# THE DESIGN AND IMPLEMENTATION OF AUTONOMOUS ROBOTIC SYSTEMS USING ROS

by

John William Robe

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

In

Computer Engineering

Approved:

| | |
|---|---|
| _____ | _____ |
| Angela Rasmussen, PhD<br>Thesis Faculty Supervisor | Gianluca Lazzi, PhD<br>Chair, Department of Electrical & Computer Engineering |
| _____ | _____ |
| Neil E. Cotter, PhD<br>Honors Faculty Advisor | Sylvia D. Torti, PhD<br>Dean, Honors College |

April 2017

## ABSTRACT

This research focuses on the development of an autonomous robotic system using Robot Operating System (ROS), with the purpose of competing in the NASA Robotic Mining Competition in collaboration with the Utah Robotic Mining Project team. While completing this research, the following discoveries were made: accurate sensor data is imperative to functioning autonomous systems, LiDAR data can be used to augment other senor data inaccuracies, filtering algorithms cannot always compensate for highly inaccurate sensors, and the correct integration of sensors, control systems, and specific robot parameters into autonomous robotics frameworks is a key step to enabling autonomy in ROS.

Autonomous robotics has the potential to be the largest leap in technology since the invention of the personal computer. Integrating autonomous robotic systems requires a real-time unified operating system. ROS is a catalyst in the development of fully autonomous robot applications. ROS is a open source robot software with a large community of developers and many resources available. ROS allows researchers to focus their time on implementing fully functioning autonomous systems on their robotic platform of choice instead of spending time developing functioning code bases.

TABLE OF CONTENTS

# I. INTRODUCTION

In this research two goals were achieved: first, to design and build a functioning robotic platform, and second, to make that robotic platform function autonomously using ROS. This research was done in collaboration with the Utah Robotic Mining Project [1] and a senior mechanical design team at the University of Utah. The research discussed in this paper, unless expressly stated, is the author's work.

With the rapid growth of autonomous robotics there must also come an improvement in the ability to design and develop them. That improvement comes in the form of Robot Operating System (ROS) [2]. Using ROS was a decision made during the design process, which allowed this research to be completed. Because of this decision, much of this research became how to integrate electronics, sensors, and mechanical design decisions into the ROS framework quickly and with maximum efficiency.

The robotic platform is designed to compete in the NASA Robotic Mining Competition [3]. This competition consists of an arena filled with Martian soil simulant (BP-1) [4] the ultimate goal is to design and build an autonomous robot capable of mining and depositing as much BP-1 as possible. The competition was used as a basis for the design decisions made in Section II and the goal for autonomous operation was to complete a NASA competition run without any user input.

The methods section, Section II, will focus on the design, selection, and development of the robotic platform including what components were selected to achieve autonomous operation. The results, Section III, will discuss how well the autonomous system was able to function on the robotic platform; and the discussion section, Section IV, will discuss why the autonomous system performed how it did and make suggestions for improvements.

# II. METHODS

In order to design an autonomous robot one must define the functionality and constraints first. Systems would then be identified and developed to achieve its autonomous

mission. Each system would have specific requirements. Mechanical systems would be designed to overcome the physical demands. Electrical systems would provide power and communication. Autonomous systems would be programmed to control the mobility platform and, at times, the communication systems. This paper will study the specifications and requirements needed for achieving autonomy in robotics while using the example of lessons learned during the process of building an autonomous mining robot for the NASA Robotic Mining Competition.

Further examination of the aforementioned autonomous robot that was designed for participation in the NASA Robotic Mining Competition [3] requires an understanding of the strict rules and well defined goals set out by NASA. The strict rules allowed the team to create a well-defined system that cleared the path for overcoming further design decisions throughout the process. The design criteria from NASA are as follows:

- The robot must weigh no more than 80kg.
- The robot must fit within 1.5m x 0.75m x 0.75m.
- The robot must be able to navigate and mine Martian soil simulant (BP-1) [4].
- The robot must be designed to lift 10kg of regolith material 0.5m above ground level to deposit BP-1 into the collector bin.
- The robot should be able to dig, move, and deposit without any user input (autonomously).
- Power consumption and Bandwidth usage are somewhat constrained and should be minimized whenever possible.
- The competition arena is an enclosed area 15m x 8m x 1m. It is surrounded by walls, however the walls must not be used to aid any autonomous systems, as that would result in disqualification from the competition.

Using these design rules, a mechanical design was created that would allow the robot to complete all of these tasks. Initially, design decisions for the mechanical systems are critical as they set limits on sensors and other electrical components, which is required before the autonomous system programming can be completed. The mechanical design

process will not be covered in-depth, rather a focus will be placed on the electrical and autonomous systems. However, because robots are inherently mechanical systems along with the electric and autonomous components, a high level review of the mechanical systems will be necessary to give context for the rest of the discussion. The mechanical design, fabrication, and Solidworks models were completed, with direction, by a senior design team in the Mechanical Engineering department at the University of Utah.

When the mechanical design was completed, all sensors, control systems, and computational platforms were specified and designed properly to enable fully autonomous operation. There are a great deal of options when it comes to all of these selections, each pros and cons, which will be discussed in Sections II-B and II-C.

Once all of the sensors, control systems, and computing platforms were locked-in, the systems were modified, programmed and designed to work with the autonomous robot framework known as Robot Operating System (ROS) [2]. Using ROS gives the engineer an advantage over other frameworks that have been used in the past, which will be thoroughly revealed in Section II-D.1.

After all of the systems were integrated into ROS, the final step was to program the goal creation and autonomous procedures that dictate the fully autonomous aspect of the robot. Given the requirements from NASA [3] for the autonomous mining robot, the autonomous system was programmed to navigate the arena, driving to the opposite side, collect regolith in the designated digging area, and finally return to the collector bin to deposit the regolith. The process would need to run for the alloted ten minutes without any further user input and cycle as many times as possible.

### *A. Mechanical Systems*

All of the mining robot's mechanical systems were first modeled in Solidworks to ensure that they would meet the requirements listed in NASA's design criteria [3]. Once it was modeled all of the mechanical components were specified giving the estimated weight and size constraints for the electrical systems. This specific mechanical design had a relatively large enclosed chassis, where all of the on-board computing and electrical

components would reside. The Solidworks renderings for the robot can be seen in Figures 1 to 5.

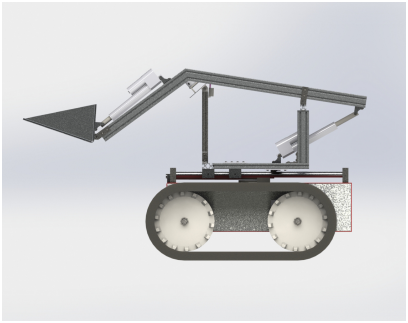


Fig. 1. Side View

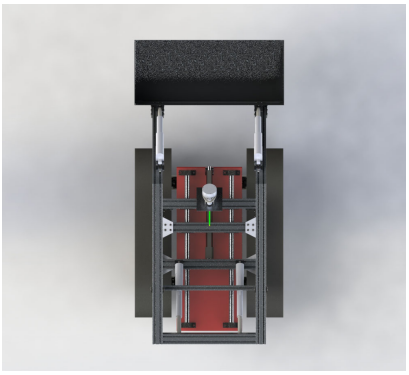


Fig. 2. Top View

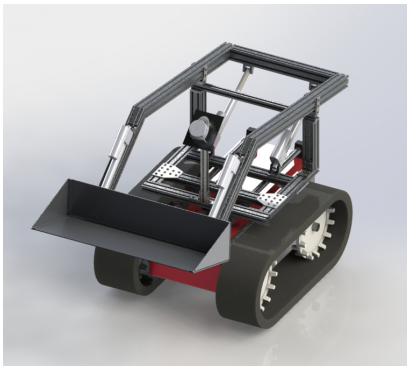


Fig. 3. Isometric View

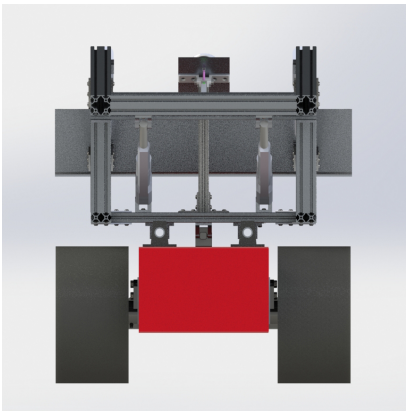


Fig. 4. Rear View

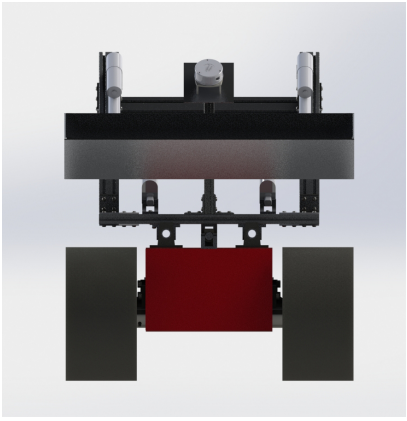


Fig. 5. Front View

*1) Mechanical Failure Detection:* With the development of fully autonomous systems, sensors were required that would provide feedback to the on-board computer to know if a mechanical system had ceased functioning, or failure would be eminent without some correction. Without proper detection, the autonomous system could give a command to a control system that would cause even more damage to the mechanical system (e.g. lowering the robot arm into the chassis when the sled has failed to move forward enough). Mechanical system failures can be detected through proper implementation of different control systems (e.g. motor current or encoder feedback), however sensors with the sole purpose of detecting mechanical failures can also be used.

On the robot that was developed during this research, two active mechanical structure sensing devices were employed. These sensors were strain gauges designed to detect the load on the robot arms (one device for each arm), for when regolith would be loaded into the bucket. Using the values received from these strain gauges it enabled the ability to determine how many kilograms of regolith was in the bucket and whether or not the autonomous system would need to dump regolith out of the bucket, to be able to lift it to the required 0.5m height.

Other sensing devices could be employed such as force sensing bolts [5], resistive force sensors [6], or visual systems pointed back at the robot from an arm, such as the MAHLI system [7] that NASA's Curiosity rover uses to do self checks, although those would be difficult to analyze autonomously in real time.

### *B. Control Systems*

The control systems of the robot are what determine how well the autonomous systems will function. Without good sensors, motors, embedded systems, and computing platforms, the autonomous system should not be expected to be able to send commands to the control systems that will achieve a desired result (outcomes of low quality sensors can be seen in Section III). Based on the Solidworks model and the estimated weight of the robot, it was determined that the robot would need high-torque drive motors and sufficient motor controllers [8]. This was also needed to be able to move the robot and

30kg of regolith over uneven terrain. High load linear actuators would also be required to lift the arms and actuate the bucket. Another high load linear actuator was needed to operate the Sliding Linearly Excavating Digger (SLED), which adjusts the center of gravity of the robot, based on how much material is in the bucket of the robot.

*1) On-board Computing:* Two on-board computing systems were purchased to test for the robot. The first system was an ARM-64 based NVIDIA Jetson TX1 [9]. This computing system features high performance and great GPU capabilities (1 Teraflop graphic processing) at a very low power consumption and form factor.

The other computer that was tested was an Intel NUC [10] (Intel x64 i7 processor, Intel HD 6500 Graphics, 8GB Ram), a very small form-factor desktop computer.

The recommended operating system for the ROS framework (Jade Turtle) is Ubuntu 14.04, which was loaded on both machines and ROS was installed on both. The Intel NUC was eventually selected over the NVIDIA TX1, due to the Intel Realsense 3D IR camera's II-B.4 dependency on the Intelx64 architecture. All electrical systems and computation for autonomy could then be done in real-time, on-board the robot through the Intel NUC.

*2) Motor Controllers:* Phidgets motor controllers [11] were selected to drive all 8 motors on the robot because of their prior success, relative ease of use, and the USB architecture. These motor controllers would connect to an onboard USB hub, then connect directly into the Intel NUC. Each motor controller includes inputs for quadrature and analog encoders and were set up to publish encoder data to ROS in real time. The motor controllers worked simply by taking a duty cycle percentage and outputting that duty cycle (in each direction) to the motor. Therefore, any calculation of required velocity or distance traveled would need to be done in the autonomy framework, instead of on the motor controller itself.

Other motor controllers would have acceptable, or even have better functionality, such as the RoboClaw 2x30A [12] which contains on-board PID and has the ability to travel at a set velocity; however for this robot, the Phidget motor controllers gave us the minimum

required functionality.

*3) Feedback Sensors:* In order for the robot to be autonomous, all of the motors need to provide feedback of their positioning/rotation. This allows for ROS to create an accurate 3D model of the robot configuration, it then uses this for pathfinding and mapping later.

For the four drive motors, optical quadrature encoders [13] were selected to provide wheel odometry. These had an accuracy of 300 cycles per revolution (CPR) which was more than enough for the low RPM system. The linear actuators [14] include analog output potentiometers which accurately represent how far the linear actuator is extended, $\pm$ 2mm, which was acceptable for the competition purposes.

An extremely important feedback sensor that should be equipped to all autonomous robots, and was included on the robot for this project, is an Inertial Measurement Unit (IMU) [15]. These are essential in providing accurate movement data to the Autonomy Framework. IMU's are necessary because the wheels and motors of the robot might be rotating, but due to terrain slippage it's entirely possible that the robot may not be moving. The IMU allows for the robot to correct for terrain slippage, or notice that it isn't moving as expected compared to the encoder feedback.

Other feedback systems include the strain gauges that were discussed in II-A.1 and one servo motor that was used to control the tilt of the LiDAR (II-B.4).

*4) Autonomy Enabling Sensors:* The feedback sensors discussed in II-B.3 are necessary for autonomy, however the autonomy enabling sensors that allow detection of the environment around the robot, costmapping, and 3D visualization, will be discussed in this section.

For this robot, a combination of two systems were selected; first, a 360° scanning LiDAR array would be mounted on top of a detection post and second, a 3D infrared camera. The LiDAR [16], mounted on a platform that would tilt up and down, would provide a three dimensional point cloud from a single planar sensor. This tilt was implemented, however the code to convert the two dimensional scan to a time-delayed three dimensional point cloud with robot movement offset was prohibitively difficult to

complete in the scope of this project. Because of this, the LiDAR was kept flat on the top of the platform and used to scan for higher obstacles.

The Intel Realsense 3D infrared (IR) camera [17] was used to generate a point cloud of distance data. This sensor enabled the robot to see what was in front of it. There are disadvantages to using an IR camera and other options could significantly have improved the vision systems for this robot, and they will be discussed in Sections III and IV-D.1.

### *C. Electrical & Communication Systems*

The electrical systems consisted of 25 different components that all needed to be wired together to function. A complete diagram for reference can be seen in Figure 6.

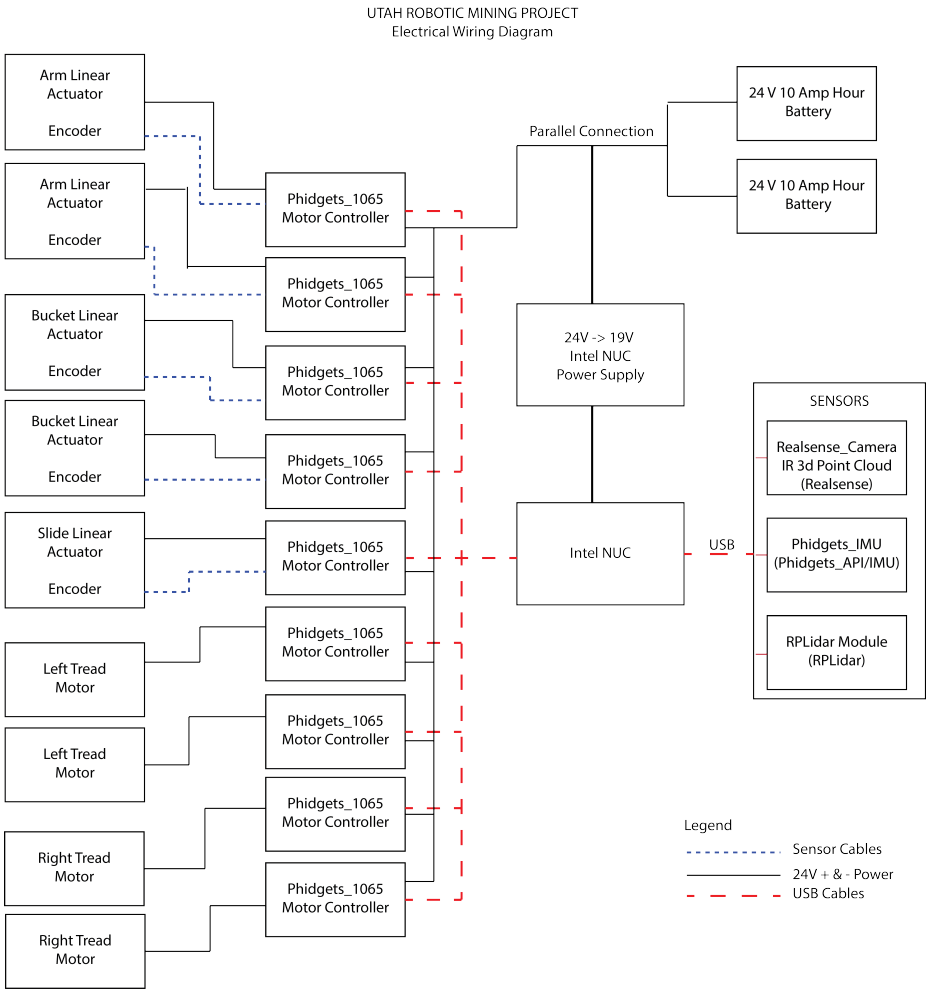



Fig. 6. Electrical Component Wiring Diagram

In Figure 6 the leftmost column lists the motors and encoders that the robot needed to drive. To the right of that, are the Phidgets motor controllers, that connected via USB to the Intel NUC on-board computer. Additionally, the motor controllers required their own individual connection to a 24V power supply. This power was provided in the form of two 24V NiMH 10AH batteries connected in parallel.

Finally the three sensors discussed in Sections II-B.3 and II-B.4 are connected and powered by USB, through the Intel NUC. The Intel NUC required an 18V power supply, so a DC-DC converter board [18] was used to facilitate this requirement.

*1) Electrical Requirements:* Due to the very large number of electrical systems, it was determined that a Printed Circuit Board (PCB) was required to connect all of the components together to the power supply. This PCB was designed and fabricated as detailed in II-C.2.

Several other requirements were given by the NASA rule book [3], from determinations based on those requirements other component selections were made.

- The electrical systems will be based on a 24V power supply.
- The robot must have an emergency stop switch, to cease all power to any system immediately when pushed.
- The battery technology used must be safe to charge and to use.
- All connections must be made via quick disconnect connectors; allowing the electrical components to quickly be taken out and serviced if needed.
- A power meter, to measure the power consumption of the robot, must be connected between the batteries and the power distribution board.

The decision to include a 24V Power system was made based on the motor and motor controller selection. All motors and linear actuators that were specified during the mechanical design phase were 24V systems, and as those consist of the highest number of systems and energy use on the robot, it was logical to have a 24V power supply to reduce power efficiency loss when converting.

The emergency stop switch is a requirement for the NASA competition, but also a

good idea for any robot (or car) being designed with autonomy in mind. The stop button was placed on the back of the robot, allowing for any person to safely press the switch to cease all robot functions.

The battery technology selected was NiMH batteries. They would be placed in parallel primarily for redundancy and secondarily for increased robot battery life and thus runtime. NiMH was chosen specifically due to the decreased chance of thermal runaway and increased safety in charging compared to LiPo or Lithium Ion batteries.

The power meter [19] was used due to the NASA rules to report how much power was used during one competition run. The less power used, the better for the competition scoring as seen in the NASA competition rules [3].

Lastly, the communication system for the robot was designed to connect to a 2.4GHz Wireless router, set up relatively near the robot. The Intel NUC had an integrated chip to connect to the router, however the antenna connection was extended and routed to an external point on the robot, thus improving wireless signal.

*2) PCB Design:* The power distribution printed circuit board (PDPCB) needed to accomplish three things, allow for parallel battery connection, isolate the battery from the rest of the system until an emergency stop switch was pressed (using a relay) and create a path for all current to travel through the power meter to get accurate power consumption measurements.

The schematic for the PDPCB was designed as seen in Figure 7 and then placed and routed as seen in 8. Once that was complete, the PCB was fabricated and all the components were soldered on as seen in Figure 9.

*3) Robot Communications:* The communications system is not ideal for robots that require long distance reliable communication, as the 801.11 A/B/G/N protocols do not allow for long distance communications without directed high-gain antennas and is prone to channel interference. This type of communication was chosen simply because of the NASA requirement of any wireless communication to be done over an off the shelf wireless router [3].

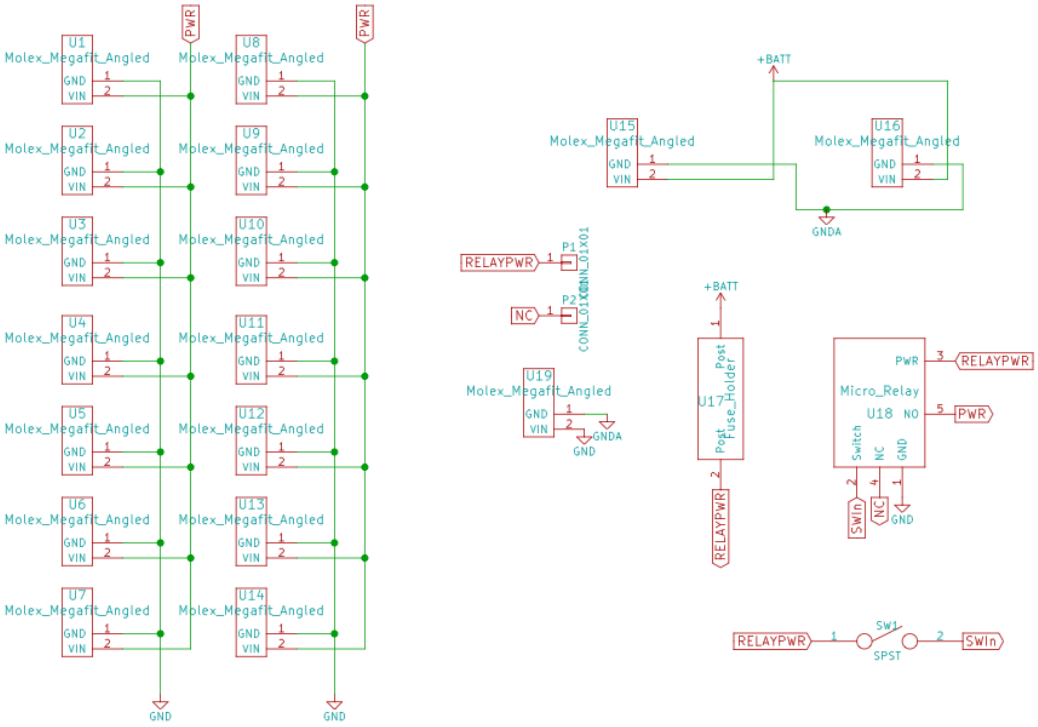



Fig. 7. Power Distribution PCB Schematic

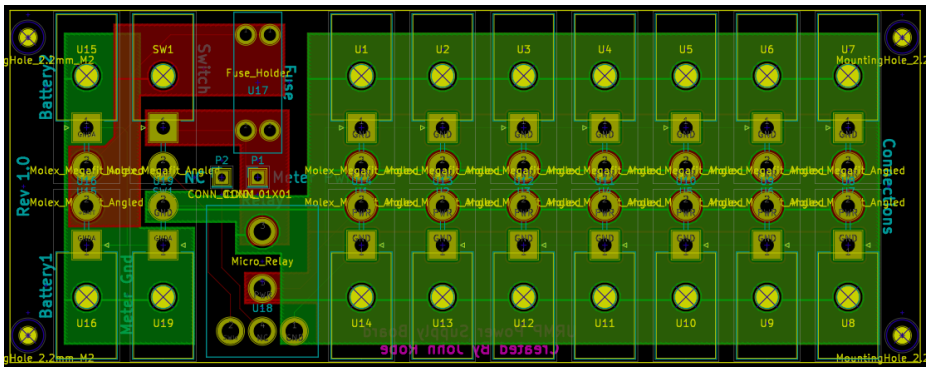



Fig. 8. Power Distribution PCB Footprint Layout

## *D. ROS Software Design*

ROS is a robust framework that allows developers to create different software ”nodes” that interact with other ROS nodes through the use of defined topics. Each topic contains important information about the robot, such as ”/odom” which contains information about the robot odometry, or ”/scan” which is the 2D LiDAR point array. These topics contain defined messages, for example the ”/odom” message is so important it has it’s own defined message type which is an ROS Odometry message. This message is formatted to

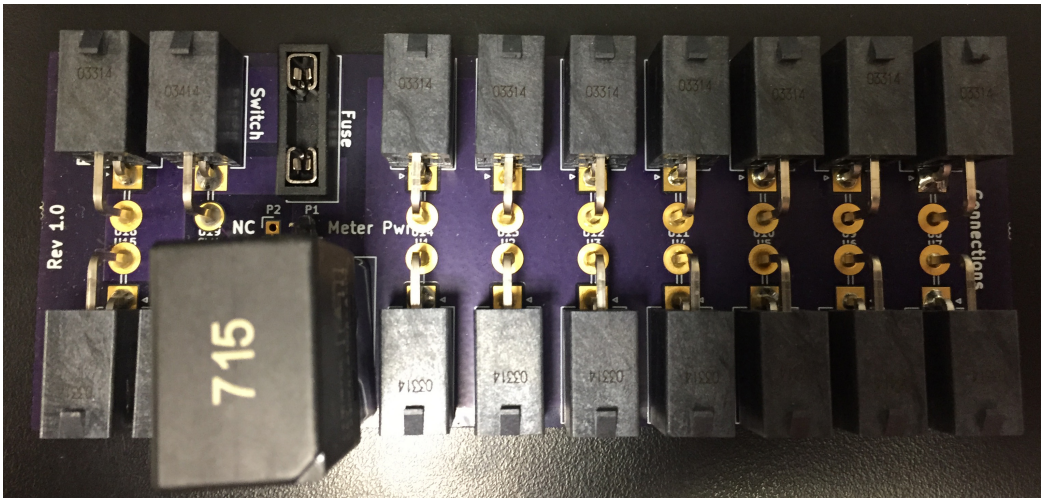



Fig. 9. Power Distribution PCB Completed

contain a header with a timestamp, a pose, and a twist message, which allows for other ROS nodes to know all of this information.

ROS also contains automatic frame transformations. This is an important concept in terms of sensor data integration. The reasoning behind this is that on a moving robot, each sensor will not be in the exact same position. One sensor (say the LiDAR) could be mounted to the top of a post near the front of the robot and may slide forward and back throughout the normal operation of the robot. But without frame transformation, there would be no way to use that information to convert the data to the center, or as ROS defines it, the ”base_link” frame of the robot.

Finally, due to the popularity of ROS, there are many nodes already written to do things such as Simultaneous Localization and Mapping (SLAM), which enables 3D environment recognition and pathfinding. One such node that was used is called RTABMap [20]. This will be discussed more in Section II-E.1.

*1) Autonomy Framework: Selecting ROS:* In continuation of the discussion of ROS, further investigation on its usefulness as an autonomous framework will be discussed. Along with the features discussed in Section II-D, ROS is specifically designed to run on robotic systems and communicate with a large number of devices. Also, as it runs on Ubuntu, it can be installed onto most architectures (which is an important consideration

with the NVIDIA Jetson TX1, as it is Arm64 based). All of these are important features, but before it was selected as the framework two other Robotics frameworks were analyzed.

The initial framework analyzed to run the autonomous systems was Microsoft Robotics Developer Studio (MRDS) [21]. This development framework contains many of the same features of ROS, but some critical system requirements were not acceptable. The first requirement is that the Robot must run on Windows 7. Because Microsoft had not released a Windows 7 install for ARM architecture at the time of selecting this framework, it would have ruled out the NVIDIA TX1 as a computation platform. The other limitation of this framework was that it made assumptions that a Microsoft Kinect [22] sensor would be used as the 3D vision system and while this sensor would have worked with the Intel NUC, it again would not have worked with the TX1. Additionally, the framework was designed for the Kinect v1 hardware, which is an inferior sensor to the Kinect v2 also released at the time. Because of these requirements, MRDS was ruled out.

The next framework was The Orocos Project robotics framework [23]. This framework has little requirements on the operating system used, mainly that it requires the ability to compile C++ code. It also contains many of the same frame transformation and kinetic library tool functionality as ROS. The Orocos project was ultimately ruled out because of the lack of documentation and support compared to ROS.

*2) Separation of Concerns:* With ROS selected as the framework for this project, full advantage was taken of the Separation of Concerns node structure. With this structure ROS recommends developers to split up their project into separate files. This allows developers to organize their code-base into different ROS nodes, for different aspect of robot functionality.

Using this structure, eleven unique nodes were identified as necessary to run the system. Some of these nodes were already developed and published for open source use and other nodes would need to be developed. The diagram of how all of these interact can be seen in Figure 10

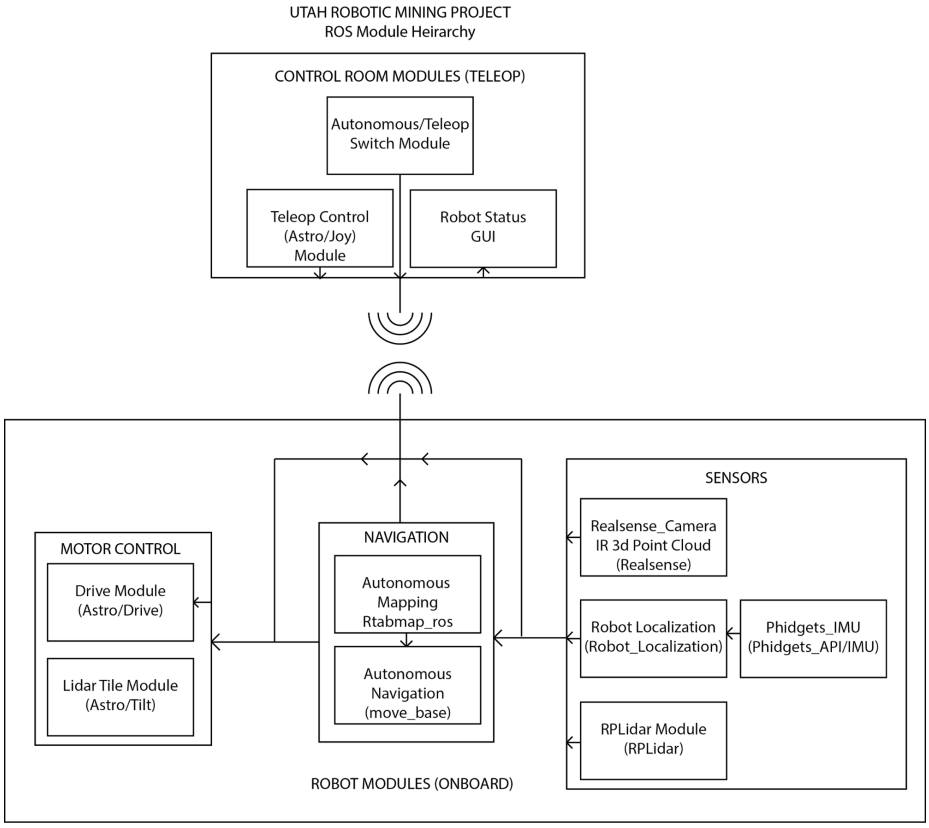



Fig. 10. ROS Module Hierarchy

*3) Node Definition & Development:* ROS nodes are pieces of software (developed in C++ or Python) that control a single aspect of the robot. Each node is developed to conform to the ROS software design methodologies and interface with the ROS framework. To create the functioning autonomous system as stated above, eleven ROS nodes were used. These ROS nodes are defined below.

The separation of concerns aspect of ROS is also applied to how it handles the relation of different components of the robot and how they compare to each other in 3D space. These are called ”transform frames” or usually shortened to TF’s. When creating autonomous systems, all frames located on the robot must be related to the robot’s ”World” frame. A visualization of all of the transform frames on this robot and how they relate to each other can be seen in Appendix D.

Drive: This node was responsible for taking velocity topic messages and setting the corresponding velocity values on the respective Phidget motor controllers using the

Phidgets ROS framework [24]. Drive also was responsible for outputting an odometry message to describe the movement of the robot treads which is an important aspect of autonomy. This node had to set definitions for each Degree of Freedom (DoF) on the robot and synchronize the movement of the motors for each DOF. This was necessary because nearly every DoF had at least two motors driving the movement. Drive was developed by Samuel Zachary, a member of the Utah Robotic Mining Project team [1].

Tilt: This node was straightforward, it took in no messages and was developed to drive a single servo to control the tilt of the LiDAR. This node was also required to publish a ROS transform between the base_link frame, and the LiDAR frame, so that the tilting of the LiDAR was represented in the SLAM Map representation.

RTABMap_ROS [20]: The development of this node was done by other research groups, but the implementation of the module with the systems was an important and time consuming process. Once implemented this node publishes a costmap for path-finding, a 3D map of the traversed world, and several transforms between the base_link of the robot and the map representation of the world. This is the SLAM module that required all sensor data to be input and all transforms to be defined for it to work. More on this will be discussed in Section II-E.1.

move_base [25]: This node takes in a costmap, a move goal (x,y coordinate) and does path-finding to see if 1. A movement to the goal is possible and 2. What velocities need to be provided to navigate the robot to that goal. What this node publishes is a ”/cmd_vel” topic which the Drive module subscribes to. This will be discussed more in Section II-E.2.

Realsense_Camera [17]: Developed by Intel, this node connects to the Realsense camera via USB and publishes a point_cloud for use by other modules. This node contains a number of different launch parameters, which have the ability to change the sensor accuracy and data points. These were all configured to give a reasonable output.

Phidgets_IMU [26]: This is a ROS Node previously developed that allowed the use of the Phidgets IMU sensor [15]. This published the complete raw accelerator and gyroscope

data from the IMU for use in other ROS systems. This data required filtering, which is why it was put into the Robot_localization node as seen in Figure 10.

RPLiDAR [27]: This is a pre-made ROS Node developed by the company that makes the RPLiDAR sensor. This node connected to the sensor over USB and published a 2D LiDAR scan with distance data of what the LiDAR was detecting. This sensor data was pushed directly into RTABMap.

Joy: This node connects to a joystick via USB and publishes velocity messages to Drive for manual motor control. This module needed to have the mappings from the joystick to each DOF for the robot built in, because Drive was made to be generic to accept velocities for DoFs. This was designed to work with an XBox 360 controller [28] by mapping each controller input to a specific robot movement. Joy was developed by Cole Mortensen, a member of the Utah Robotic Mining Project team [1].

Autonomous/Teleop Switch Module: This node subscribed to the joystick topic listening for a specific button press and published a boolean value on whether or not the robot should be running on Teleop control or autonomous control. This was designed to be the safety system in case the autonomous functions were not performing optimally.

Competition: This node was designed to create move goals, necessary for fully autonomous operation during the NASA competition run. This node also defined procedures for robot digging and robot dumping, which gave specific commands for controlling the arm and the bucket to Drive. This will be discussed more in Section II-E.3.

Robot Status GUI: ROS provides a very good 3D visualization system called RVIZ; a custom view was defined in RVIZ to display all necessary information for robot control in the 3D representation.

### *E. Autonomous Control*

Autonomous movement and control in ROS is enabled using multiple frameworks, the first of which is Simultaneous Localization and Mapping (SLAM). SLAM enables the robot to create a 3D representation environment around itself and compress those 3D points onto a two dimensional plane, called a costmap. The localization aspect of SLAM

is one of utmost importance and will be discussed more in Section III-A.1, but what it enables is for the robot to understand how it is interacting with its own representation of the world, e.g. the robot must know how it is moving around relative to all of the points of data it's collecting in order to do any kind of autonomous movement.

With SLAM and a correct costmap, the autonomous movement is enabled by "move goals". These are necessary because the robot, even if it knows the environment around itself, can't have any way of knowing where it should go, unless it is programmed to. Once a move goal is given to the robot, the slam costmap feeds into the the path-finding node and allows the path-finding node to output velocity values to attempt to move the robot to the move goal.

*1) Simultaneous Localization and Mapping (SLAM):* SLAM is the most important aspect of autonomy, it is essentially the eyes and brain for the robot. The SLAM system on this robot was implemented, as discussed before, using RTABMap [20]. This ROS node was developed to be an all-in-one SLAM system, which would enable the input of both LiDAR, Stereoscopic Cameras, or generic 3D point clouds (which enables the use of other sensors, like the Intel Realsense sensor used on this robot). RTABMap does all of the processing on the sensor input and it creates a map topic which it publishes for use in the Costmap and Path-finding node II-E.2. It also creates two occupancy grids, one for local costmap planning, and one for global planning. These two grids are created and correlated together; the global map deals with the historical data of all of the data that was ever put into the RTABMap system. The local costmap is created using real-time sensor data and changes as the sensor data changes.

RTABMap uses the local and global cost maps, localization data, and features recognized on the two maps to run a loop-closure algorithm. This algorithm correlates the differences and changes between the local and global maps to reposition and refine the entire 3D visualization system to make the data more accurate.

Dealing more with localization, RTABMap accepts data from an odometry topic and uses that data to adjust the positioning of the local and global maps that it generates.

Because of this, the odometry information sent to RTABMap is extremely important. If that odometry data is inaccurate, then all of the mapping and loop closure algorithms that RTABMap uses will not work properly and will fail.

Images of well functioning RTABMap visualizations can be viewed in Appendix A.

*2) Costmap Projection and Path-finding:* With the SLAM node implemented and the map output from SLAM the ROS move_base node [25] can be used to take that map and run path-finding algorithms on that node. The algorithms are highly customizable and take many parameters into consideration. The most important of these parameters are the definitions of the size and outline of the robot. The path-finding algorithm uses this information to be aware of the required distance between detected obstacles to navigate the robot through.

Move_base, along with the parameters, requires a ”Move Goal” to be set. This move goal comes in the form of an X,Y coordinate - this coordinate must be constrained within the boundaries of the currently known global costmap; if it is not, it is rejected. This is important, because it means that robot move goals must be relatively small on a newly created map. Move goals also cannot be automatically created with any of the autonomy nodes that have been discussed thus-far. These nodes have to be created based on the movement requirements of the robot as a whole. For the NASA competition, these movement goals would correspond with the procedure for moving from the start position to the digging area, then back to the collector bin to deposit the regolith. This procedure repeats until the time limit is over.

Finally, once Move_base has a goal and has created a path to this goal; it will output a velocity command. This velocity command is given to the drive motors and will move the robot along the calculated path.

*3) Autonomy Goal Setting:* As discussed briefly in Section II-E.2, autonomy goal setting is the part of autonomy where the robot applies it’s SLAM and path-finding to accomplish the required goal for the robot. For this robot platform, the goal setting was designed to move 6 meters away from the collector bin, dig while moving slowly for one

meter, and then drive back to the collector bin and then deposit the regolith; repeating this cycle until the 10 minute competition round has elapsed.

These move goals have to be calculated with relative positioning, to and from some object, in this case, the collector bin. In the competition, the exact starting position of the robot relative to the goal point is random. Some other kind of localization is needed because of this, specifically from the collector bin to the robot. This need is discussed more in Section IV-A.

The other requirement is that move goals must be within the currently defined map space. Because of this, the robot must be programmed to make smaller move goals, or to make rotation move goals to first attempt to increase map definition. An example autonomous program flow to move ten meters behind the robot would be to first turn the robot around, move three meters forward to get close enough to detect the correct goal location, and then set a final move goal to the now ”visible” location.

Images of the robot moving to goals in the 3D Visualization environment can be viewed in Appendix C.

## III. RESULTS

In this section, the performance of the robot platform defined in the Section II will be discussed. How well the autonomy worked will be discussed in Section III-A, the electrical system in Section III-B, control systems in Section III-C, and the mechanical systems in Section III-D. Each of these robot subsystems will be evaluated in terms of the first how well it performed in the context of a NASA competition and it will also be evaluated in terms of how well it would perform as a more generic robotic platform. This is an important distinction, because the NASA competition imposes several requirements or penalties that make sense for robots traveling to different planets, but back on Earth some of the requirements are not necessary or expected.

| Metric Name | Metric Question | Metric Score (integers) |
|---|---|---|
| Success Analysis (Boolean) | Did the system accomplish the requirement or not? | 0 or 1 |
| Achievement Score | How close was the system to accomplishing the goal? | 0 to 10 |
| Time Constraint Analysis | Did the system produce a result within a time constraint? | 0 to 10 (slow to fast) |
| Performance Analysis | Did the system produce the most optimal result? | 0 to 10 (not optimal to perfect) |

TABLE I
AUTONOMY PERFORMANCE METRICS

### *A. Autonomy Performance*

The autonomous systems did not perform fully to receive points for the competition runs during the 2016 NASA competition. The reasons for why it did not perform well will be discussed in Section IV, in this section the focus will be on how well the systems performed. In order to discuss the autonomous systems performance it is important to begin by talking about what metrics were used to measure performance.

Each system component used will be evaluated according to the metrics in each subsection below.

*1) Metric Definition:* Autonomy performance is difficult to measure without being qualitative. Because of this, an attempt to define different metrics that will give more definition to the qualitative analysis was made. The metrics are outlined in Tables I and II:

*2) SLAM Results:* Qualitative Metric Scoring as defined in Tables I and II: (note, these scores include the performance of of the sensors used and not only the performance of RTABMap)

- Success Analysis - 0 - Did not perform adequately
- Achievement Score - 3 - Didn't work in most environments
- Time Constraint Analysis - 8 - Fast enough for real time operation
- Performance Analysis - 4 - Many errors in mapping

The robot's SLAM system did not perform to the requirements needed for the competition, or even well enough for autonomous movement or environment mapping. The problem

| Metric Name | Score Range | Description |
|---|---|---|
| Success Analysis (Boolean) | 0 | The system did not meet the necessary requirements for successful operation. |
| Success Analysis (Boolean) | 1 | The system performed adequately. |
| Achievement Score | 1-3 | Does not work the majority of the time. |
| Achievement Score | 4-6 | Works some of the time, but is unstable and not completely reliable. |
| Achievement Score | 7-10 | Works the majority of the time, and is stable enough for functional autonomy in most environments. |
| Time Constraint Analysis | 1-3 | Is too slow for real-time operation. |
| Time Constraint Analysis | 4-6 | Is a potential slow down point, but operates most of the time. |
| Time Constraint Analysis | 7-10 | Is fast enough for real-time robot operation. |
| Performance Analysis | 1-3 | The system is inaccurate, prevents usable autonomy, and could potentially create unsafe situations. |
| Performance Analysis | 4-6 | The system is inaccurate, but autonomy may still function. Does not create unsafe situations. |
| Performance Analysis | 7-10 | The system produces results usable for fully autonomous operation. |

TABLE II
AUTONOMY PERFORMANCE METRIC SCORING RUBRIC

was not with the autonomy node (RTABMap), but instead with the sensors that were used to enable autonomy. Because of the sensors used, the map data from the RTABMap node would become corrupted. This type of corrupt visualization can be seen in Figure 11 and should be compared to the figures in Appendix A for reference. Additionally, it should be noticed that localization on this map is impossible, because there are no valid sources of odometry to correctly detect movement. RTABMap worked much better when multiple sources of odometry were used and specifically did well in small, well defined test courses. One additional odometry source was tried. This was the LiDAR system with optical flow odometry analysis on that LiDAR data [29]. Using this localization system, the results improved to the scores below:

- Success Analysis - 0 - Did not perform adequately
- Achievement Score - 6 - Would work well in specific environments
- Time Constraint Analysis - 8 - Functioned fast enough for real time operation
- Performance Analysis - 6 - The overall map would sometimes get errors

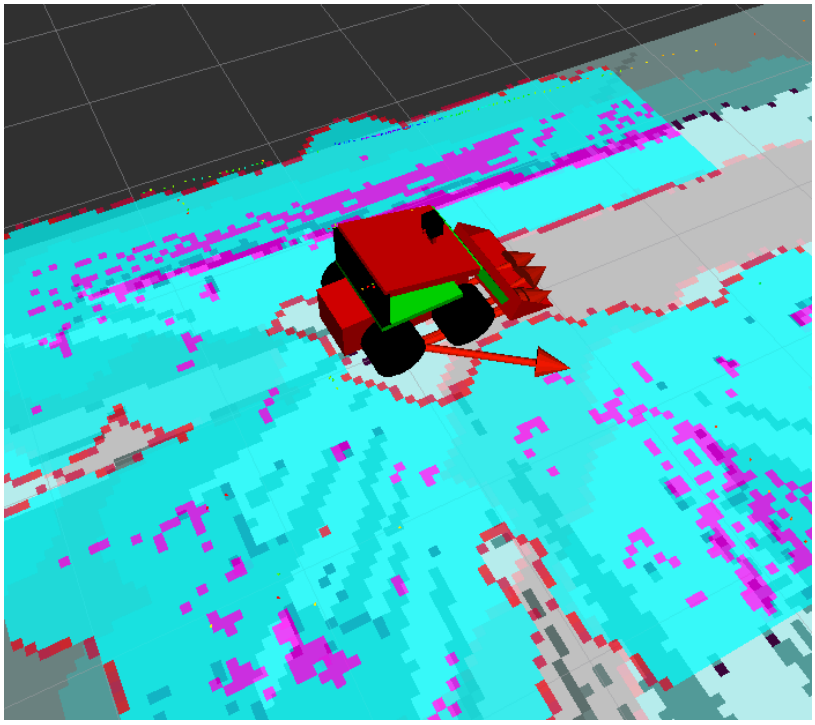


Fig. 11. RTABMap Visualization without LiDAR Odometry

Even with this small localization change, it improved the autonomous systems by a noticeable amount. It made the the mapping accurate unless the robot was in an environment without very many features to use for the Optic Flow algorithms. An example of the problems with low-feature environments can be seen in Appendix B. At the Z-level of the sensor, the NASA competition arena contains no features that the robot is allowed to detect for localization, therefore Optical Flow would not be possible. The figure shown below is of the lab and hallway with optical flow odometry enabled in Figure 12. Due to the issue of areas without noticeable features, this result cannot be given a success analysis score of 1. Overall, the SLAM system did not create 3D visualizations accurate enough to be able to have a robot physically navigate through them, because of this system failing the robot would not be able to travel autonomously.

*3) Costmap Results:* Qualitative Metric Scoring: (note, these results assume that the map given to the costmap node is a ground truth model, see Tables I and II for explanation)

- Boolean Success Analysis - 1 - Functioned Adequately

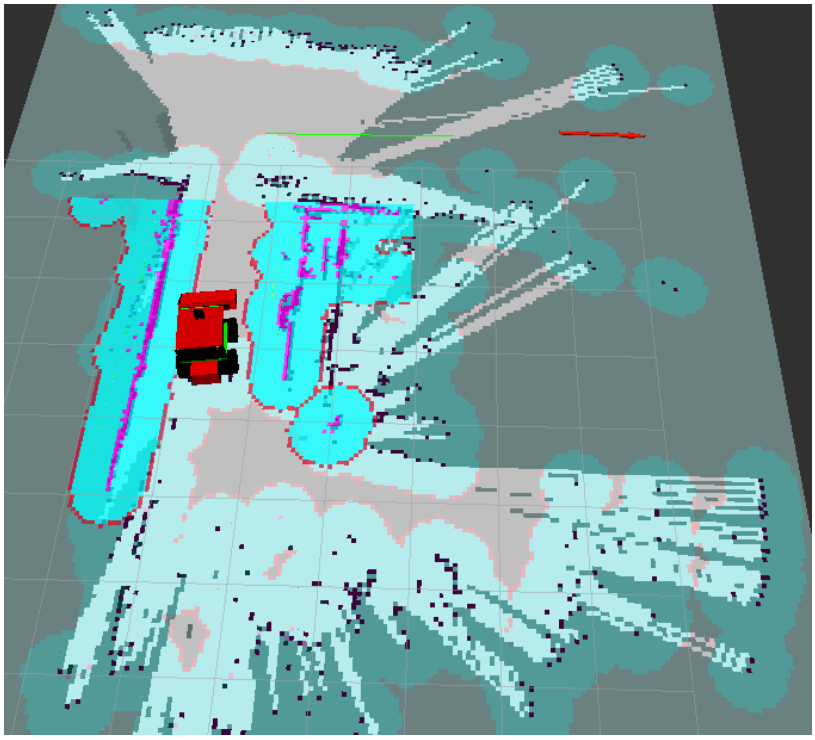


Fig. 12. RTABMap Visualization with LiDAR optical flow

- Achievement Score - 9 - Completed most of the assigned tasks, if they were possible
- Time Constraint Analysis - 8 - Functioned fast enough for real time operation
- Performance Analysis - 8 - Routes and velocities were done well, however some recovery behaviors could be fixed.

Given the scores above, it can be determined that the costmap and path-finding systems do their job well. These systems are robust and have been developed for quite some time, as the same algorithms are used in video games for AI path-finding and map navigation. Ultimately, given a well constructed map and a movement goal within the boundaries, the Move_base node will correctly move the robot through the environment when properly configured (e.g. when the path-finding algorithm is given data about the robot's size and driving characteristics). See Figure 13 for a visualization of what the algorithm develops.

In the tests run on this system, if acceptable map input was given to the path-finding algorithm and there was a move goal set, it would find the fastest route there and broadcast proper velocity values for the robot to travel along that route.

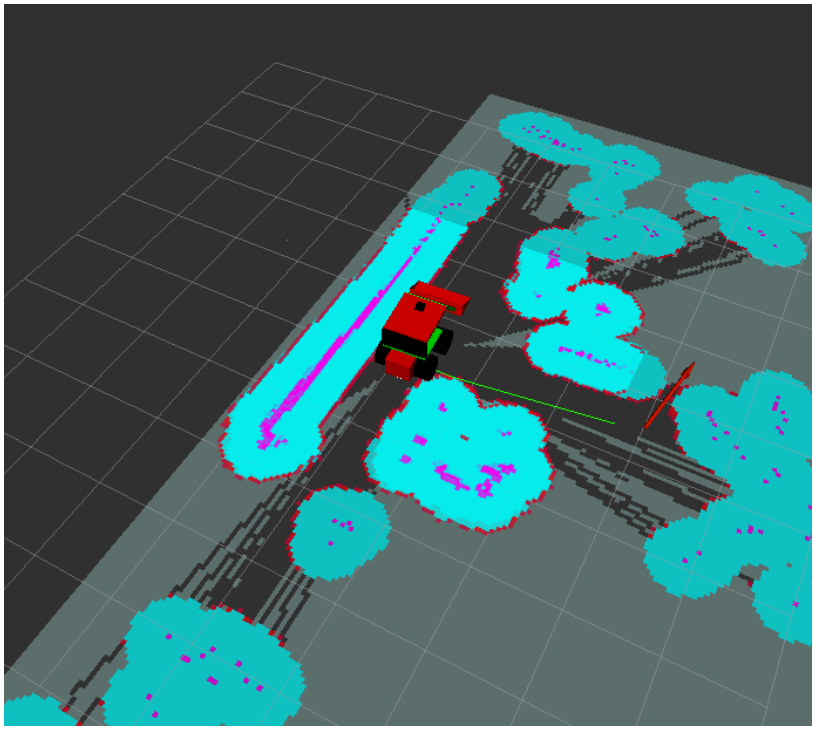


Fig. 13. Move_base path to goal

*4) Competition Goal Achievement:* Qualitative Metric Scoring (see Tables I and II):

- Success Analysis - 0 - Did not function
- Achievement Score - 2 - Planned digging and dumping procedures worked
- Time Constraint Analysis - 9 - Performed very quickly, as not many algorithms were implemented
- Performance Analysis - 2 - In a very specific scenario, this component may have worked

This aspect of the robot's autonomous systems failed to function entirely. The movement goal given to the robot during the competition round at the NASA competition was out of the bounds of the competition arena and there was no algorithm developed to attempt to place new goals if one failed. More discussion on how this aspect can be improved can be found in Section IV-A.3.

*B. Electrical & Communication Performance*

Overall the electrical and communication systems on the robot performed within NASA's competition standards very well and could see improvement regarding general standards for a generic robot. In the next section performance of the different aspects of electrical and communication systems will be discussed.

*1) Power Consumption:* The robot consumed 1A at 24V while not moving, with all motors running the robot would consume upwards of 12A. With a 20AH power system, this would mean that with a high motor duty cycle, the robot would be able to function for approximately 2-3 hours. This is a more than acceptable time frame for operation in the aspect of a 10 minute NASA competition round and also a very good time frame for most robot applications, including the physical testing of the robot prior to the competition. Also, the power systems could easily be augmented to accept more batteries in parallel to increase runtime at a cost of increased mass.

*2) Bandwidth Usage:* The robot, while communicating all data necessary for 3D visual representation on a control computer would transmit $\sim$500KB/s from the robot. In terms of real world data use and data transmission wirelessly, this meets necessary constraints for many long distance real world communication systems. For the NASA competition, this amount of data bandwidth was relatively high, but still acceptable as this would translate to only 3KG of regolith mined.

When the robot was transmitting all possible data (including data from the two infrared sensors) this data bandwidth would increase to $\sim$1.5MB/s. This is still fairly acceptable in real world situations for long distance communications.

*3) Latency:* Given the wireless setup on the robot using a 802.11 Wireless N network, latency between the robot and the control room at NASA was never more than 20ms. Data latency would increase when the on-board computer was processing, however this is a different type of latency and is more common when the robot is processing large amounts of data. When the robot is functioning fully autonomously, this latency becomes less important as no commands from the controller would be needed to carry out a task.

*4) Motor Performance:* All of the motors functioned extremely well and allowed the mechanical systems on the robot to achieve the tasks that were necessary for manual operation of the robot. The motors also consumed relatively low current for the output torque required, which helps with the power consumption and runtime requirements. The motors were adequate enough to enable fast operation of the robot, to allow for 6 trips to and from the collector bin with mining in between, in a 10 minute competition round at NASA.

### *C. Control System Performance*

All of the motor controllers and encoders that provide the position of the robot relative to itself (e.g. linear actuator encoder values) were very accurate in both outputting required velocities and reading the data back from the motor. When teleoperating the robot, the control systems respond very quickly and the motors respond without much latent notice. The control systems also gave real time feedback on the current usage, voltage supply, and temperature data for each motor. Which was useful for diagnostic and testing purposes, along with mechanical failure analysis.

### *D. Mechanical Performance*

Mechanically, the robot performed extremely well in completing its task of mining regolith and depositing as much as possible. In terms of the NASA competition, this mechanical design mined the most amount of regolith in a 10 minute competition round in the 7 year history at NASA and was awarded the Judge's Innovation award from NASA for this achievement [30].

The biggest problem with this robot for NASA was the weight. For every 0.45kg of mass sent to mars, it costs NASA $\sim$\$10,000 [31]. Due to this fact, the NASA competition requires all robots to be under 80kg and points are awarded for lower weights. This robot design was just under 80kg due to the stability and digging power that this provides, but due to the scoring system was not efficient. In real world situations 80kg makes the robot

slightly inconvenient to carry and move around, however for stability and mining ability in Earth gravity environments it is a more than acceptable weight.

The mechanical design was also built to be very rugged and durable and it has held up to that expectation. Throughout all of the tests performed on this robot, none have done any damage to the mechanical build in any significant way and a simple cleaning is sufficient to keep the systems in working order.

## IV. DISCUSSION

In this section the results presented in Section III will be analyzed and the reasons for those results will be discussed and suggestions on how those systems could be improved for the future will be made. All of the suggestions will be summarized in Section IV-D.

### *A. Autonomous Systems Analysis*

The autonomous systems did not function well enough to perform as needed for the NASA competition round. There are several reasons for this, all of which will be discussed and solutions will be proposed.

While the teleoperations systems performed extremely well, with nearly no latency in control, the sensors on-board the robot did not provide accurate data for robust autonomous control. The problem was not with the algorithms used or the methods behind creating the autonomous system. But it simply came down to bad data going into the system, which resulted in performance that didn't achieve the required goal.

The sensors that were heavily relied on for input into the autonomous frameworks are outlined below in order of importance to functioning autonomy:

1) Phidgets IMU [15]
2) RPLidar [16]
3) Intel Realsense R200 Camera [17]

The Phidgets IMU [15] will be discussed first due to its importance. This sensor was specified due to the ease of integration into the programming framework that was already being used for the Phidgets motor controllers. This decision was incorrect and

more research was required in selecting an IMU that would provide robust and accurate positioning data. This IMU was relatively good at detecting the orientation of the robot using the gyroscope on the chip, however the accelerometer was extremely inaccurate in detecting the velocity values of the robot, so much so that the data was unusable. If the localization from the IMU was used, the localization on the RTABMap representation would constantly be ”driving” to non-deterministic places in the X/Y plane. A Kalman Filter (also discussed in Section IV-B.1) was implemented to attempt to filter out the bad values, but the sensor was so inaccurate that fusing the data with other odometry sources just made the other sources worse.

The next most important sensor for the autonomous systems was the RPLidar. This sensor scanned on a 2D plane and detected obstacles on the same Z-positioning (height) as just over the top of the robot. The positioning of this sensor was suboptimal, because it was not able to detect obstacles lower to the ground that the robot would want to avoid. It was also in a position that during the NASA competition round, the only thing that it would be able to detect were the walls of the arena. In the NASA rules for the competition [3], scanning the walls is strictly prohibited therefore optical flow localization would not be possible using the walls. Because of these two issues, although the data received from this sensor was accurate and good, it was not very usable in the competition.

The Intel Realsense R200 camera was the 3D IR sensor that output a pointcloud of detected obstacles or objects as location points in three dimensions. This sensor was supposed to be scanning the ground and detecting the $\approx$0.3m in diameter boulders in the competition’s ”obstacle field”. This sensor did not function well in the competition environment due to the large amount of infra-red interference in the arena due to sunlight, it rendered the data from the sensor meaningless. That sensor data also created fake ”obstacles” where the IR interference would create points in the algorithm in comparing the two IR images. This will be discussed more in the next section IV-A.1.

*1) Anomalous Sensor Data:* As mentioned, the largest problem was bad sensor data. The IMU provided near unusable data, with velocity values often being so variable that

when held completely still, the data would fluctuate to moving as much as 0.1m/s in any direction.

The Realsense camera, even in an indoor environment with very little IR interference would have small 1-2 point anomalies, in which it would detect an obstacle very near the robot when there was nothing in the field of view. These anomalies were small enough that with good filtering on the LiDAR data, they could have been removed - however they were prevalent enough to require active filtering on the sensor data, which made it such that the Realsense camera data could not be trusted unless it had data from a separate sensor to back it up. Because the LiDAR array was unusable in the competition arena, no secondary filtering system was available and the IR interference from the sunlight diminished the accuracy of the sensor even further. Because of these object misrepresentations and inaccurate obstacle points, the Realsense Camera data was ultimately so anomalous that it was unusable.

The RPLidar sensor was accurate enough and very reliable; and almost never gave a distance value that wasn't within tolerances defined in the sensor specifications [16]. Because of this, the sensor has a great deal of use in applications other than the NASA competition.

*2) Sensor Type Disadvantages:* 2D LiDAR arrays are very useful in environments that are inherently more two dimensional, such as tall hallways, or rows of tall objects. However in environments that have shorter, or more variable, Z-positioned obstacles 2D LiDAR scanning can miss many obstacles that aren't in the exact Z plane that the laser is scanning in. There are better LiDAR sensors that enable a +/- 15 degree field of view and these LiDAR sensors would remove this disadvantage.

IMU's are extremely useful sensors and, when outputting good data, they can make map localization so accurate that nearly no loop closure between the local and global object grids is necessary (discussed in Section II-E.1). The sensor selected for this robot was very inaccurate and better IMU's are readily available; to summarize, there is nothing wrong with the technology of IMU's but this sensor was a poor fit with this robot

implementation.

3D IR camera sensors have a very large problem, which is the inability to work well outdoors. New sensors are adding apertures to reduce the IR brightness, and make working in outdoor possible, but currently those 3D IR sensors are not on the market. Two other options are available such as RADAR sensing (which is often too grainy to process) or regular stereoscopic camera image processing. Stereoscopic processing is a fairly well established technology and uses the same kind of processing that IR imagery does. A sensor created specifically for outdoor robot vision called the ZED Camera [32] is a good choice and has the potential to remove the anomalous data values from the equation in the visual sensing systems on the robot.

*3) Goal Planning & Execution:* The largest problem with Goal Planning, was the inability to detect where the robot was in respect to the collector bin. Without the knowledge of where the collector bin was in respect to the robot, proper goals were impossible to set and guesswork was required.

Systems were developed to attempt to solve this problem; a QR code was set up on the collector bin that the robot was required to detect using the 3D IR Imagery. A ROS node called find_object_2d was used to attempt this [33]. Once the QR code was detected it would provide a transform from the collector bin to the robot, allowing localization to occur and goals to be set. The problem with this QR code and object detection system was that the robot needed to be too close to the QR code in order to get a localization position (on the order of 3-5 ft). Because of this, if the robot lost localization or had an error of some kind and wasn't within 3-5 feet of the collector bin, it would essentially be lost and not be able to find its way back.

Because the object detection system was inadequate, new solutions were devised to fix this problem. LiDAR beacons were designed to be placed on the collector bin itself. Using reflectivity and distance differentials the LiDAR beacons would be scanning for the robot constantly and would relay where the two beacons thought the robot was in respect to themselves (e.g. the collector bin). This solution has the potential to be robust, as

triangulation and redundancy factor in with having two scanning beacons. Additionally, it has the potential to be extremely accurate over the 10m size of the competition arena, as most LiDAR systems have an accuracy of less than +/- 1cm [34].

With a functioning beacon system in place, the robot would theoretically have a reference to the collector bin from the base link frame on the chassis at all times; this would allow for near perfect goal setting and digging/dumping procedures. This is the recommendation that will be made for future work VI.

### *B. Robot Sensor Improvements*

In this section, possibilities of improving the poorly performing sensing systems discussed in Section II will be explored; more specifically, how to improve the autonomous systems on the robot by changing the filtering or tuning autonomous systems algorithms to work with poor data.

*1) Kalman Filtering & Sensor Fusion:* Due to the inaccuracies of the IMU and the need of detecting slippage and orientation using multiple sensors, a system was needed to be able to take in multiple odometry sensor data and combine them to produce a better odometry outcome than the results from a single sensor. Researchers using ROS have already developed nodes to accept multiple odometry sources and combine them using Kalman filtering algorithms. This node called ”robot_localization” [35] was implemented to do this sensor fusion.

With wheel odometry, IMU orientation data, and LiDAR optical flow inputs, being fused with Robot_Localization it created the improvements to the SLAM system seen in the second half of Section III-A.2. With these changes, the robot was able to navigate a fairly complex environment when there were enough features to detect for the LiDAR Optical flow.

*2) Voxel Grid Implementation:* Due to the anomalous sensor data discussed in Section IV-A.1, obstacle detection required filtering to avoid using the inaccurate data to create obstacles in the path-finding algorithm. Because of this, a voxel grid costmap system was implemented in RTABMap [20]. This grid conglomerates sensor data and

evaluates the points in specified clusters. If there are enough points in a cluster, it is marked as an obstacle in the costmap. Unfortunately, the Intel Realsense camera data was so anomalous that if the voxel grid was desensitized enough to filter out the extraneous points, it would also filter out real obstacles. Because of this, the filtering was removed from the implementation and the ultimate solution is to get a better sensor.

### *C. Human Interaction*

The initial requirement for developing fully autonomous robotic solutions is human interaction and teleoperation. If the robot cannot be controlled by a human operator, it will be impossible to make the robot function on its own. Because of this, the first step in all robotics projects is to have a controllable robot working with teleoperated controls, then attempt autonomous operation.

Additionally, with any robot moving without human input, emergency stop systems must always be in place. Autonomous systems are not robust enough yet to be fully relied upon. For safety concerns, 80kg is heavy and can cause significant damage if not safely operated (autonomously or by human controls). Because of this, safety systems are very important. More discussion on safety can be seen in Section IV-C.2.

*1) Complete Teleoperation:* The teleoperations system took commands from an XBox 360 controller and relayed them to the control systems on the robot via the wireless connection. As mentioned in Section III-B.3, the communication system performed without much latency. The control systems were accurate and responsive, so with the fast communication systems the teleoperation system as a whole functioned nominally. Because the robot worked so well under teleoperated control, it is a good sign that with better sensors an autonomous system could be implemented on the current mechanical, control system, and electrical frameworks.

*2) Emergency Safety Controls:* Several emergency stop systems were implemented on the robot; the first of which was a large, red, illuminated emergency stop button on the back of the robot. Whenever this was pressed, the power to all electrical systems from the battery was completely cut off. In terms of teleoperation, two safety systems were

implemented; First, was a button on the controller that would cease all movement as soon as the command was sent, second was a redundancy in the control systems that would check if a message was received from the controller within the last half-second. If no message was received from the joystick to the robot, the robot would automatically stop all motors. This redundancy prevents the robot from continuing to move after losing connection to the controller. Finally, when the robot was running autonomously, the operator of the robot was required to have the autonomy stop button within reach at all times. This would switch the robot over from autonomous mode, back to teleoperation mode, ceasing all movement.

### *D. Improved Robot System Requirements*

In this section all of the problems discussed above will be collated, and new systems and solutions that could potentially resolve the problems will be recommended. A robot with these recommendations implemented would be the next step in this research and is the starting point for future work VI.

*1) Improved Autonomous Sensors:* Drawing on the discussions in Section IV-A, new sensors were selected. The first decision was to remove the LiDAR scanning from the robot entirely. This sentence will be qualified with the fact that LiDAR is an extremely useful system, but much less useful in the NASA competition. If this was a robot purely for research or other constrained purposes, new LiDAR systems with improved accuracy would be specified; but because of the NASA requirement to not scan the walls of the arena, the 2D scanning LiDAR system was ineffective.

*2) Improved Control Systems:* The Phidgets motor controllers, for the most part, worked very well and enabled proper functioning of all of the motors on the robot. The major drawback for the Phidgets system is that all of the communication between the on-board computer and the Phidgets had to be done through an individual USB cable for each motor. Because of this, a large 10 port USB hub was required and the wiring of those systems was very difficult to manage and keep organized. Additionally, the ROS framework developed for Phidgets [24] included a non-deterministic bug which would

occasionally cause the motor controllers to disconnect and would crash the Drive ROS node. This was solved by restarting the node automatically as soon as it crashed, but was a large enough problem to seek an additional solution.

Because of the two problems listed above, different motor controller systems were sought out. The first selected were the Roboclaw 2x30A [12] motor controllers. These controllers contain on-chip PID processing for the encoder output of the motors which take in velocity values, using the encoder output, move the motors to make the robot move at exactly that speed taking into account the diameter of the wheels. Additionally, these motor controllers allow for two different communications inputs, first is over USB, like the Phidgets, but it also accepts serial input.

The Roboclaw controllers are a definite improvement for the drive system, however solutions for the linear actuators needed to be devised. For the linear actuators Pololu 24v23 motor controllers [36] were specified and selected. These motor controllers do not contain on-chip PID, but this is unnecessary because the linear actuator velocity can be set effectively using a duty-cycle percentage. The Pololu controllers also have two input systems, with the ability to daisy chain connections over serial. Because of this, they were selected as it would reduce wiring and system USB requirements.

The only difficulty to switching to these control systems is the programming overhead to remake the Drive node discussed in Section II-D; but the benefits for the robot overall outweigh this overhead.

*3) Improved Electrical Systems:* Due to the weight constraints in the NASA competition, the battery system needed reevaluation. Using NiMH batteries increases safety and reliability, however it has a relatively low power density compared to technologies such as LiPo or Li-on batteries. LiPo batteries were evaluated [37] that output the same $\sim$24V and will provide slightly less total Amp Hours, but will have a mass that is half that of the NiMH solution.

The PDPCB outlined in Section II-C.2, functioned very well and should be used for future work.


Fig. 14. TPU 3D Printed Treads

*4) Improved Mechanical Design:* As mentioned in Section III-D, the most concerning problem with the mechanical design was the overall mass and this problem is only relevant in terms of the NASA competition. The design's mass was 80kg, after analysis it was determined that the best mass for the robot to be successful in the competition was 40kg; this means that the robot mass needs to be halved in order to be competitive.

In order to accomplish this task, each new component needed to be weighed and solutions to decrease the weight of heavy components needed to be devised. The heaviest component on the robot, by far, were the two large rubber treads. The treads were $\sim$0.5in thick, which includes a large amount of extra material not necessary for effective movement through the regolith and the life of the treads required. In order to solve this problem, 3D printed treads using TPU material were developed that would decrease the tread weight from 14kg to under 2kg. The TPU tread design can be seen in Figure 14

The Sliding Linearly Excavating Digger (SLED) system that was developed (using two large linear rods with sliding bearings) was overbuilt and weighed 12.3kg. Instead of using two steel rods a solution using composite materials integrated into the structure

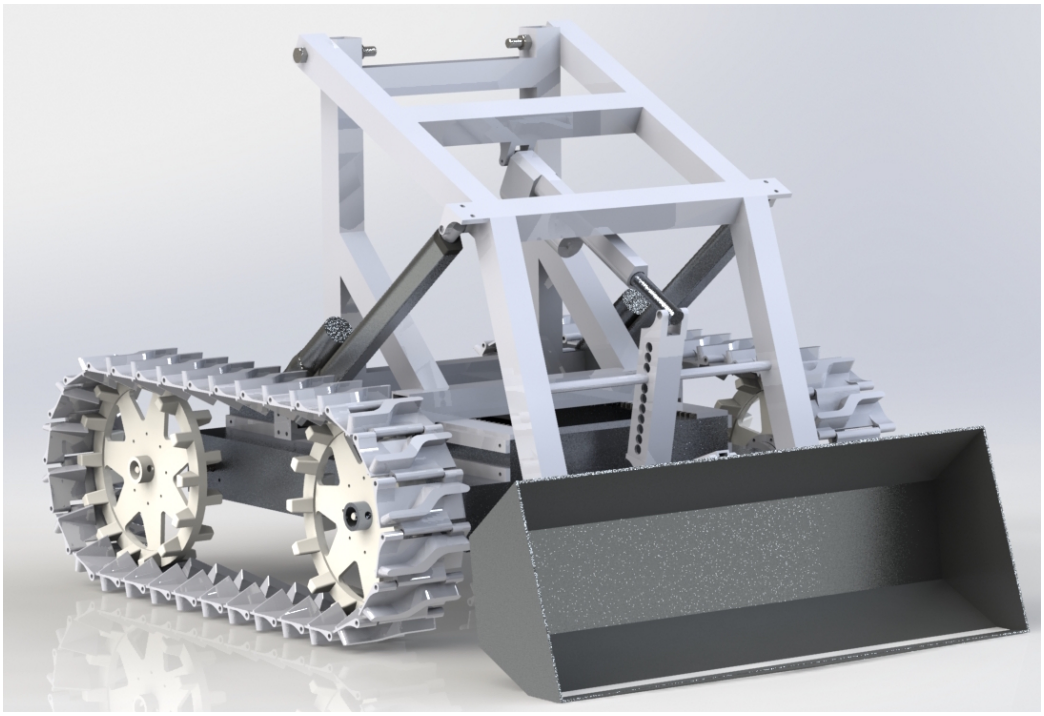


Fig. 15. New design with aluminum chassis, integrated sled, lighter treads, and new bucket.

of the chassis itself was devised. Finally, the steel frame of the chassis was an additional source of unnecessary mass. Instead of steel, an aluminum solution was devised for future work. The design for an all aluminum chassis, integrated SLED, lighter treads, and new bucket design can be seen in Figure 15 below.

## V. CONCLUSION

Although this robot system wasn't able to accomplish the goal of fully autonomous operations. The algorithms, methods, programming, and electrical implementations used were shown to be very robust and with good sensor data, it was shown that the robot would be able to operate with no human control. Because of this realization, future work can be done with better sensing systems and fully autonomous mining operations should be achieved.

The autonomy platform ROS [2] was a crucial part of this research. The abstraction and simplification of 3D visualization, path-finding, and motion-planning that are all taken care of by using ROS enabled the rapid redevelopment of systems when integration failed.

It also created a modular platform for implementing the same autonomous functionality on different robotic platforms. Because of this, ROS is the recommended framework in any future work.

Because of the success of the implementation of the autonomous system as a whole, it is shown fully autonomous robotic systems are well within the realm of possibility for many applications. Even relatively small research groups are able to develop systems in reasonable amounts of time. With the latest sensor technology decreasing in cost and increasing in accuracy and reliability, it is only a matter of time when fully autonomous robots, including self-driving vehicles become common place in our society.

## VI. FUTURE WORK

Future work in this research will be completed by implementing the changes discussed in Section IV-D. This will consist of improving the sensors used and verifying the new autonomous functions with the improved sensors. Other changes may include new and improved methods of localization - such as using LiDAR beacons to track the robot in a constrained environment. When the autonomous systems are improved, new metrics to measure autonomous performance will be implemented and the robot will be made to attempt to complete more complex tasks. In terms of the NASA competition, the overall goal in future work is to complete a robust autonomous system that will be able to mine without any user input for the entire NASA competition round.

## VII. ETHICS & SUSTAINABILITY

Autonomous robots have the potential to make many dangerous jobs safer by removing the human component. In 2015, Rio Tinto Kennecott sponsored this research with the specific goal to help develop autonomous mining operations. Autonomous mining would make enclosed coal mines, open pit mines, and various other mining operations increasingly safe for humans. It also would be more profitable for companies by requiring fewer employees to operate in a potentially unstable and unsafe environment. With robots,

operators would preform all mining tasks from the safety of an off-site system control room.

Robots can also improve safety in other fields whenever there is a hazardous environment such as space exploration, which requires some kind of physical manipulation, autonomous robots can accomplish that task without human interaction. Autonomous robotics enables humans to accomplish tasks beyond their ability by extending the limits of the human condition.

Finally, in regards to ethics and robots that are functioning autonomously; Isaac Asimov's three laws of robotics [38] very succinctly sum up what the limits of a robot's ability should be. A challenge to anyone to improve on these laws is encouraged.

# REFERENCES


[1] J. Robe. (2017). *The Utah Robotic Mining Project*. Available: http://roboticmining.eng.utah.edu/

[2] Open Source Robotics Foundation. *Robot Operating System*. Available: http://www.ros.org/

[3] R. Cannon. (2017, March 16). *NASA Robotic Mining Competition*. Available: https://www.nasa.gov/offices/education/centers/kennedy/technology/nasarmc.html

[4] L. A. Rahmatian and P. T. Metzger. (2010). *Soil Test Apparatus for Lunar Surfaces*. Available: https://www.nasa.gov/sites/default/files/bp-1_soil_testing.pdf

[5] I. Strainsert. (2017). *Force Sensing Bolts/Studs*. Available: http://www.strainsert.com/product-categories/force-sensing-bolts-studs/

[6] Phidgets (April 2, April 2). *Micro Load Cell (0-5kg) - CZL635*. Available: http://www.phidgets.com/products.php?category=34&product_id=3133_0

[7] J. P. L. NASA, (2017, April 2). *"Curiosity Self-Portrait at Martian Sand Dune"*. Available: https://www.nasa.gov/image-feature/jpl/pia20316/curiosity-self-portrait-at-martian-sand-dune

[8] Phidgets. (2014, April 2). *24V/82.6Kg-cm/33RPM 76:1 DC Gear Motor*. Available: http://www.phidgets.com/products.php?category=24&product_id=3273_1

[9] NVIDIA. (2017, April 2). *NVIDIA Jetson TX1*. Available: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html

[10] Intel. (2016, April 2). *Mini PC: Intel NUC*. Available: http://www.intel.com/content/www/us/en/nuc/overview.html

[11] Phidgets. (2016, April 2). *PhidgetMotorControl 1-Motor*. Available: http://www.phidgets.com/products.php?category=12&product_id=1065_0

[12] Roboclaw. (2017, April). *Roboclaw 2x30A*. Available: http://www.ionmc.com/RoboClaw-2x30A-Motor-Controller_p_9.html

[13] Phidgets. (2015, April 3). *Optical Rotary Encoder HKT22*. Available: http://www.phidgets.com/products.php?category=7&product_id=3531_0

[14] Phidgets. (2014, April 2). *DC Linear Actuator P5H-24-300mm*. Available: http://www.phidgets.com/products.php?category=39&product_id=3547_0

[15] Phidgets. (April 2). *PhidgetSpatial Precision 3/3/3 High Resolution*. Available: http://www.phidgets.com/products.php?product_id=1044

[16] Slamtec. (2016, May 23). *RPLidar V1*. Available: https://www.slamtec.com/en/Lidar

[17] Intel. (2016, April 3). *Intel Realsense R200*. Available: https://software.intel.com/en-us/articles/realsense-r200-camera

[18] Mini Box. (2016, April 2). *DCDC-NUC 6-48V automotive converter*. Available: http://www.mini-box.com/DCDC-NUC

[19] Tenergy. (2016, April 2). *"Watt's Up" RC Watt Meter*. Available: http://www.tenergy.com/01003

[20] M. a. M. Labbe, F., *"Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM"*; Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. Sept, p. 2661-2666, 2014. Available: http://wiki.ros.org/rtabmap

[21] Microsoft. (2017, April 2). *Microsoft Robotics Developer Studio 4.0*. Available: https://msdn.microsoft.com/en-us/library/bb483024.aspx

[22] Microsoft. (2016, April 2). *Microsoft Kinect*. Available: https://developer.microsoft.com/en-us/windows/kinect

[23] Orocos. (2017, April 2). *The Orocos Project*. Available: http://www.orocos.org/

[24] M. Guenther and B. Mottram. (2016, April 2). *Phidgets ROS Node*. Available: http://wiki.ros.org/phidgets

[25] D. Lu. (2017, April). *move_base*. Available: http://wiki.ros.org/move_base

[26] M. Guenther. (2016, April 2). *phidgets_imu*. Available: http://wiki.ros.org/phidgets_imu

[27] Slamtec. (2016, April 2). *rplidar_ros*. Available: http://wiki.ros.org/rplidar

[28] Microsoft. (2016, April 2). *Microsoft XBox 360 Controller For Windows*. Available: https://www.microsoft.com/accessories/en-us/products/gaming/xbox-360-controller-for-windows/52a-00004

[29] I. Dryanovski. (2017, April 2). *laser_scan_matcher*. Available: http://wiki.ros.org/laser_scan_matcher

[30] NASA. (2016, April 2). *2016 NASA Competition Winners*. Available: https://www.nasa.gov/feature/2016-robotic-mining-competition-winners

[31] NASA. (2010, April 2). *Advanced Space Transportation Program*. Available: https://www.nasa.gov/centers/marshall/news/background/facts/astp.html

[32] Stereolabs. (2017, April 2). *ZED - Depth Sensing Camera*. Available: https://www.stereolabs.com/zed/specs/

[33] M. Labbe. (2016, April 2). *Find_Object_2D*. Available: http://wiki.ros.org/find_object_2d

[34] Garmin. (2017, April 2). *LIDAR-Lite v3*. Available: https://buy.garmin.com/en-US/US/p/557294

[35] T. Moore. (2016, April 2). *robot_localization*. Available: http://wiki.ros.org/robot_localization

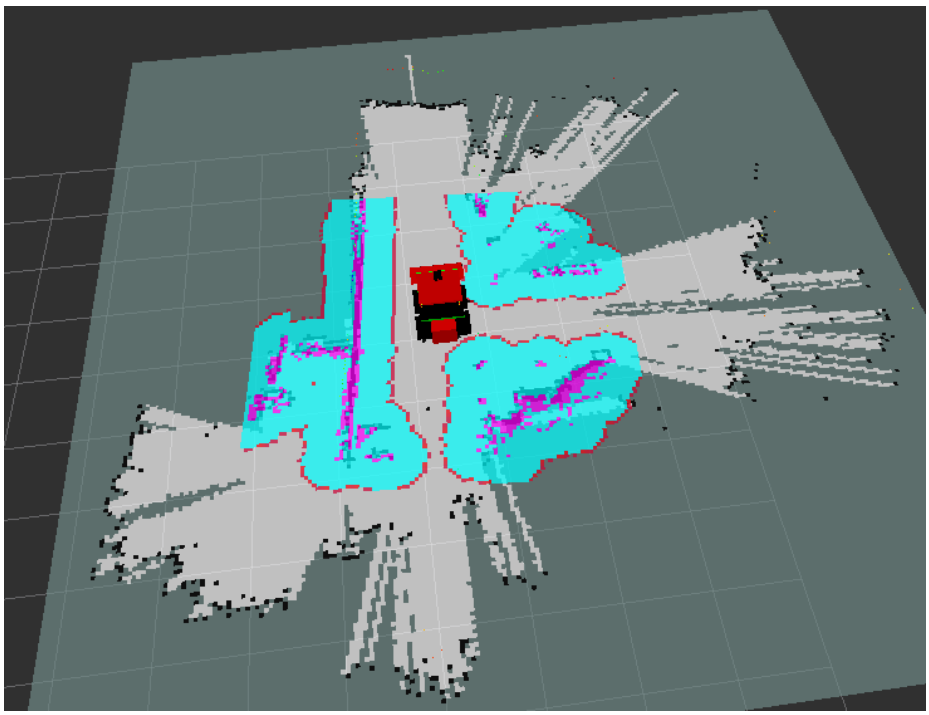
[36] Pololu. (2016, April 2). *Pololu Simple High-Power Motor Controller 24v23*. Available: https://www.pololu.com/product/1383

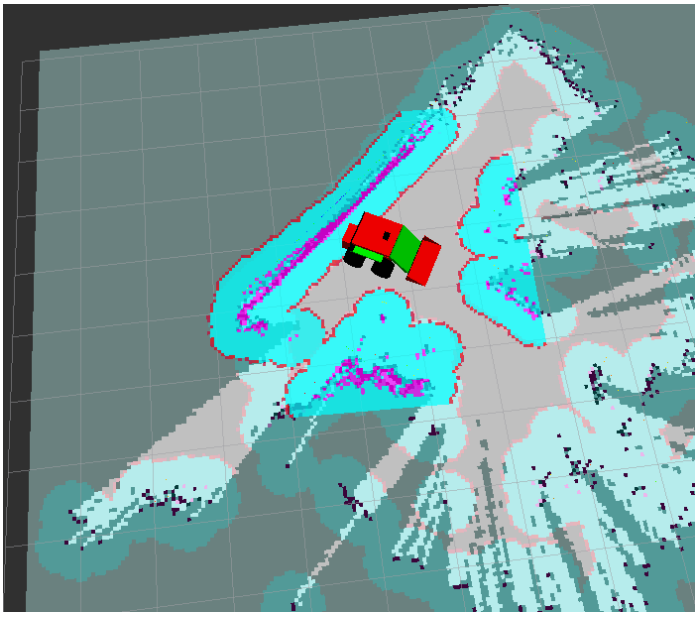
[37] Hobby King. (2017, April 2). *ZIPPY Compact 5800mAh 7S 25C Lipo Pack*. Available: https://hobbyking.com/en_us/zippy-compact-5800mah-7s-25c-lipo-pack.html

[38] I. Asimov, *I, Robot*. New York: Ballantine, 1977.

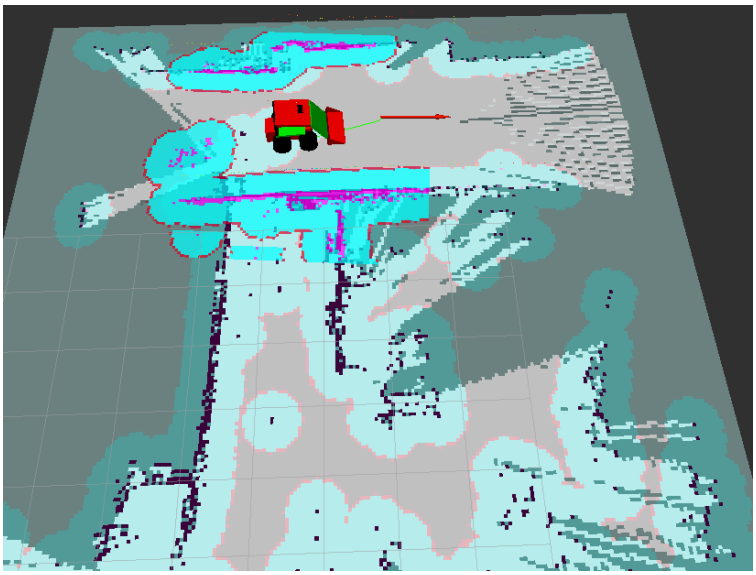
# APPENDIX

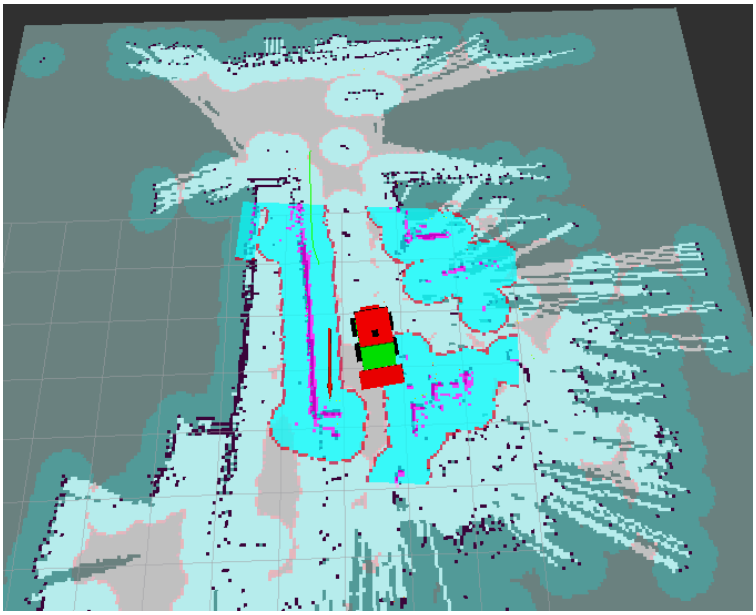
## A. *Well-functioning RTABMap Visualizations*





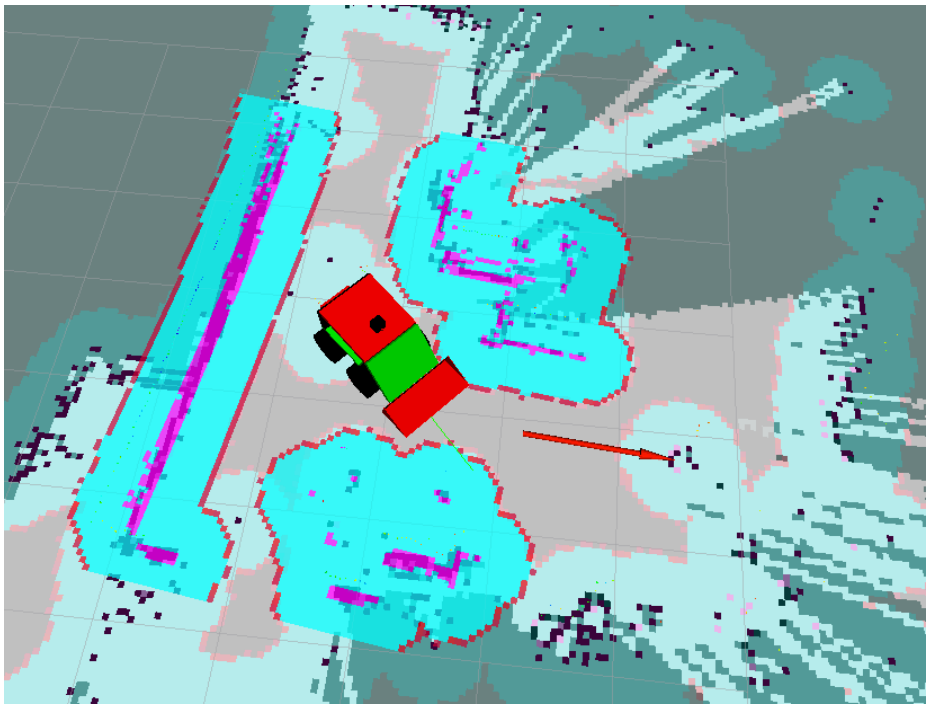
*B. ROS LiDAR Odometry Failure*

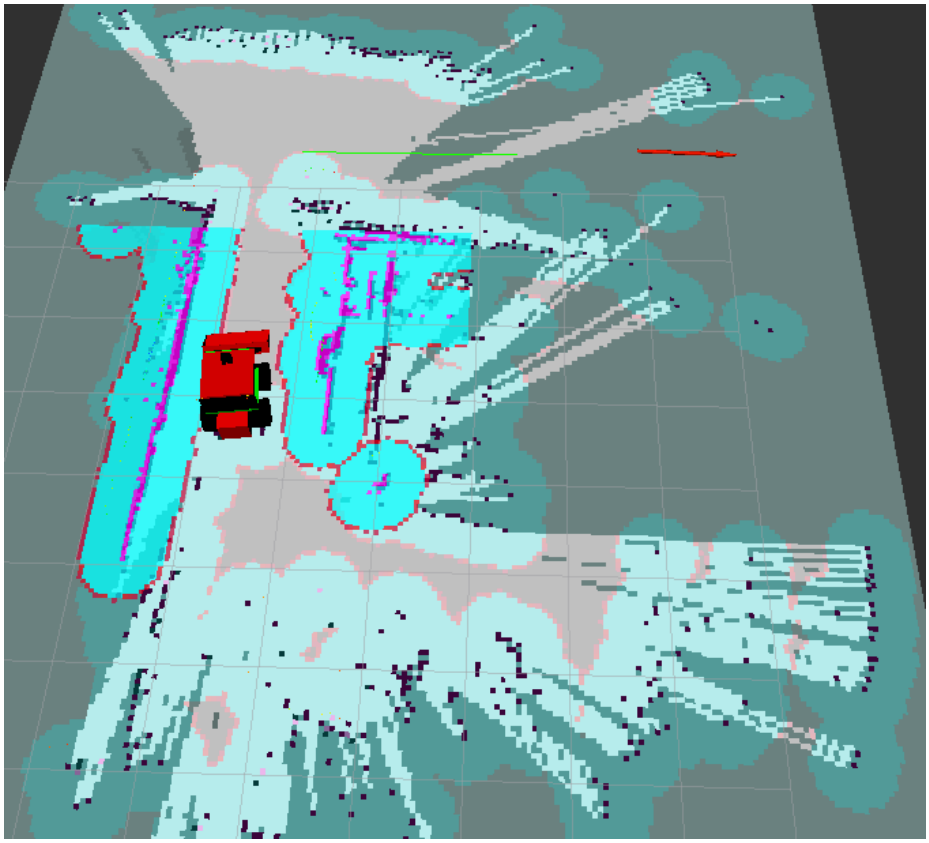




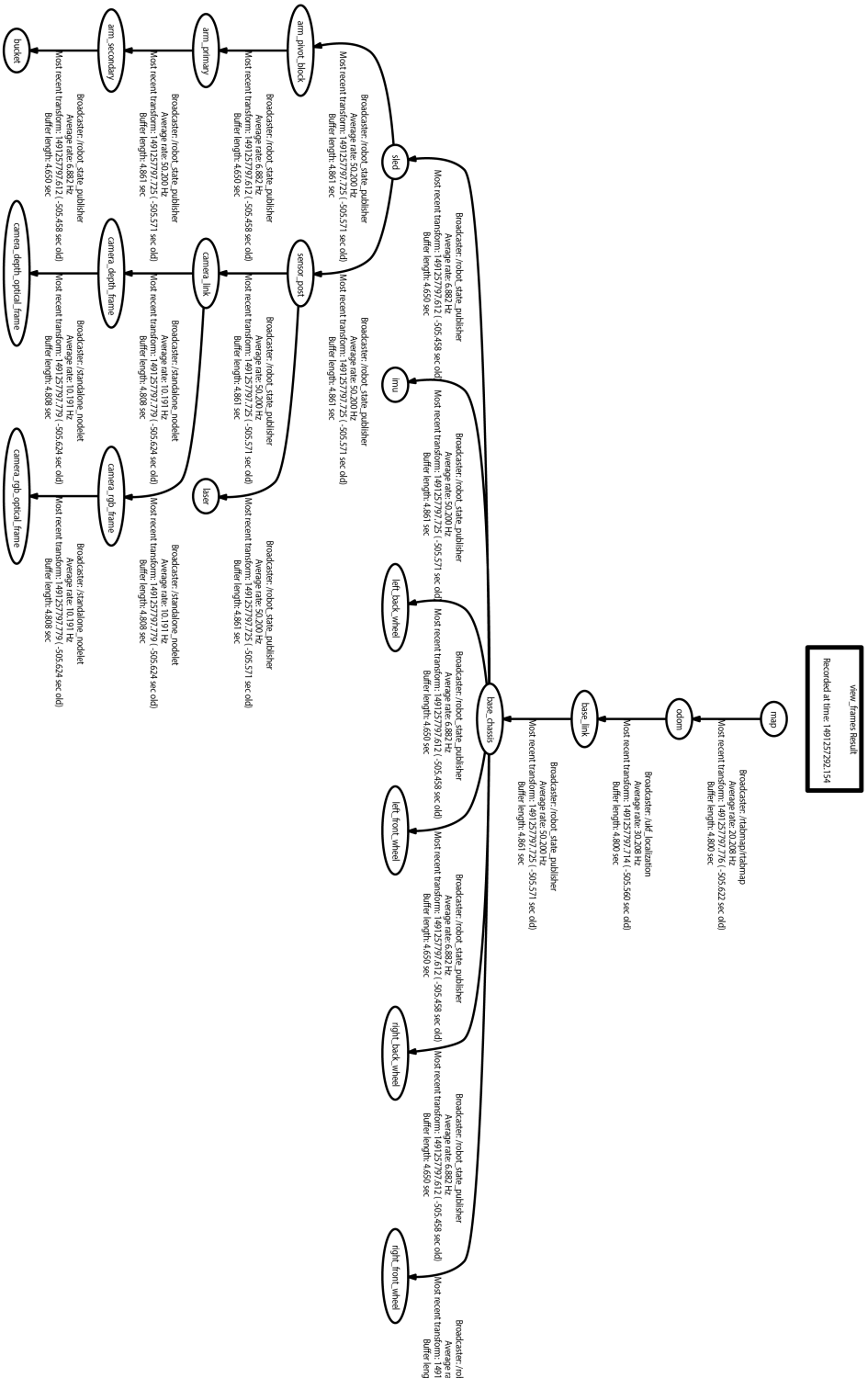
In the two figures seen above, the robot was sent into a straight hallway with no identifiable features. In the first image, it was told to achieve a move goal 5 feet down the hallway, due to the lack of features it traveled further than 5 feet and would have continued indefinitely. In the second figure, the robot was manually driven back into the lab, this was done to show the error in loop closure before and after the test.

*C. ROS movement goal achievement*





## *D. ROS Transform Frame Visualization*



| | |
|---|---|
| Name of Candidate: | John William Robe |
| Birth date: | June 9, 1995 |
| Birth place: | Burnsville, Minnesota |
| Address: | 229 W Reed Ave<br>Salt Lake City, Utah<br>84103 |