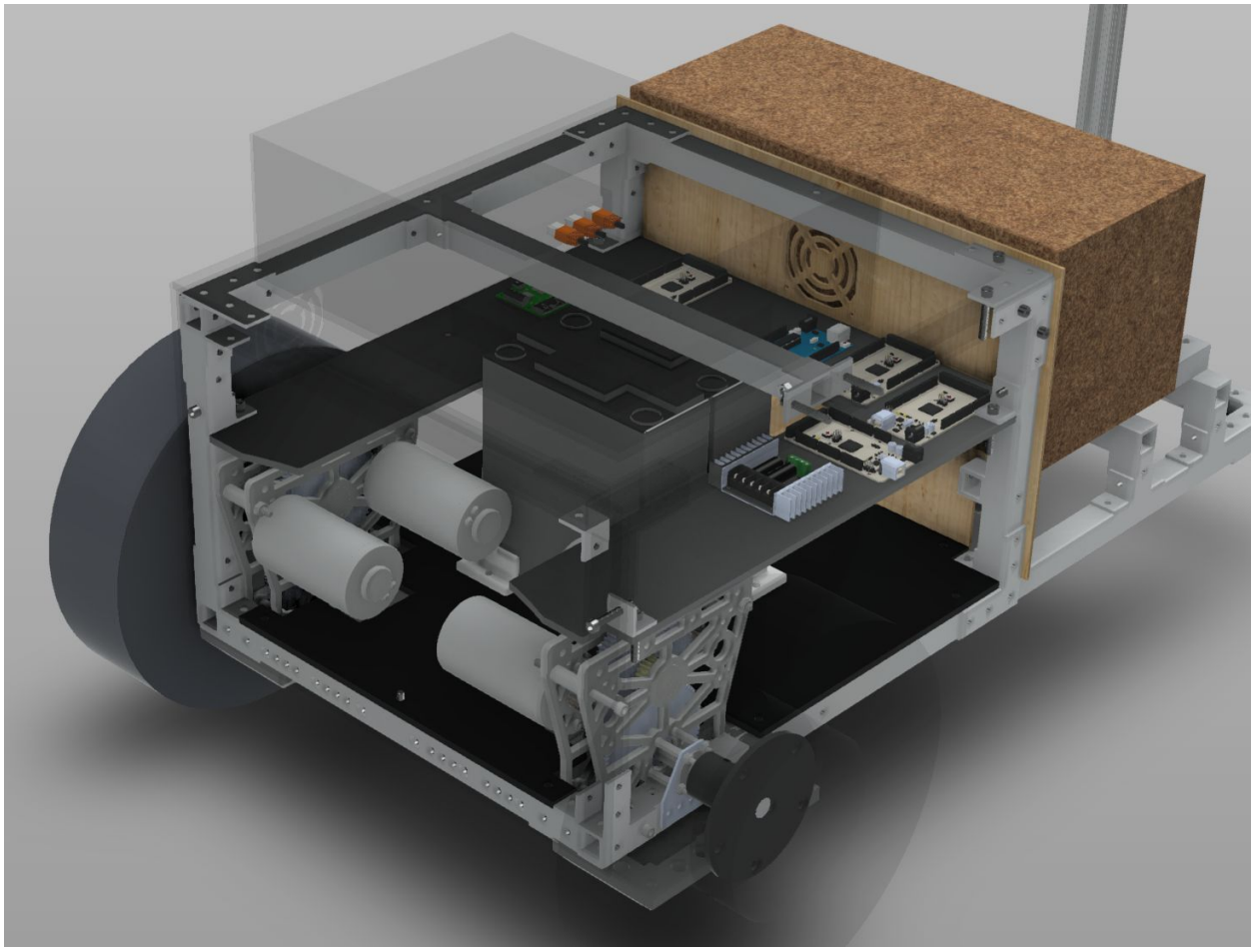
GOAT Design Report

University of Michigan - Ann Arbor



Submitted 2019-05-15

Team Captain: Gregory Meyer | gregjm@umich.edu
Faculty Advisor: Matthew Johnson-Roberson | mattjr@umich.edu
Damen Provost | provostd@umich.edu

Team Roster

First Name	Last Name	Email	Leadership Position
Aditya	Bhatt	bhattadi@umich.edu	
Akshay	Subramaniam	akshaysg@umich.edu	
Allison	Heinen	heinenal@umich.edu	
Atishay	Singh	atishays@umich.edu	
Caesar	Olivas	caesaro@umich.edu	
Cameron	Kabacinski	camkab@umich.edu	Platform Coordinator
Cedric	Bernard	cedrib@umich.edu	Engineering Director
Christopher	Marsh	marshchr@umich.edu	Controls Coordinator
Claire	Russel	russecla@umich.edu	
Daniel	Jung	ydanielj@umich.edu	Operations Director
Daniel	Castro	castroda@umich.edu	
Elton	Lin	eltonlin@umich.edu	
Eric	Liu	ericxliu@umich.edu	
Esti	Gajda	egajda@umich.edu	
Gabriel	Rababeh	grababeh@umich.edu	
Gregory	Meyer	gregjm@umich.edu	Team Lead/President
Griffith	Heller	gtheller@umich.edu	Power Coordinator
Hersh	Vakharia	hershv@umich.edu	
Himan	Yerrakalva	yerrak@umich.edu	
Ian	D'souza	idsouza@umich.edu	
Ian	Cook	ijcook@umich.edu	
Isaak	Hedding	ihedding@umich.edu	
Isha	Hameed	ishameed@umich.edu	
Justin	Lee	lejustin@umich.edu	
Karolina	Rak	rakka@umich.edu	
Kyle	Scott	kyscott@umich.edu	
Matthew	Saraceno	mattsara@umich.edu	Computer Vision Coordinator
Saad	Alkalby	salkalby@umich.edu	
Sam	Halaseh	halaseh@umich.edu	
Samuel	Hall	samjhall@umich.edu	Sensors Coordinator
Sean	Oliver	snoliver@umich.edu	

Tara	Sabbineni	trsabb@umich.edu	
Zachary	Pozsar	pozsarza@umich.edu	Business Development & Outreach Coordinator

Table of Contents

Team Information

Introduction

The Ground Obstacle Avoidance Transport, affectionately referred to as GOAT, is the first submission on behalf of the University of Michigan Ann Arbor to the Intelligent Ground Vehicle Competition. GOAT was designed, manufactured, and tested by the University of Michigan Intelligent Ground Vehicle Team (UMIGV). Founded in fall 2016, UMIGV is an engineering design team comprised of a mix of undergraduate engineering, business, and liberal arts students. Our team believes that hands-on education complements classroom learning; any student can learn through robotics, regardless of their background. Our work has a meaningful societal impact that everyone can learn from. The future is promising for autonomous robotics and we aim to incubate our members to meet the challenges of tomorrow in this field.

In our first year, through a lot of trial and error we established and scaled a team, assembled various prototype subsystems and hosted several demos to raise awareness and funding for our project. In our second year, we scaled up our team started development on GOAT for entry in the 2018 competition. This third year we've continued building up our autonomous platform in preparation for the 2019 competition. UMIGV is supported by campus partners including the Michigan Library System; Michigan Robotics Institute, the College of Engineering Multidisciplinary Design Program; optiMize Social Innovation and the Barger Leadership Institute in the College of Literature, Science & Arts. Corporate sponsors include Aptiv, Raytheon, and Ford.

Team Organization

UMIGV as a team enables students to gain valuable work experience that complements and expands upon classroom instruction. This environment teaches them how to work together collaboratively and apply the classroom knowledge in a very hands-on way; students gain soft skills and critical thinking insights which strengthens their advantage in a competitive job market. UMIGV offers value to the University by furthering its educational mission as well as an experience the University can use to recruit future students. To corporate sponsors, the team offers a tried and tested talent pool of students who have real-world project experience.

UMIGV is organized in a way to maximize the learning experience. Our leadership is comprised of five technical systems leaders, a business development coordinator, engineering director, operations director, and team leader. Each technical systems leader and coordinator of the leadership served to guide and teach members in their stewardship. The team leader, engineering director and operations director enabled the team with resources and strategy. In addition, members of the team were not bound to a particular group, allowing for flexibility in meeting team backlogs and giving members the chance to find a discipline they truly enjoyed.

Design Process & Assumptions

The design process for GOAT was initially started in the summer of 2017 with conceptual drawings and since then it has since gone through several design iterations. Our team followed the V model of systems engineering and integration as outlined by INCOSE. Each component of the robot was broken down into the smallest discrete pieces possible, leading to our five technical subsystems: Platform, Power, Sensors, Controls, and Computer Vision.

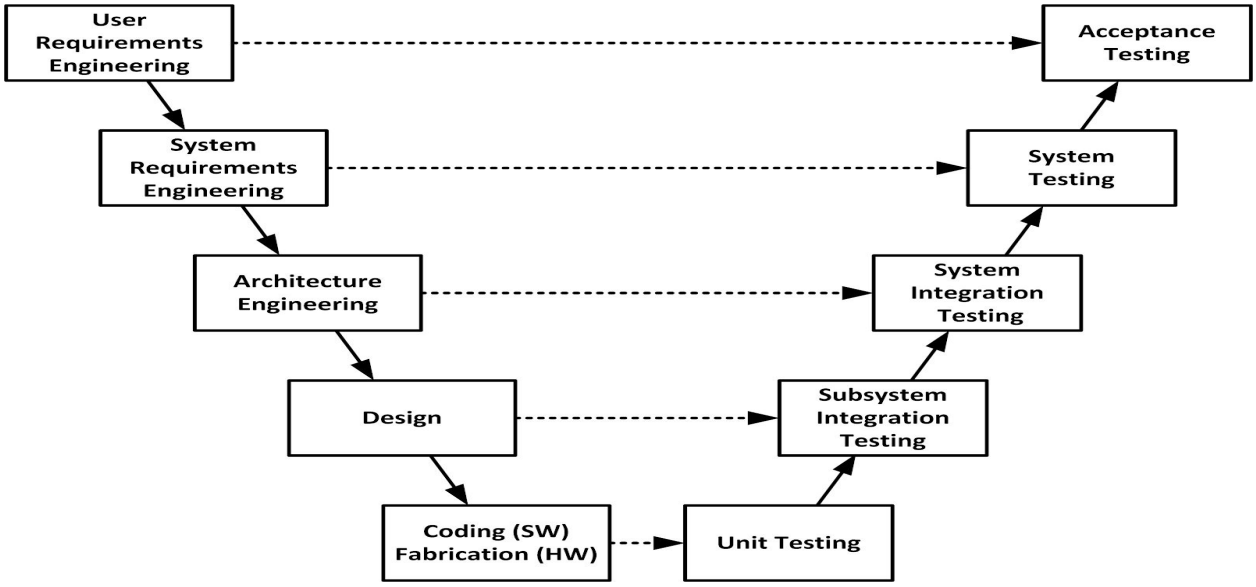


Figure 1: The V model of systems engineering

UMIGV conducted work using the Agile methodology (Figure 2). Every month, we held sprints - a period of time with fixed number of tasks - to leverage all our members participation and accommodate incongruent schedules. At the end of each month, subteams would run an in-house demo of the work they completed and create technical goals and tasks to complete by the next month.

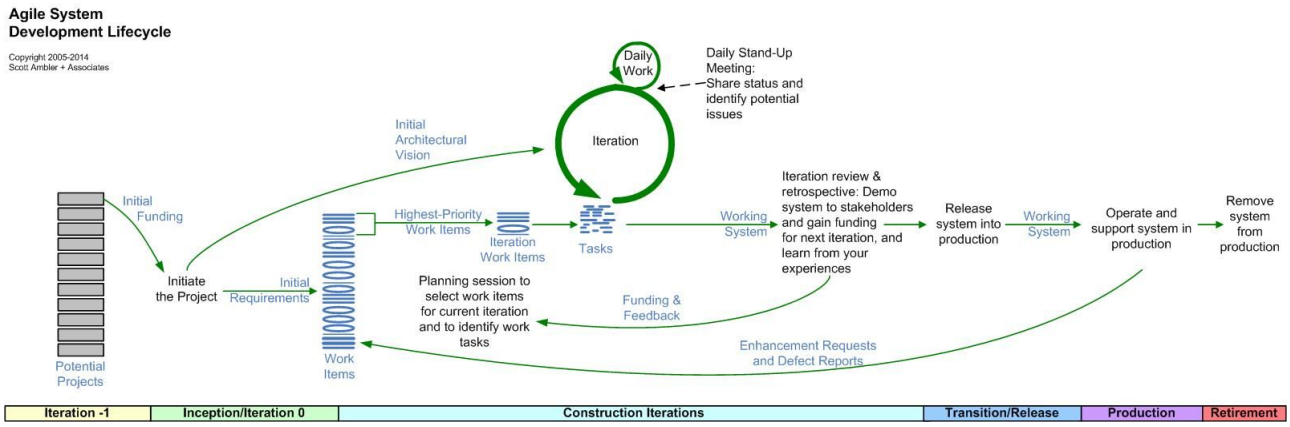


Figure 2: Agile development method

The team employed industry-standard programming practices. We used GitHub for version control and strict style guidelines to ensure our code was clean and maintainable, combined with code reviews conducted by technical leads and design reviews with our faculty and corporate sponsors. We documented our work on an internal wiki for future generations of the team. We also used the wiki as a platform to compile resources and onboard new members. We felt it was critical to lay a good foundation in terms of practice and hope to implement continuous integration, automated regression testing, and automated builds in the coming years.

Some assumptions made during the design process included:

- Only splash proofing was required for waterproofing our system
- In a modular system, a good overall big picture is important and the details can come later
- Some of our odometry was perfect (neglecting wheel slippage)

Vehicle Design Innovations

The exterior panels for the GOAT use a quick attach system to increase serviceability. In response to pains with battery swapping and charging, we designed 3d-printed brackets to allow faster replacement of batteries through the GOAT's side panels instead of having to remove the top cover and disconnect all of our compute hardware. To facilitate easier stationary testing, we designed and implemented a system that would automatically switch between wall power from an ATX power supply and the robot's internal batteries.

Platform Systems

The platform system's objective is to provide a chassis, payload support, and locomotion methods for the GOAT. The team utilized various machining methods to produce the platform. We utilized 3D printing for rapid prototyping and visualizing concepts, SolidWorks for design visualization and spatial layouts, CNC milling and water jetting for precise metalworking, and laser cutting to form our plastic sheets.

Chassis

Designed with portability, serviceability, and longevity in mind, the chassis is constructed from 1 inch square aluminum tubing in order to minimize overall system weight and maintain rigidity. The dimensions are within the competition requirements at 28 inches wide by 36 inches long and well below the height limit. The chassis is secured by custom-made mending plates and attachment brackets and secured with common 1/4 inch thread hex nuts and bolts to improve serviceability. The brackets and mending plates can be easily manufactured and the hex bolt ensures that no stripping will occur.

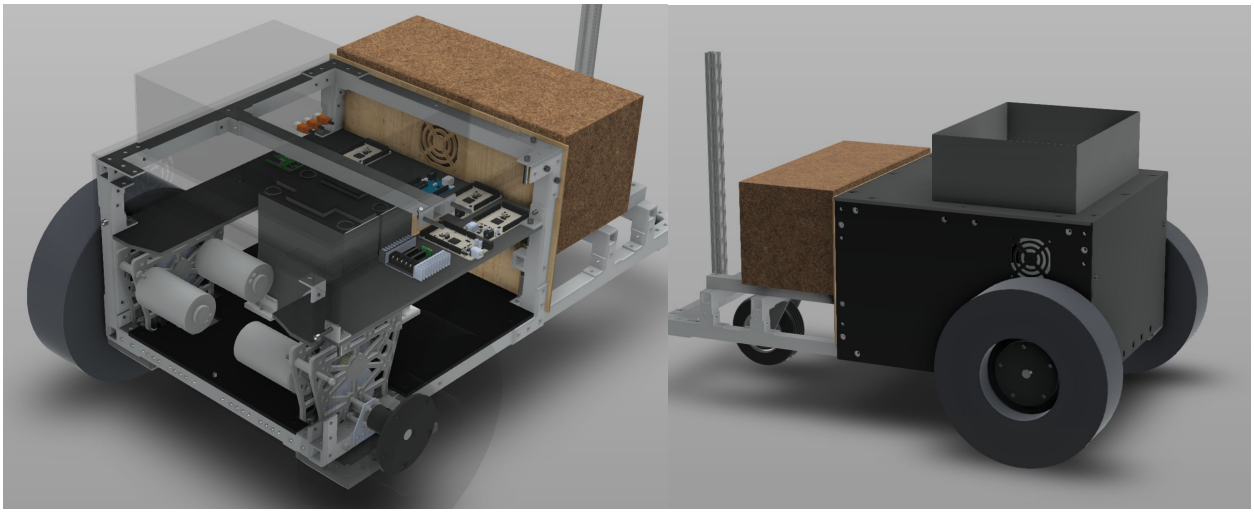


Figure 4: The structure and chassis

Structure

GOAT features a two-wheel drivetrain with a third free caster wheel to provide a balance between power, stability, and maneuverability. Since the front wheels offer a majority of the support, the design is forward-heavy (Figure 4). There is no built in suspension, but the caster wheels has proven to be useful in balancing the robot on rough terrain. Inside the main chassis of the robot, there are two gearboxes with motors, encoders, batteries, and a shelf for all power and safety components. The batteries are placed behind the drivetrain to act as a counterweight against the fairly heavy gearbox assemblies. We use 3D printed brackets to hold the batteries firm against the chassis. Above the main structure, there is a splashproof box containing the computers that also serves as a mounting point for our lidar and stereo camera. At the rear of the robot, there is a safety pole with an LED safety light and emergency stop button.

Weatherproofing

Once the frame was designed, we designed exterior panels to screw into the chassis. These panels are made out of 1/4 inch high density polyethylene (HDPE) with weather stripping between the panels and chassis, affording us a splashproof platform.

Power Systems

The power systems of GOAT incorporates all electronic components and handles power distribution and signal routing while serving as an interface between the controls and platform systems.

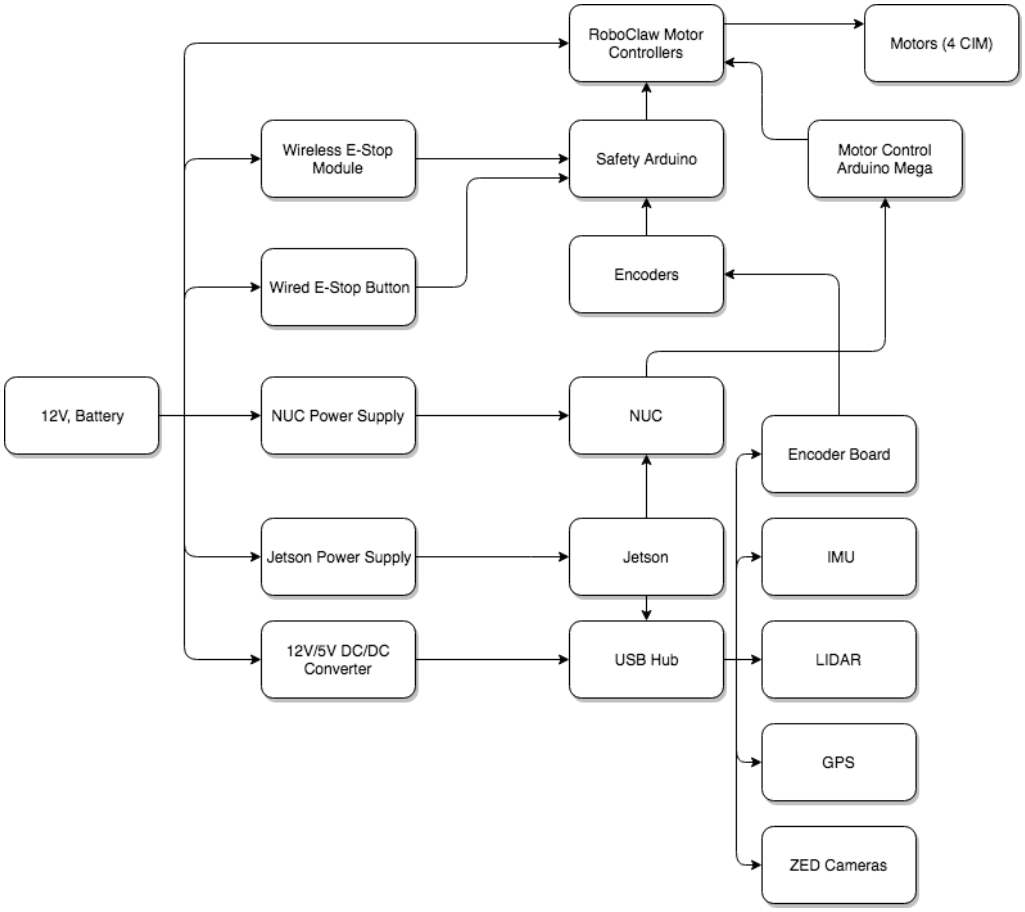


Figure 5: Block Diagram Overview of the Electrical System

Power Supply

The robot is powered by two 12V car batteries wired in parallel for a combined total of 32Ah of capacity. A 70A circuit breaker is used to connect and disconnect the batteries from the rest of the system. The motor controllers, wireless E-Stop module are wired directly off of the batteries,

while more sensitive electronics use off-the-shelf DC-DC converters. Each RoboClaw motor controller is configured to operate one side of the robot, each having two motors.

Average Power Consumption Numbers	
Motors (x4)	360W
USB Hub	5W
Jetson	15W
NUC	20W
E-Stop System	5W
Total	405W

These consumption numbers are a conservative estimate based off of manufacturer-supplied figures.

Motors

The GOAT's gearboxes are configured with two motors per output shaft, so our motor controllers are configured to bridge together the output channels as if it were operating one large motor. Using the RoboClaw motor controllers also allowed us to control the currents going through the motors.

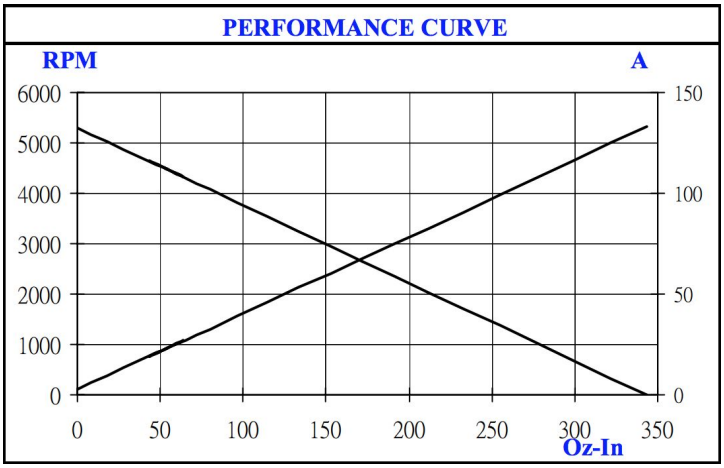


Figure 6: Performance Curve for the motors

A gear ratio of 30:1 was implemented in our gearboxes, which gives us a necessary rotation speed of 3,100 rpm from our motors to go a speed shy of 5 mph when accounting for the size of the wheels. The amount of current that could potentially be drawn at this speed can go up to around 70A per motor, necessitating a current limiting mechanism for contingencies such as sudden accelerations. We leveraged the built-in current limiting functionality of our motor controllers to limit power draw to 40A per side.

Safety Mechanisms

Being able to operate our robot safely is a key part of the competition. When enabling the robot, main power from the batteries is enabled by flipping a circuit breaker mounted on the outside of the bot, easily seen and accessible by anyone. When the robot is turned on, power is supplied to a status light, showing its current state. In an effort to make the robot more modular, we used bullet connectors to make connections between the various components.

To ensure that no safety issues arise during a run, a wired E-Stop, wireless E-Stop, and speed limiter are integrated on the robot using a dedicated Arduino microcontroller. The wireless E-Stop we selected has a range of 250 feet, allowing the robot to be safely stopped from anywhere. The large red E-Stop button and wireless E-Stop are connected to a digital I/O pin on this Arduino. Triggering either E-Stop will cause the microcontroller to send a signal to the shutdown pin on the RoboClaw motor controllers. A 5 mph speed limit is imposed on the vehicle by counting the encoder ticks with the microcontroller; if the wheels have turned at an average speed of 5mph or more for the last second, the same shutdown signal that the E-Stop systems use is sent to the motor controllers.

Sensor Subsystems

The sensors subsystem interfaces with sensors to collect observations about the world for localization and map building. Pose observations are fused to produce an estimate of the robot's current pose and twist in a plane. Odometry and lidar observations are then used for simultaneous localization and mapping (SLAM), outputting an 2D occupancy grid of the robot's environment. The controls subsystem then takes the occupancy grid, odometry, and raw lidar range data for use in path planning and obstacle avoidance.

Hardware and Components

- NVIDIA Jetson TX2 Computer
- Intel NUC7 NUCI5BNKP Computer
- NETGEAR GS105Ev2 Switch
- Velodyne VLP-16
- Stereolabs ZED Camera
- Phidgets PhidgetSpatial 3/3/3 High Precision IMU (1044_0B)
- Phidgets PhidgetEncoder High Speed 4-input Encoder Board (1047_1B)
    - HEDSS Optical Rotary Encoders (Phidgets 3530_1)
- Garmin GPS 18x USB
- Arduino Mega 2560 Rev3
- Ion Motion Control RoboClaw 2x60A Motor Controller

Hardware Design Decisions

NVIDIA Jetson TX2

The NVIDIA Jetson provides a high-speed discrete GPU suitable for real-time image processing and pairs particularly well with the ZED camera, as Stereolabs maintains an SDK specifically for the Jetson TX2. The Jetson provides exceptional performance considering its power consumption, form factor, and price. In addition, the included development board has integrated HDMI, Ethernet, USB, and WiFi to speed up development.

Intel NUC

The Intel NUC provides a laptop CPU in a compact package without requiring the expertise of a dedicated embedded system. The i5 processor was selected after analyzing the expected computational power required to run navigational subroutines, such as path planning. The NUC provides all of the amenities to be expected from a high-end ultrabook without the peripherals not required for autonomous operation.

NETGEAR Switch

Strong long network connectivity was required to capitalize on the ROS' distributed system capabilities. The ZED camera can output 3D video at up to 2K 15FPS, so high bandwidth communication was a must. Having an Ethernet switch allows us to quickly link in new devices like laptops. Combining this modular system with the nature of ROS allows for quick visualization by connecting to the robot's local network with an Ethernet cable.

Velodyne VLP-16

Unexpectedly low obstacles in last year's competition necessitated that we act quickly to install and integrate the Velodyne VLP-16 3D lidar; our previous lidar only scanned in a plane. The Velodyne boasts significantly increased range, accuracy, and weatherproofing over the RPLIDAR we used last year and has become an integral part of our sensor systems.

Stereolabs ZED Camera

The primary draw of the ZED camera is its low cost and high depth sensing range. Compared to other RGBD solutions, the ZED camera offers much higher depth cloud resolution through software processing of the stereo images. Unlike systems that rely on infrared light, like the Kinect, the ZED retains nominal functionality in direct sunlight. The Stereolabs development team has provided a rich SDK with ROS integration included, speeding up deployment cycles by reducing hardware and embedded development time.

Phidgets IMU

The Phidgets ecosystem has a strong draw due to its well-documented and maintained C library and ROS integration. Another draw is the relatively low cost of the Phidgets system, but this results in less accurate sensor readings with higher noise.

Phidgets Encoder Board

As with the IMU, the Phidgets ecosystem is easy to work with and has strong open-source support. Unlike the IMU, no off-the-shelf system for differential drive odometry was available, so software was developed to interface with the C library and generate twist estimates from wheel encoder sensor data.

Garmin GPS 18x USB

Last year's GPS was connected via an RS-232 serial port and required significant jury-rigging to get powered as well -- it was designed to be powered from a 12V cigarette lighter socket. The 18x USB offers similar functionality, but with a more developer-friendly USB interface. The 18x is designed for automotive applications and as such comes weatherproofed, a significant factor in our decision to keep using the same model.

Arduino Mega

A simple hardware/software layer was required to interface between our ROS layer and the serial interface of the RoboClaw motor controllers. Using an Arduino Mega allows for us to process ROS messages on a lower-level device, allowing maximum abstraction of the drivetrain to the ROS stack. In addition, the RoboClaw's manufacturer provides and maintains an Arduino library to interface with the velocity controls of many motor controllers connected over serial, speeding up development and reducing testing time.

RoboClaw Motor Controllers

The RoboClaw offers tight integration with velocity commands, having built-in PID position and velocity control. In addition, a wide variety of customization and diagnostic options are a significant quality-of-life boost while interfacing with hardware. For example, the motor controllers monitor battery voltage and motor current and will limit performance characteristics as necessary to stay within user-specified operating parameters.

Software Stack & Control Subsystems

All of the robot's software is powered by Robot Operating System (ROS) running on a base Ubuntu 16.04 installation. In line with our modular design philosophy, ROS was selected as the robot's operating system due to its extensive modularity, community support, and power features. ROS is a distributed networking and communications library allowing multiple devices to

work together. A ROS computation graph is divided into discrete nodes that can publish and subscribe messages to build a network of information. Nodes communicate with each other over TCP, allowing them to connect to nodes on other computers through our Ethernet switch. This system facilitates the communication between different processes and enables the team to work on independent tasks; each software subteam can develop nodes entirely separately from the others.

The goal of the controls subsystem is to navigate the robot through a series of waypoints while avoiding obstacles with data from the sensors subsystem. The controls subsystem receives an occupancy grid, lidar data, odometry data (pose and twist estimates), and coordinate transform data, then uses this data to build navigational costmaps and path plan through them. A* was chosen as our path planning algorithm because it is optimal and lets the robot reach its goal more effectively. Goals are generated by transforming GPS coordinates into the robot's world frame and updating the broadcast goal as each set of coordinates is reached.

Sensor Fusion

Sensor fusion between IMU and encoders is accomplished through an Unscented Kalman Filter, which is more forgiving than an Extended Kalman Filter when it comes to calibrating the sensor odometry. The ZED camera is used for white line detection and generates its own occupancy grid which is merged into a 2d costmap with the occupancy grid from the LIDAR. The GPS was chosen to be left out of odometric sensor fusion due to its non continuous nature, which testing revealed significantly reduced the accuracy of pose estimates.

SLAM (Simultaneous Localization and Mapping)

This year, we switched to an off-the-shelf pose-graph SLAM solution, Google Cartographer. Cartographer offers a robust and highly configurable solution that permits us high confidence in the quality of generated maps, especially in noisy environments. Cartographer also has greater potential for scaling up than our previous particle filter SLAM approach, as its sparse representation of the environment allows for significantly more space to be mapped before running into memory usage constraints.

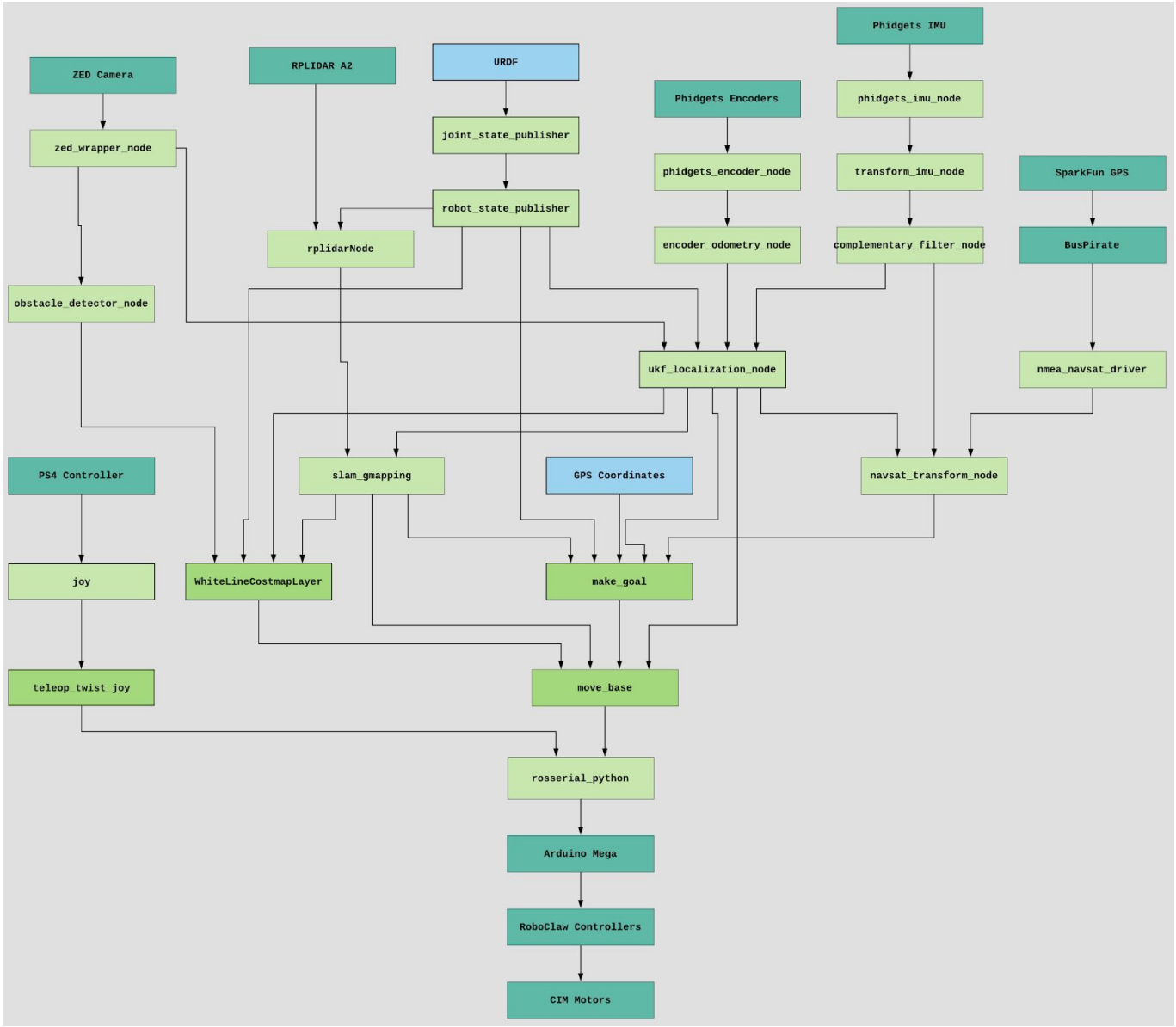


Figure 7: Software stack

Failure Prevention & Risk Mitigation

Hardware Failure

In the case of hardware failure, such as non-functioning sensors, the general troubleshooting process is as follows. 1) Check that status lights are lit and indicate nominal operation. 2) Check that connector cables are securely attached. 3) Verify that software nodes are running and messages are being transmitted. 4) Run ROS troubleshooting like roswtf, rqt_graph, and view_frames to verify that the node and message graphs are properly set up.

SLAM Failure

In case of SLAM scan matching algorithm failure, the newest odometry information is used to estimate the current pose of the robot. SLAM nodes are updated using forward projection according to the optimal solution for the pose-graph.

Testing

Simulations

The robot was simulated in Gazebo. This software was chosen given its integration with ROS is well documented, it is highly configurable, and it is open source. The sensor's hardware was simulated as accurately as possible using Gazebo plugins, which provided all the information necessary for the controls stack to run. The information was published on different ROS topics to which our control subscribed to and used to transmit velocity commands back into Gazebo.

On top of this, RViz was used to visualize the information that was going into the navigation stack which made debugging easier. The map generated by the SLAM algorithm was also visible on RViz, which made it possible to test how the map would be built in competition.
Different environments were chosen to resemble the competition, emphasizing on the presence of construction cones.

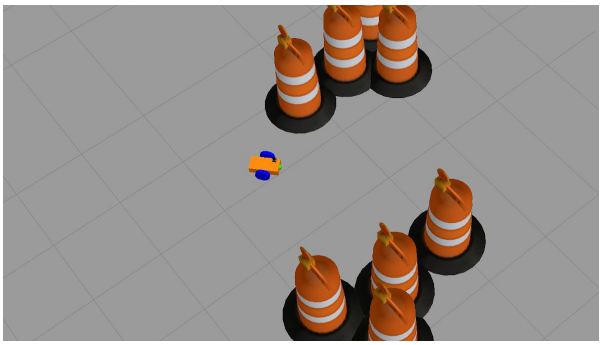


Figure 8: Snapshot of the robot avoiding obstacles in Gazebo.

Performance Testing to Date

The robot's sensor systems have been extensively tested with teleoperation, allowing for verification of sensor fusion and SLAM approaches without using path planning or navigation software. Individual sensors have also been extensively tested before integration for functionality verification.

Initial Performance Assessments

In initial autonomous operation mode testing, the robot shows exceptional path-planning and SLAM capabilities. Maps generated show high accuracy, track corners well, and account for inaccuracies in odometric pose estimates.

Figure 9: Test map of room shown with RViz