Robocart

System Design for the First-Generation Autonomous Golf Cart

Submitted by

Elizabeth A. Miller
Robotics Engineering

Advised by

Professor Alexander Wyglinski, *Advisor-of-Record*
Professor Taskin Padir, *Co-Advisor*

September 2014–March 2015
Project Code: MQP-AW1-PATH

A Major Qualifying Project Submitted to the Faculty of Worcester Polytechnic Institute in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science.

Abstract

Inspired by ongoing research and continuous developments in autonomous vehicles, the Robocart MQP focuses on the system development for a first-generation autonomous golf cart vehicle and wireless system server. By creating the foundation for a modular and interdisciplinary system, visualization software and mechanisms can be intuitively integrated. The end result of this project is a better understanding of the efficiency of each subsystems against the real-time challenges required for an autonomous, wireless, and vision-based system. In conclusion of this project, recommendations in mechanical, electrical, and algorithm development were formed to promote further research and enhance rider usability.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ALV Autonomous Land Vehicle

API Application Program Interface

CAD Computer Aided Design

CAM Computer-Aided Manufacturing

CMU Carnegie Mellon University

CNC Computer Numerical Control

DARPA Defense Advanced Research Projects Agency

ECE Electrical and Computer Engineering

GPIO General Purpose Inputs and Outputs

GPS Global Positioning System

GUI Graphical User Interface

IMU Inertial Measurement Unit

IN-DASH Intuitive Dashboard

LIDAR Light Image Detection and Ranging

MQP Major Qualifying Project

PWM Pulse Width Modulation

RALPH Rapidly Adapting Lateral Position Handler

RAM Random Access Memory

RBE Robotics Engineering

RC Remote Controlled

ROS Robot Operating System

RXD Receive Data

SIFT Scale Invariant Feature Transform

SSH Secure Shell

TXD Transmit Data

URDF Universal Robotic Description Format

USB Universal Serial Bus

VIAC VisLab's Intercontinental Autonomous Challenge

VPN Virtual Private Network

WPI Worcester Polytechnic Institute

Acknowledgements

The Robocart Team would like to acknowledge the following individuals for providing support and assistance throughout the entirety of our project. Without all of you, this project would not have been possible.

To our Advisor, Professor Alexander Wyglinski: *Thank you for meeting with us weekly and guiding us in the right direction for the duration of this MQP.*

To Professor Ken Stafford: *Thank you for helping us determine logistics for storing our wireless server within the Rec Center Loading Dock.*

To Meredith Merchant: *Thank you for providing us with access and granting us space in the Rec Center Loading Dock to work on our MQP.*

To Tracey Coetzee: *Thank you for placing part orders for the Robocart MQP in a timely manner.*

To Prateek Sahay: *Thank you for creating the CAD model of the Robocart and working with us to develop mechanical systems for the first-generation autonomous golf cart.*

Executive Summary

Motivation

The promise of saving 1.2 million lives a year [10] and solving traffic congestion problems has struck a chord with scientists, engineers, and programmers around the world. As result, autonomous vehicles are expected to contribute to roughly half of the total cars produced by the early 2030s [11]. Starting with the development of basic cruise control in 1948, the Robocart MQP aims to convert a 1999 club golf cart into an autonomous vehicle [1]. Specifically, this project has focused on system development: wireless communication between the (mobile) autonomous golf cart and separate (stationary) server and subsystem deployment to form the necessary platform for future integration of vision and mechanical systems.

Proposed Design

The proposed design created in this MQP for developing the Robocart is split into two parallel branches: one devoted to software and vision system development (by Gabriel Isko) and another devoted to design and implementation of hardware–steering and braking–systems (by Prateek Sahay). Figure 1 is a block diagram, which shows how both of these branches interact with and complement each other.

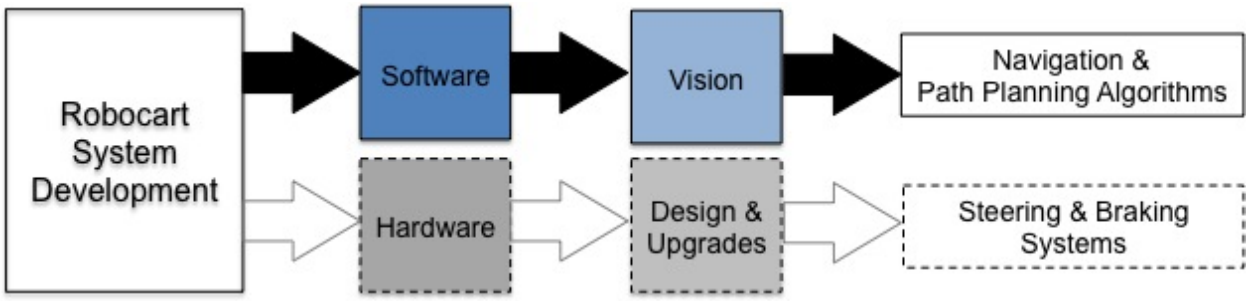


Figure 1: There are two MQPs working in parallel to develop the Robocart. This report focuses on the system-wide development. As shown by the block diagram, the Robocart system development can be broken down into two parallel tracks: software and hardware. Upon completion of each parallel branch of the Robocart system, the mechanical modifications (dashed, gray hardware blocks) will interface directly with the software and vision system (solid, blue blocks).

Key objectives were developed for this project from brainstorming sessions and are given as follows:

> **Objective #1:** *To develop the software framework and system configuration for an autonomous ground vehicle.*
>
> **Objective #2:** *To provide an Application Program Interface (API) for intuitive integration of software packages with hardware components.*
>
> **Objective #3:** *To promote forward-thinking and drive holistic development by providing recommendations for future Robocart projects.*

Once we determined these objectives, we created milestones and tasks for each teammate to implement. As the project progressed, each teammate branched off into a research area of interest: Elizabeth worked on system designs and integration; Gabriel was responsible for software development; and Prateek developed and implemented mechanisms. More about the project logistics can be found in Chapter 3.

Implementation and Experimental Results

I focused directly on developing the Robocart system architecture and creating plans for interfacing software with mechanisms. An MQP by Gabriel Isko is focused on developing the vision system for the Robocart server, and an MQP by Prateek Sahay is focused on implementing mechanical upgrades to the Robocart vehicle. I have made considerations to ensure compatibility between both the software and the actual hardware installed on the Robocart. The System Diagram for the Robocart is presented within Figure 2.

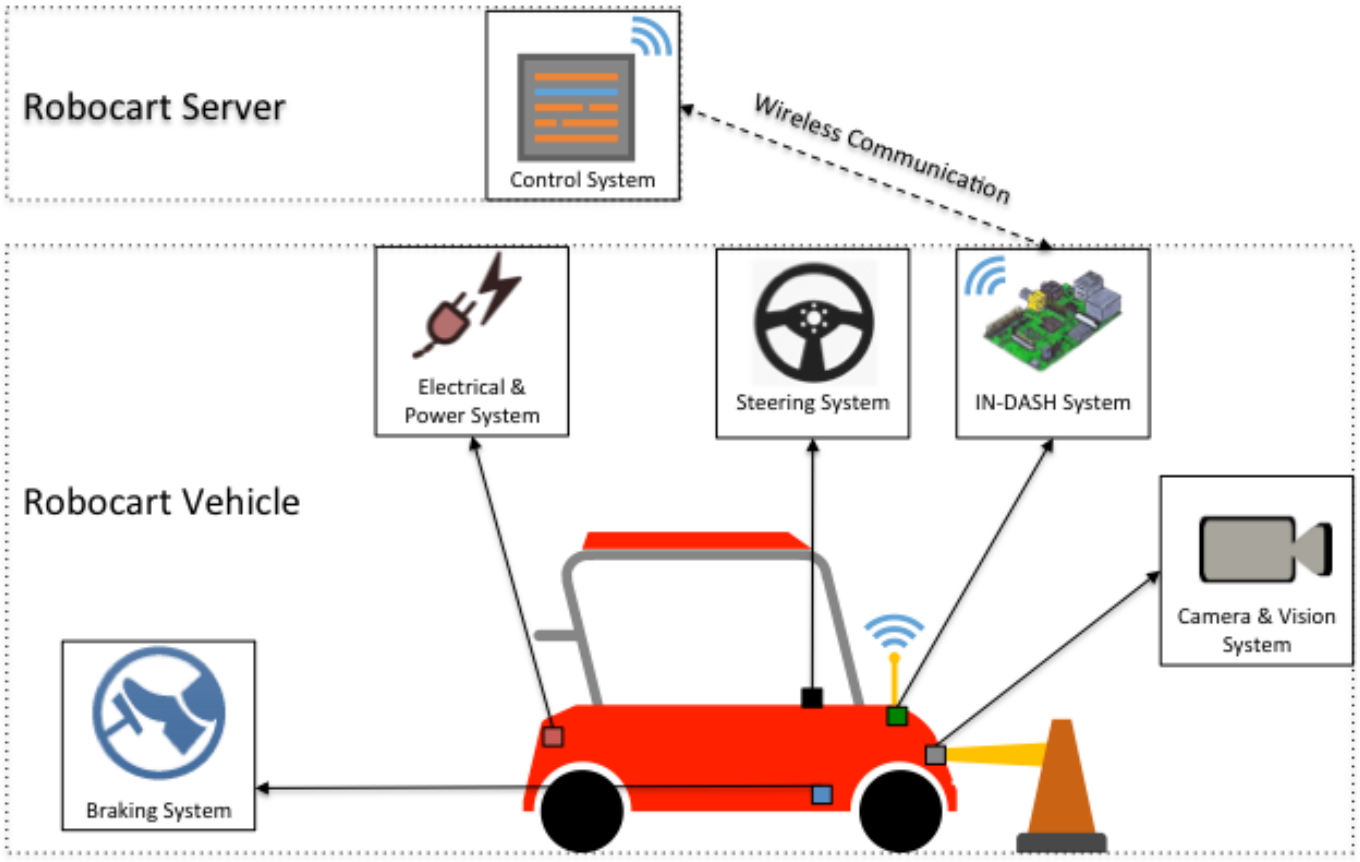


Figure 2: Each block represents a susbsystem and its respective location on the Robocart. Note that the control system is located separately from the Robocart vehicle and is part of the Robocart server. The Robocart server and vehicle are connected to the WPI network via an ethernet cable and wireless USB dongle, respectively. Information from the Robocart vehicle is transmitted to and received from the server with input and output reactions based on the Robocart's environment. More about the system configuration and integration can be found in Chapter 5.

A key result of this project is the wireless transmission of real-time camera data to the Robocart server from the Raspberry Pi Cameras. These images obtained from the Robocart IN-DASH system were processed using the Robot Operating System (ROS) on the Robocart server.

Conclusions and Recommendations

Overall, I conclude that developing an autonomous ground vehicle from a golf cart is a feasible concept worth continuing. However, because the Robocart is composed of various different subsystems that are interdependent, I determined that it would be worth having separate, more focused, MQPs within each discipline. In this section, I have outlined the five critical areas for future MQPs as well as specific steps to take within those disciplines in order to continuously improve and add value to the next-generation Robocart.

Recommendation #1: *I recommend that there is a Robocart MQP focused on Power Systems Engineering.*

I propose that there is a Robocart MQP that focuses directly on developing an on-board power system. Instead of relying on 120 VAC, the Robocart needs to be retrofitted with a mobile, rechargeable power system that will support all of the current system electronics. I suspect that a future team working on the power system may want to have a 12V drop-down circuit to 5V to power smaller devices such as the Raspberry Pi computers and cameras and any other smaller electronics.

Recommendation #2: *I recommend that there is a Robocart MQP focused on Usability, Safety, and Ergonomics.*

I believe that having a mechanical engineering team focused on designing for usability, ergonomics, and safety will add value to the next-generation Robocart because the autonomous vehicle must accommodate various riders comfortably and safely. Mechanical engineering background will come in handy when working with the Robocart because there is a signifi-

cant amount of design, bodywork, machining, and implementation necessary in order to have a structurally sound vehicle. I also recommend that the mechanical team considers adding in a second row of seating to allow four passengers versus the current maximum of two. One idea that I thought of to allow more passengers is to rotate the current row of seats to face the back and add in a second row to face the current row. This would allow all four people to face each other while riding on the Robocart. Rinspeed's concept car (shown in Figure 3) shows a visual of this seating design.

Figure 3: Seating Design for Rinspeed's Concept Car: microMAX Urban Vehicle [12]. Rinspeed's microMAX Urban Vehicle is designed with comfort and rotating seating. Front-seat passengers will be able to rotate their seats to face the back-seat passengers while the (autonomous) vehicle drives to the destination [13].

Recommendation #3: *I recommend that there is a Robocart MQP focused on path planning and algorithm development.*

I recommend that the next MQP focuses on algorithms and software development, specifically adding path-planning algorithms to the Robocart server. It is important that the Robocart is able to navigate without colliding into any obstacles. This task can be accomplished through further software development concentrated on path-planning algorithms such as A-star or wavefront. Path-planning takes into account a predicted total cost of movement and, using this information, will attempt to decide which path is the most profitable both in terms of safety, and

efficiency. The Robocart would be further enhanced through the use and implementation of path-planning algorithms.

Recommendation #4: *I recommend that there is a Robocart MQP focused on developing Graphical User Interfaces (GUI).*

I recommend that a group focuses on user input and Graphical User Interfaces (GUI). Usability is an important part of the Robocart, since it is indented to pick up riders and bring them to their destinations. Having a touch screen with user-input capabilities would provide more of a customized experience for the riders. Also, the GUI can be used to display error messages; if something is malfunctioning, the rider would be able to visually see the problem and take manual control of the Robocart.

Recommendation #5: *I recommend that there is a Robocart MQP focused on systems engineering, integration, and testing.*

I recommend that a group focuses purely on coordinating the logistics between each of the above proposed MQPs as well as the complete system design for the Robocart. Systems integration will play an even larger role once the Robocart subsystems are further developed because each subsystem must be compatible with another to ensure complete functionality. I also recommend that the next MQP focuses more on the system-wide testing component by conducting thorough verification and validation of each subsystem separately and combined.

Lastly, I recommend a second systems engineering integration MQP focusing on collaborative, wireless networking between multiple ground and aerial vehicles. I suggest that this project is not undertaken until a single Robocart vehicle works flawlessly with the Robocart server. Creating a collaborative network will require a repository of working software in order to be successful. Collaborative networks of autonomous vehicles can be used to relay information for the military and government applications; airspace data can be transmitted to the land vehicles, and ground data can be sent to the air vehicles. Refer to Figure 4 for an example of this collaborative network.

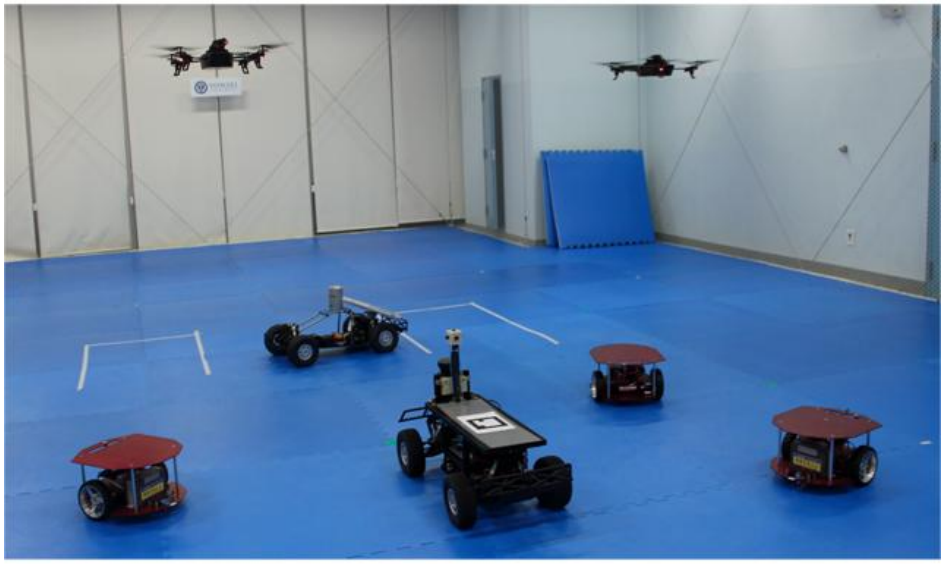


Figure 4: An example of a collaborative network consisting of ground and aerial vehicles. Data is relayed from/to the aerial vehicle to/from the ground vehicle to obtain more reliable information and greater accuracy about the vehicles' surroundings [14].

Limitations of this Research

I realize that this project has various limitations which are identified as follows:

1. This MQP only explored the software and system design of the Robocart. The results are limited to simulation and emulation-based testing, which I hope is further tested with the mechanical system implemented on the Robocart vehicle in the future.

2. The customer needs and functional requirements I developed were based on the team's perspective of the needs and requirements for an autonomous vehicle. Further research, surveys, and focus groups would be helpful to ensure that numerous perspectives are covered in terms of customer needs.

Potential Uses of the Recommendations

I created these recommendations for use by future advisors and teams working on next-generation Robocart. If these recommendations are considered, I am hoping to see the following improvements to the Robocart system:

- Fully developed and working subsystems within power systems, mechanical engineering, and software that can be integrated to form a robust autonomous ground vehicle.
- A system testing and deployment plan that can be used to satisfy customer needs and functional requirements for the Robocart.
- A user interface that is both safe and ergonomic.
- Safety dispatch messages to users on-board the Robocart through the IN-DASH system.
- Mobile ground vehicle with a reliable on-board power system.
- Real-time data transmission and processing to/from the Robocart vehicle from/to the Robocart server.

Chapter 1

Autonomous Vehicles of Today

Autonomous vehicles have been in development for over 65 years [1]. In 1948, cars were introduced to the first cruise-control systems [1]. By 2030, cars are expected to be fully autonomous and driverless. The promise of saving 1.2 million lives a year [10] and solving traffic congestion problems has struck a chord with scientists, engineers, and programmers around the world. Thanks in part to advances made in computing technologies over the past 30 years, inexpensive sensors, reliable object recognition, and real-time, portable, large-scale data analysis has become a reality. Inspired by ongoing research today from around the world, this MQP aims to explore autonomous vehicle technology from the perspective of modern vision algorithms combined with affordable sensing technology.

Several other labs have demonstrated the viability of autonomous cars in general, such as those of Google and the Autonomous Systems Laboratory at the University of California, Santa Cruz, but these systems use expensive, bulky, and ungainly roof-mounted Light Image Detection and Ranging (LIDAR) detectors [15]. The mission of this MQP is to explore vision-based systems for autonomous vehicles that use sensors in a manner that is similar to the way a human perceives his/her environment. The VisLab of Parma University in Italy, as well as several others, have demonstrated the viability of this option [15].

Autonomous vehicle research has exploded in past decades, due to increased fascination

with driverless vehicles and the impact they can have on society. Cars today come with options for adaptive cruise control, lane detection, and automated parallel parking. Further advanced autonomous vehicles blend human-control with autonomous systems, and as result, have the ability to control brakes and alert drivers of dangers [7]. Tracing the origins and historical discoveries of autonomous vehicle technologies leads us to the basis for the Robocart MQP.

1.1 Major Milestones in Autonomous Vehicles

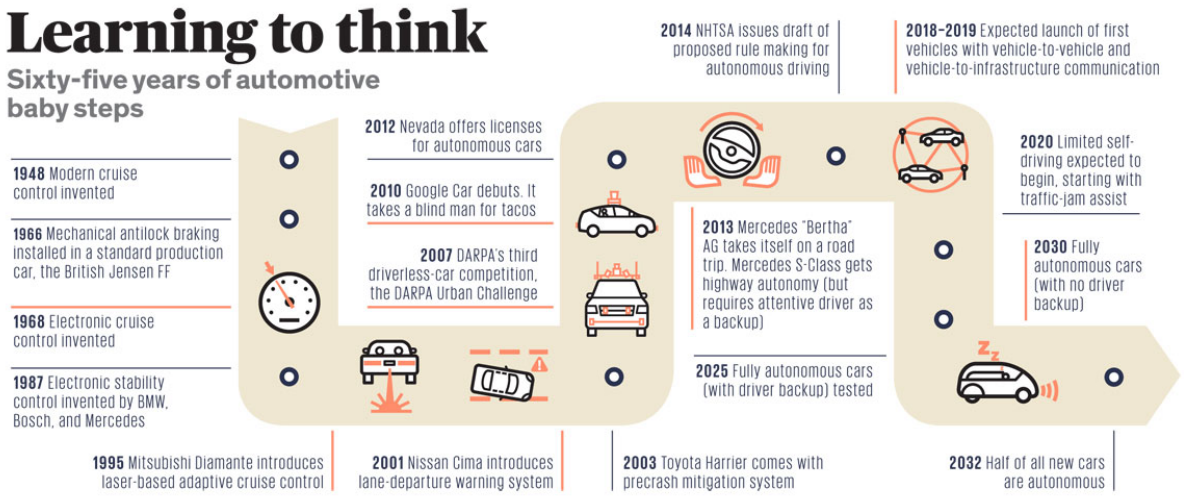


Figure 1.1: This timeline depicts the development of autonomous vehicles from 1948 to today. Starting with the invention of cruise control in 1948, vehicles are becoming more autonomous and less operator-dependent. It is expected that by 2032, half of all new cars will be fully autonomous [1].

Early on, fully-autonomous vehicles (ones that did not rely on devices embedded into roads) were few and far between. Before Martin Marietta, in conjunction with several of the research facilities and funded by Defense Advanced Research Project Agency (DARPA), introduced the Autonomous Land Vehicle (ALV) Project in 1985 [15], [16]. Martin Marietta's ALV used computer vision and laser scanning for sensing and six server racks for path correcting. It successfully traveled a half mile on an empty road in 1985, but was notoriously fickle and easily tricked by shadows and small variations in lighting [17]. During the same time period, Ernst

Dickmanns in Munich introduced saccadic vision and Kalman probabilistic filters for use in autonomous vehicles [15].

A decade later, in 1995, Carnegie Mellon developed the Rapidly Adapting Lateral Position Handler (RALPH) which used computer vision to determine the location of the road ahead to autonomously steer a car as two researchers controlled the throttle and brakes [15], [18]. Dean Pomerleau was able to "teach" an artificial neural network to drive the car (it learned to use the grass as boundaries) and was able to successfully drive on a highway at 55 mph [19]. Researchers from Carnegie Mellon were able to use this software, in a project called computer vision and No Hands Across America, to drive an autonomous car from Pittsburgh, PA to San Diego, CA for over 98% of the journey [15], [18]. By this point, autonomous path planning was in existence, but there were still many issues to be resolved before a car could actually drive itself.

DARPA also hosts a Grand Challenge prize competition for autonomous vehicles. Competing universities and organizations build autonomous vehicles to race through the competition field. This competition is over ten years old and has helped set the foundation for autonomous vehicles used in research, today. For example, the DARPA Grand Challenge in 2005 challenged universities to have driverless cars traverse a 132 mile-long off-road driving course in the Mojave Desert [20]. The competition was actually the second of its kind—the first DARPA Grand Challenge in 2004 was unsuccessful [19]. The competitors again took a wide number of approaches, utilizing combinations of Global Positioning Systems (GPS), radar, LIDAR, computer vision, sonar, and machine learning to navigate a trafficless desert course at speeds up to 25 mph [15]. The winning autonomous car, Stanley of Stanford University, used machine learning to distinguish errant sensor readings. Stanley was equipped with sensors that could detect bumpiness, differing light conditions, and accounted for them using probability distributions. Essentially, Stanley could calculate the accuracy of its readings and make fewer errors—only about 1 error in 50,000 readings [19].

Four years later, in 2010, the VisLab from the University of Parma in Italy constructed a

fully-electric autonomous vehicle that embarked upon and completed a VisLab Intercontinental Autonomous Challenge (VIAC): an 8,000 mile road trip from Parma to Shanghai [10]. Refer to the prototype vehicles in Figure 1.2. Throughout the journey, the vehicle encountered a variety of traffic, road, and weather conditions [15]. Unlike cars from the DARPA Grand challenge, the VisLab vehicle largely relied on image processing for local mapping. Other sensors on board included laser-scanning and GPS, but the lasers were mainly used for detecting terrain [10]. VisLab proved the reliability and viability of vision algorithms rather than the use of complex sensors—more akin to the way humans navigate.

Beginning in 2011, Google started a self-driving car project that leveraged their mapping technology in order to navigate roads (see Section 1.3.1). This prompted the Nevada Department of Motor Vehicles to issue the first Driver's License for an autonomous vehicle. Along the way, Google discovered more challenges with autonomous driving, including the need to program behavior for moving through a four-way intersection and for city driving.

In 2014, Volkswagen implemented the AdaptIVe Project (see Section 1.3.2) with the objective of creating autonomous vehicles that function in various levels of traffic and different driving scenarios. Specific goals include navigating a traffic jam, parking in a parking garage, and creating a robot taxi.

Autonomous vehicles of today, and even within the next 20 years [11], have high potential to provide intuitive and innovative solutions to everyday transportation needs. Autonomous vehicle research today can be grouped into three distinct sectors—government regulations, motivation of automobile manufacturers, and motivation within research institutions.

Deeva, the new VisLab autonomous vehicle

Vehicle for VIAC, Vislab Intercontinental Autonomous Challenge

BRAiVE, VisLab's first x-by-wire vehicle

Vehicle equipped by VisLab for pedestrian detection in urban areas

Vehicle for driving tests under the influence of drugs, alcool, medicines

Figure 1.2: VisLab has many prototypes for testing autonomous navigation and path planning. As shown above, majority of the autonomous vehicles are sedan styles, whereas a the Vehicle for the Vislab Intercontinental Autonomous Challenge is more similar to a van. What separates VisLab from some of the other organizations, is their focus on algorithms and cross-vehicle compatibility. In the future, I expect to see enhanced vehicle autonomy influenced by VisLab's algorithms.

1.2 Government Regulations and Issues with Autonomous Vehicles

The lack of consistent laws regarding autonomous, driverless vehicles in the United States is the main barrier hindering widespread implementation. Due to the fact fact that automated driving is a new technology, it requires legislative and regulatory action from federal and local governments. Figure 1.3 shows which states have passed, rejected, or considered implementing autonomous vehicle laws.

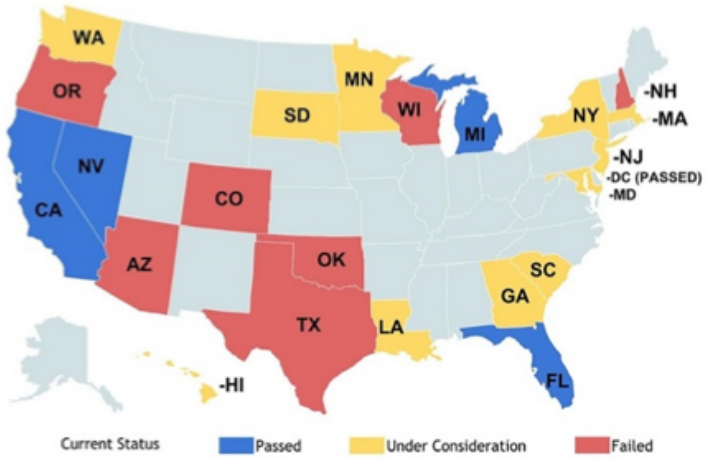


Figure 1.3: Majority of the states in the U.S. have not considered passing laws for autonomous vehicles. Of the states that have filed bills, only four states have passed: California, Nevada [21], Florida, and Michigan. In order to promote the development of autonomous vehicles, legislatures will need to consider passing autonomous vehicle laws and incentives for use on public roads and interstates.

Autonomous, driverless vehicles provide a multitude of benefits such as:

- Faster reaction time for decision-making
- Increased safety by eliminating driver error and thus reducing the number of traffic fatalities and crashes
- Reduced costs from accidents and damages

- Reduced congestion at peak-hours
- Greater fuel economy
- Greater mobility for handicapped drivers

Downsides of autonomous vehicles include policies that need to be developed around their operation. As our vehicles get smarter, complex questions such as *"Who is at fault in the event of a crash?*" or *"If the vehicle is lost or stolen, is there enough encryption to protect the information on board?*" are asked by policymakers in order to form ideas about how to mandate autonomous vehicle use. Even though there seems to be more benefits than barriers of autonomous vehicles, only four states, (California, Nevada, Michigan, and Florida), have passed bills allowing the use of autonomous vehicles on roadways [3]. As time progresses, and automakers' innovations become more reliable, governments will have to modify their policies in order to keep up with the modern technology. Autonomous vehicle development has been driven by both technology companies (such as Google), the automotive industry, and academic research. The next sections delve into the motivation behind such development.

1.3 Motivation for Autonomous Vehicle Development

Vehicles are becoming more autonomous and less operator dependent. The advances in technology and algorithm development has influenced the growth in availability of autonomous features that cars have today. Autonomous vehicle development has been influenced directly by the contributions of Google, car manufacturers, and academic research. The following Sections 1.3.1-1.3.3, provide insight about the motivation for and contributions to autonomous vehicle development.

1.3.1 Google's Autonomous Vehicle Technology

Google has been developing autonomous vehicles in order to "shoulder the entire burden of driving" [22]. Their testing is focused on the location of a vehicle based on three main areas: GPS technology and external sensor data; object identification and classification; and algorithm-based decision-making. As sophisticated as their vehicles are, they still only perform well in easily-traversable terrains and clear weather.

Google's autonomous cars use a 360-degree laser sensor to scan around the vehicle for objects. Radar scans measure the speed of vehicles ahead, while an orientation sensor tracks the car's motion and balance, and wheel-hub sensors help the vehicle determine its location [23]. While plans to make these cars fully autonomous are in the works, researchers have been investigating areas such as the quality of driving and networking to further advance this technology.

Google's autonomous cars have driven over 700,000 (urban) miles and have never caused an accident and are capable of processing chaotic urban environments [24]. Google cars are able to capture information from their surroundings, which is constantly used to make navigation decisions. Google's autonomous vehicle testing takes place around their headquarters in Mountain View, California. LIDAR scans of the environment provide visual feedback for testing and algorithm modifications. Figure 1.4 shows this visual feedback called,"Terminator Vision", which is used by the car to navigate "real" roadways with varying obstacles such as pedestrians (in and out of crosswalks), bicyclists, stop signs, railroad crossings, parked cars, and construction zones [5].

As Google's technology becomes more advanced, it is expected that "self-driving vehicles...could take your job in 20 years" [11]. In recent years, the development of autonomous cars has accelerated, and well-known automakers are racing to be the first to offer mass-produced autonomous cars. Section 1.3.2 provides more insight about the influence of car manufacturers on the development of autonomous vehicles.

Figure 1.4: Google cars are able to capture a ton of information about their surroundings. This image provides visual feedback of a Google car maneuvering in the city. City driving is considered "more complex than a mile of freeway driving" because "...vehicles must scan their environments and make (safe) decisions or predictions based on the information and data at hand..." [5].

1.3.2 Autonomous Vehicle Development by Car Manufacturers

Large car manufacturers play the most direct role in the development of today's autonomous vehicles. Recently, Volvo integrated radar, cameras, and laser technology in order to autonomously steer, brake, and accelerate [25]. Volvo is one of the first companies to make a fully autonomous vehicle available to consumers through a small Swedish-government-sponsored program called "Drive Me." The goal of this initiative is to have 100 [autonomous] vehicles on the roads by 2017 [25]. Customers who purchase cars with Volvo's autopilot technology are able to drive on roughly 30 miles of selected roads in Gothenburg, Sweden. Other car makers such as Toyota, Mercedes-Benz, BMW, Nissan, and GM are increasing the use of sensory systems within their automobiles to stay competitive in the industry.

Volkswagen's Automated Driving Applications and Technologies for Intelligent Vehicles (AdaptIVe) Project is a large-scale, unique, multi-company initiative geared towards overcoming the barriers of introducing autonomous vehicles to market [26]. AdaptIVe started in January of 2014 and will last through June 2017. The project is based on research, development, and

Figure 1.5: Funded by the European Union, the AdaptIVe Project aims to test “new sensors, software, and wireless networks in vehicle-to-vehicle and vehicle-to-infrastructure communications to make cars more ‘aware’ of their environments" [6].

testing strategies of autonomous vehicles [26]. Some barriers to consider when introducing autonomous vehicles to the commercial market include having robust vehicles that are able to work in changing and chaotic environments; can interact with human operators of varying skill levels; can seamlessly switch between manual, partially-autonomous, and fully-autonomous modes; and that follow uniform legal implications and easily-regulated liability [26].

In order to overcome these barriers, the AdaptIVe Project has created a model hierarchy (see Figure 1.6) by concentrating on research and development strategies. One AdaptIVe Research initiative includes understanding how drivers interact with the autonomous vehicle and how the vehicle interprets the current status based on input commands and its environment. Functional requirements, decision strategies, and test cases are used to develop the control system infrastructure. Experiments are conducted to improve algorithms and implementation methods. Research is also conducted on the legal implications of operating an autonomous vehicle on roadways. Consistent terminology within the standard naming conventions must be met in order for autonomous vehicles to be sustainable within society [26].

The AdaptIVE Project also focuses on development of systems for varying environments.

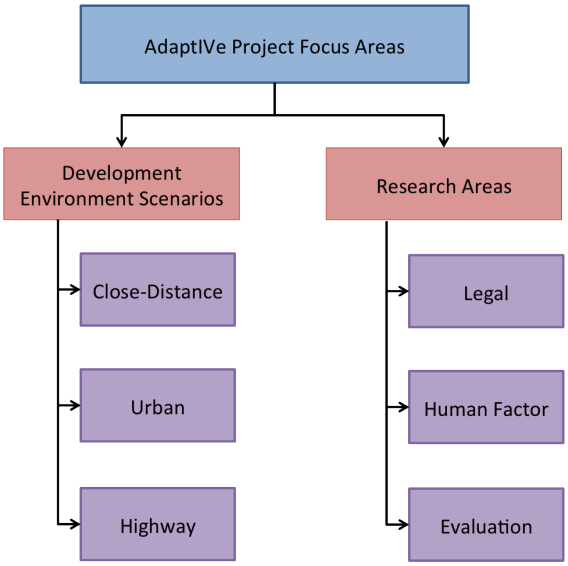


Figure 1.6: Chart created using information from [26]. The AdaptIVe Project focuses on close-Distance, Urban, and Highway Development Environment Scenarios and conducts research within the legal, human, and evaluation areas.

The main areas of concentration include operating at low speeds (less than 30 km/hr); parking by remote control in outdoor parking spaces, lots, and garages; maneuvering in crowded environments; stop and go traffic; default safe-stopping mechanisms; and improving the reliability of fail-safe vehicle platforms [26]. Development testing occurs within Urban environments, close-distance roadways, and highways. High-precision localization and mapping with and without the use of GPS is tested as well due to the limitations in accuracy within urban settings.

1.3.3 Role of Academic Research

Carnegie Mellon University (CMU) has a research division focused solely on autonomous driving technologies. They aim to make driverless vehicles practical. Some recent research from CMU related to driverless, autonomous vehicles includes improving driving quality through a behavioral planning framework, and connecting the environment to autonomous vehicles. They plan to achieve this by increasing the availability of technologically-advanced infrastructure capable of transmitting data to vehicles. As mentioned in Section 1.1, the DARPA chal-

lenges are a major motivating factor to develop autonomous vehicles in universities.

1.4 Project Objectives

This Robocart MQP is focused on developing a modular system architecture for a first-generation autonomous golf cart. By creating the foundation for a modular and interdisciplinary system, visualization software and mechanisms can be intuitively integrated. The end result of this project is a better understanding of the efficiency of each subsystems against the real-time challenges required for an autonomous, wireless, and vision-based system. Key objectives for this project are given as follows:

> **Objective #1:** *To develop the software framework and system configuration for an autonomous ground vehicle.*
>
> **Objective #2:** *To provide an Application Program Interface (API) for intuitive integration of software packages with hardware components.*
>
> **Objective #3:** *To promote forward-thinking and drive holistic development by providing recommendations for future Robocart projects.*

1.5 Report Structure

The rest of this document explains the design details and motivation specific to the system and software development for the Robocart. Chapter 2 explains the background information needed to understand the design decisions made on Robocart. Chapter 3 outlines the initial proposal set forth at the beginning of the project. Chapter 4 explains the Robocart server and vehicle configuration. Chapter 5 delves into the details to configure the software-hardware interface of the IN-DASH system. Finally, Chapter 7 gives a summary of the project deliverables and my recommendations for next-generation Robocart development.

Chapter 2

Robocart Overview

2.1 Systems Engineering Introduction

This section gives a brief introduction to systems engineering and its application to the Robocart MQP. A system is defined as a set or assembly of connected parts that form a complex whole. I focused on the system design and configuration for the intuitive dashboard (IN-DASH) systems[1] in this MQP. Systems engineering is a subject concerned with two main disciplines: the top-down development of products within a technical field, as well as a field concerned with system process optimization and management [27]. According to systems engineering fundamentals, systems engineering is defined as “a logical sequence of activities and decisions that transforms an operational need into a description of system performance parameters and a preferred system configuration" [27]. Systems engineering is an interdisciplinary field that utilizes process management to verify and validate integrated system solutions to satisfy customer requirements. Since the Robocart is a complex vehicle, it is necessary to have a logical approach to ensure appropriate systems integration. A common systems engineering process flow chart is given in Figure 2.1.

[1] Refer to Chapter 5 for more on the IN-DASH configuration.

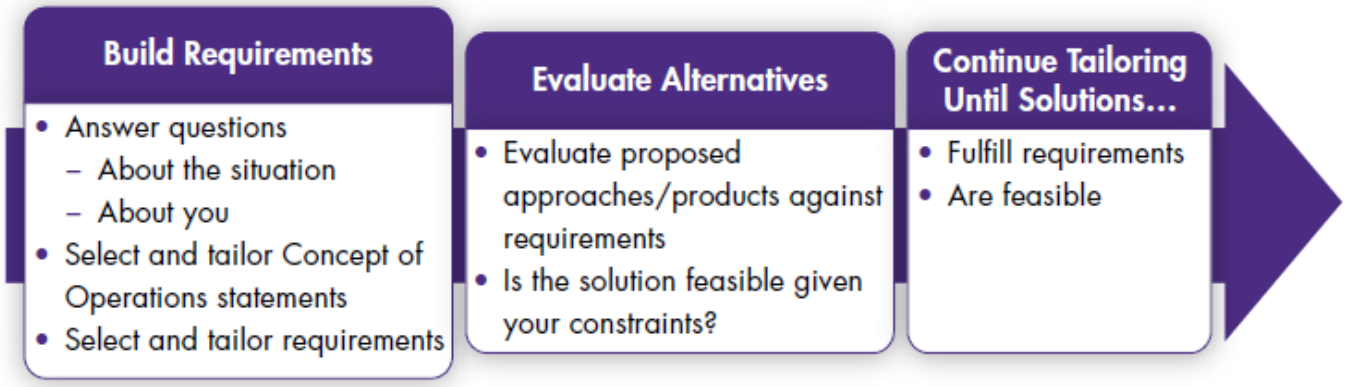


Figure 2.1: This chart provides a visual procedure for developing a system. The first step is to build requirements by answering questions about the situation and select requirements based on current functionality. Next, the proposed approaches are evaluated and the solution is checked against the original requirements. Finally, solutions are modified until all requirements are successfully fulfilled.

For this MQP, I imposed a similar systems engineering process (as shown in Figure 2.1) to build our project requirements, evaluate alternative designs, and make adjustments until the conclusion of this project. In A-term, the team focused on building our requirements and selecting the initial proof-of-concept. For majority of B and C-terms, we were in the the second (Evaluate Alternatives) and third blocks (Continue Tailoring Until Solutions). Understanding the process and having a visual execution plan helped our team reach milestones and expectations for the duration of the MQP. Chapter 3 includes more information on the logistics of this MQP.

2.2 Overview of Ackermann Steering

Ackermann steering is a method of four-wheeled steering where "each wheel travels in a curved path about a point located on an extension of the rear axle center line" [8]. Ackermann steering places the center of rotation outside of the body of the vehicle, on the same line as the back wheels to prevent slippage during turns. Refer to the model in Figure 2.2.

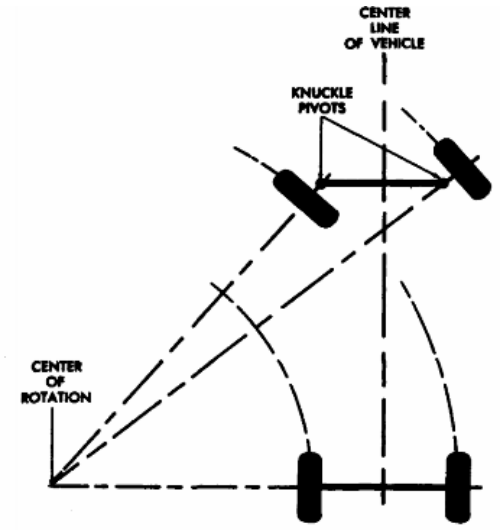
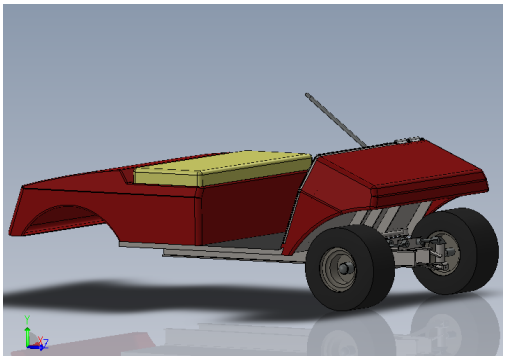


Figure 2.2: Ackermann Steering places the center of rotation outside the body of the vehicle on the same line as the back wheels. This configuration provides greater traction for turning and prevents slippage.

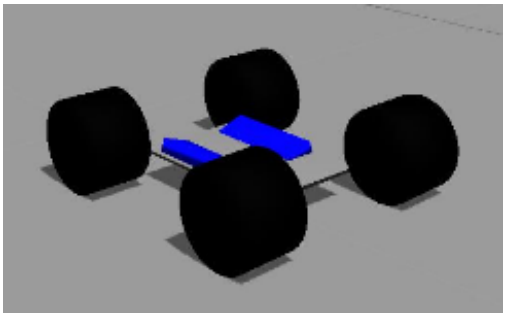
The Robocart has Ackermann steering but also considers motion in the third dimension; as the steering wheel turns, there is a curved bar attached to each wheel and the center of the vehicle to allow more length in a consolidated space. This results in a non-linear steering system. For the purposes of this report and simulation, I have represented the golf cart as a lumped model shown in Figure 2.3b. A comparison of the Computer Aided Design (CAD) model by Prateek Sahay and the lumped model is shown in Figure 2.3. I also considered linear and non-linear output to the steering system by allowing software configuration for different controllers.

2.3 Power

The power system of the Robocart was not dealt with during this MQP. I suggest and highly recommend that the follow-up projects focus directly on designing and testing a power system to supply all on-board electronics and motors the necessary power for mobile operation. The prototype system was designed and tested using 120VAC power. I took into consideration designs that would be easily adaptable and compatible with an on-board DC power system.



(a) This is a to-scale model of the Robocart. Note that the back wheels, roof, and steering wheel have not been added to the model yet.



(b) This model is an Ackermann Vehicle designed for simulation in Gazebo. Note that this lumped model consists of four wheels and a body. The steering (gear) system cannot be modeled due to the limitations of closed-loop systems in Gazebo.

Figure 2.3: Comparison between the Robocart CAD Model and Lumped Model.

Appendix A provides a list of specifications for the electronics and motors on the Robocart that would need to be incorporated into the future on-board power system. See Section 7 for more future work power ideas.

2.4 Chapter Summary

Robocart is a first-generation autonomous golf cart that attempts to integrate software, hardware, and sensors to detect obstacles and react appropriately. Robocart's power system in is purely 120 VAC with consideration for a fully independent power system in future iterations. The team decided to implement a stereoscopic vision system with object recognition algorithms

versus using expensive LIDAR sensors or ultrasonic sensors. Figure 2.4 shows a concept diagram for Robocart system modules. Chapter 3 provides details about the holistic design of the Robocart system as well as the organizational framework for team milestones and individual responsibilities in order to complete this project.

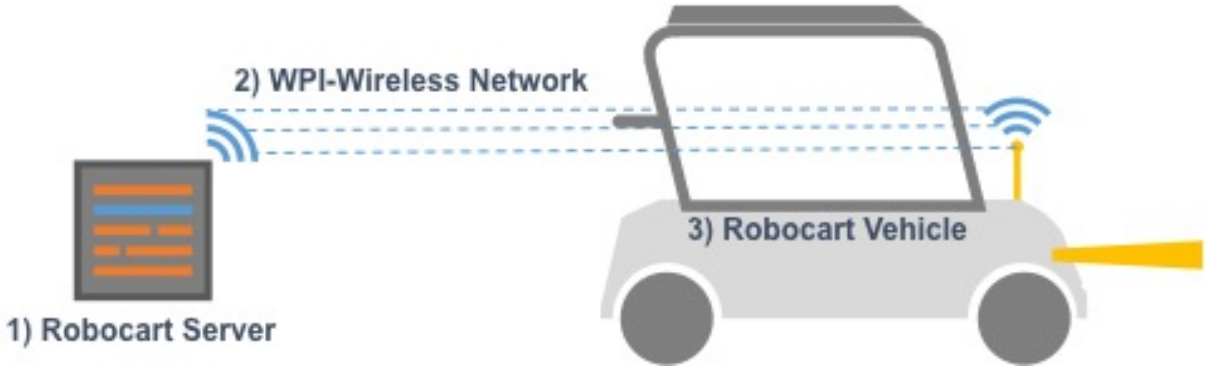


Figure 2.4: This figure shows the three main modules of the Robocart System: Robocart Server, WPI-Wireless Network, and the Robocart Vehicle. The Robocart server and vehicle are connected to the WPI network via an ethernet cable and wireless USB dongle, respectively. Information from the Robocart vehicle is transmitted to and received from the server with input and output reactions based on the Robocart's environment. More about the system configuration and integration can be found in Chapter 5.

Chapter 3

Project Approach

3.1 System Configuration for Software and Hardware

Due to the nature and complexity of the Robocart, there are two MQPs working towards the same requirements in addition to this report. While this report focuses specifically on the system configuration, there is a Robocart MQP being pursued by Prateek Sahay devoted to mechanical implementation and by Gabriel Isko devoted to vision and software development. Figure 3.1 is a block diagram, which shows how both of these MQPs interact with and complement each other.

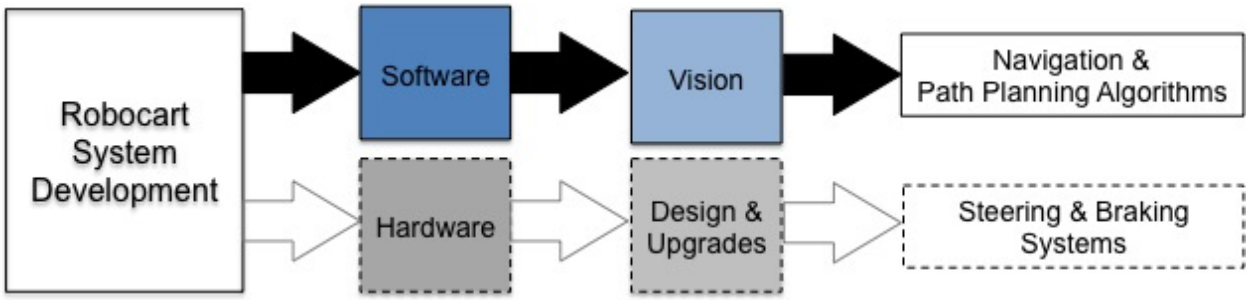


Figure 3.1: There are two MQPs working in parallel to develop the Robocart. This report focuses on the Software and System-wide development. As shown by the block diagram, the Robocart system development can be broken down into two parallel tracks: software and hardware. Upon completion of the Robocart system, the mechanical modifications (dashed, gray hardware blocks) will interface directly with the software and system design (solid, blue blocks).

The solid blocks in Figure 3.1 represent the main topics of exploration in this MQP. They include vision, navigation and path planning, and simulation/emulation. The dashed blocks represent the mechanical subsystems that are out of the scope of this MQP report and are covered by the MQP report of Prateek Sahay. Before developing any of our objectives and goals, the team conducted brainstorming sessions to help make project decisions. The following Section 3.2 explains the brainstorming and decision-making process.

3.2 Project Brainstorming and Decision-Making

We used brainstorming to help generate a list of ideas for this MQP. Within the first few team meetings, we discussed concepts, mechanisms, software, and system setup based on what we wanted to work on for the duration of the project. Some of our initial project ideas for Robocart included the following concepts[1]:

1. Develop three modes for Robocart operation: (1. fully autonomous, 2. manual, 3. Remote Controlled (RC)).

2. Conduct an analysis between the efficiency of OpenCV and ROS or OpenCV and MATLAB for vision-based algorithm development.

3. Design the system configuration by identifying and implementing necessary subsystems (software and hardware) for the Robocart.

4. Develop software for image capture and processing.

5. Run testing on hardware/software components to verify the system setup.

6. Ensure that our implementation(s) would contribute to the longevity of the project and would be compatible with next-generation upgrades and developments.

[1]Note: Not all of our original ideas from the brainstorming session are given. This list shows our top six ideas that we worked with and modified to create our final objectives.

We decided to work on #3, #4, and #5 while also following idea #6. Our decision to create objectives based on those ideas was based on both teammate's strengths and the 21-week time period to submit our deliverables. As the project progressed, the ideas turned into objectives, which were then further decomposed into specific software, integration, and testing goals. Each teammate then branched off and focused on the discipline of most interest: Elizabeth worked on holistic system development and deployment plans; Gabriel worked on vision and algorithms for the Robocart server and IN-DASH systems; Prateek developed and implemented mechanical systems for the Robocart vehicle. Sections 3.3- 3.4 provide more information about the metamorphosis of our ideas to the finalization of concrete requirements for this MQP.

3.3 Project Objectives Revisited

The Objectives for this MQP were given in Chapter 1 and are reintroduced as follows:

> **Objective #1:** *To develop the software framework and system configuration for an autonomous ground vehicle.*
>
> **Objective #2:** *To provide an Application Program Interface (API) for intuitive integration of software packages with hardware components.*
>
> **Objective #3:** *To promote forward-thinking and drive holistic development by providing recommendations for future Robocart projects.*

Objective #1 stems from ideas #3 and 4; objective #2 is based on idea #4; and objective #3 is a result of ideas #5 and 6. We recognized the need to decompose these objectives into smaller, more specific goals based on three categories: software, integration, and testing. The Systems/Software objectives were determined through brainstorming sessions. Table 3.1 shows the smaller system and software goals relevant to Objective #1. Table 3.2 shows the smaller API goals created from Objective #2. Table 3.2 shows the goals developed from Objective #3.

Table 3.1: Objective #1 decomposed into Smaller System/Software goals.

Goal #	**Description**	**Category**
1-1	Use Image Processing to learn about the environment.	Software
1-2	Utilize ROS packages for the foundation of navigation and path planning.	Software
1-3	Configure Gazebo to include a cart model with Ackermann Steering[2].	Testing
1-4	Configure RViz to accept data output from Raspberry Pi Computers and Cameras	Integration

Table 3.2: Objective #2 decomposed into Smaller API goals.

Goal #	**Description**	**Category**
2-1	Create an API for General Purpose Inputs and Outputs (GPIO) pins on the Raspberry Pi and ROS	Software
2-2	Create a software package between ROS/RViz and ROS/Gazebo.	Software

The Systems/Software Objectives are decomposed into three main categories: software, simulation/emulation, and testing. Software includes the code framework necessary for image processing, ROS communication, and Application Program Interfaces (APIs) to the Raspberry Pi cameras and output steering motor. Simulation and emulation is the integration component of this MQP. We aim to provide an accurate lumped model of the Robocart within our simulation/emulation to obtain reliable data. Testing includes the use of the simulation/emulation data to draw conclusions and conduct an analysis for future system recommendations and next-generation Robocart MQP project ideas.

Table 3.3: Objective #3 decomposed into Forward-thinking and Development Goals.

Goal #	Description	Category
3-1	Generate test cases and run software tests	Testing
3-2	Use test results to determine whether requirements have been met.	Testing
3-3	Provide system requirements, wiring diagrams, and recommendations for further Robocart development	Integration

3.4 System Requirements

We started out this project by considering the needs of the end-user (qualitative features) before creating specific quantitative functional requirements. We used a standard industry model (see Figure 3.2) to help organize the process for completing tasks.

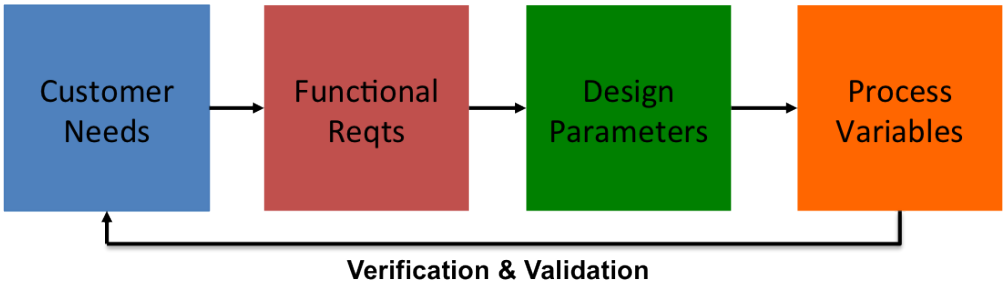


Figure 3.2: The Robocart System design process began by first assessing customer needs. Once the needs were determined, we created functional requirements based on these needs. Design parameters, or how to implement the design, as well as process variables (how to produce the deliverables), were developed over the duration of the MQP. Finally, the whole process was tested through verification and validation. Modifications to the system were made as needed.

This figure shows how product ideas start as a customer need and are turned into specific quantified functional requirements. Design parameters are created based on these requirements; functional parameters are created on the design parameters; and finally the whole chain is verified through testing and validation to ensure that the product meets the original customer needs and functional requirements. We used this model as a guideline to satisfy the project requirements.

Together, we generated a list of customer needs. We wanted to ensure that the end-user's expectations were met for First-Generation Robocart and that there was room for continuous

improvement by the incoming Robocart teams. The customer needs defined for this project are listed as follows:

Customer Need #1: To autonomously detect and avoid obstacles while the Robocart is in operation.

Customer Need #2: To insure safety while the Robocart is in operation.

Customer Need #3: To provide visual feedback of detected objects in front of the Robocart.

From the customer needs, we created quantitative functional requirements. Each of the functional requirements has a priority ranking for easy testing during verification and validation. Priority rankings are given as follows:

- **Priority 1** - The requirement is a "must-have" for completion date: March 6, 2015
- **Priority 2**- The requirement is needed for improved functionality. Completion will increase immediate benefits; however incompletion will not have a major effect on the overall system.
- **Priority 3** - The requirement is considered "nice-to-have" and may include new functionality to enhance and optimize performance and aesthetics.

The following list provides the major requirements and respective priority for the first-generation Robocart. Requirements are numbered such that each requirement stands for a Project Requirement (PR) with a unique number that represents both the priority (p) and an identification number (n) (PR_p-nn).

Robocart System Requirements List

PR_1-00 *The system shall support two riders.*

PR_1-01 *Negative braking will occur at all times except when an object is not detected.*

PR_1-02 *The system shall have three emergency stop buttons placed in strategic locations.*

PR_1-03 *The system shall not exceed two riders at any given time.*

PR_1-04 *Each passenger seat will be equipped with a safety harness.*

PR_1-05 *The system shall have a clear signal when in operation.*

PR_3-06 *The dashboard electronics will be accessible to all users.*

PR_2-07 *The system will respond to obstacles less than two-feet away by slowing down and stopping.*

PR_1-08 *The system will detect obstacles directly in front of it within two-feet.*

PR_2-09 *The system will react to detected obstacles by steering in the opposite direction.*

PR_2-10 *The system will come to a complete stop if there is not a clear route around a detected obstacle.*

The remainder of this report is based on the two main disciplines of the project: software engineering and systems engineering as well as our conclusions and recommendations. The next sections outline the project logistics, milestones, and individual responsibilities.

3.5 Project Logistics

Project tasks were delegated using Teamwork.com, an online project management tool. The Milestones for this project are given by term in Figures 3.3-3.5.

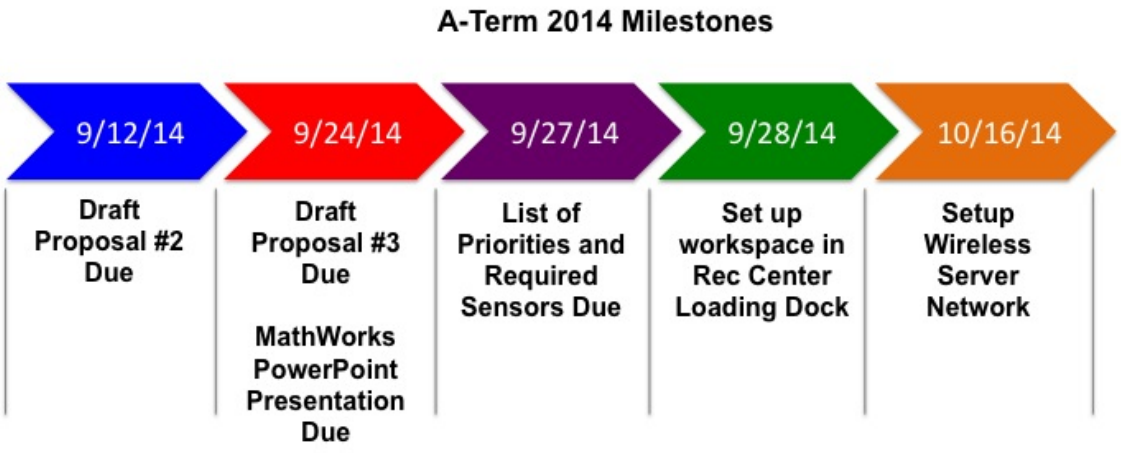


Figure 3.3: This figure shows the milestones created for A-term 2014. We successfully wrote two draft proposal papers, created a workspace in the Rec Center Loading Dock, and setup the wireless server on the WPI-network.

For C-term, we decided make a Gantt Chart with all of our Milestones so that we had a visual aid to keep us on track for the remaining term of the project. We have included a photo of the Gantt Chart for C-term because it shows the critical deadlines and deliverables for the completion of this project. See Figure 3.7.

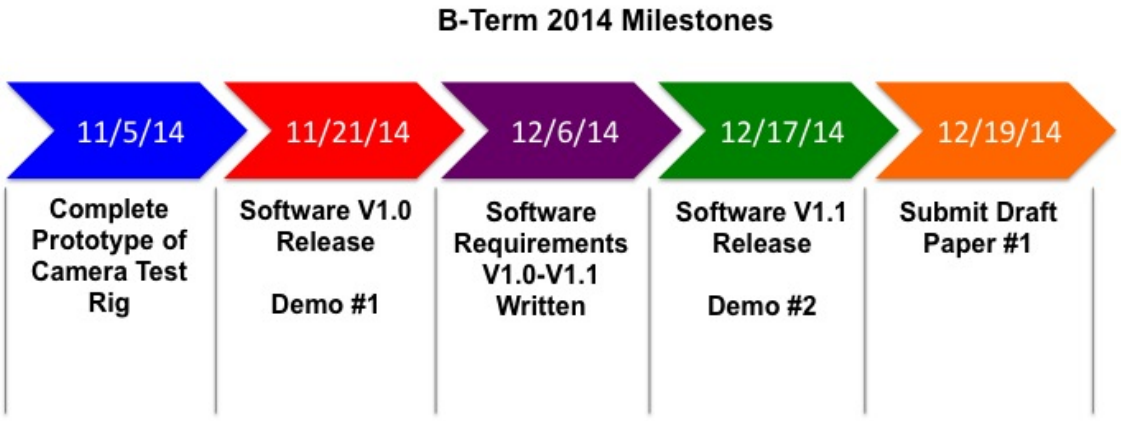


Figure 3.4: This figure shows the milestones created for B-term 2014. We made a prototype of the Raspberry Pi Computer/Camera mount, released two software versions, and presented two demonstrations. Finally we submitted the first draft of the MQP report.

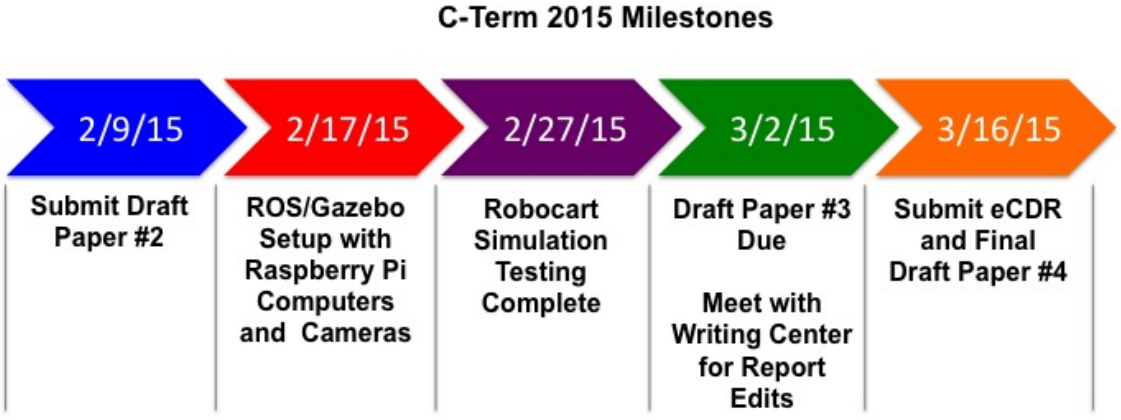


Figure 3.5: The C-term milestones are presented in this figure. We submitted two more report drafts, configured ROS/Gazebo/RViz to conduct simulation testing, and submitted our final report and eCDR for completion of the project.

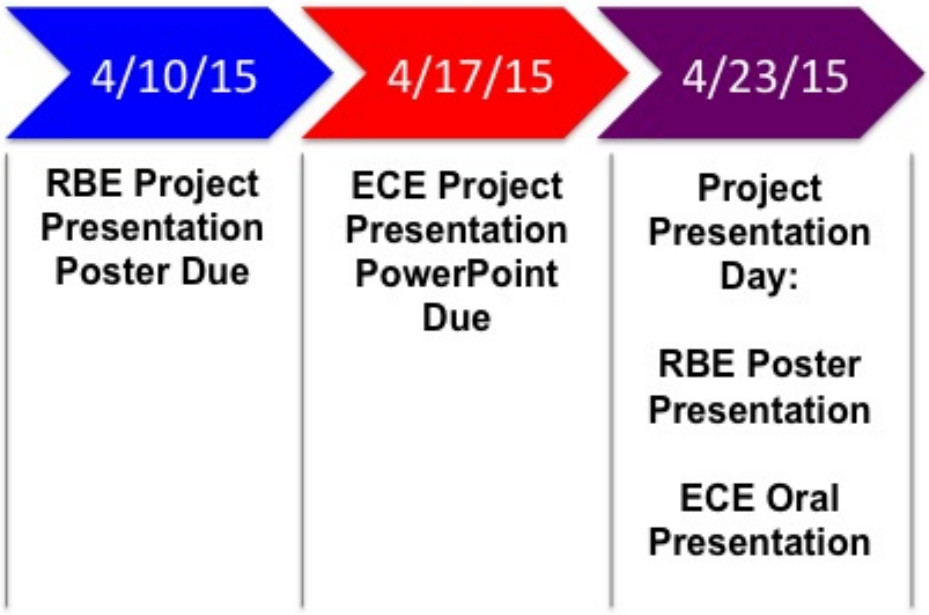


Figure 3.6: Even though D-term is not an official term for this project, there are still critical dates to prepare for Project Presentation Day (4/23/15). We have decided to present for both Robotics Engineering (RBE) and Electrical and Computer Engineering (ECE) departments and have allocated deadlines for the respective poster and PowerPoint required for these presentations.

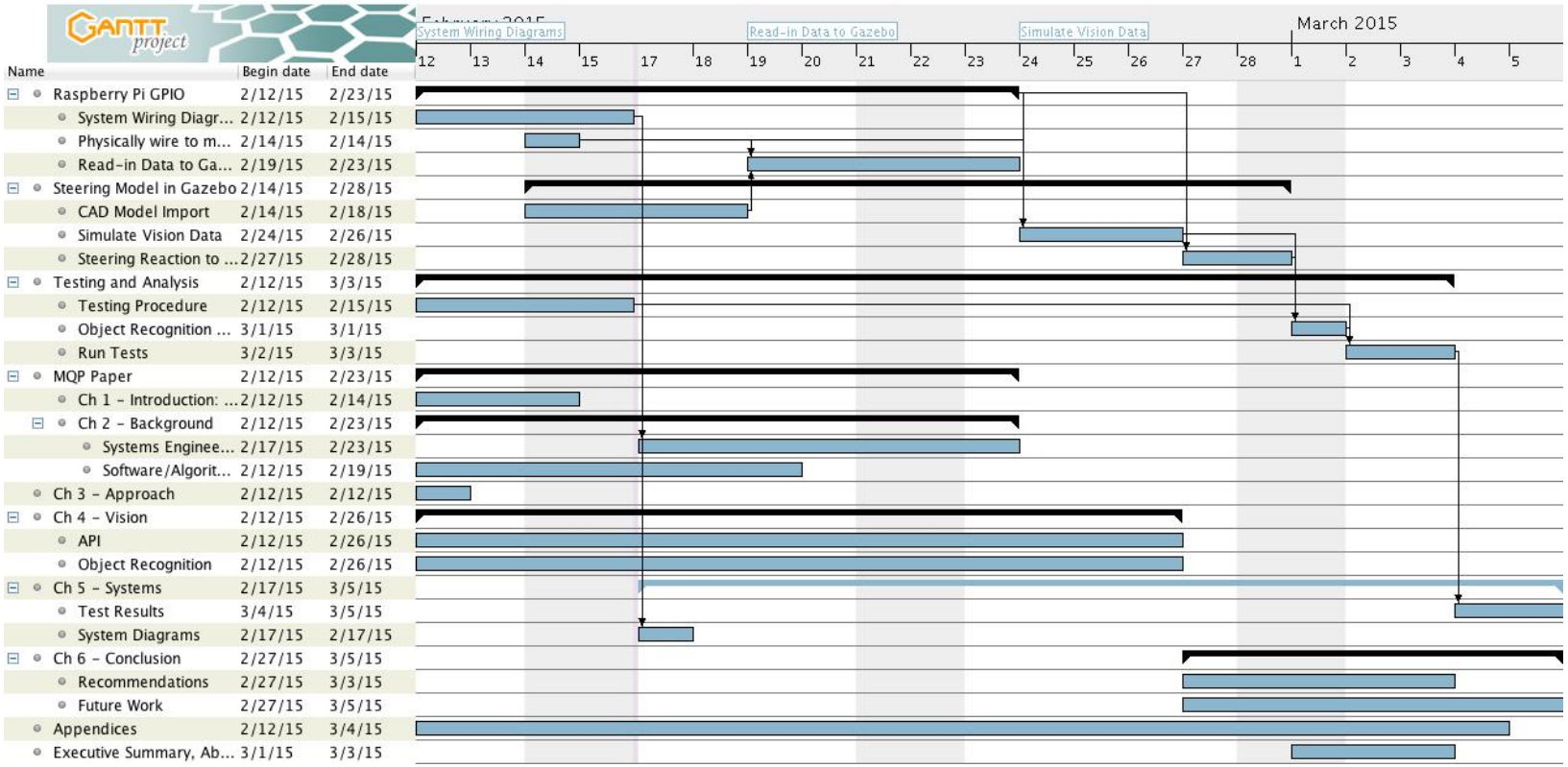


Figure 3.7: This Gantt chart was developed in Gantt Project to ensure that all deliverables and deadlines were met for completion of this MQP. We suggest creating a 21-week version of the Gantt Chart within the first week of starting the MQP and updating it frequently to account for unexpected changes and adaptations.

3.6 Division of Tasks

Tasks were divided among teammates based on the subject matter. Elizabeth was responsible for system design and integration; Gabriel was the main software developer; and Prateek implemented mechanisms to the Robocart vehicle. To keep organized, Elizabeth maintained the Teamwork.com portal (robocart.teamwork.com) by adding in due dates for each person's task and updating team Milestones. Our meeting schedules for A14–C15 are shown below in Tables 3.4– 3.6.

Table 3.4: A14 Regularly Scheduled Team Meetings and Check-ins

Day	**Time**	**Attendees**
Tuesdays	1300-1400	Robocart Team
Wednesdays	1300-1400	Robocart Team
Fridays	1200-1300	Robocart Team and Professor Wyglinski
Fridays	1300-1400	Robocart Team
Number of Scheduled Hours/Wk	4 Hours	

Table 3.5: B14 Regularly Scheduled Team Meetings and Check-ins

Day	**Time**	**Attendees**
Tuesdays	1200-1400	Robocart Team
Wednesdays	1100-1200	Robocart Team
Fridays	1200-1300	Robocart Team and Professor Wyglinski
Fridays	1300-1400	Robocart Team
Number of Scheduled Hours/Wk	5 Hours	

Table 3.6: C15 Regularly Scheduled Team Meetings and Check-ins

Day	**Time**	**Attendees**
Mondays	1300-1500	Robocart Team
Tuesdays	1300-1500	Robocart Team
Wednesdays	1200-1230	Robocart Team and Professor Wyglinski
Wednesdays	1230-1430	Robocart Team
Thursdays	1300-1500	Robocart Team
Number of Scheduled Hours/Wk	8.5 Hours	

Each teammate also worked on the Robocart MQP roughly 3 hours per day, at minimum for approximately 25 hours/week.

3.7 Chapter Summary

The Robocart MQP took place over 21 weeks through A, B, and C-terms 2014–2015. System objectives are as follows...:

Objective #1: *To develop the software framework and system configuration for an autonomous ground vehicle.*

Objective #2: *To provide an Application Program Interface (API) for intuitive integration of software packages with hardware components.*

Objective #3: *To promote forward-thinking and drive holistic development by providing recommendations for future Robocart projects.*

...and were developed based on three main customer needs:

Customer Need #1: To autonomously detect and avoid obstacles while the Robocart is in operation.

Customer Need #2: To insure safety while the Robocart is in operation.

Customer Need #3: To provide visual feedback of detected objects in front of the Robocart.

Milestones were then created for each term and were based on the goal statements. Team meetings and check-ins with Professor Wyglinski were held daily and weekly, respectively. Chapter 4- 6 gives an overview of the system configuration and integration for subsystems of the Robocart.

Chapter 4

Robocart Subsystems

Robocart is composed of seven subsystems developed in this MQP as the Robocart system structure. The subsystems can be placed within a system diagram, which is shown in Figure 4.1. Note that the IN-DASH system represents the vehicle-side wireless networking system and the control system represents the server-side wireless networking system.

Each subsystem above is described as follows:

Subsystem #1 Intuitive Dashboard (IN-DASH): Contains the computer and camera system that detect obstacles in the front of the cart.

Subsystem #2 Steering (Mechanical): Software-controlled steering of the cart; composed of a Brushed DC motor, sprocket, and chainset.

Subsystem #3 Braking (Mechanical): Software-controlled (negative) braking that controls whether the cart is stationary or in motion.

Subsystem #4 Camera and Vision: Cameras relay information to the computers where vision algorithms process that data for obstacle detection.

Subsystem #5 Electrical and Power: On-board DC power system for all subsystems of the Robocart.

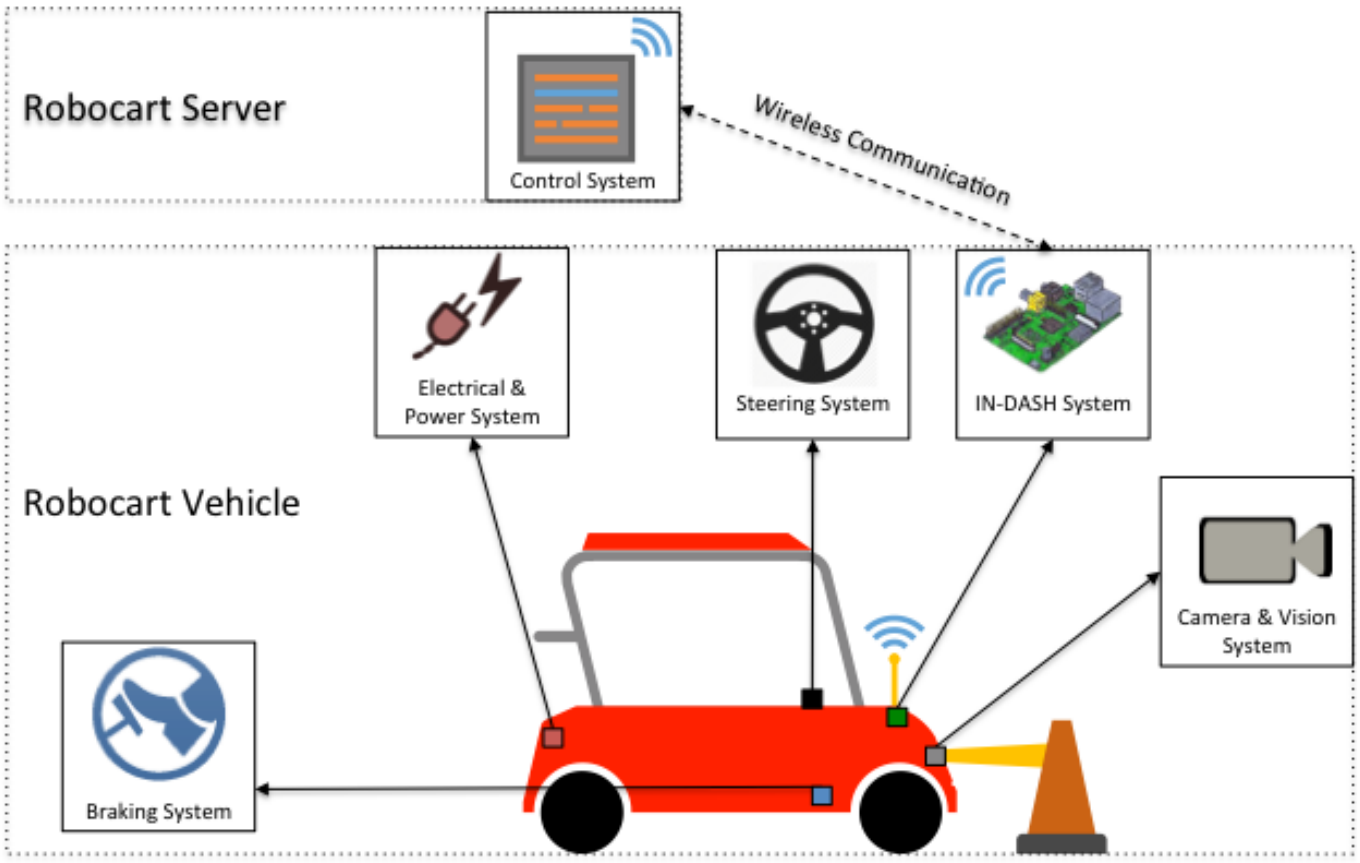


Figure 4.1: Each block represents a susbsystem and its respective location on the Robocart. Note that the control system is located separately from the Robocart vehicle and is part of the Robocart server.

Subsystem #6 Software Control: Contains ROS packages and interfaces for hardware communication and data processing

Subsystem #7 Wireless and Networking: Communication bridge between the mobile Robocart and stationary Robocart server (Refer to Figure 4.2).

The Robocart vehicle is equipped with computers that can connect wirelessly to the Robocart server. The Robocart server is hosted on the WPI network; we connect the Robocart vehicle to WPI-Wireless for bidirectional data transmission. We are able to connect to the Robocart server from any Internet access point because it was assigned a static hostname (robocart@mqpburris.wpi.edu) through WPI; this includes external Internet access points. Figure 4.2 shows this wireless configuration for the Robocart vehicle and server. We have included

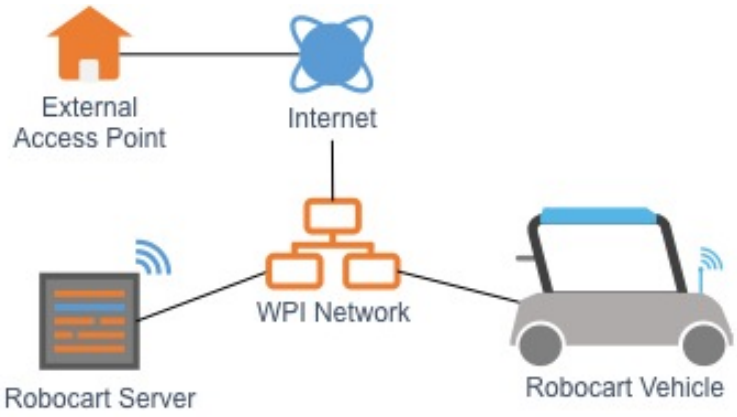


Figure 4.2: The Robocart system is composed of the Robocart server and Robocart vehicle which communicate with each other using wireless communication through the WPI Network and the Internet. We have configured the Robocart networking system to allow connections to the Robocart server from both access points within the WPI-Wireless network or from external access points connected to the World Wide Web.

instructions for accessing the Robocart server in Appendix C.

In order to operate, the IN-DASH system relies on power from the electrical and power system. Therefore, as a result of the interaction map, we can infer that each of these systems is dependent on the others. I hope that the next-generation Robocart team utilizes system integration techniques to combine each subsystem into a robust and reliable integrated system. The remainder of this chapter is devoted to the integration of the IN-DASH system with the software-hardware interface to control the steering system.

4.1 Chapter Summary

The Robocart system is composed of seven subsystems given as follows: IN-DASH system, steering (mechanical) system, braking (mechanical) system, camera and vision system, electrical and power system, software control system, and wireless and networking system. These systems work together to provide the basic framework for autonomous functionality. These systems are interdependent of another, which form the basis for an integrated Robocart server and vehicle system. Chapter 5 explains the configuration of the IN-DASH system.

Chapter 5

Intuitive Dashboard System (IN-DASH)

The Intuitive Dashboard (IN-DASH) system contains the input/output devices for the front of the Robocart. IN-DASH provides the Robocart with critical information about objects and obstacles that may be in its current trajectory. The following components make up the IN-DASH System:

- Two Raspberry Pi Model B boards.
- Two Raspberry Pi cameras.
- Two USB wireless USB dongles (Realtek RTL8188CUS chipset).
- One Composite Display monitor.
- One USB hub.
- One USB keyboard.
- One USB mouse.
- 120 VAC power supply. [1]

[1] A battery operated power system was not tested in the scope of this MQP.

The Raspberry Pi boards (#1 and #2) are each connected to cameras and are responsible for relaying local images over the WPI-Wireless network to the Robocart server (also on the WPI network). The Raspberry Pi boards are connected to the Robocart server via a ROS node where the object-recognition and vision algorithims process the image data. Refer to Chapter 4 for more information on these algorithms on the Raspberry Pi boards and the Robocart server.

IN-DASH is partially mounted on the front hood and the internal dashboard of the Robocart. The Raspberry Pi Boards and cameras are housed inside of an acrylic mount, designed by Prateek Sahay from the Robocart Mechanical MQP. Refer to Figure 5.1.

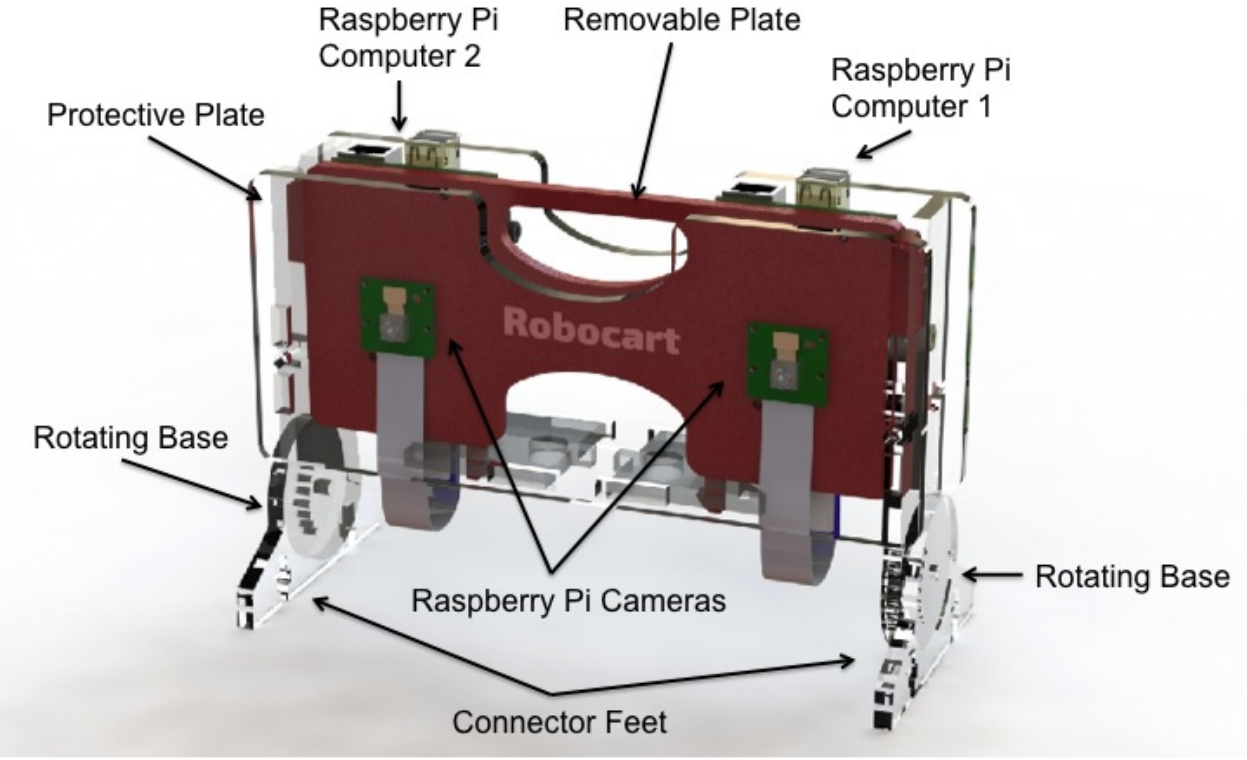


Figure 5.1: The Raspberry Pi Mount holds both of the Raspberry Pi Computers and the Raspberry Pi Cameras. The feet of the Raspberry Pi Mount connect to the front hood of the Robocart and are attached to a rotating base that can be adjusted for varying viewing angles.

5.1 IN-DASH System: Data Transmission Diagram

We connected a USB keyboard and mouse on Raspberry Pi board #1 using a USB hub. We used composite cables to connect both Raspberry Pi boards to a multi-input composite display monitor and also connected USB wifi dongles to utilize wireless communications between the Robocart and the Robocart server. The main goal of the IN-DASH is to provide not only visual feedback locally, but also user-input via the keyboard and mouse. We expect that another MQP in the future will create a more user-friendly Graphical User Interface (GUI) for the IN-DASH. See Chapter 7 for more information on future work. For the scope of this MQP, the power system is presumed to be 120V AC using the component's respective AC wall adapter. We decided to create a wiring and information flow diagram to better represent how the IN-DASH system is configured. The two Raspberry Pi computers are each connected to a Raspberry Pi camera and a 5VAC/2A power supply. Raspberry Pi #1 is connected to the composite display, which is powered by 12VAC/2A, and a USB hub which connects the keyboard, mouse, and wifi dongle. A second USB wifi dongle is connected to Raspberry Pi #2. The wifi dongles allow wireless connection (data transmission) to and from the the Robocart server. Refer to Figure 5.2 for the information flow and wiring diagram for the IN-DASH system.

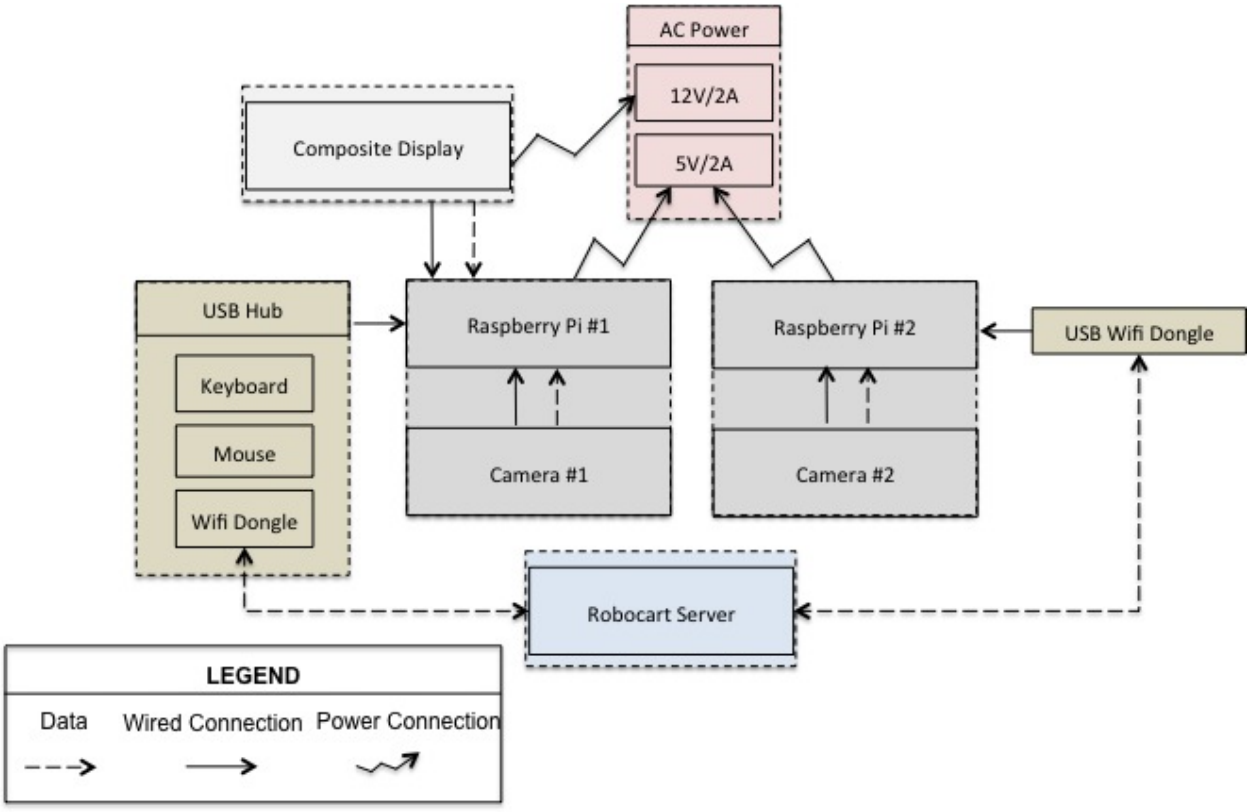


Figure 5.2: To read the diagram, follow the arrows leading to/from a component block. A dashed arrow represents data or information flow; a solid arrow represents a physical wire connection; a zig-zag arrow represents a power connection. The arrow heads point in the direction of the connection (i.e. “the USB hub is connected to Raspberry Pi #1").

5.2 Integrating the IN-DASH Stereoscopic Camera System

The purpose of the IN-DASH System is to provide the framework necessary to receive and transmit data from Robocart’s local surroundings to the Robocart server (via WPI-Wireless) for further processing. Our setup of the IN-DASH system is shown in Figure 5.3.

One of the overall system requirements for Robocart is to maintain wireless connectivity from the on-board IN-DASH system to the Robocart Server. We had numerous issues with the configuration of the Raspberry Pi wireless USB dongles. First off, the wireless dongles did not recognize or have the ability to scan for wireless networks. Both were recognized as USB devices; however, the scanning capabilities were not present. Secondly, we had issues connecting to WPI-Wireless, which is necessary in order to connect to the Robocart Server.

Refer to Appendix B for the instructions we used to correctly connect Raspberry Pi computers to WPI-Wireless.

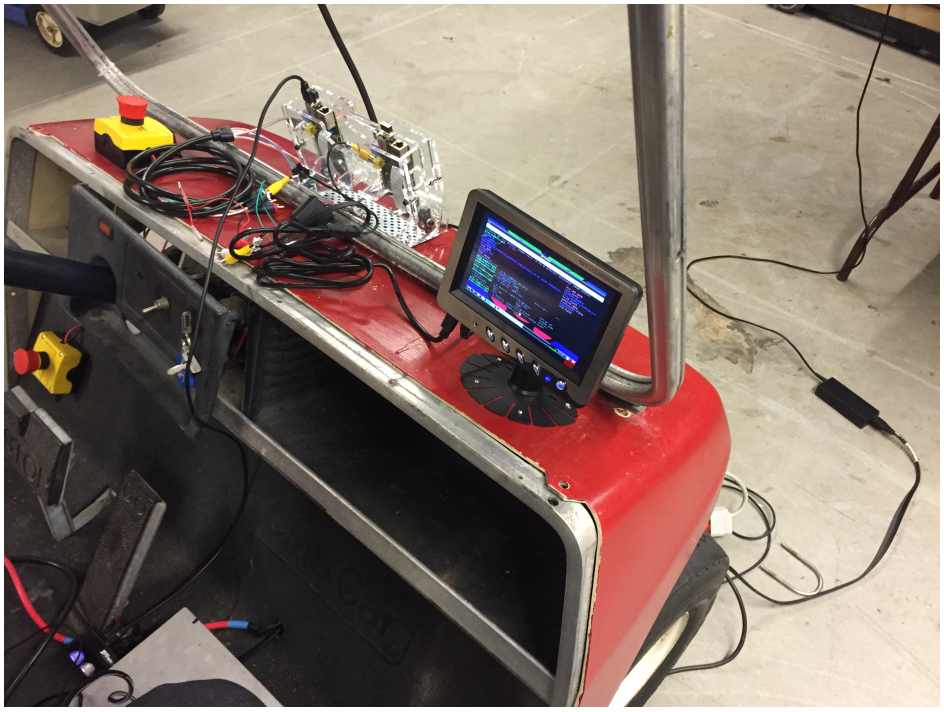


Figure 5.3: The Raspberry Pi Camera Mount, Composite Display, and USB Peripherals (Keyboard/Mouse) are mounted to the front of the Robocart. The Composite Display provides visual feedback to passengers on the Robocart, while the Raspberry Pi Computers and Cameras obtain visual information about the Robocart's surroundings.

5.3 Image Capture from the IN-DASH System

We tested the IN-DASH system by running local tests on the Raspberry Pi computers. We captured both still images and real-time video feed from the Raspberry Pi computers. Figure 5.4 shows the the local testing of the Raspberry Pi computers and cameras. The real-time video stream is shown in Figure 5.5 as a still image. Prateek Sahay is standing in front of the Raspberry Pi Cameras which capture his movement and display it locally on the composite monitor of the IN-DASH system. The data from these images and real-time camera feed is then sent directly to the Robocart server via the WPI-Wireless network for further processing in ROS.

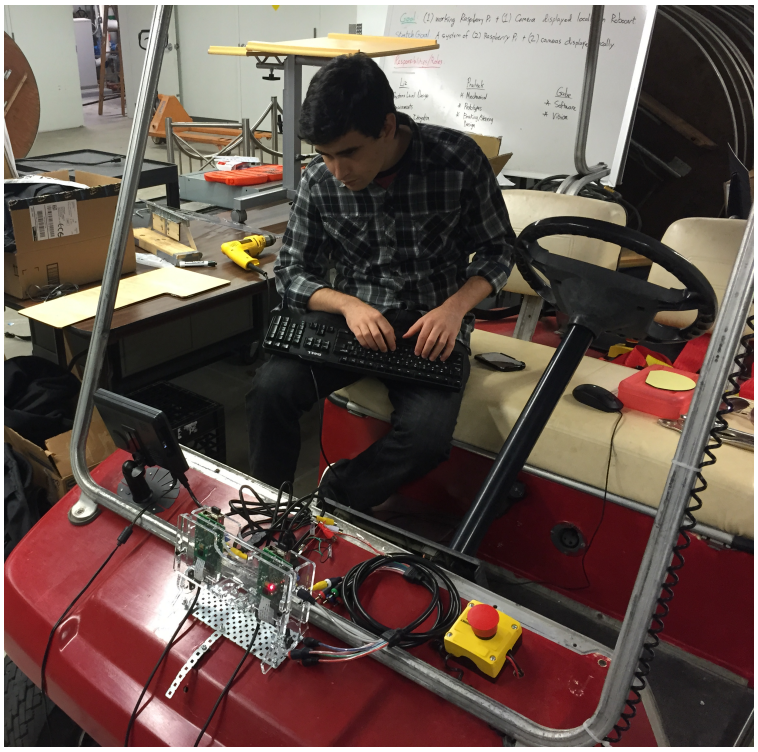


Figure 5.4: We tested the Raspberry Pi Cameras by running software locally on the Robocart (shown here) and through the Robocart server.

Figure 5.5: The Raspberry Pi Cameras are capturing live data of the surroundings, which is displayed on the Composite Monitor of the IN-DASH system. Images are captured and processed on the Robocart server for object recognition.

5.4 Chapter Summary

The IN-DASH system contains input (camera) and output (mechanical steering) components for object detection in the front of the Robocart vehicle. Two Raspberry Pi computers within the IN-DASH system house wifi dongles to communicate wirelessly with the Robocart server. The Robocart's added wireless connectivity promotes the use of complex vision algorithms for object detection by allowing appropriate hardware to support intensive processing. The Robocart server is stored in a secure location, and therefore does not hinder on-board power requirements or vehicle mobility. Chapter 6 explains the mechanical and software integration of the Robocart steering system.

Chapter 6

Steering System: Software and Hardware Integration

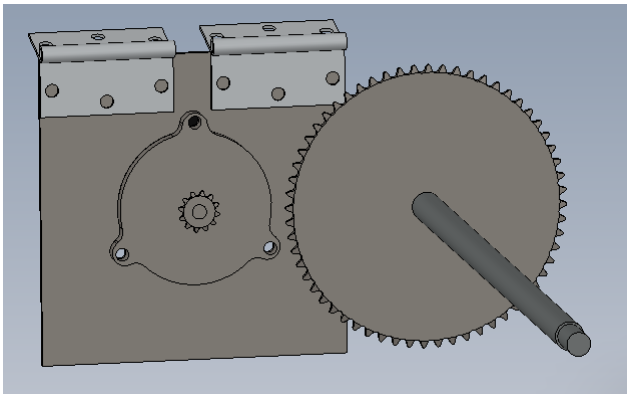
The steering system is composed of a sprocket and chainset, a motor, and a motor controller interface to the Raspberry Pi General Purpose Input and Output (GPIO) pins. The mechanical steering system was designed by Prateek Sahay, and the CAD models are given in Figure 6.1.

As the Brushed Motor turns, a 12-tooth sprocket rotates a 60-tooth sprocket attached to the steering column. From Equation 6.1, we obtained a gear ratio of 5.

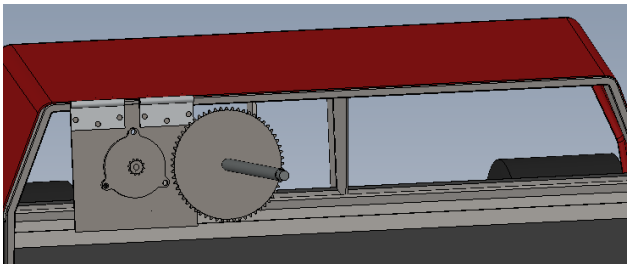
$$Ratio = \#teeth(driven)/\#teeth(driver) = 60/12 = 5 \tag{6.1}$$

A drawing of the sprocket system is shown in Figure 6.2. Note that the driven is the steering shaft sprocket and the driver is the motor sprocket. We suggest viewing the mechanical report by Prateek Sahay for more information about the calculations and decisions made when designing the mechanical steering system.

Due to the closed-loop nature of the sprocket system, we were not able to simulate this system in Gazebo; Gazebo supports tree-structures and cannot reference parent components that form a closed-loop. However, we were able to configure Gazebo with an Ackermann vehicle package that, once fully implemented with Ackermann messages, will be able to receive



(a) The Steering System is composed of two Sprockets and a Chainset connected to a Brushed DC motor. The driver sprocket has 12-teeth and the driven sprocket (steering shaft) has 60-teeth. This results in a gear ratio of 5.



(b) Steering System within the Robocart Dashboard. The Steering System is mounted to the frame of the Robocart using two hinges.

Figure 6.1: Robocart CAD model of the Mechanical Steering System

poses from our ROS packages and steer away from obstacles seen by the Raspberry Pi cameras.

The next Section 6.1 further explains the software interface for the steering system.

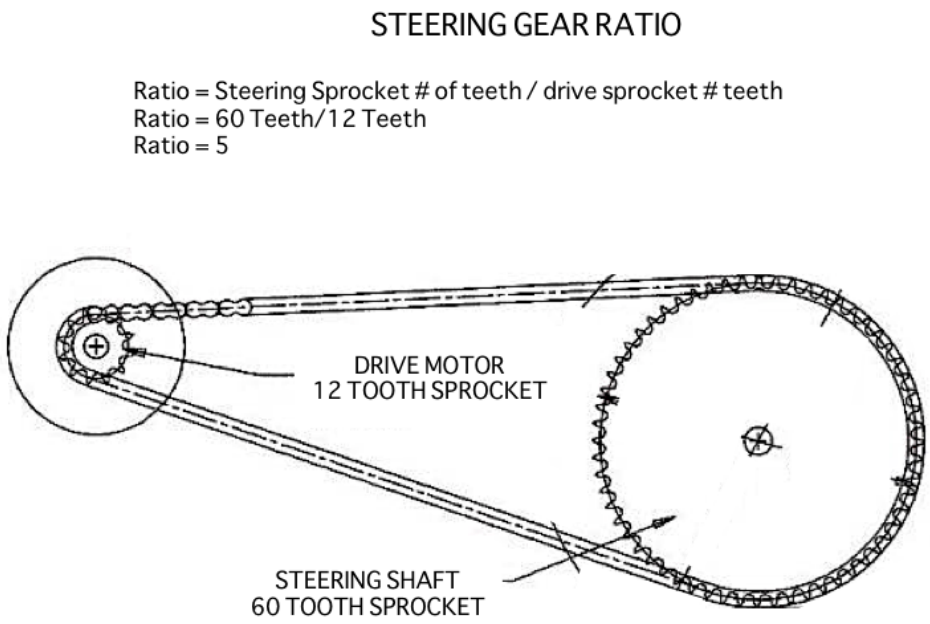


Figure 6.2: As the Brushed Motor turns (drive motor), a 12-tooth sprocket rotates a 60-tooth sprocket attached to the steering column (driven steering shaft).

6.1 Steering System: Software Interface

To communicate with the mechanical steering system, we decided to further utilize built-in capabilities of the Raspberry Pi computers. Each Raspberry Pi computer has a set of GPIO pins that are configured via software to transmit data using serial communication. The transmit data (TXD) and receive data (RXD) GPIOs are wired directly to the chip side of a single Sabertooth (2x60) Motor Controller. The motor-side pins (pins M1A and M2A) of the Sabertooth are wired to the Brushed DC Motor (XYD-13). The wiring diagram for this interface is given in Figure 6.3. Note that the model of the Raspberry Pi computers is represented by the GPIO pins and not the entire Raspberry Pi board.

We chose to use the TXD/RXD GPIO pins versus the Pulse Width Modulation (PWM) pin to allow the Raspberry Pi computers more processing capabilities for vision and object detection. TXD/RXD allow serial data communication, which does not require such a high processing rate as PWM.

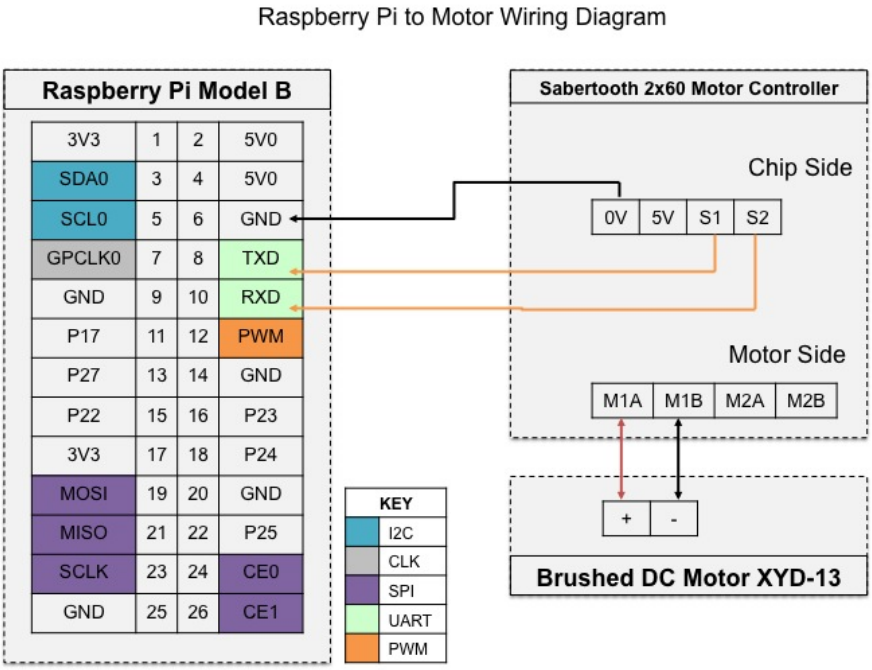


Figure 6.3: The GND, TXD, and RXD GPIO pins from the Raspberry Pi are connected to the Chip Side 0V, S1, S2 pins of the Sabertooth 2x60 Motor controller, respectively. The Motor Side of the Sabertooth 2x60 Motor Controller is connected to the positive and negative terminals of the Brushed DC Motor XYD-13.

6.2 Gazebo Ackermann Steering Model Configuration

Due to Objective E1, we decided to configure Gazebo with an Ackermann Steering cart model. We have assumed a lumped cart model with Ackermann Steering to reduce complexity of translating the Universal Robotic Description Format (URDF) models into Gazebo. To set this model up, we installed ROS Hydro on our local machine. For detailed ROS Hydro installation instructions, visit the ROS wiki at
http://wiki.ros.org/hydro/Installation/Ubuntu.

Next, we cloned the open-source ackermann_vehicle package from its github repository. The ackermann_vehicle package and links to its repository can be found here:
http://wiki.ros.org/ackermann_vehicle. Following that, we compiled our catkin workspace and were able to launch the Gazebo Ackermann Steering model using the following terminal commands:

```
#source the directory your catkin_ws is in
$ source /home/<YOUR_USER_NAME>/catkin_ws/devel/setup.bash
#launch Gazebo including the Ackermann vehicle model
$ roslaunch ackermann_vehicle_gazebo ackermann_vehicle.launch
```

Gazebo will launch including the Ackermann model. Figure 6.4 shows the simulation window. Note that the default Gazebo Grid is turned off in this view.

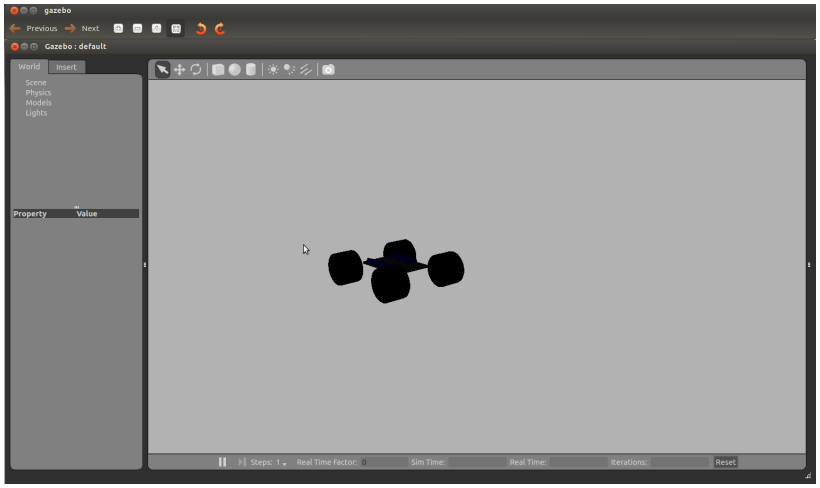


Figure 6.4: This is a screenshot of the Gazebo Simulation window with the Ackermann Vehicle (lumped) model of the Robocart.

6.3 Chapter Summary

Based on the live camera data from the IN-DASH system at the front of the cart, obstacles are detected and the steering system, in simulation, can react in order to navigate the Robocart out of harm's way. The Robocart's obstacle detection accuracy was tested using ROS and RViz, which was needs further testing and analysis due to the quality of the measurements recorded and observed. Chapter 7 provides the conclusions and recommendations as a result of this MQP.

Chapter 7

Conclusions and Recommendations

Overall, I conclude that developing an autonomous ground vehicle from a golf cart is a feasible concept worth continuing. However, because the Robocart is composed of various different subsystems that are interdependent, I determined that it would be worth having separate, more focused, MQPs within each discipline. In this section, I have outlined the five critical areas for future MQPs as well as specific steps to take within those disciplines in order to continuously improve and add value to the next-generation Robocart.

Recommendation #1: *I recommend that there is a Robocart MQP focused on Power Systems Engineering.*

I propose that there is a Robocart MQP that focuses directly on developing an on-board power system. Instead of relying on 120 VAC, the Robocart needs to be retrofitted with a mobile, rechargeable power system that will support all of the current system electronics. I suspect that a future team working on the power system may want to have a 12V drop-down circuit to 5V to power smaller devices such as the Raspberry Pi computers and cameras and any other smaller electronics.

Recommendation #2: *I recommend that there is a Robocart MQP focused on Usability, Safety, and Ergonomics.*

I believe that having a mechanical engineering team focused on designing for usability, er-

gonomics, and safety will add value to the next-generation Robocart because the autonomous vehicle must accommodate various riders comfortably and safely. Mechanical engineering background will come in handy when working with the Robocart because there is a significant amount of design, bodywork, machining, and implementation necessary in order to have a structurally sound vehicle. I also recommend that the mechanical team considers adding in a second row of seating to allow four passengers versus the current maximum of two. One idea that I thought of to allow more passengers is to rotate the current row of seats to face the back and add in a second row to face the current row. This would allow all four people to face each other while riding on the Robocart. Rinspeed's concept car (shown in Figure 7.1) shows a visual of this seating design.

Figure 7.1: Seating Design for Rinspeed's Concept Car: microMAX Urban Vehicle [12]. Rinspeed's microMAX Urban Vehicle is designed with comfort and rotating seating. Front-seat passengers will be able to rotate their seats to face the back-seat passengers while the (autonomous) vehicle drives to the destination [13].

Recommendation #3: *I recommend that there is a Robocart MQP focused on path planning and algorithm development.*

I recommend that the next MQP focuses on algorithms and software development, specifically adding path-planning algorithms to the Robocart server. It is important that the Robocart is able to navigate without colliding into any obstacles. This task can be accomplished through further

software development concentrated on path-planning algorithms such as A-star or wavefront. Path-planning takes into account a predicted total cost of movement and, using this information, will attempt to decide which path is the most profitable both in terms of safety, and efficiency. The Robocart would be further enhanced through the use and implementation of path-planning algorithms.

Recommendation #4: *I recommend that there is a Robocart MQP focused on developing Graphical User Interfaces (GUI).*

I recommend that a group focuses on user input and Graphical User Interfaces (GUI). Usability is an important part of the Robocart, since it is indented to pick up riders and bring them to their destinations. Having a touch screen with user-input capabilities would provide more of a customized experience for the riders. Also, the GUI can be used to display error messages; if something is malfunctioning, the rider would be able to visually see the problem and take manual control of the Robocart.

Recommendation #5: *I recommend that there is a Robocart MQP focused on systems engineering, integration, and testing.*

I recommend that a group focuses purely on coordinating the logistics between each of the above proposed MQPs as well as the complete system design for the Robocart. Systems integration will play an even larger role once the Robocart subsystems are further developed because each subsystem must be compatible with another to ensure complete functionality. I also recommend that the next MQP focuses more on the system-wide testing component by conducting thorough verification and validation of each subsystem separately and combined.

Lastly, I recommend a second systems engineering integration MQP focusing on collaborative, wireless networking between multiple ground and aerial vehicles. I suggest that this project is not undertaken until a single Robocart vehicle works flawlessly with the Robocart server. Creating a collaborative network will require a repository of working software in order to be successful. Collaborative networks of autonomous vehicles can be used to relay information for the military and government applications; airspace data can be transmitted to the land

vehicles, and ground data can be sent to the air vehicles. Refer to Figure 7.2 for an example of this collaborative network.

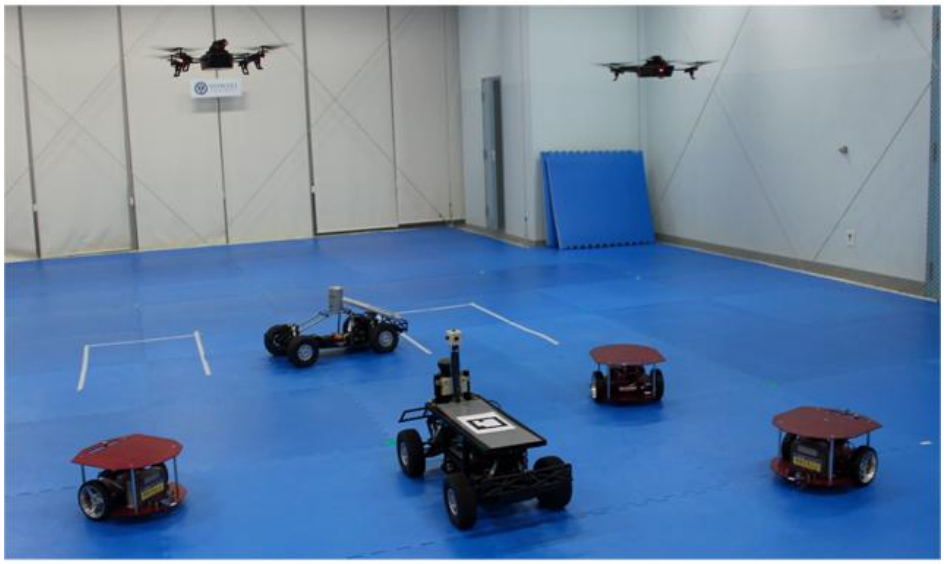


Figure 7.2: An example of a collaborative network consisting of ground and aerial vehicles. Data is relayed from/to the aerial vehicle to/from the ground vehicle to obtain more reliable information and greater accuracy about the vehicles' surroundings [14].

7.1 Limitations of this Research

I realize that this project has various limitations which are identified as follows:

1. This MQP only explored the software and system design of the Robocart. The results are limited to simulation and emulation-based testing, which I hope is further tested with the mechanical system implemented on the Robocart vehicle in the future.

2. The customer needs and functional requirements I developed were based on the team's perspective of the needs and requirements for an autonomous vehicle. Further research, surveys, and focus groups would be helpful to ensure that numerous perspectives are covered in terms of customer needs.

7.2 Potential Uses of the Recommendations

I created these recommendations for use by future advisors and teams working on next-generation Robocart. If these recommendations are considered, I am hoping to see the following improvements to the Robocart system:

- Fully developed and working subsystems within power systems, mechanical engineering, and software that can be integrated to form a robust autonomous ground vehicle.
- A system testing and deployment plan that can be used to satisfy customer needs and functional requirements for the Robocart.
- A user interface that is both safe and ergonomic.
- Safety dispatch messages to users on-board the Robocart through the IN-DASH system.
- Mobile ground vehicle with a reliable on-board power system.
- Real-time data transmission and processing to/from the Robocart vehicle from/to the Robocart server.

References

[1] P. E. Ross, “Driverless Cars: Optional by 2024, Mandatory by 2044 - IEEE Spectrum,” 2014. [Online]. Available: http://spectrum.ieee.org/transportation/advanced-cars/driverless-cars-optional-by-2024-mandatory-by-2044

[2] VisLab. (2015) Automotive. [Online]. Available: http://vislab.it/automotive/

[3] U. H. of Representatives, “How Autonomous Vehicles Will Shape the Future of Surface Transportation,” p. 1, 2013. [Online]. Available: http://transport.house.gov/calendar/eventsingle.aspx?EventID=357149

[4] A. Stoklosa, “California attempts to wade into the uncharted waters of autonomous-car regulation,” *Car and Driver*, 03 2014. [Online]. Available: http://blog.caranddriver.com/california-attempts-to-wade-into-the-uncharted-waters-of-autonomous-car-regulation/

[5] ——, “Google shows off how its autonomous vehicles aren‘t killing cyclists or hitting parked cars,” *Car and Driver*, 04 2014. [Online]. Available: http://blog.caranddriver.com/google-shows-off-how-its-autonomous-vehicles-arent-killing-cyclists-or-hitting-parked-cars/

[6] C. Atiyeh, “European manufacturers leading r&d for autonomous cars we may actually want to drive,” *Car and Driver*, 02 2014. [Online]. Available: http://blog.caranddriver.com/european-manufacturers-leading-rd-for-autonomous-cars-we-may-actually-want-to-drive/

[7] U. D. of Transportation. (2013, 08) Model systems engineering documents for adaptive signal control technology (asct) systems. [Online]. Available: http://ops.fhwa.dot.gov/publications/fhwahop11027/sec_b.htm

[8] "Characteristics of Fifth-Wheel (Wagon Steer) Steering Page," in *Engineering Design Handbook - Automotive Series - Automotive Suspensions: (AMCP 706-356)*. U.S. Army Materiel Command, Nov. 2012. [Online]. Available: http://app.knovel.com/web/view/swf/show.v/rcid:kpEDHASAS1/cid:kt00AC8JW7/viewerType:pdf/r design-handbook-18?cid=kt00AC8JW7&page=4&b-q=ackermann steering&sort_on=default&b-subscription=TRUE&b-group-by=true&b-sort-on=default&q=ackermann steering

[9] Minidoodle. Gear ratio. [Online]. Available: http://jleibovitch.tripod.com/id111.htm

[10] J. L. Kent, "Driverless van crosses from Europe to Asia," 2010. [Online]. Available: http://edition.cnn.com/2010/TECH/innovation/10/27/driverless.car/index.html?iref=allsearch

[11] J. Fingas, "Self-driving vehicles and robotic clerks could take your job in 20 years," 03 2015. [Online]. Available: http://www.engadget.com/2015/03/08/robots-may-take-more-jobs/

[12] J. Holloway, "Rinspeed releases details of micromax swarm car concept," *Gizmag*, 02 2013. [Online]. Available: http://www.gizmag.com/rinspeed-micromax/26392/

[13] C. Weiss, "Rinspeed shows what the self-driving car will be like to ride in," *Gizmag*, 12 2013. [Online]. Available: http://www.gizmag.com/rinspeed-self-driving-concept/30104/

[14] J. H. Kim, "Multi-uav-based stereo vision system without gps for ground obstacle mapping to assist path planning of ugv read more at:

http://phys.org/news/2014-09-air-ground-based-robot-vehicles.html#jcp," *Electronics Letters*, vol. 50, no. 20, pp. 1431–1432, 09 2014. [Online]. Available: http://phys.org/news/2014-09-air-ground-based-robot-vehicles.html

[15] J. Schmiduber, "Professor Schmidhuber's Highlights of Robot Car History," 2011. [Online]. Available: http://people.idsia.ch/ juergen/robotcars.html

[16] L. Martin, "Driving Forces: Lockheed Martin's Autonomous Land Vehicles," 2012. [Online]. Available: http://www.lockheedmartin.com/us/100years/stories/alv.html

[17] M. Novak, "DARPA Tried to Build Skynet in the 1980s," 2013. [Online]. Available: http://paleofuture.gizmodo.com/darpa-tried-to-build-skynet-in-the-1980s-1451000652

[18] D. Pomerleau, "RALPH: Rapidly Adapting Lateral Position Handler," *IEEE Symposium*, no. September 25-26, 1995, 1995. [Online]. Available: http://www.cs.cmu.edu/ tjochem/nhaa/ralph.html

[19] J. Davis, "Say Hello to Stanley," 2006. [Online]. Available: http://archive.wired.com/wired/archive/14.01/stanley.html?pg=1&topic=stanley&topic_set=

[20] DARPA, "DARPA Grand Challenge 2005," 2005. [Online]. Available: http://archive.darpa.mil/grandchallenge05/gcorg/

[21] C. Dobby, "Nevada state law paves the way for driverless cars," 2011. [Online]. Available: http://business.financialpost.com/2011/06/24/nevada-state-law-paves-the-way-for-driverless-cars/?__lsa=e443-35b3

[22] E. Guizzo, "How Google's Self-Driving Car Works," 2011. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works

[23] L. Gannes, “Here’s What It’s Like to Go for a Ride in Google’s Robot Car,” 2014. [Online]. Available: http://recode.net/2014/05/13/googles-self-driving-car-a-smooth-test-ride-but-a-long-road-ahead/

[24] A. Davies and A. Gallery, “Self-driving cars will make us want fewer cars,” 03 2015. [Online]. Available: http://www.wired.com/2015/03/the-economic-impact-of-autonomous-vehicles/

[25] “Volvo Car Group‘s first self-driving Autopilot cars test on public roads around Gothenburg - Volvo Car Group Global Media Newsroom.” [Online]. Available: https://www.media.volvocars.com/global/en-gb/media/pressreleases/145619/volvo-car-groups-first-self-driving-autopilot-cars-test-on-public-roads-around-gothenburg

[26] “Automated Driving Applications and Technologies for Intelligent Vehicles - AdaptIVe FP7 project- Automated Driving Applications and Technologies for Intelligent Vehicles.” [Online]. Available: http://www.adaptive-ip.eu/

[27] “Systems Engineering Management,” in *Systems Engineering Fundamentals*. U.S. Department of Defense, Jun. 2001. [Online]. Available: http://app.knovel.com/web/view/swf/show.v/rcid:kpSEF00001/cid:kt00TYSBI1/viewerType:pdf/roc engineering-fundamentals?cid=kt00TYSBI1&page=2&b-q=system engineering&sort_on=default&b-subscription=TRUE&b-group-by=true&b-search-type=tech-reference&b-sort-on=default&b-toc-cid=kpSEF00001&b-toc-root-slug=systems-engineering-fundamentals&b-toc-url-slug=purpose&b-toc-title=Systems Engineering Fundamentals

[28] M. Bertozzi, A. Broggi, M. Cellario, A. Fascioli, P. Lombardi, and M. Porta, “Artificial vision in road vehicles,” *Proceedings of the IEEE*, vol. 90, no. 7, 2002.

[29] H. Fountain, “Yes, Driverless Cars Know the Way to San Jose,” 2012. [Online]. Available: http://www.nytimes.com/2012/10/28/automobiles/yes-driverless-cars-know-the-way-to-san-jose.html?pagewanted=1&_r=0

[30] A. Goncalves and S. Joao, “Low Cost Sensing for Autonomous Car Driving in Highways.” [Online]. Available: http://welcome.isr.ist.utl.pt/img/pdfs/1663_hans_icinco07.pdf

[31] L. Hardesty, “Think Fast, Robot,” 2014. [Online]. Available: http://newsoffice.mit.edu/2014/think-fast-robot-0530

[32] R. I. Hartley and P. Sturm, “Triangulation,” *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146–157, 1997. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1077314297905476

[33] A. Heyden and M. Pollefeys, “Multiple view geometry,” in *Emerging Topics in Computer Vision*, 2005, pp. 45–107.

[34] A. Iliafar, “LIDAR, Lasers, and Logic: Anatomy of an Autonomous Vehicle,” 2013. [Online]. Available: http://www.digitaltrends.com/cars/lidar-lasers-and-beefed-up-computers-the-intricate-anatomy-of-an-autonomous-vehicle/

[35] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, “Cloud-based robot grasping with the Google object recognition engine,” *2013 IEEE International Conference on Robotics and Automation*, pp. 4263–4270, May 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6631180

[36] T. Lassa, “The Beginning of the End of Driving,” 2013. [Online]. Available: http://www.motortrend.com/features/auto_news/2012/1301_the_beginning_of_the_end_of_driving/

[37] D.-J. Lee, J. Archibald, X. Xu, and P. Zhan, “Using distance transform to solve real-time machine vision inspection problems,” *Machine Vision and Applications*, vol. 18, no. 2,

pp. 85–93, Nov. 2006. [Online]. Available:
http://link.springer.com/10.1007/s00138-006-0050-2

[38] D. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999.

[39] M. de Paula, "Autonomous driving," *Popular Science*, May 2014. [Online]. Available: http://www.popsci.com/blognetwork/tags/autonomous-driving

[40] S. J. D. Prince, "Computer vision : models , learning and inference," 2012.

[41] P. Stenquist, "On the Road to Autonomous, a Pause at Extrasensory," 2013. [Online]. Available: http://www.nytimes.com/2013/10/27/automobiles/on-the-road-to-autonomous-a-pause-at-extrasensory.html?pagewanted=all

[42] D. Vernon, "An Optical Device for Computation of Binocular Stereo Dispairity with a Single Static Camera," in *Opto-Ireland 2002: Optical Metrology, Imaging, and Machine Vision*, vol. 38, 2003.

[43] D. L. Baggio, *Mastering OpenCV with Practical Computer Vision Projects*, 2012.

[44] "Feature Matching — OpenCV 3.0.0-dev documentation." [Online]. Available: http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

[45] F. Brunet. (2011, 07) First definitions and concepts. [Online]. Available: http://www.brnt.eu/phd/node16.html

[46] J. V. Does. (2014, 08) Openmvg photo reconstruction. [Online]. Available: http://blog.htmlfusion.com/openmvg/

[47] D. Bray. (2014, 09) Getting a raspberry pi on worcester polytechnic institute (wpi) wifi (wpa-eap). esologic.com. [Online]. Available: http://www.esologic.com/?p=1088

Appendix A

Bill of Materials: Electronics & Motors

Model	Description	Material	Quantity	Location on Robocart
XYD-13	Brushed DC Motor	Magnet	1	Steering System
Sabertooth 2x60	Motor Controller	PCB	1	Steering System
Raspberry Pi Model B	Computer with GPIO capabilities	PCB	2	IN-DASH
Raspberry Pi Camera	Camera compatible with Raspberry Pi Model B	PCB	2	IN-DASH

Appendix B

Configuring Raspberry Pi to WPI-Wireless

These instructions were developed and modified from [47].

B.1 Modify /etc/network/interfaces on the Raspberry Pi to accept a wired and wireless connection

First, we modified two files within the root directory of each Raspberry Pi. In a terminal window type the following:

```
$ sudo bash #assume you have admin permissions to root
$ nano /etc/network/interfaces
```

Then use this code to allow both the wireless (wlan0) and ethernet (eth0) connections to scan for networks.

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
iface wlan0 inet manual
```

```
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

We resaved the interfaces file with the above changes and the Raspberry Pi recognized all available wireless networks.

To connect to a network, we needed to set up the wpa_supplicant file.

B.2 Configure the wpa_supplicant file

We opened another root terminal and edited the wpa_supplicant.conf file using the following commands:

```
$ sudo bash #assume you have admin permissions to root
$ nano /etc/wpa_supplicant/wpa_supplicant.conf
```

The wpa_supplicant file was then configured with the following lines of code:

B.3 Connect to WPI-Wireless using Raspberry Pi

Due to the high security and regulations of WPI-Wireless, it was difficult to obtain internet access on WPI's campus using Raspberry Pi. Using the WPI-Wireless setup as a guide, we manually configured each Raspberry Pi with the certificates and authentication needed in order to successfully connect to WPI-Wireless.Before starting these steps, visit WPI's netreg page to register the MAC address of the Raspberry Pi.

We used the following configuration to connect both Raspberry Pi computers to WPI-Wireless:

1. Configure the correct Timezone, Date, and Time on the Raspberry Pi.

   ```
   $ sudo raspi-config
   ```

   Select Internationalization Options and follow the prompts

2. Create a directory (non-root) to store the certs.

```
$ mkdir /home/pi/certs
```

3. Download the Certificate files from https://wpi-wireless-setup.wpi.edu/ and move them into the certs directory.

4. Use OpenSSL to convert the certificate.p12 file to a certificate.pem file.

```
$ openssl pkcs12 -in /home/pi/certs/certificate.p12 -out /home/pi/certs/ce
```

When prompted for a password, use your WPI account password.

5. Setup the /etc/network/interfaces file to look like the following:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
```

6. Open root and navigate to the wpa_supplicant.conf file.

```
$ sudo bash
# enter root password if prompted
$ nano /etc/wpa_supplicant/wpa_supplicant.conf
```

7. Configure the /etc/wpa_supplicant/wpa_supplicant.conf file to look like the following:

```
#ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
#update_config=1

network={
 ssid="WPI-Wireless"
 key_mgmt=WPA-EAP
 proto=WPA2
```

```
    pairwise=CCMP
    group=CCMP
    eap=TLS

    identity="[YOUR_WPI_EMAIL]@wpi.edu"

    ca_cert="/home/pi/certs/CA-[A_BUNCH_OF_NUMBERS].pem"
    client_cert="/home/pi/certs/certificate.pem"
    private_key="/home/pi/certs/certificate.p12"
    private_key_passwd="[YOUR_WPI_EMAIL_PASSWORD]"

    priority=1
}
```

8. Configure the bootup to initialize the wireless dongle each time.

   ```
   $ crontab -e
   ```

   Scroll to the end of the file and add the following lines:

   ```
   @reboot sudo wpa_supplicant -c/etc/wpa_supplicant/wpa_supplicant.conf
   -i wlan0
   @reboot sudo /sbin/dhclient wlan0
   ```

   Save the file and exit.

9. Reboot the Raspberry Pi. If it is configured correctly, then WPI-Wireless will be recognized within three to five minutes after reboot.

Appendix C

Accessing the Robocart Server:

Connection Instructions

Before attempting to connect to the Robocart server, please contact robocart@wpi.edu or Elizabeth Miller (eamiller@wpi.edu) to obtain the Robocart server password.

C.1 Secure Shell (SSH) using PuTTY on Windows

These instructions are for connecting to the Robocart server using PuTTY on Windows computers. We assume that you already have PuTTY installed on your machine.

1. Launch PuTTY
2. On the top, left-hand side, click Session
3. Find the category: “Host Name (or IP Address)" (See Figure C.1 and type in the following:

   robocart@mqpburris.wpi.edu

4. At the bottom right, click “Open"
5. You may be asked to verify the host key. Click “Yes” to continue. See Figure C.2.

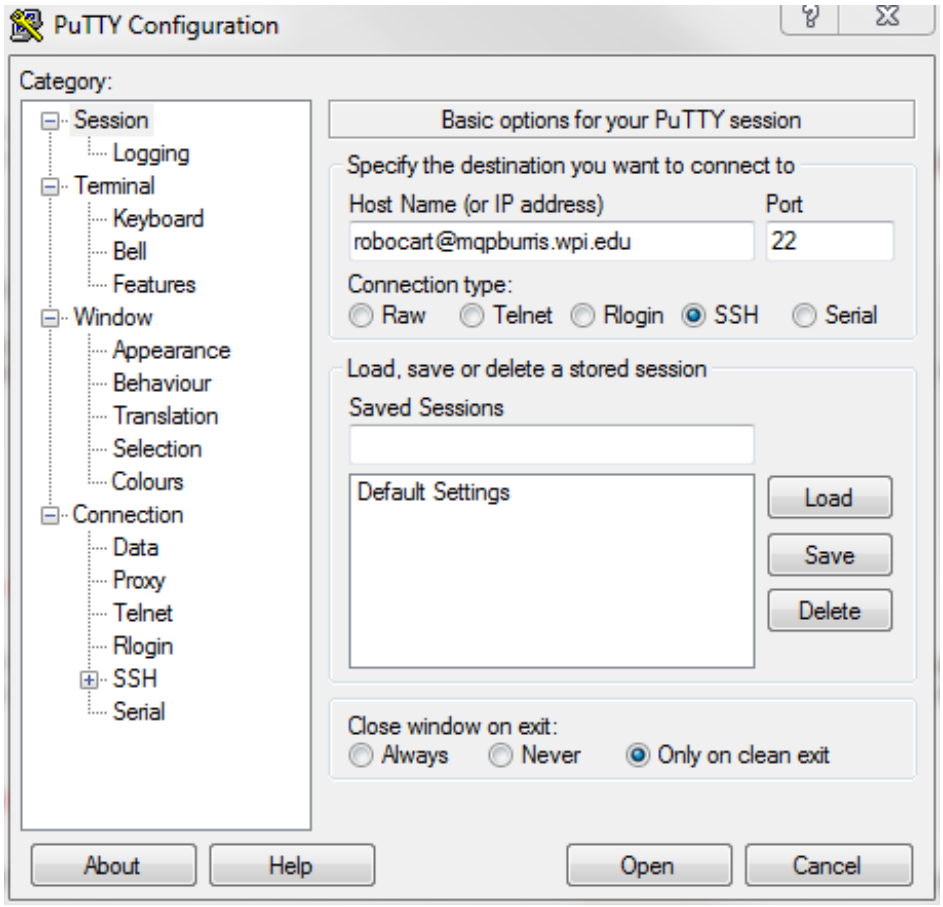


Figure C.1: This is a screen shot of the PuTTY window where you type in the host name (robocart@mqpburris.wpi.edu).

6. You will be prompted to enter in the Robocart server password

7. Type the password and hit enter. You are connected to the Robocart server when you see the following output. Refer to Figure C.3.

```
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic x86_64)
```

To exit the server, type "exit" and hit enter.

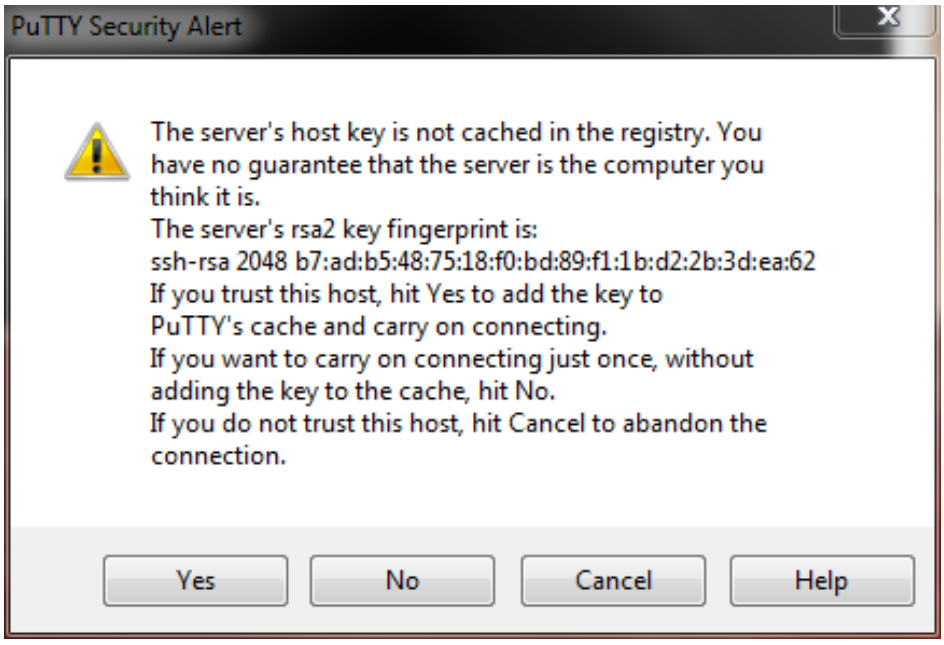


Figure C.2: This is a screen shot of the PuTTY window when asked to verify the host key.

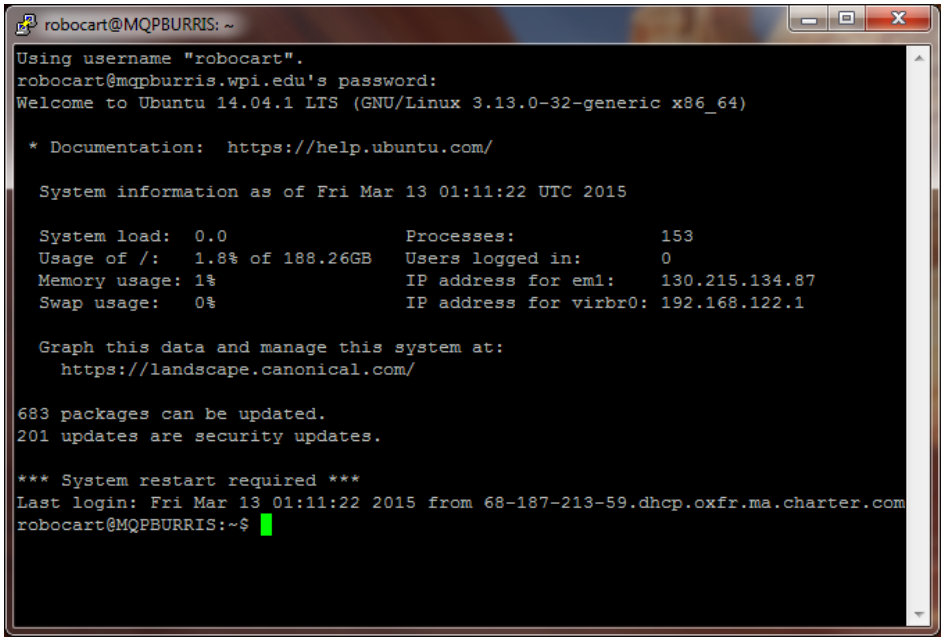


Figure C.3: This is a screen shot of the terminal window when connected to the Robocart Server on Windows7.

C.2 Secure Shell (SSH) using Terminal for Linux/Mac OS

These instructions are for connecting to the Robocart server using PuTTY on Linux and Mac computers.

1. Launch Terminal
2. Type the following:

```
$ ssh robocart@mqpburris.wpi.edu
```

3. You may be asked to verify the server's host key. Type in 'y' or 'yes' to continue.
4. You will be prompted to enter in the Robocart server password; the output is given as follows:

```
robocart@mqpburris.wpi.edu's password:
```

5. Type the password and hit enter. You are connected to the Robocart server when you see the following output:

```
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic x86_64)
```

Once you have completed these steps, your terminal window should look like the screen shot given in Figure C.4 for Linux and Figure C.5 for Mac OS.

To exit the server, type "exit" and hit enter.

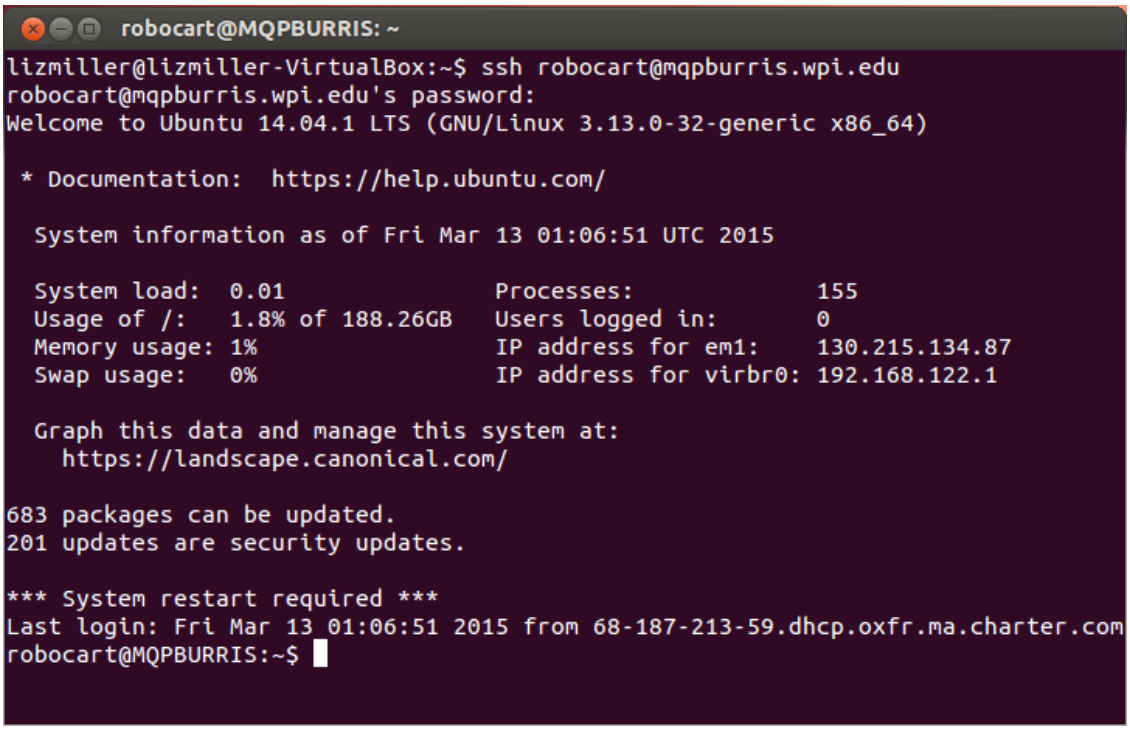


Figure C.4: This is a screen shot of the terminal window when connected to the Robocart Server on Linux.

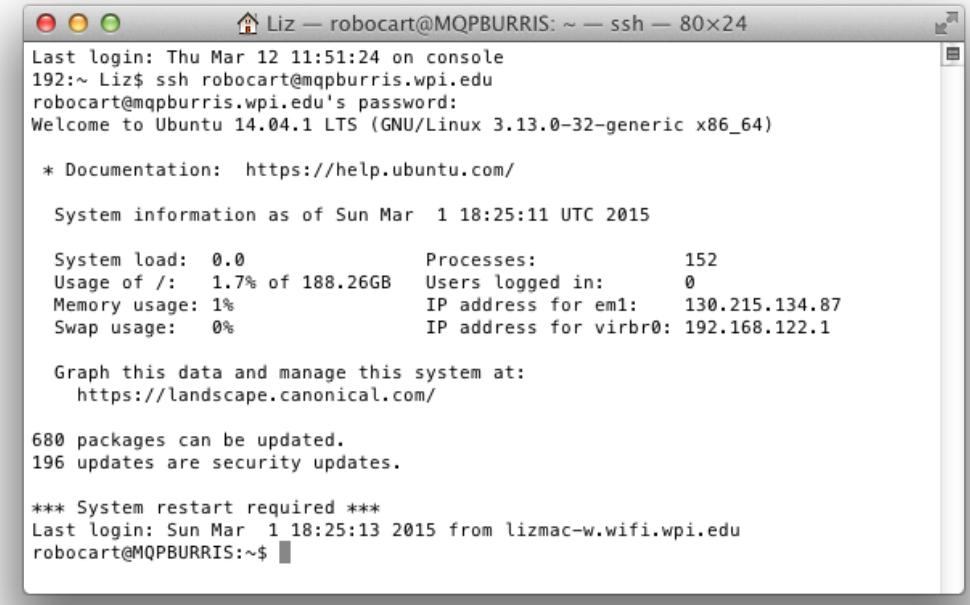


Figure C.5: This is a screen shot of the terminal window when connected to the Robocart Server on Mac OS.