Final Design Report

NavG - NGCP

Gonzalo Arana
Alexander Garcia
Justin Nguyen
Andy Velazquez

Introduction

Our Capstone team is assisting the Northrop Grumman Collaboration Project (NGCP) with sensing and intelligence for one of their new vehicles. NGCP's focus this year is to perform a rescue operation in a disaster zone using four autonomous vehicles, two from CP SLO and two from CP Pomona. The sensing and navigation system our team is designing is for the Cal Poly San Luis Obispo Unmanned Ground Vehicle (CPSLO UGV). This system will be used to autonomously and safely guide the CPSLO UGV through an obstacle course that simulates a field of debris in an actual natural disaster.

Project Overview

To safely navigate the obstacle course, the UGV will need to be constantly examining its surroundings, checking if there are any unoccupied and unvisited areas near it, and keeping record of all movements for backtracking. To handle the environmental awareness, the system uses a LiDAR and a stereoscopic camera in order to get as much information about its surroundings as possible. This sensor data is gathered, stored, and visualized in real time in a software platform named Robot Operating System (ROS). ROS also provides a global and local map that are capable of being updated in real time and display a grid of occupied and unoccupied spots nearby. This occupancy grid is then used to make decisions about where to travel and these decisions are ideally kept in a record that can be accessed later.

Clients & Community Partners

The primary client for this project is the Cal Poly SLO NGCP group. Northrop Grumman, who initially started and funded this project 8 years ago, is our client indirectly through NGCP. Our point of contact for Nothrop Grumman and issues relating to the NGCP program is Dr. Lynne Slivovsky. Our point of contact for the CPSLO UGV team is Michael Di Giorgio.

Stakeholders

Besides Northrop Grumman, the CPP and CPSLO NGCP groups, and our Capstone group, other stakeholders in this project include emergency services, police, fire, search and rescue, FEMA, organizations that conduct rescue for disasters, and volunteers for disaster rescue. Disaster victims and communities as a whole around disaster areas are also stakeholders.

Project Goals and Objectives

Our team's goal is to design and implement the sensing and intelligence system for the CPSLO UGV that will allow it to complete its mission. The objectives necessary to meet this goal are as follows:

1. Interface with the mechanical design to ensure the sensing and intelligence system can adequately control the vehicle with the constraints set by the car’s components, mechanics, and capacity.
2. Design a fully-tested object detection and avoidance system to enable autonomous driving within the constraints laid out by the customer.
3. Design a fully-tested waypoint navigation system to provide guidance to the vehicle and (coupled with the collision detection and avoidance system) allow the vehicle to navigate to its intended target.
4. Support a manual driving mode that enables the CPSLO UGV to be safely tested and operated.
5. A fully integrated solution with the rest of the NGCP tools and ecosystem with ample documentation to ensure smooth knowledge transfer between teams and the customer.

Project Outcomes and Deliverables

By the end of next quarter, we will have all of the main navigation peripherals that the CPSLO UGV will use, as well as libraries written for interfacing with these peripherals. We will have also implemented basic autonomous navigation and obstacle avoidance using these peripherals. Given the different timelines that our team is operating on compared to the greater NGCP team, we will be demoing this functionality at the end of next quarter using a test vehicle separate from the real CPSLO UGV. This demo will likely take place at the Educational Flight Range, near Cuesta College. The demo will involve setting up an obstacle course at this site, consisting of obstacles such as cinder blocks, large rocks, and ramps. If there is sufficient progress made on the CPSLO UGV by the end of next quarter, then this demo may take place using the CPSLO UGV.

Background

Mission Overview

This design challenge provided by Northrop Grumman for the 2019-2020 school year is an autonomous rescue system composed of two terrain vehicles and two aerial vehicles. The goal for these vehicles is to perform a two-stage rescue. Stage one is reconnaissance and initial approach. This stage involves using an unmanned aerial vehicle (UAV) to determine where a person in need is located and instructing a larger, unmanned ground vehicle (UGV) to approach the area as quickly as possible without using any obstacle detection. Before leaving base, the larger UGV will load a smaller, rescue UGV inside of itself. The second stage is obstacle avoidance and extraction. As the large UGV reaches the disaster area, it will need to switch into an obstacle avoidance mode that will allow it to safely traverse debris as it approaches the person in need. Once at the destination, the smaller UGV will unload itself, approach and prepare the person for extraction, and await the person's extraction using a vertical takeoff and landing (VTOL) vehicle. As the VTOL completes extraction, the two UGV robots will conviene, reload, and return to base.

Similar Systems

Modern Self-Driving Cars

There are six main components for a self-driving car. The three main hardware components consist of sensors, V2X, and actuators. Sensors describe the components that allow the vehicle to gather information about the environment. V2X allows for communication between the vehicle and other components in the environment. Actuators enable the physical movement of the car.

The three main software components for self driving cars are perception, planning, and control. Perception describes the analysis of data obtained from sensors and V2X. Planning refers to the decision making process that the car undergoes given the data that it can perceive. Control refers to the conversion of planned actions into actual movement through enabling the actuators.

Our role in the project allows us to focus on three of these components: sensors, perception, and planning. Modern self-driving cars often use sensors such as GPS, IMUs, cameras, and LiDAR. According to Nvidia, a fully autonomous driverless car would need the computing power to perform 320 trillion operations per second and keep power consumption at 500 watts. This computing power is what allows self driving cars to perform the perception and planning required to navigate a self-driving car. Given these insights into traditional self-driving cars, we have decided on a set of similar peripherals to look into using for the CPSLO UGV in order to perform the same type of behavior as traditional self-driving cars, but on a much smaller scale.

Peripheral Information

RPLiDAR A2

The RPLiDAR A2 is a 2D LiDAR. It has a scanning range of 18 meters and can retrieve 8000 samples per second. There are ROS nodes available for use with the RPLiDAR. We are able to get 2D Laser Scan data from these nodes, which can be easily used as an input for SLAM and route planning. It's also relatively low cost for a lidar, costing $370.

ZED Mini Stereoscopic Camera

The ZED Mini utilizes high definition stereo video in order to create 3D depth. It also has a built in accelerometer and gyroscope, allowing us to have an IMU built directly into the camera. The ZED Mini also has plenty of sample ROS nodes available on the internet. These nodes are able to retrieve data from the camera in point cloud form, allowing for easy integration with SLAM and local route planning tools. It is a little pricey, with a price tag of $399. However, this is significantly cheaper than other alternatives for providing a 3D point cloud, such as a 3D LiDAR.

Jetson Nano

The Jetson Nano is capable of running Desktop Linux, and by extension, ROS. ROS provides a lot of open source interfaces with popular peripherals, namely the ones that we planned on using for the vehicle. It also has support for SLAM, Simultaneous Localization and Mapping, an essential feature for performing navigation without the use of GPS. The Jetson Nano itself is able to process data from multiple sensors in parallel, and has the computing power to utilize neural networks and computer vision. It's also rather inexpensive, costing only $99. Overall, it was able to connect with the peripherals that we chose to use for this vehicle, while also providing a means for easy support to use these peripherals.

Engineering Specifications

Stakeholder Requirements

The following stakeholder requirements were derived from the RFP from Northrop Grumman by our team and the NGCP management.

Stakeholder Requirements	
ID	**Requirement**
SD1	The vehicle shall have mechanisms to load and transport the CPP UGV into the disaster zone.
SD2	The vehicle shall hold a payload volume of 12x7x7in. (LxWxH).
SD3	The vehicle shall house all avionics on a removable electronics tray.
SD4	There shall be adequate documentation describing the assembly and operation of the UGV
SM1	The vehicle shall have autonomous waypoint navigation.
SM2	The vehicle shall be able to autonomously detect and avoid obstacles during navigation.
SM3	The vehicle shall utilize first person view (FPV) video devices for all manual control through the GCS.
SM4	The vehicle shall be able to maneuver over obstacles of 2" in height.
SM5	The vehicle shall be able to drive up a 36% grade slope.
SM6	The vehicle shall have a range of 3000ft. on flat ground.

SM7	The vehicle shall have the capability of mission speed of 3mph on level surface.

Table 1: Stakeholder Requirements

Capstone System Requirements

The system requirements below outlines how we will design our vehicle. These requirements flowdown from system requirements for the entire CPSLO UGV requirements we derived from the stakeholder requirements. The CPSLO UGV requirements can be found in our appendix.

Capstone System Requirements				
ID	**Subsystem**	**Requirement**	**Meets Stakeholder Requirements**	**Meets BUGV System Requirements**
SR1	MECH	The vehicle shall have enough power to transport the avionics package	SD1	BUGV-SR
SR1.2	ELEC, MECH	The vehicle avionics shall fit on a removable electronics tray	SD3	BUGV-SR.2
SR2	ELEC, MECH	The vehicle shall drive in a controlled manner.	SM7	BUGV-SR3
SR2.1	ELEC, SENS	The vehicle shall be capable of 3mph while operating.	SM7	BUGV-SR3.1
SR2.1.1	ELEC, SENS	The vehicle shall measure its ground speed to within +/- 0.5 mph.	SM5, SM7	BUGV-SR3.1.1
SR2.2	ELEC, SENS	The vehicle shall have precise control of its speed to within +/- 0.5 mph.	SM5, SM7	BUGV-SR3.3
SR2.3	ELEC	The vehicle shall have a kill switch.	-	-
SR3	ELEC, SENS, SW	The vehicle shall be capable of short range automated navigation via external waypoint.	SM1	BUGV-SR4

SR3.1	SW	There shall be an interface for the user to input waypoints into the vehicle.	SM1	BUGV-SR4.1
SR3.2	SENS, SW	The vehicle shall autonomously detect obstacles during waypoint navigation.	SM1, SM2	BUGV-SR4.2
SR3.2.1	SW	The vehicle shall be capable of evaluating collisions at at least 1 Hz.	SM1, SM2	BUGV-SR4.2.1
SR3.3	SW	The vehicle shall autonomously avoid obstacles during waypoint navigation.	SM1, SM2	BUGV-SR4.3
SR3.4	SENS	The vehicle shall determine its position.	SM1	BUGV-SR4.4
SR3.4.1	SENS, SW	The vehicle shall have position knowledge of +/- 0.5 ft.	SM1	BUGV-SR4.4.1

Table 2: Capstone System Requirements

Capstone System Use Cases

The following use cases outline the human interfaces our end deliverable will need to cover.

Capstone System Use Cases						
ID	**Name**	**Actor(s)**	**Description**	**Normal Flow**	**Exceptions**	**Assumptions**
1	Waypoint Interface	Operator	Input waypoint to UGV	Operator will input two waypoints which will then start the car to autonomously navigate between the two waypoints.	Kill switch is enabled.	N/A
2	Kill Switch	Operator	Hardware or software implementation to disable the vehicle.	Enabling kill switch will force vehicle to stop and cease all operations.	N/A	Kill switch has redundancy to ensure reliability.

3	Post-Mission Analysis	Customer, Tester, Operator	Log telemetry for later analysis	Telemetry will be streamed to various clients connected to the network the vehicle operates on. This real time data can be visualized in real time on these clients. Telemetry and raw operation metrics will also be stored on the vehicle for later retrieval.	N/A	Vehicle will have a communications link that will allow data to be streamed.

Table 3: Capstone System Use Cases

Design Development

Hardware

All hardware concept designs were created with the goal to empower sensor testing by providing a platform that is easily modifiable, and makes it easy to test a variety of different scenarios that the final CPSLO UGV might encounter. Additionally, a wide variety of sensors were considered to provide effective navigation data.

Plexiglass board with mounted components

The idea of a plexiglass board came up as an initial method of beginning sensor testing early without relying on the need for a prototype vehicle.

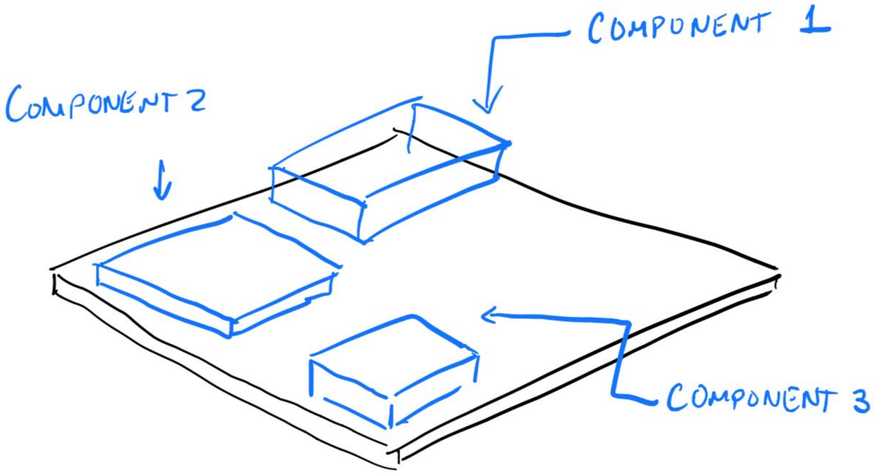


Figure 1: Plexiglass board with mounted components

RC car with mounted components

As the quarter progressed and the project details were clarified, it became evident that it would be more effective to test our peripherals while in motion to ensure that our navigation system is robust and reliable. Below are some more specific designs involving this testing rig.

1) **RC Car with ultrasonic sensor on edges:** Placed along the edges of the rc car, ultrasonic sensors can be used to send and prevent collisions with objects that our car is approaching. The limitation is that each sensor has a relatively small sensing range. In order to cover a full side, we may require 3 or even 4 sensors.

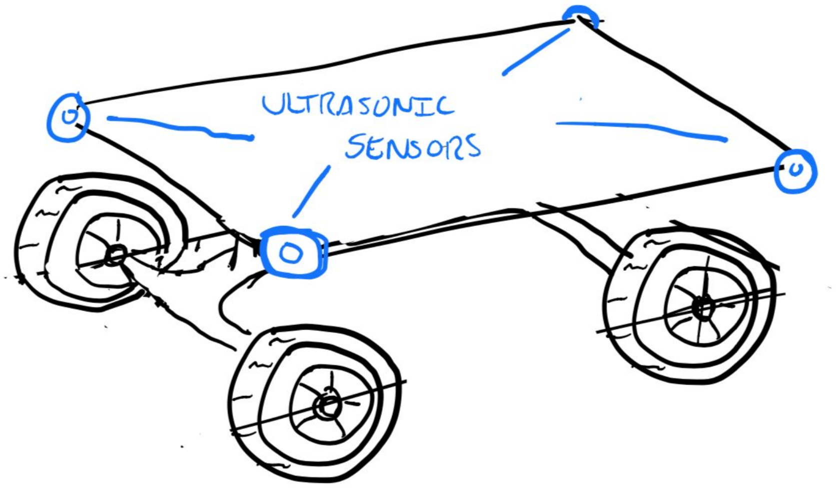


Figure 2: RC car with ultrasonic sensors

2) **RC Car with LiDAR:** Unlike the ultrasonic sensors, a LiDAR is able to get more points of coverage by spinning 360 degrees. The only downside with this design is that the LiDAR returns a 2-d plane of points, rather than a 3-d mapping. This means that our object detection would only happen at a specific height every time a full rotation is made.

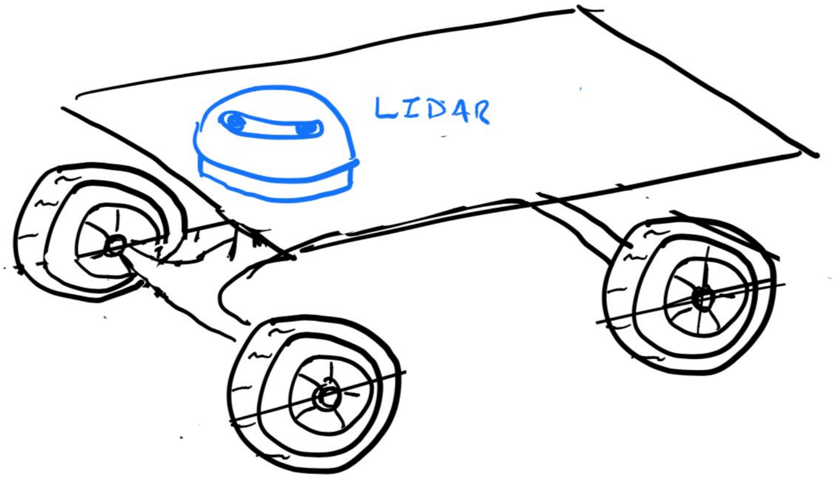


Figure 3: RC car with LiDAR

3) **RC Car with Stereoscopic Camera:** Because we will always be driving forward, it may be worth it to trade off 360 degrees of coverage for a wider field of view of data at the front of the UGV. A stereoscopic camera is able to take two images and interpolate the depth of different items in the images. The benefit is that the front would have extensive coverage. The downside is that the rest of the car would have none.

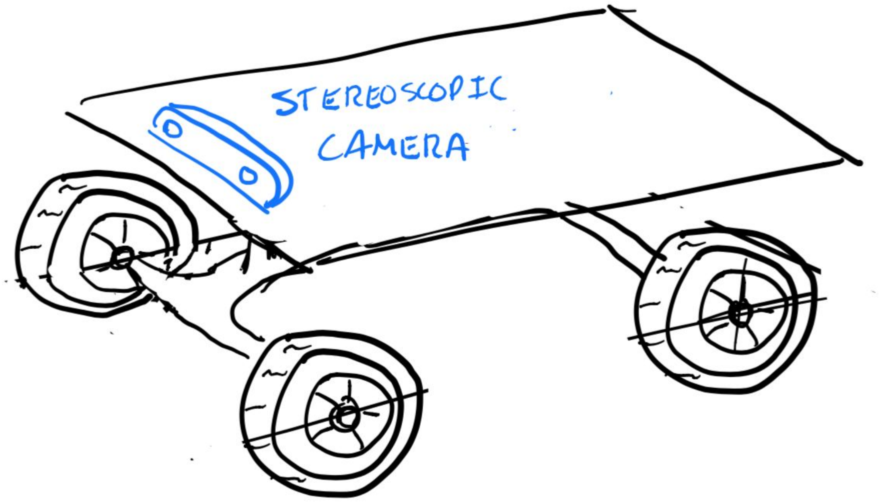


Figure 4: RC car with stereoscopic camera

In the end, we decided to stick with using just the stereoscopic camera and the LiDAR. We developed the vehicle using these two components for all of our visualization needs.

Stereoscopic Camera

We started off trying to use an Intel RealSense D435i camera. There was a lot of experimentation done in order to get the camera working with ROS, but there were compatibility issues between the camera and the computer that we were using, the Jetson Nano. As a result, we chose a different stereoscopic camera to use instead, the ZED Mini. This camera was fully supported by the Jetson Nano, and there were a ton of sample ROS nodes available for it.

Pixhawk

Partway through Winter Quarter, we learned that the NGCP software team would be using a Pixhawk in order to control the final vehicle. In order to maintain full compatibility with the future vehicle, we added one to our plexiglass board. We also needed an external GPS and compass to work with the Pixhawk, so that was also added to our board.

Software

After installing ROS onto the Jetson Nano, we began software development by trying to interface with both the ZED Mini stereoscopic camera and the LiDAR. Both of these devices had sample ROS nodes available that would launch the device and display visual information in real time using RViz. There were many sample ROS nodes available for us to further test with. One notable one was *zed_rtabmap_example*, which uses the *rtabmap_ros* package in order to generate a 3D map from the data produced by the ZED camera.

After we were confident about the compatibility of the sensors and the feasibility of our plans, we wrote our own ROS nodes to accomplish our specific tasks. We started off by setting up our *tf* node to handle the spatial transformations between the Jetson Nano, RPLiDAR, and ZED Mini camera. We modified the *zed_rtabmap_example* node in order to take in LiDAR data as well and use our *tf* node. We then worked on our node that handles the core logic for the SLO UGV. This node takes in an occupancy grid that we generated with *rtabmap* as well as odometry data from the ZED Mini. The node also implements *teb_local_planner*, which can perform local navigation based on obstacle data. From here, we were able to send commands to the Pixhawk using MAVROS, which has the ability to convert between ROS topics and MAVLink messages. MAVLink messages are the messages used by vehicles running ArduPilot, the firmware loaded onto the Pixhawk.

Final Detailed Design

Our final system design was an acrylic board with evenly spaced holes that could be used for mounting each component in a variety of configurations. The top down view in Figure 5 distinctly shows our chosen placement for each component. This design was chosen because it allows easy access to each sensor and can be quickly added or removed from the rc car for testing or coding purposes. On the acrylic itself we have mounted a 2d LiDAR, a stereoscopic camera, the Jetson Nano, the PixHawk PX4, the PMIC, and additional supporting equipment.

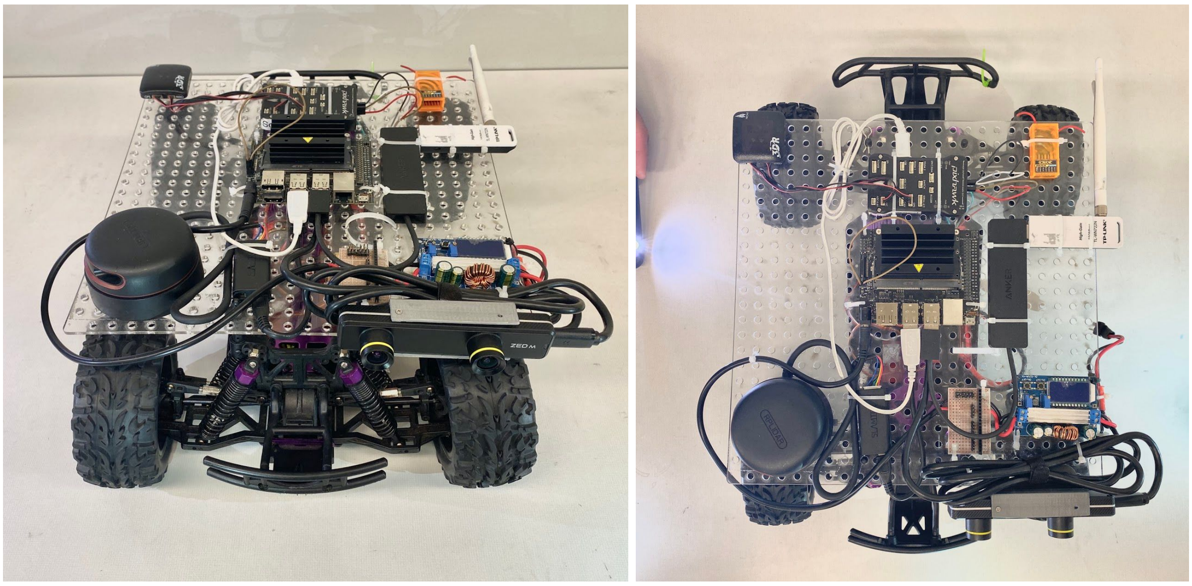


Figure 5. Angled and top down views of the vision system

The software architecture of the final system consists of a Robot Operating System (ROS) package that contains logic for tying dependencies for mavlink communication, navigation using the teb local planner, ZED mini SDK, and scan data from the RPLidar. All together, the system takes the data from the environment and a destination waypoint to generate an occupancy grid and a series of poses to traverse the occupancy grid to reach the final destination. Figure 6 shows a high level system diagram of the software architecture for our vision system as well as the external components that it will be connected to.

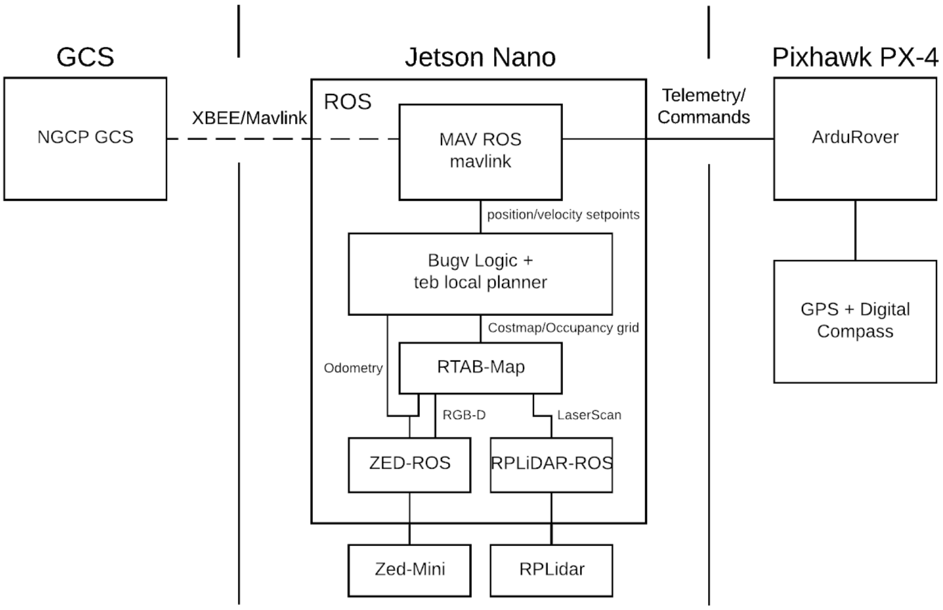


Figure 6: Software Schematic

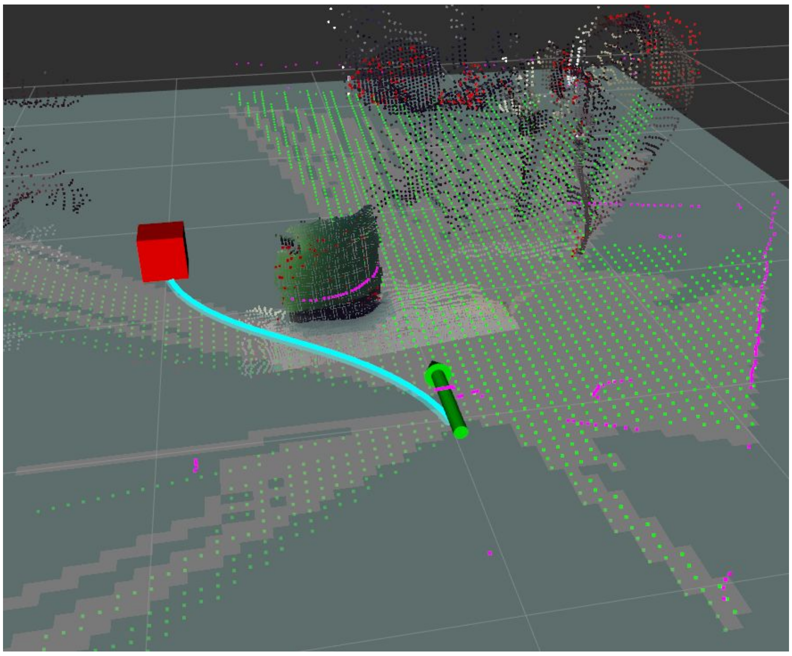


Figure 7: Arbitrary Goal Mapping and Pathing Visualization with Camera View

In the navigation system repo (https://github.com/justinnuwin/NGCP-Capstone-UGV) the core of the project lives in NGCP-Capstone-UGV/ros/src/**bugv**/. This is the ROS node for the Bugv which ties together the various technologies used in the navigation system.

The launch directory stores ROS launch configurations for the various parts of the navigation system. The rviz directory stores the ROS visualization configuration for the graphical visualization tool.

The src directory stores the source files for the navigation system. The current modules are bugv_control, bugv_nav, and bugv_logic. The control module concerns interfacing with the PixHawk such as setting it into Guided/auto mode, and commanding it to drive the vehicle. The navigation module handles mission planning and navigation. It uses the teb_local_planner to plan routes using waypoints. The logic module ties the two other modules together.

The URDF directory stores the robot definition file which contains the location and orientation of all the sensors on the robot to align the frames of robot telemetry and fuse the data.

The Bugv ROS node and its dependencies are portable to all architectures that support ROS. In our case, the testbed utilized the Nvidia Jetson Nano which runs on the aarch64 architecture. Many other embedded processors like the Raspberry Pi also support ROS and all the dependencies for running the Bugv ROS node.

System Integration & Testing

Capstone System Requirements Met

SR1 - The vehicle shall have enough power to transport the avionics package
SR1.2 - The vehicle avionics shall fit on a removable electronics tray
SR2.1 - The vehicle shall be capable of 3 mph while operating.
SR3.2.1 - The vehicle shall be capable of evaluating collisions at at least 1 Hz.
SR3.4 - The vehicle shall determine its position.
SR3.4.1 - The vehicle shall have position knowledge of +/- 0.5 ft.

Capstone System Requirement Partially Met

SR2.3 - The vehicle shall have a kill switch.
SR3 - The vehicle shall be capable of short range automated navigation via external waypoint.
SR3.1 - There shall be an interface for the user to input waypoints into the vehicle.
SR3.2 - The vehicle shall autonomously detect obstacles during waypoint navigation.

For SR2.3, a hardware killswitch has been added to the PixHawk, but for some testing purposes it has been removed. It is recommended to keep the switch attached and enabled. A software killswitch is implemented in the bugv_nav module which controls the state of the PixHawk. Several failure modes have been identified which will put the PixHawk into HOLD mode such as disconnection from the controller process’ network connection.

For SR3.1, A robust method for entering waypoints has not been established. Currently the waypoints are coded into the bugv_nav module and require the node to be recompiled when updated. Implementing a TCP or GCS listener to dynamically update this is built in functionality for the Mavros dependency hence the requirement is partially met.

See below for an explanation for SR3.2. This is tied to the lack of closed loop control.

Capstone System Requirement Not Met

SR2 - The vehicle shall drive in a controlled manner.
SR2.1.1 - The vehicle shall measure its ground speed to within +/- 0.5 mph.
SR2.2 - The vehicle shall have precise control of its speed to within +/- 0.5 mph.
SR3.3 - The vehicle shall autonomously avoid obstacles during waypoint navigation.

During the design phase of this Capstone project, we had initially planned to test our navigation closed loop with an RC car that was available to our team. During the implementation phase we quickly found that the vehicle we were using (RedCat Racing Volcano EPX 1/10 scale) would not work for our testbed since the vehicle is geared for quick acceleration and high speeds. The vehicle does not include onboard odometry either. The motor on the vehicle used was not geared for low speed or low acceleration operation which is necessary for the precise movements we would be testing the Bugv under. We worked around this difficulty by designing the navigation system to work in tandem with the control system of the vehicle. We demonstrated our system's ability to do this by driving the vehicle manually following the waypoints it set by eye. These requirements are only defined as not met since they do not encompass the original vision that they were written under, which is the idea of closed loop control of the vehicle using the telemetry and commands from the navigation system. Since there was a pivot in the approach to the project after some implementation was completed, we decided to stick with this vehicle to accomodate the approaching deadline and pushed the odometry requirements to the NGCP team since they are developing a vehicle which will support this natively in their control system.

Future Work

Tie odometry from sensor telemetry and sensor fusion into the PixHawk navigation system. This only applies if the NGCP team decides to use the PixHawk as their primary navigation controller rather than opting to use the ROS navigation stack. Some implementations that were looked at include OpenKAI and extensions to ROS.

A more robust method for entering waypoints can be developed which could tie into the NGCP Mavlink based GCS. The functionality for adding waypoints is already implemented in the bugv_nav module and the MavROS dependency. Fully flushing out the implementation requires testing with the NGCP GCS.

Obstacle avoidance and pathing could be made much more robust with further testing and a better idea of the course the robot will be navigating through.

Management Plan

Equipment that is being handed off to NGCP:

- Acrylic Board
- Jetson Nano
- RPLiDAR A2M8
- Zed-Mini
- PMIC

NGCP Equipment On Loan:

- 3DR PixHawk PX4
- 3DR GPS+Digital Compass
- Killswitch

Capstone/External Equipment:

- OrangeRX 6-Ch RC Receiver
- 4-Port USB Hub
- TL-WN722NV1 USB WiFi adapter
- 2S and 3S LiPos
- LiPo Battery Charger
- RC Car

With the equipment that is being handed off to NGCP, the supplied user guide, and GitHub repository should provide all the necessary documentation to allow the NGCP team to get up to speed and develop with the navigation system that we have designed. In addition, to facilitate the hand off, one of the members of the Capstone group will assist the NGCP team with integration in the Spring Quarter.

References

Guilherme, Affonso. “ROBOTIS-JAPAN-GIT/turtlebot3_slam_3d.” GitHub, 6 Dec. 2018, github.com/ROBOTIS-JAPAN-GIT/turtlebot3_slam_3d.

Huang, Sam. “How the Autonomous Car Works: A Technology Overview.” Medium, Medium, 25 Apr. 2018, medium.com/@thewordofsam/how-the-autonomous-car-works-a-technology-overview-5c1ac468606f.

Stewart, Jack. “Self-Driving Cars Use Crazy Amounts of Power, and It's Becoming a Problem.” *Wired*, Conde Nast, 6 Feb. 2018, www.wired.com/story/self-driving-cars-power-consumption-nvidia-chip/.

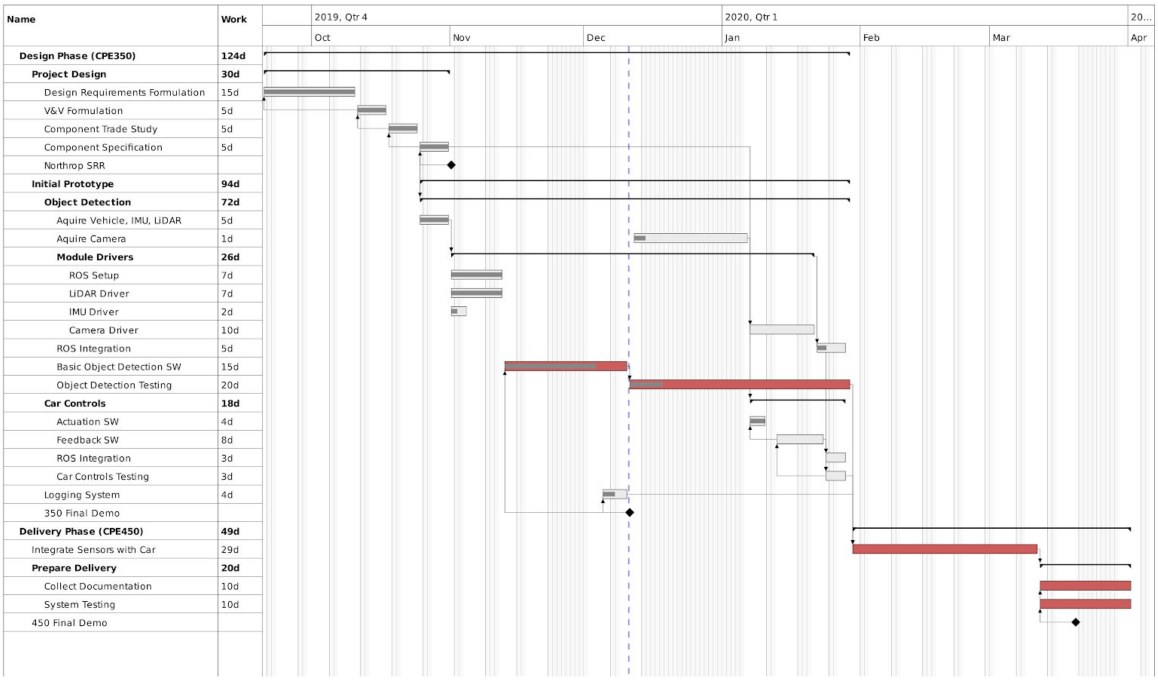
Appendix

Bill of Materials

Part Name	**Price**
Zed Mini	$449
RPLIDAR A2	$319
Jetson Nano	$99
Pixhawk	$100
Plexiglass Board	$10
3DR uBlox GPS with Compass	$90
Total Cost	$1067

Table 4: Cost Breakdown of Prototype

Gantt Chart



Name	Work
Design Phase (CPE350)	124d
Project Design	30d
Design Requirements Formulation	15d
V&V Formulation	5d
Component Trade Study	5d
Component Specification	5d
Northrop SRR	
Initial Prototype	94d
Object Detection	72d
Aquire Vehicle, IMU, LiDAR	5d
Aquire Camera	1d
Module Drivers	26d
ROS Setup	7d
LiDAR Driver	7d
IMU Driver	2d
Camera Driver	10d
ROS Integration	5d
Basic Object Detection SW	15d
Object Detection Testing	20d
Car Controls	18d
Actuation SW	4d
Feedback SW	8d
ROS Integration	3d
Car Controls Testing	3d
Logging System	4d
350 Final Demo	
Delivery Phase (CPE450)	49d
Integrate Sensors with Car	29d
Prepare Delivery	20d
Collect Documentation	10d
System Testing	10d
450 Final Demo	

Figure 8: Gantt Chart for Fall and Winter Quarter Planning

NGCP CPSLO UGV System Requirements

NGCP CPSLO UGV (BUGV) System Requirements						
ID	**Subsystem**	**Requirement**	**Meets Stakeholder Requirements**	**Verification Method**	**Verification**	**Validation**
BUGV-SR1	MECH	The vehicles shall have adequate horsepower to transport the CPP UGV.	SD1	D	Load CPSLO UGV with CPP UGV and verify control systems under constraints listed under this requirement.	The primary role of the CPSLO UGV is to transport the CPP UGV.
BUGV-SR1.1	MECH	The vehicle shall have adequate volume to house the CPP UGV.	SD1	D	Load CPSLO UGV with CPP UGV and verify fit within mechanical constraints XX.	The primary role of the CPSLO UGV is to transport the CPP UGV.
BUGV-SR1.1.1	MECH	The vehicle shall have a payload volume of at least 12x7x7in (LxWxH).	SD2	I	Physically measure payload volume and cross reference with CAD design.	The primary role of the CPSLO UGV is to transport the CPP UGV.
BUGV-SR1.2	ELEC, MECH	The vehicle avionics shall fit on a removable electronics tray	SD3	I	All electronics should reside on electronics tray during normal operation. The tray shall be removable.	This will allow quicker processing in the event of testing or electrical hardware failure/replacement.
BUGV-SR2	MECH	The vehicle shall be built to traverse obstacles and rough terrain.	SM4, SM5	D	Drive the vehicle through uneven terrain with obstacles	The UGV is intended for use in disaster scenarios

					larger than 2" tall. Determine if the vehicle was able to successfully move through.	in which terrain environments may not be the most desirable
BUGV-SR2.1	MECH	The vehicle shall have enough torque to maintain its speed up a 36% grade slope,	SM5	D	A 36% grade slope shall be built similar to operating conditions and the CPSLO UGV shall be able to traverse the slope at half and full speed at +/- XX.	The UGV would be unable to carry itself and the payload up a hill without enough torque
BUGV-SR2.2	MECH	The vehicle shall be able to maneuver over obstacles 2" in height.	SM4	D	Drive the vehicle through uneven terrain with obstacles up to 2" tall. Determine if the vehicle was able to successfully move through.	Obstacles of up to 2" need to be traversed over
BUGV-SR3	ELEC, MECH	The vehicle shall drive in a controlled manner.	SM7	D	Drive the vehicle on an obstacle course around different objects.	The UGV requires manual control
BUGV-SR3.1	ELEC, SENS	The vehicle shall be capable of 3mph while operating.	SM7	T	Test driving the vehicle at max speed and confirm avionics read the same speed.	Required per specification.
BUGV-SR3.1.1	ELEC, SENS	The vehicle shall measure its ground speed to within +/- XXmph.	SM5, SM7	T	Test driving the vehicle at max speed and confirm avionics read the same	Required in order to ensure that the UGV does not drive above

					speed.	its max speed of 3mph.
BUGV-SR3.3	ELEC, SENS	The vehicle shall have precise control of its speed to within +/- XXmph.	SM5, SM7	T	Test driving the vehicle at max speed, half speed, and quarter speed and confirm avionics read the same speed.	Required in order to ensure that the UGV does not drive above its max speed of 3mph.
BUGV-SR4	ELEC, SENS, SW	The vehicle shall be capable of automated control via waypoint navigation	SM1	T	Designate a destination for the vehicle to drive to, and allow the software to navigate itself towards the destination. Determine if the destination was reached.	Required per specification.
BUGV-SR4.1	SW	There shall be an interface for the user to input waypoints into the UGV.	SM1	I, T	Verify interface exists to input waypoints. Test that waypoints are loaded onto the UGV correctly.	The UGV will need to be given waypoints to complete the automated navigation portion of its mission.
BUGV-SR4.2	SENS, SW	The vehicle shall autonomously detect obstacles during waypoint navigation.	SM1, SM2	D	In situ test of UGV in waypoint navigation mode detecting obstacles in its path.	The UGV must intelligently avoid obstacles to arrive at its destination.
BUGV-SR4.2.1	SW	The vehicle shall be capable of evaluating collisions at atleast XX Hz.	SM1, SM2	T, A	Developmental and in situ test of vehicle evaluating collisions with	The UGV hardware must be quick enough to

					objects at the XX rate or better, theoretically allowing the UGV to navigate at half the speed of XX.	respond to the environment in real-time
BUGV-SR4.3	SW	The vehicle shall autonomously avoid obstacles during waypoint navigation.	SM1, SM2	D	In situ test of UGV in waypoint navigation mode not coming within XX of obstacles.	The UGV needs to autonomously detect and avoid obstacles
BUGV-SR4.4	SENS	The vehicle shall determine its position.	SM1	T	Measure position and move the vehicle and remeasure position.	The UGV needs to know its position for autonomous waypoint navigation
BUGV-SR4.4.1	SENS, SW	The vehicle shall have position knowledge of +/- XX ft.	SM1	T	Measure position and move the vehicle and remeasure position. Characterize performance inside and outside the uncertainty area.	Position knowledge is required for waypoint navigation.
BUGV-SR5	SW	The vehicle shall have a manual control mode.	SM3	D	Use the software to manually drive UGV through the GCS link.	The CPSLO UGV requires manual control
BUGV-SR5.1	ELEC, SW	The vehicle shall support local manual operation.	SM3	I, D	There shall be a connector on the UGV to connect a computer/controller and	The CPSLO UGV requires manual control

					manually drive the vehicle.	
BUGV-SR5.2	SW	The vehicle shall support remote manual operation.	SM3	D	Use the software to manually drive UGV through the GCS link.	The CPSLO UGV requires manual control
BUGV-SR5.2.1	SW	The vehicle shall stream a FPV video feed to the driver.	SM3	D	The software to manually drive the UGV through the GCS link shall stream FPV video as well.	In order to manually control the UGV, the driver must have an idea of the environment and possible routes.
BUGV-SR5.2.1.1	SW	The FPV feed shall have resolution of XX.	SM3	I	The FPV stream shall transmit images of XX resolution.	THE UGV requires first person view
BUGV-SR5.3	ELEC	The vehicle shall have a communications range of at least 3000ft.	SM6	T	Test data loss at different ranges and confirm that it is below some tolerable range	The UGV requires range of 3000ft

Table 5: NGCP CPSLO UGV System Requirements