

MULTI-AGENT MAPPING IN A GPS-DENIED ENVIRONMENT

By

Kourosh Arasteh
Junwon Choi

Final Report for ECE 445, Senior Design, Spring 2019
TA: Amr Martini

2 May 2019
Project No. 76

Abstract

The objective of our project was to localize and map a given environment using multiple agents. We use two ground agents, a sky agent, and a controller agent to localize and map a given test environment. The sky agent is responsible for localizing the ground agent with a high level of confidence to ensure we know where the ground agent is located within the environment. The ground agent is responsible for mapping its current surroundings to update the global map of our test environment. Finally, our controller agent was responsible for collecting localization information from the sky agent and map data from the ground agent to build a global map and instruct ground agent movements. All three low-cost, low-resource agents work together to build a global map of topographical features.

Contents

1. Introduction	1
2 Design.....	3
2.1 Sky Agent	3
2.2 Controller Agent	3
2.2.1 Ground Agent Localization.....	3
2.2.2 ROS Data Pipeline	3
2.3 Ground Agent Sensing Subsystem	4
2.3.1 Inertial Measurement Unit (IMU).....	4
2.3.2 LIDAR.....	5
2.4 Ground Agent Motion Subsystem	6
2.4.1 L298N Motor Driver	6
2.4.2 Tracked Chassis	7
2.5 Ground Agent Control Subsystem.....	7
2.5.1 Microcontroller and FTDI USB-to-UART Interface	7
2.5.2 Wi-Fi Chip	8
2.6 Ground Agent Power Subsystem.....	8
2.6.1 Lithium Polymer Battery.....	8
2.6.2 Switching Buck Converter	9
3. Design Verification	10
3.1 Ground Agent Localization	10
3.2 ROS Structure	10
3.3 Ground Agent Sensing Subsystem	11
3.3.1 IMU	11
3.3.2 LIDAR.....	11
3.4 Ground Agent Motion Subsystem	13
3.4.1 Directional Control via Serial Input.....	13
3.4.2 Safe Acceleration and Velocity Limiting.....	13
3.4.3 Turning In-Place.....	14
3.5 Ground Agent Power Subsystem.....	15
4. Costs	16
4.1 Parts	16
4.2 Labor	16
5. Conclusion.....	17

5.1 Accomplishments.....	17
5.2 Uncertainties	17
5.3 Ethical considerations	17
5.4 Future work.....	17
References	19
Appendix A Requirement and Verification Table	20

1. Introduction

Many military base locations and military construction sites in developing countries are often GPS-denied and/or contested environments [1]. For such construction projects, collaborative robots pose an attractive solution for mapping topographical features in a large-scale construction space. However, without a robust mapping process for the environment, it would be impossible to develop a plan for distributed autonomous construction. An accurate and dynamic mapping of the region(s) of interest is necessary before autonomous construction robots can be deployed. In applications of collaborative robotics, keeping track of the state of the environment is often attempted using individual agents that simultaneously perform localization and mapping onboard. To do so, these agents require powerful computation capabilities and constant communication with both GPS satellites and D-GPS towers. However, there are applications of robots like these in locations that are GPS-denied or contested to the point that extraneous long-range communication should be avoided or is unavailable entirely. Therefore, our problem statement is as follows:

How do we keep track of a large, sparse map between several ground agents, without using GPS or long-range localization technologies?

Our approach to this problem statement is to discretize localization to a sky agent and mapping to two ground agents. Our block diagram for our full system is outlined by Figure 1, where we can see that our project is divided into 6 main blocks: ground agent control subsystem, ground agent power subsystem, ground agent motion subsystem, ground agent sensing subsystem, controller agent command subsystem, and the sky agent subsystem. The ground agent control subsystem is mainly responsible for interfacing with our ESP32 Wi-Fi chip and ATmega2560 microcontroller. The ground agent power subsystem is responsible for providing the correct amount of power to each of our subsystems onboard the ground agent. Our ground agent sensing subsystem is responsible for interfacing with the LIDAR, IMU, and motor controllers, and the ground agent motion subsystem is responsible for moving the robot via DC motors and a motor controller. The controller agent command subsystem is responsible for hosting the video processing and ROS structure involved with localization, and the sky agent subsystem is simply a USB camera plugged into the controller agent to provide a live video stream.

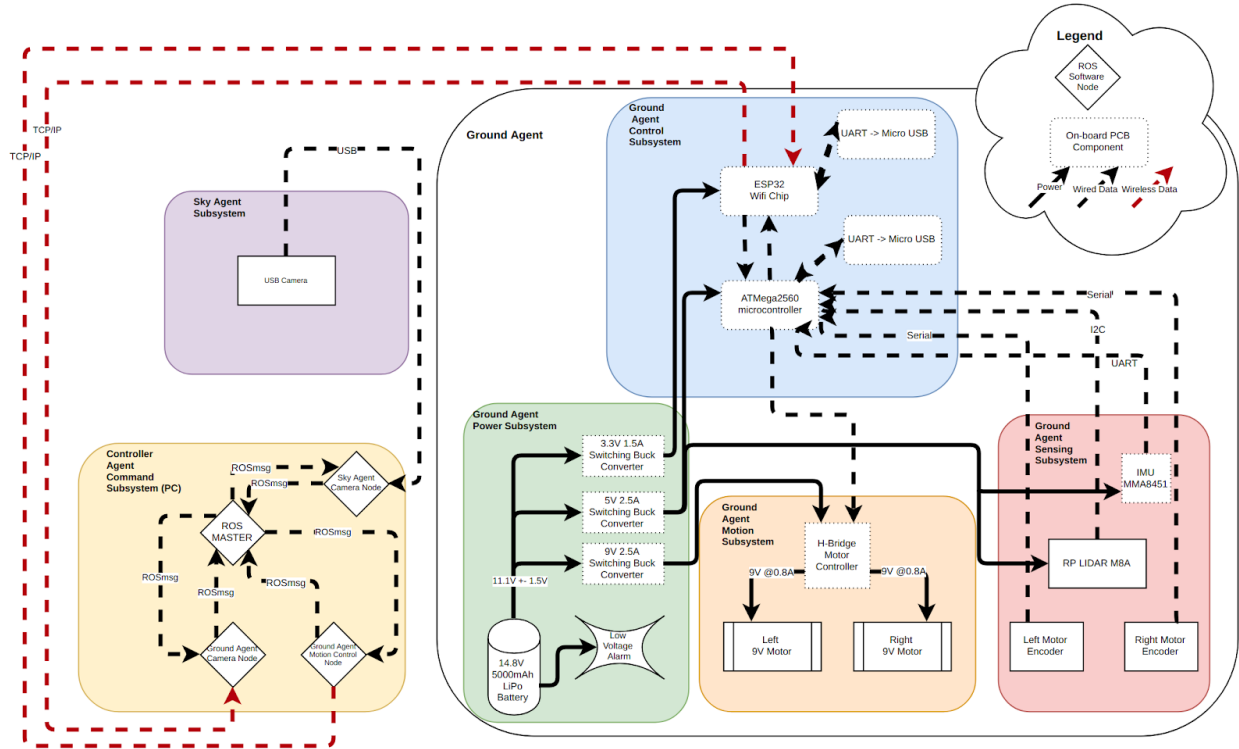


Figure 1: High-level block diagram

Our high-level performance requirements are as follows:

1. Sky agent should be able to localize the ground agent's position within $\pm 5\text{cm}$ and orientation within $\pm 15^\circ$ within an $8' \times 8'$ space outdoors, not exceeding a 15° grade.
2. System should be able to combine up to 64 LIDAR point clouds, each measuring $1' \times 1'$, from multiple ground agents with zero overlap into the global map using location and orientation data obtained from the sky agent as well as inertial data from the Ground Agent to estimate slope at each sample point.
3. System should be able to detect and create bounding boxes for objects within the stage similar in size to the ground agent (20cm^2 to 0.1m^2 top area).

Throughout the course of the semester, no block-level changes were made to our project.

2 Design

2.1 Sky Agent

Our sky agent is a USB webcam connected directly to our controller agent with a cable, so our design choices for the sky agent were limited to the kind of camera we wanted to use. We decided to use a 720p USB webcam to reduce the cost of our project. If cost weren't a factor, a 4k USB camera would be much more useful and help improve our video processing algorithm later described in Section 2.2.1, which explains the importance of video resolution in our project.

2.2 Controller Agent

The controller agent is a laptop PC running Linux 16.04 and is responsible for two core components of our project: 1) processing the sky agent video to localizing ground agents and 2) hosting the ROS structure, which served as our data pipeline for LIDAR point clouds from ground agents and localization data from the sky agent. The main reason for using a Linux machine was because ROS is extremely well-documented for Linux 16.04 and can be integrated seamlessly into projects on a Linux machine. We could've chosen to use a Mac as our controller agent, but ROS does not formally support MacOS and ROS integration was a key component in our project. We did not have a Windows machine available for use as the controller agent and using the UIUC EWS Windows machines are out of the question because we would not be able to install the libraries and dependencies for necessary for ROS or OpenCV.

2.2.1 Ground Agent Localization

Our localization algorithm uses contour detection to find the identifying arrow marker on top of each ground agent. Once the contours of the arrow have been found, we're able to extract the center point of the arrow as the ground agent's location and compute the orientation angle of that arrow to determine which direction our ground agent is facing. The algorithm is capable of distinguishing between each ground agent and providing unique localization data respectively by isolating the separate colors of the identifying arrow marker using a bitmask. The bitmask completely filters out anything within the video frame that isn't a blue or red arrow, allowing us to get accurate and stable readings for each of our ground agents.

One other algorithm considered was the Scale-Invariant Feature Transform (SIFT) algorithm [9]. SIFT is able to acquire the same location and orientation angle, but much more reliably than a contour-based feature detection algorithm. We initially tried to use SIFT but observed that we were getting terrible keypoint matching and feature detection in real-world performance. Upon further research, we found out that SIFT struggles to perform well with low resolution imagery. Given that our sky agent was a 720p camera, which was then further cropped as a result of image warping, the video frames were extremely low resolution and, as a result, SIFT performed poorly, which was the main reason why we chose to use a contour-based algorithm.

2.2.2 ROS Data Pipeline

The Robot Operating System (ROS) is an open-source framework that provides a communication infrastructure for message-passing between processes. We chose to use ROS because it allows us to easily define a structure or pipeline for data passing. This is useful because the controller agent is passed a stream of data from the localization code (as shown in Figure 3 by the Associated Output) and ground agents are also passing in entire LIDAR point clouds, and ROS makes it very simple to manage, organize, and identify the flow of various data.

Figure 2 shows a diagram of our ROS structure and the different kinds of data being passed between agents. Our project involves three separate agents, with some needing to interacting with more than one kind of data. Without ROS, data passing between processes and agents would be a significantly more difficult task.

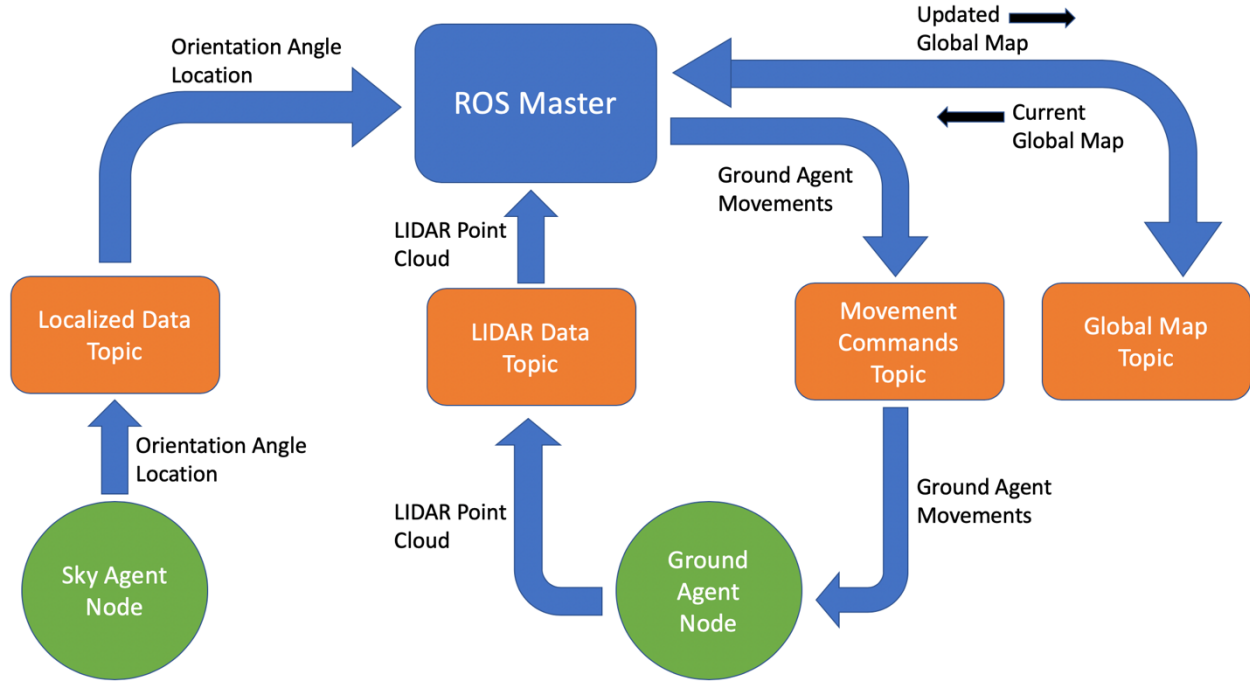


Figure 2: ROS Structure

2.3 Ground Agent Sensing Subsystem

2.3.1 Inertial Measurement Unit (IMU)

In order to estimate the robot pitch, yaw, and roll on a sloped surface, we require at least an accelerometer capable of inertial measurement within 5 degrees. We decided that using the MMA845X family of accelerometers would minimize cost and space taken up in the robot, as this family of chips had the smallest form factor available when compared to similar chips like the SEN-09269 or LIS3DH. In considering the MMA845X family of accelerometers, we selected the MMA8451, which has the highest precision of the family, with a built-in 14-bit ADC.

The design for data flow is outlined in Figure 3, where data is passed from IMU over I2C to the ATmega2560 microcontroller. The microcontroller does some basic data mapping and conversion of inertial data to Euler angles, and packages this data with LIDAR information for easy Serial access via Wi-Fi chip or USB Serial port.

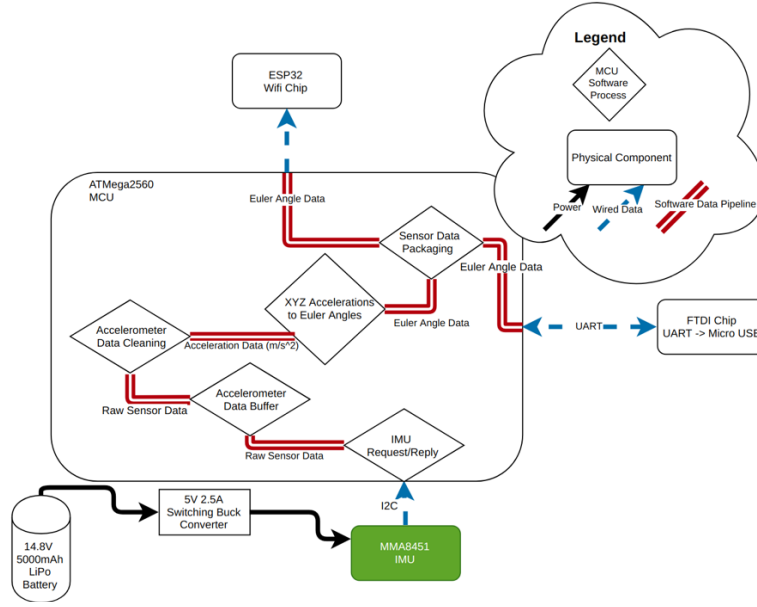


Figure 3: IMU Data Pipeline

2.3.2 LIDAR

In the problem statement outlined in the introduction above, we emphasize the use cases of this project in topographical mapping. In homogeneously colored or textured topographies, or in inconsistent lighting conditions, using single-lens or stereographic cameras might not be sufficient to capture features of a sloped ground surface. For these reasons, we elected to utilize a low-cost 2D LIDAR. Among low-cost 2D LIDARs, the RPLIDAR A2-M8 stood out as a good balance of accuracy and cost. At \$300, it is remarkably more affordable than industry standard 2D LIDARs like the Hokuyo UST-10LX, which costs \$1,600. However, the A2-M8 is still able to generate point clouds with an advertised angular resolution of 0.9 degrees and distance error of ~1%. [1]

Upon testing the LIDAR, we found that the ATmega 2560's UART serial buffer was not able to keep up with the transmission speed of the A2-M8, and only about 25% of each scan was captured by the microcontroller. Because we expected an angular resolution of about 1 degree, we were able to layer four scans into each frame to generate a complete representation of the space around each Ground Agent. This is evidenced in Figure 4 below, where each scan makes up about 90 points between 0 and 360, and the combination of 4 scans yields an acceptable angular resolution.

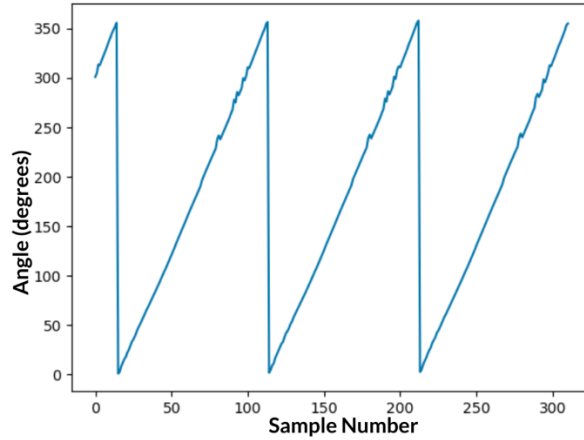


Figure 4: LIDAR Scan Resolution

2.4 Ground Agent Motion Subsystem

The Ground Agent motion control input flow is outlined in Figure 5 below, where character data passed over Serial input, either from the ESP32 Wi-Fi chip or USB Serial port. This character data is parsed by the ATmega2560 microcontroller, which then maps these characters to motor control commands and sends these commands to the L298N motor controller.

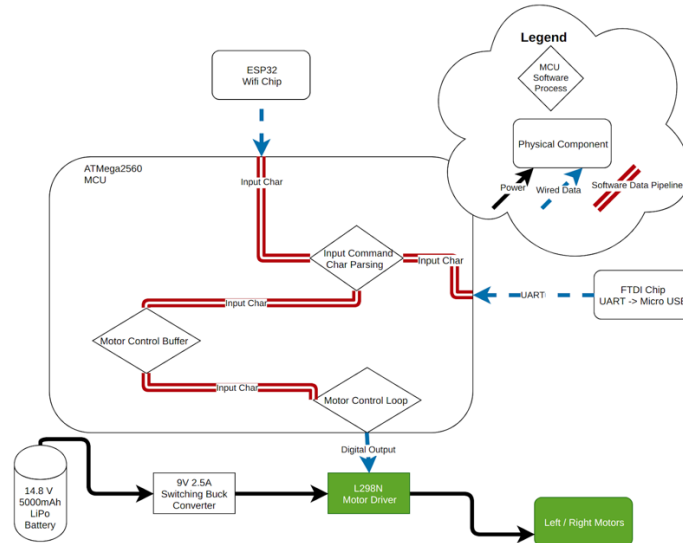


Figure 5: Motion Control Pipeline

2.4.1 L298N Motor Driver

We originally planned on constructing our own dual-channel h-bridge motor driver, but because our motors were rated for up to 2.5 A combined current draw, this would necessitate the use of power MOSFETs and heatsinking, which could be hazardous if improperly constructed. We elected instead to use the L298N Dual H-Bridge motor driver, which safely provides 2A to each channel with an integrated heatsink. This was the lowest cost motor driver we could find, where alternatives like the Cryton 10A dual-channel motor driver we originally planned on using cost around four times the price of the L298N. The L298N is controlled via PWM signal and digital inputs from the microcontroller, shown in Figure 6, and controls its outputs from the digital inputs given in Table 1:

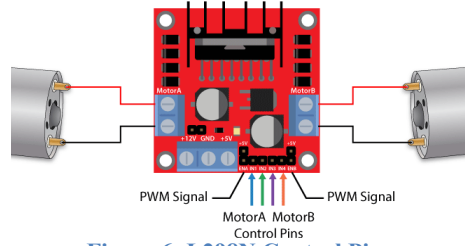


Figure 6: L298N Control Pins

Table 1 L298N Input Mapping

ENA	IN1	IN2	Output Behavior
0	N/A	N/A	Motor A is off
1	0	0	Motor A is stopped (brakes)
1	0	1	Motor A is on and turning backwards
1	1	0	Motor A is stopped (brakes)
1	1	1	Motor A is on and turning forwards

2.4.2 Tracked Chassis

We decided to purchase a pre-fabricated chassis to minimize the mechanical design component of this project. A chassis that included integrated motors and encoders would be ideal to prevent integration issues between the motors and drivetrain. Given that our Ground Agents are intended for outdoor use in rough terrain, we opted for a tracked chassis over a wheeled one, which also simplified the control necessary for motion and steering. Beyond these factors, we purchased the Mountain Ark chassis based on its high availability of spare parts, light aluminum construction, and relatively low profile, which was needed to ensure that our LIDAR would be able to map objects close to the ground.

2.5 Ground Agent Control Subsystem

2.5.1 Microcontroller and FTDI USB-to-UART Interface

We decided that the programmable microcontroller on each Ground Agent would be the ATmega2560. The microcontroller exists mainly as a hub to buffer, store, package, and pass on sensor data from the Ground Agent Sensing Subsystem to the onboard Wi-Fi Chip. In considering competitors to the ATmega2560, including the ATmega328P and PIC32 boards, we found that the ATmega2560 has an optimal number of UART ports (4 ports on the ATmega2560 vs. 1 port on the ATmega328P and 2 ports on the PIC32) [2]. Having at least three UART ports allows us to be able to communicate with the LIDAR, ESP32 Wi-Fi chip, and a Serial monitor for logging, debugging, and flashing purposes. While other PIC32 chips have more than 2 UART ports (the PIC32MX575F256H has 6 UART ports) [3], these alternatives are not programmable or supported by common open source IDE's and logging software to the same extent that ATmega microcontrollers are by the Arduino IDE and community resources. The Arduino IDE and online support resources were very useful in our design process, and we ultimately decided to use the most capable microcontroller supported by this system, the ATmega2560.

We wanted to be able to quickly upload new code to the microcontroller from a PC and found that a USB cable was the easiest method besides purchasing a programmer, so we needed a USB-to-UART interface. We selected the FTDI 232R as it was a well-documented, low-cost method for communicating with an ATmega microcontroller over USB.

2.5.2 Wi-Fi Chip

The Wi-Fi chip is needed to interface the microcontroller with the Controller Agent. It must handle sending LIDAR and IMU data packages to the Controller Agent, as well as receiving control inputs from the Controller Agent. We decided to use 2.4 GHz Wi-Fi over alternatives like 5GHz Wi-Fi or Bluetooth based on the volume of data we expect to pass from a Ground Agent at a given time, and the greater range that 2.4GHz Wi-Fi has over both 5GHz and Bluetooth, which would aid in potential scaling of the mapping environment for this project. As the LIDAR would produce the most data needing to be passed over Wi-Fi, we estimate the maximum desired throughput of our communication in Equation 1:

$$10 \frac{\text{rotations}}{\text{second}} \cdot \frac{360 \text{ degrees}}{\text{rotation}} \cdot \frac{1 \text{ sample}}{0.9 \text{ degrees}} \cdot \frac{32 \text{ bits}}{\text{sample}} = 16 \text{ kbps} \quad (1)$$

Given that the typical transmission output is 20 Mbit/sec for the ESP-32 Wi-Fi chip at 2.4GHz bands, the above throughput of 16kbps is easily attainable.

We selected the ESP-32 Wi-Fi chip over competitors like the ESP-8266 due to the fact that the ESP-32 is the successor to the ESP-8266, with higher typical clock frequency and addition of SRAM and Flash memory. [4]

2.6 Ground Agent Power Subsystem

The power subsystem structure is outlined in Figure 7 below, where two TPS54290 Switching Buck converter chips provide the three power supply levels required by the system. Here, output inductors and capacitors, bias and output resistors, and pull-down resistors and capacitors provide the optimal operation conditions as specified in the TPS54290 datasheets [5].

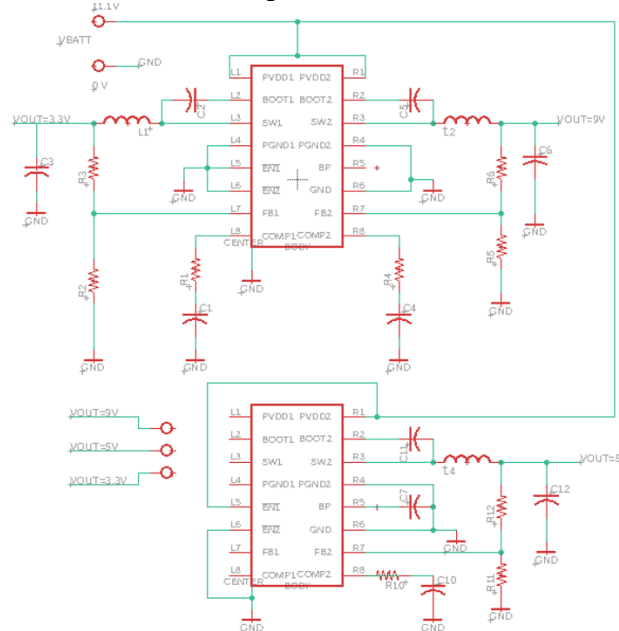


Figure 7: Power Supply Circuit Schematic

2.6.1 Lithium Polymer Battery

We determined that each Ground Agent requires a battery, as tethered power would not be an option in the field. In order to leave an acceptable voltage buffer between our battery voltage and any voltage regulator that we would decide to use, we decided that a 4-cell battery (4S) would

provide this buffer more safely than a 3-cell (3S) battery would. In the analysis of Ground Agent power consumption outlined in Table 2, we found that a 5000mAh would reliably be able to power each Ground Agent for over an hour of constant usage, an acceptable mapping time period for multiple mapping runs before recharging.

Table 2 Component Power Draw Characteristics

Component	Model	Voltage Range (V)	Average Voltage (V)	Maximum Current (mA)	Average Current (mA)
LIDAR	RPLIDAR A2M8	4.9 - 5.5	5	600 Operating, 1500 Start-up	450
Motor Controller	Qunqi	5 - 35	9	2400 - 9000	400 - 2400
IMU	Arduino MMA8451	3.3 - 5	5	165	0.006 - 0.165
Wi-Fi Chip	ESP-32	2.2 - 3.6	3.3	1200	500
Microcontroller	Atmega 2560	1.8 - 5.5	5	200	0.14 - 10
Switching Converter	TI TPS54290	4.5 - 18	10	10	1.65 10 Switching
USB to UART FTDI	FT232RL	0.5 - 6	4.0 - 5.25	24	2
System Characteristics			10	~12000	~3400

In selecting the type of battery, our main motivators were safety, weight, and power density. We found that Lithium Polymer (LiPo) batteries offered an excellent combination of low weight, relative safety, and high-power density when compared to Lithium Ion, Lead Acid, and Nickel-Metal Hydride batteries, at a trade-off of higher cost. Lithium Polymer batteries also have the potential to damage themselves or stop working if their cell voltages drop below a manufacturer specified threshold of 3.0 V. In order to prevent coming anywhere near this voltage, we use low-voltage alarms set to alert at 3.3 V.

2.6.2 Switching Buck Converter

In the early design of our system, we found that we required 3 power supplies; a 9 V supply at 2.5 A, a 5 V supply at 2.5 A, and a 3.3 V supply at 1.5 A. This was needed to power our motors, control circuitry, and Wi-Fi chip, respectively. We initially planned on using a voltage divider circuit to produce these supplies. After learning more about power regulators, we decided upon a switching regulator over a linear voltage regulator to optimize efficiency. Calculating the potential power dissipation for our three desired power supply lines from the 14.8 V battery seen in Equation 2 below, we found that our maximum power dissipation would be approximately 27 W for each Ground Agent, while a Switching regulator would produce less dissipation at 90% efficiency [5].

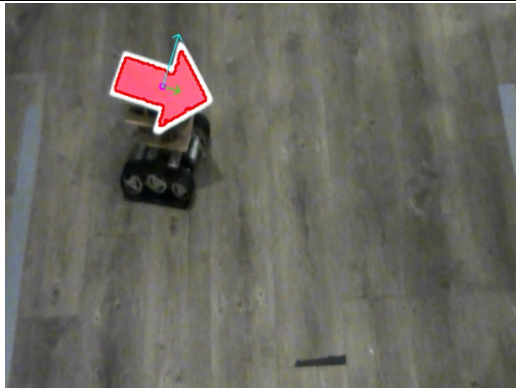
$$\begin{aligned}
 (14.8\text{ V} - 9\text{ V}) \cdot 2.5\text{ A} &= 14.5\text{ W} \\
 (9\text{ V} - 5\text{ V}) \cdot 2.5\text{ A} &= 10\text{ W} \\
 (5\text{ V} - 3.3\text{ V}) \cdot 1.5\text{ A} &= 2.55\text{ W}
 \end{aligned} \tag{2}$$

3. Design Verification

3.1 Ground Agent Localization

Our goal was to be able to localize the ground agent's position within $\pm 5\text{cm}$ and orientation within $\pm 15^\circ$. In Table 3, we can see that we were able to successfully detect and localize our ground agent in this video frame. The measured orientation angle of the ground agent for this specific example was 105° (0° being the 12:00 position) and the output given by our code meets our verification requirement of $\pm 15^\circ$ in Table 10.

Table 3 Ground Agent Localization Output

Localized Ground Agents	Associated Output
	<pre> Angle: 106.52774485815404 degrees Center Location: (199, 105) Angle: 104.13531624795684 degrees Center Location: (199, 105) Angle: 105.97312271301331 degrees Center Location: (199, 104) Angle: 104.66170886188858 degrees Center Location: (200, 105) Angle: 104.31926342869151 degrees Center Location: (199, 105) Angle: 105.04558289599856 degrees Center Location: (201, 104) Angle: 102.58449767660669 degrees Center Location: (200, 103) Angle: 104.79225112926072 degrees Center Location: (199, 105) </pre>

3.2 ROS Structure

Our goal was to be able to verify that our ROS Structure can pass at least 16kB of data from one node to another without loss or corruption of data. We chose to test this by passing one full LIDAR scan measuring 134.3 kB in size from the ground agent node to the controller agent (Figure 2 shows a detailed diagram of our ROS structure). The reason we chose to test our structure with a LIDAR point cloud between the controller agent and the ground agent is because one full LIDAR scan is the largest data packet that'll be passing through our ROS structure at any given point in time. Table 4 shows the terminal outputs of the data being sent by the ground agent and the data that was received by the controller agent. As we can see, the data that was sent was received without loss or corruption.

Table 4 ROS Structure Verification

Active Nodes	
<pre> kalinuxmsi@kalinuxmsi-WS60-6QJ:~/catkin_ws\$ rostopic list /controller_agent_10324_1556148754252 /ground_agent_lidar_10301_1556148751567 </pre>	
Data Packet Sent by Ground Agent Node	Data Packet Received by Controller Agent Node

<pre> kalinuxmsi@kalinuxmsi-WS60-6QJ: 232.25 1.70 230.00 3.52 228.00 8.83 227.25 12.39 227.50 14.48 228.25 18.19 229.75 22.12 232.25 26.61 235.25 29.91 239.50 32.05 244.75 36.38 251.00 39.28 258.50 42.50 266.75 46.83 277.75 48.91 289.50 53.02 303.50 56.14 319.75 59.33 340.00 62.16 365.00 65.36 397.00 69.02 436.25 72.23 484.75 75.19 548.50 78.41 </pre>	<pre> kalinuxmsi@kalinuxmsi-WS60-6QJ: ~/catkin_w 232.25 1.70 230.00 3.52 228.00 8.83 227.25 12.39 227.50 14.48 228.25 18.19 229.75 22.12 232.25 26.61 235.25 29.91 239.50 32.05 244.75 36.38 251.00 39.28 258.50 42.50 266.75 46.83 277.75 48.91 289.50 53.02 303.50 56.14 319.75 59.33 340.00 62.16 365.00 65.36 397.00 69.02 436.25 72.23 484.75 75.19 548.50 78.41 </pre>
---	--

3.3 Ground Agent Sensing Subsystem

3.3.1 IMU

Our IMU is required to satisfy a high-frequency querying scheme and provide accurate pitch data. The Serial Logs below and corresponding code show that the IMU is being queried at 115200 baud rate without missed or duplicate sensor data. In order to validate the accuracy of pitch data, we orient the robot pitch from -45 degrees to +45 degrees in 5-degree increments and calculate the IMU-generated pitch angles. We expect the pitch values to be within the 5-degree band of the ground truth, which is indeed the case, as evidenced by our testing output in Figure 8. In this verification test, our mean error was 1.525 degrees, or 4.7% across all measurements, well below our goal mean error of 5 degrees from Table 10.

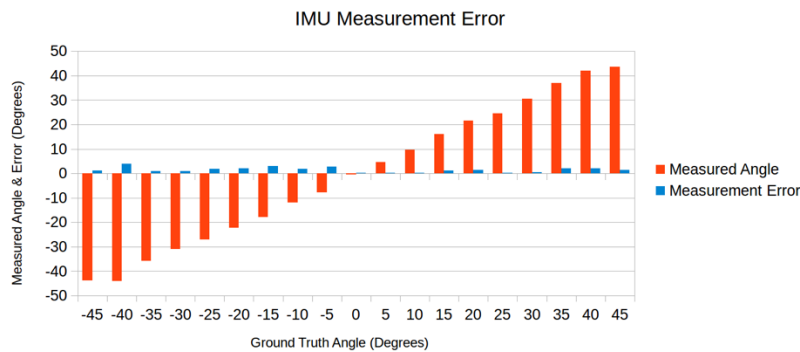
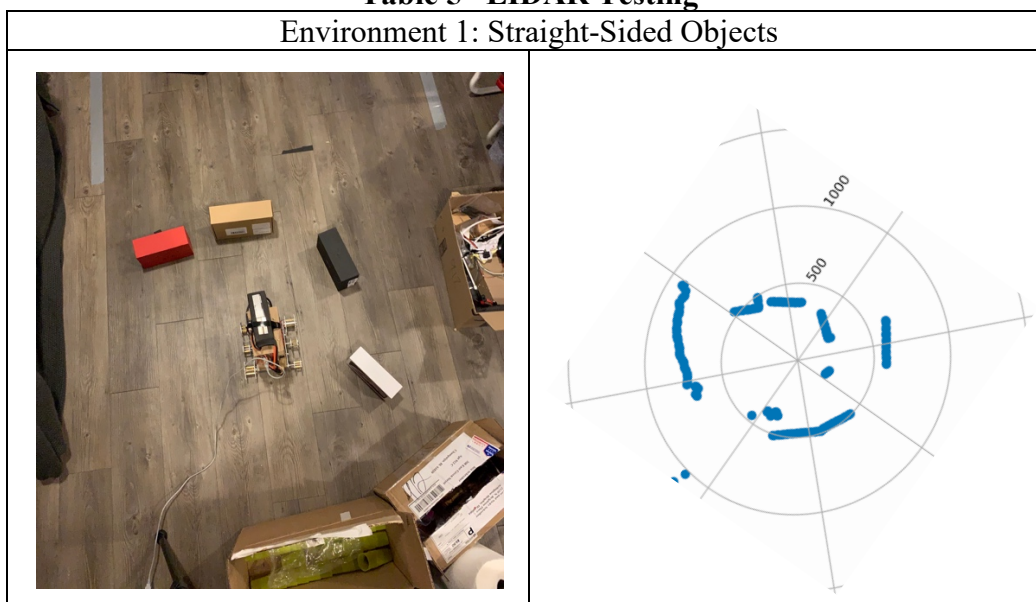


Figure 8: IMU Measurement Verification

3.3.2 LIDAR

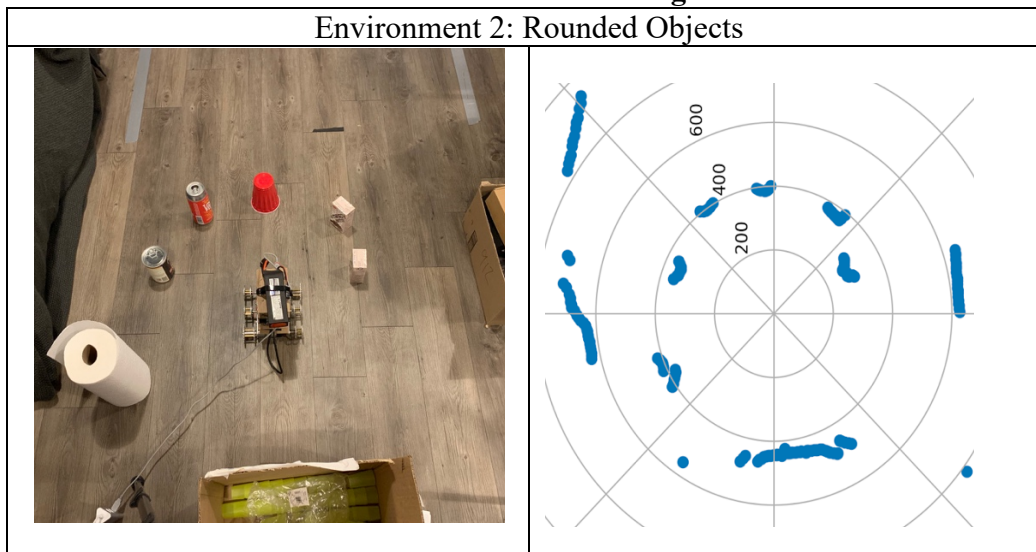
To validate the accuracy of our LIDAR scans, the Ground Agent is set in a prescribed, structured environment of obstacles. We then compare the Point Clouds to this test environment to verify that all obstacles of the correct scale (per HLR3) are recognized by LIDAR returns, and that we have at least 360 samples per LIDAR frame. The following test environment setups and LIDAR point clouds demonstrate the verification of proper operation according to Table 10.

Table 5 LIDAR Testing
Environment 1: Straight-Sided Objects



We observe in Table 5 above that the Ground Agent is able to characterize the faces of all four obstacles, including the smaller white box with the narrow side facing the LIDAR. The Ground Agent is also able to pick up returns from the couch on the left side of the image and surrounding cardboard boxes. In all objects, the LIDAR point clouds are dense and uniform along the reflective edge.

Table 6 LIDAR Testing 2
Environment 2: Rounded Objects



In Table 6 above, we challenge the Ground Agent to characterize very small rounded objects, at and below the bottom range of our HLR3 specification. In all 6 objects, the Ground Agent is able to characterize the roundedness of the faces, and captures nearly a semicircular return for these objects, which will allow for more robust LIDAR stitching.

3.4 Ground Agent Motion Subsystem

3.4.1 Directional Control via Serial Input

The Ground Agent is able to take character-based commands via a Serial Port of the Atmega2560. These character commands {'F', 'B', 'L', 'R', 'S'} represent a 10-cm forward step, 10-cm backward step, left turn in place, right turn in place, and stop command. As demonstrated in Final Demo and revalidated below in Figure 9, our log output on this serial port reveals that the Ground Agent is responsive to all of the above commands.

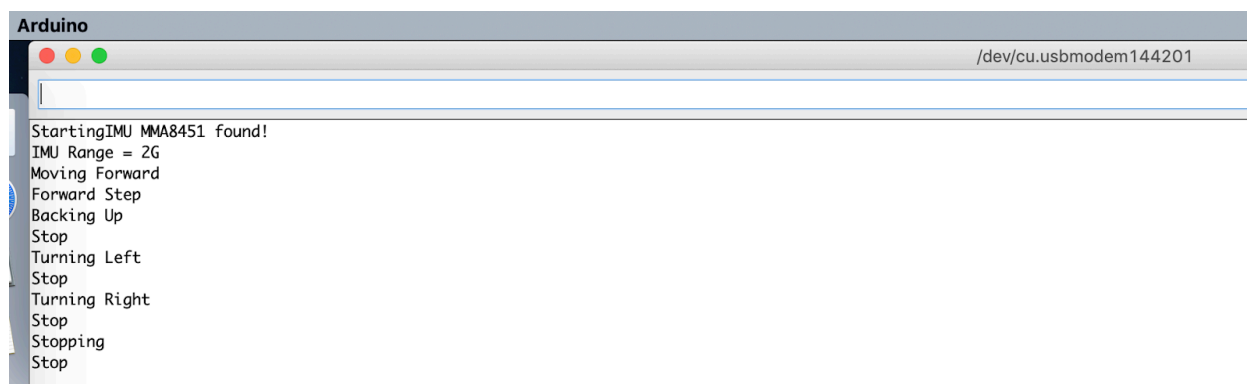


Figure 9: Serial Character Command Verification

Furthermore, testing across 10 of each of the commands and measuring the physical movement of the Ground Agent, we observe this motion to be in-spec regarding the requirement for motion precision, as evidenced below in Table 7.

Table 7 Motion Precision Validation

Iteration	Forward Distance (mm)	Backward Distance (mm)	Left Turn (Drift of Center of Mass) (mm)	Right Turn (Drift of Center of Mass) (mm)	Stopping Distance (mm)
1	106	98	30	44	15
2	109	103	43	40	32
3	108	106	54	42	33
4	99	95	35	43	16
5	106	102	41	53	13
6	107	104	40	35	18
7	110	98	37	28	17
8	105	104	47	34	30
9	106	105	49	21	23
10	106	107	51	38	18
Mean	106.2	102.2	42.7	37.8	21.5

3.4.2 Safe Acceleration and Velocity Limiting

The Ground Agent velocity is limited by the PWM signal provided by the Atmega2560 through an L298N H-Bridge Motor Driver. In practice, this PWM signal is limited in Duty Cycle to

$(100/255) = 39\%$. Figure 10 below shows the rotary speed and Figure 11 the corresponding linear velocity of the Ground Agent for the full range of PWM values, verifying that our Duty Cycle produces a ground velocity well below our velocity limit.

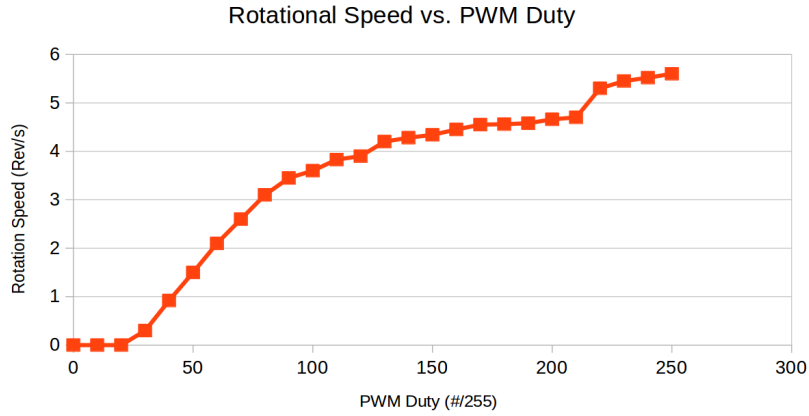


Figure 10: Rotational Speed Verification

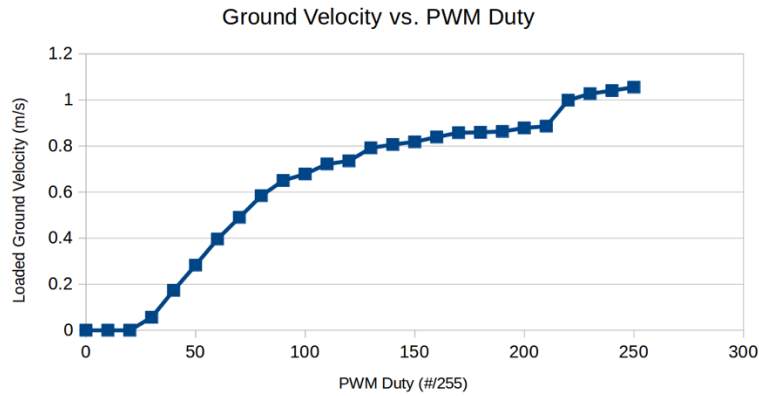


Figure 11: Ground Velocity Verification

3.4.3 Turning In-Place

Testing of 360-degree turn drift of center-of-mass followed the scheme of the command testing, where we attempt 10 360-degree turns with the Ground Agent and validate that each of these turns results in a drift of center-of-mass bounded by $\pm 5\text{cm}$. Table 8 below verifies that our in-place turning is in-spec.

Table 8 Turn Precision Validation

Iteration	Drift Distance (mm)
1	21
2	14
3	43
4	49
5	25
6	53
7	44
8	41
9	58

10	40
Mean	38.8

3.5 Ground Agent Power Subsystem

Following the RV protocol delineated below, we supply a voltage of 13.2V to the Power Supply Subsystem in place of the LiPo Battery and plot the Supply Voltages from a sweep with incremental change of 0.2V up to peak voltage of 16.4V. In Figure 12, we see that our 3.3 V and 5 V supplies remain within specification, with means of 3.29 V and 5.01 V respectively.

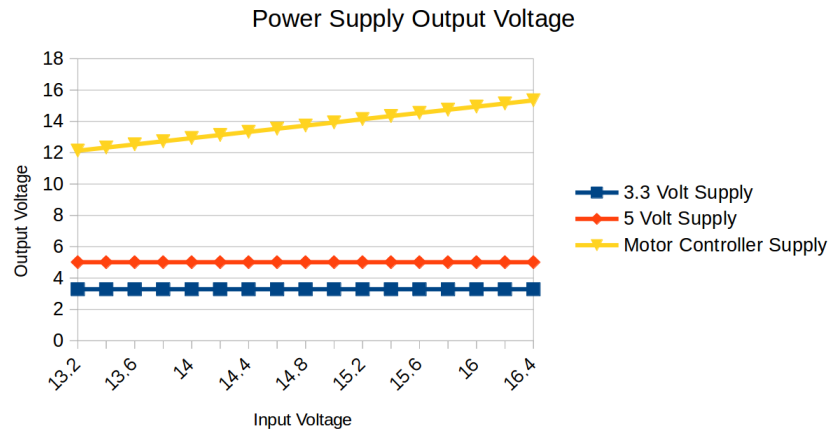


Figure 12: Power Supply Sweep Validation

The supply for our motor controller, requiring between 5 V and 35 V, has a mean of 13.7 V and a range of ± 3.21 V. The reason for this fluctuation is due to the specificity of output inductor values for the switching buck converter. As we only had access to a 10 μ H inductor due to ordering issues, this resulted in imperfect power conversion across the specified battery band, though it still resulted in acceptable operation of the motor driver.

4. Costs

4.1 Parts

Our total parts cost was \$1234.62, as shown in Table 9.

Table 9 Parts Costs

Part	Manufacturer	Model	Retail Cost (\$)	Bulk Purchase Cost (\$)	Qty	Actual Cost (\$)
LIDAR	SLAMTEC	RPLIDAR A2-M8	329.00	299.00	2	658.00
LiPo Battery	Turnigy	Graphene 5000mAh 4S 45C	74.32	74.32	3	222.96
Microcontroller	Atmel	ATMega2560	12.35	10.14	4	49.40
Webcam	Logitech	C310	27.49	27.49	1	27.49
Wi-Fi Chip	Espressif	ESP32 WROOM-32	3.80	3.80	4	15.20
Chassis w/DC Motors & Encoders	Mountain_Ark	Tracked Chassis	49.99	49.99	2	99.98
Motor Controller	Qunqi	L298N Motor Driver	6.99	6.99	3	20.97
IMU	Adafruit	MMA8451	7.95	7.95	2	15.90
DC-DC Regulator	Texas Instruments	TPS54290	6.03	6.03	8	48.25
USB to UART Chip	FTDI	FT232R	4.50	4.50	4	18.00
70" Tripod	Albott	70" Tripod	43.99	43.99	1	43.99
Ceramic Resonator	Murata	CSTCE16M0V53-R0CSTCE16M0V53-R0	0.50	0.5	4	2.00
100nF Capacitor	Yageo	AS0805KKX7R9BB 104	0.46	0.12	20	5.52
10 μ H Inductor	Bel Signal	SCIHP1367-100M	1.74	1.16	4	6.96
Total						1234.62

4.2 Labor

Utilizing the expected BS Computer Engineering salary for UIUC graduates, Equation 3 shows an estimated \$9,600 of labor cost.

$$2 \text{ engineers} * \frac{\$40}{\text{hour}} * 12 \frac{\text{hours}}{\text{week}} * 10 \text{ weeks} = \$9,600 \quad (3)$$

5. Conclusion

5.1 Accomplishments

At the conclusion of this project, our Ground Agents are able to collect rich LIDAR and IMU data, then process and package this data in real-time. They are able to move through their environment freely, power all necessary onboard components, and receive and act upon motion instructions that are given over Serial input. The Wi-Fi board on the Ground Agent is able to post data to an IP address on its local network. The Sky agent is able to communicate with the Controller Agent over the ROS structure implemented and provide image data with low latency. The Controller agent is able to warp the Sky Agent image to a degree necessary to localize and estimate the pose of both Ground Agents. It is also able to pull down data from a given IP address and post this data to an internal ROS topic for processing. The Controller Agent can send motion instructions to a ROS topic for posting and send or receive LIDAR data to and from a ROS topic.

5.2 Uncertainties

The central failing in our system is the lack of integration between the ATmega2560 microcontroller and the ESP-32 Wi-Fi board. We were not able to get reliable communication of Serial data over UART between these components without the use of a 5 V to 3.3 V bidirectional Logic-Level Converter, which we were not able to construct or purchase due to timing constraints regarding shipping. Our power output was also uncertain at times due to irregular behavior in at LIDAR or drive motor start-up. This was due to the use of combined generic RLC components to approximate the desired values to ensure proper regulator operation. In order to verify the correct RLC component sizes, we wrote a software script to perform the necessary calculations and generate RLC component sizing.

5.3 Ethical considerations

Our project has one main risk in ethics violation, and that's regarding privacy. At a high level, our system can map and localize any given area using multiple ground agents and an eye-in-the-sky. Unfortunately, it is possible that an individual, or a group of individuals, with malicious intent can use our technology to acquire localized map data of a property or region without consent. This is a direct violation of principle 1.6 in ACM's Code of Ethics, which states that computing professionals have a responsibility to respect the privacy of the public and other professionals. [8] Misuse of our technology is also a violation of IEEE's Ethics Code #1, which states that engineers should hold the welfare of the public paramount and strive to comply with ethical design. [6] This is reiterated in the IEEECS Code of Ethics section 3.12, mentioning that we must 'Work to develop software and related documents that respect the privacy of those who will be affected by that software. [7] This is why we ensure that all intermediate data, not including the final global map, is deleted, and that the final result of the process, a global map, will be carefully marked indicating that data was collected, reminding the user of privacy constraints with this system.

5.4 Future work

In the future, there are four main improvements we'd like to make to our project. First, it'd be very beneficial to upgrade to a higher resolution 4K camera for our sky agent. As described in Section 2.2.1, the extremely low-resolution video feed we were working with had a direct impact on the performance of the SIFT algorithm. By upgrading to a 4K camera, we'd be able to

observe significantly better performance from SIFT, which is more robust than our contour-based algorithm. Secondly, it would be helpful to integrate RViz, a visualization tool within ROS, into our ROS structure to visualize the global map being built in real-time. Thirdly, it'd be interesting to scale our project to more than two ground agents and observe what kinds of performance increase/decrease is achieved. Lastly, our project would benefit from a more ruggedized ground agent chassis because the low-cost chassis we used for our project repeatedly broke and fell apart which increased time spent repairing the physical robot. Investing in a higher quality robot chassis would vastly improve the mechanical reliability of our ground agents.

References

- [1] T. Huang, RPLIDAR A1 - Slamtec - Leading Service Robot Localization and Navigation Solution Provider. [Online]. Available: <https://www.slamtec.com/en/Lidar/A2>. [Accessed: 14-Feb-2019].
- [2] “ATmega2560 Datasheet,” *Microchip*. [Online]. Available: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf. [Accessed: 29-Apr-2019].
- [3] *PIC32MX575F256H Datasheet*. [Online]. Available: <https://www.microchip.com/PIC32MX575F256H>. [Accessed: 29-Apr-2019].
- [4] S. Santos and Jimmy, “ESP32 vs ESP8266 - Pros and Cons,” *Maker Advisor*, 29-Apr-2019. [Online]. Available: <https://makeradvisor.com/esp32-vs-esp8266/>. [Accessed: 30-Apr-2019].
- [5] *TPS54290 Datasheet*. [Online]. Available: <http://www.ti.com/lit/ds/symlink/tps54290.pdf>. [Accessed: 27-Apr-2019].
- [6] “IEEE Code of Ethics,” IEEE - *Advancing Technology for Humanity*. [Online]. Available: <https://www.ieee.org/about/corporate/governance/p7-8.html>. [Accessed: 08-Feb-2019].
- [7] “Software Engineering Code of Ethics,” *content - IEEECS*. [Online]. Available: <https://www.computer.org/web/education/code-of-ethics>. [Accessed: 08-Feb-2019].
- [8] “ACM Code of Ethics,” *Association for Computing Machinery*. [Online]. Available: <https://www.acm.org/code-of-ethics>. [Accessed: 08-Feb-2019].
- [9] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

Appendix A Requirement and Verification Table

Table 10 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
Controller Agent		
Ground agent LIDAR point cloud ROS node is recognized, and pipeline is able to pass at least 16kB of data to this node	<ol style="list-style-type: none"> 1. <i>Initialize and run ROS structure</i> 2. <i>Check that PC recognizes the node for ground agent</i> 3. <i>Push a data packet of at least 16kB in size from PC</i> 4. <i>Check that the entire data packet was received by ground agent's node without lose</i> 	Y
Ground agent movement command ROS node is recognized and receives movement commands without loss	<ol style="list-style-type: none"> 1. <i>Initialize and run ROS structure</i> 2. <i>Check that PC recognizes the node for ground agent</i> 3. <i>Push chain of movement commands to ground agent from PC</i> 4. <i>Check that all movement commands are received in order, without loss</i> 	Y
Sky agent's ROS node is recognized and pushing location and orientation data to PC without loss	<ol style="list-style-type: none"> 1. <i>Initialize and run ROS structure</i> 2. <i>Check that the PC recognizes the node for sky agent</i> 3. <i>Push location and orientation data to PC from sky agent</i> 4. <i>Check that location and orientation data is received without loss</i> 	Y
Sky Agent		
Sky agent is able to localize one ground agent and provide location coordinates and orientation angle	<ol style="list-style-type: none"> 1. <i>Achieve birds-eye view with sky agent</i> 2. <i>Ensure that ground agent markers are detected</i> 3. <i>Determine orientation angle and location of ground agent markers</i> 	Y
Sky agent is able to distinguish between 2 separate ground agents and provide distinct orientation angles and	<ol style="list-style-type: none"> 1. <i>First, make sure that sky agent is able to localize one ground agent</i> 	Y

location coordinates for both ground agents according to high-level requirements.	<ol style="list-style-type: none"> 2. <i>Filter incoming video stream to distinguish specific colors as unique ground agents</i> 3. <i>Localize each unique agent separate and push localization data for each individual ground agent</i> 	
Ground Agent Power Subsystem		
<p>Distribute 12-15 V DC ($\pm 5\%$) @ 2.5 A ($\pm 20\%$) to the Motor Controller.</p> <p>Distribute 5 V DC ($\pm 10\%$) @ 2.5 A ($\pm 20\%$) to the Microcontroller, LIDAR, and IMU.</p> <p>Distribute 3.3 V DC ($\pm 10\%$) @ 1.5 A ($\pm 20\%$) to the Hall Effect Sensors and Wi-Fi chip.</p>	<ol style="list-style-type: none"> 1. <i>Connect the system power leads to a power supply in the bottom of expected battery range, 12.4 V.</i> 2. <i>Measure the output of each of the regulator's and verify that output voltage and ripple current are within specification.</i> 3. <i>Repeat this process sweeping power supply up to 16.4 V, peak of expected battery range.</i> 	Y
Ground Agent Motion Subsystem		
Directionally Controllable via char command codes. Commands: {in-place left turn, in-place right turn, forward 10cm, back 10cm, stop}. Turns must be executed with minimal center-of-mass drift ($\pm 5\text{cm}$), motion must be executed with precision ($\pm 2.5\text{cm}$) and stopping must be immediate ($\pm 2.5\text{cm}$).	<ol style="list-style-type: none"> 1. <i>Utilize a test environment including obstacles of the scale mentioned in high-level requirements.</i> 2. <i>Log the commands from the MCU serial port and confirm that each command results in behavior in the set mentioned in Requirement</i> 	Y
Not accelerate or decelerate rapidly during operation. Not reach velocities above 1 m/s.	<ol style="list-style-type: none"> 1. <i>Power up the robot and operate in the staged environment.</i> 2. <i>Observe closely to see whether or not the robot moves in an unsafe acceleration scheme.</i> 3. <i>Observe the robot velocity measurement from Hall Effect Sensors.</i> 	Y
Be able to turn in place, staying centered on the same spot, $\pm 5\text{cm}$.	<ol style="list-style-type: none"> 1. <i>Power up the robot and operate in the staged environment.</i> 2. <i>Attempt to turn the robot 360 degrees and observe the starting and ending center locations.</i> 	Y
Ground Agent Sensing Subsystem		
IMU: Provide an accurate inertial measurement of 3-dimensional pose	<ol style="list-style-type: none"> 1. <i>Utilize a testbench circuit including the IMU and a power</i> 	Y

when completely halted, within 5 degrees, at an I2C transmission rate of 115200 Hz.	<i>supply to log the 3-axis inertial measurements on a physical testbench of 5 degree offset incline orientations.</i> 2. <i>Log data and compare logs to ground truth.</i>	
LIDAR: Provide an accurate representation of at least the immediate radial foot around the LIDAR, providing data to the control system with at least one sample per degree.	1. <i>Utilize a test environment including obstacles of the scale mentioned in high-level requirements.</i> 2. <i>Log the LIDAR data from the MCU serial port, and plot to verify the scan is representative of the environment, including all obstacles of the scale mentioned or larger, with at least one sample per degree of the scan.</i>	Y