Arcus: A UAV for Planetary Exploration

MRSD Team B

15 December 2016

Clare Cui
Maitreya Naik
Angad Sidhu
Logan Wan

1. Abstract

This report details the progress on the autonomous, multi-modal mapping UAV project started during the Fall 2016 semester. The UAV will be used to test autonomy algorithms that make use of multiple onboard sensors for mapping, localization, and data collection. This application is further explained in the project description and use cases sections. The system-level requirements are then defined and accompanied by the functional and cyberphysical architectures that construct the system, fulfilling the requirements. The current system status is then explained in detail by focusing on each of our subsystems - software, mechanical, and electrical - which validates specific system requirements given in the fall validation experiment (FVE). We are on schedule and have mostly integrated the mechanical and electrical subsystems, allowing us to focus primarily on the software subsystem next semester. The project management section of this report shows the plans highlighted for Spring semester, including important milestones and what we are planning for the spring validation experiment. The largest goal we need to work towards is autonomy, along with fusing sensor data and improving our mapping algorithm. Finally, conclusive remarks are given about the work we have accomplished and the work we have ahead.

Table of Contents

2. Project Description

NASA is planning an exploratory mission on Mars to probe cave networks for information on their structures and the mineral composition on the surface layers of rock and sediment. These findings can further inform our scientific understanding of Mars and its suitability as a location for human inhabitation. However, it is extremely costly to send humans to scout and look for desirable landing locations and areas of interest for future scientific missions. Arcus is a UAV (unmanned aerial vehicle) that will generate real-time map data to give scientists and users the critical information necessary to make informed decisions on how to allocate precious resources when on Mars. A big part of operating on another planet is the lack of absolute positioning via GPS. Arcus will combine LiDAR (light detection and ranging), RGB (red green blue), along with hyperspectral (non-visible light) imaging sensors to create data-rich maps that will highlight specific objects of interest fused with spatial geographic information all without the aid of GPS. The final vehicle will be designed to fly inside of, discover, and autonomously map fully enclosed, non-uniform, gps-denied environments.

In the scope of the MRSD project, Arcus will be a remotely piloted UAV that will generate this map information in real-time. It is intended to be used as a rapid prototyping platform for developing navigation and planning algorithms, as well as providing basic software and hardware modularity to accommodate different types of imaging sensors. The platform can navigate by tele-operation or autonomously. The platform and the software provided allow researchers to easily hook into ROS to retrieve point clouds and a textured mesh map describing the environment. Researchers would then develop algorithms which analyze the map and make decisions regarding future behavior. For example, if a branch of a cave seems to tighten in diameter the robot might make the decision to ignore that route to explore because it might be too small to fly through. However, if the entrance to that branch is coated in rich minerals that have not been observed before, the robot might make the decision to traverse down that branch to gather more information. In short, the purpose of the robot is to perform SLAM in a GPS-denied environment and collect structural and RGB color information with low error in both its state estimate and its map.

3. Use Case

Prior to a fully manned mission to Mars, NASA will send out robotic scouting missions to determine both the suitability of habitation and a potential landing spot for a human habitat on the planet. Similar missions are planned for the Moon, where scientists are trying to uncover the actual quantity and location of water around unique geographical landmarks. A mapping UAV will be prepared for the unique requirements of each mission. A moon UAV will likely be propelled by rocket-powered thrusters, whereas the Martian atmosphere would accommodate propeller-driven flight. Sensors such as spectrometers, hyperspectral imagers, and distance ranging equipment would be loaded for the detection of various molecules, water, or precious resources. Mission parameters will be loaded onto the vehicle, such as the general location of interest coordinates and specific features to search for.

Upon delivery as a rocket payload to the terrestrial surface, the UAV will deploy from its base station. A rough map generated from an orbiting observer as well as simulations and calculations will give us a rough estimate of landing location. The UAV will then plan a path from the landing zone to some objective point. While en route, due to communication latency, the UAV will have to make decisions that will optimize its battery life to focus on successfully completing mission objectives. As seen in Figure 1 below, real-time fusion of map structural data with hyperspectral imaging data will allow for the drone to provide the researchers a map that would allow the researchers to quickly identify specific features to follow up with higher-resolution data capture. After a predetermined amount of battery consumption, the UAV will return back to the landing site for high-bandwidth data communication back to Earth. Using these high-fidelity data, scientists will then be able to better inform their hypotheses about the characteristics of the celestial body, and much more quickly determine its suitability for human habitation or resource acquisition.

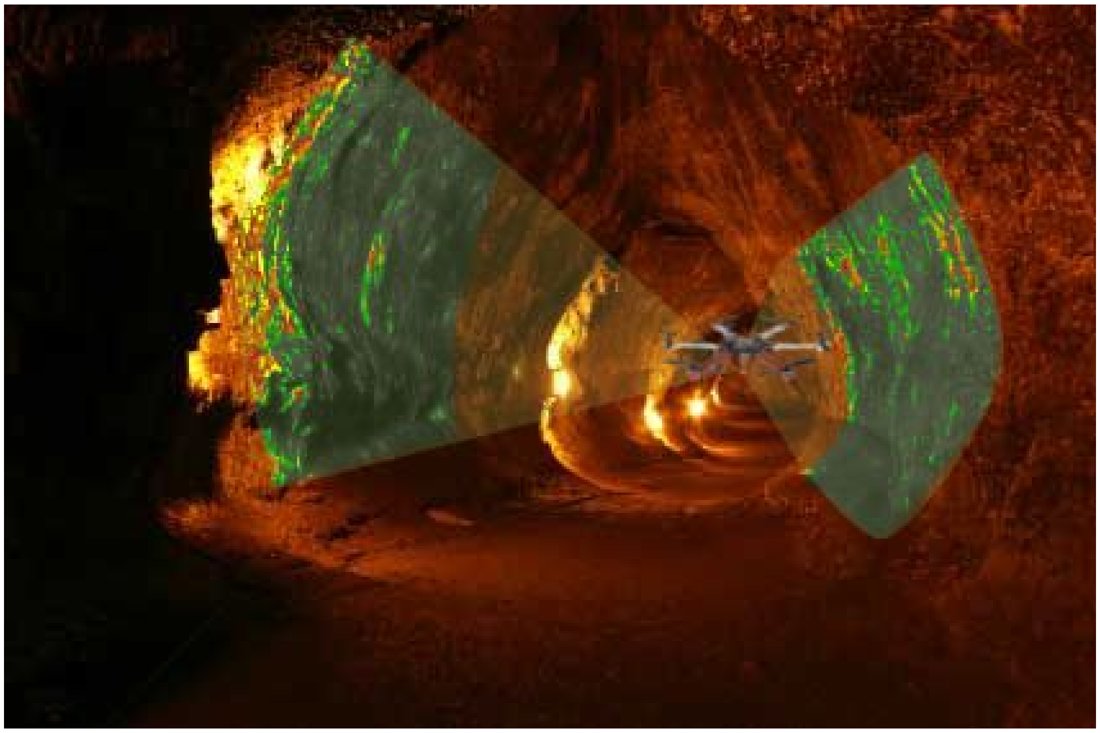


Figure 1: Site operation mock-up

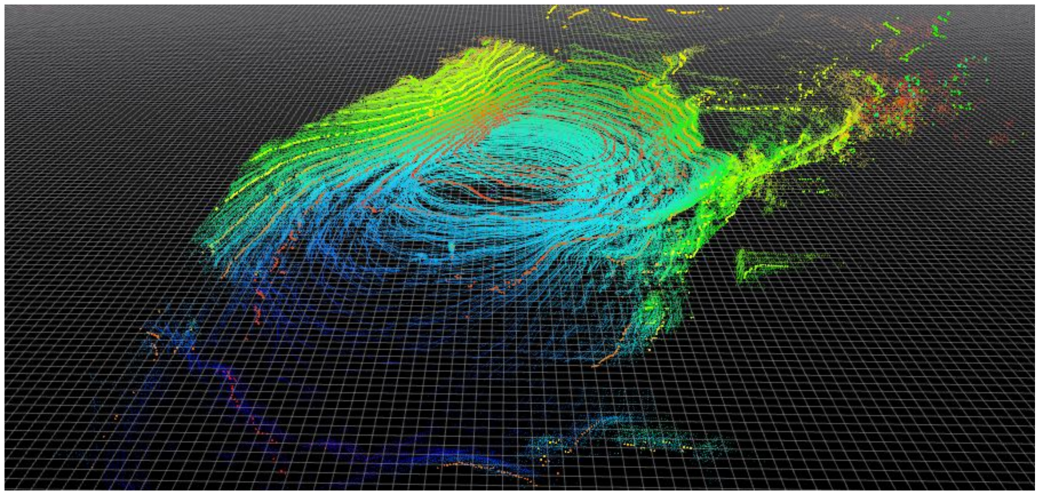


Figure 2: Example of a point cloud of data at LaFarge Quarry.

4. System-level Requirements

4.1 Mandatory Performance

MPR1	Fly at a minimum speed of 0.25 m/s
MPR2	Generate a map with a voxel size of at most (50 cm)^3
MPR3	Be capable of storing a minimum of 2 imaging modalities in map
MPR4	Map 50,000 m^3 in at most 15 minutes
MPR5	Localize accurately so that drift in pose is at most 1 meter / meter traveled
MPR6	Map with errors less than 10%
MPR7	Update map with fresh sensor data at 1 Hz
MPR8	Provide map back to user at least 0.5 Hz
MPR9	Be tele-operable at a range of at least 20m

4.2 Mandatory Non-Functional

MNF1	Must have enough battery to operate for 15 minutes
MNF2	Additional sensors should be easy to integrate into software.
MNF3	Battery should be easily accessible for hot swapping on successive runs.

4.3 Desirable Performance

DPR1	Safely land and takeoff at 0.3 m/s
DPR2	Wirelessly controllable up to 100m distance from user
DPR3	Wirelessly transmit maps and video data up to 100m distance from user
DPR4	Data point position resolution of at least 15 cm
DPR5	Should be able to fly up to 100m relative altitude

4.4 Desirable Non-Functional

DNF1	Additional perception sensors can be easily mounted.
DNF2	User base station can be easily transported
DNF3	Vehicle should be easily transportable
DNF4	Vehicle operation process should be minimally complex to minimize startup time
DNF5	Vehicle computer should be powered without draining main vehicle battery

5. Functional Architecture

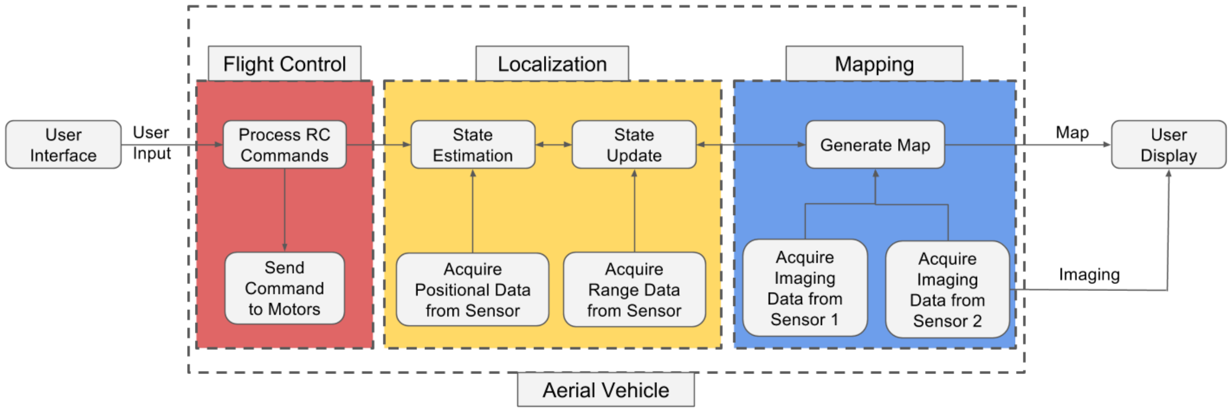


Figure 3: Functional architecture diagram

The functional architecture for this project is visualized in Figure 3. Because the aerial vehicle will be controlled through teleoperation, its sole input will be from the user, who utilizes a physical interface to direct the vehicle. For autonomous navigation, the user input will be a list of desired waypoints that the planning software will parse into vehicle movements to the flight controller. The expected output will be a real-time mapping of the environment that the vehicle is traversing through. The raw sensor data will also be provided to the user, for later post-processing and further analysis.

Once the vehicle receives remote input from the user, it processes the commands, adjusting the speed of the motors to travel at the specified velocity. Simultaneously, the robot conducts state estimation, keeping itself stable in the air from positional data acquired from a sensor. After knowing its approximate state, the robot is then able to take in imaging data from different sensors to perceive its environment and update its state estimate. At a minimum, there will be two mounted sensors whose data will be fused together to provide a 3D map. Additionally, images from one of the RGB cameras will be sent directly to the ground control station to aid the user in manual flight of the UAV. The imaging sensor-fused map will be sent back to the user, updating frequently such that the user is always able to see where the robot is in space.

6. Cyberphysical Architecture

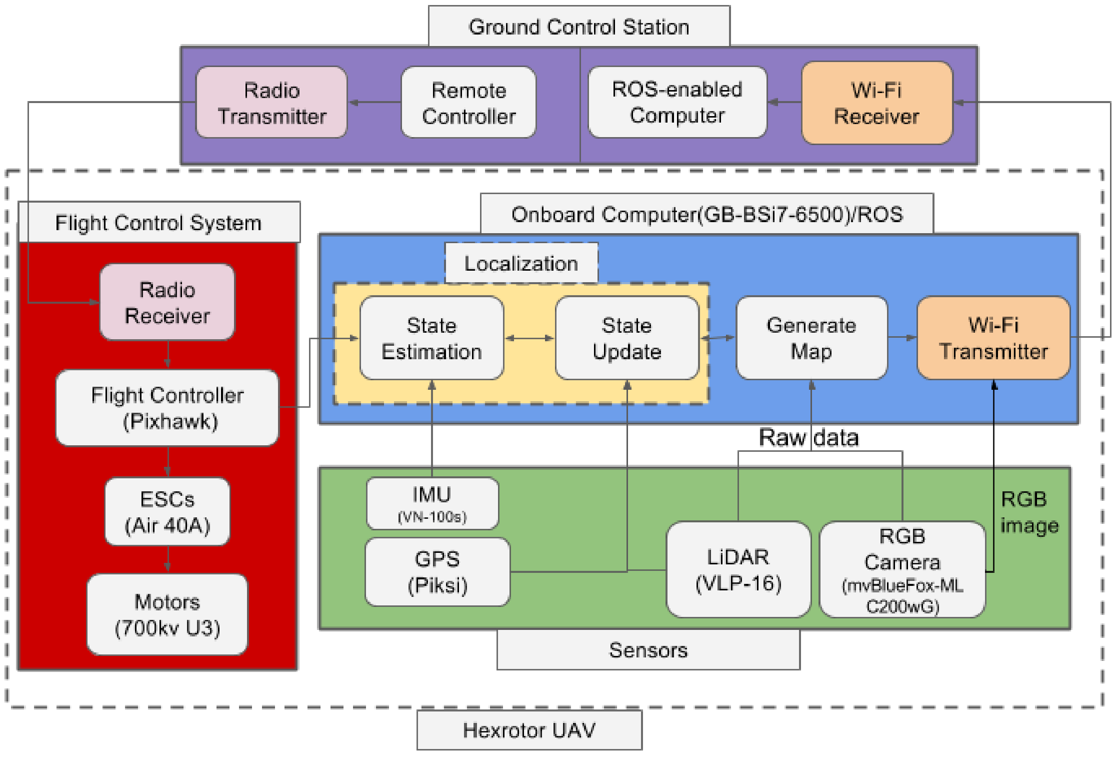


Figure 4: Cyberphysical architecture diagram

Shown in Figure 4 is the cyberphysical architecture diagram for the project. The User Interface from the Functional Architecture changes into a Ground Control Station. The Ground Control Station and the hexrotor are linked on a Wi-Fi ROS network hosted on the Hexrotor.

A 2.4 GHz remote controller at the Ground Control Station sends commands to the receiver, which communicates with the flight controller, in our case the PX4 Pixhawk [3]. The flight controller processes these commands and sends them to electronic speed controllers (ESCs [4]), which engage and control the motors [5] and connected propellers, enabling flight. At the same time, the onboard computer is performing state estimations to determine the pose and orientation of the robot, which is informed by an inertial measurement unit (IMU). The RTK (real time kinematic) GPS is solely for ground truth comparison and is not used for state estimates and updates. The state is updated with information from point clouds generated from the LiDAR (light detection and ranging), which localize the robot in its surroundings. As the robot traverses through the air, it will generate a map, which is a composite of the LiDAR and RGB sensor data. The images from the RGB cameras along with maps that are generated by the onboard computer will be sent through the radio transmitter over 5 GHz radio back to the receiver on the ROS network. This map is displayed to the user at the ground control station.

7. Current System Status

7.1 Fall System Requirements

Table 1 lists the requirements that were demonstrated for this fall semester validation experiment. Requirements MPR2, MPR4, MPR5, and MPR8 are software requirements while MPR1 and MPR9 are hardware requirements. An emphasis was placed on completing the physical platform and hardware integration to ensure that data could be collected with a fully integrated device with spare time to revise any potential issues encountered.

Table 1: Fall Requirements Validated

#	Description	Requirement
1.2	Provide map back to user at a frequency of at least 0.5 Hz.	MPR8
1.3	Generate a 3D map with voxel resolution of at most 50 cm^3. Map volume of basement in less than 20 minutes.	MPR2 MPR4
2.1	Fly at a speed of at least 0.25m/s. Be tele-operable at a range of 20m.	MPR1 MPR9
2.2	Drift should be less than 1 meter per meter travelled.	MPR5

7.2 Overall System Depiction

The overall system status can be seen as shown in Figure 5.

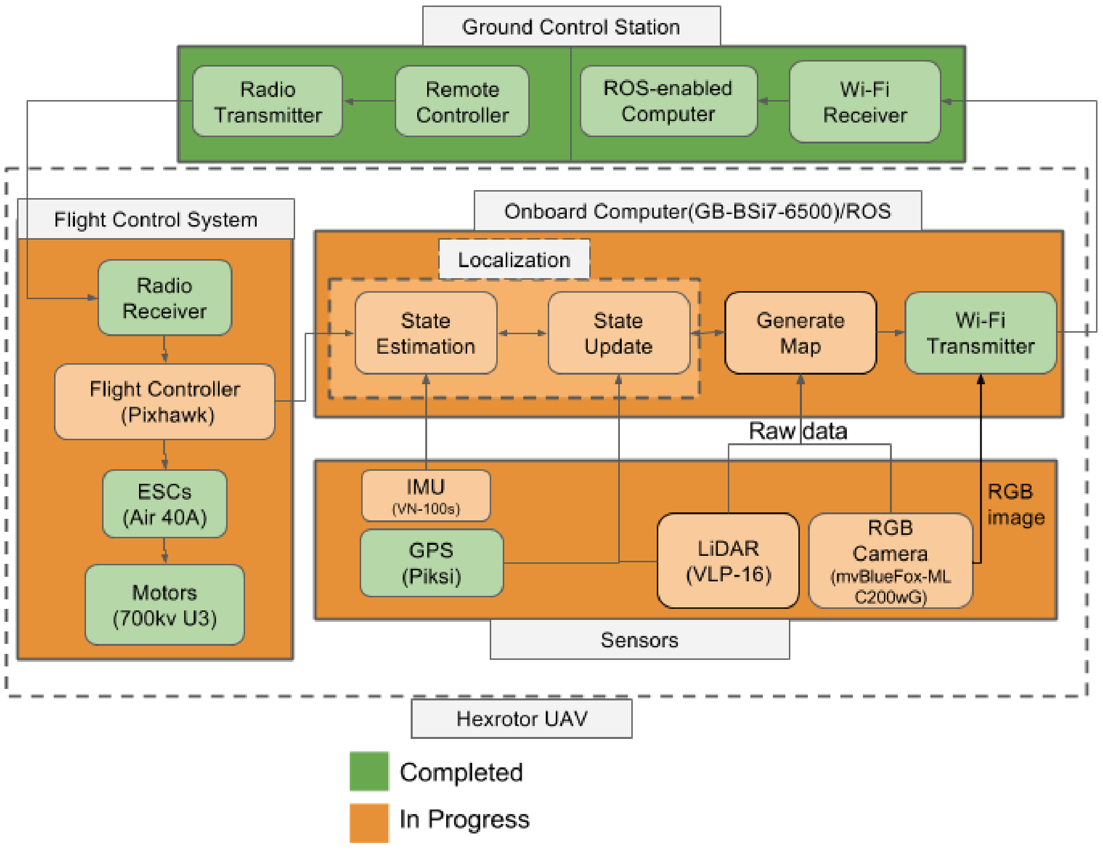


Figure 5: Current overall system status

7.3 Software Subsystem

The software subsystem status aligns closely with our fall validation experiment (FVE) requirements. Figure 6 below describes the flow of data from the sensors to subunits and represents a high level overview of the majority of work that was performed on the software subsystem.

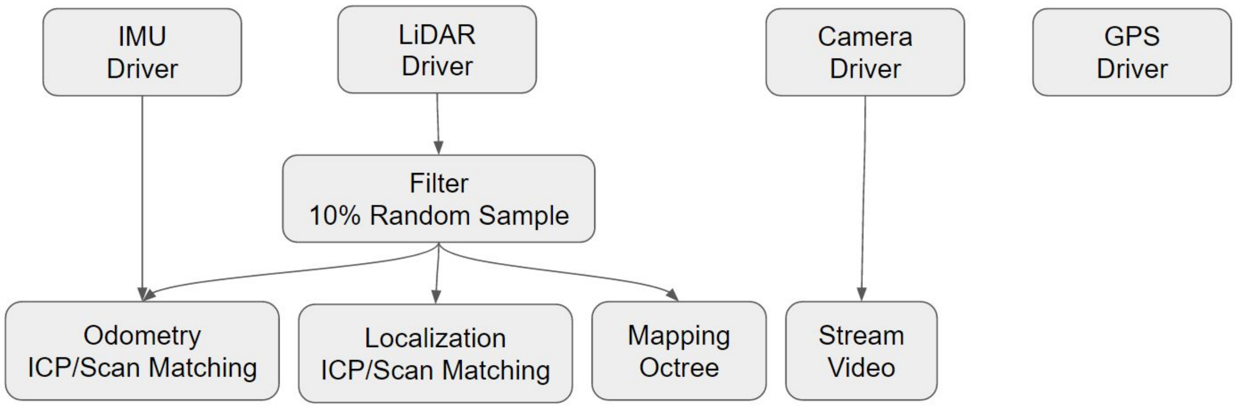


Figure 6: Data flow of sensor data to functional subunits

At the lowest level there are drivers for four sensors: IMU, LiDAR, Camera, GPS. These drivers were acquired online through public repositories. The LiDAR and Camera drivers are fully functional and do not require maintenance. The IMU and GPS driver, however, are no longer maintained and do not provide all of the necessary and desired functionality for our use case. It is possible that modification may be necessary in order to expose a larger subset of each device's functionality and to increase robustness.

The major function depicted in this graph is the SLAM algorithm. The dataflow for the algorithm starts with raw point cloud data originating from the LiDAR. These point clouds are throttled through a randomly sampling filter and reduced to 10% of its original size. This is to reduce processing time for later steps in the algorithm and allows the system to more ably meet the requirement MPR4. The odometry package then performs iterative closest point (ICP) with both the filtered point cloud and an initial orientation estimate from the IMU. This provides a state estimate which is then further refined by scan matching and updating the vehicle state relative to the generated map. Then the point cloud is projected into the map given the state estimate. The map is currently structured as an Octree where each voxel contains points that have been collected during scans. The minimum voxel size can be configured to a value and is currently set to 5 cm. This validates the requirement MPR2. In order to reduce the bandwidth of transmitting the map, an incrementally built map can be utilized. This means that only the new points added to the map are transmitted, which allows for the system to meet requirement MPR8. Another function depicted is the ability to stream the videos we are capturing. This is simply a republisher node which encodes RAW images from the camera into a Theora stream reducing network bandwidth needs from 60 Mbps to 300 kbps.

Aside from the software mentioned here we have also explored various calibration softwares like Kalibr as well as writing our own package for LiDAR-Camera calibration. We are able to get reprojection error for camera intrinsics down to about 1 pixel. However, remaining calibration results still require further testing and tweaking to get more accurate values.

In addition to the actual architecture and calibration software we have been using, various bits testing and setup software has been developed. For example, we have developed odometry testing scripts that enable us to quickly measure the performance of our algorithm. We also developed launch scripts for various different configuration of our sensor array in order to speed up testing and development as it can eat up time and battery life in the field to start up everything manually.

7.4 Mechanical Subsystem

Our mechanical subsystem has been fully integrated as of end of fall validation. All sensors (LiDAR, RGB camera, IMU) were mounted underneath the chassis in a custom 3D-printed undercarriage that also held the batteries. The CAD models for this with mounted sensors can be seen in Figure 7 below. It was necessary for the LiDAR to hang at the bottom of the chassis in order to get a full view of the environment. Although the view is partially obstructed by the legs of the UAV, this has not proven to hinder our mapping capabilities. The RGB camera is angled downwards so that its field of view intersects with the LiDAR to make sensor data fusion feasible.

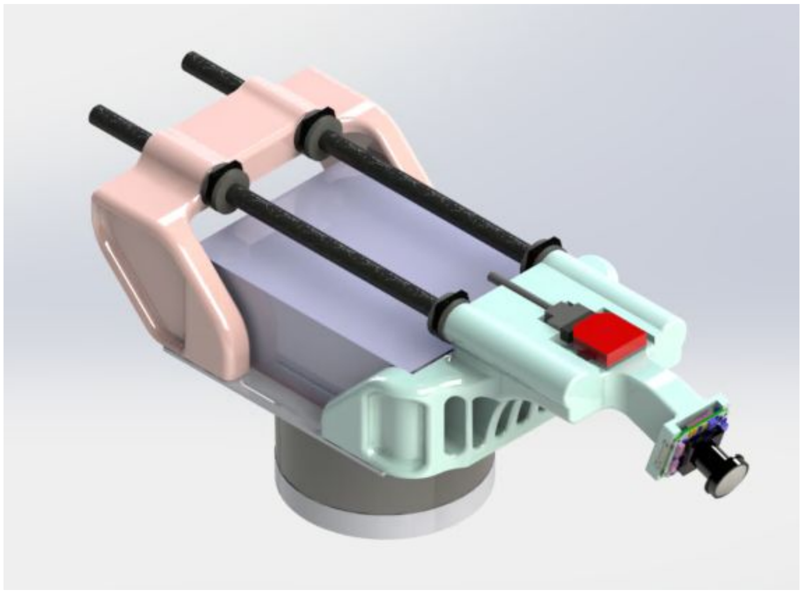
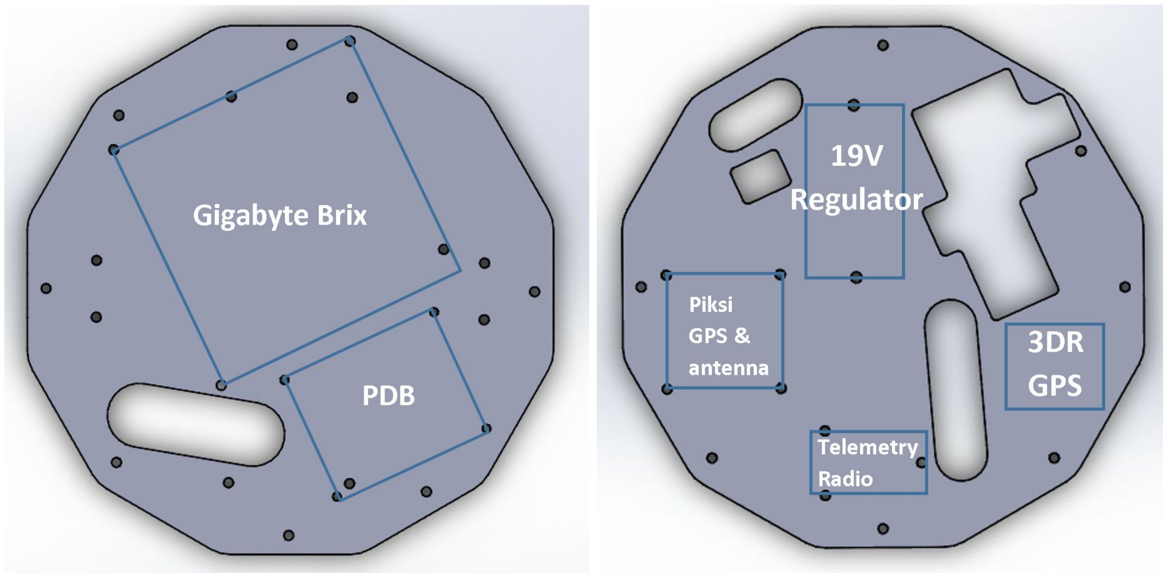


Figure 7: Rendering of the mounted sensors and batteries with the custom mounts.

For mounting the rest of the hardware components, it was necessary to make custom mounting plates that were carefully planned for space efficiency and ease-of-access. There were three levels to the upper chassis. Starting with Platform 1 as the base, the components mounted at each level can be seen in Table 2 below. The laser-cut plates were made of Delrin, and the cutting pattern can be seen in Figure 8. Because the first layer was included with the rest of the chassis, only platforms 2 and 3 needed to be made.

Table 2: Organization of components for each level of the top subassembly.

Platform 1	Pixhawk flight controller
	RC receiver
	FPV telemetry radio
	Switch
	Buzzer
Platform 2	Gigabyte Brix
	Power Distribution Board (PDB)
Platform 3	Piksi GPS
	Piksi GPS antenna
	Piksi telemetry radio
	3DR GPS antenna
	19 V voltage regulator



(a) **(b)**

Figure 8: (a) Platform 2 and (b) Platform 3 that show the locations of parts to be mounted.

Since there were several expensive and heavy pieces of hardware being mounted on the UAV, we decided to do a test flight to ensure the propellers and motors were capable of handling the total system payload. The components not critical for test flight were dismounted and individually weighed. The LiDAR, camera, IMU, and cabling for the bottom half of the chassis weighed 930 g. The top half which included the computer, power distribution board, delrin plates, GPS, and cabling was 750 g. We fabricated a bottom dummy weight from a block of aluminum and sourced weights for the top dummy payload to create a system that weighed, in total, 5.35 kg. This system can be seen below in Figure 9. Overall, the flight test was successful and was able to help us validate MPR1 by proving that our system was capable of flying with its total payload. The minimum speed was proved out in future test flights. After this, we were able to move forward with flying our full system with all the mounted sensors and hardware.

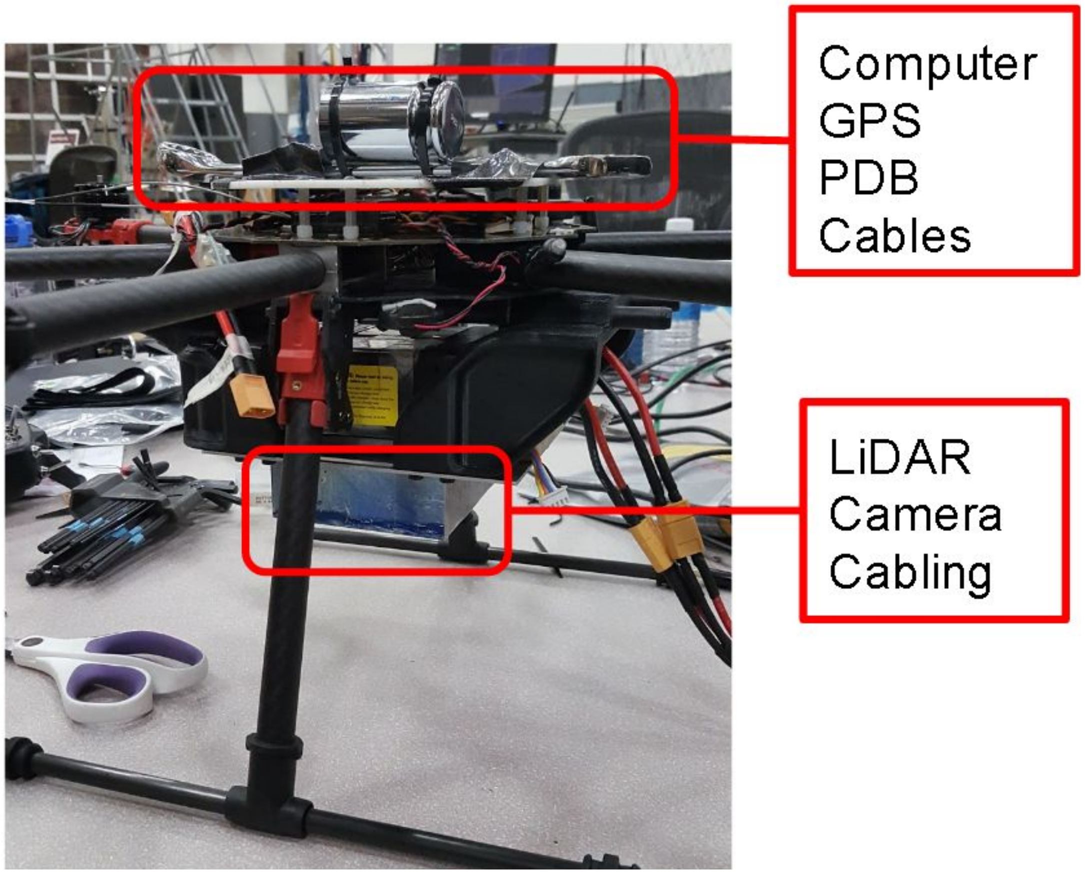


Figure 9: UAV with dummy payload for experimental flight test

The current UAV with completed hardware integration can be seen below in Figure 10.

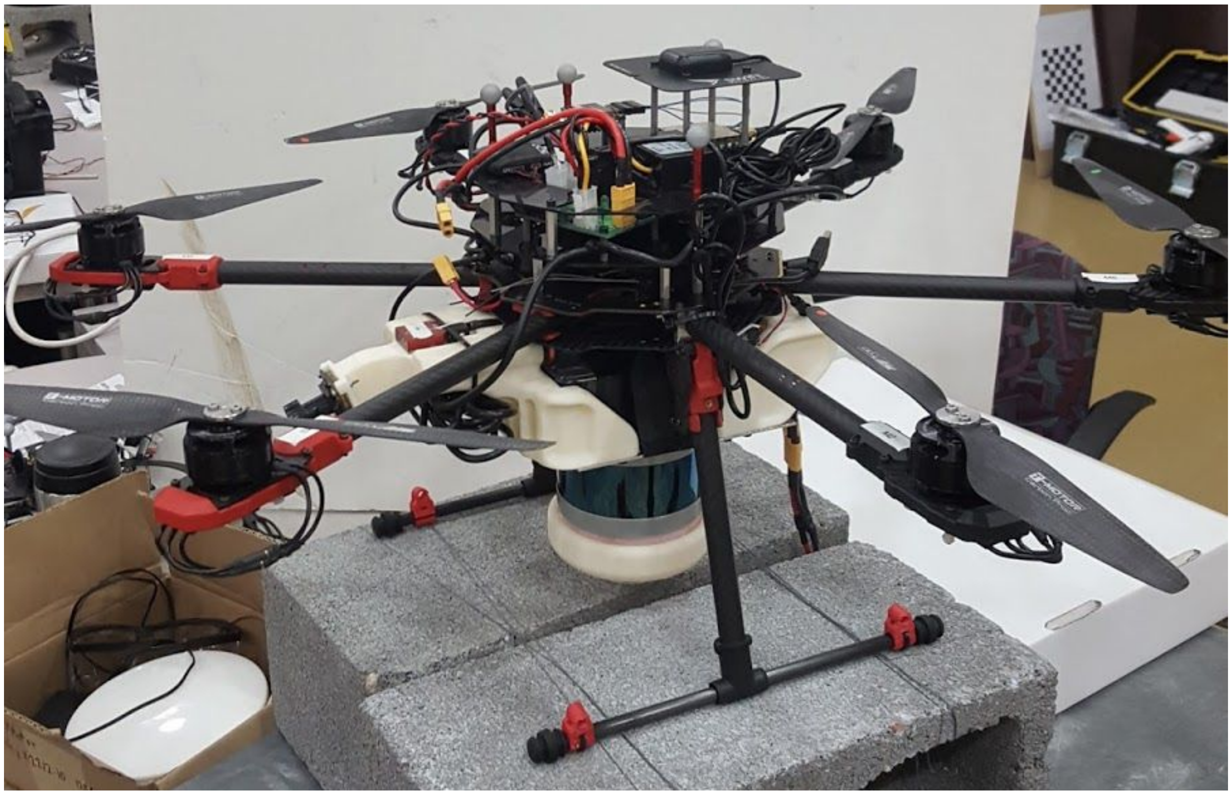


Figure 10: UAV with all sensors and hardware mounted

7.5 Electrical Subsystem

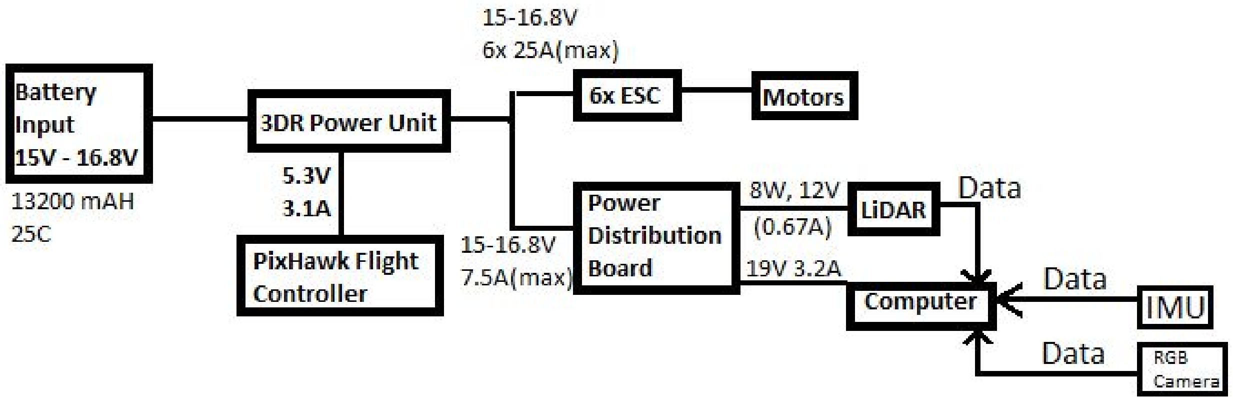


Figure 11: Electrical System

Our current electrical system diagram is depicted in Figure 11 above. The Power Distribution Board (PDB) was designed to power the computer and LiDAR off of the battery power and involves voltage regulation and circuit protection elements, such as overcurrent, overvoltage, and reverse voltage situations. We also made a custom cable for powering the LiDAR and interfacing the computer. Table 3 below shows the result of the PDB test.

Table 3: Power Distribution Board Test Results

Device	PDB Test Input Voltage (V)	Board Input Current Capacity	Observed Max Input Current	Board Rated Output Voltage	Observed Output Voltage
Gigabyte Brix [2]	15	5A	0.55A (Ideal)	19V	18.8V
			1.7A (Startup)		
			1.7A (OpenCV Compile)		
Velodyne VP-16 [1]	15	2.5A	0.68A	12V	11.65V
Brix + VP16	15	7.5A	2.5A	As above for each device	As above for each device

A couple of challenges faced were:

- The 3DR power module and the PixHawk reset the power going to the right of the 3DR module if the motors collide with an obstacle. This reset also restarts the computer and may cause harm to its memory. Hence we have made a revised circuit diagram, as seen in Figure 12, where we have separated our PDB from the 3DR Power Module path.

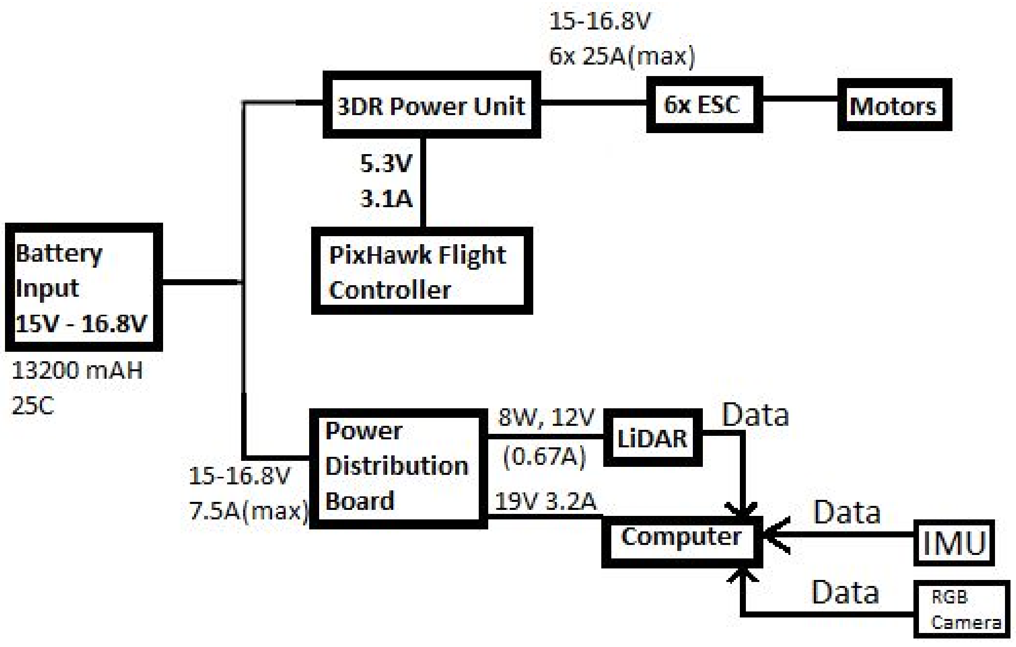


Figure 12: Revised Circuit Diagram

- Due to lack of switches in the power lines, we had to unplug the batteries once we were ready to power down the system. Hence, the electrical system will be revised and 5 switches will be added:
    - Switch 1: At Battery terminals
    - Switch 2: Between Battery and 3DR Module
    - Switch 3: At Battery input of PDB
    - Switch 4: At LiDAR output of PDB
    - Switch 5: At Computer output of PDB

7.6 Performance Evaluation

Our system performed much better than expected in terms of performance characteristics of the SLAM algorithm. Its error at the end of the run came out to be 0.04m/m traveled while running in real time. This validates requirement MPR5 and MPR4. Figure 13 below shows the entire drift over distance traveled curve. It starts out high and decreases due to significant jitter introduced in odometry laser scan matching averaging out over a large distance traveled. This jitter is something we will need to investigate further in order to reduce overall error both in the map and in the state estimate. Figure 14 below shows the total drift over time which ends up being 4.2 m total over the entire run.

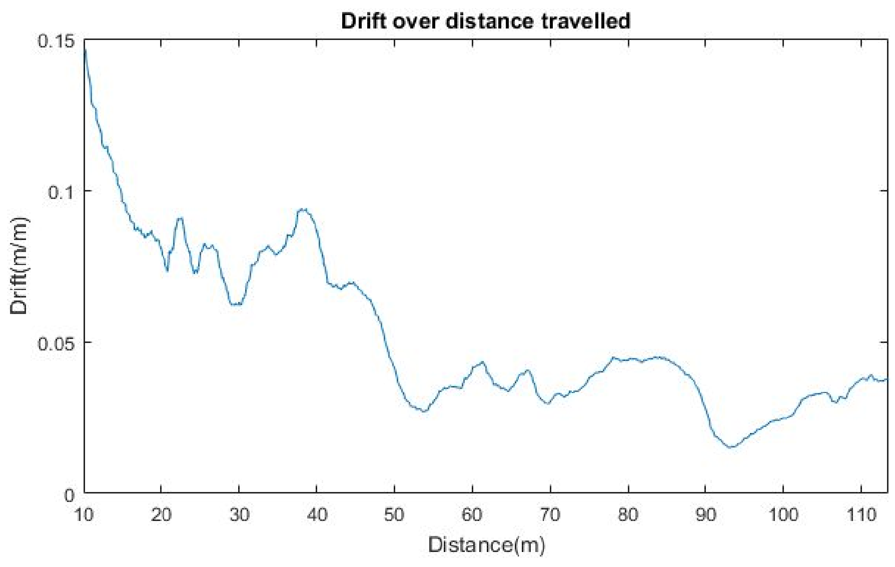


Figure 13: Drift over Distance Traveled

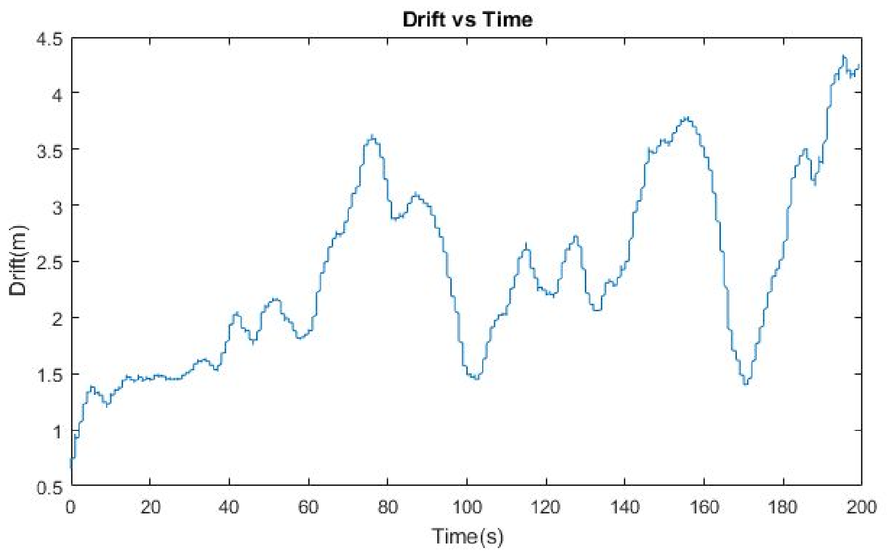


Figure 14: Drift over Time

By design we met requirement MPR2. As explained in the software subsystem description, the map is based on an Octree which has a configurable minimum cell size. We met MPR8 by demonstrating that our base station was receiving updates to its map at frequencies of around 10 Hz at close range. MPR1 and MPR9 were validated by looking at odometry and calculating distance traveled and velocity.

7.7 Strengths and Weaknesses

Our biggest strength is our timing on schedule. We are ahead of schedule in having a SLAM framework that works reasonably well and operates in real-time. We had initially considered this a risk which would require a lot of work to reduce its algorithmic complexity. It, however, is still a risk as we may need to invest more time in this once we address some of the other issues in SLAM. Another strength is we essentially have a first version prototype with most of the functionality of our final robot for spring. This gives us another semester to revise hardware and software to address fundamental shortcomings in design and to expand scope.

Our most immediate weaknesses largely deal with flight and flight startup. We have witnessed unexpected RF interference, resulting in control shutout and unknown yaw issues resulting in further teleoperation issues. We also encountered some issues with GPS and not getting a proper lock. Another issue we are having is our WiFi adapter does not have a long enough range to stay connected to our network. This mainly affects our ability to monitor data feeds on our robot which is a requirement for us. These are the most serious hardware issues we would like to remedy. In terms of revising hardware, we would need to clean up our cabling, add switches for easy setup and tear-down and revise our PDB to get a slightly lower profile. In terms of software, our weaknesses lay in our ambitiousness to start on it. We did not get the opportunity to integrate IMU and RGB measurements like we were hoping to. We also did not get the chance to investigate integration of the mapping framework into our SLAM algorithm. These were all tasks we had scheduled due to optimism but descoped as they did not fall directly in line with our requirements for FVE.

8. Project Management

8.1 Work Breakdown Structure

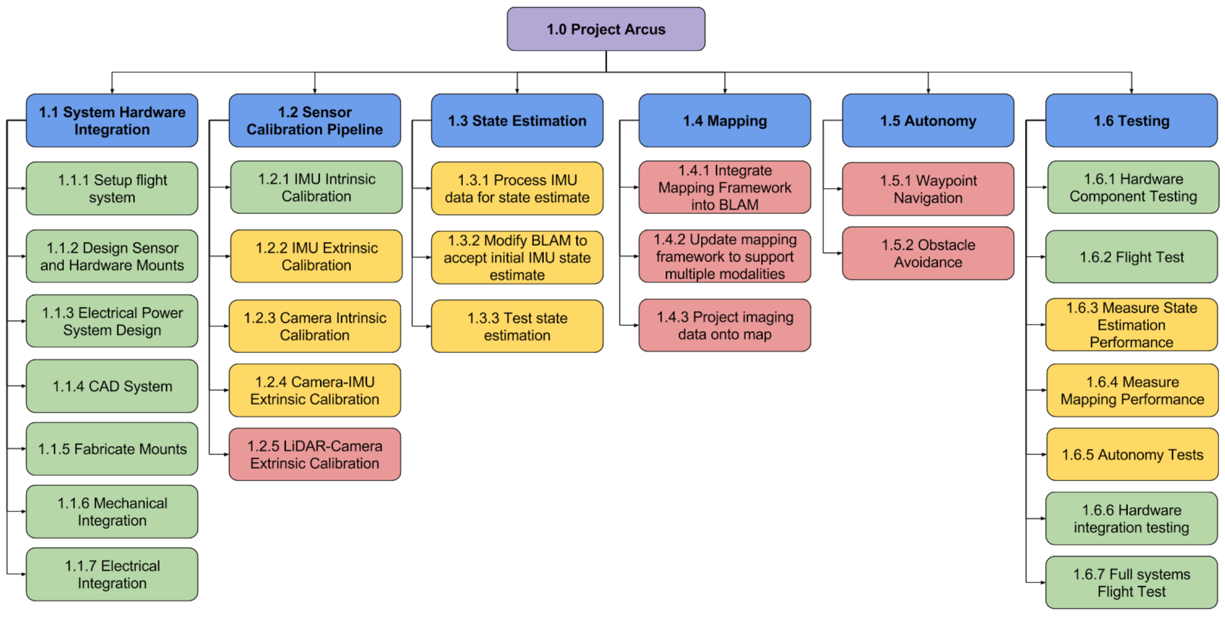

1.0 Project Arcus
1.1 System Hardware Integration
1.1.1 Setup flight system
1.1.2 Design Sensor and Hardware Mounts
1.1.3 Electrical Power System Design
1.1.4 CAD System
1.1.5 Fabricate Mounts
1.1.6 Mechanical Integration
1.1.7 Electrical Integration
1.2 Sensor Calibration Pipeline
1.2.1 IMU Intrinsic Calibration
1.2.2 IMU Extrinsic Calibration
1.2.3 Camera Intrinsic Calibration
1.2.4 Camera-IMU Extrinsic Calibration
1.2.5 LiDAR-Camera Extrinsic Calibration
1.3 State Estimation
1.3.1 Process IMU data for state estimate
1.3.2 Modify BLAM to accept initial IMU state estimate
1.3.3 Test state estimation
1.4 Mapping
1.4.1 Integrate Mapping Framework into BLAM
1.4.2 Update mapping framework to support multiple modalities
1.4.3 Project imaging data onto map
1.5 Autonomy
1.5.1 Waypoint Navigation
1.5.2 Obstacle Avoidance
1.6 Testing
1.6.1 Hardware Component Testing
1.6.2 Flight Test
1.6.3 Measure State Estimation Performance
1.6.4 Measure Mapping Performance
1.6.5 Autonomy Tests
1.6.6 Hardware integration testing
1.6.7 Full systems Flight Test

Figure 15: Work breakdown structure

The work breakdown structure, seen as Figure 15 above, divides the general tasks for the project among six separate categories: system hardware integration, sensor calibration, state estimation, mapping, autonomy, and testing. The individual tasks are then color-coded according to their current work status. Green: done, yellow: in progress, red: not begun yet. These are the highest-level tasks in each functional category, with more fine-grained day-to-day tasks in our project management software, Asana.

8.2 Schedule

Below is a simplified GANTT chart of our upcoming planned tasks for the spring semester, seen as Table 4.

Table 4: Gantt chart of spring schedule

	PR7		PR8	PR9	PR10	PR11	PR12		~SVE
	1/20	2/3	2/17	3/3	3/17	3/31	4/14	4/28	5/12
Hardware									
Wiring cleanup	🟩								
Hardware refresh	🟩								
Wireless debugging	🟩	🟩							
Flight Software									
Pixhawk / ROS interface	🟨								
Testing									
FARO Scan			🟧						
Mapping error script			🟧	🟧					
Localization									
IMU integration	🟥	🟥							
Mapping									
Occupancy grid integration	🟪	🟪							
RGB point cloud coloring		🟪	🟪						
Mesh generation				🟪	🟪				
Mesh texturing						🟪			
Planning									
Initial obstacle detection			🟦	🟦					
Trajectory planning				🟦	🟦				
Obstacle avoidance					🟦	🟦			
Waypoint navigation						🟦	🟦		
GUI Development					🟥	🟥	🟥		
System Debug							🟩	🟩	🟩
FVE Demo Ready								🟩	🟩

There are no outstanding tasks from the fall semester that remain to be completed in the spring semester. After evaluating the previously proposed spring semester schedule, we have decided to revamp our plan for the entire spring semester. As well as an updated schedule, we will also update progress review milestones, validation requirements, and the final SVE. These changes were made due to the fall validation experiment nearly fulfilling all goals of the initially proposed spring validation experiment.

As per these updated requirements, the major outstanding milestones involve increasing the data fidelity of the map for end-users, as well as integrating a basic planning software module that will enable waypoint navigation as well as avoidance of obstacles of a predetermined minimum size. These requirements are detailed below.

8.3 Test Plan

8.3.1 Progress Review Milestones

Table 5 shows the next milestones we plan to reach in time for each of the six progress reviews in Spring semester.

Table 5: Progress review milestones

PR7	- Hardware refresh complete, cabling updated - Interface setup between flight controller, ROS
PR8	- FARO scan of LaFarge complete - IMU integrated into localization algorithm - Map type switch to occupancy grid
PR9	- Point cloud to mesh generation functional - Point cloud colored with RGB data
PR10	- Wireless (RF, GPS) communications and sensing debugged - Map mesh textured with RGB data - Obstacle detection functional
PR11	- Waypoint navigation functional
PR12	- Full waypoint navigation functional - Mapping, localization, planning stack integrated with vehicle

8.3.2 Spring Validation: Full System Experiment

Due to the project moving more quickly forward than expected, increased autonomy capability has been scoped into the spring validation experiment demonstration. The demonstration, along with the following validation requirements seen in Table 6 will be shown during the SVE.

Table 6: Updated SVE requirements

SPR1	Avoid obstacles of at least 100cm x 100cm x 100cm in size
SPR2	Reduce odometry error to less than 0.5m/meter traveled
SPR3	Display colorized point cloud
SPR4	Reduce mapping error to 0.25m/point
SPR5	Display textured mesh map
SPR6	Autonomously navigate to waypoints 30m away while avoiding unmapped obstacles of size referenced in SPR1

With the fulfillment of these requirements, the vehicle will demonstrate a minimum level of autonomy required for the further development of autonomous exploration algorithms

The full system experiment will demonstrate the combined functionality of the various subsystems and their capability in an example use case. The test will occur in LaFarge Quarry, an outdoor environment that mimics the final use case environment, whose depiction is located in Figure 16. The completed UAV, base station, and GPS will be needed to complete this experiment. The following steps describe the testing procedure:

1. Show video of vehicle mapping the quarry in real-time
2. Show video of point cloud RGB colorization, mesh generation, and mesh texturing **[SPR3, SPR5]**
3. Display updated map metrics with improved error values **[SPR2, SPR4]**
4. Display final result in RViz / GUI
5. Power on vehicle, localize in 3D map in RViz / GUI
6. Place unmapped obstacle to environment
7. Command vehicle to distance waypoint in quarry through obstacle, then return **[SPR1, SPR6]**

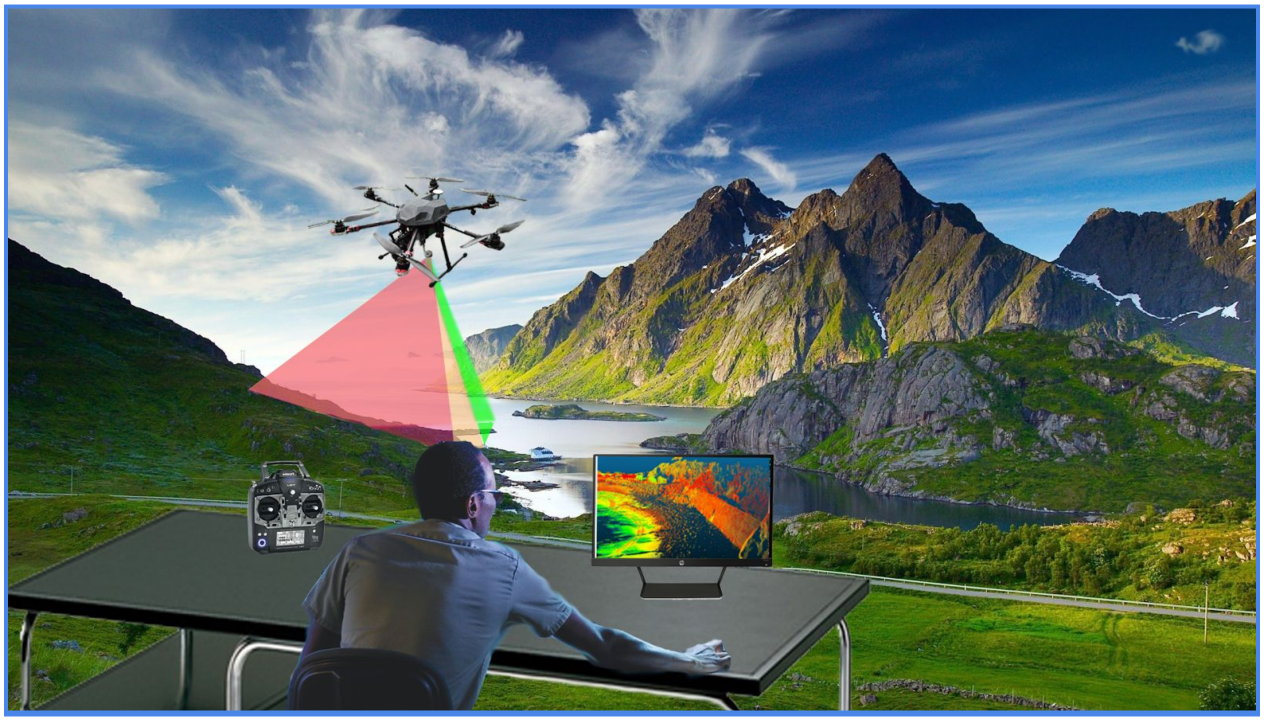


Figure 16: Ideal SVE experimental setup

8.4 Parts and Budget

Below in Table 7 is a list of equipment that has either already been acquired by the project sponsor or has been purchased by the team. These are all components in use on the vehicle or have been transformed into functional parts installed on the vehicle.

Table 7: Bill of Materials. Items without a subtotal were provided by the project sponsor

Type	Part	Cost	Quantity	Subtotal
GPS	Piksi		1	
Telemetry Radio	915 MHz SiK Radio		1	
LIDAR	Velodyne VLP-16		1	
Computer	GB-BSi7-6500		2	
RAM	Corsair 2x8GB		1	
SSD	SAMSUNG 850 EVO M.2 250GB		1	
Computer Power Supply	DCDC-NUC		1	
Camera Lens Holder	CMT821		2	
Camera Lens	DSL219D-650-F2.0		2	
Autopilot	Pixhawk		1	
Flow Sensor	PX4Flow		2	
Frame	Tarot 680Pro		1	
Motors	700kv U3		8	
ESCs	Air 40A		8	
Propeller	CF 13x4.4		5	
12v Regulator	RMRC 5V/12V BEC		1	

Battery	6600mAh 4-Cell/4S	2	
RGB Camera	mvBlueFox-MLC200wG		
Electronics, cabling			(421.17)
Assembly hardware, fabrication components			(542.35)
Testing expenses			(30.00)
		Total Expenditure	(993.52)

8.5 Risk Management

The updated risks for the project are included in Table 8 below and the corresponding likelihood-consequence chart can be seen in Figure 17. For spring semester, we have added two risks that have been identified as Risk 9 and 10 in Table 8. The most critical new risk is that we could potentially crash the UAV through autonomy. This goes hand in hand with Risk 6, which is a risk that the UAV crashes through human tele-operation. These risks can be mitigated with extensive testing in safe environments and practice flying smaller UAVs for all team members. Another new risk for next semester has to do with the mapping framework. The Robust Adaptive Systems Laboratory has provided us with a framework based on occupancy grid mapping and probability belief distribution that we would like to replace our current octree framework with. There are some expected challenges in integrating a new mapping framework, and, given that this is not successful, we may not be able to validate SPR3 or SPR5. To mitigate this risk, we may try to modify the octree framework or search for a different one to use.

We have successfully mitigated some anticipated risks, which have, accordingly, been removed from the risk management table. For the risk of Velodyne damage, we have attached a foam-padded crash cap on the bottom of the Velodyne. Being unable to map online was also a previous risk, but the fast processing speeds of the Gigabyte Brix have resolved this issue. We have also found a resolution to the risk given that the Lafarge Quarry is inaccessible; we are able to use the Planetary Robotics Laboratory netted arena that has Vicon cameras for tracking, and GPS is not required in this case. Finally, we were able to resolve our issues with the calibration software, Kalibr, by using a different setting for the video: 8 bit grayscale.

Table 8: Risk Management Table. L stands for 'Likelihood' and 'C' stands for Consequence. Under the 'Type' category, T stands for 'Technical,' S stands for 'Scheduling,' and B stands for 'Budgeting'

ID	Risk	Description	Type	Req.	L	C	Mitigating Actions
4	Unable to form loop	Despite best efforts, we are unable to close loops in our SLAM package	T	MPR6	2	2	- Find another loop closure library

	closures with BLAM						
5	Asynchronous timing between Velodyne LiDAR and RGB camera	LiDAR scans at 10 Hz, RGB scans at 100 Hz (check these numbers), so pixels may be assigned incorrectly, resulting in a warped/inaccurately sensor-fused map	T	SPR3, SPR5	3	2	- Delay RGB colorization until LiDAR point cloud map developed or attempt to predict the UAV's pose in the future based on current velocity/FOV & colorize immediately - Throw out extra RGB and synchronize with LiDAR - Trigger the cameras with the LiDAR
6	Pilot crashes UAV during flight	Vehicle damage, bodily, property harm possible	T, S, B	MPR1	3	5	- Get extensive experience flying UAVs for all team members - Emergency landing procedure
8	RF interference	Operating the UAV at 2.4 GHz for RC control and at 5.0 Ghz for wifi / GPS leaves system vulnerable to RF interference from other devices, causing us to lose control of the UAV or online mapping goes down	T	MPR2	4	5	- Static testing in RF noisy environments - Construct a test to validate UAV communications performance - Look at flight logs - See if limiting RC controller channels is possible
9	RASL mapping framework cannot be integrated into BLAM	Mapping framework incompatible or too difficult to integrate with BLAM	T	SPR3, SPR5	3	2	- Find another framework to use - Modify current framework code to make it more compatible with our purposes
10	Robot crashes while flying autonomously	Vehicle damage, bodily, property harm possible	T, S, B	SPR1	3	5	- Testing in simulation - Extensive testing indoors in a safe environment

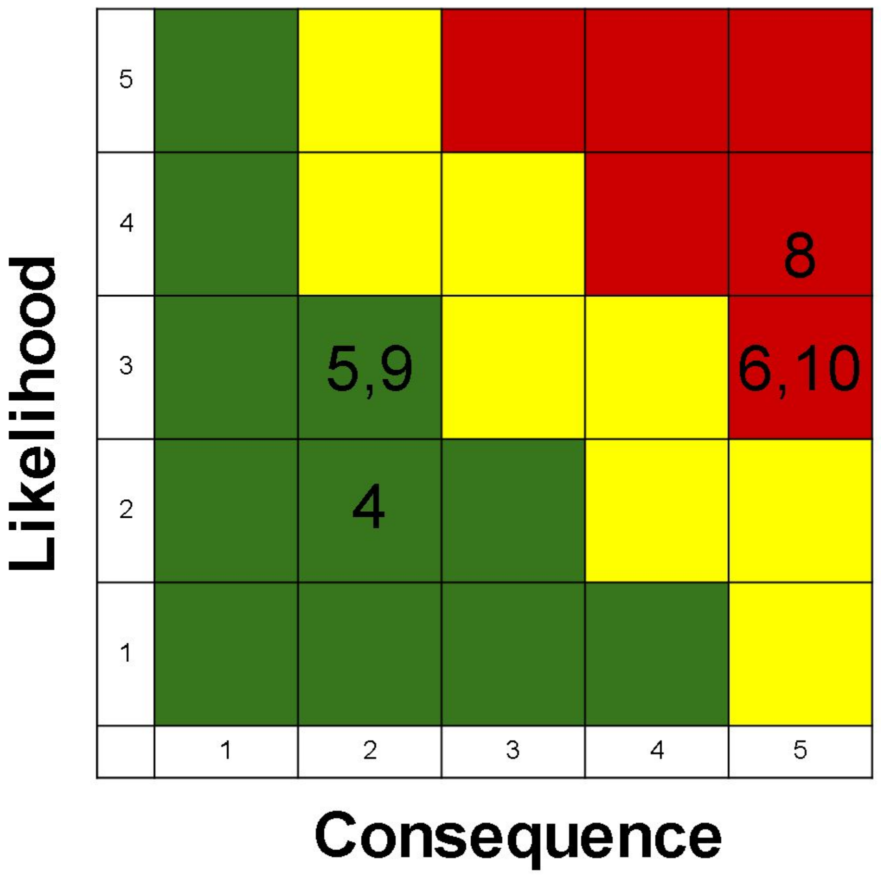


Figure 17: Likelihood-consequence chart with current risks plotted.

9. Conclusions

We have learned a couple of key lessons about building and designing a robot during the course of this semester. For example some technical lessons like calibration for a fisheye lens can be a very challenging task and that using a camera with a more standard rectangular projection is much easier to work with. Also hardware switches for powering up a robot is much quicker and easier and can lead to cleaner cabling than just manually disconnecting power connectors. Similarly, adding modularity tends to increase the serviceability of the robot, as well as making more amenable to future unplanned changes. Plugging in peripherals into the computer, as well as adding sensors and payload modules should be simple and not require disassembly of the vehicle.

The key activities for spring consist of rescoping to add in more functionality related to our use case. This consists of meeting our original requirements of mapping with small error and mapping in real-time with multiple imaging modalities. The key activity here is adding in the additional RGB camera data to augment the point cloud information. The next key activity is switching over to an occupancy grid map. This will make autonomy, our third key activity, much easier as it would be very difficult to do with a voxel grid approach. We are also hoping to create a user interface to make it simple and easy for a researcher to startup the associated mapping software. These are the key activities we would like to accomplish in order to envision our use case.

10. References

[1] "Website." [Online]. Available: http://shop.gopro.com/cameras/hero-session/CHDHS-102-master.html. [Accessed: 13-Dec-2016].

[2] "USB 3.0 Camera Module | USB3 Camera Board." [Online]. Available: https://www.e-consystems.com/See3CAM-USB-3-Camera.asp. [Accessed: 13-Dec-2016].

[3] "Home - Pixhawk Flight Controller Hardware Project." [Online]. Available: https://pixhawk.org/. [Accessed: 13-Dec-2016].

[4] R. M. Rc, LLC, and the Best Source for All Your FPV Needs, "T-Motor ESC - AIR 40A, 2-6 cell, No BEC [TM-ESC-AIR-40A] - $39.99 : Ready Made RC LLC, The Leader in All Things FPV, RC, and Beyond." [Online]. Available: http://www.readymaderc.com/store/index.php?main_page=product_info&cPath=505_507_518&products_id=3832. [Accessed: 13-Dec-2016].

[5] R. M. Rc, LLC, and the Best Source for All Your FPV Needs, "Tiger Motor - U3 KV700 [U3-KV700] - $109.90 : Ready Made RC LLC, The Leader in All Things FPV, RC, and Beyond." [Online]. Available: http://www.readymaderc.com/store/index.php?main_page=product_info&cPath=505_506_516_528&products_id=2640. [Accessed: 13-Dec-2016].