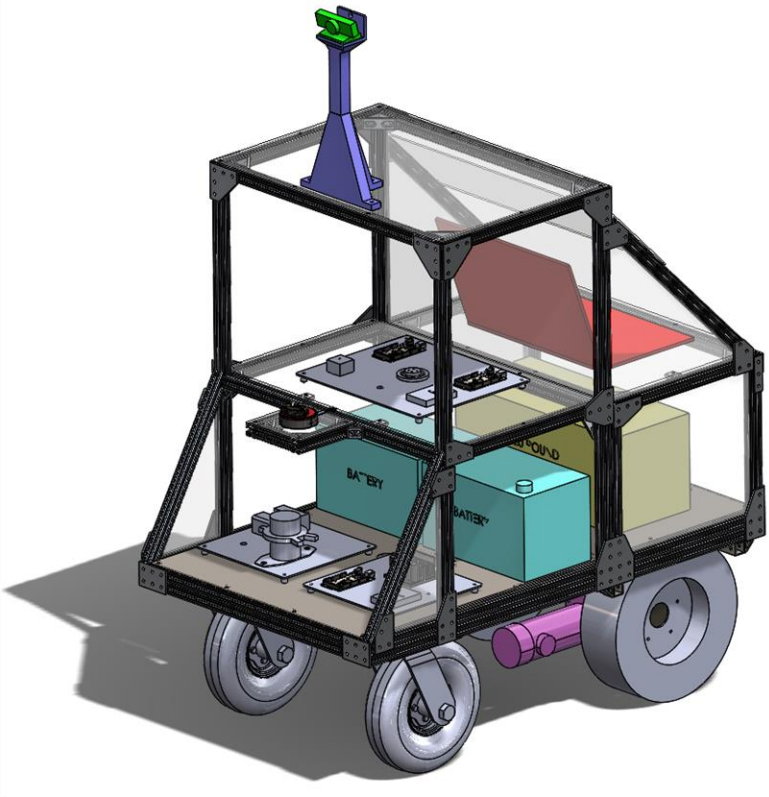
Western Illinois University

The Mystery Machine



Date: 5/25/2019

Peter Stock	Pstock738@gmail.com
Hunter Doyle	Hl_doyle@wiu.edu
Trent Moorehead	Tpmoorehead@gmail.com

Advisor: Dr. Il-Seop Shin

Statement: The time and effort put into this project from these students was equivalent to that of a Senior Design course. The report was composed under my guidance as a professor of Western Illinois University.

Table of Contents

Introduction

The project is an autonomous vehicle that consists of three subsystems; obstacle avoidance, line detection, and waypoint navigation. The vehicle uses LIDAR to avoid obstacles, a GNSS and IMU to determine location and orientation, and a webcam to detect the white lines that the vehicle must not cross. This project is a capstone project for WIU School of Engineering and it will represent WIU in the IGVC in June. The interdisciplinary nature of this project provided all the group members an opportunity to learn from each other in areas such as mechanical design, electrical design, computer programming and system design approaches.

The project team consists of 3 members, Hunter Doyle, Trent Moorehead and Peter Stock, all of which are now recent graduates from the engineering program at Western Illinois University. All testing, construction, and simulating was done while these individuals were in their senior year of college undergraduate school. Hunter Doyle is a mechanical engineering major with an emphasis in design. She has spent the last year working as a technical quality engineer for John Deere. Trent Moorehead is a general engineering major with an emphasis in electrical. He has spent the last 3 years working at an electrical engineering consulting firm specializing in power systems. Peter Stock is a mechanical engineering major with an emphasis in robotics. He has spent the last year working at Vizient, which deals in manufacturing solutions.

Method of Investigation

This project was conducted in a manner that is consistent with the methods learned in our Senior Design class. The group members met on a weekly basis to discuss project ideas and keep track of weekly goals. This project required all members to perform individual research as well as draw on past experiences. The first step was identifying the objectives of the project. The group members each created an individual objective tree and combined them to create a group objective tree. The next step was identifying how to functionally accomplish these objectives. The group decided that accomplishing the objectives would be dependent upon the logic created in the programming. It is for

this reason that the group developed a flowchart that depicted the desired logic of the subsystems. The group then created test robots for the obstacle avoidance and waypoint subsystems. These robots were used to fine tune the logic before being implemented on the full-scale vehicle. The test robots can be seen below.

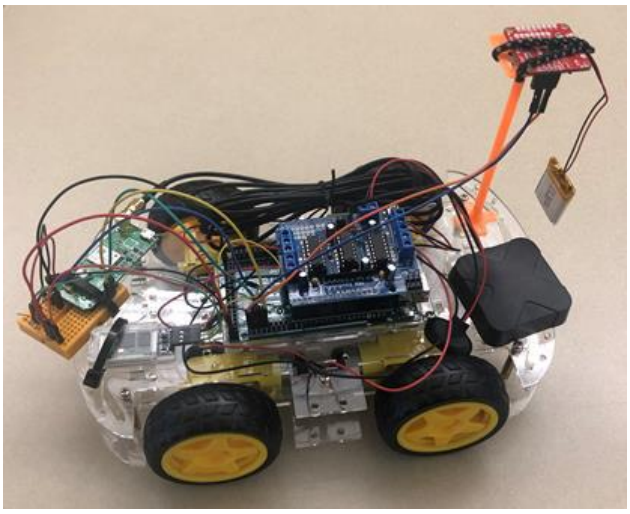


Figure 1 Waypoint Navigation Test Robot

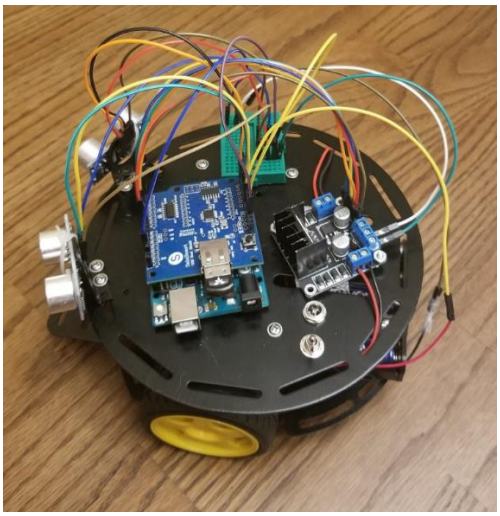


Figure 2 Obstacle Avoidance Test Robot

The final logic of the subsystems and the integration process will be discussed in the sections that follow. The design of the chassis and the selection of the sensors was based on the IGVC requirements.

Mechanical Design

CAD Design

For our chassis, we had three major designs that we integrated together to make our final concept. Our basic layout that we wanted to achieve for our vehicle would be to have our camera span about five feet high with our LIDAR being set at around two feet. There were other components that we placed such as the computer which was on the back of our robot. We also had our batteries and payload of twenty pounds placed as far back as possible so as to apply most our weight on our motors. Lastly, another key item we needed to strategically place was our antenna and that was set on the top of our robot. Below shows a SolidWorks view of what our chassis design ended being.

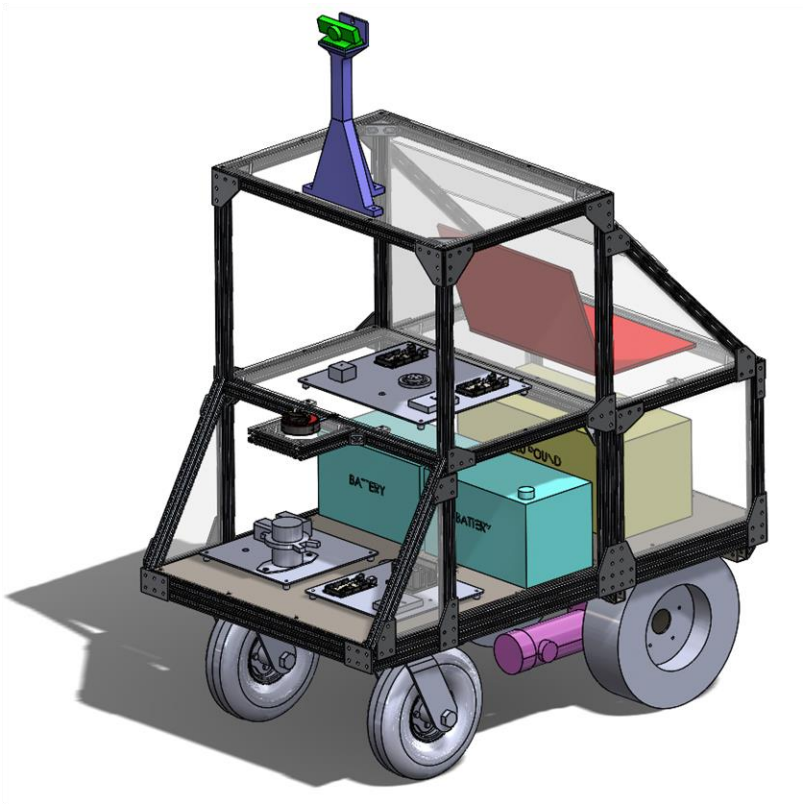


Figure 3 Final Chassis Design

Fabrication

For rough dimensions, we had our chassis be a little more than two feet wide, three and a half feet long, and a little less than five feet tall. These measurements fit in the requirements we have for the competition. Also, another key component to our chassis was figuring out the weight and if it was manageable. We decided on 80/20 T- slotted aluminum due to its rigidity and light weight.

Once the frame was made we started on fabricating and cutting holes in the steel we ordered for our chassis. To help quicken the results of our processes we created and used drawings for our sheets of acrylic and steel. The most complex part of our fabrication process was the motor mounting mechanism. We modeled it after a car and went for simplicity due to time restraints.

After we completed fabrication we worked towards assembling our chassis. There would be times where we would make alterations to our design and therefore need to disassemble our chassis. However, we designed for there to be plenty of mounting area so that no extra pieces of acrylic or steel would need to be purchased. Roughly, the chassis ended up weighing around 200 lbs once we had all our components mounted. The heaviest parts were the two batteries, weighing in at 95 lbs, and the motors which were roughly 45 lbs. We compensated for this weight predicament by buying and using wheels with great traction since we had motors that would be strong enough to push our heavy vehicle through muddy or soggy conditions.

Electrical Design

Wiring Diagram

After the selection of the motors, the design of the electrical system was completed. The wiring diagram is shown below.

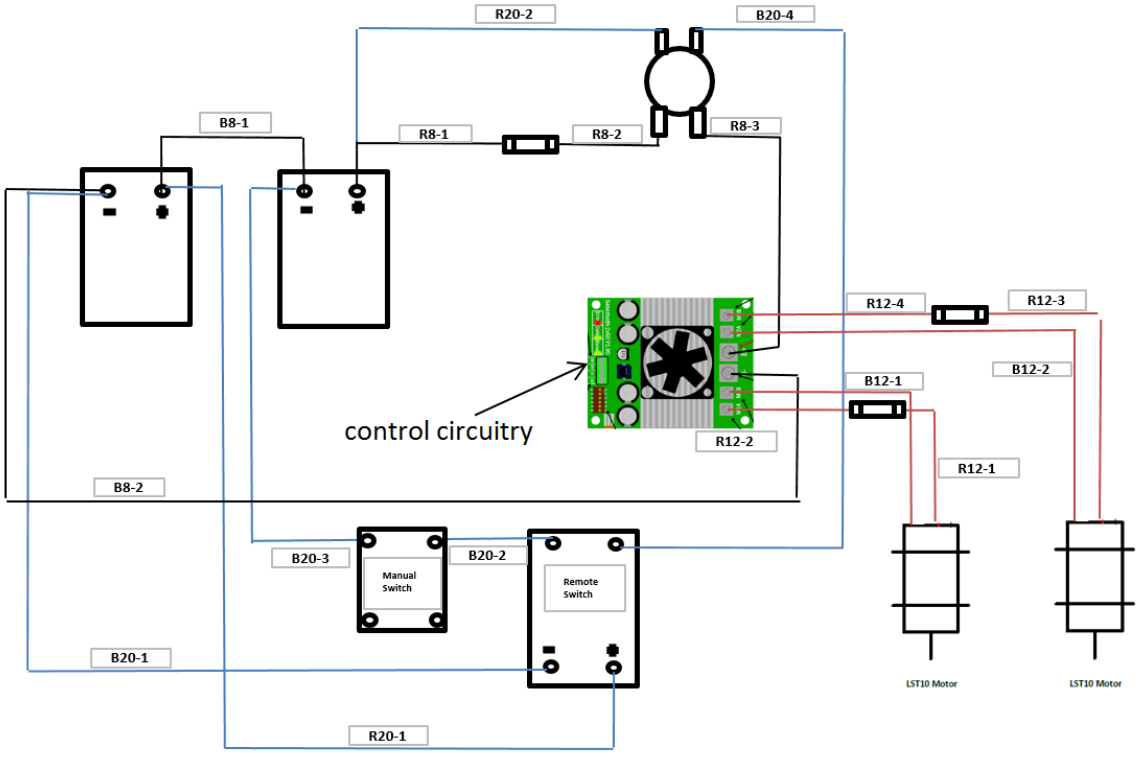


Figure 4 Wiring Diagram

Electrical Safety

Three fuses were used in the circuit to protect against short circuits and motor overloads. These fuses were sized based off of the maximum anticipated current in those portions of the circuit in both steady state and start up conditions. The main function of the fuses is to protect the components of the circuit. For general safety, and to comply with IGVC rules, a manual and remote e-stop were implemented. The manual e-stop and remote e-stop are wired in series with the secondary side of the electromagnetic disconnect switch. The current on the secondary side of the switch flows through a solenoid, creating a magnetic force that closes a contact and allows power to flow on the primary side. If either of the e-stops interrupts the current to the secondary of the disconnect switch, the contact opens and the current to the motors is disrupted. The last safety feature is an LED indicator that is on when the vehicle has power and is blinking when it is in autonomous mode. An Arduino will then send a control signal to the relay or transistor to power on the LED in the required fashion. An addendum to this report will be completed once this system has been implemented.

Software Design

Obstacle Avoidance

The group is currently using a LIDAR for real time obstacle detection and avoidance, but it is recommended that future teams implement a SLAM application. This would allow for mapping of the surroundings that would aide in the decision-making process for multiple runs on an obstacle course.

Before developing the code for obstacle avoidance, the functions of the subsystem had to be finalized. Due to time constraints the final concept chosen was the simple, but effective. The Figure below illustrates the final concept.

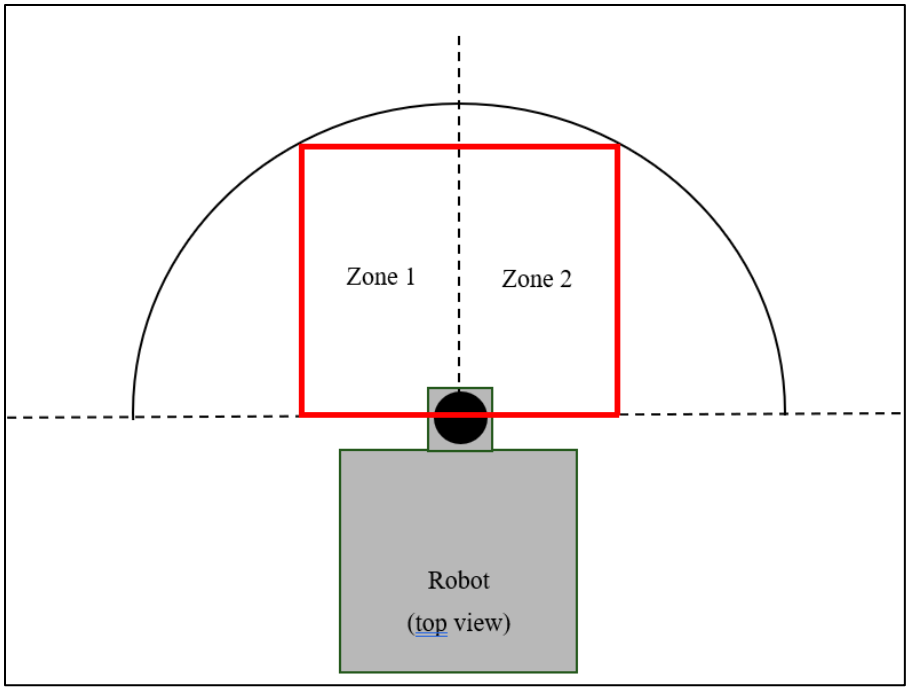


Figure 5 Obstacle Avoidance Final Concept

In this concept all of the data points behind the dotted horizontal line are filtered out as they have no bearing on the forward motion of the vehicle and the vehicle itself would be detected as the nearest object. The LIDAR then looks for the object with the smallest radial distance and defines that as the minimum distance. The code then uses the radial distance of the object and the angle of the object with respect to the horizontal axis to determine if the object is within the frame of the vehicle (Zone 1 or Zone 2). If it is outside the frame of the vehicle, then the vehicle continues on its current path. If the object does fall within the frame of the vehicle then the vehicle will turn left or right depending on which zone it falls in. Once the object is outside the frame of the vehicle the LIDAR will recommend the vehicle go forward again. The actual direction the vehicle will travel is dependent on all three subsystems and the integration code.

Waypoint Navigation

As per the Intelligent Ground Vehicle Competition rules, the auto-navigation course will include a set of four predetermined waypoints. To qualify for the auto-navigation

obstacle course, the vehicle needed to find a single two-meter-wide waypoint. Position accuracy was one of the most important components for autonomous vehicles. A Global Navigation Satellite System (GNSS), an antenna, and an inertial measurement unit (IMU) were needed to achieve autonomous navigation.

The GNSS's raw data was received as NMEA (National Marine Electronics Association) sentences, as shown in Fig. 6. Therefore, the use of the TinyGPS++ library was necessary to parse the data into readable information, presented in Fig. 7.

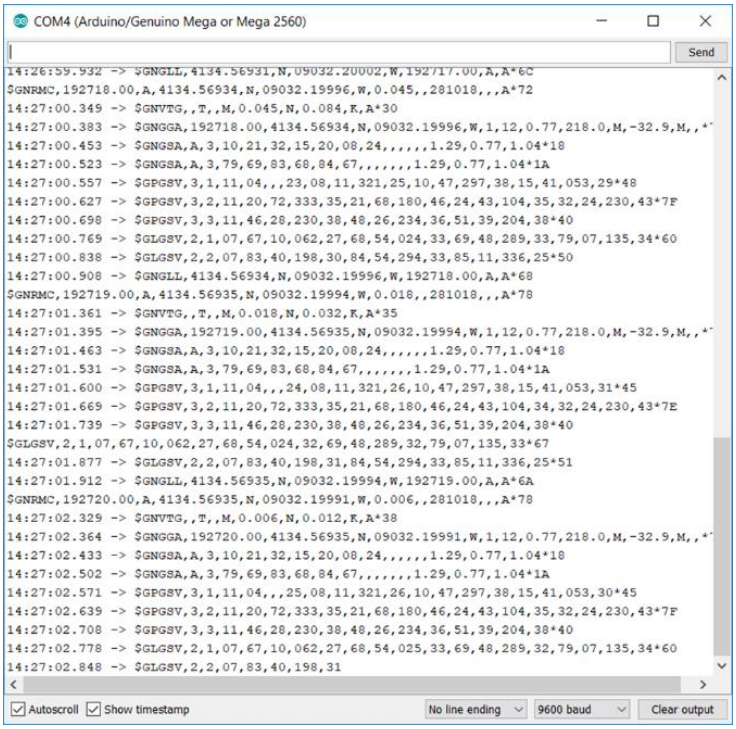


Figure 6 NMEA sentences collected from the GNSS

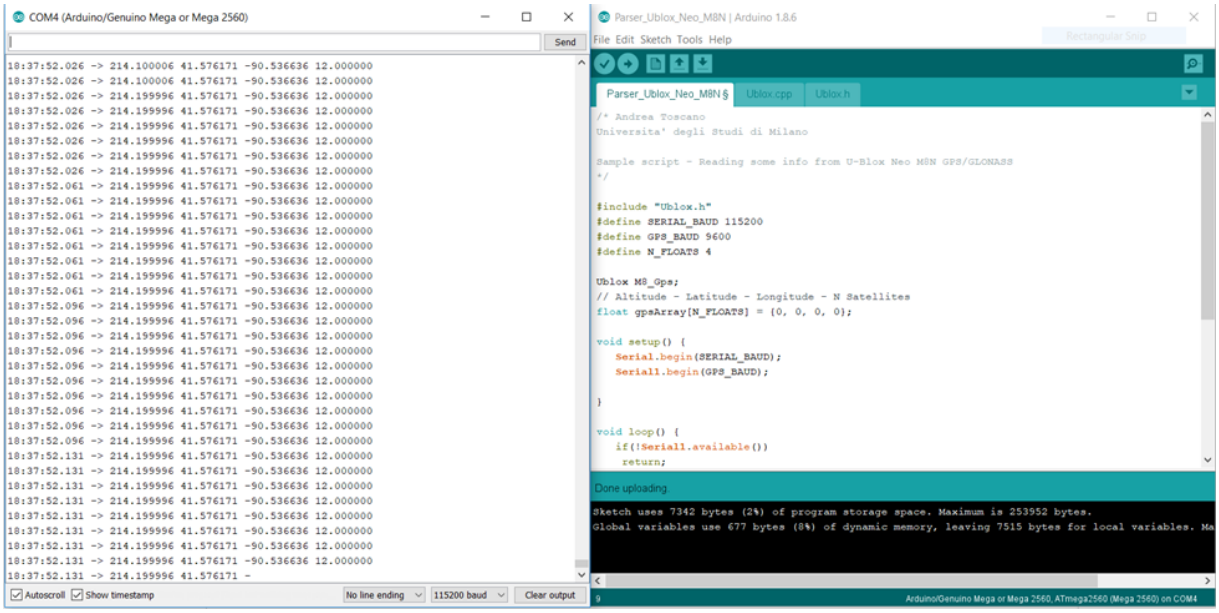


Figure 7 GNSS data collection in decimal degrees of altitude, latitude, longitude, and number of satellites

The measured data from the IMU was used to calculate the heading of the vehicle. For a more robust system, additional information collected from the IMU could be communicated to other sensors. Figure 8 illustrates some of the data that can be collected from the IMU such as roll, pitch, and yaw. The GNSS and IMU would communicate to the Arduino and the Arduino would send the corresponding signals to the motor drivers. Since an Arduino was used for the navigation controls, C would be the main programming language used in this application.

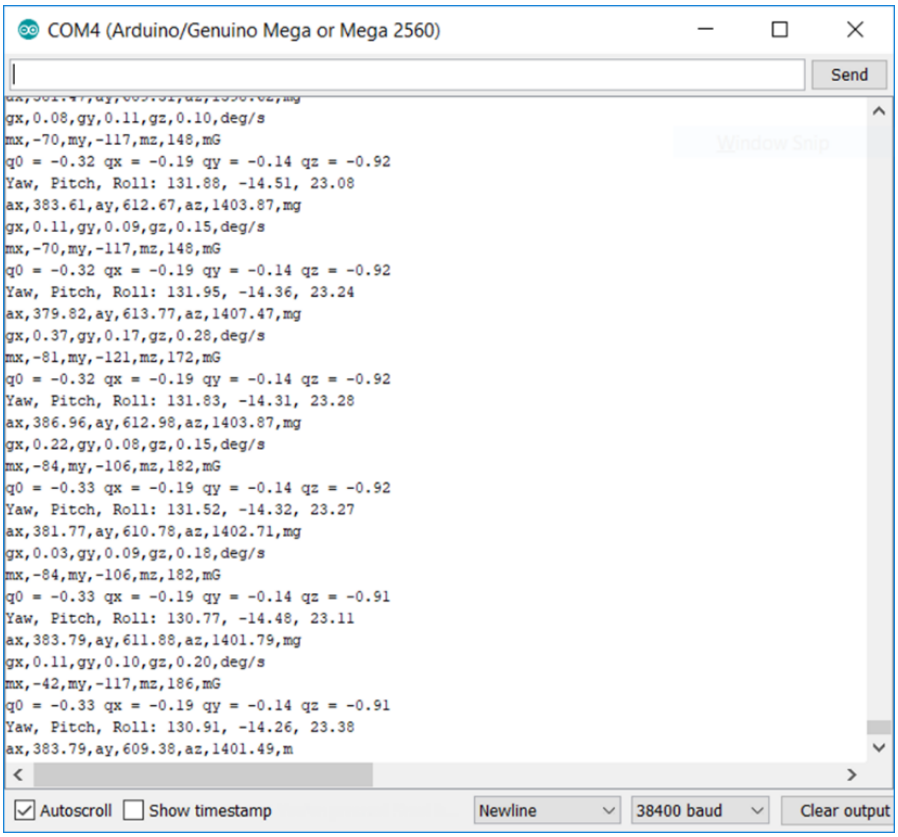


Figure 8 IMU data collection of roll, pitch, and yaw

The waypoint navigation subsystem used the data collected from the GNSS and the IMU. This data determines the location and orientation of the robot. The current location of the robot and the location of the given waypoint were compared to calculate the distance between the two locations. Once the distance between the two locations was zero, the robot had reached the desired waypoint. However, if the waypoint was not yet reached, the robot used the IMU and the GNSS data to determine the desired direction of travel. The IMU calculated the direction the robot was facing from 0 to 360 degrees. North was indicated by a reading of 0 or 360 degrees, East was 90 degrees, South was 180 degrees, and West was 270 degrees. The GNSS was used to calculate the angle between the robot's current location and the desired waypoint location. This angle was compared to the angle measured from the IMU to determine which direction the robot needed to turn, if at all.

Line Detection

To complete the task of line detection we chose to use a webcam. We used the Genius 120-degree ultra-wide-angle camera that would be elevated relatively high on our robot so that we would have the capabilities of seeing and registering both white lines our vehicle would need to stay between. For our programming language we used python since we were able to find a lot of open-sourced code that would help us in making our own. A couple libraries that we would also need to utilize would include, but not be limited to, opencv, rospy, and numpy.

Below is a picture of what we were able to produce with our code in the earlier phases. Notice how the two lines are being detected and the area in the middle is green, showing us the area between the two lines.

Figure 9 Real Life Use of Code

We then needed to have our motors respond to commands being sent from our camera code. We achieved this goal by utilizing previous variables from our commands. Leftx_base and rightx_base were the two primary values we would be looking at since we could find the center between the white lines with respect to the center of the

camera. If at any point we would be offset we would have commands telling our robot to turn and work its way back to the middle of the white lines.

ROS Integration

Robot Operating System (ROS) was used to integrate the camera, LIDAR, and waypoint navigation subsystems. ROS is an open-source, meta-operating system used for creating robots. It provides services such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message passing between processes, and package control. ROS is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex robot behaviors. The diagram shown in Fig. 10 shows the general flow of communication in ROS between each of the subsystems and the motor controller. A node is a ROS term that defines the code that is processed for a particular file. Thus, each of our subsystems have their own nodes that collect and process data. ROS nodes are represented by an oval shape in the diagram. A topic in ROS is like a pipeline for messages to be sent. Any node can either publish or subscribe to topics. Publishing a message on a topic means the node sends data to a topic. Subscribing to a topic means it is reading the data that is being sent to a topic. ROS topics are represented by a rectangle in the diagram.

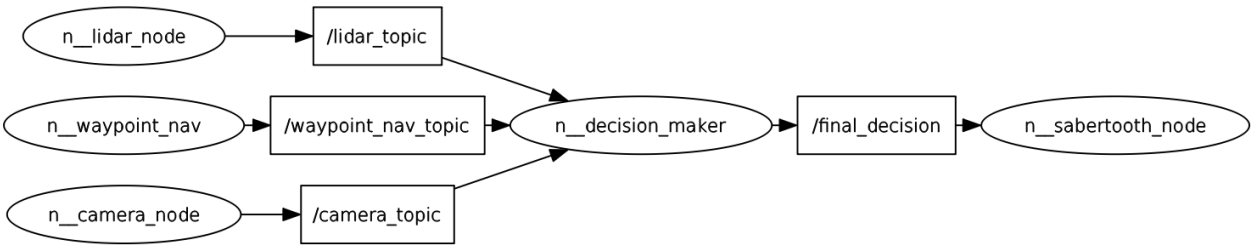


Figure 105 ROS Integration Setup Graph

Therefore, each of the subsystems are publisher nodes because they send messages to their respective topics. The decision maker node is both a publisher and a subscriber because it receives the messages from each of the subsystem topics and sends a message to the motor controller node. It uses the data from each of the subsystems to

determine which direction the robot should go during the competition. Once the decision is made, the decision maker node publishes a message to the motor controller. The motor controller node subscribes to the decision maker topic and sends the appropriate signal to the motors.

Failure Modes/Points and Resolution

Failure mode/point: Fabrication of vehicle wouldn't start until early February of 2019.
Resolution: Prepare drawings for a quick fabrication process while programming each component to work separately in subsystems before integrating onto our large-scale robot.

Failure mode/point: During motor testing we found the rotational speeds to be significantly less then what the spec sheet stated online.
Resolution: We dismantled the motor, checking if any parts were faulty, but in the end had to cut the built-in secondary breaks who's only function was decreasing the operating speeds for our motors.

Failure mode/point: Camera code would quit when too much light was being projected on the lens.
Resolution: This was due to a histogram error we had set up in our code. We simply changed the array of values we were receiving so that a lower valued collection of pixels wouldn't error out our code.

Failure mode/point: First IMU component purchased wouldn't function with an Arduino Mega.
Resolution: Due to their low cost we purchased a different one so we could stick with our Arduino.

Failure mode/point: LIDAR would turn perfectly with an obstacle on its left side, but when on the right it simply kept going straight.

Resolution: We specified more heavily the angles we wanted the LIDAR to turn left and right on.

Simulations Employed

Finite Element Analysis was a key simulation process we followed to ensure our steel plate for our batteries and payload would be strong enough to uphold both parts. We also conducted hand calculations for slope and drive to see how much force and speed we would need to overcome inclines, muddy terrain, and more.

Performance Testing

As stated earlier, subsystems played a key part in early testing of our components and code. They also were helpful due to fabrication issues we ran into along the way. We also did a lot of testing for our motors to see the different speeds we would want to operate at.

Performance Assessments

We are very close to finishing our integration process but have yet to figure out the coding issues we are encountering in ROS. Every component works great on its own and that is something we are proud to have accomplished given the time frame we had for working on this robot.

We have yet to encounter motor or construction issues. Our weight doesn't yet seem to be causing us problems but we have yet to test it on a very muddy day. It drives and operates well on a soggy environment as well.