University of Michigan Dearborn
Ohm Mk VI

Team Members: S. Mahimkar, M. Abraham, D. Vanden Berg,
Z. Nelson, K. Topolovec

Faculty Advisors: Dr. M.Putty, Dr. S. Rawashdeh

DATE: 5/15/2018
Team captain: S. Mahimkar

I, Dr. Samir Rawashdeh of the Department of Electrical and Computer Engineering at the University of Michigan Dearborn, certify that the design and development of this sixth iteration of the Ohm vehicle by the individuals on the design team is significant, unique to this iteration of the vehicle, and is equivalent to what might be awarded credit in a senior design course.

X S. Rawashdeh Digitally signed by Samir Rawashdeh, Ph.D. Date: 2018.05.15 18:18:14 -04'00'

ABSTRACT

This paper presents Ohm Mk VI, a robot designed and used by the University of Michigan - Dearborn for the 26th Annual Intelligent Ground Vehicle Competition (IGVC). Ohm MK VI, used in the 2018 competition, is based off of the platform 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 was completely redesigned to allow for a solid foundation for future iterations.

INTRODUCTION

The Intelligent Systems Club of the University of Michigan - Dearborn has entered the 2018 Intelligent Ground Vehicle Competition with two new and three returning members. The main goal of this year's team is to learn and establish an understanding in Robot Operating System (ROS) to improve the overall efficiency 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 undergraduate students, many of whom plan to participate in future competitions. The team member composition is displayed in **Table 1**.

Table 1: Team Ohm Composition

Name	Email	Class	Role
Siddharth Mahimkar	smahmka@umich.edu	Computer Engineering, Senior	Captain, Software
Matthew Abraham	mjabraha@umich.edu	Computer Science, Junior	Software Lead
Zachary Nelson	zdnelson@umich.edu	Computer Science, Senior	Software
Daniel Vanden Berg	djvanden@umich.edu	Electrical Engineering, Senior	Electrical Lead
Kenneth Topolovec	ktopolov@umich.edu	Electrical Engineering, Junior	Electrical

This paper will begin with a description of design innovations, then cover mechanical and electrical 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
Obstacle detection algorithm	Previous detection algorithm detected large groups of obstacles as a single obstacle and discarded contours	Group and average close-by points to maintain contour even for large groups of obstacles
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: Electrical Cost

Electrical	Qty	Unit cost	Price
LiDAR	1	$5,000	$5,000
GPS	1	$5,000	$5,000
Laptop	1	$1,100	$1,100
Camera	1	$20	$20
6s LiPo 8 AH (+)	4	$77	$308
6s LiPo 12 AH (+)	1	$80	$80
Battery Management (+)	1	$50	$50
Motor controller	1	$390	$390
Total Electrical Cost			**$11,948**

Table 3b: Mechanical Cost

Mechanical	Qty	Unit cost	Price
Frame	1	$260	$260
Motors	2	$450	$900
Total Mechanical Cost			**$1,160**

Table 3c: Vehicle Cost

Overall Category	Price
Electrical	$11,948
Mechanical	$1,160
Estimated Retail Price	**$13,108**
Actual Cost	**$438**

MECHANICAL DESIGN

The vehicle used for this year's competition is one that has been with the University of Michigan-Dearborn for some time now. It has participated in numerous IGVC and Autonomous snowplow competitions in the past. The vehicle is made primarily of plywood and uses a differential drive steering control scheme which is aided by a trailing caster. 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.

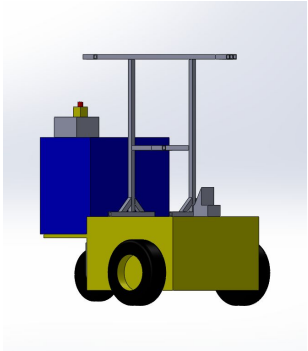

Figure 1: Robot design

Structure design

The mechanical design base of the robot was originally designed for the Autonomous Snowplow Competition in 2010, and has been repurposed for the past few years for the Intelligent Ground Vehicle Competition. The robot is made almost entirely of wood with four long metal threaded poles. The robot has three pieces of plywood, each which act as a level within the robot. The metal poles are placed vertically and threaded through each piece of plywood. Each level is secured to the metal poles using two nuts to hold each level in place.

The robot utilizes a single rear caster and propelled by twin 24 volt NPC DC motors with integrated 24:1 gearboxes, providing a maximum of .81 horsepower each and a maximum of 120 rpm. The tires are 0.33m and work well on a grassy field. There is an aluminum mast that houses the GPS and the camera. The battery is housed in the laptop cabinet where it is easily accessible.

Table 4: Vehicle dimensions

	Vehicle	Requirements
Width	0.81m	0.61m - 1.21m
Length	0.92m	1.21m - 2.13m
Height	1.46m	1.82m maximum
Mast height	0.96m	-
Mast length	1.01m	-

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	1	4.5 Kg	4.5 Kg
Mast	1	4.5 Kg	4.5 Kg
Frame	1	10 Kg	10 Kg
Total weight			46 Kg

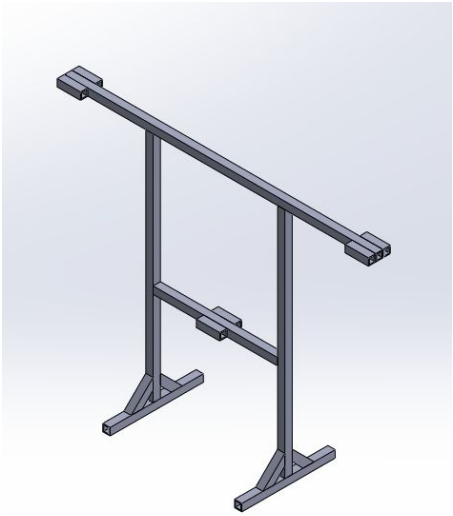

Figure 2: GPS mast

The payload tray was placed on the top of the robot between the legs of the mast for easy access to the payload. Silicone caulk was used to waterproof the laptop control box and other sensitive components. This waterproofing is designed to keep out moderate rainfall for a short time, until the robot can be moved to a sheltered area.

ELECTRICAL COMPONENTS AND DESIGN

Overview

This year's electrical design builds upon the design from previous years. However, 2 major changes were implemented to improve the performance and safety of of the robot. First, the lead-acid batteries were replaced with LiPo batteries. Switching the power source eliminated 85lbs from the total weight, increasing the performance of the new batteries and decreasing strain on the frame. It also necessitated the design of a reliable battery management system to maintain battery life and performance. Second, the circuit breaker was moved toward a higher potential to help eliminate the risk of floating voltages causing a short circuit.

Power Distribution System

There are three main supply voltages that power the components of the robot. 24v, 12v and 120v AC. **Figure 3** shows the component voltage breakdown.

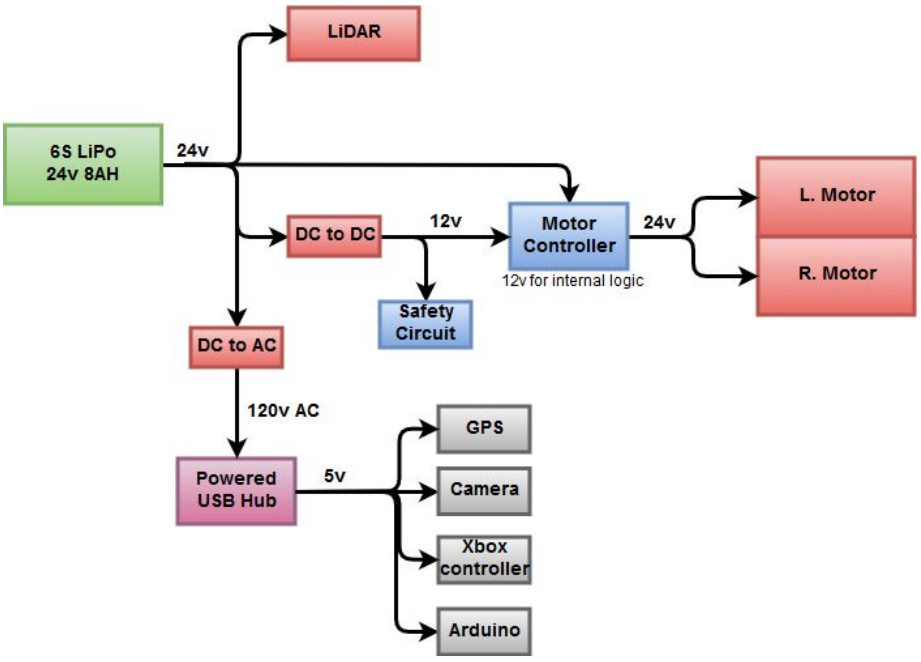

Figure 3: Component Voltage Breakdown

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.

Table 6: Battery specifications

Capacity	Nominal voltage	Overcharge	Charge time
8AH	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.

A low cell voltage monitor has been connected to the battery which sounds an alarm when any individual cell begins to fall below 3.4 volts. When the alarm activates a signal is sent to the Arduino microcontroller activating an LED indicator.

Power Budget

Table 7 shows the power draw of major components along with a 30W buffer to compensate for efficiency loss and minor components (i.e motor controller). The Powered USB hub includes all components connected.

Table 7: Power Budget

Component	Average Consumption	Max Consumption	Operating Voltage	Source
Motors (both)	200W	720W	24 to 36 volt	Battery
LiDAR	20W	45W	10v to 30v	Battery
Powered USB Hub	10W	10W	5v	DC to AC
Efficiency loss	30W	30W	-	-

The total power consumptions adds up to approximately 800W. Batteries have a 177W capacity. This gives the robot approximately 13 minutes of runtime at maximum power consumption. On average total power consumption adds up to approximately 260W, the total runtime is approximately 40 minutes.

Safety System

On the top of the robot, there is a safety light that indicates the drive mode as per competition requirements. This light operates at 12v and is controlled by an Arduino Uno with a relay shield.

To improve 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 4**. “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.

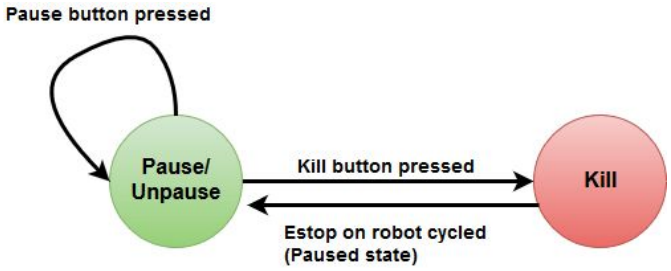

FIgure 4: Pause Kill state machine

Sensor Suite

Ohm use three 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 5**.

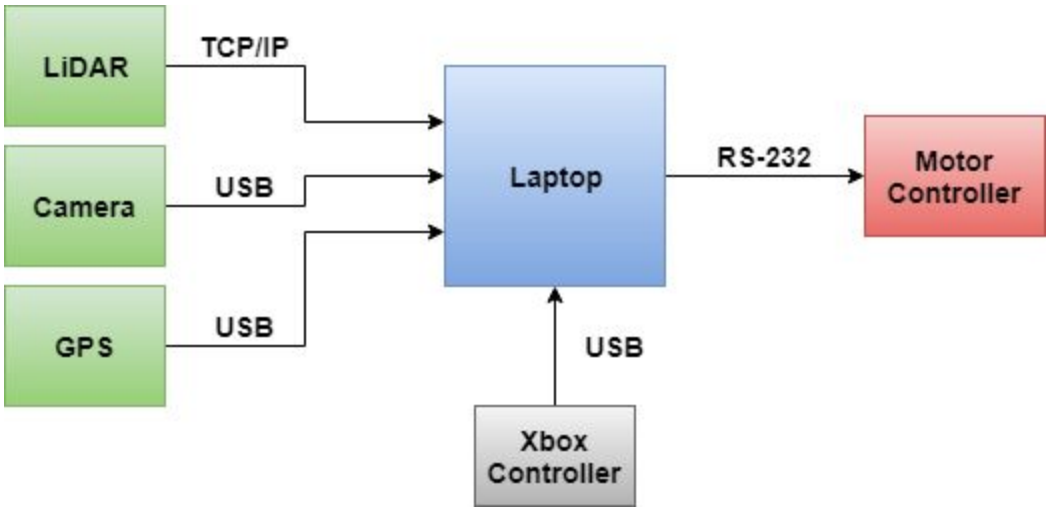

Figure 5: Software interface

LiDAR

Ohm uses a SICK LMS-111 Lidar. This LiDAR is used due of its high reliability and accuracy and has been implemented in multiple 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 wide angle camera was selected to assist the vehicle detect the lanes, potholes, and obstacles. The Logitech C525 HD Webcam was selected because of its overall cost, and 120° field of view. The camera resolution is configured to output a 820 x 468 pixel videostream, is connected to the laptop via USB, and interfaced with OpenCV.

GPS Unit

Ohm uses the VectorNav VN-300 differential GPS. This system is used because of increased accuracy and very high heading accuracy. It also has a built-in IMU which it uses in conjunction with a built-in Kalman filter to prevent large jumps in heading and position.

Processor (Laptop Computer)

Ohm uses a Lenovo Thinkpad X260. This laptop was used in previous years due to its robustness, small form factor, and durability. It uses an i5-6300U cpu, 8GB ram, and is dual booted with Windows 10 Pro and Ubuntu 16.04, the latter of which is used during competition. The processor is the main interface between sensors, and motors. Specifically, it takes input from the sensors and wireless controller and controls the motors accordingly.

SOFTWARE

Overview

The software for this year's robot is built in part upon the previous year's code base. These portions of the code base consist of hardware interfaces and functionality to transform sensor data into a form useful for decision-making systems to interpret.

Two main goals guided this years software design. First, the robot must have basic functionality above all else (i.e. lane following and obstacle avoidance). Second, the systems designed for that task must be well documented and modular, such that new functionality could be added or portions replaced in the next iteration without disturbing basic functions.

All software running on the robot is written using the ROS framework, which allows for separate programs to communicate with each other over TCP/IP. This enables programs (called nodes in ROS terminology) to have a singular focus, e.g. identifying obstacles, motion planning, or communicating with motors.

The robot operates in two modes, manual and autonomous. While in manual mode, the indicator on the top of the robot stays solid to indicate it is in manual mode, as per competition rules. Control in this mode is given by a human operator using a joystick or gamepad, generally a standard Xbox 360 gamepad.

The main control algorithm for the robot runs in its own node, and sends a linear velocity and angular velocity for the robot to maintain. The speed of the robot is fixed, though that fixed speed can be changed in configuration. The main algorithm controls the heading of the robot, pointing it towards GPS waypoints, and away from obstacles and white lines.

Obstacle Detection

Obstacle detection is done in three steps, each in their own node. First, the data is read in from the LiDAR over an ethernet connection. The initial data set consists of 1080 distances, each representing a single ray from the sensor. First, these distances are pruned to remove max and min range distances. In the second step, the distances are converted into XY coordinates and transformed w.r.t. the robot. In the third step, points are grouped by distance, then split into subgroups and averaged. This allows us to maintain the contour and clustering found in the original data, while having a smaller footprint data set that is easier to process and pass around from node to node. This process is show in a simpler example in **Figure 6**.

To handle noise in the data, outliers are ignored up to a configurable amount of times. Ignored points are left in the data in case they are part of another obstacle.

In order to transform the detected obstacles into a format usable by the main algorithm, the point data must undergo one additional transformation. The motion planner must know the ranges of headings that are unoccupied by obstacles. These ranges are defined by a starting heading and an ending heading, both relative to north. The first range starts at the edge of the

LiDAR's sensing range (roughly 135° offset from the current heading) and continues until an obstacle is found that is close to the robot. The end heading of the range is padded to ensure that a buffer exists between free headings and occupied headings.

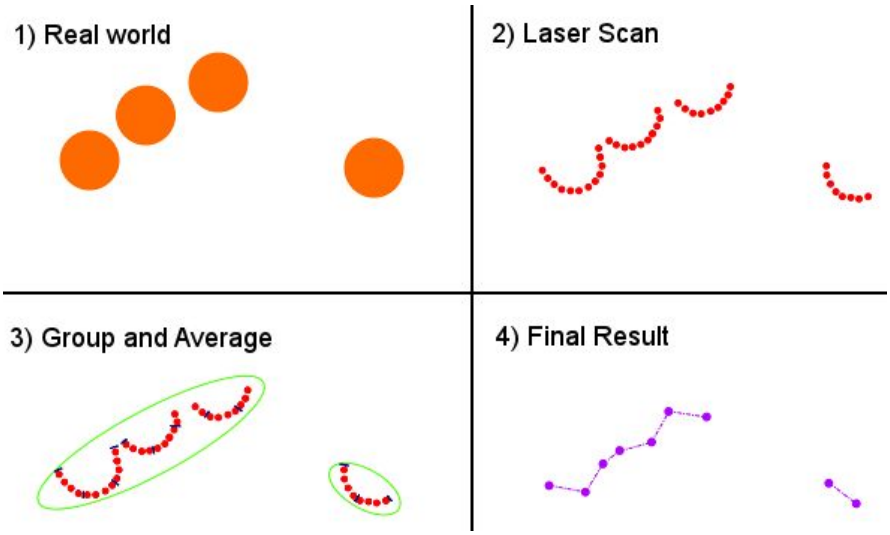

Figure 6: Obstacle detection process

Lane following and Pothole detection

To detect/avoid lanes and potholes the robot treats them both as path blocking. The first step is to resize the original image and perform a perspective transform, then change the colorspace to HSV, filter for white pixels, and filter out noise. This new image will then be compared to a set of 21 templates which mimic the vehicle turning radius. There are 10 templates that have various angles to the left and to the right and one through the middle. Each of these masks has an associated turn angle which is calculated during initialization with the middle template having a turn angle of 0°. The vision algorithm iterates through each template and calculates the number of intersections that each template produces when a bitwise AND is performed on the binary image. Templates that have intersections below a certain threshold are grouped by their turn angles and these angles are sent to the main algorithm. **Figure 7** and **Figure 8** show the process and conceptual examples respectively.

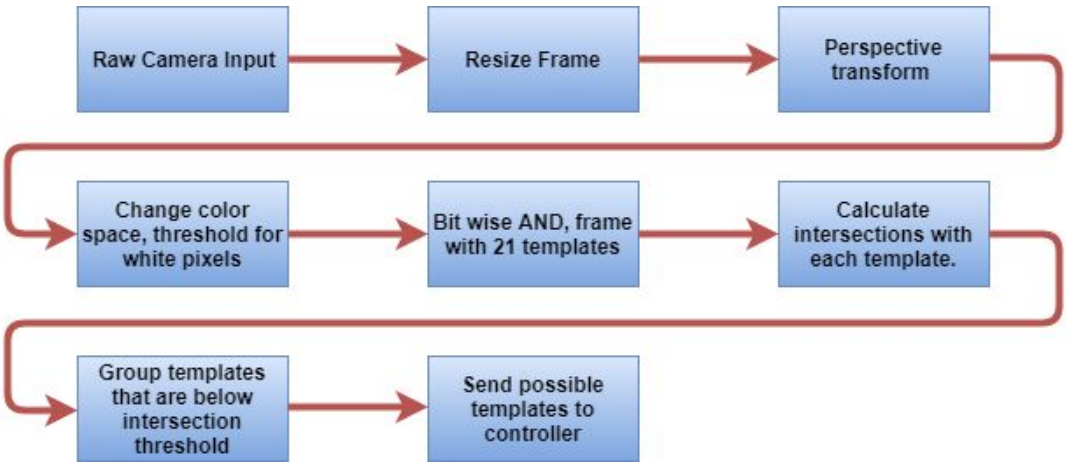

Figure 7: Vision algorithm

Figure 8: Potential paths the robot could take. Red indicates intersections. Ideal path is on the right.

Goal Selection

At system startup, the ordered list of waypoints is loaded and made available to the main algorithm. The waypoints can be either in the DMS format (degrees minutes seconds) or as decimal latitude and longitude pairs. If they are listed in DMS, each waypoint is converted to decimal latitude and longitude format when the waypoints are loaded at startup. Waypoint listings in either format also include a target heading in degrees, relative to north.

After all waypoints are loaded, the main algorithm must request a single waypoint and hit the target position and heading before another request is made. Upon hitting all the waypoints, the robot then targets its starting position as its final waypoint.

Mapping

During testing of the previous year's code, we found that the mapping solution and path planning solution were too tightly coupled to be useful. As a result, we decided to forgo a global mapping solution in favor of ensuring basic functionality.

Motion Planning

The motion planning algorithm is the main controller for the robot, and works in 3 stages. All sensor data feeds into the algorithm, and aside from human control, it is the only source of motor control for the robot. First, the algorithm checks whether it has hit the current waypoint (including its heading), and updates the waypoint if necessary. Next, it computes the most direct heading needed to hit the waypoint.

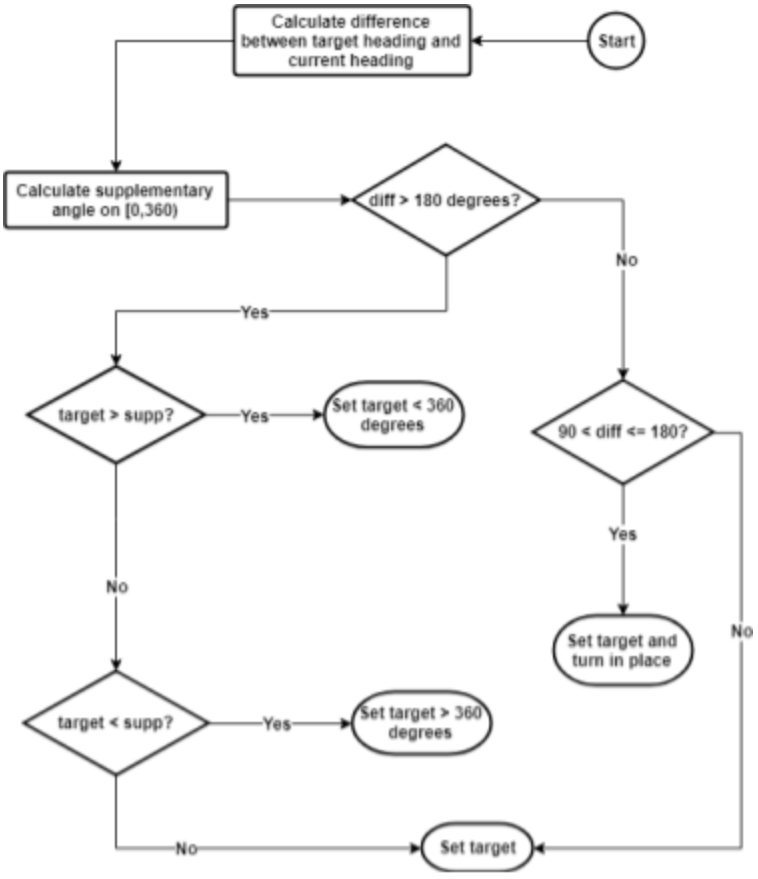

Figure 9: PID conditioning algorithm

Once the desired waypoint heading is calculated, the algorithm then looks for an unobstructed lane detection heading close to the desired heading, and checks the heading against the range of unobstructed headings provided by obstacle detection. If a suitable heading is found that is found in both lane detection and obstacle detection, then it becomes the new desired heading.

If no agreeable heading is found, the robot will instead turn in place and attempt to find a new heading that it can travel through.

Because the heading is a range that wraps around at its end points, headings supplied to the PID controller must be manipulated to produce the desired behavior. The algorithm for this is shown in **Figure 9**.

FAILURE MODES

Table 8 describes the main points of failure for the vehicle along with severity and mitigation actions.

Table 8: Failure Modes

Possible Failures	Likelihood	Severity	Action
Low battery	Low	End of run	Have multiple charged batteries on hand, swap batteries after each run
Loss of GPS heading	Low	Degradation of performance	Calibrate onboard compass to compensate for heading loss
Oversaturated camera input	Moderate	Loss of lane tracking	End run, tune color-based lane detection
Sensor Failure	Low	End of run likely	Software stops robot until failure is rectified
Wireless Emergency Stop Failure	Very Low	Possible damage to robot or injury	Software and hardware emergency stop options have been implemented.

SIMULATION

In the process of writing, testing, and debugging the software, it was necessary to simulate while the robot was in the process of hardware maintenance and electrical redesign. Components were tested separately. For some components, such as the main control algorithm, the solution was as simple as supplying fake input data to each of the functions. **Table 9** shows the simulations used for the major components tested.

Table 9: Simulations

Component	Simulation
Main Algorithm	Spoofed input data from Camera and LiDAR
Lane Detection	Use previously obtained videos as input
Obstacle Detection	MobileSim 3 and previously obtained LiDAR data

For the obstacle detection simulations, it was difficult to spoof data, so instead the Adept MobileRobots MobileSim 3 simulator was used to generate LiDAR data. However, it did not have the capability to simulate camera or GPS inputs, so different methods described above were used to test those components.

PERFORMANCE TESTING

Table 10: 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 ~3m ahead and 1m on either side
Behavior in Dead end situations	Capable of navigating out of a dead end	Have robot turn in place 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 implemented core functionality and laid the groundwork for future iterations. While we would have liked to add more advanced features such as mapping in time for the 2018 competition, the robot presented here is a safe, robust, and reliable platform, suitable for even more improvement.

REFERENCES

Bowyer, M., Mahimkar, S. Aitken, E. and Ferracciolo, B. (2016). *OHM (Ω) MK-IV*. Dearborn.

Mahimkar, S., Abraham, M., Vanden Berg, D. and Ferracciolo, B. (2017). *Ohm 5.0*. Dearborn.

Wiki.ros.org. (2018). *Documentation - ROS Wiki*. [online] Available at: http://wiki.ros.org/ [Accessed 15 May 2018].