GONZAGA UNIVERSITY
School of Engineering and Applied Science
Center for Engineering Design and Entrepreneurship

PROJECT PLAN
10/23/2019
Gonzaga University Autonomous Delivery Robot

Prepared by:

Reviewed by:

Faculty Project Advisor	Project Sponsor/Liaison
Design Advisory Board Member	Project Sponsor/Liaison

1 Project Overview

1.1 Project Summary

As the food industry becomes increasingly integrated with robotic automation, companies around the globe are starting to transition their delivery services into fully autonomous systems. However, when self-driving cars are utilized, certain areas inaccessible to public transportation are neglected. Our project is developing the software for a robot that will be able to safely navigate the Gonzaga University campus to deliver goods to various locations, as well as a simple web application for placing orders remotely. The heart of the Gonzaga campus is largely built without roads accessible by traditional delivery vehicles, so we are attempting to solve this problem. Our efforts will be heavily focused on integrating state-of-the-art techniques in computer vision into our robot so that the robot can recognize, categorize, and avoid obstacles without human intervention.

image depicts new Postmates Serve autonomous delivery vehicle

1.2 Project Objectives

The primary objective of our project is to create the software for a robot that could potentially provide a delivery service for Gonzaga's campus. There is currently a high demand for delivery services in heavily-populated areas, but we believe we've found an untapped market in the realm of product delivery on college campuses. If people have the option for food delivery from 1887 or the Marketplace, they can potentially have more free time to study or work on assignments than if they had walked to and from either location. Upon completion of the project, we plan to have software that is capable of running on a robot so that it will be able to deliver goods to predetermined locations on campus.

image depicts Domino's new DRU autonomous pizza delivery vehicle

We will also design a simple web application that end-users could utilize to place delivery orders for our robot remotely. Besides creating value for end-users, we would also like to make valuable contributions to the mobile robotics community by making our code open-source. While our project will be limited to functionality on Gonzaga Campus, we plan to provide adequate documentation so that our project can be expanded in future capstone projects to work with local vendors for delivery services. Our project is unique in the sense that our business objectives are important, but we are hoping to also learn more about computer vision and become part of the online robotics community.

1.3 Project Stakeholders

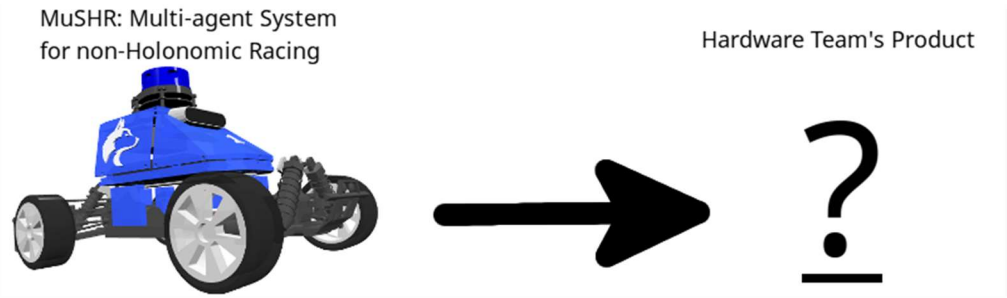
Our primary stakeholders are ourselves (the software team) and the hardware team - as we are a self-sponsored team and our success as a group in this project is very important. This project presents a unique challenge for our team, as we are responsible for deciding on an appropriate scope and MVP (minimum

viable product) for our project. This project also requires much responsibility in frequently communicating with our DAB member Ryan Hendrickson, and our faculty advisor Bruce Worobec. Both of them have agreed to help us with project management and execution.

Our other stakeholders include students and faculty at Gonzaga University that are interested in product delivery services. Many students will be indirectly affected by our product, as the robot will be traveling alongside students on Gonzaga's paths and walkways while those students are on campus. We plan to form a focus group for our project to keep lines of communication with the Gonzaga community open. This will allow us to gather the feedback necessary to provide the best product we possibly can for our end users.

1.4 Project Deliverables

Our team's primary project deliverable will be the software necessary to create an autonomous robot capable of delivering goods to various pre-determined drop-off locations on Gonzaga's campus. The hardware for the robot itself is being developed by the four-person hardware team. We are working with them to build out a robot that can successfully demo our software. To decrease our dependence on the hardware team, we are initially developing our software to work with a test platform. For this test platform, we will use MuSHR, an "open-source robotic hardware and software platform" intended for "learning and researching AI in the setting of autonomous vehicles and mobile robotics." This product was developed by a research group at the University of Washington. The MuSHR development team has agreed to donate one of their robots to us for our project. In addition to this, we will configure a robotics simulation in ROS (Robot Operating System) to make testing and iterative development easier, as well as a basic web application featuring a drop-down menu that could be used to place orders remotely. The webserver would then calculate the best path based on the robot's current location and the destination. That information would be sent to the robot wirelessly, and the robot would execute the delivery.



We also plan to create adequate documentation such that future robotics projects will be able to pick up where we left off and expand the functionality of our bot. Our documentation will be hosted on our team website, which is currently accessible at https://guadr.github.io/. There is currently little content on our site, but we plan to expand it as our project progresses. Our website will contain a high-level description of the project such that an end-user could read it and understand what our project will accomplish. We will also provide detailed documentation of our software that fully describes all of the code and low-level processes the system executes. All of our code will be hosted on Github as an open-source project.

1.5 Project Scope

For the scope of our project, the external aspect of this project is the customer that is placing an order with the web application. The simple web application will take in a location and the delivery endpoint from the user, and communicates this to the server. The server then plans the trip and forwards the optimal path to the robot. The bot then goes through the process of delivering to the specified location. As you can see, the majority of interactions take place between the web app, web server and robot.

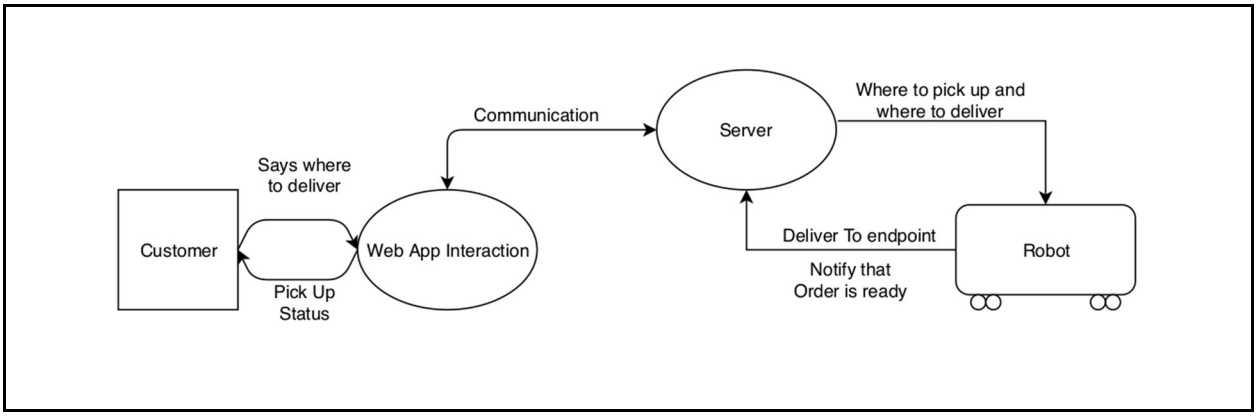


Figure 1: Context diagram showing user interaction through deliverable.

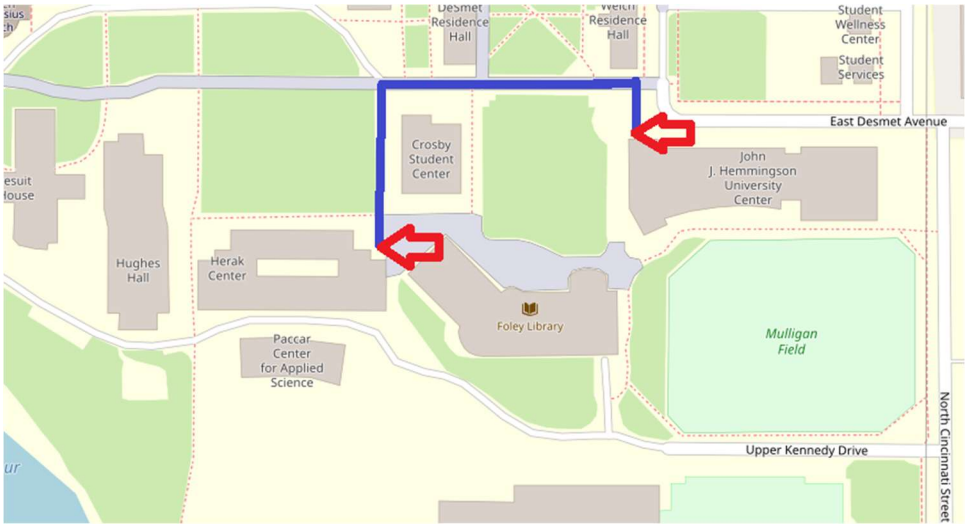


Figure 2: High level route planning on Gonzaga's campus

1.6 Related Work

In this section, we will discuss the systems that are similar to our Delivery Robot. These systems include the Amazon Scout, the Starship delivery robot and Nuro. These related systems have many similarities and differences with our robot, we will analyze these systems in relation to ours.

image depicts Amazon's new Scout, an autonomous delivery vehicle planned to be used for deliveries in urban and suburban regions

The Amazon Scout delivers Amazon orders to houses in non-rural areas. The Scout has six wheels, runs off of an electric battery and moves through suburban and urban areas at a walking pace. Based on images provided by Amazon, the robot appears to have four or more sensors in the front, as well as fog lights and brake lights. The Scout is currently in its testing phase, and Amazon plans to have human supervisors making rounds to monitor the success of the Scouts. Amazon has mentioned rolling these out on college campuses, likely due to their population density and the fact that campuses are generally rather compact. This is part of our rationale for choosing to test our project only on Gonzaga's campus, and this appears to be a good call based on the similar plans made by Amazon. The Scout is based on deliveries from Amazon, which are mostly packages that are ordered online. Our product differs from theirs in the sense that is focused on campus deliveries from campus-based stores.

image depicts Starship, an autonomous delivery robot created by Starship Technologies

The most similar existing system is from Starship Technologies. They have a small robot, currently just dubbed Starship, which is built around delivering food and packages. Based on images provided on the official Starship website, it appears very similar in design to Amazon's Scout. Starship Technologies recently have announced that they want to start rolling out the robot on campuses, which would be interesting if they rolled it out to Gonzaga. Something that separates our project from Starship's is that we are using the campus as a training area to get an MVP, but our end goal is to target places such as golf courses. It seems as if they focus on cities and urban driving, where our robot would eventually want to target specific markets that are outside of that realm. This is the main competitor in the area, and as they are expanding into new markets, we are keeping up with information about them and how to differentiate our product them theirs.

Another existing system that relates to our project is Nuro. Nuro provides a look at some of the challenges that our project could face if we look at extending the scope to include areas with streets and cars.

Nuro has partnered with the grocery chain Kroger to deliver groceries directly to people's houses. The vehicle is capable of driving on roads alongside cars and goes up to 25 miles per hour. This is a very different project from ours because it is a large vehicle which is designed for urban driving. However, it could provide us with lessons about how to deal with other vehicles which we might run into on Gonzaga's campus.

image depicts Nuro's new self-titled autonomous delivery vehicle

When comparing the Gonzaga University Autonomous Delivery Robot with the above projects we have to keep in mind that most of these companies and their projects are still in their development phase. In this way, they are continuously evolving, and we want to keep up to date with what these companies change with their product. All three projects are intended to reduce the need for human intervention in delivery services by designing a robot that can autonomously navigate and deliver small items in an urban environment. The major difference is that the scale of our project is much smaller. In terms of scale, our project will focus on delivering packages within a small area (Gonzaga University Campus). Amazon Scout, Starship, and Nuro are all intended to be used in both suburban and urban areas to deliver larger packages that were ordered through an online shopping service.

2 Project Requirements

2.1 Major Features

For a viable product we need to build a delivery system that is able to complete deliveries safely and efficiently. These are the main criteria for which we have modeled our major features. The user will choose from a list of locations on a website that will communicate with the server and down to the bot. Through these criteria, we have narrowed down our list to the features in Table 1.

Table 1: Major Features

Feature	*Description*
User can specify a delivery endpoint	Users are able to choose an endpoint from a menu on the server.
Robot can deliver items to a user specified location on campus	The robot is physically able to navigate to the location that is chosen by the user.
Robot has safety features	The robot includes protocols and fail safes to address safety concerns.
User can place an order on a simple web application	Simple web page with a drop-down list of locations where a user is able to pick the location for the bot to deliver to.

2.2 Initial Product Backlog

Table 2: Initial Product Backlog

Requirement	Description	Major Feature	Priority	Estimation (hours)
Robot - Safety	This vehicle itself will be a safety-critical-system. This means that we need to be well-informed by safety guidelines relevant to our project. Acceptance Criteria: Provide informative write-ups of MISRA C++, JAUS, and other safety critical vehicle/robotics standards to inform teammates. The product should roughly follow these safety guidelines.	Robot has safety features	1	10
Robot - Order parts	The first thing we need to do is order the parts for our test robot. Acceptance Criteria: Received all parts for the MuSHR test car.	Robot can deliver items to a user specified location on campus	2	2
Focus Group	Create a focus group from potential user base on campus. Acceptance Criteria: Hold focus group session, ask pointed questions for feedback, buy pizza.	Robot has safety features	3	4
Software Development Workflow	Standardize a workflow that will facilitate easy collaboration before any serious software development. Acceptance Criteria: Git server configuration, git commit webhooks, functioning continuous integration	Robot can deliver items to a user specified location on campus	4	5
Test Suite - Configuration	Because of the safety-critical nature of the project we will have to have tests in place. Acceptance Criteria: Configure test suite, automatic testing - gtest for C++, pytest for Python.	Robot has safety features	5	5
Robot - Assembly	Before any serious testing or development, the test robot will need to be properly assembled. Acceptance Criteria: Assembly of the robot is complete.	Robot can deliver items to a user specified location on campus	6	8
Robot - Install & Configure	Once we either receive a MuSHR donated car, the requisite software will need to be installed and configured. Acceptance Criteria: MuSHR is fully functional, test robot can be controlled via teleoperation.	Robot can deliver items to a user specified location on campus	7	10
Website – Creation	Creation and configuration of our simple web page. Acceptance Criteria: Website up and running	User can place an order on a simple webapp	8	8

Website – Functionality	Creation of a drop-down menu and communication with the web server to give directions to the bot. Acceptance criteria: Drop down menu working and communicates choice to the web server	User can place an order on a simple webapp	9	10
Robot - Simulation	Configure a simulation to work with our robot and minimize sim-to-real overhead. Acceptance Criteria: Gazebo is configured to have a working simulation for development	Robot can deliver items to a user specified location on campus	10	25
Web server - Server to Bot	Configure web server to send high-level instructions to the robot. Acceptance Criteria: Web server sends order instructions to robot.	User can specify a delivery endpoint	11	14
Web server - Bot to Server	Configure robot to respond to web server navigation instructions and to send back to web server status updates. Acceptance Criteria: Robot can send GPS data back to the controlling web server and process navigation requests.	User can specify a delivery endpoint; Robot can deliver items to a user specified location on campus	12	14
Robot - Time-stamping	We'll need to verify that our sensor input is properly time-stamped and insync. Acceptance Criteria: Sensor data is synced on the jetson according to an on-board real time clock.	Robot can deliver items from a store to a user	13	4
Robot - Sensor Calibration	We need to calibrate all of the onboard sensors to decrease errors. Acceptance Criteria: Sensors are calibrated and have limited noise.	Robot can deliver items to a user specified location on campus	14	10
Robot - Top-down Map	Use Google Maps to make a top down map of campus. Acceptance Criteria: Traced out Google Maps tiles of campus.	Robot can deliver items to a user specified location on campus; User can specify a delivery endpoint	15	5
Robot - Object Detection/Collision Avoidance	Robot can detect people/objects that have entered its path and decide on how to best avoid a collision	Robot has safety features	16	20
Robot - Waypoint Navigation	Robot can follow GPS waypoint navigation and can navigate continuously between start and end points. Acceptance criteria: Robot is calibrated and configured to work with with ros::waypoint_nav.	Robot can deliver items to a user specified location on campus	17	15
Web server - Logistics	Acceptance Criteria: Given order details, the web server can prepare correct logistics data to send the robot.	Robot can deliver items to a user specified location on campus	18	12

Test Suite - Unit Tests	Acceptance Criteria: Tests for all the software we create, maximized code coverage.	Robot has safety features	19	20
Robot - Test to Prod	Acceptance Criteria: Transfer sensors, onboard computer, and storage to the test bot. Re-calibrate sensors for the new bot.	Robot can deliver items to a user specified location on campus	20	5
Robot - Perception	Acceptance Criteria: Train models for perception	Robot can deliver items to a user specified location on campus	21	30
Robot - Planning	Acceptance Criteria: Train models for planning	Robot can deliver items to a user specified location on campus	22	30
Robot - Localization	Acceptance Criteria: Create our own SLAM implementation	Robot can deliver items to a user specified location on campus	23	30

2.3 Additional Features

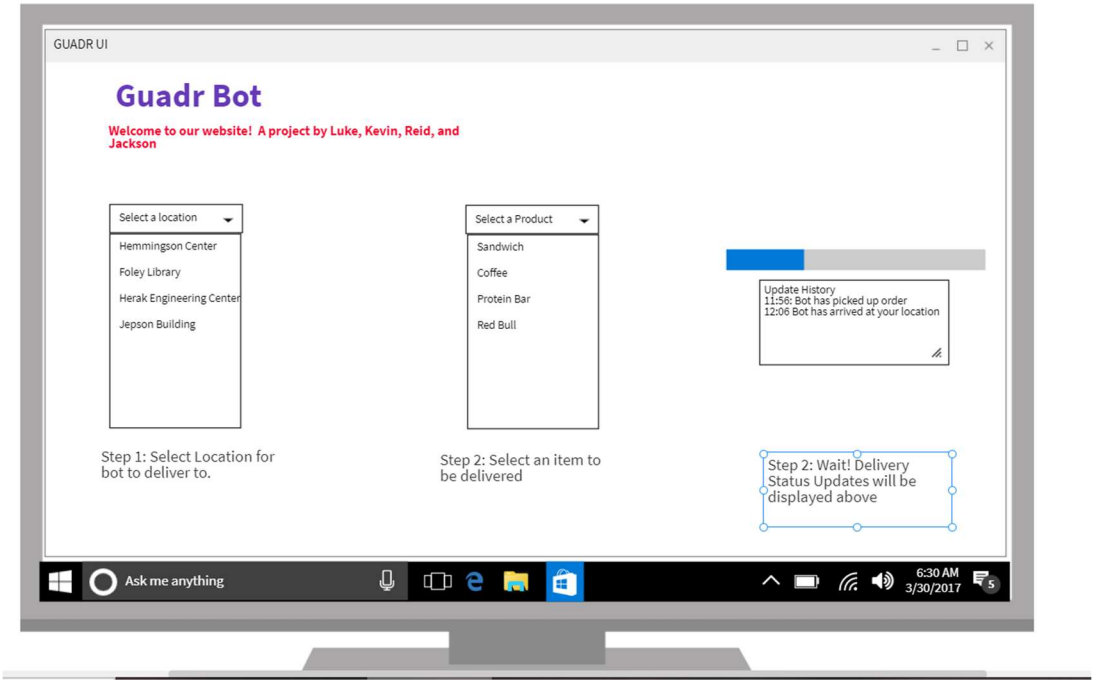
Feature	Characteristic	Constraint
Creating custom delivery locations	Allowing users to specify a location by dropping a pin on a map in our mobile application rather that selecting from a list of pre-determined delivery locations	Time constraint, software capabilities, requirement for highly-accurate GPS functionality.
Implementing advanced collision avoidance and street-capabilities	Introducing collision avoidance for paths that might include cars and busy intersections. Ability to drive alongside a crowded road with a high likelihood that no damage will be done to the robot, as well as a guarantee that the robot will not collide with any pedestrians or bikers.	Time and budget constraints, as there would be a need for expensive advanced sensors, more complex programming, and quicker response time from robot. Challenges include detecting drop-offs along sidewalks and navigating through a higher concentration of motor vehicles such as cars, bikes, and Lime scooters.

Securing item during transmission with lockbox and key-generating software	Giving users peace of mind by locking their items during transit using a lockbox that can only be opened with a randomly generated key. This would be a 4-digit passkey given to the user upon delivery, and the user would enter the passkey into a 10-digit keypad similar to what you would see on an older cell phone.	Time and budget constraints. Requirement for advanced security mechanisms, hardware requirements outside of scope of project, requires communication between lockbox, robot, central system, and user's mobile device.
Displaying live location of robot to user	Giving users an accurate idea of where on campus the robot is at any given time by displaying its location on a map	Time and hardware constraints, as providing an accurate location of the robot would require extra overhead when sending data to and from the web server
App	Giving users an application to interface with the bot. Interaction between the application and the robot through the server. • Order processing • Login • Registration • Store selection • Menus • Checkout • Tracking • Payment • Authentication • Order processing	Time is the largest factor here. The app is the first stretch goal we will work on once our MVP is finished.

3 Design Considerations

3.1 Initial User Interface Design

Our initial user interface for the web application is going to be a simple website that allows the user to select from a list of preset locations for the robot to deliver to. There will also be a dropdown menu for product selection. Once the order is placed the website will display the delivery progress and the updates sent back from the robot.



3.2 Initial Software Architecture

We describe the three major components of the minimum viable product we plan to build in Figure 2 below. It shows the interaction between the server, web application, and robot. The web application for our MVP will be simple. The backend of the web application will send the web server order details. When the web server receives order details, it will add the given order to a queue. The web server will send delivery instructions to the robot for each order one at a time. When the robot starts its delivery, it will continuously update its location to the web server. The web server will then maintain the progress report for the user application.

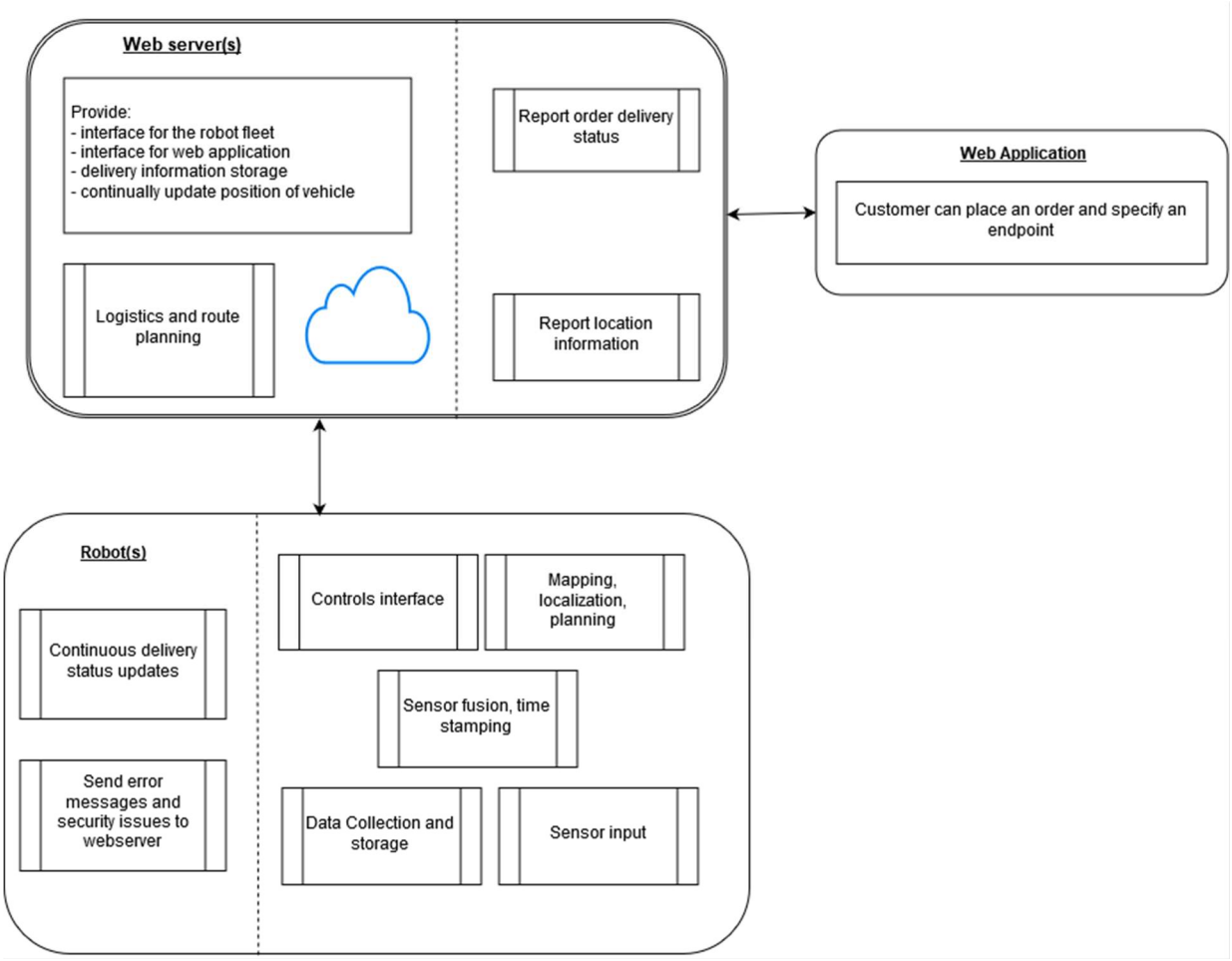


Figure 2: High-level component overview of MVP.

Figure 3 shows the component architecture of our robot concerning how sensor input data is processed. The sensors that will make up our sensor suite will all be loaded and synced onto the Jetson. This will require us to timestamp and align the data. Once we have the data synced, we perform localization, point mapping, object detection and object classification which together will inform the robots probabilistic path planning and controls algorithms. Every so often we will also have the robot communicate its GPS coordinates to the webserver. If the robot ever loses connection to the web server, we will either have the robot fail-safe and stop operating completely or have the robot return to the location it was deployed from.

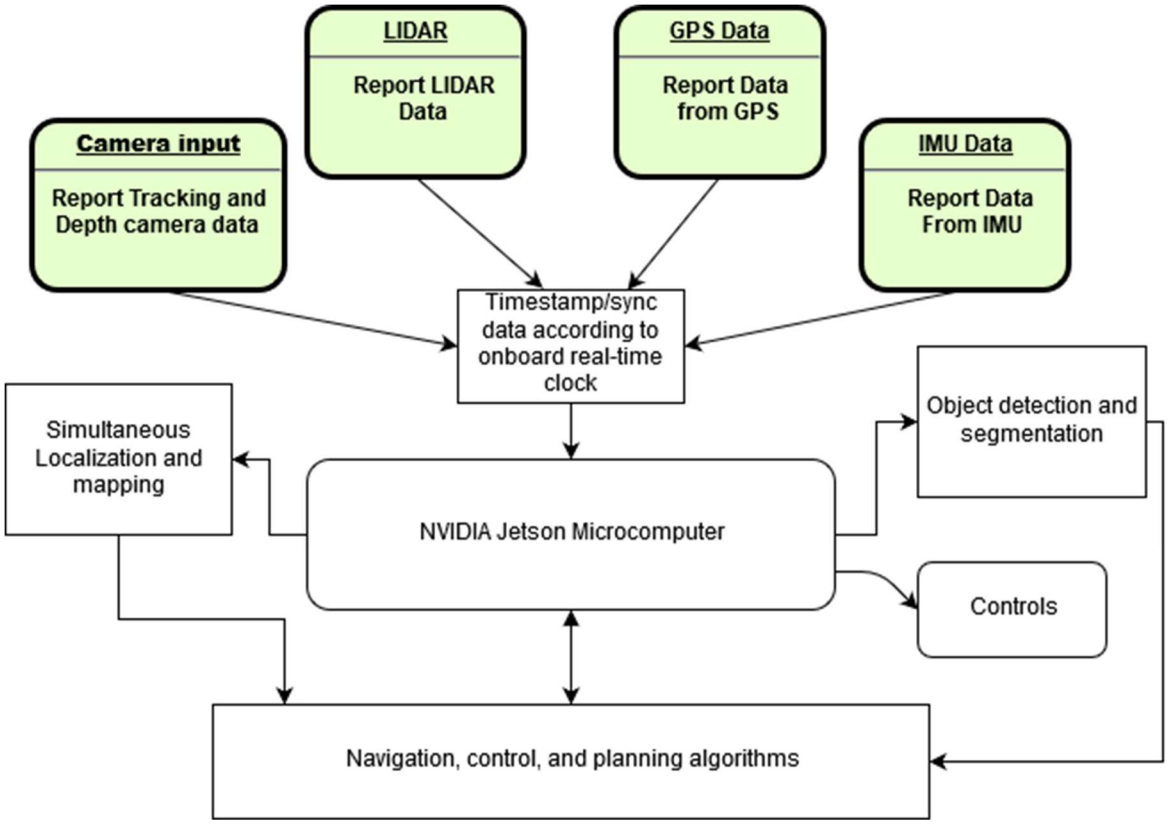


Figure 3: Data from sensor components are synced onto a controller and the forwarded to the NVIDIA Jetson. The Jetson then computes the best relative path given the sensor data.

3.3 Initial Software Test Plan

Our project is unique in the sense that we are going to have to do testing for both the hardware and software sides for our system. For hardware, the testing will be very granular, as we need to make sure that each piece is communicating effectively and accurately. First, we will need to order the parts and receive them before we can do any testing. Soon after that, we can start working on the integration between the systems, such as getting the camera and our Jetson Nano to communicate through ROS (robotics operating system). We will have to do testing for each system and its interaction with the computer, making sure that it works and doesn't mess up anything else that has been already tested. Once we have worked making these systems interact in the correct way - which will be about a month after receiving the parts - we can start working on implementing out of the box solutions for object detection, object segmentation and SLAM. Some of these – such as some SLAM techniques - have ROS nodes that should allow for easy implementation. In addition to this, we will have to work with how the robot will be communicating with the server and receiving directions for where to go and when to go. This will be something that we will work on once we can get the application communicating with the server. Near early December we hope to have all of the systems working together and communication between the robot and the application.

4 Project Risks

Due to the condensed project timeline and increased level of self-monitoring that our team will require without the guidance of a sponsor, our project will inherently have increased risk. The first major risk of our project will come from any difficulties that arise from getting approval for ordering parts and the time it will take to receive our order. Any delays with the ordering process could inhibit our ability to collaborate with the hardware side of the project. To prevent this issue, we understand what parts we need and will order so that we can begin the process ahead of time. We also applied for a free vehicle from MUSHR.io, which will help with the testing and implementation of our system once we can get these parts. If the hardware components are taking too long to deliver, we can work with a backup system and order different parts online. With adequate planning, we can mitigate the risks associated with the ordering process.

Another risk that is associated with our project is that our robot needs to be safe on campus and not hit or hurt people. To prevent this, we will always caution towards a hard stop if anything is coming too close to the vehicle. We will monitor this by building it into the system and watching how often it happens, and if it is happening very often, figure out a way to avoid it from occurring so often, such as changing the path the robot takes from point A to point B. We will also research regulations and guidelines on autonomous vehicles and ensure compliance with established guidelines to create our robot safely. We will do extensive testing with ourselves before putting it on campus with other people, and if we run into safety issues after initial product rollout, we can take steps such as expanding the distance for the hard stop or creating safer routes that avoid high-density pedestrian areas.

Since our project works together with a hardware team, we have a risk of the other team not completing their side of the project on time. To prevent this, we have established a communications system for us all to keep in contact about updates about each project and make sure we aren't hiding anything from the other team. If each team knows were the other one is at, this will help design the workflow of the other project. We will monitor this by making sure to keep talking with each other and having updates about each other's project bi-weekly.

5 Initial Product Release Plan

5.1 Major Milestones

Table 3: Major Milestones

Milestone	*Description*	*Target Completion Date*
Receive parts	Brief summary describing what will be accomplished in the milestone and the significance (as appropriate).	Last week of October
Integration of camera with on board system and web app completed	This milestone will be the integration of the cameras with our Jetson Nano where we are able to see video stream from the bot. This is a major milestone as it is the crucial element of our robot.	First week of November

Ability to specify location through server and communicate to robot	The ability to specify the location for the robot to travel to through the server enables us to have user specified locations which is a key part of the functionality of our system	Fourth week of November
Bot able to move to navigate to communicated location autonomously	Bot can wirelessly receive a coordinate and autonomously navigate to that position	Second week of January
Begin integration with the hardware team	Once the hardware team is finished with printing and getting their system finished, we will start integration between the two teams and getting the system set up for final use.	First week of February
Finish integration with the hardware team	Finish the integration so that our final project is near finished.	Third week of February
Finish testing of new robot for delivery between locations A and B and safety.	We will go through the steps for testing that we went through for our test bot for safety and delivery options.	Beginning of March.
Start implementation of additional features	If all above is completed, we will start working on integration of additional features.	Second week of March.

5.2 Initial Sprint Releases

Table 4: Sprint Release Plan

Sprint Date	*Sprint Goal*	*Backlog*	*What we will demo*
4th Week in Oct to 1st week in Nov	Receive parts, physically assemble prototype bot, integrate camera with onboard system Form focus group Design software development workflow plan	Robot – Assembly, Install and Configure Focus Group Software Development Workflow	MuSHR robot can be driven using basic hardcoded command, camera view can be displayed on computer monitor Focus group feedback can be presented in a slideshow Outline of software development workflow plan can be shown

2nd Week in Nov to 3rd Week in Nov	Creation of a basic web UI Creation of a drop-down menu and communication between the website and webserver Configure a simulation to work with our robot and minimize sim-to-real overhead	Robot Simulation Website Creation Website Functionality	Our basic UI can be seen accessed and has a drop down menu. The website can communicate with the web server. We can show our simulation running as a live demo.
4th Week in Nov to 1st Week in Dec	Configure web server to send high-level instructions to the robot Configure robot to respond to web server navigation instructions and to send back to web server status updates	Web Server: Bot to Server Web Server: Server to Bot	Demonstrate robot responding to received instructions and display the updates sent by the robot back to the webserver.
2nd Week in January to 4th Week in January	Implement autonomous driving ability using input from robot's sensors	Robot - Object Detection/Collision Avoidance, Time-stamping	Robot can navigate around dynamic obstacles
1st week in February to 3rd week in February	Implement GPS functionality and waypoint navigation	Robot – Waypoint Navigation, Top-down Map, Perception, Planning, Localization	Robot can navigate to and from various GPS waypoints such that it avoids running into obstacles
4th Week in February to 2nd Week in March	Complete web server implementation, complete safety write-ups of MISRA C++ and JAUS, Integrate application with web server	Web server – Logistics, Buyer to Seller, Order Logging Robot – Safety Write-Ups	Full web server functionality can be displayed Safety write-ups can be shown
3rd Week in March to 1st Week in April	Integrate all software components with hardware team	Fusing all Robot based backlog items with final bot	Full project should be mostly finished, but testing still needs to be done so bugs may exist
2nd Week in April To 4th Week in April	Runs tests on all aspects of software to ensure everything works	Test Suite – Unit Tests	Full project should be ready to present at the end of this sprint

6 Maintenance Considerations

No group has been identified to provide maintenance of the system aside from us (the software team) and the hardware team. If we are to hand off the project to a future senior design team or if we continue to work on it, we must maintain several aspects of the project to keep it functional and up-to-date with modern field standards.

First of all, hardware must be kept up to date by updating drivers and altering code to accommodate new functionality. Some expertise would be required for this aspect of the maintenance process, as someone may need to edit and test the source code for the robot. It is hard to provide thorough documentation for this, as we would need to know what functionality certain updates implement before their release. The individual performing maintenance on this side of the project would need knowledge of this subject. Secondly, the web server and web application would need to stay operational for the robot to continue performing, and the robot would need to maintain communication with the web server for orders to be placed and executed. This would potentially require a higher level of expertise, but all information and knowledge required for this maintenance would be provided in our documentation.

Specifications for what needs to be done in terms of maintenance, as well as how to do it, will be outlined in our documentation. This documentation will be hosted on our team website, which is currently accessible at guadr.github.io. This is an important part of our project, as we plan to make all aspects of our project open-source such that someone with an interest in robotics could build a replica of our robot and run our code on their robot using the instructions laid out in our documentation.

7 Project Management Considerations

We are a self-sponsored group, so rather than meeting with our sponsor, we meet at least twice a week as a team to work on the project. This usually takes place Saturday and Sunday afternoons, and we either meet in Herak, the Foley library, or at the house of a team member. We meet with our faculty advisor (Bruce Worobec) once a week to discuss the progress of our project and to get feedback on our work. We have stayed in contact with our DAB through email and phone, and we will continue to contact him as needed for advice and support. Also, we plan to set up a focus group that we will maintain contact with for feedback regarding the delivery robot and the website we are releasing as part of our project.

For our roles within the team, we've decided to break the project up into two major parts: webserver configuration and robotics controls. Kevin Hance and Jackson Paris will be working on robotics controls. This involves handling sensor data and designing the robot's software to make decisions about where to go based on various avenues of external input. Reid Whitson and Luke Hartman will be focusing on webserver development and configuration as well as facilitating communication between the robot and the server. This would involve building a webserver that will provide route planning for the robot based on the destination, as well as collaborating with Kevin and Jackson to facilitate OTA (over-the-air) communication. This process allows the robot to connect to the internet through the webserver for instructions on where to go. In addition to webserver configuration, Luke and Reid will be in charge of the design process for the web application we plan to host on our website for placing orders remotely.

To stay in communication, we've primarily been using Slack and basic SMS messaging. This allows us to stay in close contact with each other, as we all check our phones for Slack and SMS messages more

frequently than we check our email. We keep track of our progress through an excel spreadsheet on SharePoint, and we keep track of our future sprints through a project backlog on Google Sheets. Our methods of communication have worked well for us thus far, but we will make changes as needed to improve collaboration.

Appendix

Photograph Bibliography for 1.1-1.4

- *https://youtu.be/pckZFC_hs50 - source for Postmates Serve robot image (screen captured from video)*
- *https://zdnet1.cbsistatic.com/hub/i/r/2016/04/12/a55c3558-51c6-469f-983a-01bcf53fc09b/resize/770xaut*
- *o/d6280275deb2f58f728d97c64918fd75/dominos-dru.jpg - source for Domino's DRU autonomous robot*
- *https://www.istockphoto.com/photo/business-people-discussion-working-concept-gm855198446-140921015 - source for stock business meeting image*
  https://upload.wikimedia.org/wikipedia/commons/b/b8/Cart_pushing_rviz_holonomic.jpg - source for image depicting ROS data visualization

Photograph Bibliography for 1.5

- *https://www.openstreetmap.org/#map=17/47.66679/-117.40250*

Photograph Bibliography for 1.6

- *https://cdn.geekwire.com/wp-content/uploads/2019/01/amazon-scout-630x420.jpg - source for Amazon Scout autonomous delivery robot image*
- *https://cdn.vox-cdn.com/thumbor/1KkEnmp9x_SXoIW6yJM0ogCeIjY=/0x0:2362x1641/920x613/filters:focal(993x633:1369x1009):format(webp)/cdn.vox-cdn.com/uploads/chorus_image/image/54390867/Delivery_robot_3.0.jpg - source for Starship autonomous delivery robot image*
- *https://icdn2.digitaltrends.com/image/digitaltrends/a025_c013_0612s1-2.jpg - source for Nuro autonomous delivery robot image*

Resources used for 1.6

- https://www.washingtonpost.com/technology/2018/12/19/tired-going-grocery-store-arizona-robot-driven-car-will-deliver-groceries-your-home/
- https://blog.aboutamazon.com/transportation/meet-sco
- https://www.theverge.com/2019/1/23/18194566/amazon-scout-autonomous-six-wheeled-delivery-robot
- https://www.starship.xyz/