Warrior Robotics Team

Project Veronica | IGVC 2021

Faculty Advisors

Dr. Abhilash Pandya [ECE] | apandya@wayne.edu

Dr. Marco Brocanelli [CS] | brok@wayne.edu

Team Captains

Nnamdi Monwe [CS] | President | nnamdimonwe@wayne.edu

Lloyd Brombach [CS] | Vice President | Lloyd@wayne.edu

Keena Pandya [ISE] | Managing Director | kpandya@wayne.edu

"I certify that the design and engineering of the Wayne State University Robotics Team has been significant and equivalent to what might be awarded credit in a capstone design course".

Abhilash K. Pandya, Ph.D.
Associate Professor Department of Electrical & Computer Engineering

Marco Brocanelli, Ph.D.
Assistant Professor Department of Computer Science

https://warriorrobotics.eng.wayne.edu/

Contents

The Team

Warrior Robotics is a student organization that is housed under the College of Engineering at Wayne State University. We are a team of engineering students comprising of both undergraduate and graduate-level students with a wide variety of disciplines. We share a common interest in building and testing robotic systems. Since robots are complicated assemblies, we created an interdisciplinary team to tackle various parts of our development and integration cycles.

This year the team has introduced a new vehicle called Veronica for IGVC 2021, which is the second generation of our previous vehicle, Joy from IGVC 2019. The new vehicle has an upgraded drivetrain, modular hardware, an improved electrical system and has gone through a complete software overhaul.

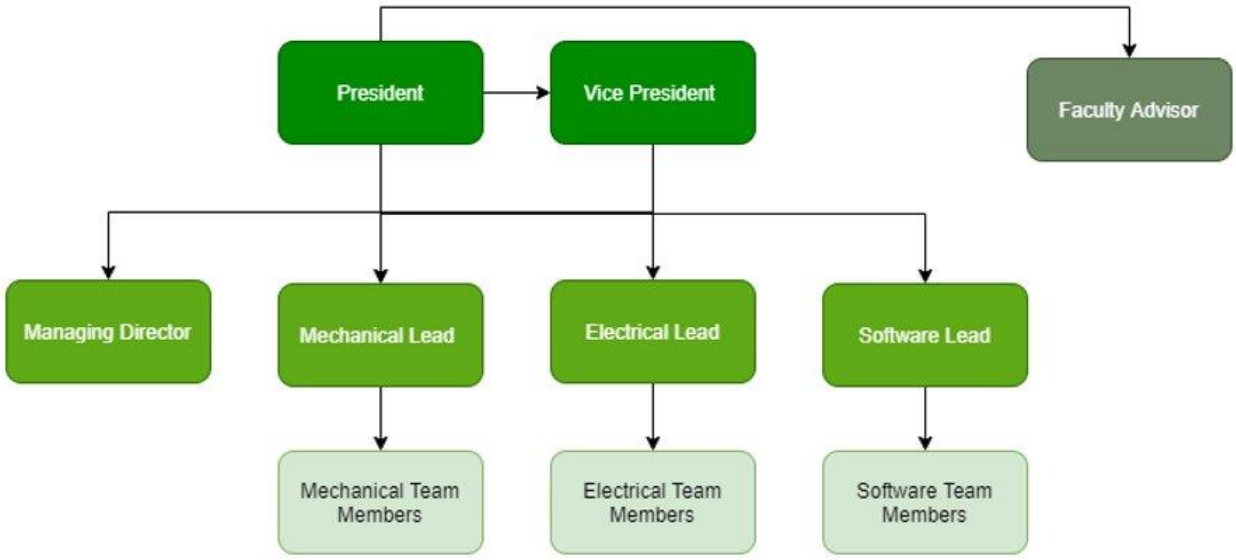

Figure 1: Our Organization's Structure

Figure 1.0 shows the team structure. We have a team lead for the electrical, mechanical, and software teams, respectively. These team leads and the administrative team, which comprises the organization's president, vice president, and managing director, report to our faculty advisors. Segmenting the team in such a manner ensures a proper chain of command and accountability for significant components of our builds.

Our team has spent countless hours designing, building, and testing our competition robot for this year. Below is a list of students who invested their free time (this activity is not yet approved for Capstone or course work) in the development of Veronica for IGVC 2021.

Name	Degree Program	Sub Team
Nnamdi Monwe	Computer Science	Admin/Software/Mechanical/Electrical
Lloyd Brombach	Computer Science	Admin/Software
Keena Pandya	Industrial Engineering	Admin/Mechanical/Electrical
William Jackson	Mechanical Engineering	Mechanical/Electrical
Varshith Solipuram	Mechanical Engineering	Software
Cameron Hanson	Electrical Engineering	Software
Ciah Green	Computer Science	Software
Aayush Patel	Computer Science	Software

Design Overview

The team learned a lot from the first-year experiences (especially from the mentors and 2019 evaluation/judges) during the design and competition of their previous 2019 IGVC robot. In the 2019 version, not enough effort was put into the initial design of all our subsystems. This resulted in various points of failure for the vehicle. Our mechanical design was very functional but was designed without considering efficiency. It was not as efficient at transferring all the power from the motors to the ground. The lack of an inefficient drivetrain resulted in additional strain to the power system, which had to be over-engineered to compensate for it. It also had an adverse effect on our software as we had to change our approach to implement our planning algorithms. In addition, we did not have redundant hardware available for quick swap-out, and the failure of some components (motor drivers, for instance) had us scrambling at the last minute. We have fixed many of the issues with our design this year.

This year we designed Veronica from the ground up to be modular and maintainable for all our subsystems. We also adopted a design review process in our development lifecycle to

improve the overall quality of our build. We utilized good design practices that led to a lighter vehicle, a safer and more expandable electrical system, and a robust software package that can be modified for future use.

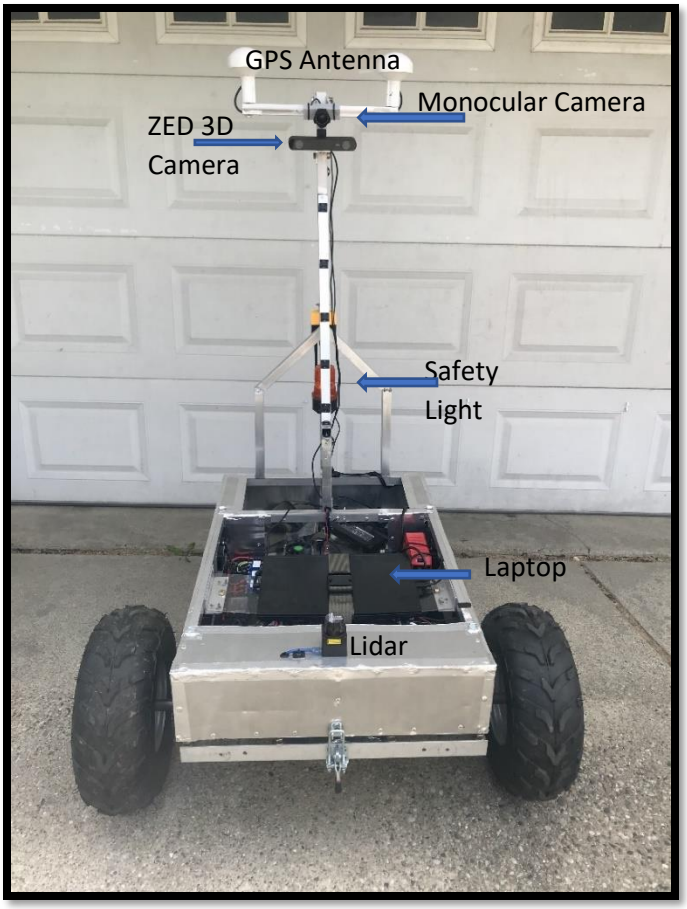

Figure 2: Fully assembled robot. Note that all work was done in an off-campus home/garage due to limited access.

Mechanical Systems

Overview

Veronica is a two-wheeled differential drive robot. Its structural elements are made entirely out of aluminum. The system is powered by two independently driven DC motors that have been geared up to increase the torque output to the wheels. It has a broad and stable base yet is light enough to enable agility and maximize its maneuverability on the course.

Chassis Design

The chassis is comprised of two main sections. The upper section is constructed out of one square-inch aluminum tubing and one-inch aluminum angle; then, it is wrapped in a thin aluminum sheet. This allows for a super lightweight upper section in which all the electronics and critical systems are housed.

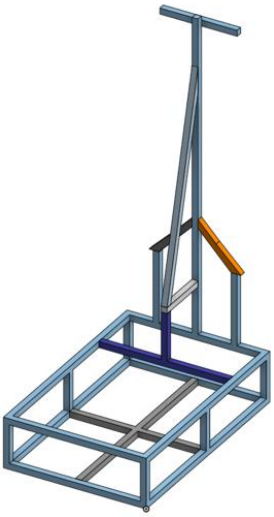

Figure 3: Upper Chassis Assembly

The lower section of the robot is also constructed from similar materials but with a greater wall thickness to ensure the integrity of the drivetrain. Additionally, we opted to use aluminum brazing technology (as seen in Figure 4.0 below) to fuse portions of the frame to reduce weight and increase overall structural integrity.

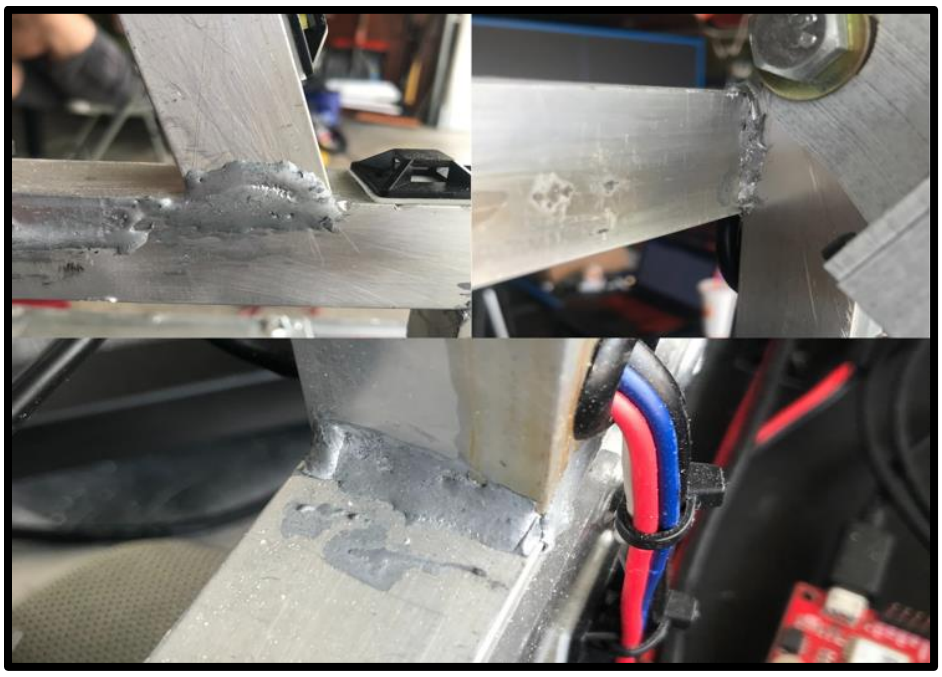

Figure 4: Brazed Aluminum Joints

Since both frame halves were developed to be independent of each other, we had to use reinforced toggle clamps to attach them together. This enables the robot to be easily detachable and allows for improved maintainability.

Figure 5: Toggle Clamps

Drivetrain

Veronica's drivetrain is similar to that of our 2019 IGVC robot because we are still using a hybrid system of gearboxes and chain linkage. The significant difference is that we drastically reduced the gearing ratio of the gearbox down to 20:1 as opposed to 60:1. We did this to reduce the stresses caused by excessive torque to both the lower chassis and the drive sprockets. The placement of all the motors, gearboxes, and axle bearing can all be adjusted, making the robot more serviceable.

Sensor Placement

Another critical area we overlooked that hindered the performance of our previous vehicles is the placement of our sensors. This year we were more vital with Veronica's design as we took sensor placement into account when designing the upper section of the chassis. We made sure to make the mast extra tall to accommodate the GPS Antennas and help increase our camera's field of view. We also expanded the frontal area of the chassis to accommodate a Lidar or an additional stereo camera without having to tamper with the chassis. Our lower latency sensors and instrumentation that require USB access are all housed within reasonable proximity to the host computer.

Weather Proofing

The team took extra measures to make Veronica water resistant this year. We started by surrounding the bottom portion of the chassis, where the motors and encoders are housed with a protective mylar covering. The mylar serves as a protective barrier against mildew from the grass and other debris that can interfere with the robot's operation. The upper chassis also employs several protective measures against the elements. We used a thick piece of acrylic to cover the top of the chassis and used foil tape to seal off any exposed gaps between the aluminum. In addition to this, we also added water-resistant intake ducts with filters to pull in fresh air for cooling our electronic components.

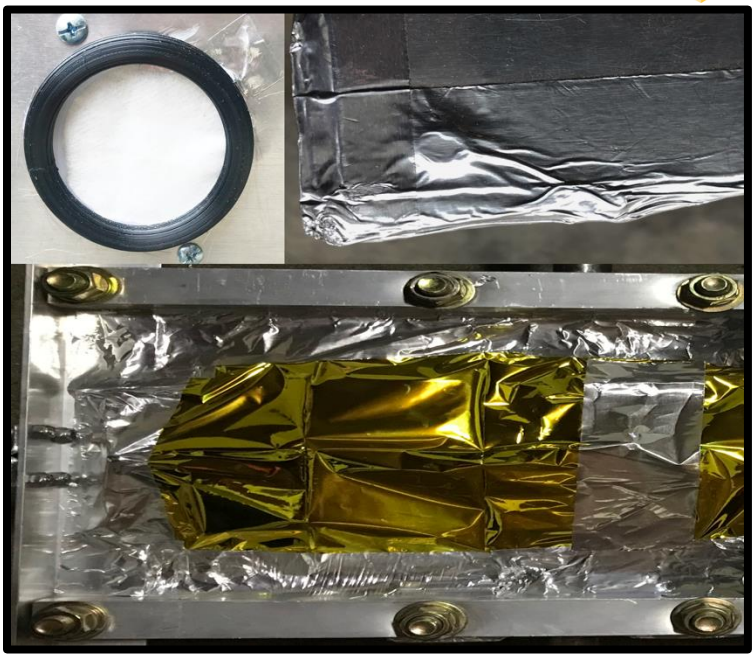

Figure 6: Waterproofing Measures

(Mylar sheet [top left], foil tape [top right], intake ducts [bottom])

Electronics

Overview

The electrical system of the robot is very complex, robust, and packed with a whole host of safety features to ensure the reliable operation of the robot. The electrical system is segmented into three separate subsystems comprising a power circuit, auxiliary circuit, and an inverter circuit.

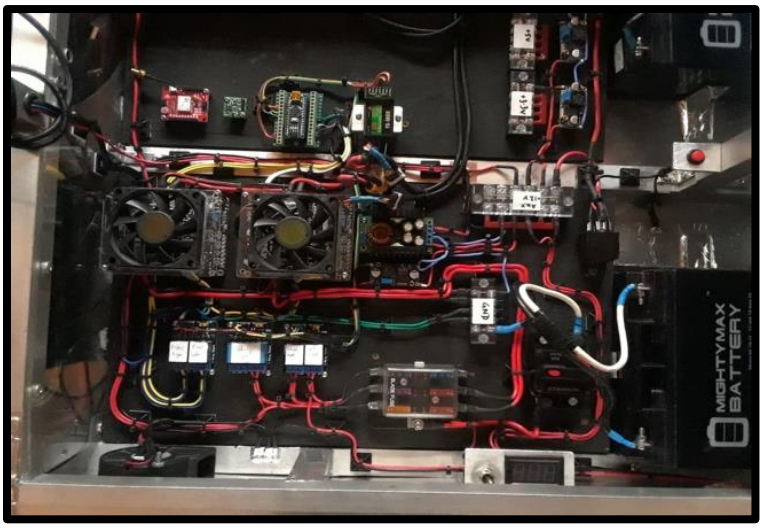

Figure 7: Preliminary Image of Power and Auxiliary Circuit

Mounting

Due to the constantly changing nature of the robot, we wanted a mounting system that can allow us to shuffle around components and that would not degrade in performance over time. To satisfy those requirements, we chose to mount the electrical systems on a flexible piece of light plywood. Doing this allowed us to make several backup pieces in case the current one gets severely damaged. It also allowed us to templatize the mounting of components to enable them to be hot swapped. In Figure 9.0, we can see the mounting panel covered in a grid of standoffs for easy placement of the electronics.

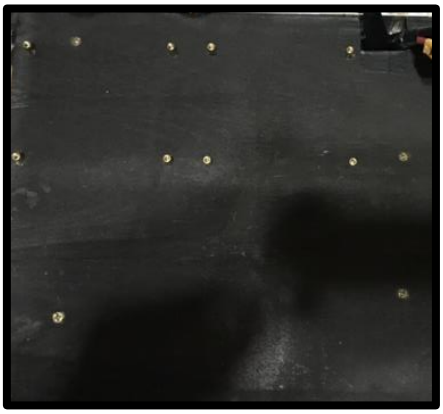

Figure 8: Mounting Panel for Electronics

Power Requirements

To design the robot in a safe and reliable manner, we needed to know our theoretical peak power consumption. Taking this peak power consumption and other factors like the current draw of each component, we can source the appropriate wire gauges and connectors to ensure the integrity of Veronica's electrical system.

Component	Quantity	Power Consumption (V * I = W)	Total Consumption
MSI GF65 Gaming Laptop	1	19.5V *9.23A = 180.0W	180.0W
ZED 2 Stereo Camera	1	5.0V * 0.380A = 1.9W	1.9W
Sparkfun GPS-RTK NEO-M8P-2	1	5.0V * 0.035A = 0.175W	0.2W
Cytron MD30C Motor Driver	2	12.0V * 30.0A = 360.0W	720.0W
Arduino Nano	2	5.0V * 0.019A = 0.095W	0.2W
Relay	3	5.0V * 0.020A = 0.1W	0.3W
Intake Fans	2	12.0V * 0.18A = 2.16W	4.32W
Safety Light	1	12.0V * 1.25A = 15W	15W
Cooling Fans	2	12.0V * 0.30A = 3.60W	7.2W
			929.12W

Table depicting the total power consumption of the robot's components.

The robot's maximum theoretical run time at peak power can also be calculated from the total consumption power. To accomplish this, we need to use the formula:

$$Battery\ Life = \frac{Battery\ Capacity(Ah)}{Current\ Consumption\ (A) * (1 - Discharge\ Safety(about\ 20\%))}$$

Total Power consumption of the system: 929.12W
Voltage rating: 12V

Battery capacity is 18Ah, and we want a 20 percent discharge safety net, and our device consumes about 77 amps, so we would be able to run the device for 0.187hrs or a total of 11 minutes and 22 seconds at peak power.

Power Supply and Distribution

Veronica's electrical system is split up into two subsystems: a primary circuit and an auxiliary circuit. The main circuit is used to handle more power-hungry applications such as powering the motors and safety light. The auxiliary circuit, on the other hand, is used for powering lower consumption devices such as our battery monitor and relay banks. The main advantages of separating the two systems are improving battery life, having a reliable power source for critical sensors, and making the system easier to troubleshoot. Figure 9 shows the component map of the system. The figure shows the power supply and the invertor (used for addition power to the laptop) as well as the other main components and electronics of our system.

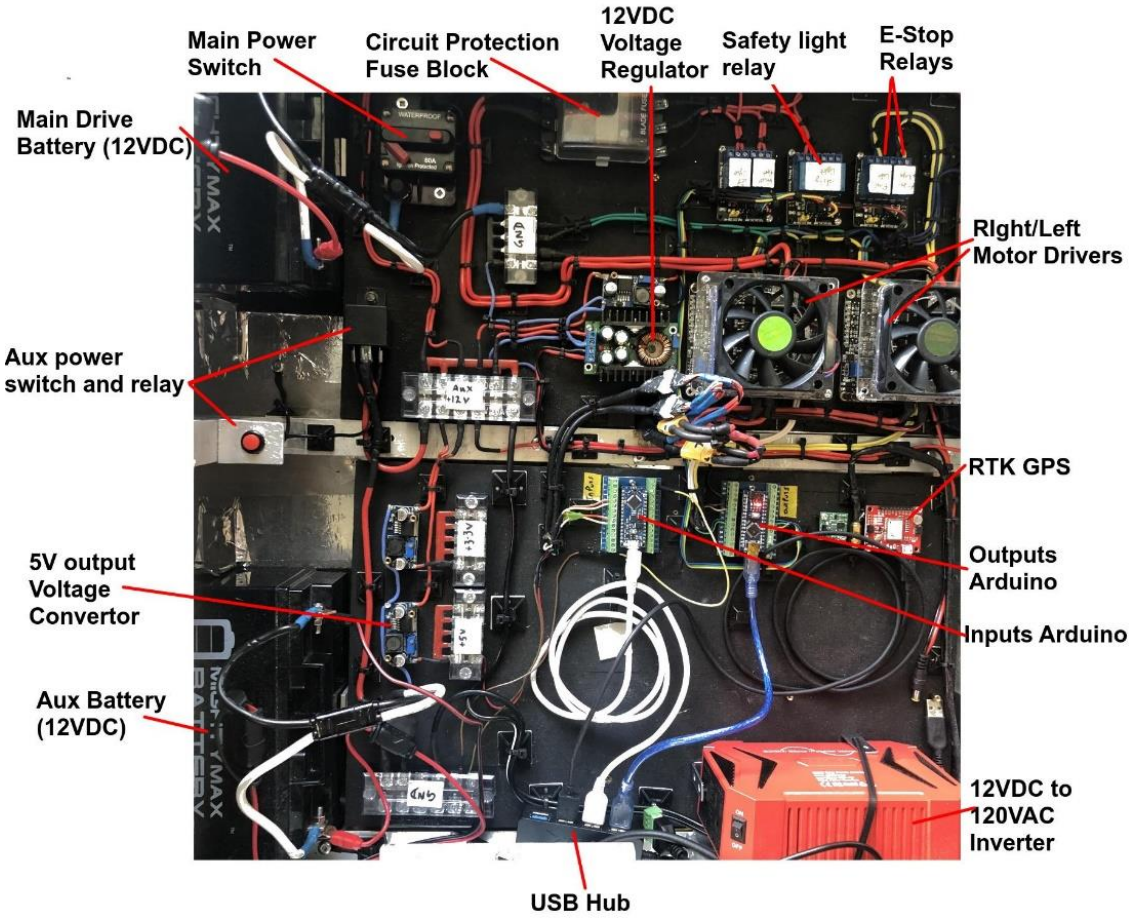

Figure 9: Component Map

E-Stops

Our robot complies with IGVC Auto-Nav rules standard and has a mechanical and wireless E-stop. The two emergency stop systems work both independently and in tandem to ensure proper vehicle disengagement in the event of an anomaly. The mechanical E-stop works by triggering a relay to cut off the PWM signals going to the motor drivers. Cutting of the PWM signal ensures the motor drivers slowly bring Veronica to a halt rather than abruptly jerking the entire system.

The wireless E-stop operates in the same manner as the mechanical E-stop, but it sends the trigger signal over a long-range radio rather than being hardwired directly to the system. The receiver for the wireless E-stop is also connected to the same relay as the mechanical E-stop. Wiring the two systems in parallel allow the motors to be cut off by either of them without jeopardizing their integrity.

Figure 10: Mechanical E-stop (left) & Wireless E-stop (right)

Software

Overview

The primary software for the competition uses a Robot Operating System which is a distributed set of nodes for each of our sub systems. At first the system (Ego node) waits for which mode that is selected. For instance, if a qualifying round needs obstacle detection, that particular mode would be selected by the user and the system would launch the appropriate files to enable that mode. The Ego node has at is disposal has several packages of software to choose from include, Perception, Navigation, Drive, Localization Information, and an Alexa package. These packages have many different nodes which subscribe to inputs and publish outputs. For example, the node for path planning in the Navigation package, subscribes to the robot's location and map data as well as a goal location and publishes a list a waypoint to navigate to that goal while avoiding obstacles. These waypoints that are published are subscribed to by the motion planner. In turn

the motion planner publishes velocity commands to the motor control system which is an Arduino connected to the motor drivers (See Figure 11).

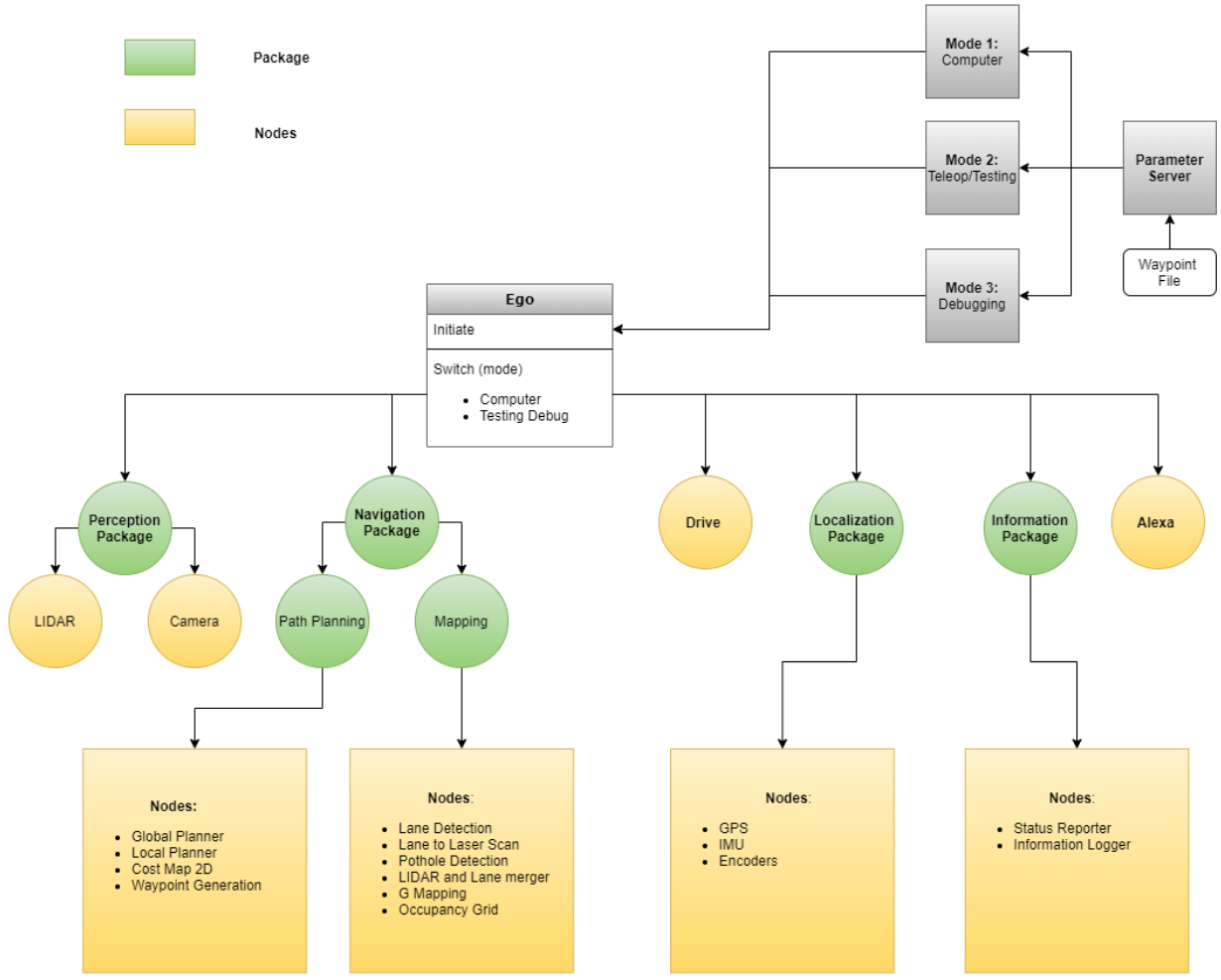

Figure 11: Node/ Package structure of the software

Software Architecture

Figure 12 shows our node structure and the various information that is passed between the nodes. For example, the motion planner publishes a velocity command and the node 'twist_to_motors' subscribes to the velocity command and publishes left and right wheel target velocities. The movebase costmap plugin subscribes to odometry and sensor data to create a map that is then published for our path planning nodes.

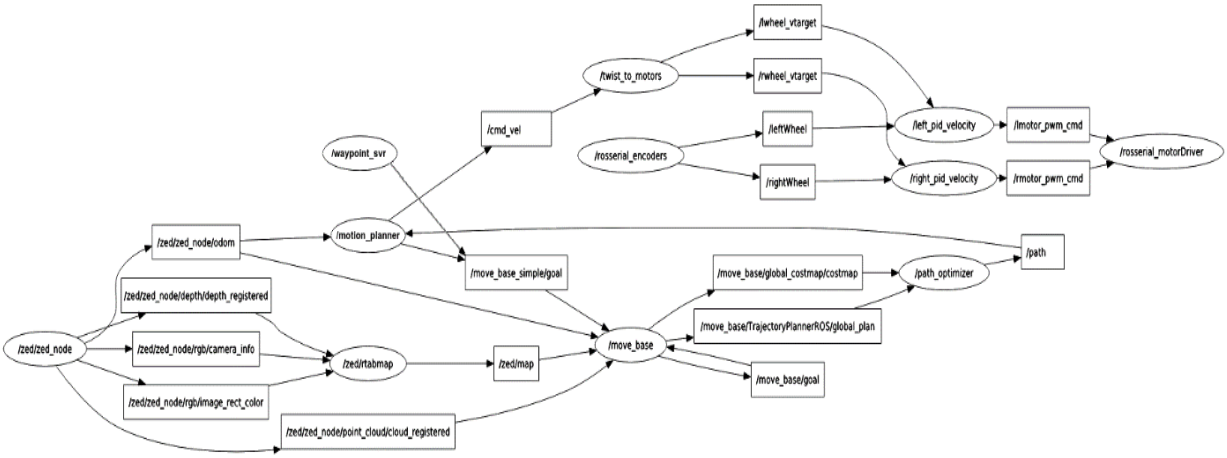

Figure 12: High level Communication inputs and outputs of the nodes of the software system.

There are several different modes that the system can go into. Here we describe the major modes.

Debugging Mode

This launches with sub-mode as arguments. If hardware is chosen, it will only initialize Camera, LiDAR, GPS, IMU, Encoder and look for positive message publishes. The status LED bar will indicate which sensor is working. If software is chosen, the master bag file of all raw sensor data is played back and is checked against expected results. Also launched a visualizer to make sure any software changes work as intended.

Lane Mode

This mode is only to be used for the lane-keeping qualification portion of the competition. The robot will rely on vision and Odom data to stay within the bounds of the lane. This can use the same one as the main competition mode.

Alternative launch for lane keeping.
Will directly calculate the cross-track error from images use the PID follower for pure lane keeping.

Speed Mode

This mode is only to be used for the speed testing qualification portion of the competition. The robot must maintain a minimum speed of 1mph and a maximum speed of 5mph while simultaneously lane keeping. Will launch the same file as lane keep mode.

Obstacle Mode

This mode is used for the obstacle avoidance qualification portion of the competition. The robot has to demonstrate that it can avoid static obstacles placed on the qualification course.

Waypoint Mode

This mode is used for the waypoint navigation qualification portion of the competition. The robot will be assigned GPS waypoints in which it has to fall within one meter center of accuracy.

Status Check

This menu option will be used to determine the health status of all the critical systems that are onboard the robot. Essentially just the hardware part of debug mode.

Kill All

This menu option will be used to quit all active robot operations. Uses '-9' (SIGKILL) flag so all nodes, including roscore terminate immediately but keep the GUI open so that other processes can be relaunched.

Image Processing

In addition to point-cloud data that provides obstacle and free-space information for our mapping software, images from the on-board cameras are used for lane and simulated pothole detection. Using the OpenCV computer vision software library, images are first grabbed, blurred, and undistorted. This undistorted image is then processed with a binary thresholding algorithm that determines which parts of the image are likely lane lines or potholes. This lane line data is cropped and transformed in perspective from camera-view to a birds-eye-view to it can be plotted into an occupancy grid. Since our early mapping software (Gmapping) accepted only laser scan data for obstacle input, we convert the lane data to the format of a ROS laser scan message that can be merged with data from the laser scanner. Our new mapping software (RtabMap) can also accept laser scan data, so we have not changed this method.

Binary Threshold

The output of this step is an image where pixels with a high probability of lanes are assigned a max value of 255, and the rest are suppressed to zero. This is achieved using a combination of operations that represent the criteria for a pixel being a lane. We use the fact that the lanes are usually white, part of an edge, and mostly vertical. First, the Sobel operator along x is applied to get the absolute binary threshold. Second, the gradient magnitude is the threshold. Third, the direction of the gradient is evaluated, and finally, color thresholding is applied. The 'Edge' part is detected using (1 or (2 & 3)), and this is masked with the color threshold using (Edge & White Color).

Perspective Transform

The unnecessary part of the image is cropped off, leaving only the ground by applying a Region-of-Interest mask. Then a perspective transform is applied to get a bird's eye view of the lanes to help calculations downstream.

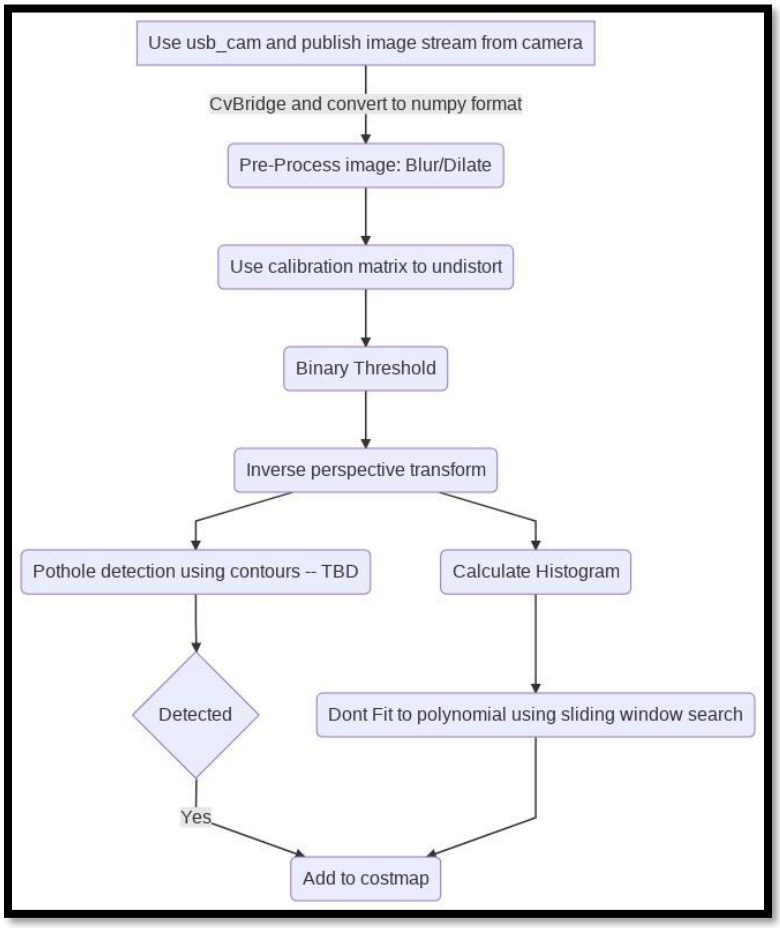

Figure 13: Algorithm for optical lane detection and plotting

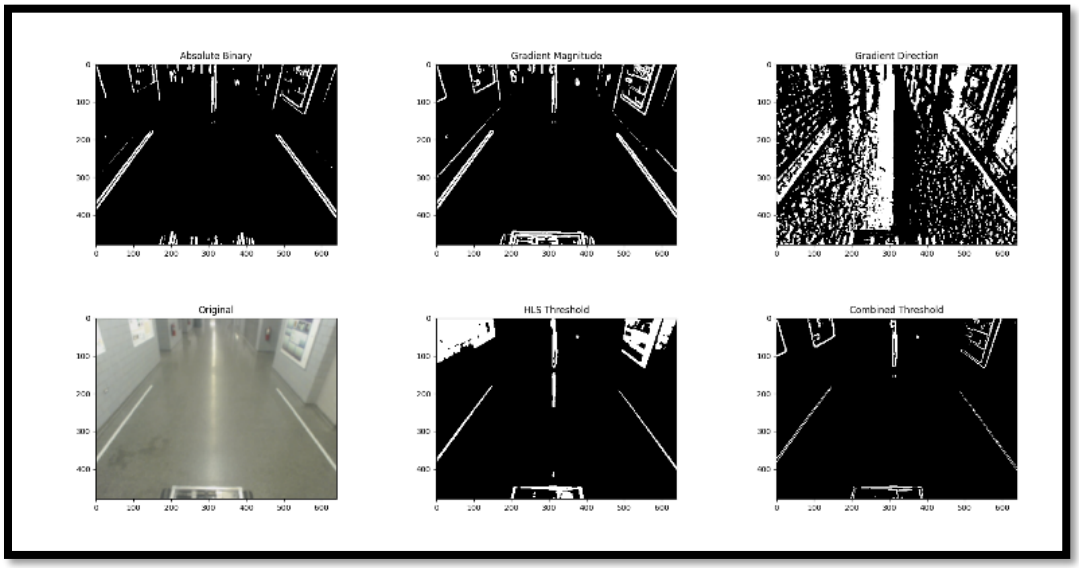

Figure 14: The steps/outputs for the lane detection algorithm

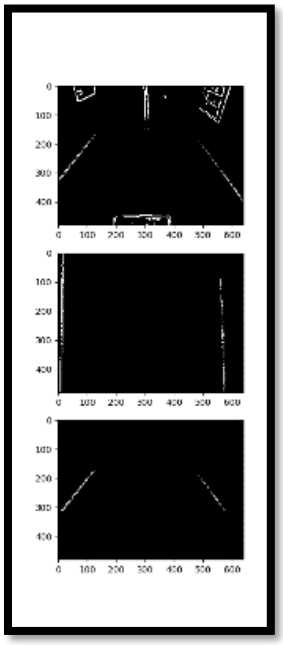

Figure 15: Perspective transform of the lane data.

Lane Fit

The processing leaves an image where the pixels have a high probability of being lanes. We can then find a column-wise histogram and classify the two peaks as left and right lanes. We then slice the image into uniform boxes, find the mean of all the pixels in each box, classify it as a centroid and use that to realign the next sliding window. The location of the histogram peaks is chosen as a starting point for the sliding window search. Doing this easily solves the problem of finding curved lanes and improves upon Hough Transform in both speed and accuracy. The centroids from all the points are fit two quadratic, and the equation of the lane is found. This allows to extrapolate the lanes and find the radius of curvature.

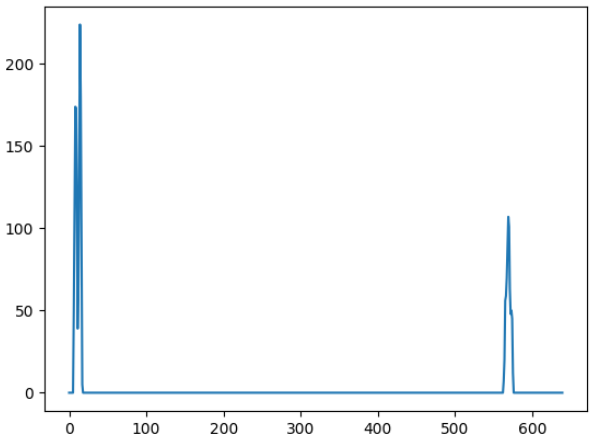

Figure 16: Histogram to determine the lane markers.

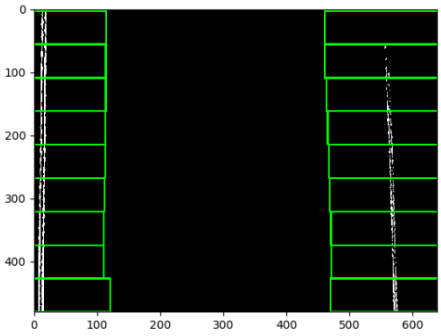

Figure 17: Visualization of the histogram

Visualizing all the steps

Below shows the visualization of all the previous steps. The equations of left and right lanes are calculated, and the vehicle's relative position is calculated along with the radius of curvatures.

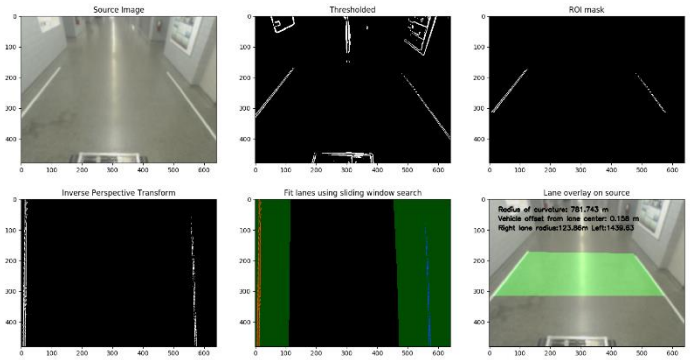

Figure 18: An example of the execution of the lane detection software

Simulation

Through Gazebo we were able to create a Universal Robot Description File (URDF). We included robot dimensions in the URDF and camera modeling to create the simulation model. In the model, each wheel had a subscriber input data which held velocity inputs. This meant that any node could then publish velocity information to each wheel. After completing our model, we were able to use a verification process in which we published the same velocity information to both the actual robot and the simulated robot. From this verification process we deemed that the results were significantly similar and therefore could be used to predict real life conditions to some extent. Obviously, the simulation does not account for variations in conditions such as wheel slippage, collisions, unexpected inclines, etc. In addition, we did not model the PID control system which is integrated in the real robotic system. In conclusion, the system is operational and allows us to do some testing when access to the actual robot is limited. This system could allow us to run code

during the COVID-19 pandemic where meetings and access to the robot were limited. In future years on this team, we will place more emphasis on this useful technology.

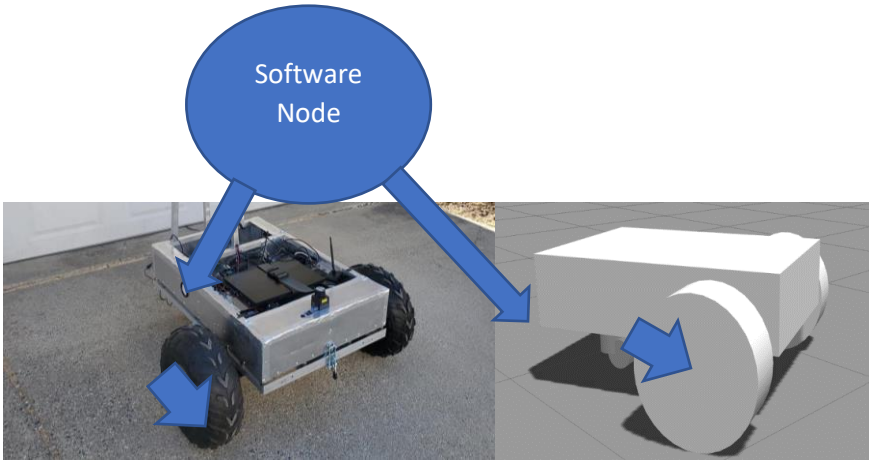

Figure 19: Simulation and actual robot. The Software node sends the same velocity commands to the drive train. The simulation mirrors the actual robot's movements.

Innovations

Invertor installation for laptop battery extension

Running Ubuntu on our on-board laptop was a drain on the laptop battery and allowed us 45 minutes of run-time. To solve this problem, we installed 1 invertor on our auxiliary battery which converted DC 12-volt power to AC 120 that was used to power the laptop. These batteries were easily swappable and allowed us to extend the life of our laptop.

Lane to laser scan message

One of the other innovations involved our lane detection software. We converted the detected lanes to a laser scan message that could then be published to an advanced mapping software. This allowed for filtering of stray data and noise that were the result of the algorithm artifacts.

Key Learning Experiences

The fabrication of Veronica proved to a fantastic learning experience well beyond any classroom experience. We collectively have learned so much as a team about standard practices that are utilized in the field of engineering. In terms of hardware, we learned a lot about depth cameras and the data that it generates. We learned a lot about how to integrate hardware (Arduino) into the ROS environment. This allowed us to keep our software very modular. In terms of software, we learned how to develop the software from the systems perspective and spent a lot of time on design the block diagram of the software. We focused on what packages and nodes that needed to be developed and how all the messages would be subscribed by and published by the various nodes.

There is much more to developing a robot than hardware and software. The most important item is team building and making sure that each team member feels like a contributing member

and is always supported. This in our opinion is of greatest value. We want our team to grow and gain knowledge and skills well-beyond the competition.

We hope to keep on further developing our team, knowledge base and skillsets in our respective areas and come next year with a more innovative and improved version of the robot. In all we are grateful to have had the opportunity to work on such an ambitious undertaking.

Acknowledgements

The creation and funding of this group could not have been possible without the expertise of advisors Dr. Marco Brocanelli, and Dr. Abhilash Pandya. They have contributed an invaluable wealth of knowledge and expertise to our club. We would also like to thank Dean Sondra Auerbach, for helping us facilitate the necessary resources that helped keep the club going. A debt of gratitude is greatly owed to all the faculty and members of Wayne State University's College of Engineering, who have continuously helped to support us in our endeavors. Finally, we would like to thank the dean of the college of engineering, Dr. Farshad Fotouhi for facilitating the approval and support of the organization, without him none of this would be possible.