IGGY

UNITED STATES MILITARY ACADEMY

WEST POINT, NY 10996

The 23rd Intelligent Ground Vehicle Competition

Oakland University, Rochester MI

Cadets John Anderson, Jorge Figueroa-Cecco, Jonathan Gomez, Daniel Li, Timothy Park, Tyler Peterson, Peter Williams

Faculty Advisors: Dr. Peter Hanlon, LTC Christopher Korpela, MAJ Dominic Larkin, COL Robert Sadowski, COL Robert Kewley

I certify that the engineering design present in this vehicle is a significant and satisfies the requirements of our two semester senior design project course in an accredited, undergraduate EE, CS, IT, and SM Curriculum.

Dominic M. Larkin
MAJ, US Army

1. INTRODUCTION

The United States Military Academy at West Point has formed an interdisciplinary team of 7 Cadets from Electrical Engineering, Computer Science, Information Technology, and Systems Engineering majors to participate in the 2015 annual Intelligent Ground Vehicle Competition (IGVC). This year's IGVC team will be representing USMA with the "*IGGY*", a modification of the "Black Knight Rising" model which participated in the 2013 competition. This model uses the same 80/20 aluminum chassis and sensor device hardware with an updated RoboteQ DC motor controller and a Novatel Inertial Navigation System (INS) with both Global Positioning System (GPS) antenna and Inertial Measurement Unit (IMU) devices. The robot operates using a modified image of the Robot Operating System (ROS) from previous years to use given GPS waypoints and input from its sensors in order to successfully navigate the IGVC obstacle course.

1.1. Design Process

All seniors in the CS/EE Capstone Program are required to use the Agile Development Process based on the Scrum Model[1] depicted in Figure 1. The requirements and constraints for this project are identical to those provided by www.igvc.org for the 2015 competition. For the purposes of this project, our clients are the judges for the competition as, our project advisors, and the Capstone Program directors.

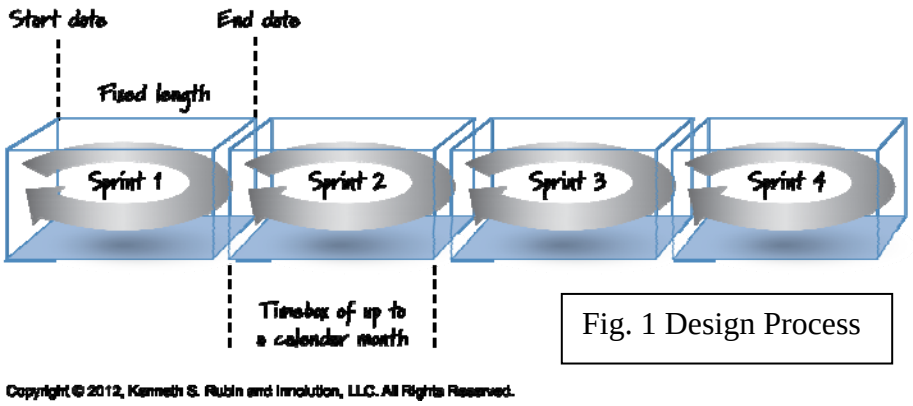


Fig. 1 Design Process

In August of 2015, our team began with the chassis that the 2013 team had designed and built, and had been modified significantly by the 2014 team. This has been the source of many advantages as well as disadvantages. One of the major challenges we dealt with was trying to learn a system that two different teams designed and modified but did not properly document, or write effective "how-to" manuals. Furthermore, besides the lack of "how-to" documentation, there was also a lack of clear understanding which components were completely functioning, partly functioning, or were not functioning at all. Therefore, in some areas, our

[1] Kenneth S. Rubin, "*Essential Scrum: A Practical Guide to the Most Popular Agile Process,*" Innolution, LLC., XX, 20XX

team felt comfortable continuing previous work but in other areas we started from scratch. A significant amount of time was spent determining which areas belonged in each category. As we learned new hardware components and software programs, we verified our understanding by testing the current system. When components did not work as expected, we would often times change the design to a simpler yet effective design. Thus, IGVC has challenged our abilities as adaptive and creative problem solvers.

1.2. Design Overview

Given that AY2014 team's vehicle meets most of the mechanical design requirements for the vehicle (height, width, etc), our team decided to keep the vehicle design and focus on replacing some of the internal hardware components and further refine the software components. Ease of manufacture was important because of limited manpower and a priority on designing hardware and software systems, not the actual vehicle. Flexibility of the design was a priority to allow future modifications. The aluminum frame is covered by a transparent plastic to provide a view of all the internal components and still provide cover of the components from the weather. The plastic box houses most of the sensors, except the sensors mounted on a tower to provide the maximum field of view. From top to bottom, the sensors on the tower are the LiDAR, the Bumblebee camera, and the GPS antennae. The entire vehicle is driven by two wheels and is powered by a battery pack located on the undercarriage of the vehicle.[2]

The software suite of the vehicle relies on ROS, the Robot Operating System. "The ROS is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a variety of robotic platforms."[3] In other words, ROS gives us a software based framework upon which to build our sensors. Once each sensor publishes its data to the ROS, the ROS can feed that data as input to various programs that we wrote to analyze, manipulate, or otherwise process the data into useful information. With that information, we can send commands to ROS for it to control the motors and move the vehicle. It is a way of connecting a variety of sensors, software suites, and motors together. The advantage of using ROS is that it is a common system that others have already tried and

[2] Baker, Stuart, Kyle Yoder, Nicholas Fettinger, Wei-hung Chen, and David Choe. *Black Knight Rising Design Report*. West Point: United States Military Academy, 2014. N. pag. Print.
[3] *ROS*. N.p., n.d. Web. 25 Aug. 2014. <http://www.ros.org/about-ros/>.

tested and as such, there are many pre-built packages and tutorials that we were able to learn from or use in our own project.

The auto-navigation concept relies on continuous map building through Gmapping. Gmapping is a ROS package and provides laser-based SLAM (Simultaneous Localization and Mapping). “Using Gmapping, we can create a 2-D occupancy grid map (like a building floor plan) from laser and pose data collected by a mobile robot.”[4] Laser-based packages are well suited for taking advantage of the LiDAR data and depth images from the Bumblebee camera. SLAM is an algorithm that simultaneously determines where on the map the robot is located and creates that map that the robot is located in. This is a continuous loop where the robot captures data from its environment and matches it to the location data from the GPS and updates the virtual map. Thus, accurate and regular measurements are required to create a coherent map that the robot can use to determine the shortest path to the GPS coordinate.

1.3. Team Organization

The IGGY team is comprised of seven undergraduate senior Cadets from two academic departments. The team lead is Systems Engineer, John Anderson. Responsible for all the hardware design and implementation are three Electrical Engineers, lead by Tyler Peterson and supported by Daniel Li and Timothy Park. Responsible for the software design and development are two computer science majors, Jorge Figueroa-Cecco and Peter Williams. Responsible for the network and wireless capabilities is Jonathan Gomez, an Information Technology major. Each of the Cadets depends on a faculty advisor to provide expert knowledge on various problems that have been solved throughout the year.

2. MECHANICAL SYSTEM

2.1. Base

The wheel base of the vehicle has seen a couple of iterations, both before the time of the current Cadet team. In the first iteration, the vehicle operated on an electrical wheelchair base, with six wheels and the middle pair of wheels driving the vehicle. Last year the system was upgraded to a four-wheel system, with a 2x4

[4] "Gmapping." *ROS*. N.p., 6 Aug. 2014. Web. 25 Aug. 2014. <http://wiki.ros.org/gmapping>.

forward wheel drive. Currently, we are using the same four wheel system as last year. As an improvement, we added wheel encoders to provide the robot information about movement of the vehicle.

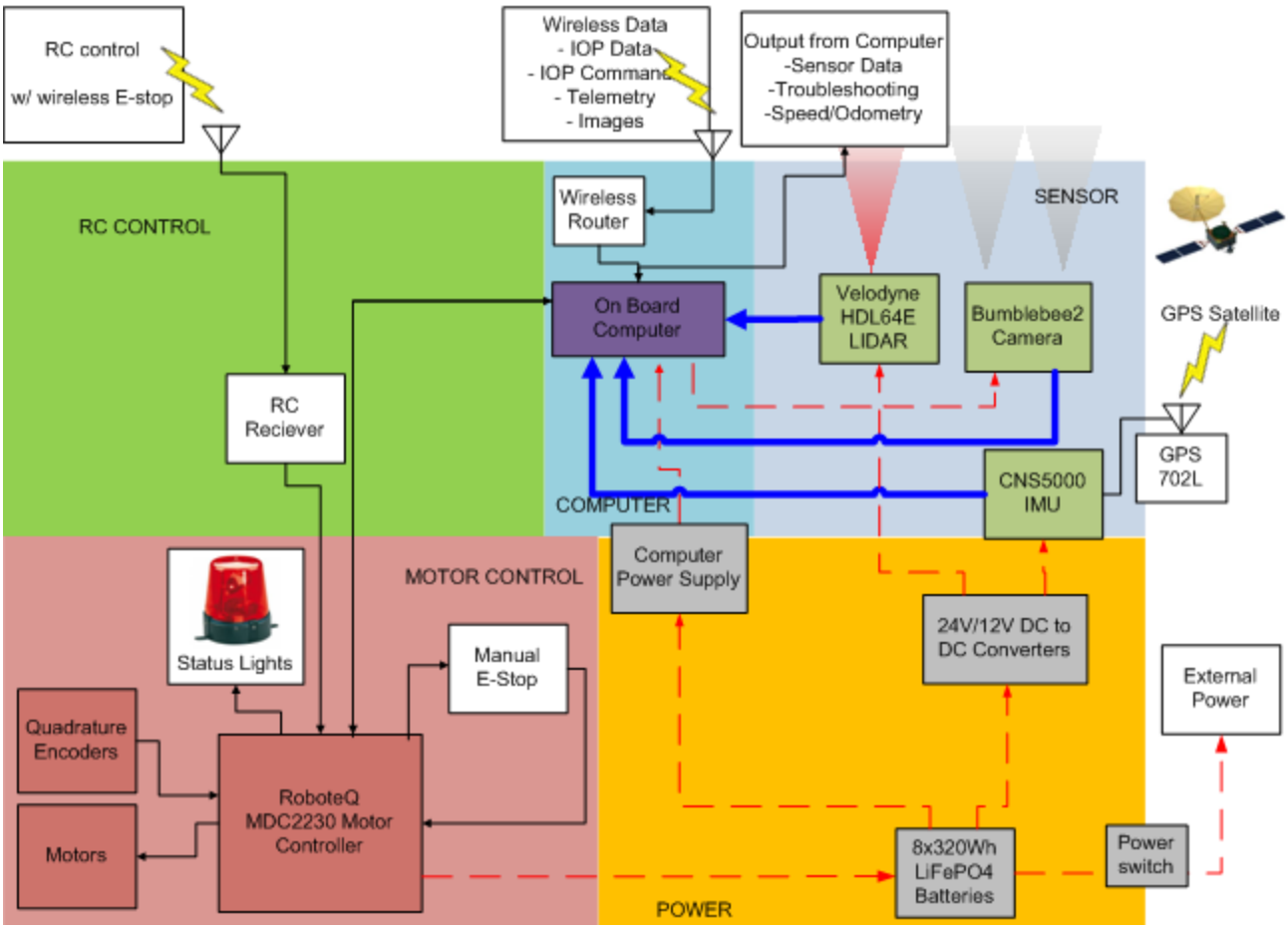
2.2. Chassis

The chassis was designed three years ago by Cadets in the Mechanical Engineering department. The chassis itself has not changed since then, however we have improved on the level of weatherproofing by using rubber tubing to cover any electrical wires protruding from the chassis (such as the cables to the GPS, Bumblebee camera, and LiDAR). This year, we added a LCD monitor mount underneath the waterproof lid. This allows a view of the operating system and we can use a wireless keyboard and mouse to change settings, interface with the robot, or do anything we need to do without physically connecting to the robot.

3. ELECTRICAL/POWER SYSTEM

The robot operates remotely using a series of 320Wh battery pack on the undercarriage of the chassis. The power is routed using DIN rail high power wire, terminals and circuit breakers and each circuit breaker feeds into one component of hardware so that each component can be powered on or off independently from the rest. The circuit breakers also prevent damage to the hardware due to overload or short circuit.

3.1. Hardware Block Diagram:



3.2. Power Bus

The DIN rail is not necessarily new to this year's project. It was used in the past, but was removed last year. We quickly realized that adding or removing hardware devices to the vehicle was more difficult than it should be, and all devices of the robot had to be powered on or off together. This year, we decided to reinstall it to provide some key development and safety features. As a start, the DIN rail's modularity adds to the flexibility. Each hardware component has its own circuit breaker which enabled independent usage of each component. Additionally, the DIN rail allows for safe development of the vehicle. The DIN rail makes it easy to determine which devices are currently drawing power, thus when someone needs to disconnect a device, they can do so without electrocuting themselves or damaging any hardware. This enables our computer science team and electrical engineering team to work on the vehicle in parallel. The computer science majors can power on the motherboard and any necessary devices to develop the software suite, while the electrical engineers power

down anything else they need to work on. Hence, the DIN rail prevented work interruptions and ensured the safety of anyone working on the vehicle.

3.3. Drive System

The vehicle drives on a single differential drive motor controller which receives both software commands from the onboard computer and analog command from the remote controller. This is a new controller from previous years, where each motor was controlled by its own motor controller and breadboard that received analog command from the remote controller and the motherboard. The advantage of today's motor controller is that one controller controls both motors and it can directly interface with ROS. The Roboteq motor controller is supported by a ROS driver which enables the team to receive information from the motor controller directly and also send commands from the motherboard to the controller.

4. Safety Systems

4.1. E-Stop

The chassis has a manual emergency stop red button that meets the safety requirements from IGVC. When the button is pressed, the motor controller is shut off and all power is cut off from the motors. The button must be released before continuing operation of the vehicle.

4.2. Wireless E-Stop

The remote controller also has an e-stop function that remotely disables the vehicle. Since the remote controller is effective to at least 100 meters, it meets the requirements for our wireless E-stop.

4.3. Warning Lights

The LED red light strip is mounted to the mast of the vehicle. When the vehicle is powered on, the LED will turn a solid red. When the vehicle begins navigating in an autonomous mode, the LED will flash red.

5. SENSOR SUITE

5.1. LiDAR

Our Light Detection and Ranging (LiDAR) device is a Velodyne HDL-64E is capable of identifying obstacles in a 360 degree field of vision and converting these instances into point clouds that the computer processes. The LiDAR contains 64 fixed-mounted lasers with a 26.8 Vertical Field of View, 5 to 15 Hz user-

selectable frame rate, less than 2cm distance accuracy, minimum of 0.9 meter range, and maximum 120 meter range for foliage (~0.80 reflectivity).[5] Fundamentally, the LiDAR scans the area around the vehicle and provides a list of distances for each laser point. That data can easily be plotted to a 3D map that a human can view and understand to represent the area around the vehicle. The purpose of the LiDAR is to enable obstacle detection.

5.2. Camera

IGGY uses a Pointgrey Bumblebee2 stereoscopic camera for image capture. The camera is responsible for capturing video of the area directly in front of the robot and sending those images to the operating system for visual processing. The primary purpose of the camera is to enable detection of the white lines on the course so the vehicle can navigate between them. The secondary purpose to enable detection of the red and blue flags, however, this year did not have the time to develop this feature.

5.3. GPS (Global Positioning System) / IMU (Inertial Measurement Unit)

A GPS is a sensor that provides satellite-based navigation, and enables the vehicle to locate itself relative to GPS coordinates. An IMU is a sensor that measures velocity, orientation, and gravitational forces using accelerometers and gyroscopes. Together these sensors provide us with plenty of information regarding the location and movement of the vehicle. The vehicle itself is equipped with a Novatel GPS-702L dual-frequency antenna that “allows users to take advantage of the improved positioning accuracy provided by L-Band services… real-time Differential GPS positioning with meter to decimeter-level accuracy.”[6] The antenna connects to the CNS-5000 INS, which is a combination of a KVH fiber optic gyro-based IMU and Novatel’s OEM6 GNSS precision receiver technology. Together this provides full 100 Hz position, velocity, and attitude sensing, and 0.1 meter horizontal position accuracy.[7] The CNS-5000 connects to the motherboard via USB. Information is then processed by the ROS for navigational use. Our team wrote a custom driver capable of raw IMU data, raw GPS, or Kalman filters of GPS and IMU.

---

[5] "High Definition Lidar HDL-64E." *Velodyne*. N.p., 2014. Web. 25 Aug. 2014. <http://velodynelidar.com/lidar/products/brochure/HDL-64E%20Data%20Sheet.pdf>.
[6] "Antennas GPS-702L." *Novatel*. N.p., 7 Feb. 2014. Web. 25 Aug. 2014. <http://www.novatel.com/assets/Documents/Papers/GPS-702L.pdf>.
[7] "CNS-5000." *KVH*. N.p., 2015. Web. 25 Aug. 2014. <http://www.kvh.com/Military-and-Government/Gyros-and-Inertial-Systems-and-Compasses/Gyros-and-IMUs-and-INS/INS/CNS-5000.aspx>.

6. COMPUTING

6.1. Computing Hardware

The computer is based on a custom-built computer and serves as the nerve center of the vehicle. Previous teams determined that neither laptops nor pre-built computers could provide the features or performance required for this project for a reasonable price. This year we maintained the same computer hardware including: a Bulldozer core AMD FX-8350 CPU running on an ASUS M5A99FX motherboard equipped with 16GB of DDR3 RAM and an EVGA GTX-660 TI GPU. At the time of selection, the CPU provided the greatest multi-threaded performance. The GPU was selected to enable parallel processing of LiDAR and Bumblebee data, and NVIDIA CUDA is highly specialized in that sort of computation.

The computer is connected to a wireless router that enables a remote desktop application to mirror the computer's operation onto an iPad or remote laptop. This allows wireless connection to the computer so that users can make adjustments on the fly or even remove the on-board monitor (to reduce weight and power consumption) and still be able to see what the computer is doing.

6.2. Robot Operating System

The Robot Operating System (ROS) is the computing software for robotic vehicles of choice on this project. As described below, the ROS provides the framework upon which our sensors communicate to the computer and the computer itself can perform any computations needed to enable navigation and obstacle avoidance. ROS was new to all of the members on this year's team, and fortunately, ROS has a very comprehensive tutorial program online that we used to help climb the learning curve. Nevertheless, ROS is very complicated and incredibly capable, as such a difficult tool to use. The ROS was crucial however to developing some of the tools we used, such as Gmapping, White Line Detection, and LiDAR segmentation. Furthermore, an advantage of ROS is that we can record all of the data received by ROS and then use that as a continuous playback to test any changes in software without having to use the actual sensors themselves. This was particularly useful when the computer science majors needed to work on the software but the electrical engineers needed to power down one or more sensors.

6.3. Visual Processing

6.3.1. White Line Detection

The Bumblebee camera provides us with the video needed to detect white lines on the course. The software is responsible for transforming the image to isolate white lines in view of the camera and then use that to inform the auto-navigation to navigate between those lines. The camera captures the image and sends it to ROS. ROS puts the image through a couple of transformations to create the clearest picture of white lines in the image. First, the image is scanned for pixel colors. Any colors that are considered white in color (user-defined setting) are kept in the image, the rest are removed. Then the image goes through thresholding to provide a cleaner looking picture and remove the grainy-ness of the image. The third step is the Canny Edge Detection, which highlights only the white pixels that form an edge of the image, which enables better detection of lines as opposed to anything else on the image that might appear white to the vehicle (such as the sun's reflection off of grass leaves). Lastly, the image goes through Probabilistic Hough Transform which is the final component that cleans up the image and creates a picture that is coherent and appears just as the raw image but only shows the white lines.

Our goal from the start was to produce an image that a human could easily understand to be the raw image with only white lines showing. We successfully completed that task and furthermore enabled easy user interface to calibrate each of the thresholds and settings for every part of the code in real time. When we execute the White Line Detection, two images appear representing the raw image and the processed image. Each image has a set of scroll bars that can adjust the settings individually. The idea is that in some future, the ROS can calibrate itself to be most effective given the lighting conditions (cloudy, sunny, dark, light, outdoors, indoors) to detect white lines. For now, the user will have to calibrate the code upon arrive to a new setting.

6.3.2. Obstacle Detection

The LiDAR provides us with a point cloud that represents the physical world around the vehicle. Our goal is to develop code that can automatically segment the point cloud into two point clouds, "ground" and "not_ground." The "ground" point cloud is self-explanatory; it is all the points in the cloud that represent the ground. To make the system simpler, we assume that anything above the ground is to be considered an obstacle, thus the "not_ground" point cloud represents all of the obstacles around the vehicle. Previously, IGGY used a

code that would perform some analysis of the point cloud to determine which points were obstacles and which were part of the ground. We found that this code did not function properly and we could not troubleshoot it, thus we took a simpler approach. Our code simply segregates points based on a single vertical measurement from the ground, so any points above the threshold are an obstacle; any points below the threshold are part of the ground. This is possible because each point has an x, y, and z measurement. Our code loops through each point.z and makes the determination of which point cloud to copy that point to.

6.4. Navigation

The robot incorporates the variety of sensor data to create a virtual map. This map describes where the vehicle is free to travel, and obstacles are represented by removing pieces of the map where the vehicle may not travel. Thus, the path planning algorithm can only select a path without any obstacles and plans around them. With the map created, the planner forms a line to the GPS coordinate and bends the line wherever the line falls off the map or crosses through an obstacle on the map. As the vehicle moves, the map is updated and expanded as GPS, IMU, and visual data perceive new or changing information.

To create the majority of the map, the LiDAR provides a point cloud which is segmented and the "not_ground" segmented point cloud is passed into the map as where obstacles located. In the future, the White Line Detection feature will be able to place white lines on the map as well, but currently, we cannot do so. Thus, our team can create a map of the usable terrain that can be used by the robot to navigate. This creates a 2D map of locations that the robot can use to navigate. To supplement the LiDAR's data and provide dead-reckoning capabilities, the IMU (inertial measurement unit) provides linear and rotational acceleration information. Finally, the survey grade GPS receiver provides an absolute position of the vehicle and a compass to determine heading, and help determine where the waypoints are on the map. The output of the onboard sensors and their respective software suites are combined via ROS onto the Gmapping package. The Gmapping package provides a visual representation of the world around the robot. The navigation algorithm will use the map along with the current location and known GPS waypoints to determine the most appropriate path while avoiding obstacles on the map.

6.4.1. Goal Setting

A user interface provides the ability for the user to input GPS waypoint coordinates. These coordinates are placed in a navigation stack that the robot uses to do path planning.

6.4.2. Simultaneous Localization and Mapping (SLAM).

The SLAM allows the LiDAR and the White Line Detection algorithm to populate a 3D representation of the environment around the vehicle. Each time the vehicle moves, sensor data from the IMU and GPS update the map so that obstacles will move on the 3D map relative to the robot's position just as they do in the real environment.

6.4.3. Global Path Planning

After the 3D map is created, the robot determines the shortest path between the current location to the first GPS waypoint on the navigation stack. Then commands are sent to the motor controller to move the vehicle. The path planning algorithm has two aspects. Global Path Planning shows the start point to end point, akin to a map with the major highways and roads. The Local Path Planning looks at the surroundings and adapts, akin to a driver driving on the road and passing other cars and staying in his lane.

7. PERFORMANCE PARAMETERS

No performance tests were made to determine the current performance parameters of the vehicle. Therefore, we've included previous year's test to help create a benchmark for this year's development.

Performance Parameter	Units	Predicted	Tested
Top Speed	mph	5	7.3
Max Incline	°	7	30
Reaction Time	s	0.05	0.05-1.00
Battery Life	hr	1.75	2.5
Obstacle Detection Dist	m	120	100
Waypoint Accuracy	cm	20	38

8. PERFORMANCE ANALYSIS

The current robot can perform LiDAR ground segmentation to separate the point cloud into two, only keeping the point cloud that represents obstacles on the map. The GPS and IMU can provide sensor data when in plain view of the sky. The Bumblebee camera can perform White Line Detection. At this point, we have

successfully, put the point cloud into the SLAM and need to test it with the GPS and IMU. The White Line Detection has not been put on the map, it may be necessary to create a depth image first before putting that data into the map. Lastly, the navigation stack needs to be connected to the map.

9. SUMMARY

All in all, the team this year has learned through significant trial and error how to use the vehicle built in previous years. Some aspects of the vehicle had to be replaced or fixed in order to progress the working state of the vehicle. The motor controllers were removed and replaced with a single controller that interfaces with ROS. The power bus was removed and replaced with a DIN rail. The ROS version was updated to the latest version. The Kubuntu OS was also updated to the latest release.

Our experience on this project is significant to each Cadet in their own way. Not a single aspect of this project had been taught to any of the Cadets prior to this work, so the learning curve was very steep, yet rewarding. Being able to learn something completely new and apply it in concert to the rest of the project provided each Cadet with a real experience of what we might experience in some future project.

Furthermore, our contributions to this project mean that in the future when we join a more advanced and complicated project such as a self-navigating vehicle that research teams might work on, we can understand the fundamentals of what is required for auto-navigation as well as combining different discipline strengths together.