IGVC 2015-2016

Metaknight

University of Central Florida
Robotics Club at UCF

Team Lead
Therese Salas
tsalas@knights.ucf.edu

Kevin Almeida
kalme003@knights.ucf.edu

Ryker Chute
chuter@knights.ucf.edu

Nathan Duenas
nduen007@knights.ucf.edu

Christine Erwin
ceerwin@knights.ucf.edu

Jonathon Gambrell
jonathangambrell@gmail.com

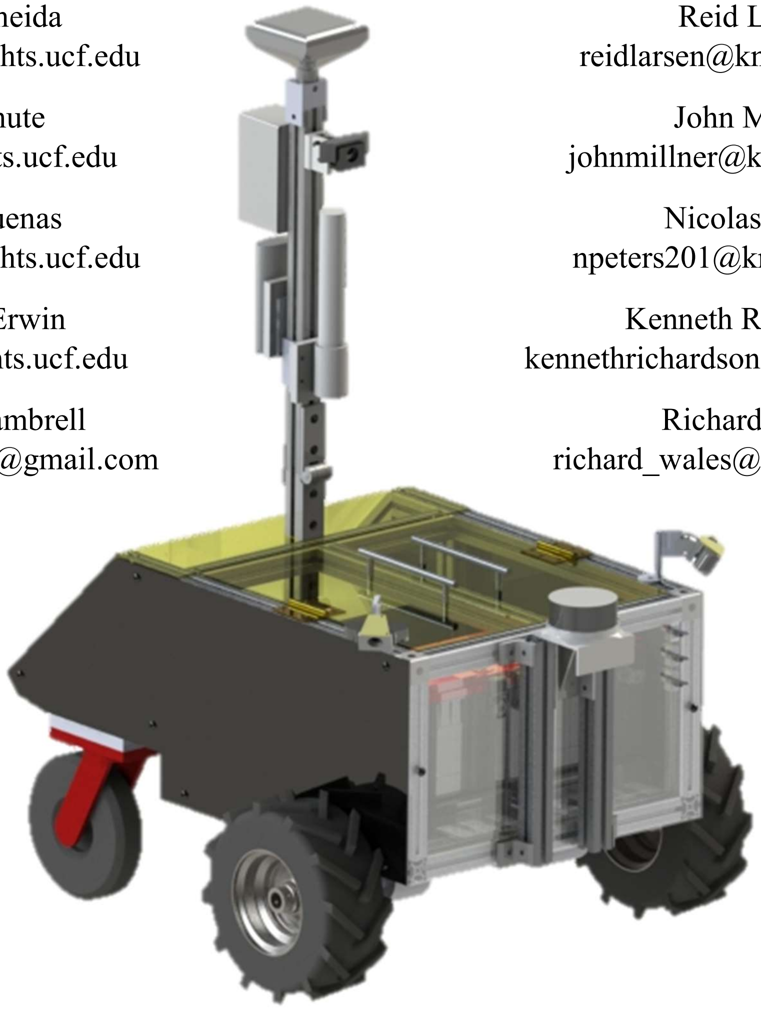
Reid Larsen
reidlarsen@knights.ucf.edu

John Millner
johnmillner@knights.ucf.edu

Nicolas Peters
npeters201@knights.ucf.edu

Kenneth Richardson
kennethrichardson@knights.ucf.edu

Richard Wales
richard_wales@knights.ucf.edu



Faculty Statement: I certify that the vehicle has been significantly modified for this year's competition.

x Crystal Maraj

Crystal Maraj

5/15/2016

Table of Contents

1 Team Structure and Design Process

1.1 Introduction

Metaknight is the autonomous ground vehicle entry from the Robotics Club at the University of Central Florida (UCF). This vehicle is programmed to move from one GPS waypoint to another while avoiding various obstacles. The Robotics Club is a registered student organization at UCF comprising of undergraduate to graduate students in interdisciplinary fields. The club works on various projects and competitions which give engineering students hands-on experience and to increase the understanding of real-world engineering applications and robotics. Intelligent Ground Vehicle Competition (IGVC) is one project which a group of students were assembled to include mechanical engineers, electrical engineers, and programmers. This team worked diligently together to build Metaknight for the IGVC competition.

1.2 Team Structure and Design

1.2.1 Organization

The Robotics Club at UCF's IGVC project comprises of ten UCF student members. These members are divided by their respective discipline, with individual mechanical, electrical, and programming teams. Within these teams, each member is given a particular task with a set deadline. Once a week, the entire IGVC team meets to discuss the progress made on those tasks. The team also communicates via Telegram (i.e., a mobile instant-messaging application, for immediate contact to ask any questions). The IGVC project timeline started with weekly meetings from September 4th, 2015 and will continue until June 10th, 2016.

1.2.2 Task Management

The teams of mechanical engineers, electrical engineers, and programmers have their own unofficial leads that receive tasking from the project lead whom delegate those among their group. These members include: Mr. John Millner leading the electrical team, Mr. Nicolas Peters leading the programmers, and Ms. Therese Salas acting as both mechanical and project lead. As aforementioned, each task assigned to members has a deadline (e.g., tasks such as creating a Computer Aided Design (CAD) model or providing a schematic of power distribution). This promotes efficient and distributed work to complete the vehicle in a time-effective manner.

1.2.3 Design Process and Constraints

The planning phase of this project involved the full team meeting and brainstorming new designs for the ground vehicle platform to compete this year. As a result of the planning phase, integral parts were priced and quoted to create a budget. Within the budget, specific details were highlighted and the programmers formulated their plan for obstacle avoidance and lane detection. With this information, the mechanical team started the design for sensor placement and chassis upgrades. Next, the mechanical team created the CAD for the chassis and sensor mounts while the electrical team planned the schematics for power distribution and communication. During this design stage, the programmers had a small platform to test the formulated code. When the mechanical designs were finished, appropriate parts were ordered, machined, or 3D printed. The electrical team then started working on wiring the ground vehicle and the programming team imported their code to begin real-world testing.

1.3 Effective Innovations

From a mechanical standpoint, innovative designs and techniques were made to create the most modular and functional vehicle. Differences in last year's design and this year's include: the reversing of the drive configuration, the mast relocation, the flattening of the rear hood, and the mounting of the motors. Additionally, the vehicle was reversed so that the rear of the vehicle is now the front. The mast was reduced to a single 1.5" beam and moved to the middle of the vehicle. A hinge was also added to the mast and secured with a barrel bolt for better mobility of the vehicle itself. With the mast relocated, the center of mass shifted, so the motors were moved back to be closer to the center of mass. The mounting of the motors changed by adding a bearing for support between the wheel and the motor. The bearing takes on any torque the axle of the motor would undergo during turns. The rear of the chassis was redesigned by flattening the original design of two angles for incremental improvement.

Other innovations include the battery power distribution and slightly modifying the coding from last year's IGVC competition. The batteries were connected in series, providing a full 24 Volts. This year, the batteries are connected in parallel for safety and effortless charging. The software used to have problems with the thresholding due to last year's obstacle course that included small white flowers the vehicle would classify as part of the white lane lines. The improvement for this year involved the cameras repositioned to improve the distinction of the flowers from the lines. Upgrades to the coding will allow for a more direct path between waypoints.

2 Mechanical Design

2.0 Mechanical Design Overview

Metaknight was designed mechanically around predetermined sensors. A significant portion of framing of the vehicle was built in 2014, however for this year's competition some changes were made to streamline the chassis. Additionally, the drive configuration was changed so that the abdomen of the vehicle is considered the front and the angled "hood" of the vehicle is considered the rear. This change allows the drive wheels to navigate smoothly with casters to support the rest of the vehicle. The mast was placed in the center for sensor mounting and to allow room for the placement of the Velodyne LiDAR at the front of the vehicle to have a full view of its surroundings.

2.1 Chassis Design

Metaknight's frame is contrived largely of 15 series (1.5 in^2) T-slotted anodized aluminum for its adaptive capabilities. Figure 1 illustrates the frame design which consists of three central sections: front, rear, and mast. The front is a flat-faced design with a 3D printed Velodyne sensor mount elevated for maximum sensor efficiency in its autonomous capabilities.

The Velodyne is held in place by two 80/20 beams protruding from the front face. With this short flat design, there is plenty of room for batteries, electronics, and the camera atop the sensor mast to have a clear view of the ground around the vehicle. Underneath the vehicle are the motors mounted with custom F-shaped aluminum blocks and the axles which are supported by large bearings to withstand the torque of the payload. The rear of the autonomous vehicle is fashioned like the hood of a car, and holds the computer, IMU, Velodyne board, and Arduino. The sensor mast is made of the same 80/20 framing used for the chassis. The mast is cut in half and a hinge and pin was added to allow the mast to be folded down for increased portability when traveling.

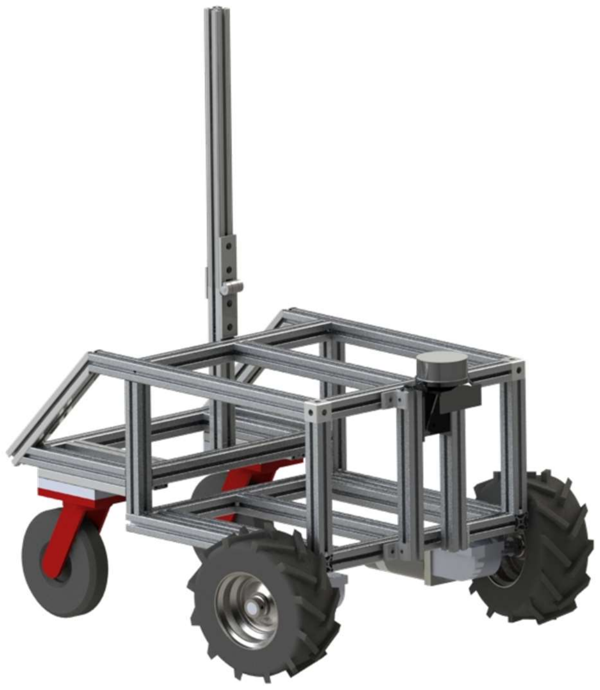


Figure 1

2.2 Sensor Placement

The sensors used on Metaknight include a Velodyne 3D LiDAR, Logitech cameras, motor encoders, GPS receiver, and Sparton IMU. Due to its full Field-Of-View (FOV), the Velodyne was given the most visual room, sitting on the front of the vehicle. Its placement was chosen for its 360° around, 18° fan for up to 100 feet capabilities. Three cameras were strategically positioned for the best view of lane lines. Specifically, two cameras were placed on either side of the chassis towards the front of the vehicle to get a better view closer to the ground. The third camera is located on the mast, looking down to the front of the vehicle for a full view of the driving area. The motor encoders are inlaid in the motors, and are wired through the bottom panel into the front of the vehicle. The mast was built to hold all external sensors and the E-stop. Its placement was strategically moved to the center of the vehicle to allow the Velodyne the most range and maintain the center of balance of the vehicle. Externally mounting of the GPS receiver to the mast is permitted because it is weatherproof. Finally, the Sparton IMU is mounted in the rear, centered to the vehicle. This placement keeps it from enduring any kind of impact that might shift since the placement of the IMU needs to remain constant for its readings.

2.3 Mechanical Failure Points

Metaknight is designed to be safe and durable and can withstand common weather conditions and terrain while able to avoid obstacles. The chassis is made of 80/20 anodized aluminum and the components for the motors are machined steel. Simple parts used for sensor mounting were made with 3D printed ABS material on high to medium density. For this year's competition, the emergency stop was placed on the upper back of the mast for easy access and improved visibility. The mast was set to the middle of the vehicle and reduced in size to keep the vehicle compact and efficient. The sensors attached to the mast and located externally on the vehicle was a consequence of excess amounts of rain during the 2015 IGVC competition. As a result, Metaknight was equipped with weather proofing material. Rubber weather stripping was lined in the openings of the acrylic panels and around the doors on the top. Originally the strips were intended as an extra layer between the panels and frame of the vehicle, however they were too thick. The panels themselves are made of acrylic, allowing full coverage of the chassis and internal components while still providing protection for low impact force. The placement of the paneling reduces high impact forces on it, due to the low driving speed and type of obstacles in the competition. Overall, the vehicle is designed to be safe, weather-resistant, and durable enough to last at least another year.

3 Electrical Design

3.0 Electrical Design Introduction

Metaknight contains a simplified and federalized system of circuits for assembly (see Figure 2), and designed to allow for quick and easy troubleshooting. There are three "centers," "Auxiliary," "Motors," and "Logic." The goal of this design allows for easily localization when a problem is presented, which can be identified, replaced or repaired.

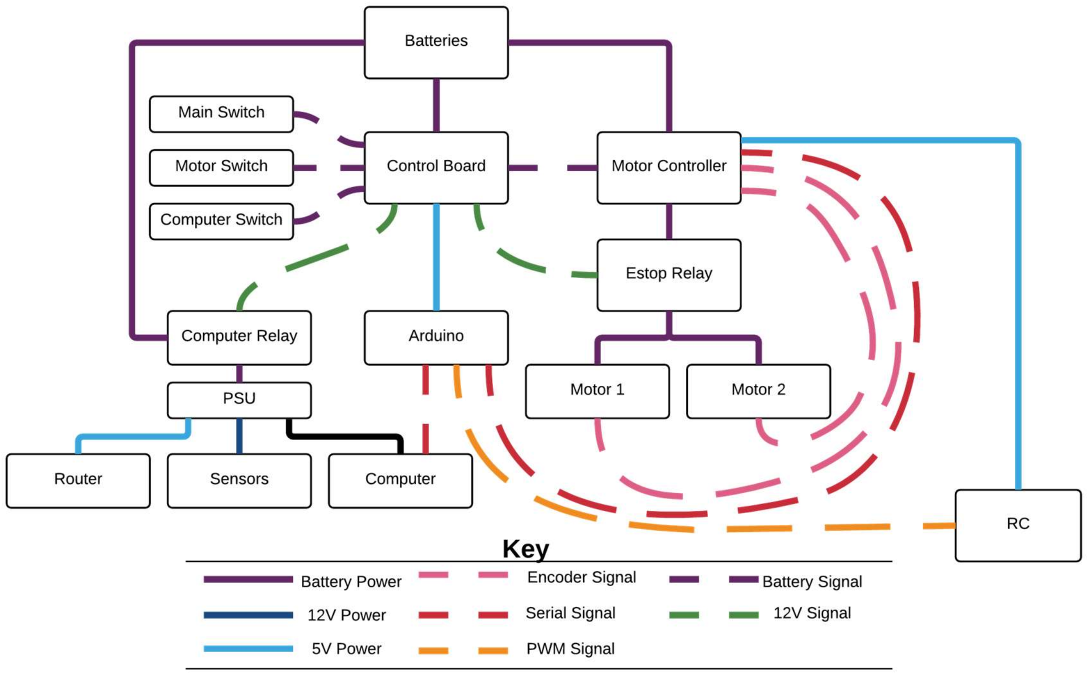


Figure 2

All parts are crimped using 3M Faston connector types to simplify construction, and make replacement easier. 3M Faston crimps were chosen because of their high availability at consumer stores, high durability and vibration resistance, and ability to cleanly connect and disconnect.

3.1 Sensors

The following sensors are used to maximize autonomous performance in the vehicle:

- The Velodyne VLP-16 is a low-power consumption device that uses LiDAR with 360-degree FOV and detects in 3D with a range of up to 100 m.
- Three Logitech c930e web cameras are implemented: one on top of the mast and two on either side of the front of Metaknight. The shallower angle of the two cameras prevents glare from the sun, which is an improvement from our previous design which were located on both sides of the mast.
- Quadrature optical encoders precisely captures motor shaft movement.
- The Hemisphere MiniMax dGPS is weatherproof, fit for outside use. Using its MGL-3 antenna, the receiver applies differential GPS correction signals to the input GPS signals at a rate of 5 Hz.
- The Sparton Altitude Heading and Reference System 8 is fixed on the back of Metaknight. Its strategic location far from the motors minimizes the magnetic interference.

3.2 Electronic Design

3.2.1 Communications

The communication protocols and hierarchy between all devices were chosen so that the computer has the ability to control and receive diagnostic information from as many devices and subcomponents as possible (see Figure 3). All devices are capable of speaking to the main computer either directly through USB/Serial interfaces and Ethernet or through an Arduino (which communicates with the main computer by USB) with the notable exception of the mechanical Emergency Stop switch and power switches (i.e., Main Power, Motor Power, and Computer Power) which are kept completely separate from the system for safety and reliability.

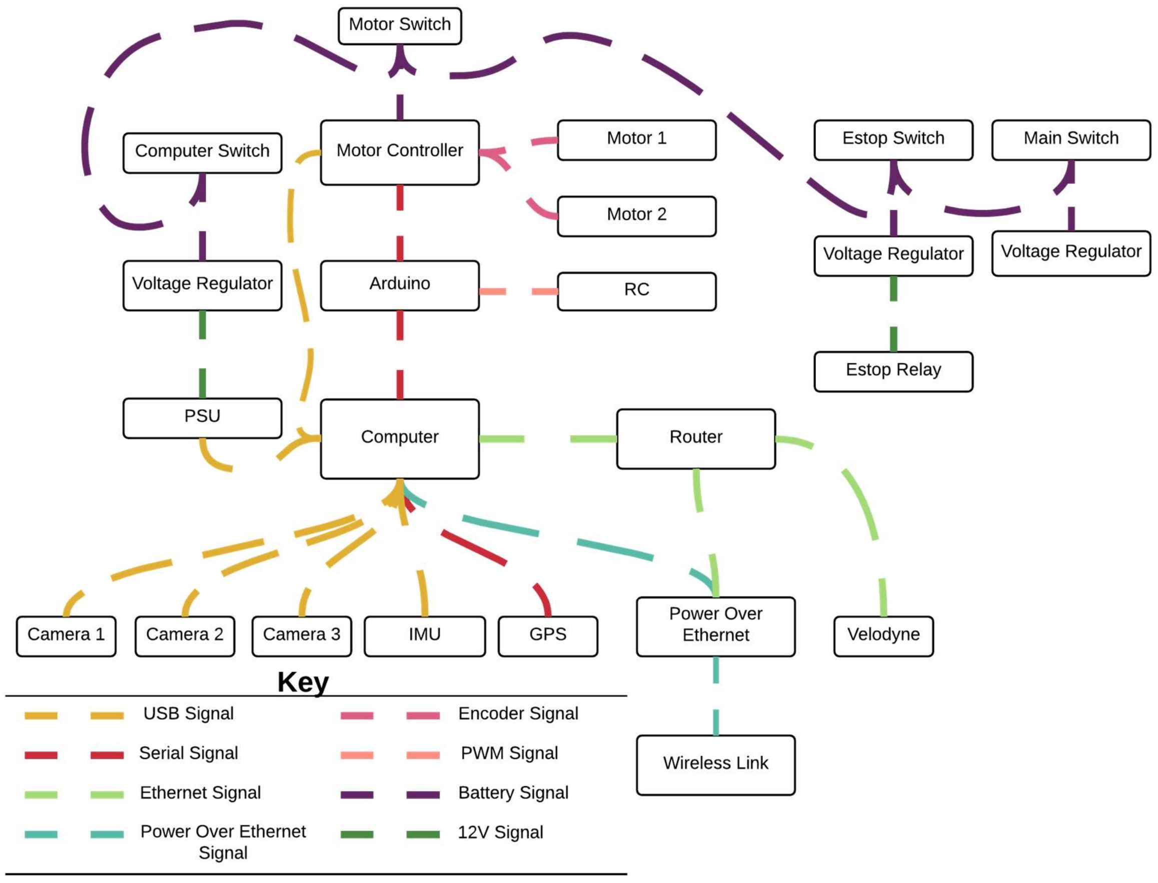


Figure 3

Metaknight has the ability to communicate with a debug cart over airMax protocol using an Ubiquiti Rocket M5 airMax receiver/transceiver. airMax was chosen over WiFi because of its stability outside, long range, and low probability of other teams saturating the bandwidth in competition environments. The debug cart has the ability to receive data and provides a first person view of what the system is seeing, interpreting, and reacting to thus allowing for access and fine tuning the program as needed.

Metaknight also contains a radio control receiver and controller for manually driving Metaknight in training/practicing mode to retrieve test data, recovering Metaknight if lost, and generally for portability in moving Metaknight around when off course. The radio controller has the ability to initiate an emergency stop command to the motor controller which will immediately stop the motors

3.2.2 Power Distribution and Usage

Metaknight functions on two 12V Deep Cycle AGM 55 Amp Hour Batteries in parallel. Placing the batteries in parallel is a change from last year design when they were in series (see Figure 4). This change allowed the following:

- safer system with reduce damage from shorts and chance of shocks.
- increase efficiency when regulating voltage.
- reduce the chance of brownouts in the computer when a large load is applied to the batteries from the motors.
- increase longevity of batteries.

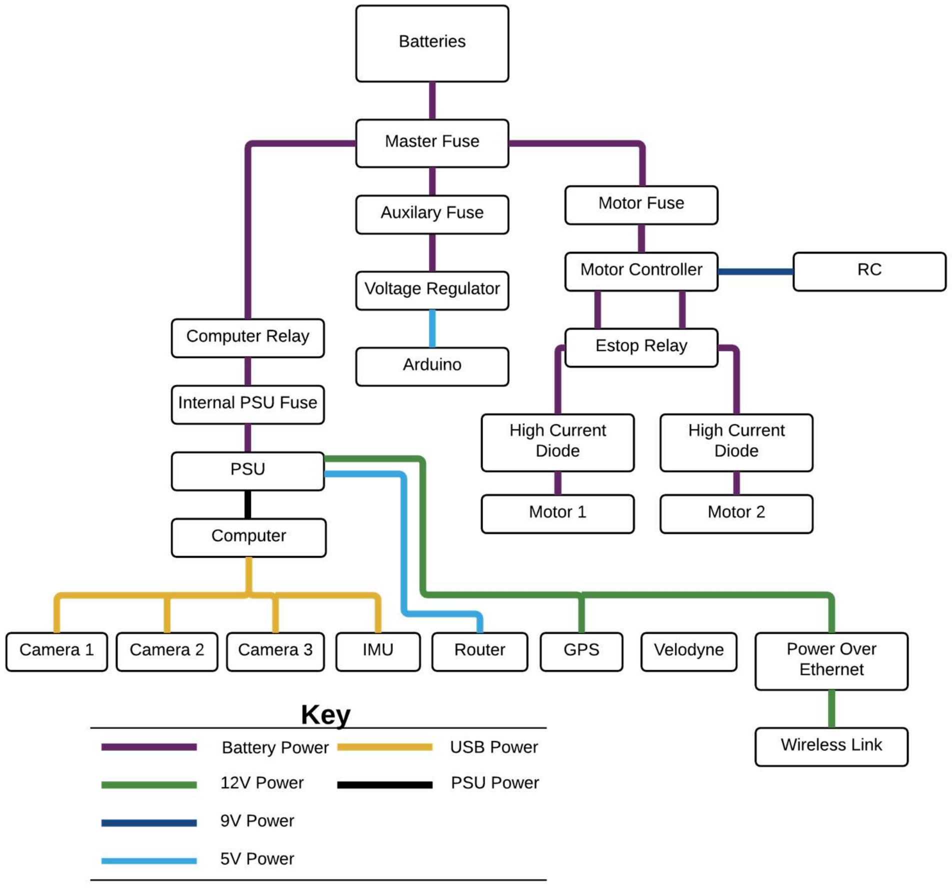


Figure 4

All systems are fused conservatively in wiring and components. This reduces the chance of a crimp failure, crash, part failure, or human error. Each section of the vehicle is fused including Motors (Motors, Motor Controllers), Logic (PSU, Computer, Sensors), and Auxiliary (Arduino, control board). The Motor Controller (Roboteq MDC2230) is capable of limiting current to each motor to prevent sending too much current through the system in a non-failure mode.

For the computer, the team used the MiniBox M4-ATX, 250w output PSU for the computer, and uses the 12V and 5V buses to regulate power to the router, GPS, Velodyne, and the ubiquiti antenna. This device was selected for its ease of use, heavy reliability, and USB diagnostics data such as battery voltage and power consumption.

3.2.3 Control board

The Control Board for Metaknight (see Figure 5) is a simple custom Printed Circuit Board designed to manage the control circuitry which reduce clutter and creates a centralized point for the switch logic to meet and drive the relays and power-on lines.

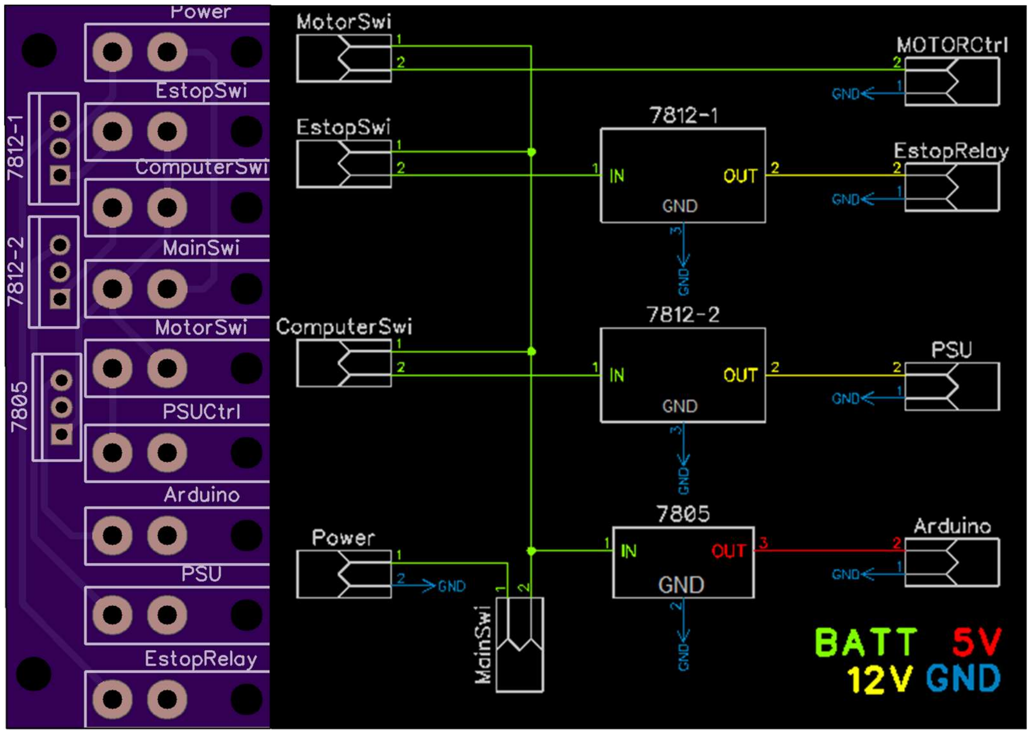


Figure 5

3.2.4 Emergency stops

Metaknight contains two emergency stop functions that can be used to safely disable the vehicle in an emergency (see Figure 6). There is a high visibility emergency stop switch on the center back of the robot approximately less than 4 feet from the ground. This "hard" emergency stop is linked directly to a high power relay on each of the motors, so that when pressed, power from the motors is immediately cut off, and any residual power is immediately drained to ground. This hardware level emergency stop does not interfere with the computer or sensor power. The second emergency stop is located on the RC controller and is connected directly to the motor controller where it can toggle a specialized emergency stop flag inside the controller's firmware that cuts off input power and drains any remaining power in the motors.

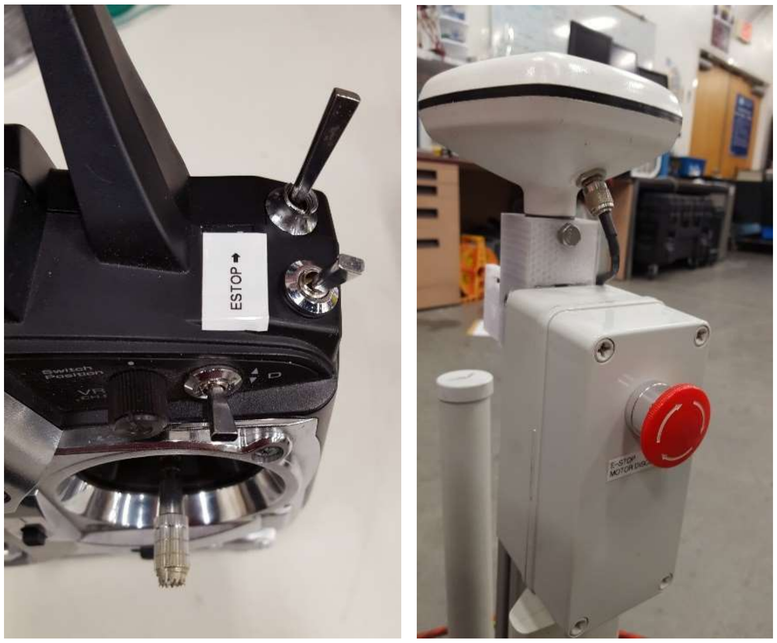


Figure 6

3.3 Electrical Failure Points

The electrical system is designed to minimize failure points. However, they are not completely unavoidable. All the subsystems of the robot operate under the condition that the main switch is activated. If the connection between the main switch and the rest of the system is lost, such as a wire gets knocked out of place, the other parts would not be able to be powered and Metaknight would not be able to run autonomously. In this case, the wire could be replaced or repositioned to restore connections. This could happen with any component, but the resolution is easy and quick with this modular design.

4 Software Design

4.0 Software Overview

4.0.1 Introduction

Last year the Robotics Club switched all of the project platforms from a complete in-house codebase to an architecture built on the Robot Operating System (ROS). ROS provides significant benefits through easily integrated custom solutions with the wealth of well-tested solutions developed throughout the ROS community. This rich ecosystem provides options for configurable implementations of common robotics software functionality including sensor interfacing, 2D navigation, and data logging. ROS itself is essentially a message handling architecture built on XML-RPC (Remote Procedure Call). This architecture provides for communication between modular software components called nodes by subscription to and publishing of messages on named topics using well-defined message types. This message handling affords ROS its modular nature, allowing nodes to fulfill their intended purpose without direct awareness of any implementation deeper than this message interface.

4.0.2 ROS Development Toolset & Visualization

Community developed packages are not the only advantages provided by ROS over previous in-house solutions. Several tools exist as part of larger ROS support for simulation of entire robot systems and visualization of both simulated and real sensor data. The two most frequently used tools in our development process are the Gazebo simulation suite and the visualization tool rviz.

ROS visualization (rviz) is a ROS package that provides for visualization of a large variety of sensor data and other robot state information. Important in development of this year's platform were the ability to view LIDAR scans, raw camera feeds, and real-time views of many data structures used by the robot's navigation systems. These data structures include the current map being used, several different paths generated by the path-planning systems, and the current navigation goal or waypoint.

4.0.3 Network Communication

Point-to-point wireless communication for data communication is handled by using two Ubiquiti Rocket M5 BaseStations. Both ends are equipped with a 13dBi omnidirectional antenna. The IP network will be using a standard class-C subnet with a network address of 192.168.1.0/24. The setup has been tested using JPerf to provide a throughput of 75 Mb/s with a latency averaging 15ms across approximately 100ft. We believe that this setup can provide a reliable consistent connection for the scope of the project. Due to the nature of the course, interference is projected to be minimal. With other teams on the field, there may be a possibility of degraded performance due to channel overlap within the 5 GHz spectrum but the system is capable of multiple interference mitigation techniques to minimize this issue if necessary.

4.1 Localization

Three separate sources of odometry information are generated using the ROS *robot_localization package*. Two key nodes are used from this package, *robot_localization_ekf* and *navsat_transform_node*. The *robot_localization_ekf* node uses an extended Kalman filter (EKF) to fuse robot state measurements obtained from odometry sensors into a single robot state estimate in the robot's world (or map) frame. Two of these nodes are used. The first fuses only continuous sensor data consisting of wheel encoder data and IMU measurements to provide an odometry estimate that is not subject to the discrete jumps possible when using GPS data. The second incorporates continuous sensor data as well as the discrete measurements obtained from the GPS unit, providing the best estimate possible of the robot's odometry at any instant. Figure 7 shows an overview of the flow of various important measurements and node outputs in obtaining these odometry sources. The *navsat_transform_node* uses a GPS fix and the robot state estimate from the second EKF result to produce an odometry source using primarily the GPS data. This node also publishes a transform between the robot's map frame and the Universal Transverse Mercator (UTM) frame, which is a method of representing locations on the Earth in a

2D Cartesian coordinate system. This transform between the map frame and UTM frame allows navigation waypoints to be specified easily as UTM grid coordinates, which can be obtained from supplied latitude/longitude GPS coordinates. Figure 7 shows the important transforms necessary for this use.

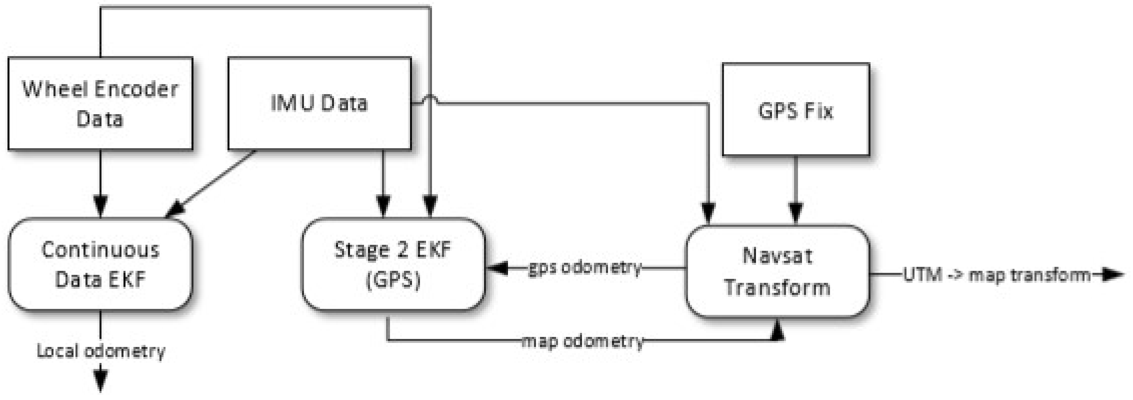


Figure 7

4.2 Mapping

Map building is accomplished by use of the ROS *gmapping* package. This package provides an implementation of the SLAM algorithm, which both continuously builds and updates the current map and serves as a primary means of localization. The laser scan generated from the vehicle's LIDAR is merged with the three faked laser scans generated by the lane detection cameras. The resulting scan is transformed to a new coordinate frame at the center of the frames where the LIDAR and camera scan origins lie. The ranges reported in the merged scan, the robot's current orientation, and the transform between the merged scan's coordinate frame and the map are then used to populate obstacles.

4.2.1 Lane and Obstacle Detection

Lane detection is accomplished using three Logitech c930 webcams, one at the top of the vehicle sensor mast and two at the front of the vehicle closer to the ground. The cameras are tilted downward and placed to capture lane lines within several meters of the vehicle's footprint. Incoming images are put through a processing pipeline using the OpenCV libraries to obtain an image containing only best-fit approximations to lane lines in the image. The resultant images are then analyzed to produce spoofed LIDAR scans containing the ranges to lane lines detected in each camera's field of view.

First, thresholding is applied on white and orange to obtain two binary images. Figure 8 shows the white thresholding on the left and the orange thresholding on the right. The image obtained via white thresholding contains the location of potential lane lines as well as white components of course obstacles such as barrels. The image obtained via orange thresholding contains parts of the image containing obstacles. This second image is then transformed using the OpenCV morphology open operation, expanding the detected orange areas to fill in detected white space like barrel lines between orange obstacle components. By subtracting the morphed image from the white thresholded image, a binary image containing only lane lines is obtained.

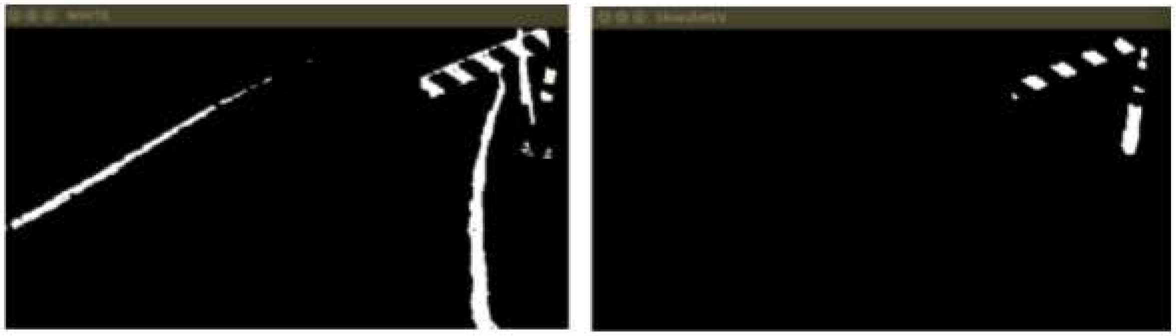


Figure 8

4.2.2 Semantic Inference

Lane line approximations are then obtained from the filtered image. The image is first split into a grid of uniform cells. In practice grid sizes of 4x4 to 8x8 function adequately with the available computing resources. The OpenCV fitLine function is then applied to each cell. This function applies a weighted least-squares algorithm to generate a straight-line approximation fitting the line detected in the supplied dataset. The straight line approximation replaces the data in the cell, and the cell is inserted back into the image. By applying the fitline approximation on individual cells in the image, potential gaps hidden by obstacles or poor quality of incoming images are bridged. Applying this technique to each cell in the image yields an image with piecewise approximations of the complete lane lines detected.

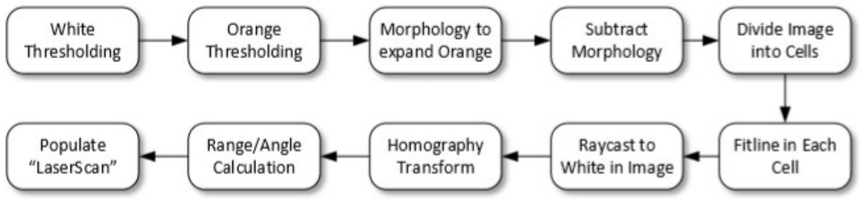


Figure 9

Raycasting is then performed from the bottom center of the image at incremental angles. Bright blobs in the binary image that are encountered are considered lane lines. Pixel (x,y) coordinates are obtained and converted to distances from the camera using a homography transformation and the camera's known orientation. Each detection's range and the angle along the camera's field of view, taking the origin as the center of the image, is recorded for map population in a faked LIDAR scan. This process allows treating lane lines in the mapping and path planning services as if they were any other course obstacle.

4.3 Navigation

4.3.1 Navigation Goals

Using pre-existing ROS packages sensor interfacing, mapping, path planning, localization, sensor fusion, and complex coordinate frame transformations, are readily available as part of the core ROS distribution. These packages are integrated into the platform with custom image processing nodes to complete course navigation goals.

4.3.2 Path Planning

General navigation tasks such as path planning and local obstacle avoidance are accomplished by use of the ROS Navigation Stack. At the highest level, the Navigation Stack uses incoming LIDAR scans, the current map, robot odometry information, and geometric transforms between coordinate frames to produce velocity commands for local navigation. Internally, this is accomplished by separate local and global path planning and generic recovery behaviors for when path planning fails. Both planners are reconfigurable at runtime through a large number of parameters for tuning navigation performance.

The global path planner (*navfn/NavfnROS*) generates paths from the current robot location to an arbitrary waypoint supplied to the Navigation Stack using Dijkstra's algorithm. This path avoids known obstacles between the current location and destination and is regenerated if new map information is discovered between these points. An important capability of the global planner is that movement goals can be set for any frame that has a transform path from itself to the map's coordinate frame. The goal's frame is specified in the call to the Navigation Stack's path planning services, and the goal coordinates are then always transformed to map coordinates. This allows waypoints to be specified in the map itself, relative to the robot's current location, and (as detailed in the Localization and Waypoints section) even indirectly from GPS coordinates.

The local path planner uses incoming Velodyne scan ranges to determine where obstacles are located within a definable range from the robot. The local planner (*base_local_planner/TrajectoryPlannerROS*) uses this information to generate paths adhering as closely as possible to the global path while avoiding the locally detected obstacles. In the event all local paths result in collision with detected obstacles, the local path planner is able to initiate the Navigation Stack's recovery behaviors in an effort to yield valid local paths.

4.4 Software Failure Points

The largest problem encountered for this year's competition is the retention of consistent programming members for the team. While the Robotics Club attracted a lot of dedicated mechanical and electrical members, little to no new members had any programming experience or interest in programming. While we held one member from the previous year that had experience with IGVC, he graduated during the production of the vehicle and left the team. While he attempted to train multiple people for this year, they did not commit to the project and are no longer a part of the team either. Our final resort led to assigning programmers from different projects at the last minute to help out the best they could, with only some documentation to guide them through the vehicle.

5 Integration Management

Integration of the entire platform was planned to be a step-by-step process. The mechanicals started working on building and upgrading the hardware of the robot while the electricals wired it up and together, they got it moving. While it was mobile the programmers were able to test out their code and work with the camera vision, lane avoidance, and obstacle avoidance.

Though the plan was step-by-step, the implementation was altered slightly. Since there were problems with the vehicle properly avoiding certain obstacles, the sensors had to be repositioned and rewired. For example, the original design had only one camera on the mast, but since that was not sufficient in seeing the lane lines two more cameras were added to the sides of Metaknight. There was a lot of trial and error, but this was not unexpected. The team properly performed together structurally and efficiently.

6 Testing

6.1 Simulation

By testing the vehicle in a digital environment, bugs can be worked out without compromising the physical vehicle. Gazebo is used for this goal. It is a general purpose robotics simulator including full graphical presentation, customizable physics simulation, and extension of its feature set through user-made plugins. As with ROS itself, many plugins are available throughout the ROS and Gazebo communities. The Gazebo development provides packages for wrapping ROS features and providing an interface for communication with a simulated robot using ROS messages. Figure 10 shows an example of running a vehicle in Gazebo.

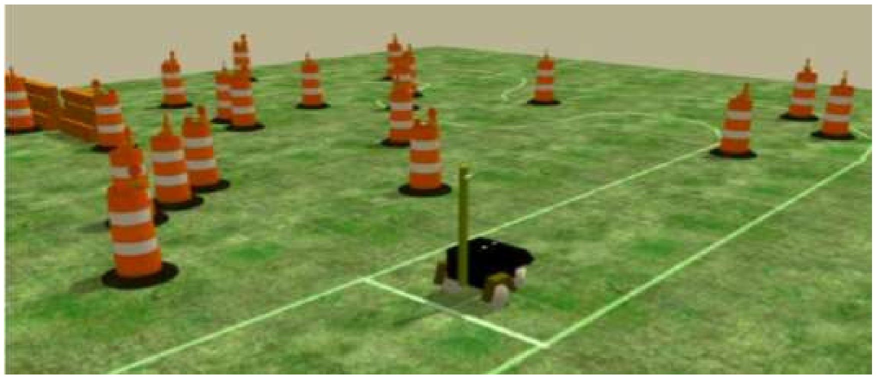


Figure 10

6.2 Small-Scale Real-World Testing

By investing in a small platform known as Turtlebot, the team was able to practice autonomous capabilities while Metaknight was in construction. The Turtlebot platform is a Kobuki base with tiers added for sensor mounting. Those sensors include an Xbox Kinect for the camera and infrared laser point cloud, a Linksys gaming adapter for wireless communication, and a ROS-capable netbook. By using this platform, the programming team tested out lane following and thresholding. Figure 11 shows one of processes in testing.

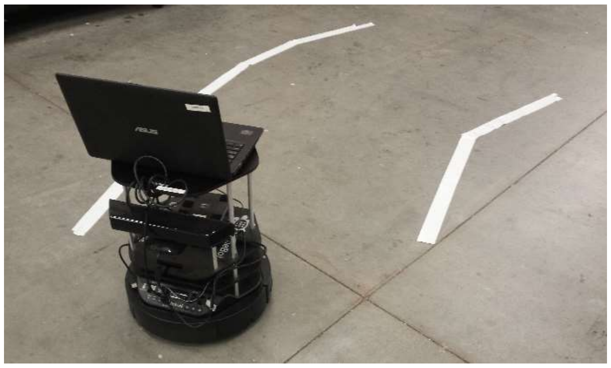


Figure 11

6.3 Large-Scale Real-World Testing

Once our competing platform was fully built and ready for testing, the programming team was able to take the code used in testing the small-scale platform and integrate it into Metaknight. Figure 12 shows a recent picture of the completed model. When putting together the full platform, all components fit the way they were planned to, but there were a few components that required the iterative method. The drive configuration had to be changed, of course but in changing that configuration, the remote control directions were unstable. After much investigation, it turns out there were crossed wires and the problem was confronted. Software problems occurred when the platform was tested outside. The cameras were having difficulty white balancing, causing the thresholding not to work properly.

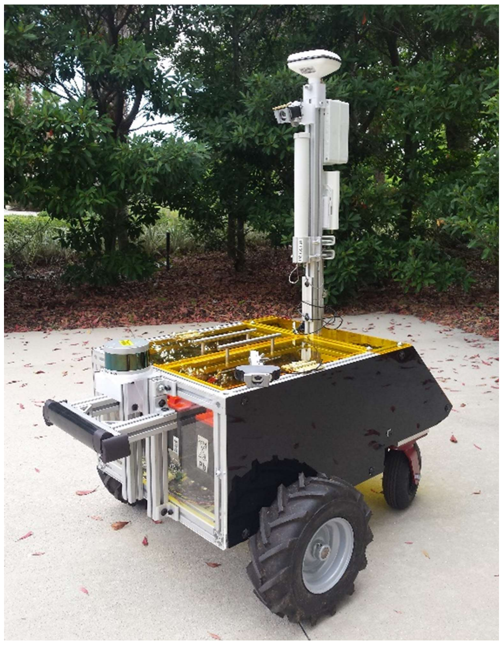


Figure 12

7 Current Performance

Metaknight is a very unique and innovative platform that was built for the competition but also to provide hands-on experience for the students working on it. The students have successfully learned a lot from this project, so even though there weren’t a lot of veteran members working on this project, the next generation of roboticists will be more knowledgeable and able to perform even better for years to come. The hardware of the vehicle is fully complete, with every detail laid out in the CAD also available in the full-scale model. There are no points on the hardware that are expected to fail, and every precaution has been taken for possible failures.

Currently, the platform can successfully drive autonomously. Metaknight has full capability to detect obstacles, attempt to avoid obstacles, and drive between given waypoints. There are some problems that have not been resolved, such as the thresholding issues and untested challenges like distinguishing small white flowers from painted white lane lines. There are plans to resolve this problem, but at the present they have not been implemented and there is no precise way to replicate the environment in which those problems occurred.

Although there are still points to test and ways to add on to the current model, the platform is considered complete. There is always room for improvement and there is more innovation to come, so until the competition this work in progress is confidently meeting expectations.