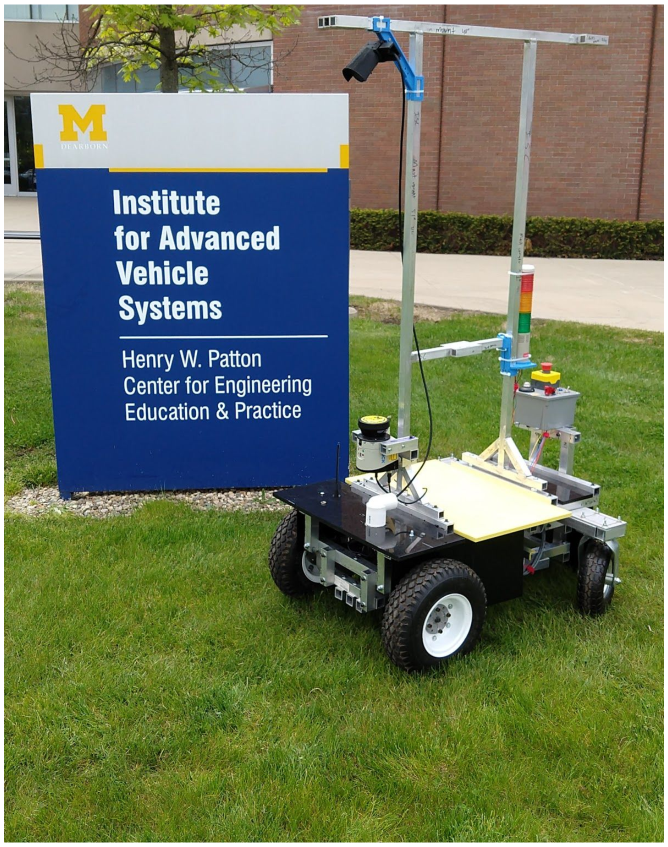
University of Michigan Dearborn
AluminOHM


M
DEARBORN
Institute for Advanced Vehicle Systems
Henry W. Patton Center for Engineering Education & Practice

Team Members:
Siddharth Mahimkar, Matthew Abraham, Kenneth Topolovec,
Jared Hagerty, Timothy Tapper, Nutan Gangapure, Caroline Ham,
Dhimant Khuttan, Joseph Kennedy, Antonio Alioto, Kyle Sanez

Faculty Advisors: Michael Putty Ph.D

DATE: 5/15/2018
Team captain: Siddharth Mahimkar

I, Michael Putty Ph.D of the Department of Electrical and Computer Engineering at the University of Michigan Dearborn, certify that the design and development of AluminOHM is significant and unique, and is equivalent to what might be awarded credit in a senior design course.

X Michael W. Putty

ABSTRACT

This paper presents AluminOHM, a robot designed and used by the University of Michigan - Dearborn for the 27th Annual Intelligent Ground Vehicle Competition (IGVC). AluminOHM, is a newly designed robot based off the original Ohm robot used in previous competitions. Changes to the electrical system has allowed the team to experiment with new technologies to be used on other robots within the club. The software platform is a continuation from the previous year's competition to allow for a solid foundation for future iterations.

INTRODUCTION

The Intelligent Systems Club of the University of Michigan - Dearborn enters the 2019 Intelligent Ground Vehicle Competition with 8 new and 3 returning members. The main goal of this year's team is to learn and establish a solid foundation and functionality an understanding in Robot Operating System (ROS) to improve the overall robustness of the robot and to mature the team's knowledge of various robotics concepts. This year's strategy is to utilize key successes from the existing platform and learning how to improve upon weaknesses in the current and previous designs.

The team consists of 10 undergraduate and 1 graduate students, many of whom plan to participate in future competitions. The team member composition is displayed in **Table 1**.

Table 1: Team AluminOHM Composition

Name	Email	Class	Role
Siddharth Mahimkar	smahmka@umich.edu	Computer Engineering, Senior	Captain
Matthew Abraham	mjabraha@umich.edu	Computer Science, Senior	Software Lead
Kenneth Topolovec	ktopolov@umich.edu	Electrical Engineering, Senior	Electrical Lead
Jared Hagerty	jwhagert@umich.edu	Mechanical Engineering, Senior	Mechanical Lead
Timothy Tapper	ttapper@umich.edu	Computer Engineering, Junior	Software
Nutan Gangapure	nutan@umich.edu	Electrical Engineering, Graduate	Software
Caroline Ham	cyham@umich.edu	Computer Science, Sophomore	Software
Dhimant Khuttan	dkhuttan@umich.edu	Electrical Engineering, Freshman	Electrical
Joseph Kennedy	josephpk@umich.edu	Robotics Engineering, Freshman	Electrical
Antonio Alioto	afalioto@umich.edu	Electrical Engineering, Junior	Electrical
Kyle Sanez	ksanez@umich.edu	Robotics Engineering, Senior	Mechanical

This paper will begin with a description of design innovations, then cover mechanical electrical and software systems in finer detail. After those sections there will be a detailed description of the software strategy, an overview of failure modes, and the performance analysis to conclude the report.

DESIGN PROCESS

This year the team utilized an iterative design approach prioritizing core functionality across all subsystems first, through a 3-step design, test, improvement process. Each iteration adds new features, and moves from functioning design to functioning design. For example, in the first iteration, the features being implemented only encompassed basic lane detection and obstacle avoidance, while leaving more advanced features like mapping or high-level path planning, for later iterations.

DESIGN INNOVATIONS

This year's team intended to improve on the previous year's accomplishments by redesigning the vehicle's main software platform, and replacing/adding sensors in areas of need. **Table 2** describes the areas which needed improvement and why, as well as what was completed to improve the vehicle. **Tables 3a-c** describes the cost of the robot, with "+" indicating actual cost to the team this year. The remainder of this report will discuss these improvements and how they were implemented.

Table 2: Design Innovations and Reasoning

Areas to be Improved or Added	Reason for Improvement or Addition	Improvement Design
Redesigned frame	Previous frame was made of wood and was deteriorating.	Frame is made with aluminum
Exposing obstacle detection data	Makes it easier to add functionality in future iterations	Raw obstacle data is now published in ROS instead of just what the control algorithm needs
Steering Behavior	Previous control software was given limited options for turns to make, making it difficult to find a more optimal path	Decision for best path to take is placed in control software, not sensor interpreting software
Weight reduction	Robot is difficult to move manually, reduce stress on aging frame	Replace 2x Lead acid with single 6s LiPo

Table 3a: Mechanical Cost

Mechanical	Qty	Unit cost	Price
Frame (+)	1	$300	$300
Motors	2	$450	$900
Plastic (+)	5	$30	$150
Total Mechanical Cost			**$1,350**

Table 3b: Electrical Cost

Electrical	Qty	Unit cost	Price
LiDAR	1	$5,000	$5,000
GPS	1	$5,000	$5,000
Computer	1	$650	$650
Camera (+)	1	$650	$650
6s LiPo battery	6	$80	$480
Battery Management	1	$50	$50
Motor controller	1	$390	$390
Misc (+)	1	$100	$100
Total Electrical Cost			**$12,320**

Table 3c: Vehicle Cost

Overall Category	Price
Electrical	$12,320
Mechanical	$1,350
Estimated Retail Price	**$13,670**
Actual Cost	**$1,200**

MECHANICAL DESIGN

The vehicle used for this year's competition was built on the lessons learned from past robots. The mechanical design is original to this robot and has not been used in prior competitions. The vehicle is made primarily of aluminum and uses a differential drive steering control scheme which is aided by two trailing casters. The CAD model of the robot is shown in **Figure 1**. **Table 4** and **Table 5** provide the dimensions and weight distribution of the robot respectively.

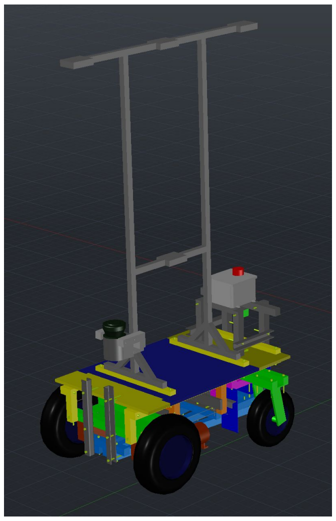


Figure 1: Robot design

Structure design

The robot is made almost entirely of aluminum with plastic enclosing the frame and electronics for environmental protection. The robot has been manufactured from 1 ¼” x 1 ¼” aluminum square tubing, bolted together. The center of the frame houses the main electronics shown in **Figure 2**

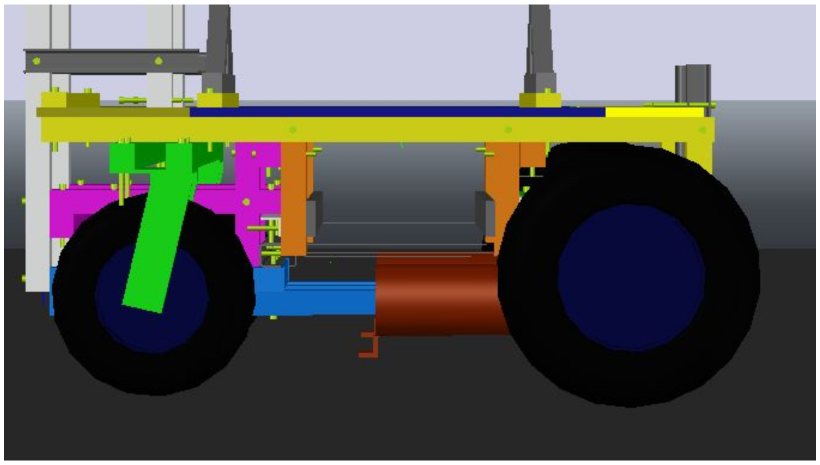


Figure 2: Electronics location

The robot utilizes two rear casters and is propelled by twin 24 volt NPC B81 brushed DC motors with integrated 18:1 gearboxes, providing a maximum of .81 horsepower each and a maximum of 180 rpm. The tires are 0.33m and work well on a grassy field. There is an aluminum mast serves as a mount point for the GPS, IMU, camera, and LiDAR as shown in **Figure 3**. The battery is housed in the front of the robot, along with the computer for accessibility

Table 4: Vehicle dimensions

	Vehicle	Requirements
Width	0.78m	0.61m - 1.21m
Length	0.94m	1.21m - 2.13m
Height	1.79m	1.82m maximum
Mast height	1.37m	-
Mast length	1.02m	-

Table 5: Vehicle weight distribution

Major Component	Qty	Weight	Total
NPC motors	2	9 Kg	18 Kg
Drive wheels	2	4.5 Kg	9 Kg
Caster	2	2 Kg	4 Kg
Mast	1	4 Kg	4 Kg
Frame	1	10 Kg	10 Kg
Total weight			45 Kg

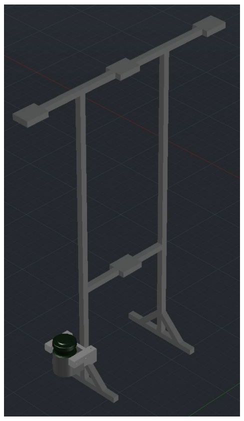


Figure 3: Sensor mast

ELECTRICAL COMPONENTS AND DESIGN

Overview

We implemented a system to monitor the battery levels in the robot and to provide an interface with the user to display information such as its drive mode (manual/autonomous), its drive status (paused/e-stopped/drivable), and battery low voltage warnings. In addition, the majority of the electrical components are now encased in an electrical box to isolate them from the environment and to make the system much more compact.

Power Distribution System

There are three main supply voltages that power the components of the robot. 22.2v, 19v and 12v DC. It is important to note that, although the typical operating voltage of the LiPo battery used is 22.2v, at a full charge, each of the six cells charges to 4.2v, resulting in a 25.2v supply. **Figure 4** shows the component voltage breakdown.

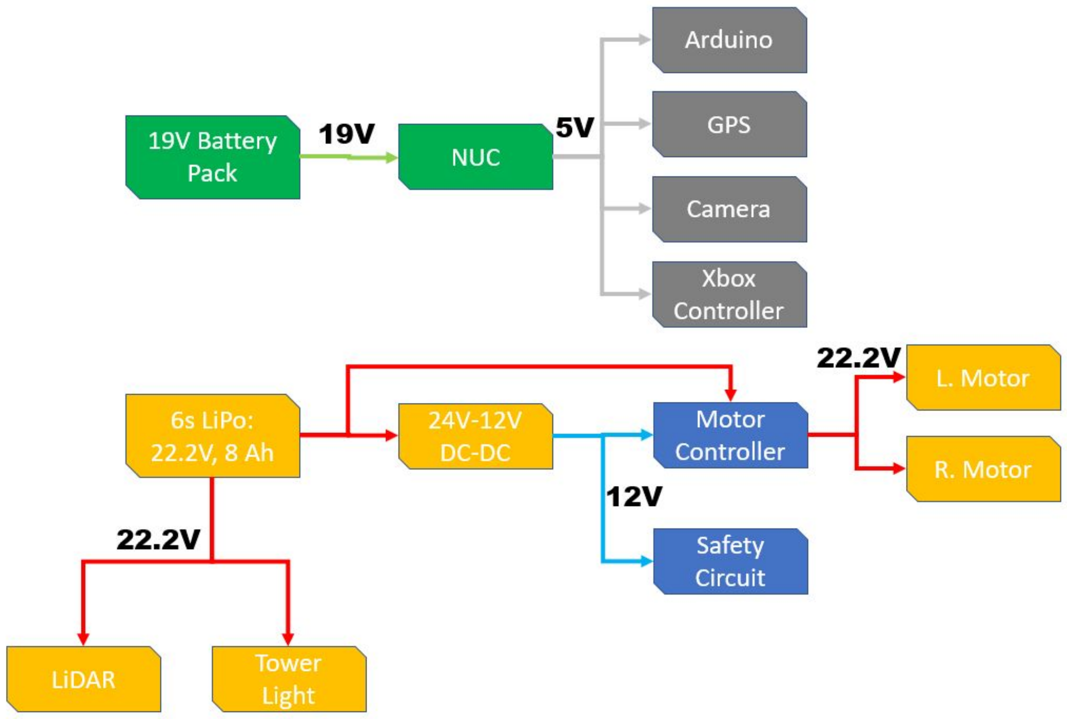


Figure 4: Component Voltage Breakdown

Batteries

A single LiPo battery (specification provided in **Table 6**) will power the robot. There are also 3 additional identical batteries on hand, to swap out with, in the event that one battery is running low. The battery is located inside of the components box which allows for easy access to swap batteries. Furthermore, we also have a 12AH battery on hand this gives us approximately 50% more run time should we decide that we want a longer runtime.

Table 6: Battery specifications

Capacity	Nominal voltage	Overcharge	Charge time
8AH/12AH	22.2v	25.2v	~3 hrs

Battery Management System

Since the cells of a LiPo battery do not drain evenly, precautions have been taken to ensure the safety of the battery. A battery charger with cell balancing capabilities was purchased to ensure even cell voltages after each charge. An Arduino Uno is used to monitor the voltage levels of the battery, and this is interfaced with a buzzer which will be used to indicate to the user when batteries are low.

Power Budget

Table 7 shows the power draw of major components in the electrical system, an additional 2.5% is added to each component to account for efficiency loss and minor components. Altogether, the system uses roughly 350W on average, when driving uphill the robot will draw about 750W. The 8000mAh battery operating at 22.2v equates to roughly 175 Watt-Hours, which will provide the robot with about 40 minutes of runtime at average power and approximately 15 mins at maximum power before needing another charge. The 25C rating of the battery allows for 200A of current to be drawn at any instant, which is more than enough to meet our maximum power needs.

Table 7: Power Budget

Component	Avg Power (Amps)	Max Power (Amps)	Voltage	Source
Motors	300W (13A)	720W (32A)	24 to 36 volt	22v Battery
LiDAR[1]	20W (0.9A)	45W (2.0A)	10v to 30v	22v Battery
Safety Circuit + Diagnostics	5W (0.4A)	5W (0.4A)	12V	22v Battery
Computer[2]	20W (1.0A)	76W (4.0A)	19V	19v Battery
Powered USB Hub[2]	5W (1A)	10W (2A)	5v	19v Battery (usb port)

Safety System

To ensure the safety of our robot, a wireless relay control system installed to work in tandem with a physical emergency stop button to control a solenoid. The wireless estop has been tested reliably out to 70m. The relay directly controls the solenoid which controls power to the motors.

The wireless relay remotes have two buttons that are configured into two modes: pause and kill shown in **Figure 5**. “Pause Mode” toggles the robot from disabled to active. It is intended for testing or when setting up the robot. “Kill Mode” will immediately cut power to the motors by activating the solenoid. In order to re-enable the robot after it has been “killed”, the physical emergency stop button must be cycled. This is intended to force the operator to walk up to the robot and make sure the robot is safe to operate again.

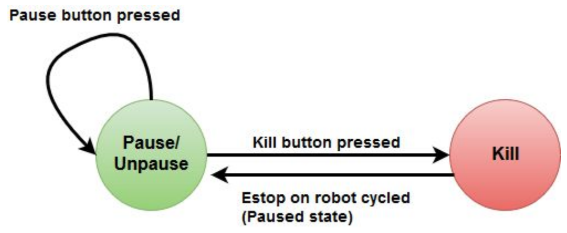


FIgure 5: Pause Kill state machine

[1] Lidar will only draw 45W in cold weather
[2] Not factored into drive time, as it is only used to power the computer and computer peripherals.

Indicator System

Ohm has indicators for many different aspects of the robot. The robot makes use of a three-colored tower light, each color designated to indicate one of three drive statuses. **Figure 6** shows the operation of the tower light

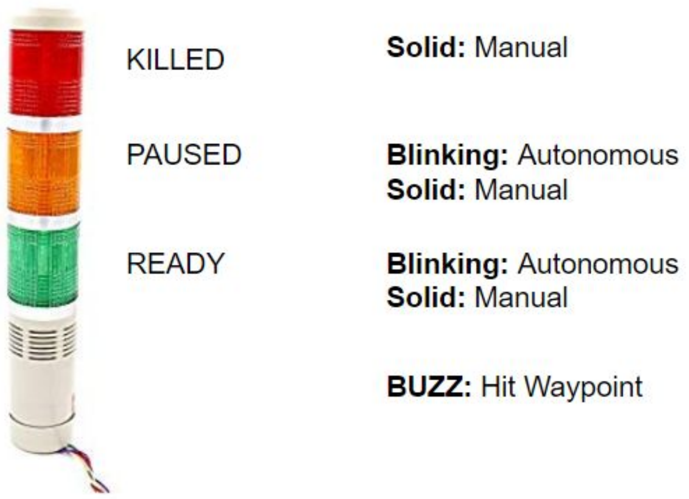


Figure 6: Tower light operation

Sensor Suite

Ohm use four main input sensors (descriptions provided below) to help it navigate the course. These sensors are all processed on a laptop computer which then sends the appropriate commands to the motor controller as shown in **Figure 7**.

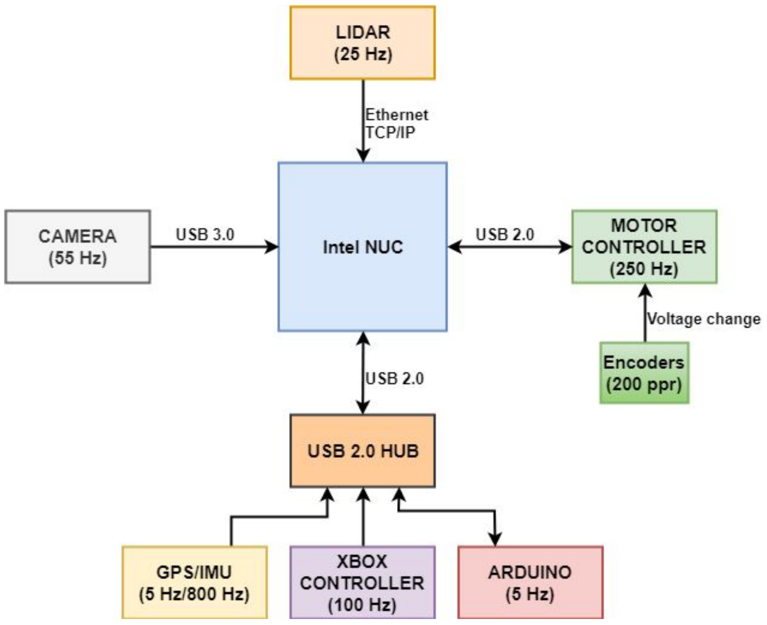


Figure 7: Hardware Interfaces

LiDAR

A SICK LMS-111 Lidar is used due to its high reliability and accuracy and has been implemented in various weather conditions. The scan range is 20m at a frequency of 25 Hz with an angular resolution of 0.25°, and has 270° field of view.

Camera

A Point Grey (Flir) Chameleon3 3.2 MP camera was selected to assist the robot in detecting the lanes, and potholes due to its significantly higher image quality, higher frame rate and IR noise reduction compared to a traditional webcam. The camera is fitted with 110° field of view lens and a linear polarizing filter. The camera resolution is configured to output at a resolution of 2048 x 1536 pixels, it is connected to the laptop via USB 3.0, and interfaced with OpenCV.

GPS

AluminOHM uses the VectorNav VN-300 dual antenna GPS. This system is used due to the increased positional and heading accuracy compared to a Garmin Marine GPS 19x HVS system the the team has used in the past. It also has a built-in 10-axis IMU which it uses in conjunction with a built-in Kalman filter to prevent large jumps in heading and position. The IMU is also used for navigation.

Encoders

To get accurate odometry, AluminOHM uses two Kubler Turck 200 PPR (pulses per revolution) quadrature encoders, one mounted on each motor. These encoders are connected to the Roboteq motor controller, which handles pulse counting. These values are polled by the software system to calculate the robot's position.

Processor

AluminOHM uses an Intel NUC small form factor PC to perform all computations and is the main interface for all sensors. It uses an I7-6770HQ quad core CPU with 8GB of RAM, running on Ubuntu 16.04 with ROS Kinetic.

SOFTWARE

Overview

AluminOHM's software in 2019 is a continuation of our goals from from 2018, namely system modularity, and establishing a baseline for future work. AluminOHM's software incorporates two new features: the utilization of the ROS Navigation Stack and slam_gmapping packages, and the introduction of a state machine. There were also several smaller changes to hardware interfaces to support the two major changes.

Navigation and Obstacle Avoidance

Navigation is handled by the ROS Navigation Stack (NavStack). The task of goal finding is broken into two parts, a global planner and a local planner. The global planner generates a path as a series of different positions and orientations to reach a waypoint. The local planner then attempts to follow this path while avoiding nearby obstacles.

In order to avoid obstacles, the NavStack creates an occupancy map and populates it with data from the camera and LiDAR. This map can also be initialize with a previously generated map, allowing us to use previous runs to to improve generated paths for the current run. We set the cell size of the map to 256 cm^2, as this allows the robot enough precision to mark non-traversable space without losing free space, while also keeping the size of the map in memory fairly low around 100,000 grid cells.

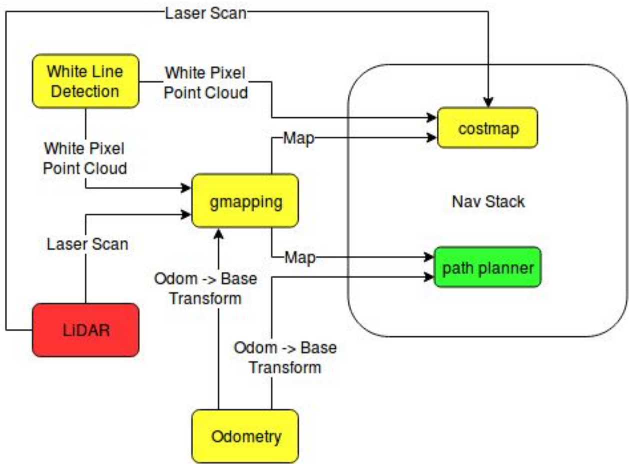


Figure 8: Navigation and SLAM subsystem

SLAM

In order to localize, AluminOHM uses slam_gmapping to produce a transform between the map and odometry positions. GMapping is fed the raw LiDAR scans and odometry to produce this transform. However, in certain parts of the course where obstacles are further away than the range of our LiDAR, we cannot use it alone to localize. To address this problem we convert the point clouds produced by the camera into LiDAR scans and provide this to GMapping as a secondary sensor, allowing the robot to localize anywhere in the field.

Lane and Pothole detection

To detect/avoid lanes and potholes the robot treats them both as obstacles. Since both types of obstacles are white, the robot looks for white in the image, and calculates how far those white pixels are relative to the robot. The camera detects objects by first doing a perspective transform to effectively give a top down view of the area it sees and flattens any 3D object into a 2D space. Since the field can be considered flat for the entire field of view, it is safe to assume that an object at a given pixel coordinate has an associated distance. Once the calibration is done, white pixel coordinates are passed into a line equation (shown below) which outputs the distances as a point cloud in meters. The point cloud data is then passed to the map.

$$U = a * Xp + b$$
$$V = c * Yp + d$$

Where ***U*** and ***V*** are the ***X*** and ***Y*** distance respectively in meters relative to the robot. ***a,b,c,d*** are constants. ***Xp*** and ***Yp*** are the pixel coordinates. **Figure 9** shows the the process, **Figure 10** shows the perspective transform and **Figure 11** shows the thresholding.

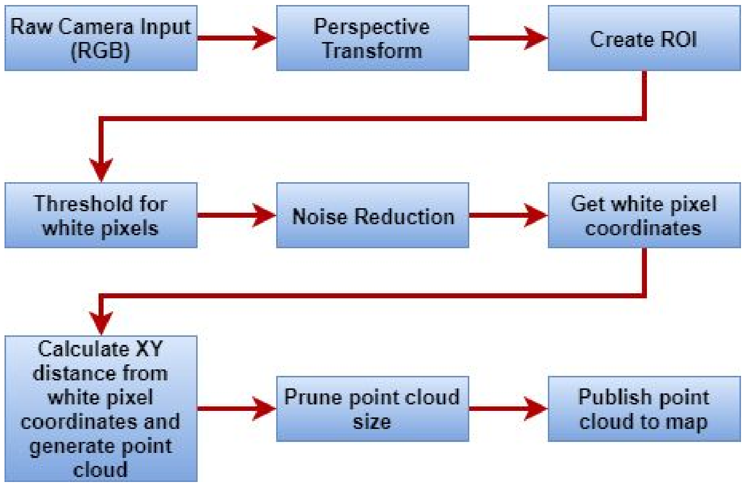

Raw Camera Input (RGB)
Perspective Transform
Create ROI
Threshold for white pixels
Noise Reduction
Get white pixel coordinates
Calculate XY distance from white pixel coordinates and generate point cloud
Prune point cloud size
Publish point cloud to map

Figure 8: Vision algorithm

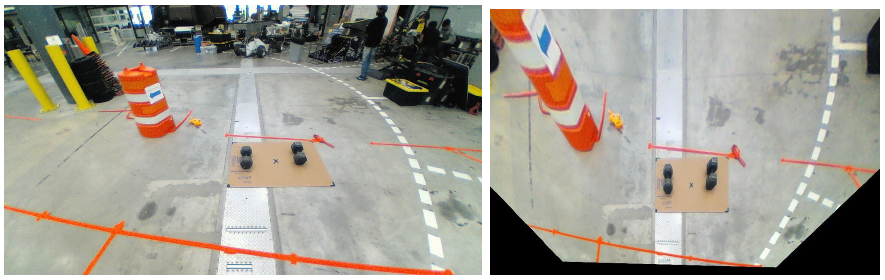


Figure 9: Before and after performing a perspective transform

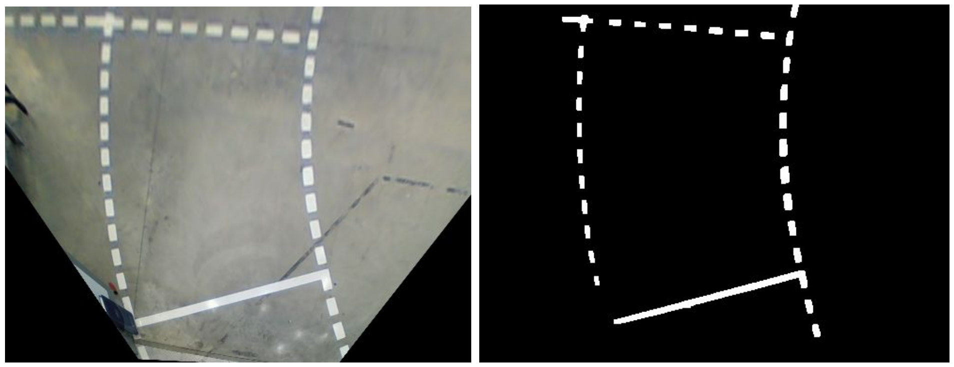


Figure 10: Finding the white lines

Goal Selection

At the beginning of the run, the robot loads a list of waypoints from a previously prepared text file. Waypoints may either be in DMS format, decimal lat/lon pairs, or as XY coordinates. Each waypoint is composed of a position and a target heading. The list of waypoints can also be related to a specific frame of reference, such as robot oriented, or map oriented. Once the goals have been loaded from the text file, the goal selection module feeds the waypoints to the NavStack, as each waypoint is reached.

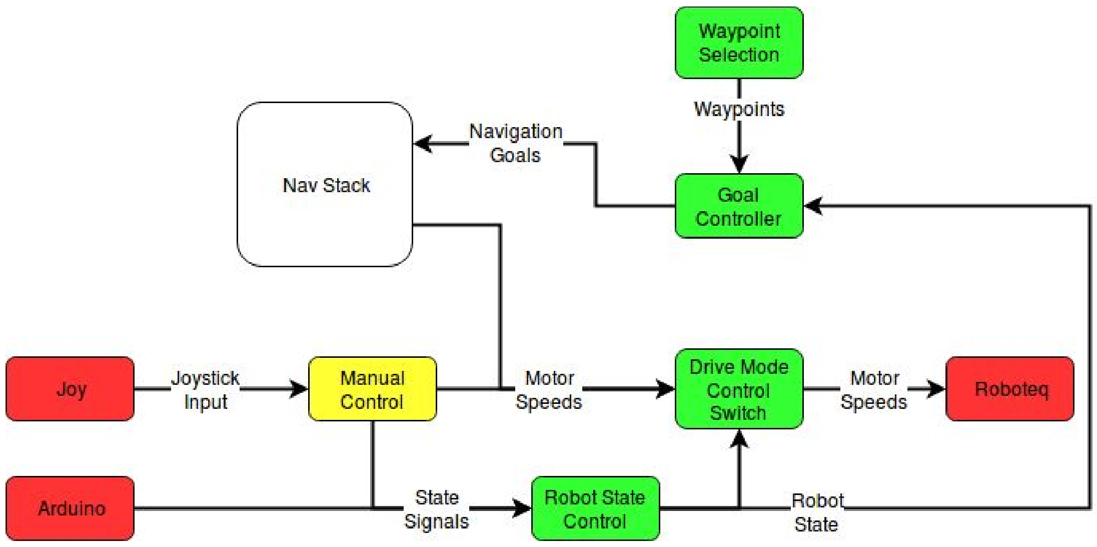


Figure 11: Robot state machine and associated modules

State Machine

AluminOHM uses two small state machine to handle switching driving modes and to handle kill and pause signals from the emergency stop system. This helps with separating out functionality that was previously copied across multiple components, decreasing coupling and making it easier to switch out components. Signals come into the state machines from the joystick and from the Arduino controller, which listens to the emergency stop system. The state is then published to all other software components.

The system is comprised of five states: Manual (M), Autonomous (A), Pause (P), Kill (K) and Low Voltage (L), R is a reset condition. The software subscribes to internal signals from other nodes as well as signals from the arduino. The machine state is then published out for the tower light and the controls system. Manual is the default state for safety concerns and as such is the only state that the Kill state can go to unless it receives a low voltage signal. On startup the machine is in the manual state. **Figure 12** shows a high level diagram for our robot controller state machine.

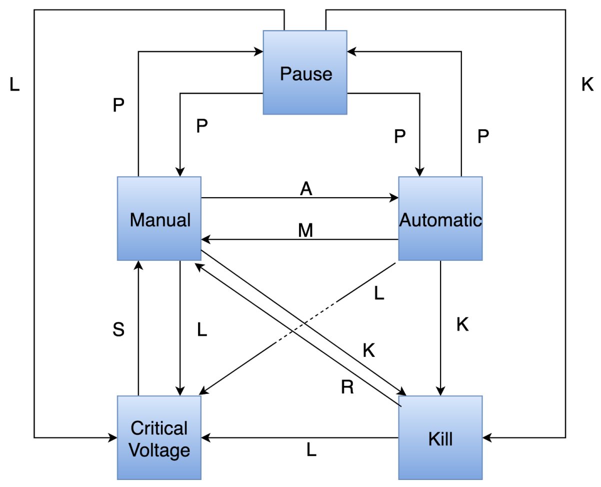


Figure 12: Robot Controller state machine.

FAILURE MODES

Table 8a-c describes the main points of failure for mechanical, electrical and software subsystems, along with severity and mitigation actions.

Table 8a: Failure Modes Mechanical

Possible Failures	Likelihood	Severity	Action
Water infiltration damaging components	Low	High, Damaged components may end competition	Panels deflect water and electronics are enclosed.
Payload	Low	Moderate, End of run	Payload sits in an enclosure and is tied down
Casters dig into the ground during backup	Low	Moderate, performance degradation	Reduce weight on back of robot

Table 8b: Failure Modes Electrical

Possible Failures	Likelihood	Severity	Action
Components come loose	Low	Moderate, End of run	Tie lines together and secure to electrical box
Battery dies early	Low	Moderate, End of run	Charge batteries not in use, don't over use battery
Physical or wireless e-stop fails	Low	High, End of run, possible DQ	Have spare relay, thorough testing to ensure risk mitigation

Table 8c: Failure Modes Software

Possible Failures	Likelihood	Severity	Action
Poor GPS Fix	Moderate	Moderate, performance degradation	Convert waypoints to robot-local frame, wait for better conditions
Inconsistencies in map	Low	Moderate, performance degradation	Correct inconsistencies in the map using image editors
Localization inaccuracy	Moderate	Moderate, performance degradation	Have spare relay, thorough testing to ensure risk mitigation

PERFORMANCE TESTING

Table 9: Performance Summary

Category	Requirements	Analysis
Speed	2.2m/s	Tune software to limit speed as little as necessary
Ramp	Capable of climbing up to 30° incline.	Tested on varying inclines. Confident to 30°
Reaction Times	Maintain a system update rate of 10Hz	Take advantage of configurable output rates, and limit rates in software where necessary
Battery Life	30 minutes on grass with gently rolling slopes	Performed endurance test on grass. Actual runtime ~1 hour
Distance of Obstacle Detection	Maximum obstacle detection with LiDAR is 20m. Robot reacts within 2m.	Tune robot to react within larger radius if necessary
Distance of Lane Detection	Detect white lines	Camera can effectively see only ~4m ahead and 1.5m on either side
Behavior in Dead end situations	Capable of navigating out of a dead end	Have robot turn in place/backup until a path can be found

CONCLUSION

This year the main goal was to have a solid foundation in software design and implementation for future iterations of the robot. Utilizing the iterative design process, we have constructed a durable and lightweight aluminum platform, implemented core software functionalities, robust safety system and laid the groundwork for future iterations.