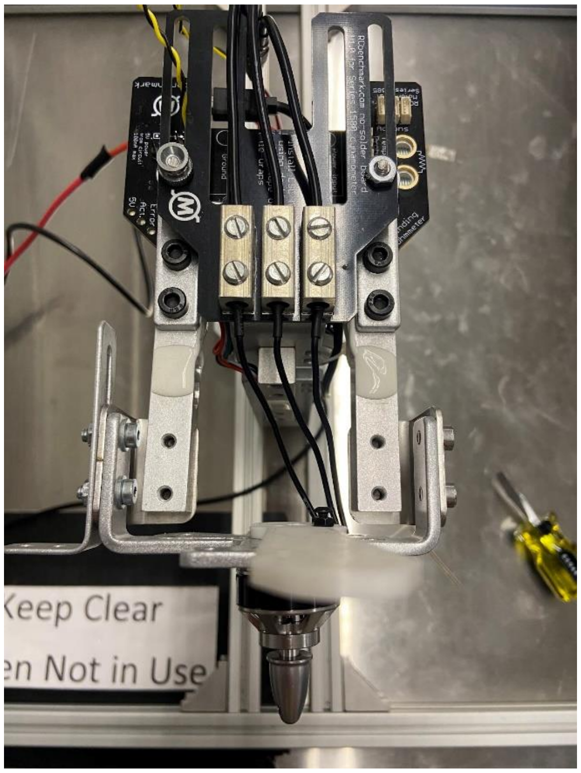


Figure 25: Thrust Stand with Motor (bottom) Attached to ESC

5. The motor and arm of the UAV can be seen screwed into the thrust stand in Figure 26 (the propeller of the motor is not seen in the picture because it is not attached yet). The arm is being used as a spacer to attach the motor to the stand.
6. The three wires of the motor are plugged in and screwed tight in the corresponding holes for the correct esc wires (the esc and motor wires could go in any hole as long as they are connected to the correct pair).

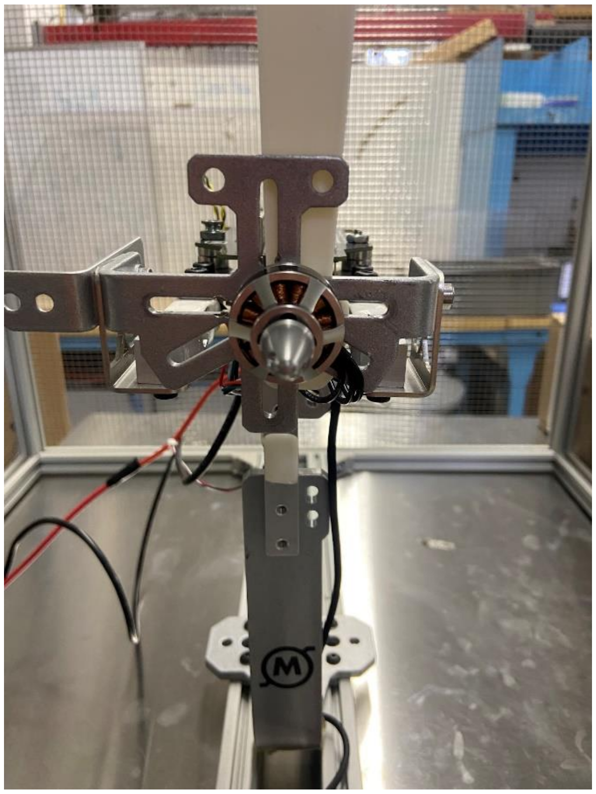


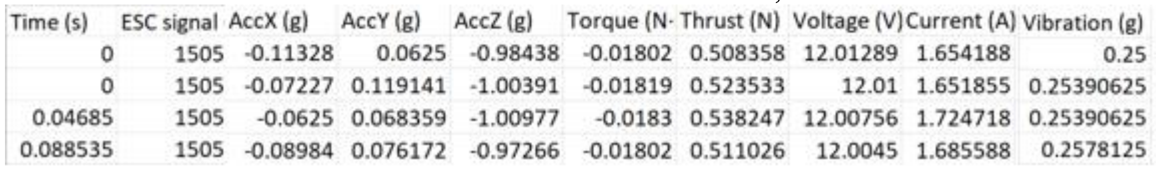
Figure 26: Motor (without propellor) Attached to Thrust Stand

7. The red and black wires seen coming from back of the board in Figure 26 are plugged into a battery and then the setup of hardware for the thrust stand is completed.

Setting up the hardware of the thrust stand can be a long process as it is difficult to screw motors into the stand. To use RC Benchmark, the hardware needs to be set up correctly or else the program will not be able to connect. The first setback that we had was that sometimes despite the hardware being set up properly, when the connect button is pressed on the computer, the software is still unable to connect. Our group at first had assumed that we set up the hardware incorrectly, however when consulting Chris and the other MQP students who have used the stand before we were informed that sometimes the software is inconsistent and will often not connect. So, after putting the hardware back to the original configuration and trying to connect a few times it finally connected properly.

In the software there were several things that we needed to change before we could test the motor. For our motor to run we had to set the safety cutoffs above what our motor needed to work (such as the current being set to a 4-amp cutoff, because 3.3 amps should result in half thrust for our motor). Then once we selected the graphs we wanted displayed (Thrust being the most important) we went into the manual control section of the software, turned the ESC power on half strength, and recorded the data. It seemed to be working as intended and the data we were receiving all seemed to be around the same values for each of the different motors, so we took down our motor from the stand and went back to the lab to analyze the data (Table 7).

Table 7: Motor 1 Thrust Stand Data, Test 1



Time (s)	ESC signal	AccX (g)	AccY (g)	AccZ (g)	Torque (N·	Thrust (N)	Voltage (V)	Current (A)	Vibration (g)
0	1505	-0.11328	0.0625	-0.98438	-0.01802	0.508358	12.01289	1.654188	0.25
0	1505	-0.07227	0.119141	-1.00391	-0.01819	0.523533	12.01	1.651855	0.25390625
0.04685	1505	-0.0625	0.068359	-1.00977	-0.0183	0.538247	12.00756	1.724718	0.25390625
0.088535	1505	-0.08984	0.076172	-0.97266	-0.01802	0.511026	12.0045	1.685588	0.2578125

Table

The relevant parameters that the thrust stand records data for are shown in Table 7, this data was recorded over 200 intervals of time for all four motors and compared, and the following Table 8 was created from the raw data which led us to believe that something was not right.

Table 8: Average Thrust for All Motors, Test 1

	Motor 1	Motor 2	Motor 3	Motor 4
Average Thrust (Newtons)	0.521871	0.70975	0.65288	0.75174

Once we looked at the data and converted the numbers for the average thrust into grams, we quickly realized that something was not right. The data, as seen for one motor and the average thrust values for the other three in Table 8 show that each motor was only generating between 50 and 75 grams of thrust at what we thought to be half power. This was an issue because each motor was listed to lift ~195 grams at half power, if the data we collected was correct, the UAV would not be able to fly, and the manufacturers spec sheet would be inconsistent with the performance.

To determine what the issue was in our test we first reviewed the hardware set up. The thrust stand had a dedicated ESC and battery that was for use of everyone, and it was suggested just to use the standard thrust stand ESC and battery. We realized that having a different ESC with different amperage limits and a battery with a different voltage would impact the expected performance of our motors; so, we decided to retest motor 1 with our battery and the given ESC, and then our battery and our ESC to see if there are any noticeable changes.

After testing with our battery and the given ESC, the average thrust for motor 1 increased by 0.3 Newtons, which was noticeable, but not significant enough to solve our problem. Then the third test with our battery and our ESC (which we ran through our PDB as well to verify compatibility of our components) went up 0.4 Newtons from the original test.

Making these changes did increase motor performance, however the increase in thrust still resulted in half of the expected and required thrust for this UAV to work properly. After talking to other teams for ideas on what to do next to make it work, we were told that the thrust stand produces unreliable data at times and it may not be our setup, so we decided to keep these results in mind and continue assembling the MAV. With unreliable thrust stand data, the thrust capacity of the motors was verified by lifting the fully assembled MAV.

3.4 **Vicon**

The path learning how to operate the Vicon motion capture system was filled with many obstacles that challenged us and ultimately created a much more complete understanding of the system hardware and software. Every problem solved in the process allowed us to understand a different component of the system and generate a more complete user's guide (Section 7). Steady progress over the course of months resulted in the ability to track object both live and recorded and stream live position and attitude data to a MATLAB script.

The Vicon motion capture software was set up to determine the position of the MAV within the testing area. Vicon Tracker software creates a virtual 3D capture volume using a infrared sensors placed around the room. This software has the ability to relay live three-dimensional position and orientation data to the ground station to provide information necessary to fly quadrotor autonomously. The series of figures below show screenshots of the software and its capabilities.

Eight Vicon Vero cameras surround the capture volume, so that the MAV is visible from multiple cameras at any given time. The cameras themselves track the MAV by detecting infrared light reflected of spherical tracking markers, as seen in Figure 27 which are placed asymmetrically on the object is to be tracked. The Vicon markers must be placed asymmetrically on the UAV so that the Vicon software can determine the axes and orientation of the MAV, without accidental axis inversion that can occur with a symmetrical placement.

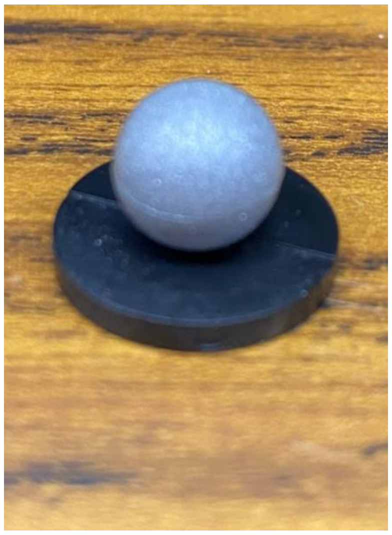


Figure 27: Vicon Reflective Tracking Marker

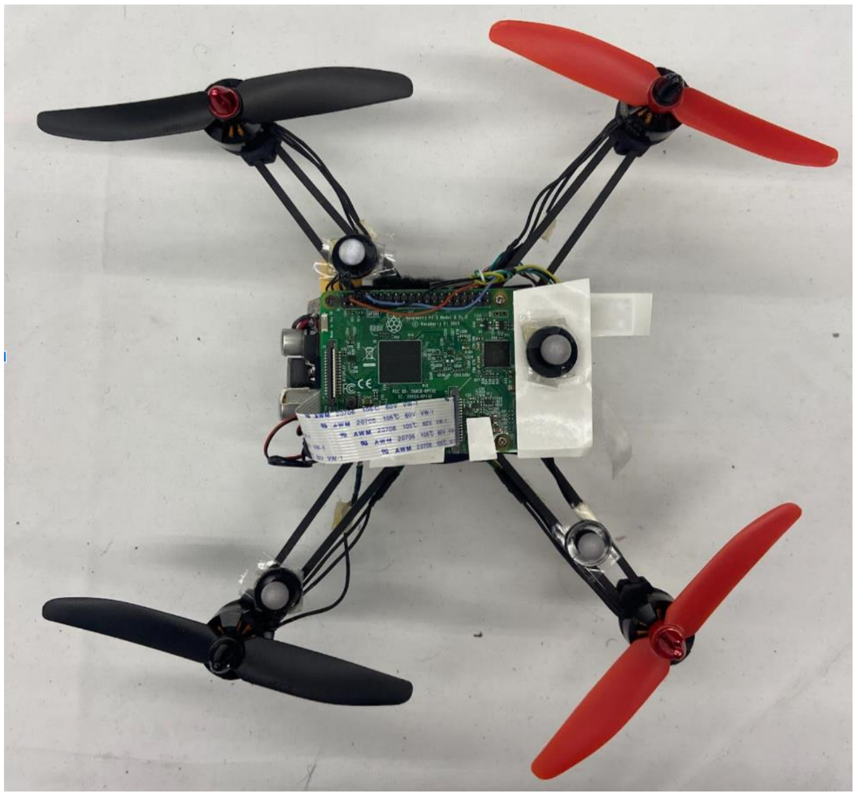


Figure 28: Example Quadrotor with Attached Markers

A UAV from a previous MQP that was available was used as a test UAV for the Vicon system while we were simultaneously creating the MAV for this project. Figure 28 above is an image of that test UAV, with asymmetrically placed reflective markers. If the markers were placed in the shape of a square, one on each of the arms of the UAV, the Vicon software would be unable to differentiate them, because the position of each marker would the same relative to the others. In a symmetrical pattern, the Vicon software struggles to track the orientation through complex maneuvers, and the axes in the software are constantly flipped 180 degrees and back. Vicon publications and guides state to place them asymmetrically, however it was not until testing that the reasoning behind this became evident.

When making an object in Vicon Tracker, the selected tracking markers that make up said object are only known in relation to each other. There is no way for Vicon to differentiate between the tracking balls besides the fixed position of one compared to the others on the object being tracked. Similar to the need for asymmetry, varied differences between markers enables more accurate orientation readings.

The first step in tracking the MAV was setting up the hardware of the Vicon system. Getting information on this step, and much of the software process, was difficult, due to few online publications or forums other than official Vicon documents. The Vicon user guides are helpful, however they have many different products and set ups, and sifting through them to find specific information can be difficult. Although no WPI groups had worked with the system extensively, professor Cowlagi, the project advisor referred us to a student that worked with the system in the past. We were able to contact Chase St Laurent, who graciously agreed to lend his knowledge of the Vicon hardware setup.

To prepare for Chase, we organized all the PoE cables equipment that had been stored in the lab, including the cameras, tripods, and switch box. Chase was able to help us mount the Vicon Vero cameras to the tripods and gave us suggestions about where to place them in the room to get the best view of the capture volume. We were also shown how to wire the cameras to the Vicon station at the computer and how to orient and number the cameras in the Vicon software. It was a productive meeting that got cut short with a software error, in which the cameras were detected by the tracking software but not transmitting any data, that we did not know how to solve. Before he left, he gave us some tips on how to calibrate the cameras with the wand, unfortunately the cameras could not be calibrated due to the software issue, but it was nonetheless useful information later in the project.

The day after Chase was with us, we attempted to complete Vicon tracker software setup to the point we stopped, however we were unable to. Chase had downloaded the most recent version of Vicon Tracker, however it was only available from his account, so we were unable to load the software. When attempting to download the most recent version to our own accounts, we were not able to because we did not have admin privileges on the computer as Chase did. We reached out for help first from Vicon customer support, who were extremely helpful and responded quickly with suggestions, and second to Professor Ed Burnham to obtain admin privileges for the Vicon ground station computer.

Vicon customer support responded that day with suggestions about network settings we could adjust (details of our troubleshooting process and what settings/configuration were changed can be found in Section 8), which we also needed administrator access to modify.

After a short wait, we were able to obtain admin accounts on the computer and we could continue with the process. Progress was made with the suggestions from Vicon support, and we were able to transmit data from the cameras to the software and view the live feed of each individual camera. The next step was then to calibrate the cameras using the wand. The wand is a T shaped device approximately one meter long with red LEDs placed along it as seen in Figure 29 below.

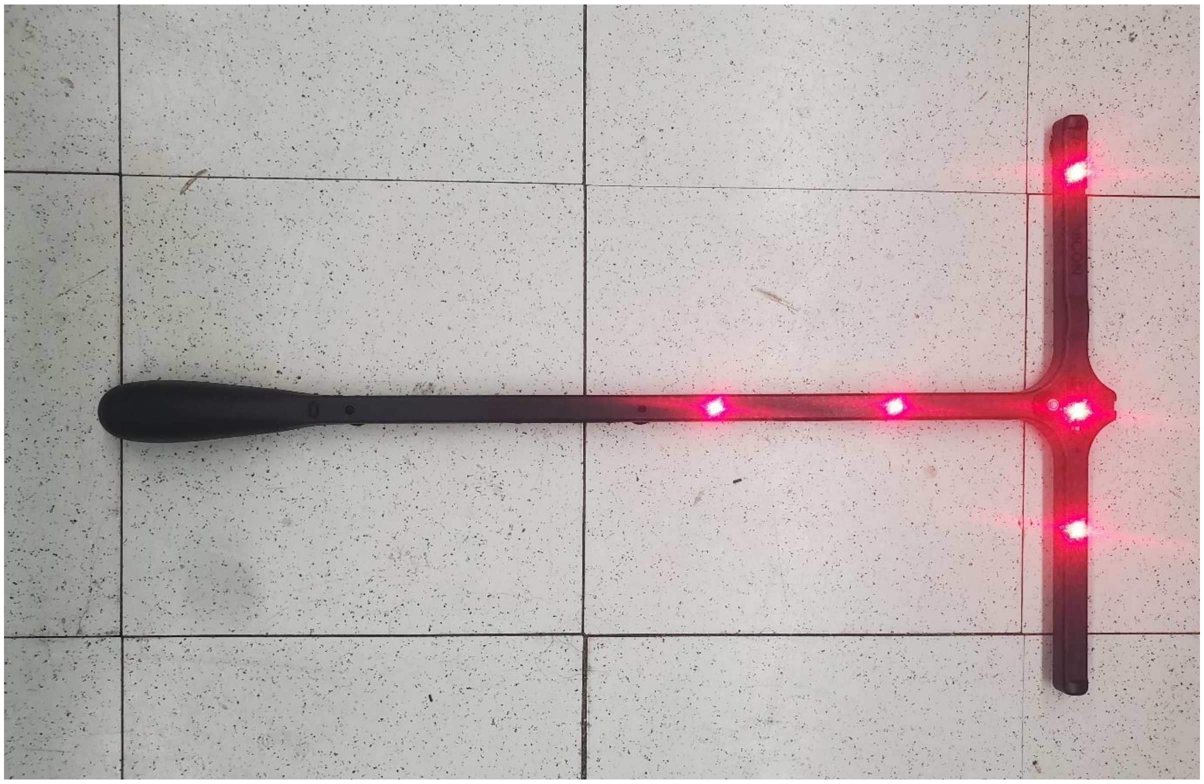


Figure 29: Vicon Active Calibration Wand

With the cameras transmitting data, we could see the wand on our screen as seen in one of the camera views as yellow (+) symbols (Figure 30). However, upon clicking the calibrate button on the left side of the screen, the program would display an error message and promptly crash.

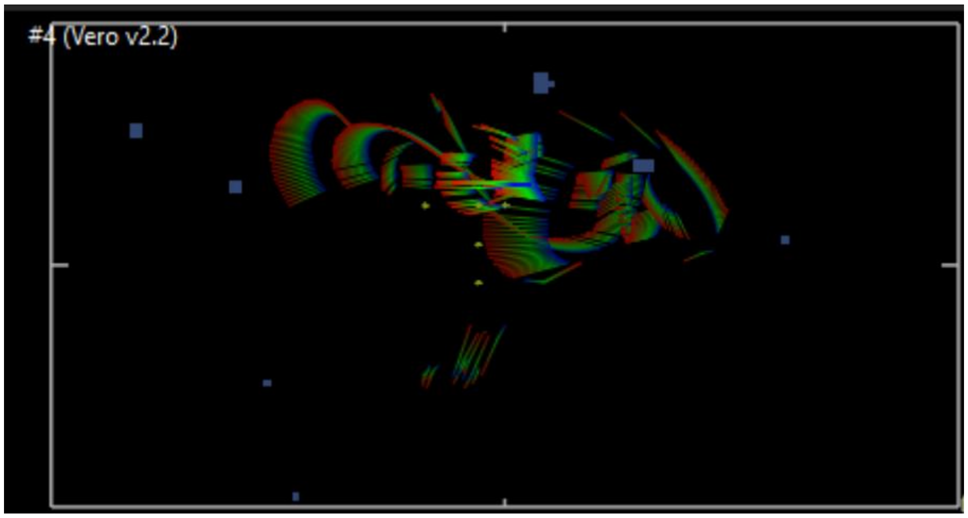


Figure 30: Vicon Camera View During Calibration

After contacting Vicon support and describing the problem, they suggested a Zoom call to resolve the issue live. On the call, we modified various settings, including disabling Windows

Firewall, as the cameras are connected to the PC via ethernet, and the firewall can block unrecognized networks. Upon seeing the error, Vicon support recommended that we ensure that the monitor was plugged directly into the Mini DisplayPort of the graphics card instead of the motherboard of the PC. Unfortunately, at the time there were no available adapters to connect the HDMI monitor to the Mini DisplayPort compatible graphics card. After the meeting, we found a second monitor, as well as a series of adapters to connect the original HDMI monitor to the graphics card, but upon using it, the same calibration error occurred.

We decided to restart the computer as part of the troubleshooting process. Upon restart, the BitLocker encryption software enabled on the PC completely prevented the use of Windows without an encryption key, which we did not have. We contacted Professor Burnham once again to obtain the encryption key and regain access to the system. After some time, we received the key and were able to continue.

However, the same calibration error persisted. Although the issue was not resolved, we decided to continue using the graphics card port to avoid other potential problems. We once again set up a meeting with Vicon support to troubleshoot. After several hours and multiple Vicon support personnel, the calibration error was resolved. (Complete details of the solution can be found in Section 8, although not all unsuccessful attempts were documented). With this progress, we were able to successfully calibrate the cameras and use the test UAV to create an object inside of the software the next day (Figure 31).

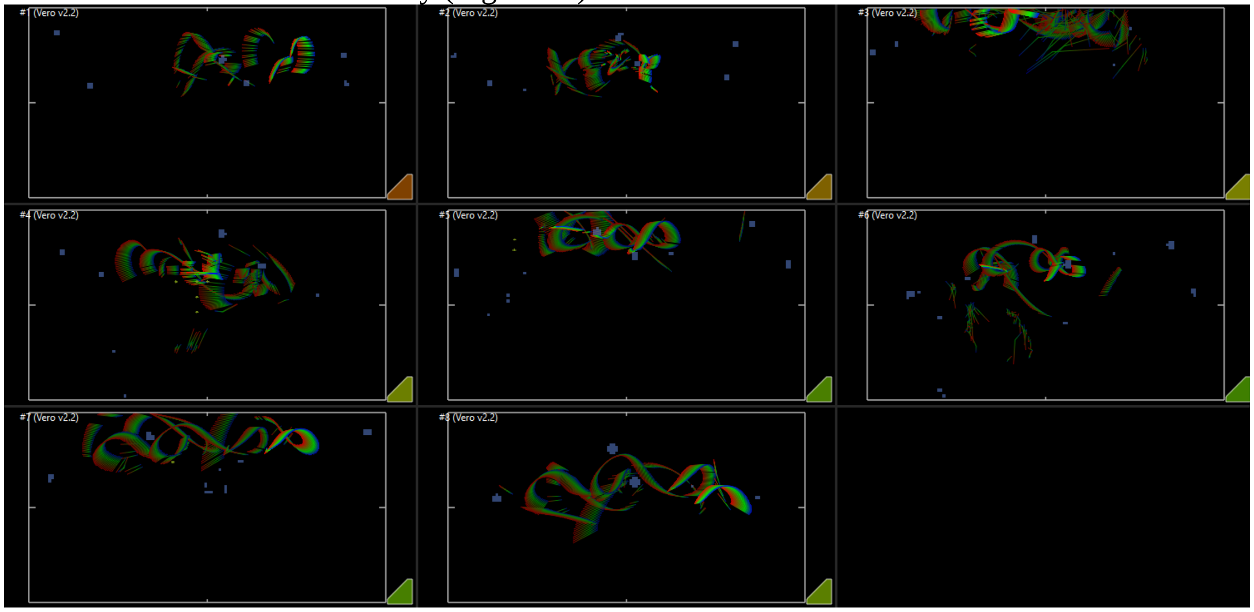


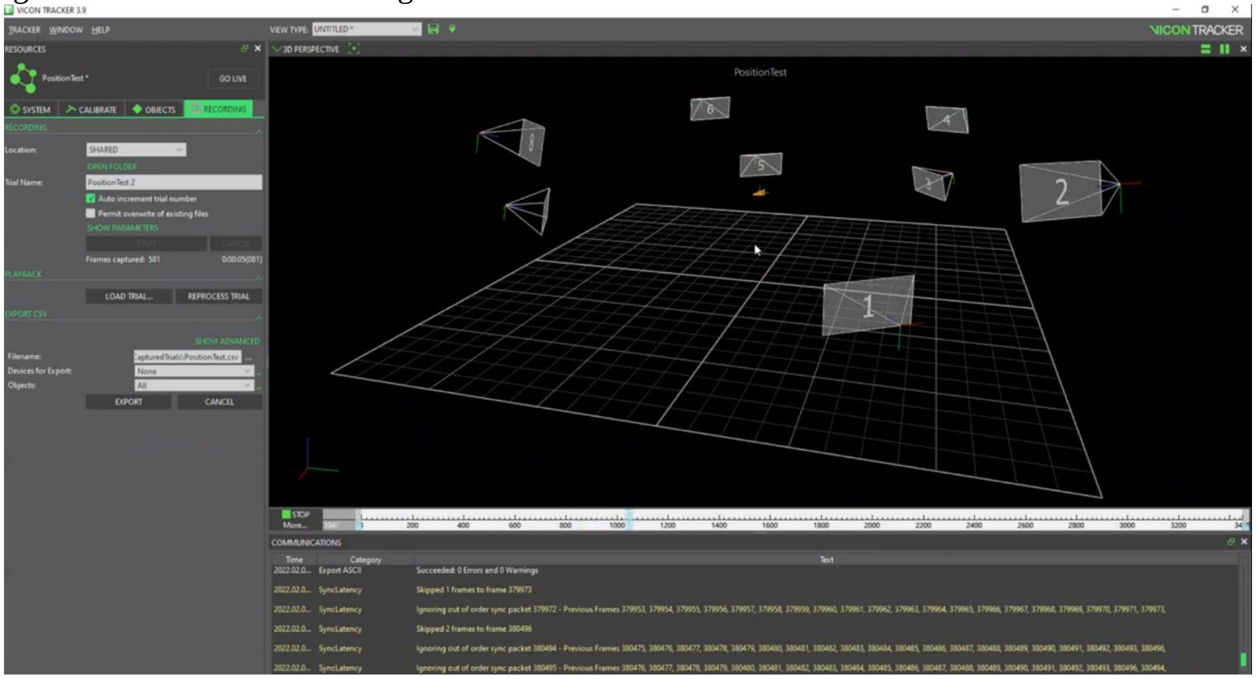
Figure 31: Camera View During Calibration

The setup of the Vicon system was nearly complete. The final step was to track objects in real time and send live data to third-party programs. While attempting to continue our progress to send live data, we were stopped yet again by a different BitLocker encryption key. The previous key used was not correct and yet again we turned to Professor Burnham for help in search of a permanent solution. The decision to reimage the computer with a fresh installation of Windows was made. With nothing stored on the PC that could not simply be reinstalled, this could eliminate many errors. Later in the week, Professor Burnham came to the lab and reimaged the computer, giving us a fresh start.

There was, however, a benefit to this fresh start, now that we had a completely erased computer to work with, we had the opportunity to follow the Vicon manual that we had been producing throughout the process of us learning how to work with it. Although it was much easier to set up for a second time with our experience, we still encountered new obstacles.

When we first tried to download the Vicon tracking software, we were not allowed to because of licensing reasons. We had not run into this problem the first time we set up Vicon because the program was already downloaded, and the license was set up. We thought that the license was just a Vicon USB that is always plugged into the computer, however after talking with Vicon customer support again they emailed us a digital license that we need as well as the USB. With the licensing figured out we were able to download the latest Vicon tracker program and continued to follow our manual, making it more detailed as we went along. Again, we came across a new problem that we had not seen before: the cameras were sending data, but we could not see any images on the screen. This was different than an earlier issue where we could not see anything on the screen because of the cameras not sending any data, so it looked the same on the surface, however we knew the cameras were sending data this time because their blue lights were on (blue lights indicate the cameras are on and sending data). This issue prompted another zoom meeting with Vicon customer support, where we went over what we did last time to make it work and what things we have tried already. After several failed solutions he suggested redownloading the drivers to the Nvidia graphics card. Although the computer said in the Nvidia control panel that the drivers were fine, this ended up being the problem. When the computer was reimaged, it wiped the drivers off the graphics card, and so redownloading them solved our problem.

After the new set of setbacks, we were able to get back to the point we were at before, with calibrated cameras and a computer we could log into the next step was yet again to collect live data of objects moving in the flight area and send it to our ground control station in real time. A video of the Vicon system live tracking the object can be found at https://youtu.be/IGXECrBQFx8 with a screen shot in Figure 32. Note that the low frame rate and stuttering is due to the screen recording software and not Vicon Tracker.



RECORDING
Location: SHARED
OPEN FOLDER
Trial Name: PositionTest 2
[✓] Auto increment trial number
[ ] Permit overwrite of existing files
SHOW PARAMETERS
Frames captured: 581 0:00:05(081)
PLAYBACK
EXPORT CSV
SHOW ADVANCED
Filename: CapturedTrials\PositionTest.csv
Devices for Export: None
Objects: All

```
Time          Category       Text
2022.02.0...  Export ASCII   Succeeded: 0 Errors and 0 Warnings
2022.02.0...  SyncLatency    Skipped 1 frames to frame 379973
2022.02.0...  SyncLatency    Ignoring out of order sync packet 379972 - Previous Frames 379953, 379954, 379955, 379956, 379957, 379958, 379959, 379960, 379961, 379962, 379963, 379964, 379965, 379966, 379967, 379968, 379969, 379970, 379971, 379973,
2022.02.0...  SyncLatency    Skipped 2 frames to frame 380496
2022.02.0...  SyncLatency    Ignoring out of order sync packet 380494 - Previous Frames 380475, 380476, 380477, 380478, 380479, 380480, 380481, 380482, 380483, 380484, 380485, 380486, 380487, 380488, 380489, 380490, 380491, 380492, 380493, 380496,
2022.02.0...  SyncLatency    Ignoring out of order sync packet 380495 - Previous Frames 380476, 380477, 380478, 380479, 380480, 380481, 380482, 380483, 380484, 380485, 380486, 380487, 380488, 380489, 380490, 380491, 380492, 380493, 380496, 380494,
```

Figure 32: Screenshot of Vicon Tracker Position Test

For this step we went back to following Vicon's setup guides, because this was an area that was better documented on their behalf. We downloaded a data stream SDK (software development kit) from their website and some MATLAB code also from their website that helped us achieve this goal. We modified their code to produce graphs by making matrices of the Euler angles and position vectors using timestamps. We used these graphs to check and make sure that Vicon was working correctly and the data it was producing was correct. We did this by moving the test UAV in specific paths that would create obviously identifiable shapes when plotted. These tests and graphs can be seen in our results section in greater detail. The ground station software that we are using can use data from MATLAB to control MAV, so the next step was to set up ground station to receive data. A manual for how to set up the whole system can be found in Section 7.

3.5 Ground Station (Mission Planner)

Familiarizing ourselves with the ground station software early on consisted mainly of plugging in the Pixhawk 4 mini to a computer with a micro-USB to USB cable and performing calibrations. As mentioned earlier in section 2.11 ,we chose to use Mission Planner instead of Qgroundcontrol because the telemetry radio connection between ground station PC and Pixhawk worked more consistently with Mission Planner.

Before we could attempt a test flight, we first had to calibrate and configure the MAV within the ground control software. In order to do this we followed the steps in the setup section of Mission Planner. With the Pixhawk connected through telemetry we first downloaded the current firmware from Mission Planner for a quadrotor with our motor configuration and went through the calibration steps for the Pixhawk alone first. Next we connected the ESCs and motors, without the propellers attached, to the Pixhawk and tried to calibrate the motors and ESCs through the setup section. This is where we ran into the bulk of our time spent on Mission Planner. After connecting an Xbox controller to the computer and binding the throttle, yaw, pitch and roll to different inputs on the controller we started our first attempted to spin the motors manually. For greater control in intial testing, we decided to use a manual controller, as an autonomous control scheme could introduce unwanted error. At first we only connected and powered one motor/ESC pair to the Pixhawk, but several error messages would display when we would attempt to arm it. The errors would range from logging errors to failsafe problems to calibration errors. Each error was researched throught he Ardupilot and Mission Planner websites and forums until we found a solution that worked for each. The only error message we needed to account for regularly was a “Throttle too high” error upon arming the MAV. In the binding process of the controller, the two variable triggers are treated as a single axis with the right trigger increasing the value from 50% and the left trigger decreasing the value. While this could be helpful in the case of a pilot accidentally releasing the throttle during flight, it required the left trigger to be fully depressed to hold the the throttle at zero to arm. These errors were solved in multiple areas of Mission Planner, however a many of the fixes were made in the configuration settings.

The errors were solved until all four motors were spinning correctly and we could assemble the MAV and run a test flight. The test flight is discussed further in the results section. We practiced flying with the controller and tuning the Pixhawk so that it stabilized the MAV better. If someone were to make the MAV autonomous, Mission Planner would also be required receive the telemetry data from another source (such as MATLAB as part of the Vicon SDK) and relay it to the Pixhawk 4 mini.

4 Results

4.1 Vicon Experimental Results

To determine if the Vicon motion capture system was working properly we conducted several tests to see if the plots that Vicon was outputting to us were correct. To accomplish this, we manually moved the test MAV (holding it, not flying it) in simple paths in which we know what the graphs of the yaw, pitch and roll should look like, such as keeping the yaw and pitch constant while only rotating the MAV in the roll plane. The first of these experiments was a baseline angle test in which we just placed the test MAV motionless on the origin of Vicon system (Figure 33).

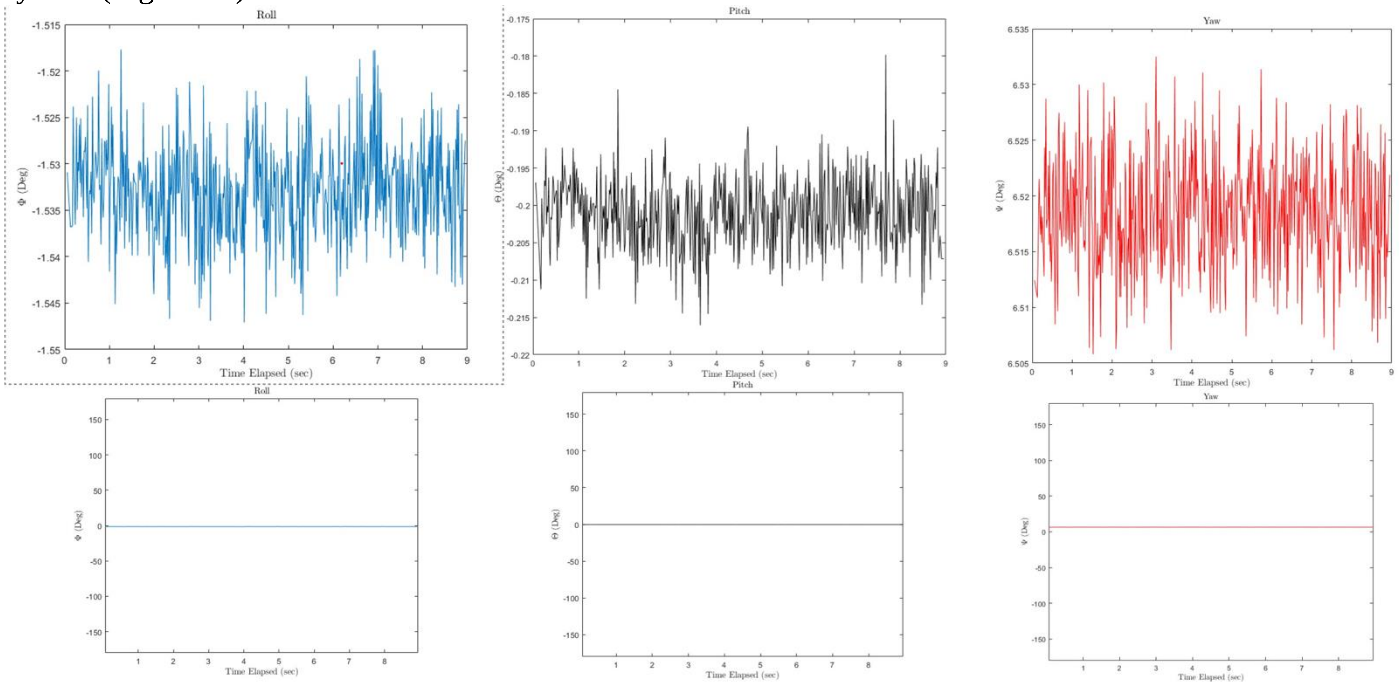


Figure 33: Vicon Baseline Euler Angle Test

The plots turned out almost exactly as expected. The yaw, pitch and roll are all constant, the first row of graphs does not look constant because Vicon, like any sensor, does have noise, however each of the angles varies only by around 0.2 degrees when stationary because of the noise. This is acceptable because as shown in the second row of graphs, on a scale with a full range of motion the Euler angles appear constant. Important to note is that this is raw data with no filtering applied, in a practical application, filters would be applied to provide better state estimates.

The second of these experiments was a manual yaw rotation, where we held the test drone in the air and spun it in a complete circle while keeping it level (Figure 34).

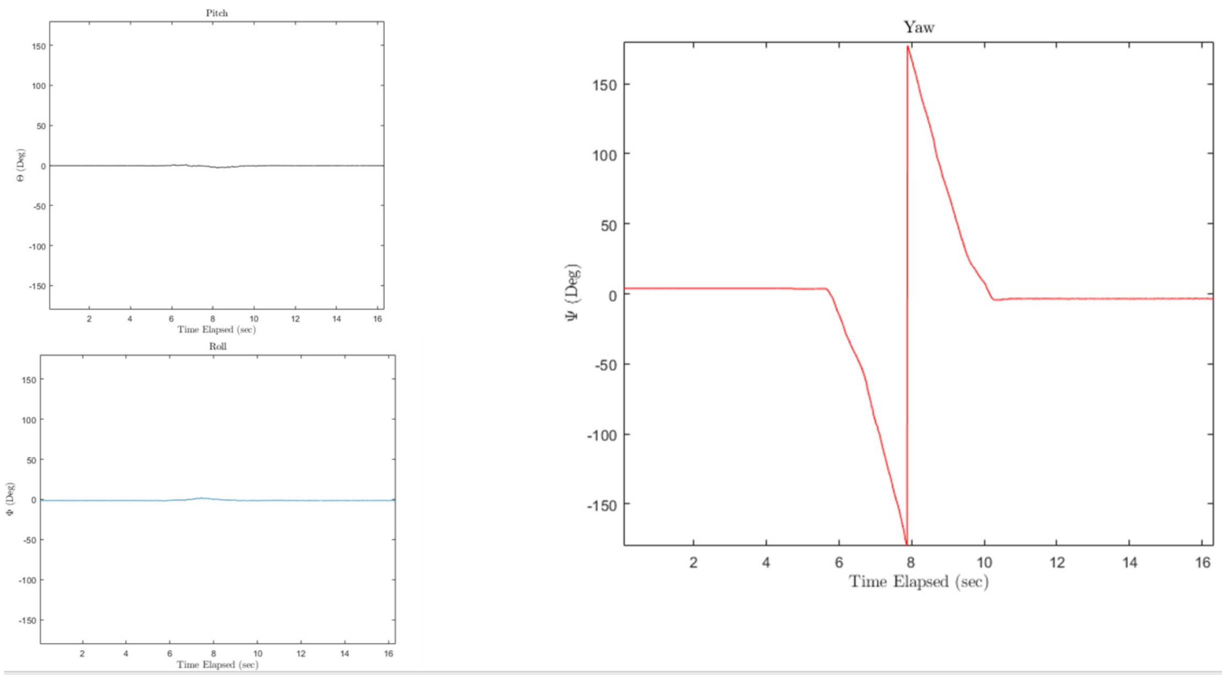


Figure 34: Vicon Manual Yaw Test

Again, the plots turned as expected, in this case we only should see the yaw angle move 360 degrees as it was turned in a full circle while the pitch and the roll stay constant. The yaw graph is shown to not be smooth because the yaw axis is set from –180 degrees to 180 degrees, and when it passed –180 degrees it jumped up to 180 and continued down instead of smoothly travelling to -360 degrees.

Similar tests were performed for the roll and pitch, for the roll test the test MAV was held in the air and tilted 90 degrees to the left and then 90 degrees to the right (Figure 35).

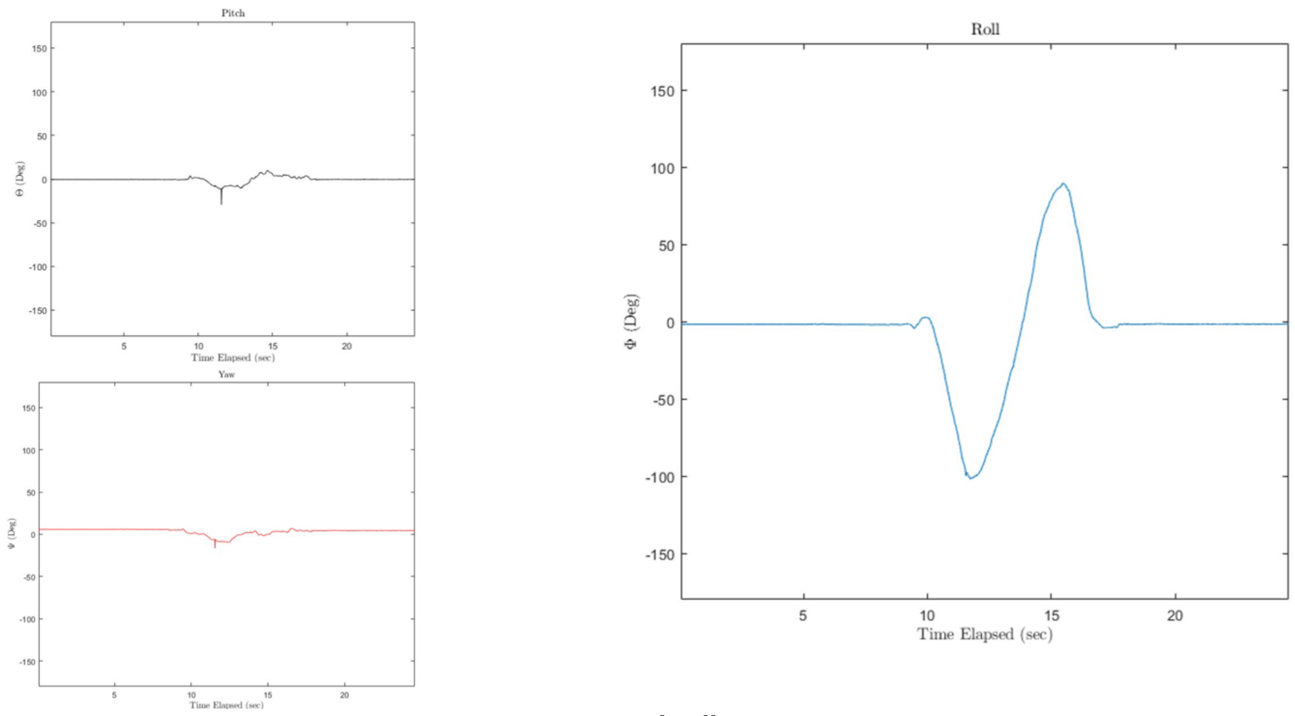


Figure 35: Vicon Manual Roll Rotation Test

The resulting graphs for the roll test are as expected, but with a little more variance in the pitch and yaw than with the yaw test, this is due to the person holding it being less steady, however for the pitch test we got odd results when it came to the roll and yaw.

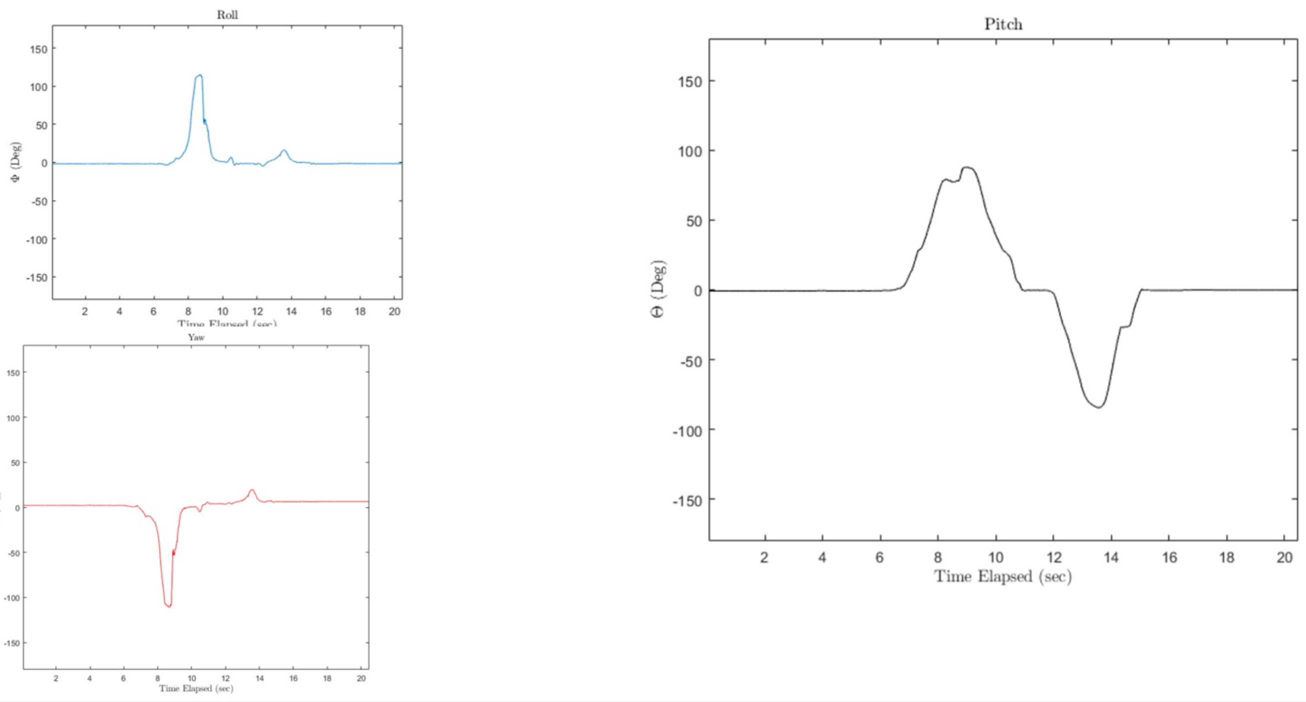


Figure 36: Vicon Manual Pitch Rotation Test

We repeated the pitch test several times, and each time we got the results we were expecting from the pitch graph after tilting the test MAV up 90 degrees and then down 90 degrees, however each time the roll and yaw also indicated a direction change when the test MAV was first tilted upwards (Figure 36). This was not expected, but Vicon may use a different definition for Euler angles than expected. Changing the orientation of the test MAV could find the plane Vicon defined as pitch through trial and error, however as long as the system is tracking pitch correctly it will work. For future flight, ensuring that the axes defined as an object in Vicon Tracker match those of the physical MAV is important.

Similar to the Euler angle tests, the position measurements were verified with the object at a standstill with noise on the order of ≈ 0.1 mm, well within a reasonable value (Figure 37).

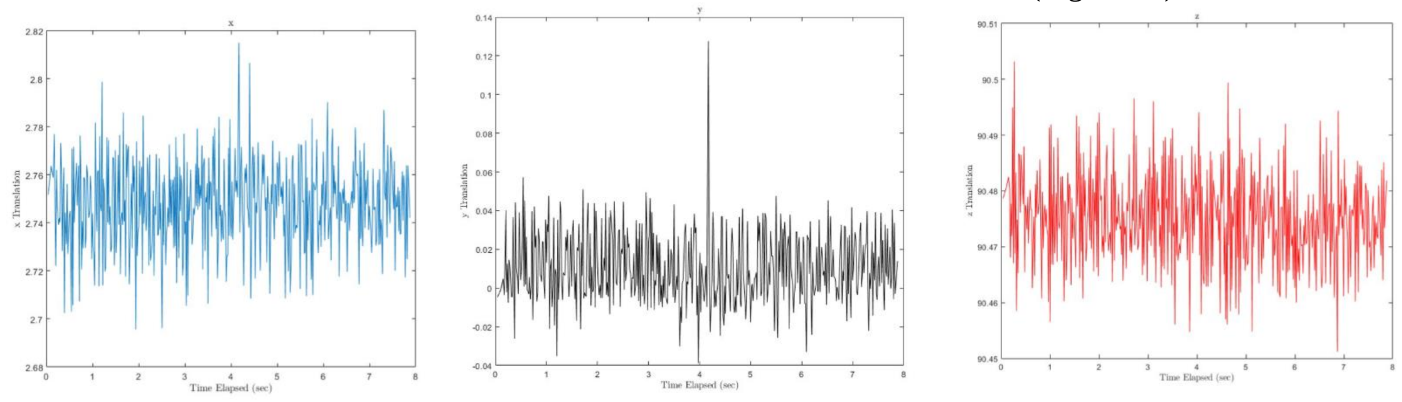


Figure 37: Vicon Baseline Position Test

The test MAV was placed at the origin of the Vicon plane, that is why the x and y positions are close to zero, though human error when placing the test MAV caused the small distance from exactly 0. Initially we expected all 3 coordinates to be at zero because the test MAV was on the ground, however the z coordinate varies around 90 millimeters. This is because the origin of the Vicon axis is on the ground and the object coordinates are measured from its center of volume, hence when the test MAV is on the ground its center of volume is above the ground and the z coordinate is above 0.

The next position test that was conducted was lifting the test MAV straight up and then straight down.

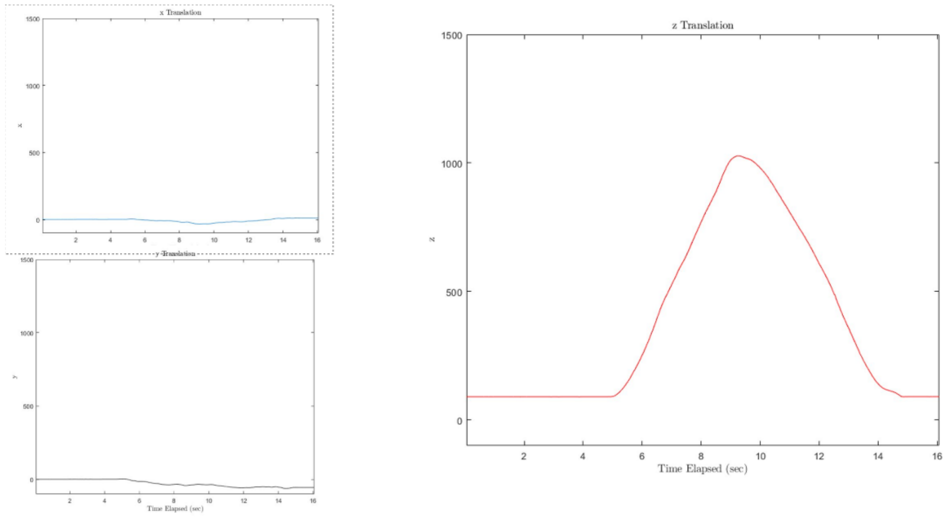


Figure 38: Vicon Manual Vertical (z axis) Translation Test

The position data was plotted expected from this test; the x and y coordinates stayed near zero, human error cause slight movement in them because we did not lift it perfectly straight, and the z coordinates are shown to go up and then back down over time (Figure 38).

For the final position test, we moved the test MAV on all of the axes consecutively during the same ‘flight’, first lifting the test MAV straight up, then moving it left, right and back to center in the x direction, forward, backward and back to center in the y direction, followed by lowing the test MAV back down to the ground.

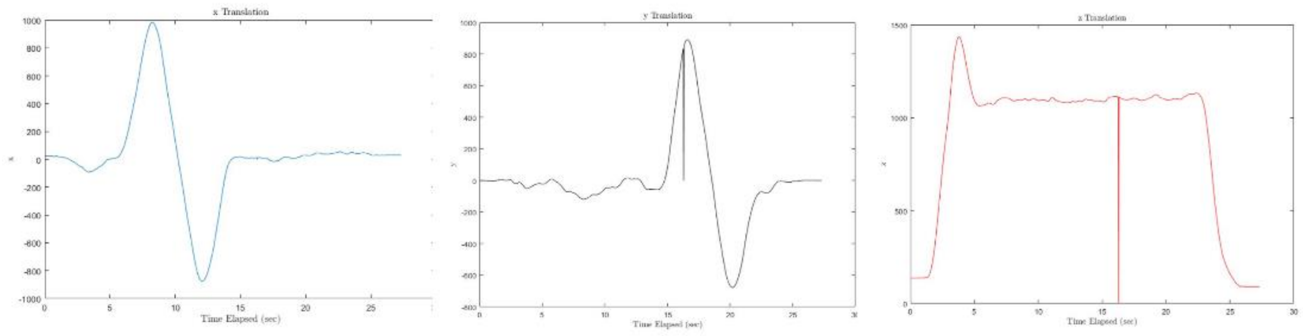


Figure 39: Vicon Manual 3-axis Test

The results from this test were also as expected and therefore a success. All three graphs represent the movement of the test MAV correctly, the only error occurred when the Vicon cameras lost sight of the test MAV and all three coordinates went to zero for a split second before the cameras relocated the test MAV and had the correct coordinates again. This can be seen in the graphs as a straight line to 0 can be seen at the same time stamp with the plot then resuming its trajectory on all three graphs (Figure 39). The default state for position data in the Vicon system is zero, so any loss of tracking will result in all values being set to zero.

The Vicon software provides us with much more information along with the Euler angles and position of the MAV, such as velocity and acceleration of the MAV, that are useful but harder to independently verify if correct. With the plots from Vicon that we have recorded being correct though, it does appear that the whole of the Vicon system operates exactly as expected.

4.2 MAV Experimental Results

The MAV without a payload (microcontroller and sensors) conducted several tests with manual control via Mission Planner. The Vicon motion capture system was also live and used to track the MAV throughout the flight. Vicon Tracker was used to send data to be sent to and stored via MATLAB. Due to insufficient time, stable flight was not achieved. However, a flight test with both video and motion capture data was observed.

Figures 40 through 41 illustrate the data captured by Vicon Tracker and plotted in MATLAB. Initially, the MAV was placed near the origin of the capture volume. Visible from these plots is the total flight length of approximately 13 seconds. Also visible is an error in the system near the 42 second mark where all translations displayed as zero. This is a result of not enough information being gathered by the cameras at that moment, resulting in the default value of zero being output. At this point in time, one or more cameras were likely occluded by a team member recovering the MAV, as the landing gear had not been installed for that test.

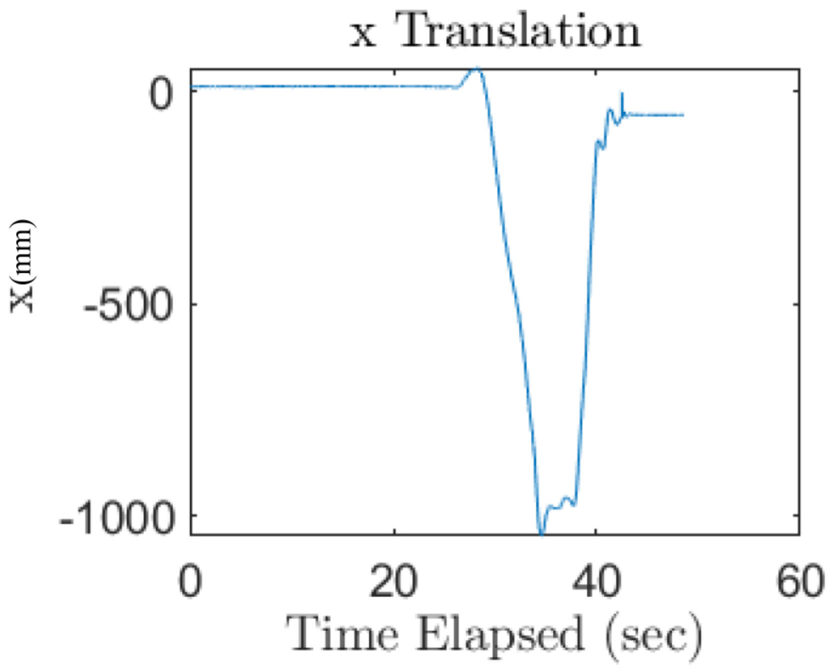


Figure 40: x-axis Motion Capture Data of Flight

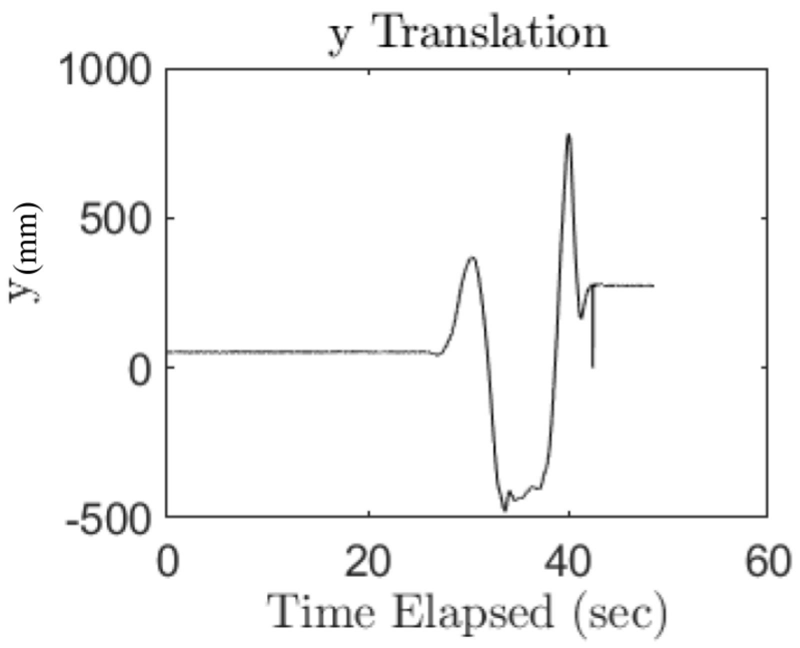


Figure 41: y-axis Motion Capture Data of Flight

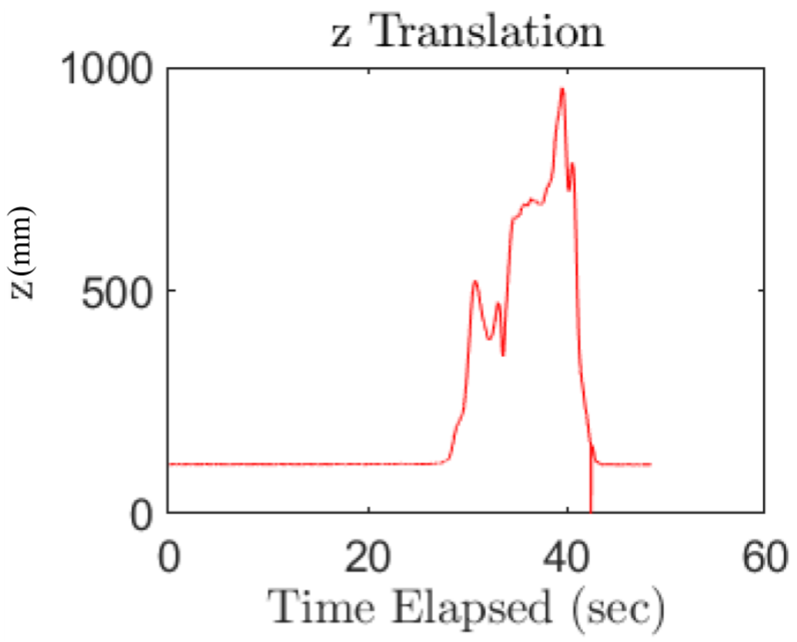


Figure 42: z-axis Motion Capture Data of Flight

A video of this flight can be found at https://youtu.be/G5ZFRYQ1MrE (Figure 43).

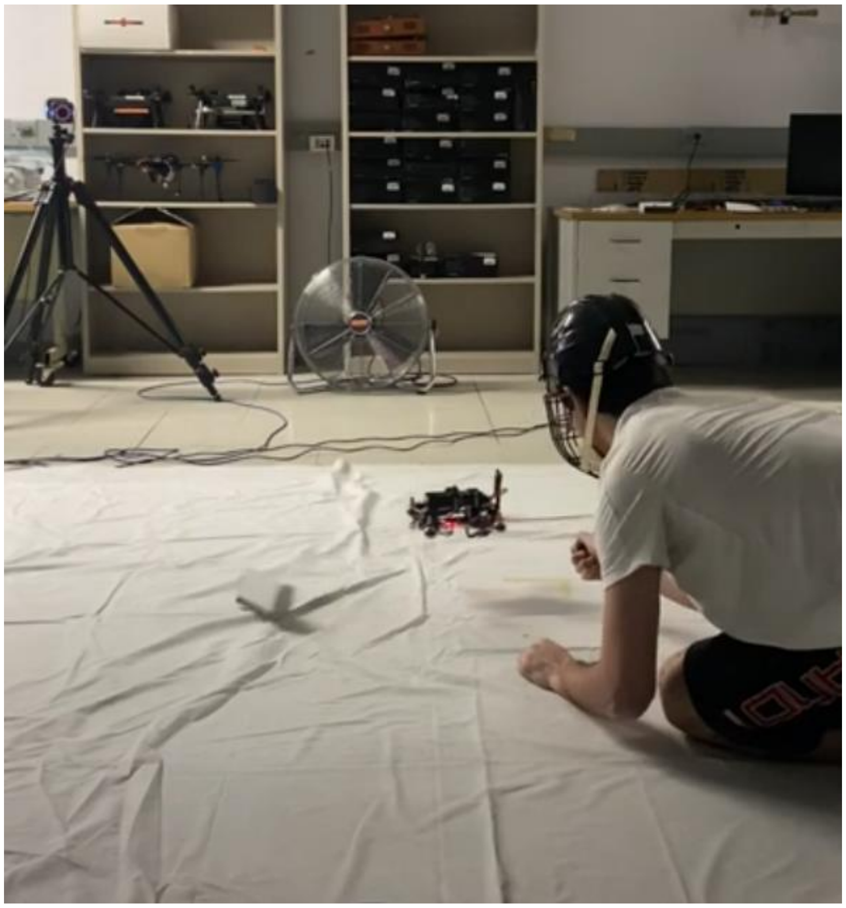
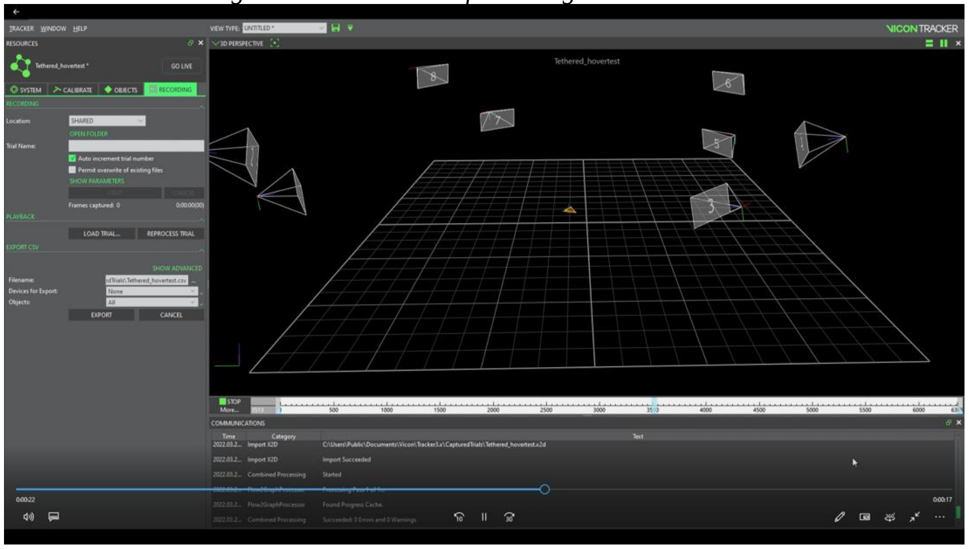


Figure 43: Screenshot of Test Flight Video



```
Time        Category              Text
2022.03.2…  Import X2D            C:\Users\Public\Documents\Vicon\Tracker3.x\CapturedTrials\Tethered_hovertest.x2d
2022.03.2…  Import X2D            Import Succeeded
2022.03.2…  Combined Processing   Started
2022.03.2…  Flow2GraphProcessor   Found Progress Cache.
2022.03.2…  Combined Processing   Succeeded: 0 Errors and 0 Warnings
```

Figure 44:Screenshot of Motion Capture Recording of Test Flight

A video of the recorded motion capture data from this flight can be found at https://youtu.be/CCB50qh9wYo (Figure 44).

Although being manually controlled adds a certain instability, the flight was conducted in "stabilize mode" within the Mission Planner software. This mode permits manual flight, but selflevels the roll and pitch axes. With this mode, simply managing the throttle input should allow purely vertical motion of the MAV. However, this was not the case. Instead of vertical level flight, the MAV began to roll and pitch, and manual adjustments were needed to avoid a collision. The motion of the MAV was too uncontrollable to fly, and the throttle was lowered to land. Subsequent flight tests proved no more successful, despite tuning adjustments within Mission Planner.

At any time when propellor were spun while in a flight configuration, the 3D printed PLA motor arms of the frame visibly deformed and bent upwards in the direction of thrust. A deflection of at least 0.5 cm was observed, although no reliable measurement methods were available. This likely had a major effect on the ability of the Pixhawk to create stable flight.

5 Conclusions

The goal of this project was to design a low-cost, lightweight (≈ 500 grams) autonomous MAV capable of a variable sensor load that utilized Vicon motion capture system for indoor flight. The MAV was to be easily constructed by a single person with easily manufactured or acquired parts. A microcontroller would be required to process incoming sensor information and support the wide variety of sensors that could be installed to the MAV.

Throughout the project research on previous projects and existing MAVs was considered for our design. In addition, a several microcontrollers were researched and tested to determine that the ODROID-XU4 would be the most appropriate for this project. From there, work to connect the ODROID to the Pixhawk flight controller was completed. In addition, two team members spent several months understanding and troubleshooting the Vicon system, creating a manual to aid future teams. Finally, a prototype MAV was designed, constructed, and flown manually in a basic configuration to demonstrate a degree of flight capability.

No flight tests were able to show steady, level flight. However, a 13 second test flight showed the ability of the propulsion system to maneuver the MAV and respond to manual control inputs. This test also verified that all components of the MAV were compatible with each other. Additionally, the functionality of the Vicon system was tested in a realistic scenario, live tracking an MAV in flight and providing real time data. This data was then able to be read by a MATLAB script and plotted. This sets a baseline for further manipulation of data by MATLAB, such as implementing a Kalman filter to provide state estimations of the motion capture data and reduce noise. To manual created to operate and troubleshoot the Vicon system thoroughly details the steps needed to set up the hardware and software components of the system, with information on resolving the types of problems encountered.

One of the major lessons learned with this project was the importance of the tuning and configuration settings of the ground station software. Despite having an MAV that was physically capable of flight and recommended standard configuration/tuning settings, flight of more than a few seconds was impossible without further work in the software. Underestimating the time required to complete this was a major shortcoming of this project. In the Vicon aspect of the project, the value of the Vicon customer support team was evident. We strongly recommend any groups that use the system in the future use the resources they provide.

5.1 Recommendations for Future Work

For anyone who may continue work on this project, the first step we recommend would be to tune the Pixhawk in Mission Planner such that the MAV is capable of stable flight. This could be done with a manual controller like we were, or for an autonomous flight mode. A suggestion for making the MAV more stable in terms of center of mass is adding small ridges or guides on the top of the frame, such that the battery falls into the same place every time it is strapped on, eliminating the human error in attempting to strap the battery in the same location every time to maintain a consistent center of mass. Further tests are also needed to verify the total flight time capability of the MAV once stable flight is achieved. It is still unknown what the true power consumption of the MAV with a fully functioning microcontroller onboard is. Additional sensors would also reduce the flight time due to both increased mass and power consumption.

Another suggestion is to use carbon fiber motor arms for the MAV. The 3D printed arms deform when the motors are spinning with propellors attached and we believe it may be a cause of instability in the system. Our original plan was to have the arms be carbon fiber for this exact reason. However, ordering them was out of the time frame for this project.

A key aspect in making the MAV truly autonomous would be to fully integrate data from the Vicon motion capture software into ground station software and communication with the Pixhawk. Utilizing standard GPS modules is a straightforward process but incorporating other sources of telemetry data could prove more difficult.

Finally, it is our hope that the work completed to create Vicon manual in Section 7 can help future teams have a more seamless setup and troubleshooting process. With the help of the manual, we would like future teams to continue to generate meaningful data.

6 References

1. Valavanis, Kimon and Vachtsevanos, George. *Handbook of Unmanned Aerial Vehicles* . s.l. : Springer, 2015.
2. *Unmanned Aerial Systems - Search and Rescue Robots* . Rudin Konrad, Daniel Serrano and Pascal Strupler. s.l. : InTechOpen, 2017.
3. Airborne Personal Reconnaissance System (PRS) for Dismounted Soldiers. *Teledyne.* [Online] https://www.flir.com/products/black-hornet-prs/.
4. Sadraey, Mohammad. *Design of Unmanned Aerial Systems.* s.l. : Wiley , 2020.
5. Daly, David. A Not-So-Short History of Unmanned Aerial Vehicles. *Consortiq.* [Online] https://consortiq.com/uas-resources/short-history-unmanned-aerial-vehicles-uavs.
6. Vyas, Kashyap. A Brief History of Drones: The Remote Controlled Unmanned Aerial Vehicles. *Interesting Engineering.* [Online] https://interestingengineering.com/a-brief-history-of-drones-the-remote-controlled-unmanned-aerial-vehicles-uavs.
7. Whittle, Richard. Air and Space Magazine - The Man Who Invented the Predator. *Smithsonian.* [Online] https://www.smithsonianmag.com/air-space-magazine/the-man-who-invented-the-predator-3970502/.
8. *DJI* . [Online] https://www.dji.com/.
9. *UAV Applications: Introduction.* Valavanis, Kimon and Vachtsevanos, George. Handbook of Unmanned Aerial Vehicles, s.l. : Springer, 2014.
10. *Parrot.* [Online] https://www.parrot.com/us.
11. *The Basics of Quadcopter Anatomy.* Sustek, Michael and Úředníček, Zdeněk. MATEC Web of Conferences, 2018, Vol. 210.
12. Brushless VS. Brushed Motors. *KDE Direct.* [Online] https://www.kdedirect.com/blogs/news/brushless-vs-brushed-motors.
13. 5 Reasons Brushless Motors are Better than Brushed Motors. *Celera Motion.* [Online] April 13, 2021. https://www.celeramotion.com/news/articles/5-reasons-brushless-motors-better/.
14. Drone Frames. *Drone Nodes.* [Online] https://dronenodes.com/drone-frame-racing-freestyle/.
15. MAVROS. *GitHub.* [Online] https://github.com/mavlink/mavros/blob/master/mavros/README.md.
16. *pixhawk.* [Online] https://pixhawk.org/.
17. *7-Zip.* [Online] https://www.7-zip.org/.
18. mavROS Installation . *gitHub.* [Online] https://github.com/mavlink/mavros/tree/master/mavros#installation.
19. HolyBro. [Online] http://www.holybro.com/product/pixhawk-4-mini-qav250-kit/.
20. Dronecode. Software Overview. *PX4 Autopilot.* [Online] [Cited: March 1, 2022.] https://px4.io/software/software-overview/.
21. Vicon. Tracker. *Vicon.* [Online] [Cited: 3 1, 2022.] https://www.vicon.com/software/tracker/.
22. GrabCad. [Online] https://grabcad.com/library?page=1&time=all_time&sort=recent.
23. test, testy. *test.* 2022.
24. National Society of Professional Engineers. Code of Ethics for Engineers. [Online] https://www.nspe.org/sites/default/files/resources/pdfs/Ethics/CodeofEthics/NSPECodeofEthicsforEngineers.pdf.
25. *Robust Vision-based Obstacle Avoidance for Micro Aerial Vehicles in Dynamic Environments.* Lin, Jiahao, Zhu, Hai and Alonso-Mora, Javier. 2020.

7 Vicon Motion Capture System Manual

This section provides an overview of the setup and use of the Vicon system.

7.1 Hardware and Physical Setup

Table 9 provides a list of all hardware components used in the physical setup of the Vicon Vero system.

Table 9: Vicon Hardware Components

Component	Quantity
Vicon Vero Cameras	8
PoE (power over ethernet) cables	8
Tripods	8
D-Link DGS-1026MP 26-Port Gigabit Max PoE Switch	1
Ethernet Cable	1
Active Calibration Wand	1
Desktop Computer w/Vicon Software	1

Physical Setup Process:

1. The first step in setting up the Vicon system is placing each camera on a tripod. Each Vero camera can be attached to a tripod with a standard threaded camera mount.
2. Next, each tripod with a camera is placed around the outside of the capture volume (the desired area in which the system will track objects). The goal of this setup is to ensure that an object to be tracked can be seen by multiple cameras from multiple angles anywhere it is placed in the volume. To achieve this the individual tripods can be adjusted to provide different heights and angles. Figure 45 shows a setup that has been used and may serve as a starting point for the purposes of physical setup. The positions of the cameras will likely be adjusted later during the calibration phase.

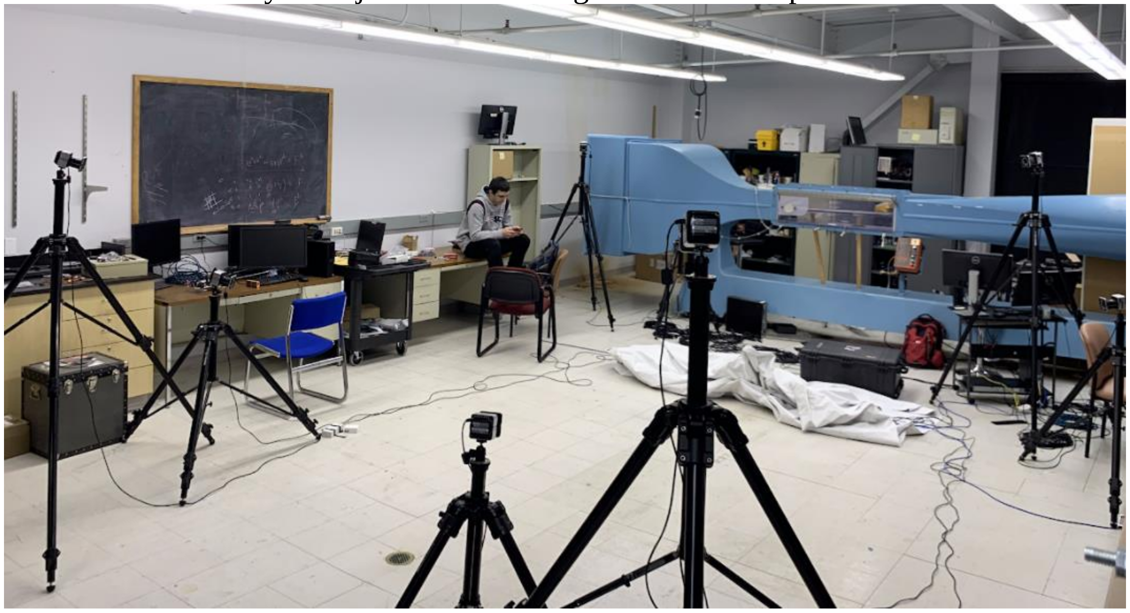


Figure 45: Baseline Vicon Lab Setup

3. Each camera must then be connected via PoE cable to the switch box, making sure that the end with the small box is near the camera as shown in Figure 46.

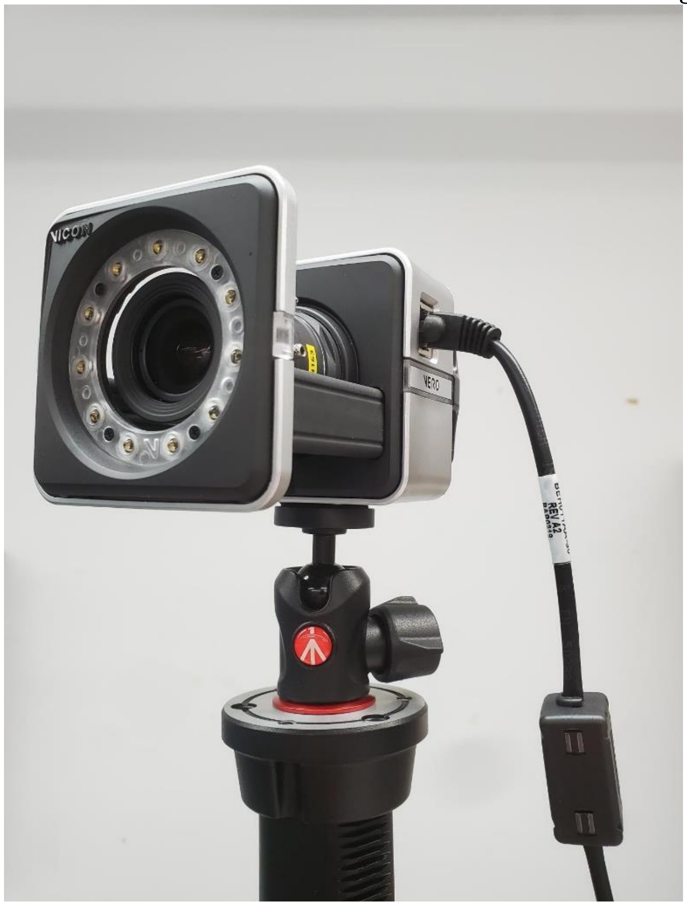


Figure 46: Vicon Vero Camera Connected Via PoE

4. Next, plug each camera into the numbered ports on the network switch in the same order they are physically placed in, starting with a camera connected to port 1, and moving around the perimeter with the last camera in port 8. Connecting the cables in the same order as the cameras are placed may help with future troubleshooting (Figure 47).

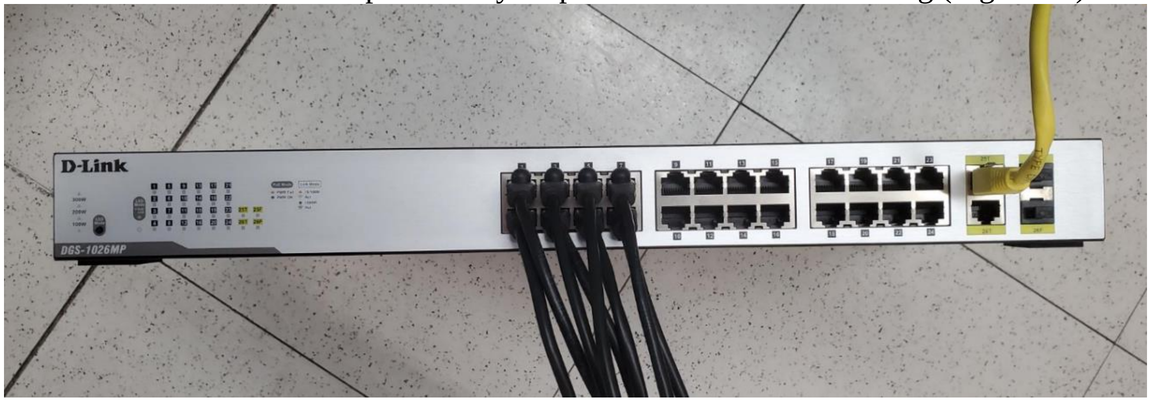


Figure 47: Switch Box with Camera Inputs and PC Outputs

5. From here, the ethernet cable can be connected from one of the output ports to the PC (Figure 48). Note that the green circled ports should be used to connect the switch box and monitor via Mini DisplayPort. The regular DisplayPorts circled in red are connected

to the motherboard, not the GPU and should not be used. The ethernet port circled in red should be used for internet connection, not connection to the Vicon system.

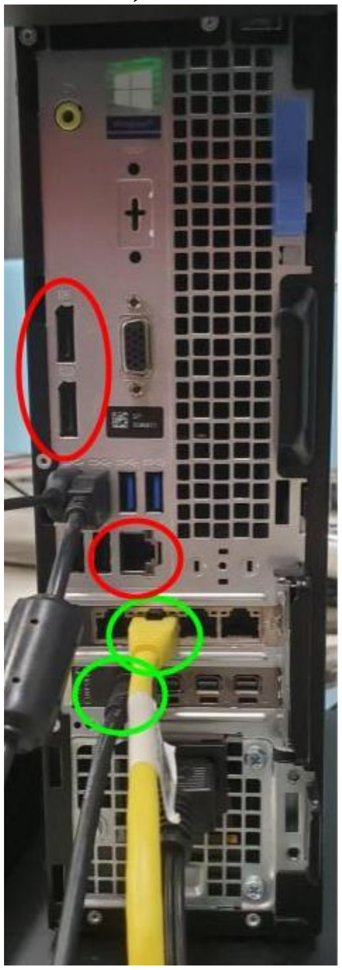


Figure 48: PC Connections

6. The final step is to plug in the power cable for the switch box. Because the Vero cameras are connected by PoE cables, both power and data are transmitted over the single cable between the cameras and switch box. The cameras will automatically boot when powered.

7.2 Software Setup and Troubleshooting

The program used track objects within the motion capture system is Vicon Tracker. At the time of writing, the current release is Tracker 3.9. This software can be downloaded directly from vicon.com. With the hardware in place and Tracker installed, the next step is to make sure all eight cameras are connected to the system and transmitting data. Figure 49 shows a screenshot of the Tracker program with all cameras connected and transmitting, if the boxes or triangles to the left of the camera name are grey instead of green, they are not transmitting data. If this is or any other problem arises, refer to the troubleshooting table later in this section.

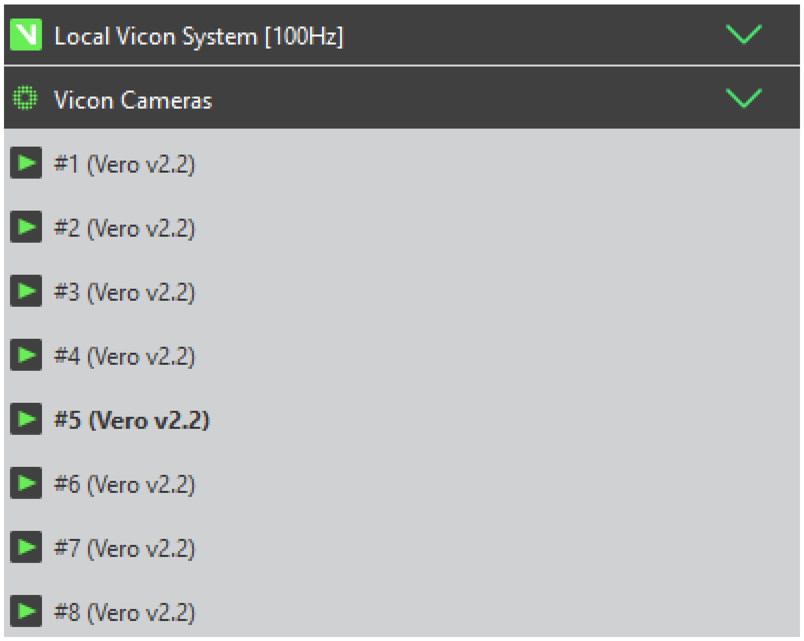
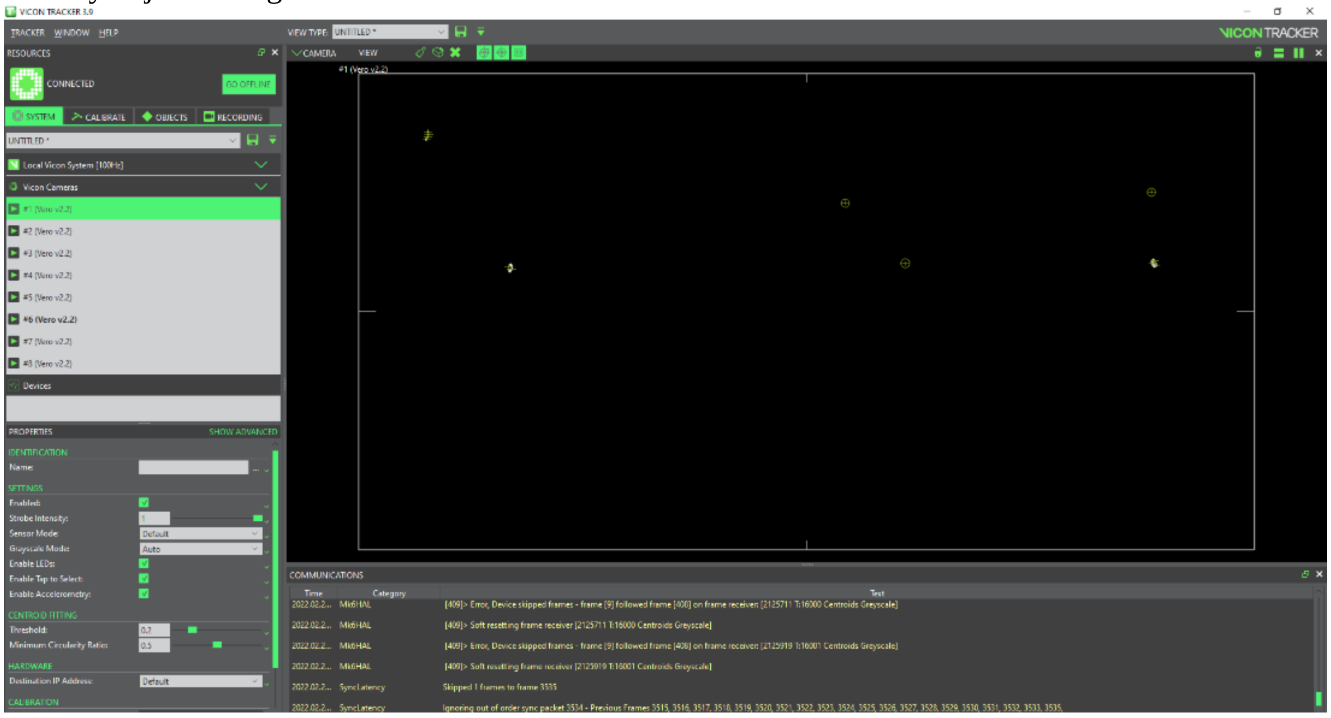

Local Vicon System [100Hz]
Vicon Cameras
#1 (Vero v2.2)
#2 (Vero v2.2)
#3 (Vero v2.2)
#4 (Vero v2.2)
#5 (Vero v2.2)
#6 (Vero v2.2)
#7 (Vero v2.2)
#8 (Vero v2.2)

Figure 49: Screenshot of Vicon Tracker Camera Connections

To additionally verify that the cameras are transmitting data, select either an individual or group of cameras by clicking and dragging over the camera names, and select “CAMERA” from the dropdown menu at the top left of the viewing window. This will display the live feed of what the selected cameras are detecting. Figure 50 shows the view of an unmasked camera in a room with no markers. The yellow symbols indicate detected infrared reflections present in the room. “3D PERSPECTIVE” and “3D ORTHAGONAL” offer 3D views of the positions of the cameras and any objects being tracked.



IDENTIFICATION
Name:
SETTINGS
Enabled: ✓
Strobe Intensity: 1
Sensor Mode: Default
Grayscale Mode: Auto
Enable LEDs: ✓
Enable Tap to Select: ✓
Enable Accelerometry: ✓
CENTROID FITTING
Threshold: 0.2
Minimum Circularity Ratio: 0.5
HARDWARE
Destination IP Address: Default
CALIBRATION

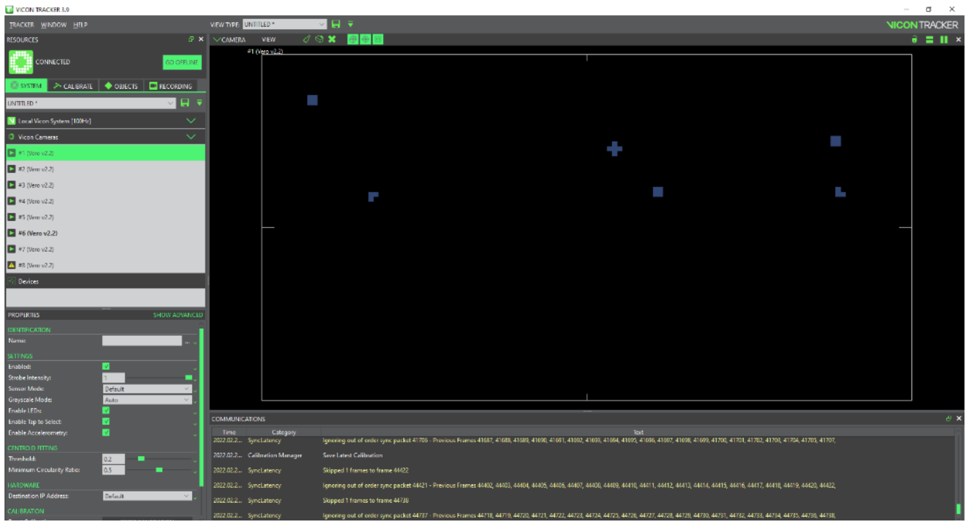
COMMUNICATIONS

```
Time        Category      Text
2022.02.2...  Mk6HAL       [409]> Error, Device skipped frames - frame [9] followed frame [408] on frame receiver: [2125711 T:16000 Centroids Greyscale]
2022.02.2...  Mk6HAL       [409]> Soft resetting frame receiver [2125711 T:16000 Centroids Greyscale]
2022.02.2...  Mk6HAL       [409]> Error, Device skipped frames - frame [9] followed frame [408] on frame receiver: [2125919 T:16001 Centroids Greyscale]
2022.02.2...  Mk6HAL       [409]> Soft resetting frame receiver [2125919 T:16001 Centroids Greyscale]
2022.02.2...  SyncLatency  Skipped 1 frames to frame 3535
2022.02.2...  SyncLatency  Ignoring out of order sync packet 3534 - Previous Frames 3515, 3516, 3517, 3518, 3519, 3520, 3521, 3522, 3523, 3524, 3525, 3526, 3527, 3528, 3529, 3530, 3531, 3532, 3533, 3535,
```

Figure 50: Unmasked Camera View in Vicon Tracker

Masking:

At this point, it will be clear that the cameras are detecting many things in the room that are not reflective markers. This is a result of the system detecting reflections in the room, whether it be objects present in the room, a reflective floor, or even cameras detecting each other. Because detecting these reflections can interfere with tracking objects, it is important to eliminate them either physically, or by masking them in Tracker. If possible, remove the object creating the reflection from the field of view, or cover it with something non-reflective, such as a large canvas to cover the floor. However, it will be impossible to eliminate all reflections, so masking with software is required. To mask, navigate to the “CALIBRATE” tab. Under “CREATE CAMERA MASKS” click START. After several seconds, after any reflections detected by the cameras are masked click STOP Once masked, the camera view will look similar to Figure 51.



IDENTIFICATION
Name:

SETTINGS
Enabled: ✓
Strobe Intensity: 1
Sensor Mode: Default
Grayscale Mode: Auto
Enable LEDs: ✓
Enable Tap to Select: ✓
Enable Accelerometry: ✓

CENTROID FITTING
Threshold: 0.2
Minimum Circularity Ratio: 0.5

HARDWARE
Destination IP Address: Default

CALIBRATION

```
Time         Category              Text
2022.02.2... SyncLatency           Ignoring out of order sync packet 41706 - Previous Frames 41687, 41688, 41689, 41690, 41691, 41692, 41693, 41694, 41695, 41696, 41697, 41698, 41699, 41700, 41701, 41702, 41703, 41704, 41705, 41707,
2022.02.2... Calibration Manager   Save Latest Calibration
2022.02.2... SyncLatency           Skipped 1 frames to frame 44422
2022.02.2... SyncLatency           Ignoring out of order sync packet 44421 - Previous Frames 44402, 44403, 44404, 44405, 44406, 44407, 44408, 44409, 44410, 44411, 44412, 44413, 44414, 44415, 44416, 44417, 44418, 44419, 44420, 44422,
2022.02.2... SyncLatency           Skipped 1 frames to frame 44738
2022.02.2... SyncLatency           Ignoring out of order sync packet 44737 - Previous Frames 44718, 44719, 44720, 44721, 44722, 44723, 44724, 44725, 44726, 44727, 44728, 44729, 44730, 44731, 44732, 44733, 44734, 44735, 44736, 44738,
```

Figure 51: Vicon Tracker Masked Camera View

Notice that the areas masked match the signals detected in Figure 50. Masking will create “dead-zones” in the camera’s field of view where any data points are ignored, including the actual object being tracked. This can decrease tracking performance, especially when many reflections must be masked. Because of this, it is preferable to physically resolve the reflection over software masking. Operating the system in a dim room with any windows or extraneous light sources covered may help reduce reflections.

Calibration:

After masking, the next step is calibration. Several iterations of the calibration process may be necessary to create an appropriate placement of the cameras. Tracker has a robust, automatic built-in calibration that can be initiated by navigating to the “CALIBRATE” tab and clicking “START” under “CALIBRATE CAMERAS”. After initializing the calibration, turn on the LEDs on the Active Calibration Wand and move it throughout the capture volume, aiming to have multiple cameras detecting the wand simultaneously. It may also be beneficial to move the wand throughout the entirety of the capture volume and in different orientations. The goal of calibration is to allow the cameras to detect their locations and orientations relative to each other. To do this, they must be oriented in a manner that any location in the capture volume is visible to multiple cameras. As calibration is occurring, a list of the cameras and the number of frames they have detected is visible in Tracker. Also visible will be live camera perspectives showing the path taken by the wand. This view can be seen in Figure 31 located in Section 3.4. After the predetermined amount of data necessary is collected, the calibration will automatically stop and begin processing. After processing, the calibration will be complete, and any errors will appear in the program. If the cameras are not placed correctly, Tracker may be unable to mesh the different views into one 3D space. If this happens, take note of which cameras are not linked with each other and adjust their physical locations to allow for more field of view overlap between them.

After calibration, the motion capture system has determined the relative position of all 8 cameras and the object it is tracking, but it has no reference to the space that it is in. As such, the virtual orientation of the cameras in Tracker may not match the real-world setup. To fix this, place the Active Calibration Wand in the center of the capture volume and turn on the LEDs. Then, click START under "SET VOLUME ORIGIN". This will set the origin and axes directions of the capture volume based on the orientation of the wand as the cameras view it.

Creating Objects:

To track a specific object, it must be registered in tracker to allow capturing the object as a whole, rather than the single point of a marker. This also allows orientation data to be produced. To register an object, first attach reflective tracking markers to it so that it can be viewed by the cameras. Technically, only three markers are necessary to track an object, but using 4 or more is advisable, as markers can become occluded from the cameras by the object itself. Too many markers in a small area, however, can create noise that interferes with tracking. Some experimentation with number and placement of markers may be necessary. For the case of this project, four markers, with three placed on various parts of the motor arms and one placed on top functioned well. It is also important to make sure that the markers are placed asymmetrically, so that the object's orientation does not unintentionally become inverted while tracking. Next, place the object in the center of the capture volume such that all cameras can view the markers. Navigate to the OBJECTS tab. Each marker of the object should be visible in the 3D PERSPECTIVE view. While holding the ctrl key, click each marker so that each is selected. On the left side of the screen next to "Create Object", type in the name of the object and click CREATE. Now, Tracker should recognize the object and the visual appearance in the 3D perspective view will be similar to Figure 52 where the markers are treated as a singular connected entity with an orientation denoted by the axes shown. Once objects have been created, displaying/tracking them can be toggled at any time with the checkboxes in the OBJECTS tab. While tracking an object instead of individual markers, Tracker can provide real time position and orientation data of the object via a software development kit.

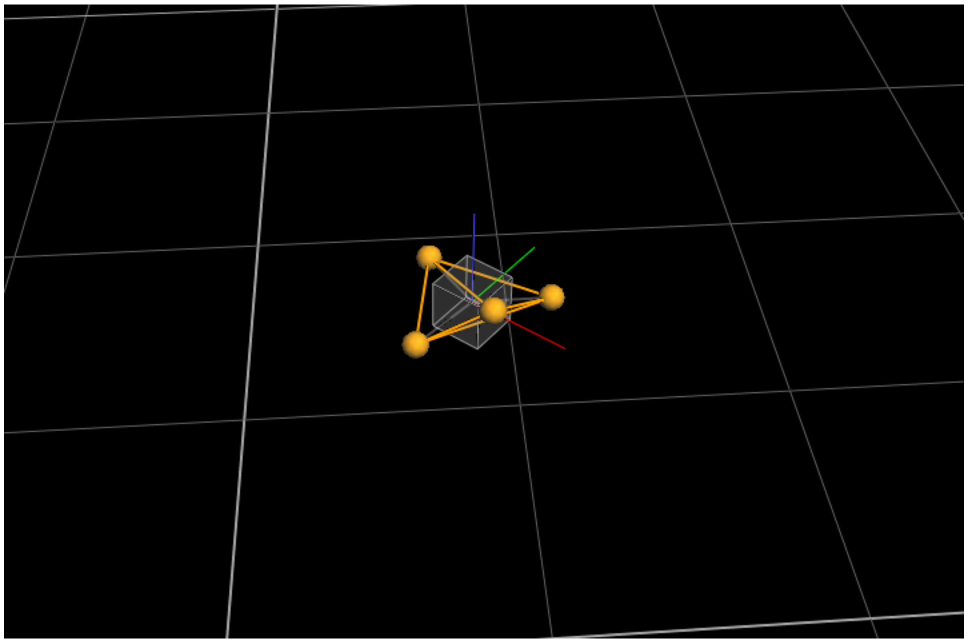


Figure 52: Object in Vicon Tracker

7.3 MATLAB SDK Integration

Vicon provides a software development kit (SDK), Vicon Datastream SDK, for the integration of Vicon Tracker software with a variety of third-party programs. For this project, the SDK was used to stream live data from Vicon tracker to MATLAB. For up-to-date information refer to the Vicon Datastream SDK developer's manual found on vicon.com. This manual contains extensive information on the installation of the SDK and functions that are included. Much more information than that used for the basic tests in this report is available.

7.4 Troubleshooting

Many troubleshooting issues involved in the setup of the motion capture system originate in PC setup, often because of incorrect settings or privileges. Below is a list of some obstacles encountered by our team, and how they can be resolved. Many of these solutions require administrator privileges to the PC to enact, it may be necessary to contact whoever is responsible for managing such privileges. Another thing to note is that these troubleshooting steps have all been written with Windows 10 in mind, if the system is updated to Windows 11 in the future, the steps may be different. If the recommended solutions are unable to solve the problem, try contacting Vicon Support directly, they are quite responsive via email and are incredibly helpful with troubleshooting issues. At the time of writing, they can be reached at support@vicon.com.

Obstacle	Solution
Cameras are visible in Vicon Tracker Software but do not transmit any data	First try disabling the Windows Defender Firewall as seen in the solution below this one. If that does not work, this may be caused by incorrect network card settings. Navigate to edit the following settings: Control Panel → Network and Internet→

	Network and Sharing Center→ locate the unidentified network that is from the switch box → Properties → select Internet Protocol Version 4 → Use the following IP address: Set the IP address to 192.168.10.1 and the subnet mask to 255.255.255.0. Navigate to the advanced configuration settings and verify the following settings: From the Unidentified Ethernet port page: Properties → Configure→ Advanced: Jumbo Frames/Packets: enabled and set to 9014 Receive Buffers at its max Transmit Buffers at its max Receive Side Scaling is on Max RSS Queues at its max Interrupt Moderation is off/disabled Interrupt Moderation Rate is off/disabled
Cameras are not detected by PC, even when plugged in and powered on	You may need to disable the Windows Defender Firewall for Vicon Tracker to allow the system to transmit the data through the network. Open Control Panel and navigate to Windows Defender Firewall. Click on “Allow an app or feature through Windows Defender Firewall” and select Vicon Tracker. Enable the correct network and click OK.
Vicon Tracker program crashes upon clicking “Calibrate”	This is likely caused by a system setting involving the Vicon Tracker’s access to the dedicated GPU (graphics processing unit) in the PC. 1. Ensure that the monitor being used is plugged into the dedicated GPU instead of the motherboard. 2. In the Windows settings app go to System > Display > Advanced Graphics Settings. Choose Vicon Tracker from the pulldown menu. Click “options” and choose the option

	for high performance which will set the dedicated GPU as preferred. 3. Go to program files folder Find the folder labeled Vicon→ right click on the Vicon folder→ Properties→ Security → Edit→ Click on users or any group that you want to give access to→ Check the full control box in the allow column→ Apply→ Ok→ 4. Install the latest graphics card driver from the NVIDIA website
Single camera becomes disconnected while in use (displays red LED)	Manually unplug and reinsert the PoE cable connected to the camera
Unlisted Issues	1. Make sure the latest version of tracker is being used, if not, install it from vicon.com. 2. Ensure that the most recent drivers for the graphics card are installed, even if Windows states that the most recent NVIDIA drivers are installed, check the NVIDIA website directly and manually install them if needed.