Worcester Polytechnic Institute
Digital WPI

Major Qualifying Projects (All Years) | Major Qualifying Projects

April 2014

PARbot: Personal Assistive Robot

Jeffrey Francis Orszulak
Worcester Polytechnic Institute

Julien Henri Van Wambeke-Long
Worcester Polytechnic Institute

Kevin Lucas Burns
Worcester Polytechnic Institute

Nikhil Shyam Godani
Worcester Polytechnic Institute

Olivia Josephine Hugal
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/mqp-all

Repository Citation
Orszulak, J. F., Van Wambeke-Long, J. H., Burns, K. L., Godani, N. S., & Hugal, O. J. (2014). *PARbot: Personal Assistive Robot*. Retrieved from https://digitalcommons.wpi.edu/mqp-all/1897

This Unrestricted is brought to you for free and open access by the Major Qualifying Projects at Digital WPI. It has been accepted for inclusion in Major Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.

Worcester Polytechnic Institute

PARbot:

A Personal Assistive Robot

Authors:

Kevin Burns

Nikhil Godani

Olivia Hugal

Jeffrey Orszulak

Julien Van Wambeke-Long

Keywords:

WPI Robotics Engineering

Robot Operating System

Assistive Technology

SLAM

Computer Vision

PARbot:
A Personal Assistive Robot

A Major Qualifying Project Report
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfilment of the requirements for the
Degree of Bachelor of Science
By

Kevin Burns

Nikhil Godani

Olivia Hugal

Jeffrey Orszulak

Julien Van Wambeke-Long

Date: April 30, 2014

Report Submitted to:

Professor Taskin Padir, Advisor, WPI

This report represents the work of five WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its website without editorial or peer review.

Abstract

The aging population of the United States is creating a growing need to provide assistive care for elderly and people with disabilities. As the Baby Boomer generation enters retirement, the ratio of caregivers to those that require assistance is projected to decrease [1]. There are currently no commercially available modular assistive robots that can fill this need. Our project aims to provide an alternative to current assisted living options through the development, construction, and testing of a Personal Assistive Robot (PARbot) that allows individuals with general or age related disabilities to maintain some aspects of their independence, such as the ability to shop.

Our unique solution implements a design that allows user oriented customization. Modularity is a key component in the design to allow for future expansion and potential user customization. The robot will be designed to ADA specifications to ensure that it can operate anywhere the user desires. Human Robot Interaction (HRI) will be an important aspect in our project; users should feel comfortable in the presence of the robot. Ultimately, PARbot will be capable of navigating in public areas, such as a grocery store, and utilize a Simultaneous Localization and Mapping (SLAM) algorithm for navigation while tracking its user.

Acknowledgements

We would like to thank our advisor Taskin Padir for his support through the project. Further we would like to thank the following members of the WPI community: Tracey Coetzee, Joe St. Germain, Velin Dimitrov, Dmitry Sinyukov, Fred Looft, Cameron Canale, Devon Locke and Michelle Gagnon. We would also like to thank those who supported the project financially, including the National Science Foundation, Worcester Polytechnic Institute, and the Cornell Cup as presented by Intel.

Table of Authorship

Abstract	Jeffrey
Acknowledgments	Jeffrey
1 Background	Jeffrey
1.1 Need for Robotic Aids	Jeffrey
1.2 Current Technology	Olivia
1.3 Robotic Bases	Julien
1.3.1 Mobile Robot MP-500	Julien
1.3.2 PeopleBot	Julien
1.3.3 Pioneer P3-DX	Julien
1.4 ADA Guidelines	Jeffrey
1.5 Hardware	Kevin
1.5.1 Electronics	Kevin
1.5.2 Batteries	Kevin
1.5.3 Sensors	Nikhil
1.5.3.1 Accelerometers	Nikhil
1.5.3.2 Gyroscopes	Nikhil
1.5.3.3 Ultrasonic Sensors	Nikhil
1.5.3.4 Cliff Detection Sensors	Nikhil
1.5.3.5 Rotary Encoders	Nikhil
1.5.3.6 Microphone and Speakers	Nikhil
1.5.3.7 LiDAR	Julien
1.5.3.8 RGB-D	Julien
1.5.3.9 Intel Galileo – Arduino	Nikhil
1.6 Robot Operating System (ROS)	Kevin
1.6.1 Groovy Galapagos	Kevin
1.6.2 Hydro Medusa	Kevin
1.7 Simultaneous Localization and Mapping	Julien
1.7.1 Localization	Julien
1.7.1.1 Particle Filter	Olivia
1.7.1.2 Kalman Filter	Olivia
1.7.2 Mapping	Olivia
1.7.3 QR Code Tracking	Nikhil
1.8 Path Planning	Olivia

1.8.1 Occupancy Grid	Olivia
1.8.2 Search Methods	Jeffrey
1.8.2.1 Breadth First Search	Olivia
1.8.2.2 Depth First Search	Olivia
1.8.2.3 Dijkstra's Algorithm	Olivia
1.8.2.4 A*	Kevin
1.9 Cornell Cup Competition as Presented by Intel	Jeffrey
2 Design Requirements and Specifications	Jeffrey
2.1 Stakeholders	Jeffrey
2.2 Development of Specifications	Jeffrey
2.2.1 Boston Abilities Expo	Jeffrey
2.2.2 Personas	Nikhil
2.3 Resulting Requirements	Jeffrey
3 Design and Analysis	Devon
3.1 Drivetrain	Jeffrey
3.2 Chassis Design	Jeffrey
3.3 Shopping Cart Module	Kevin
3.4 Onboard Electronics	Julien
3.4.1 Battery Selection	Jeffrey
3.4.2 Motor Selection	Kevin
3.4.3 Voltage Converters	Julein
3.5 Onboard Computer	Olivia
3.5.1 Communication	Julien
3.5.2 Networking	Jeffrey
3.5.3 Analog to Digital Conversion	Julien
3.5.4 Sensors: Navigation	Julien
3.5.5 Inertial Measurement Unit	Olivia
3.5.6 Cliff Detection	Nikhil
3.5.7 Motor Controller	Julien
3.6 Graphical User Interface (GUI)	Devon
3.7 Localization and Mapping: Navigation Sensor Selection	Julien
3.8 Tracking System	Kevin
3.9 Software	Kevin
3.9.1 Path Planning Software Development	Jeffrey

3.9.2 ROS Software Structure	Jeffrey
3.9.2.1 PARbot_Vision.launch	Olivia
3.9.2.2 PARbot_pose_transformer	Jeffrey
3.9.2.3 PARbot_search	Jeffrey
3.9.2.4 Target_sim	Olivia
3.9.2.5 Parbot_gmapping.launch	Olivia
3.9.2.6 Motion_Planning	Kevin
3.9.2.7 User Tracking	Olivia
3.9.2.7.1 QR Code Tracking	Olivia
3.9.2.7.2 Pixy Tracking	Nikhil
4 Results	Jeffrey
4.1 Foldability	Julien
4.2Modularity	Olivia
4.2.1 Hardware	Olivia
4.2.2 Software	Olivia/Kevin
4.2.3 Graphical User Interface (GUI)	Devon
4.3 Power Capacity and Consumption	Julien
4.4 Cliff Detection	Nikhil
4.5 Mapping and Path Planning	Jeffrey
4.6 User Tracking	Kevin
4.7 Motion Planning	Kevin
5 Conclusions	Jeffrey
6 Future Recommendations	Jeffrey
6.1 Ubuntu 14.04	Jeffrey
6.2 Win_ROS	Jeffrey
6.3 Android	Jeffrey
8 Appendix A: Personas	Jeffrey
8.1 Persona 1	Kevin
8.2 Persona 2	Nikhil
8.3 Persona 3	Olivia
8.4 Persona 4	Jeffrey
8.5 Persona 5	Julien
8.6 Persona 6	Devon
9 Appendix B: Development on PARbot	Kevin

9.1 Connecting to PARbot Over the Wifi Bridge	Kevin
9.2 Moving file to and from PARbot	Kevin

Table of Contents

Abstract 3

Acknowledgements 4

Table of Authorship 5

List of Figures 13

Table of Tables 15

1 Background 16

1.1 Need for Robotic Aids 16

1.2 Current Technology 18

1.3 Robotic Bases 18

1.3.1 Mobile robot MP-500 18

1.3.2 PeopleBot 19

1.3.3 Pioneer P3-DX 19

1.4 ADA Guidelines 19

1.5 Hardware 20

1.5.1 Electronics 20

1.5.2 Batteries 22

1.5.3 Sensors 22

1.5.3.1 Accelerometers 22

1.5.3.2 Gyroscopes 23

1.5.3.3 Ultrasonic Sensors 23

1.5.3.4 Cliff Detection Sensors 23

1.5.3.5 Rotary Encoders 24

1.5.3.6 Microphone and Speakers 24

1.5.3.7 LIDAR 24

1.5.3.8 RGB-D 24

1.5.3.9 Intel Galileo – Arduino 25

1.6 Robot Operating System (ROS) 25

1.6.1 Groovy Galapagos 26

1.6.2 Hydro Medusa 26

1.7 Simultaneous Localization and Mapping (SLAM) 26

1.7.1 Localization 26

1.7.1.1 Particle Filter 26

1.7.1.2 Kalman Filter 27
1.7.2 Mapping 27
1.7.3 QR Code Tracking 27
1.8 Path Planning 28
1.8.1 Occupancy Grid 28
1.8.2 Search Methods 29
1.8.2.1 Breadth First Search 29
1.8.2.2 Depth First Search 29
1.8.2.3 Dijkstra's Algorithm 30
1.8.2.4 A* 30
1.9 Cornell Cup Competition as Presented by Intel 31
2 Design Requirements and Specifications 32
2.1 Stakeholders 32
2.2 Development of Specifications 32
2.2.1 Boston Abilities Expo 32
2.2.2 Personas 33
2.3 Resulting Requirements 33
3 Design and Analysis 34
3.1 Drivetrain 34
3.2 Chassis Design 34
3.3 Shopping Cart Module 35
3.4 Onboard Electronics 35
3.4.1 Battery Selection 36
3.4.2 Motor Selection 36
3.4.3 Voltage Converters 37
3.5 Onboard Computer 37
3.5.1 Communication 38
3.5.2 Networking 39
3.5.3 Analog to Digital Conversion 39
3.5.4 Sensors: Navigation 39
3.5.5 Inertial Measurement Unit (IMU) 39
3.5.6 Cliff Detection 40
3.5.7 Motor Controller 47
3.4.11 Wiring diagram 48

3.4.12 Safety features 49
3.6 Graphical User Interface (GUI) 50
3.7 Localization and Mapping: Navigation Sensor Selection 51
3.8 Tracking System 52
3.9 Software 52
3.9.1 Path Planning Software Development Process 53
3.9.2 ROS Software Structure 55
3.9.2.1 Parbot_Vision.launch 55
3.9.2.2 PARbot_pose_transformer 58
3.9.2.3 PARbot_search 58
3.9.2.4 Target_sim 58
3.9.2.5 PARbot_gmapping.launch 59
3.9.2.6 Motion Planning 59
3.9.2.7 User Tracking 62
4 Results 65
4.1 Foldable 65
4.2 Modularity 65
4.2.1 Hardware 66
4.2.2 Software 67
4.2.3 Graphical User Interface (GUI) 67
4.3 Power Capacity and Consumption 68
4.4 Cliff Detection 69
4.5 Mapping and Path Planning 69
4.6 User Tracking 70
4.7 Motion Planning 72
5 Conclusions 74
6 Future Recommendations 76
6.1 Ubuntu 14.04 76
6.2 Win_ROS 76
6.3 Android 76
7 References 77
8 Appendix A: Personas 81
8.1 Persona 1: 81
8.2 Persona 2: 82

8.3 Persona 3: 83
8.4 Persona 4: 84
8.5 Persona 5: 85
8.6 Persona 6: 86
9 Appendix B: Development on PARbot 87
9.1 Connecting to PARbot Over the WiFi Bridge: 87
9.2 Moving files to and from PARbot 87

List of Figures

Figure 1: Caregiver Support Ratio, United States 16
Figure 2: Projected Age 80+ Population, U.S. 17
Figure 3: MP-500 Robotic Base 18
Figure 4: PeopleBot Robotic Base 19
Figure 5: Pioneer P3-DX 19
Figure 6: Intel Galileo for Arduino Environment 25
Figure 7: Image of a QR Code 28
Figure 8: Breadth First Search Examination Order 29
Figure 9: Depth Breadth First Search Examination Order 30
Figure 10: Example of a Planned Path 31
Figure 11: Data Connections Map 38
Figure 12: Cliff Sensor Mount 41
Figure 13: Cliff Sensor Arrangement 41
Figure 14: Cliff Detection Calibration Best Fit Graph 43
Figure 15: Top Down View of Cliff Detection Test Plate 45
Figure 16: Cliff Detection Sensor Mounts 46
Figure 17: MDC2230 Motor Controller 47
Figure 18: Power Connections Map 48
Figure 19: 120 Amp Switch 49
Figure 20: GIGAVAC Contactor Switch 49
Figure 21: Emergency Stop Button 50
Figure 22: Touchscreen GUI 51
Figure 23: Simulated Development Environment 53
Figure 24: Software Flow Overview 55
Figure 25: PrimeSenses Mounted on PARbot 56
Figure 26: PrimeSense Transformation Data and Representation 57
Figure 27: depthimage data produced by primesenses 57
Figure 28: Five Velocity Sets with Ten Tenticles Each 59
Figure 29: example of tentacle based motion planning 60
Figure 30: THE TENTACLE SELECTED BASED ON THE BLUE TARGET PATH AND THE PURPLE OBSTACLES 61
Figure 31: cost cross-section 61
Figure 32: Pixy Connected to Arduino Microcontroller 63
Figure 33: Object Tracking Color Pattern 64
Figure 34: Open and Closed Configuration 65

Figure 35: Power Connector Pin Out 66
Figure 36: Mounting Plate 67
Figure 37: Cliff Detection Data Stream 69
Figure 38: Atwater Kent Second Floor Map with Path 70
Figure 39: QR Tracking Test 71
Figure 40: PIXY color tracking 71
Figure 41: Stamped Pose from PIXY 72

Table of Tables

Table 1: Brushed vs Brushless....36
Table 2: Recorded Cliff Detection Data....42
Table 3: Cliff Detection Pair Value Comparison....43
Table 4: Cliff Detection States....44
Table 5: Cliff Detection Test Plate Wiring....45
Table 6: Code Runtimes for 512x512 Element Grid....55
Table 7: Transformer Power Calculations....68
Table 8: Motor Power Calculations....68

1 Background

The decreasing cost of electronics coupled with an increase in processing power has allowed for the development of robots for applications outside of production lines. Robots for simple, repetitive tasks such as vacuuming and mowing the lawn have begun to enter everyday life. More complex applications are cropping up in the automotive industry, for example, cars that can parallel park themselves are becoming more common. Few commercialized developments have been made in the form of personal assistive robots. However, a potential consumer base is expected to expand greatly in the coming decades.

1.1 Need for Robotic Aids

The age demographic of the United States is changing; there will be an increase in the percentage of elderly individuals (70 and older) expected over the next 40 years. Based on 2010 census data collected, 13 percent of the US population is over the age of 65 [1]. However, as the number of elderly people in the United States increases, the number of caregivers, especially familial caregivers, is expected to remain the same due to changes in the average size of the American family. Most familial caregivers are between 45 and 65 and hold a full time job outside the home in addition to helping their elderly relatives. As a result, the ratio of demand for Long-Term Services and Support (LTSS) to available caregivers will increase. Furthermore, in the next 10 to 15 years, the Baby Boomer generation will pass into the 75 to 85 year old age bracket, greatly increasing the need for caregivers [2].

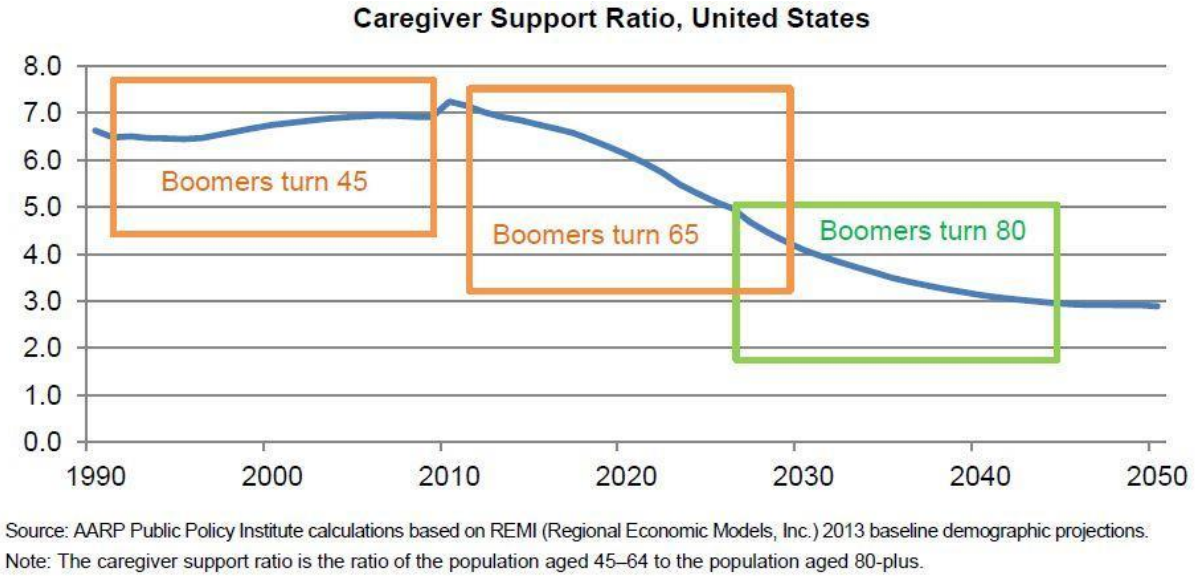

FIGURE 1: CAREGIVER SUPPORT RATIO, UNITED STATES

From 1990 to 2010, the Baby Boomer generation shouldered the majority of the caregiving role, providing support for their parents. This increased the caregiver to need ratio from 6.6:1 in 1990 to 7.2:1 in 2010, allowing for the care of the elderly to be divided between more of the population. As can be seen in Figure 1, from 2010 onward the caregiver to need ration is expected to steadily decrease as the role of caregiver passes from the Baby Boomer generation to their children. In 2030, the caregiver to need ratio is expected to be 4.1:1 and will drop to 2.9:1 by the year 2050. Figure 2 shows the percentage

of Americans that will be in the 80 years old or older category as time advances. As a result of the change in the American family structure, the elderly population will be less able to lean on their children for support as they age, meaning the need, and therefore cost, for external aid will increase in the coming 30 years [2].

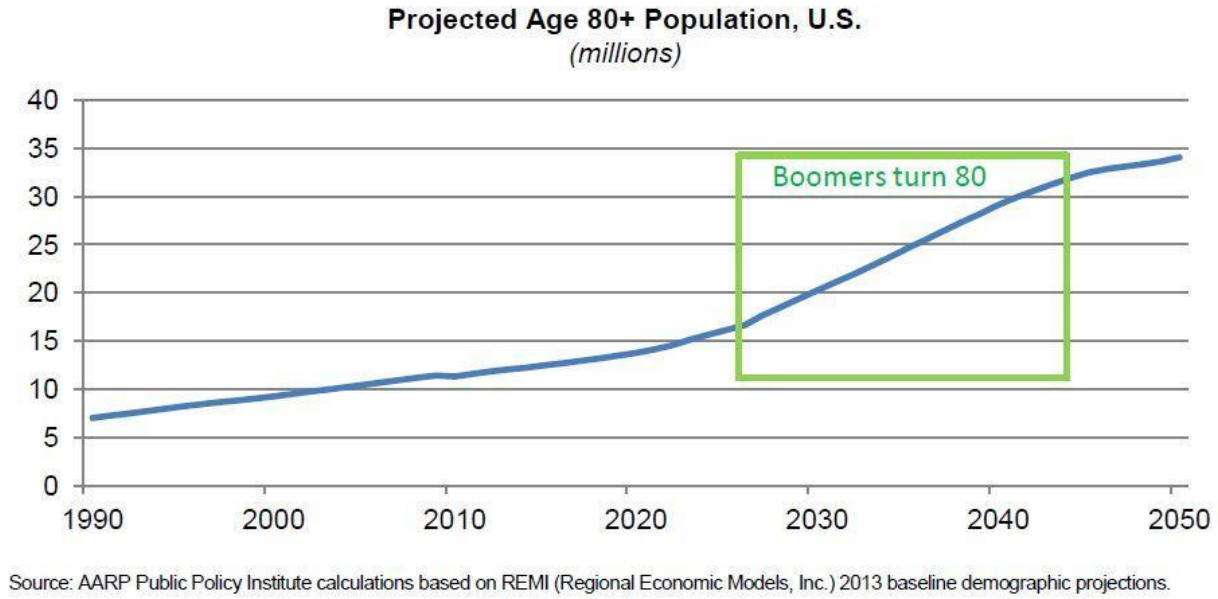

FIGURE 2: PROJECTED AGE 80+ POPULATION, U.S.

To make up for the difference between the available caregivers and the individuals needing attention, an increase in external care will be needed. This need can be filled several ways based upon the current abilities of the elderly. According to AARP, 30% of individuals aged 80 or older need help on a daily basis to perform daily task such as bathing or cooking [2]. These individuals need high levels of care, but others who are more mobile do not need constant, one-on-one, attention. Technologies such as robotic aids can be developed and utilized by some of the elderly to help with tasks such as shopping or carrying items. This would allow individuals to remain independent for longer time, decreasing the caregiving burden of the younger population.

1.2 Current Technology

There are several efforts being made in the field of robotic assistance; some of these efforts focus directly on increasing one's mobility while others increase the ability of persons with disabilities to perform various tasks. There are two main archetypes of assistive robots today: physical and social. Within this division there are further subdivisions; there are two types of physically assistive robots: Mobile and Stationary. The research into current technology is focused on mobile personal assistive robots.

The Quality of Life Center at Carnegie Mellon University (QoLC) is one of several organizations conducting research into assistive robots. One of their projects is for a Home Exploring Robot Butler or HERB [3]. HERB features a Segway base and two Barrett WAM arms. HERB can navigate in cluttered indoor locations and even microwave dinner.

Toyota has also begun working on assistive robots. They created the Human Support Robot (HSR) [4], which is designed to live with a family and provide support. The robot has an articulating arm and can be controlled by a tablet that also serves as the HSR's face. The HSR can also be used as a telepresence machine.

RTC, a Japanese company, has created the first robot to transfer humans from their beds to their wheelchairs and back. The robot is called RIBA and can lift a payload of up to 61 kg. While RIBA does all the lifting, a human operator is still present to guide the robot using tactile sensors on the robot's arms [5]. RIBA is an example of a human in the loop robot, where the robot can perform various tasks but is monitored by a user at all times.

1.3 Robotic Bases

There are currently many different robotic bases available for research and development. One of the first steps to designing something new is to see what currently available options are readily available. It is advantageous to use a ready-made base because the development cycle for hardware is shorter. With a pre-assembled base more advanced software can be written sooner, rather than spending time developing specifications and designs for each physical aspect of the drivetrain. The following robotic bases are all Robotic Operation System (ROS) enabled, which is one of the main features to be included in the project.

1.3.1 Mobile robot MP-500

MP-500, shown in Figure 3, is a robotic base which is very robust and sturdy, and is made to traverse many different environments. It is also made to handle dynamic environments similar to ones the project is defined to operate within. With 10 hours of uptime, the base also has enough time to perform many different tasks. But the robot has a very limited battery time if one considers

FIGURE 3: MP-500 ROBOTIC BASE

adding peripherals, so future modular additions might need to contain their own power sources. [6]

1.3.2 PeopleBot

FIGURE 4: PEOPLEBOT ROBOTIC BASE

The PeopleBot, shown in Figure 4, is designed for effective Human and Robot Interactivity (HRI) which is a key part of any robot that is aimed to be interacted with. It is also quite friendly looking with its molded plastic exterior, and it does not look overly industrial. The main downside to this robot is its lack of modularity; it is a relatively closed system, so future modules would be hard to develop [7].

1.3.3 Pioneer P3-DX

FIGURE 5: PIONEER P3-DX

The Pioneer P3-DX robot, shown in Figure 5, is about the size that would be sensible for the project. It can fit easily in the ADA defined doorways and hallways. This robot also has a 23 kg (50 lbs.) payload, which is a good amount to allow for future additions. It also allows for up to 3 batteries, which means that enough power could be provided for future modular additions. The downside to this robot is that it lacks additional ports; this limits the potential for extra sensors. The manufacturer also says that the robot works mostly on hard surfaces, so the soft surfaces that the project aims to traverse would not be possible. [8]

1.4 ADA Guidelines

In 1990, the Americans with Disabilities Act was passed by the United States Congress to provide a comprehensive set of guidelines for the construction of buildings. These guidelines were designed to allow individuals with physical disabilities to move about on their own. The establishment of these guidelines allows for a standard design system for buildings. This therefore ensures that all products that adhere to the standard will be able of operate in public and privately constructed ADA environments. Standardization also allows for use of select products, such as wheelchairs that suit the needs of a person with disabilities, without needing to worry if the product will fit through halls or have room to turn.

ADA restrictions govern aspects such as door and hall width, as well as the amount of room required to allow a wheelchair to turn. The ADA also defines the dimensions of a wheelchair: 32 inches wide and 48 inches long. All other restrictions regarding mobility assume these dimensions as a maximum. Additional restrictions described in ADA guidelines concern the turning radius of the wheelchair, or robot. A complete 180 degree turn must be completed in a 60 inch diameter circle.

ADA guidelines are also used when constructing access ramps for elevated doorways. The ADA states that that the maximum slope of a new ramp will be no greater than 1':12' with a maximum rise of 30 inches before a landing is required. This means that a ramp cannot rise more than 1 foot for every 12 feet of horizontal travel. Using trigonometry, the maximum slope of the ramp will be $\tan^{-1}\left(\frac{1}{12}\right)$, or 4.764 degrees. This regulation does not apply to ramps that have already been constructed, so not all "handicap accessible" ramps meet this restriction. Additionally, after a vertical rise of 30 inches, a 5'x 5' landing must be constructed in the ramp to give the operator an opportunity to rest while climbing. Railings are also provided along the length of the ramp for security. The railings extend 12 inches past the point where the ramp ends to provide additional guidance [9].

The ADA restrictions can be used to define the regulations for a personal assistive robot as well. ADA supplies guidelines that can be utilized to develop a conceptual working environment defining what types of obstacles a robot could encounter. Compliance with ADA restrictions assures that an assistive robot will be able to work with its user in many locations, especially environments designed for individuals with disabilities.

Individuals who require the assistance of a wheelchair for mobility can use specialized vans for transportation. Such vans are regular production model vehicles, such as a Honda Odyssey or Toyota Sienna, with modifications to their doors or trunk to include a lift mechanism. The two types of vehicles, side door and rear door, have similar dimension restrictions for the height of the wheelchair: approximately 57 inches from floor to ceiling. Similarly, the potential length for the wheelchair ranges from 58 to 91 inches long; there are short and long wheelbase models. The width can range from 31 to 60 inches wide. This range is due to the seating configuration; some allow for a passenger seat next to the wheelchair [10]. To enter the vehicle, a ramp is generally used, with an incline of 9 -11 degrees. While this is steeper than typical ADA regulation allows, the user is never alone when using the van; a driver is present to assist them.

1.5 Hardware

The selection process associated with the design and implementation of a robot is essential to its success. There are several factors associated with choosing a component: cost, performance, reliability, and power consumption. Each of these factors may have different weights associated with the design of the robot. For example, on a robot that operates in a remote location, reliability is more important than the monetary cost of a component. This section will explore the different components available for various functions performed by the robot.

1.5.1 Electronics

The topic of electronics covers a large number of devices and systems that connect the other components. This may include computing solutions, communications, power systems, and any other

electrical system. These systems are critical for the success of any mobile robotic system; without these systems a robot cannot process information, communicate, or act on its environment.

Computing systems come in many shapes and size. For this project only standard laptop and desktop processors (x86/x64) will be considered. Even within this relatively small subset of processors, there are still hundreds of options with power consumption from less than 1W, such as the Intel Atom Z500, to over 100W like the Intel Core i7-4960X. High level robotics tasks such as SLAM and image processing require a large amount of computational power. At the same time, mobile robots have a limited amount of power available. This means that a careful balance must be made between power usage and processing capability.

Laptops are well suited to mobile robotic systems since they include a battery, screen, and some communication system, but they are not without flaws. Laptops use proprietary power connectors, non-standard villages, and have fewer communication ports than desktop boards.

Computing solutions designed for desktop form factors provide a wide range of options for mobile robotic platforms. There are solutions available for any power budget, computational, or communication need. Desktop systems use standardized power connections and voltages, making them easy to support, upgrade, and replace. Desktop systems lack screens, wireless communications, and dedicated batteries. Though both options are viable it depends on the specific requirements of the system as to which are the better option.

Communication systems can play a major role in mobile robotics. In particular, wireless communications are essential to the field of mobile robotics. Being able to connect to a mobile robot, send commands, and receive sensor information, along with other data, can be vital to a successful robotic system. Wi-Fi, Bluetooth, xbee, and other radio frequency systems can be used for these purposes. ROS supports distributed processing, and data sharing over a network connection. This means that some of the processing can be done on a networked computer, rather than on the robot, and information can be uploaded to the ROS system. This allows more powerful computers, with higher power consumptions to be used, but not place a draw on the electrical system of the robot. To support this type of computing, robots using ROS benefit from a Wi-Fi connection. This can be done in a variety of ways, including having the robot broadcast its own Wi-Fi network. Regardless of how it is achieved, communicating with a robotic system can vastly improve the performance of the system. Communication is an extremely important part of a robotic system, but without power to run the communication systems they are useless.

Power distribution and regulation is critically important to the success of any mobile robot. Sensitive electronics, such as sensors and controllers, must be protected from voltage fluctuations that can be caused by high draw devices, such as motors. If the voltage drops too low the device can brown out, shut down unexpectedly, or give errant values. If the voltage goes too high, such devices can be permanently damaged. For less sensitive loads, it is important to make sure that power system and batteries are protected from excessive current draw. This can be done using current limiting resistors, breakers, or by ensuring that the supply will always exceed the need. For a modular robot, having plenty of available power connections for additional modules is a must. In addition to having available connections, the use of standardized quick connect allows for easy replacement of modules.

1.5.2 Batteries

Batteries are an extremely important part of any mobile robotic system; they provide the power for most mobile robotic systems. Although the power density of batteries is low compared to other systems, they do not produce emissions, noise, or any other dangerous byproducts. These properties have allowed batteries to become an integral part of everyday life. Battery performance can be measured by three criteria: energy density, specific energy, and specific power. Specific energy is the amount of energy per unit mass, while energy density is the amount of energy stored per unit volume. Specific power refers to the power to weight ratio of the battery. There are many different types of batteries are suited to different performance specifications.

Lead acid batteries have been in use for more than 150 years. They are extremely durable, reliable, and have excellent storage life. This type of battery is thermally insensitive, can withstand very high surge current, and does not degrade, even after thousands of charging cycles. Storage time is a concern in many applications and lead acid batteries can be stored for years without any significant impact on performance. Lead acid batteries have a charging efficiency of up to 92%, which is extremely high [11]. The largest flaw with lead acid batteries is weight; this limits the batteries specific power to 40W/kg. Lead acid batteries lend themselves to applications that require large amounts of power in any condition; many charge cycles and where weight is not a major concern, such as in the automotive industry and powered wheelchairs.

Nickel-metal hydride (NiMH) batteries offer many advantages over older lead acid technology. With a specific energy of up to 120 Wh/kg, a power density of up to 300 Wh/L, and a specific power of up to 1000W/kg NiMH batteries can deliver a powerful punch [12]. The downsides to this type include poor charging efficiency, around 66%, limited storage times (losing about 2% a month), and a short life expectancy of only 1000 charge cycles. These shortcomings have not stopped NiMH batteries from become extremely popular. They are used in many small electronics, and due to their relatively low cost they are used extensively in electric and hybrid vehicles.

Lithium-polymer (Li-poly) batteries have been commercially available for less than 20 years, but they have helped make the modern mobile electronic world possible. Li-poly's give unparalleled performance, with a specific energy of up to 200 Wh/kg, a power density of up to 300Wh/L, a specific power of up to 10 KW/kg, and a charging efficiency of up to 99.8% [13]. One concern when using Li-poly batteries is safety. If a Li-poly is charged incorrectly it can explode and catch fire. Since it contains lithium, water will only make this kind of fire worse. Despite these concerns, the performance of Li-polys has made them an integral part of modern electronics like laptops, cell phones, and robot.

1.5.3 Sensors

To properly understand its surroundings, a robot must be equipped with sensors. These sensors interpret the surrounding world into a format that the robot can process. Choosing a suite of sensors that will have the greatest effect on a specific platform is important and many different sensors need to be considered. This section will detail the different types of sensors that a robot might have.

1.5.3.1 Accelerometers

An accelerometer is a device that measures proper acceleration. They are used to detect and monitor vibration in rotating machinery. Inside an accelerator, micro electro-mechanical systems, MEMS

are present; these are basically tiny micro-structures that bend due to acceleration [19]. When it experiences any form of acceleration, these tiny structures bend an amount that is proportional to the acceleration on the device which can be electrically detected and used to calculate the acceleration. Accelerometers have multiple applications in industry and science because they can sense such a wide range of motion. Highly sensitive accelerometers are components of inertial navigation systems for aircrafts and Autonomous Underwater Vehicles (AUVs). They have been used to detect when the computer's suddenly moved or dropped, so the hard drive can stop before the drive is damaged. There are various applications for accelerometers in robotics such as enabling a mobile robot to balance. Accelerometers are also used in Inertial Measurement Units (IMUs) which are electronic devices that measure and report on a craft's velocity, orientation and gravitational forces, using a combination of accelerometers and gyroscopes [20]. The IMU detects the current rate of acceleration by using one or more accelerometers.

1.5.3.2 Gyroscopes

A gyroscope is a device for measuring or maintaining orientation, based on the principles of angular momentum. Mechanically, a gyroscope is a spinning wheel or disc in which the axle is free to assume any orientation [21]. Due to the disc's high rate of spin and moment of inertia, the change in this orientation due to external is small. The disc's orientation remains nearly fixed, regardless of the mounting platform's motion, because mounting the device in a gimbal minimizes external torque. Gyroscopes are now being used to keep complex robots, that would normally just fall over, upright: one such robotic example is two legged robots. Gyroscopes are also present in Inertial Measurement Units along with accelerometers as discussed above and the IMUs detect changes in rotational attributes like pitch, roll and yaw using one or more gyroscopes [20].

1.5.3.3 Ultrasonic Sensors

Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor [22]. The sensor measures the time of flight -- that is, the time it takes for the signal to reach a surface and reflect back [23]. This method of distance measurement is called echolocation and bats rely on the same principle to map their environment. Because ultrasonic sensors use sound rather than light for detection, they work in applications where photo-electronic sensors may not. However, this technology is limited by the shapes of surfaces and the density or consistency of the material. For example, foam, cloth, or even rain can distort sensor readings.

1.5.3.4 Cliff Detection Sensors

A Cliff Sensor is important to have on certain robotic systems to avoid excessive drops that otherwise might damage the robot. The iRobot Roomba models include a cliff sensor to help them avoid driving over stairwells or ledges [24]. A cliff sensor can be mechanical, optical, or even ultrasonic, however they all fulfill the same purpose. A mechanical cliff sensor is a contact that runs along the ground, looking for any drop-offs. When a drop off is encountered, the sensors generates a digital signal which indicates that the cliff has been detected. The Roomba uses an optical cliff sensor, which shines an LED onto the ground at an angle that is picked up by a sensor. When the Roomba comes across a large drop off, the reflected light from the LED is no longer detected by the receiver, and the drop is registered. Ultrasonic sensors can also be used to detect cliffs. Each type of cliff sensor has its own

strengths and weaknesses. These are primarily due to the effects of different surfaces, color and reflectivity effect optical sensors; however, texture matters most for ultrasonic sensors.

1.5.3.5 Rotary Encoders

A Rotary Encoder is used to convert the angular position or motion of a shaft or the axle to an analog or digital code and the types of information they offer can be packaged in various ways [25]. Rotary encoders are mainly two types - absolutes encoders and incremental encoders, sometimes also referred to as relative encoders. The output of absolute encoders indicates the current position of the shaft, making them angle transducers. The output of incremental encoders provides information about the motion of the shaft, which is typically further processed elsewhere into information such as speed, distance, and position. One of the more popular types of encoders is called an optical rotary quadrature encoder which is an incremental encoder. This type of encoder uses two LED emitter and receiver pairs that are about 90 degrees out of phase. Between both pairs of emitters and receivers, is placed a slotted disk that when rotated will either block or allow light to pass between the sensor/emitter pairs. As the slotted disk is rotated, the light sensors respond to fluctuations in light and cause them to output a "pulsed" voltage waveform that corresponds to the slot position on the rotating disk. Translating this into real-world measurements, by knowing how many slots are on the encoder disk (called resolution), the amount the shaft has rotated can be determined. Robots rely on encoders to determine speed, distance and joint position or other feedback that is essential for proper performance.

1.5.3.6 Microphone and Speakers

A Microphone converts sound waves into analog signals. As the sound waves reach the microphone, they move a small diaphragm back and forth within the device making the attached piezoelectric material to generate a voltage as the diaphragm moves back and forth [26]. This signal of the diaphragm motion can be captured and the recording can be analyzed by a robot for speech detection, or even detect beats to music. A Speaker works in reverse to a microphone. As a microphone detects sound waves as changes in pressure, a speaker generates pressure waves that human ears can interpret [27]. Speakers on robots allow them to generate different noises or voice instructions to indicate or provide some additional level of feedback to the user.

1.5.3.7 LIDAR

One method of sensing the environment is Light Detection and Ranging (LIDAR). LIDAR uses different forms of light to generate images of the environment. This is achieved by shining a light and measuring how long it takes for the reflection to get back. This allows the sensor to calculate distance to that point. The LIDAR does this in an arc meaning that the sensor provides a very detailed two dimensional map of distances to all objects in its field of view. By tilting the LIDAR sensor up and down a three dimensional field can be built from the data.

1.5.3.8 RGB-D

RGB-D sensors use a combination or two vision systems. One that detects visible light (RGB), and the other that detects infrared light emitted by the sensor that gives depth (D). The purpose of this combination is to allow the sensor to “see” in a similar way to a human. Stereoscopic vision systems can produce similar vision data, but converting the raw RGB data into the depth data requires a large amount of processing power. RGB-D sensors can use custom ASIC’s to process the depth information in

hardware. A leader in this technology is an Israeli company names PrimeSense. PrimeSense supplied the technology behind Microsoft's Kinect and ASUS's Xtion sensors. The Kinect intended for use with gaming systems to allow users to use their bodies as controllers. ASUS's sensor was designed for use with a PC. The Kinect uses proprietary libraries to do onboard skeletal tracking, while the Xtion uses library from OPNI NITE. OpenNI is an open source library that supports RGB-D sensors. After selling their technology to both Microsoft and ASUS PrimeSense has begun producing its own sensors.

1.5.3.9 Intel Galileo – Arduino

Intel Galileo as shown in Figure 6 is the first in a line of Arduino-certified development boards/microcontrollers based on Intel x86 architecture and is designed for the maker and education communities. The board runs an open source Linux operating system with the Arduino software libraries, enabling re-use of existing software, called sketches. This platform provides the ease of Intel architecture development through support for the Microsoft Windows, Mac OS, and Linux host operating systems. Galileo is designed to support shields that operate at either 3.3V or 5V. The core operating voltage of Galileo is 3.3V. However, a jumper on the board enables voltage translation to 5V at the I/O pins [46].

FIGURE 6: INTEL GALILEO FOR ARDUINO ENVIRONMENT

1.6 Robot Operating System (ROS)

Robot Operating System (ROS) is an open source meta-operating system that runs on top of Linux [29] ROS allows for hardware abstraction and multi-process message passing. ROS's modular nodes allow for the development of code for one robot to be used on other robotic systems. This kind of modularity allows for ROS packages to be to be added and removed as needed. For a modular robot this means that when a new module is added, very few software changes are required. Distributed computing can be a very useful tool in robotics, especially when some multiple computationally

expensive tasks must be done simultaneously. ROS has native support of distributed computing, allowing for seamless integration of multiple computers into a robotics system. ROS is a versatile tool for robotics. It also has multiple stable versions each of which has its own pros and cons.

1.6.1 Groovy Galapagos

ROS Groovy Galapagos was released on December 31, 2012 and brought with it major changes to the ROS core components [30]. Groovy introduced a new build method that is more adaptable for use with other operating systems and improves workflow. Stacks, collections of code to enhance sharing, were also removed to improve the modularity, and ease of installation of new packages. Met packages allow multiple packages to depend on one another the way stacks did, but without the added redundancies. One of Groovy's major advantages over other releases is its age. Almost a year after its initial release a vast majority of its bugs have been fixed, and there is a large pool of well tested packages for Groovy. Groovy may work well, but it is nearing the end of its supported life.

1.6.2 Hydro Medusa

Hydro Medusa is the latest release of ROS, having been released on September 4th, 2013 [31]. One of the main focuses of Hydro Medusa was converting core ROS packages to the catkin build system introduced in Groovy Galapagos. Improvements were also made to some of ROS tools, and many common packages were moved to a canonical release schedule. Overall Hydro Medusa made relatively minor changes to the ROS system while making improvements to ROS's existing features. Hydro Medusa is currently the version of ROS recommended by Open Source Robotics Foundation (OSRF), and will be supported until the end of 2014 if not longer. This continued support is an important consideration for an ongoing project such as this.

1.7 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) is a technique used in robotics to create a virtual space that the robot "sees" and plans movement through. This movement is then translated to the real world via robotic movement. After moving the robot will scan again and repeat the process. By performing this process rapidly the robot can interpret its environment in real time. SLAM allows the robot to adapt in situations with high amounts of uncertainty, such as a high traffic area. SLAM can be divided into two major topics: localization and mapping [48].

1.7.1 Localization

Localization involves using a mathematical process to determine the location of the robot in an uncertain environment. This is done in two parts; sensing, followed by applying the selected localization method to the sensor data. This can be difficult because moving adds uncertainty, so a very robust approach must be used.

1.7.1.1 Particle Filter

A particle filter is one of many ways to localize using signal data and information processing. To begin a particle filter one must first obtain a map of their environment. This map is then populated with a number of particles [32] which leads to better results. These particles represent possible places for a robot to be. They each have an x, y position and a theta, which represents the angle rotated from a predetermined zero orientation. With each of these particles one could evaluate what they "see"

against what the robot is currently seeing from the sensor data. Once all the particles have been evaluated the algorithm will assign each particle a probability depending on how well they match the current sensor data. Then the program will eliminate a certain number of the particles based upon the cutoff probability. The map will also be repopulated with a number of random particles. These random particles help the robot to re-localize if it moves in an unexpected way. For instance if a person bumps the robot so that it is no longer in the same location the random particles could be a good fit to what the robot now sees. Whereas if the random particles were not there, the robot would have to reevaluated and remove all of the previous particles before it could get a clear view of where it is. This method can be incredibly effective at localization the robot has the processor speeds necessary. Evaluating the position of thousands of particles at high speeds can become very computationally heavy. A Kalman filter can be just as effective as a particle filter but is less computationally intensive.

1.7.1.2 Kalman Filter

A Kalman Filter is an algorithm that can be used to estimate the state of a system. In robotics, this can be used as a part of SLAM to keep track of a robot's location. A Kalman filter has two steps: prediction and measurement. In the prediction step the algorithm calculates the estimate of the current position estimate of the robot including the level of uncertainty for each variable. In the measurement step the algorithm takes into account the measurements from the robot's sensors and the amount of likely error for these measurements and updates the state of the robot giving more weight to more certain variables [33]. One of the main conveniences of using a Kalman filter algorithm is that since it is a recursive algorithm it can be run in real time. An important aspect to remember when considering a Kalman filter is that it performs best in linear systems. An example of a linear system is a spring pushing a cart in a line. While a nonlinear system would be the Theory of General Relativity, so the Kalman filter is best in simple situations, but the more dynamic the system is the harder the Kalman filter becomes to implement.

1.7.2 Mapping

Mapping is the act of taking sensor data and creating a map from it. If a robot receives data from a sensor about the distances to nearby obstacles and data from odometry sensors, the robot can determine the obstacle's position in a generated map relative to its own position. This is done by drawing objects seen by the sensor onto a virtual map. When the robot turns, it keeps track of its position using odometry so new obstacles can be added to the current map. And it knows where the new data goes in relation to the old creating a new map with many more landmarks. This can be a very effective method, but due to the generally uncertain nature of sensors and the moving world around the robot, making a map can become a very difficult task.

1.7.3 QR Code Tracking

QR code or Quick Response Code is a type of two-dimensional matrix barcode as shown in Figure 7 which consists of a series of black modules arranged in a square grid on a white background which can be read by an imaging device like a barcode laser scanner or a camera. It is simply a machine-readable optical label that contains information regarding the item to which it is attached and data is encoded in the patterns present in both horizontal and vertical components of the image. It was first designed for its uses in the automotive industry to track vehicles during manufacture as it was designed

to allow high-speed component scanning. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte / binary, and kanji) to efficiently store data [45].

FIGURE 7: IMAGE OF A QR CODE

1.8 Path Planning

Path Planning is the process an autonomous or semi-autonomous robot uses to find a route from its current position to the target location. In order to plan a path, the robot must first interpret its environment into a series of nodes that can or cannot be traversed. Then, the traversable nodes must be examined using a search algorithm to find the safest, most efficient route.

1.8.1 Occupancy Grid

Creating an Occupancy Grid overlay on a map is an important step in path planning for that map. An occupancy grid takes a map and splits it into grid cells [34]. An occupancy grid can be comprised of squares, hexagons, triangles or other shapes that allow for tessellation. This means that the shape can be repeated to cover an area completely without overlap. The cells also do not have to be the same size. Sometimes it is advantageous to have different sized cells. The occupancy grid is used to determine if a robot can exist in the different cells. When an object is detected in a cell, that cell is defined as impassible. When planning a path, each potential location (cell) is defined as a node. A path consists of a list of adjacent nodes that represent the locations a robot will be on the proposed route from start to end location.

One method that results in different sized cells is to only split up cells until they have only open space or obstacles in them. This results in a few very large cells that are all open space or obstacle and many smaller cells that had to be split up to separate the obstacles from the free space. This is helpful because it results in the smallest number of grid cells for a given map, which helps with computation time, and because a robot is less likely to drive into an obstacle since it will never go to a cell that has any obstacles in them.

1.8.2 Search Methods

Many methods for searching a data structure exist, each with its own advantages and disadvantages. A search algorithm that is both time efficient and capable of always returning the optimal path are generally key to a robot's success. Some search methods use a guided search, meaning that each node is expanded using an equation to determine if it is the most viable method, non-guided search methods expand nodes blindly.

1.8.2.1 Breadth First Search

A breadth first search (BFS) algorithm is one of the two major graph searching algorithms. The process of conducting a BFS is simple [35]. In a graph of nodes (in SLAM these can be occupancy grid cells), given a starting node, the algorithm will examine all the starting node's neighbors and only then move on to searching the next node's neighbors. The search pattern of a Breadth First Search can be seen in Figure 8.

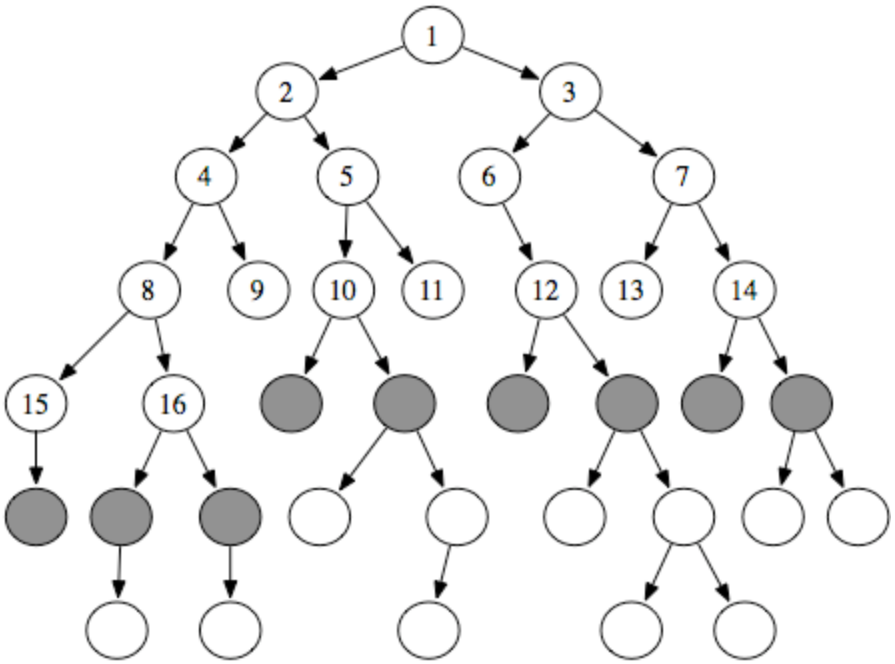

FIGURE 8: BREADTH FIRST SEARCH EXAMINATION ORDER

1.8.2.2 Depth First Search

A depth first search (DFS) algorithm also can be used to explore a graph of nodes. When given a starting node, a DFS will pick one neighbor and search it before choosing one of that neighbors' neighbors and continuing down the line [36]. The DFS algorithm will only return to the starting node when all the children of the first neighbor are explored. After doing that it will choose the next neighbor and explore down its line. The search pattern of a Depth First Search can be seen in Figure 9.

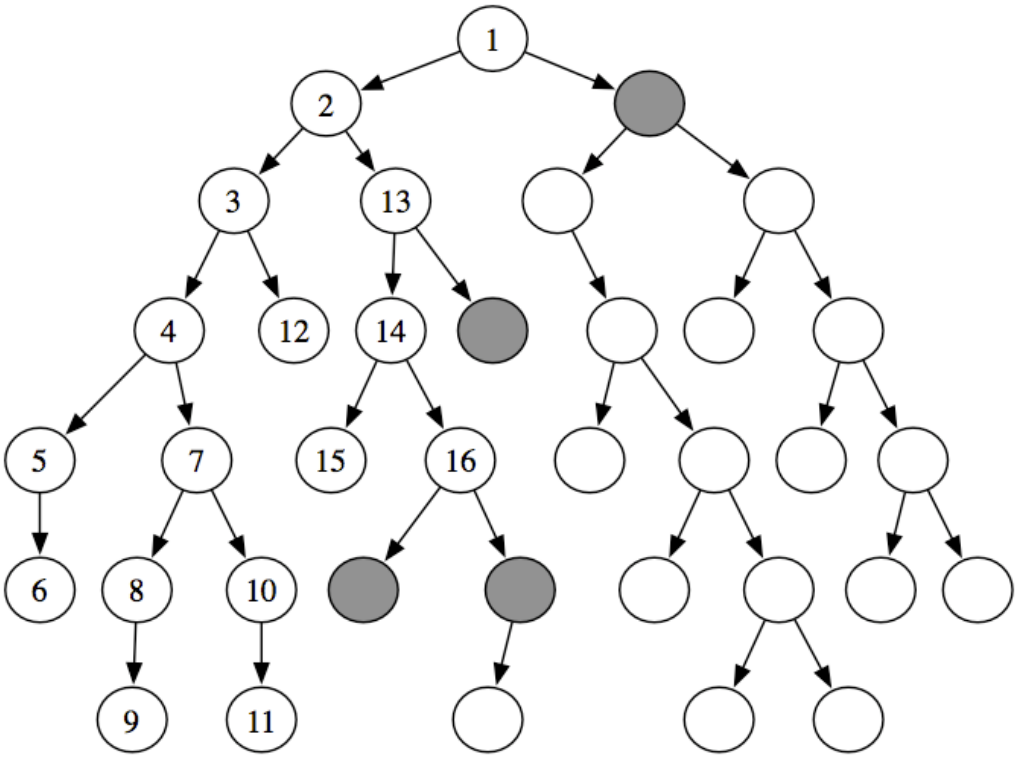

FIGURE 9: DEPTH BREADTH FIRST SEARCH EXAMINATION ORDER

1.8.2.3 Dijkstra's Algorithm

Dijkstra's search algorithm finds the shortest path to the target. This is done by exploring each node on the map, in order of lowest cost to highest cost, until it finds a path to the target. When each node is explored the neighbors or that node have their costs evaluated and updated if this new path to that node is less expensive than a previous path. To do all of this, the algorithm must keep track of all the nodes and the neighbor relationships that certain nodes share. To sort through all the nodes, they are all given an infinite cost and added to a queue. The starting position is then given a cost of zero and its neighbors are explored. The exploration of all the nodes continues from there. Unlike the A* algorithm, Dijkstra does not keep track of the nodes it has explored as the fact that all the nodes are added to the list to explore only once ensures that Dijkstra's algorithm has a finite run time.

1.8.2.4 A*

A* search algorithms find the shortest path to the destination by exploring paths with the lowest expected cost [37]. For each path segment, an estimate of the cost to reach the destination is made. The estimate for each location is derived using a heuristic function; for navigation, this could be the straight line distance between the potential next location and the destination. This heuristic function must be admissible, meaning it will never overestimate the cost of the path. The algorithm must keep track of all of the explored nodes, as well as the estimated cost to go from each explored node to the destination. When implemented, the algorithm expands the lowest cost node. The lowest cost node is the node whose heuristic value added to cost of getting to that node is the smallest. A* finds the least cost path without expanding every possibility and as such is usually faster than either a BFS or a DFS. The search pattern of an A* algorithm can be seen in Figure 10.

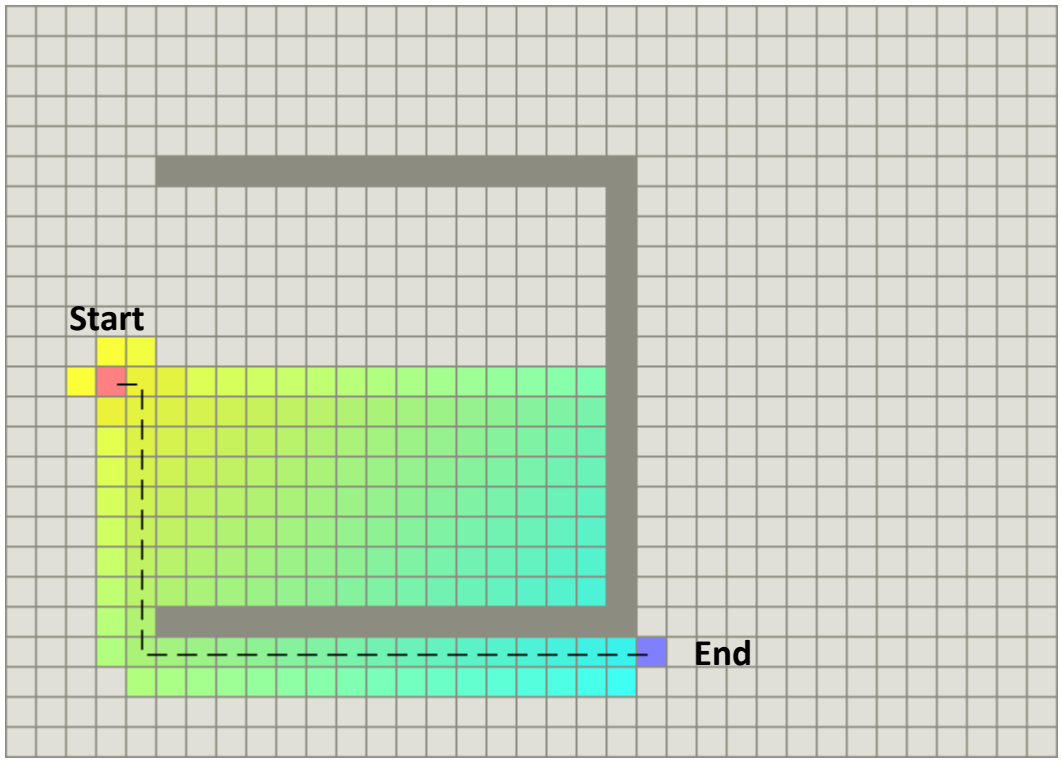

FIGURE 10: EXAMPLE OF A PLANNED PATH

1.9 Cornell Cup Competition as Presented by Intel

The Cornell Cup Competition is an annual competition that is organized by Cornell University and is currently sponsored by Intel. The purpose of the competition is to showcase the capabilities of an Intel Processor. In the case of the 2013-2014 competition, the competition provides two Intel Atom development boards and 2 Arduino Galileo boards for use in the designed system. The completion, which is intended for College students, has a strong focus on the Systems Engineering approach to the design and implementation of a system. Further, Cornell encourages participants to develop a business plan for their project to have an additional focus on developing entrepreneurial skills [47].

2 Design Requirements and Specifications

The development and construction of PARbot requires taking into consideration all the specifications and requirements laid out before and an initial design can take from. Having a clear list of design requirements and restrictions, as well as a defined working environment is critical to developing a product that meets the project guidelines. It is also important to know the reason for designing a system before developing the system requirements. The PARbot system is being designed to fulfill a developing need in the assistive care industry as a part of grant from the National Science Foundation (NSF).

2.1 Stakeholders

In order to develop the specifications of PARbot, the parties involved with the system must first be recognized. In the case of PARbot, the main stakeholders for the project are individuals with mobility or strength disabilities (users), the project team, and the project sponsors- in this case WPI, the National Science Foundation, and the Cornell Cup USA as Presented by Intel. Other stakeholders include but are not limited to the bystanders who share the environment with the potential user.

2.2 Development of Specifications

The primary goal of the PARbot project is to provide assistive care to the elderly or other individuals living with disabilities. To fulfill this goal, PARbot must be able to follow its user. ADA restrictions, described in the background section, provided a well-defined, nationwide standard for accessible building construction. These guidelines provide a starting point for physical restrictions on the robot, such as length, width and turning radius, as they are determined by the space allocated to a wheelchair under ADA regulations. Doorframes restrict the width of the robot to 32 inches, not accounting for navigational error. A typical wheelchair is no more than 48 inches long and a 60 inch turning circle is provided where 180 degree turns are required. [38]

2.2.1 Boston Abilities Expo

Boston, Massachusetts hosted an "Abilities Expo" where necessary products and services for the community of people with disabilities, their families, caregivers, seniors and healthcare professionals are brought together under one roof. Opportunities are introduced that can enrich the lives of these individuals through equipment providing mechanical assistance of various sorts. The expo aims to increase awareness of products, many of which people weren't aware of were now commercially available. "Abilities Expo" is held in several cities across the nation and each Abilities Expo offers three days of access to the latest technologies and resources for various disabilities. Informative workshops are offered on issues that resonate with the community and fun activities, such as adaptive sports, dance, assistive animal demos and a number of similar ability-enhancing activities are presented.

Attending the Abilities Expo was of great help to the PARbot team as the Expo offered new ideas and effective solutions to our problems. For example, one of the workshops offered information on "The Accessible Home" which shows the design process of a house for someone with a disability based on building connections, living and dining areas, design functional driveways and parking areas, and other functionalities in deign which would make a home more accessible. This information was of great value while performing software development and deciding how the PARbot would move around the house.

The Expo also offered a great deal of information regarding what kind of Interface for Human Robot Interaction would be ideal for the targeted users. Allowing simple, yet effective communication between the robot and its user is essential to the successful operation of the robot.

2.2.2 Personas

A clear definition of the consumer of a product is essential to the product's success; designing a product without taking into account the needs of the user results in an unmarketable product. In order to develop a concrete idea of who the users of PARbot would be, each team member developed a "persona" – a short biography of who the each team member envisioned as potential user of the assistant robot. The individual personas are included in Appendix A. The biographies resulted in a wide range of possible markets, but clear sectors emerged and one was to be selected for the first phase of the project. The design of PARbot would focus mainly on assisting the elderly or those living with physical disabilities, such as arthritis, and is confined to a wheelchair. While the scope of the project could be expanded to amputees, a narrow consumer demographic was determined to be better to design for.

The development of a series of personas also highlighted the possible uses cases that the robot would need to fulfill. Analysis of these use cases allows for further examination of how the developed system will interact with its environment and the impact that it will have on those around it.

2.3 Resulting Requirements

Resulting from the research and specifications as described above, the team developed requirements for PARbot. The team decided PARbot would need to be able to operate in an ADA environment which would set our maximum sizes. Additionally, the robot should be capable of folding to fit in the trunk of the average motor vehicle to allow the users to take their assistive device with them in public. The ability for the user to take the robot with them also means that its weight should be minimized; a maximum weight of 50 pounds was assigned to the PARbot. The robot would also need to be able to help its users, so the team decided that a carrying capacity was required and we decided the minimum necessary was about equal to the weight of several bags of groceries, approximately 10lbs. The team required that PARbot have a battery life of 4 hours, as well as the ability to map and navigate within its environment to ensure usability. Since the robot can be taken with the user into public settings, a number of different terrains will be encountered, ranging from hard wood floors and linoleum to carpet to asphalt and concrete.

PARbot is required to be a modular system, allowing for easy expansion in the future. This means that in PARbot's design, it is to be easy to modify elements of the software, simple to add or remove components, and easy to access the power and data connections for future changes. The robot must be able to follow its user and not get lost, potentially causing it to begin following another person. Additionally, the robot's design must not be intimidating; users and the public must feel comfortable around it. Since PARbot's target demographic is often more likely to feel threatened by larger, more industrial style robots, care must be taken when designing the robot casing to ensure it is aesthetically pleasing and non-threatening.

3 Design and Analysis

A robotic system consists or three main parts: the mechanical systems used to actuate various parts of the robot, the electrical systems which carry sensor data to the robot's processer and commands to motor controllers, as well as provides energy to the robot in the form of an electrical potential, and the software which provides an overarching control system to give commands to all other systems. Each of these systems have a direct impact on the others and must be designed to allow for easy interfacing; the electrical system must be able to deliver enough current so all other systems can operate without browning out. By examining each system of the robot, a clear picture of it internal working and design becomes clear.

3.1 Drivetrain

When examining options for the drive system of PARbot, two options were considered: holonomic drive and differential drive. In a holonomic drive system, Mecanum (omni) wheels are used. This type of wheel contains rollers evenly spaced around the center hub of the wheel with their axis of rotation offset from the rotation of the hub itself. This configuration allows the vehicle to move in any direction, regardless of the orientation of the wheels, and allows the robot to have a zero turning center. This means that the robot is capable turning about its central axis, rather than a point outside the wheelbase. However, to utilize this increased degree of mobility, all wheels must be driven, requiring additional motors, motor controllers, and electrical power. Further, the drive code written for the robot is more complex to take into account the increased mobility.

A differential drive design consists of having only two wheels powered, one wheel on each side. Additional wheels are non-powered castors or idler wheels. This configuration can be used with any wheel type, so allows for a larger number of choices in wheel sizes. A differential drive robot is limited in degrees of mobility based on the combination of idle wheels, powered wheels, and castors. Typically, a differential drive robot does not have a zero turning center and rotates about a point outside of the wheelbase. This means that turning a differential drive robot's turning radius is impacted by its wheelbase and requires more space to complete a maneuver.

The design of PARbot took into account the pros and cons of each type of drivetrain; the final design uses a differential drive system, with the powered wheels at the rear of the robot and a pair of castors at the front. Differential drive was selected because traditional pneumatic tires can be used. The team felt that due to the wide variety of surfaces that the robot could encounter, the durability of wheels is important. Mecanum wheels have many rollers in them, each of which is a potential failure point. Rough surfaces such as pavement could damage the rollers and carpet fibers could catch at the roller mounting points. Pneumatic tires can easily travel over these surfaces with fewer points of failure; they can also adsorb some of the vibration caused by traveling over rough surfaces. Finally, a differential drive system uses fewer motors than a holonomic drive system, meaning the overall weight of the robot can be kept lower.

3.2 Chassis Design

The chassis of a robot system not only holds all of the system components, but determines the size, shape, weight and other critical features of the system. As the requirements for the robot changed,

the chassis changed to accommodate them. Early chassis designs were large, allowing for maximum interior space and providing exposed channel for mounting external modules. To make the design more visually pleasing, the outer shell of the robot was made up of complex curves. This design was lacking in good mounting locations for certain modules. These limitations led to a new octagonal design which allows modules to be both internal and external at the same time. Giving every module internal space allows it to get power, data, and structural mounting in a standard package. Although the design has many advantages, it is very costly to produce to the high level of custom manufacturing required. The estimated weight of this robot is over 100lbs, making it nearly impossible to fit in a standard car. To make the robot better suited to the home environment it would have to be smaller, lighter, and more maneuverable. To further its suitability, a weight limit of 50 pounds was imposed.

A smaller chassis design allows the robot to maneuver easily in the home and seem less threatening to the end user. The design has significantly less interior space than other designs, therefore internal module space is not included in the design; additional components can be attached to the upper surface of the robot where additional power and data connections are available. Modules can be attached using a through-hole plate. The design includes the ability to collapse the super structure allowing the robot to fit in most vehicles. The design also reduces the need for custom manufacturing; only motor mounts and wheel hubs will need custom made. This allows the robot to be more easily assembled using off the shelf, commercial parts. The ride height of the robot is a critical factor in designing the chassis. The requirement of four inches of ground clearance force the motor mounts to join the front and back of the chassis. The final design fits all of the requirements set for it and provides space for all of the robots components.

The designs wide folding section is made smaller to make the robot appear less threatening and reduce the chance that it will hit obstacles. This change drastically changes the profile of the robot and the amount of space for the robot's vision sensors. The reduced space poses a risk to the vision sensors since they could be damaged by impact with the chassis. The design change requires additional parts to be ordered, but parts are available to temporarily secure the folding section. These temporary supports are less stable than hoped, yet the system is stable enough for testing to proceed.

3.3 Shopping Cart Module

The shopping cart module is designed to allow PARbot to tow a shopping cart. The module is intended to adjust to fit most shopping carts in use today. It is designed to be adjusted to the correct height by the user, placing the sloped side of the hooks such that the robot will lift the cart when it backs up. The robot can then be told backup using the touch interface. To release the cart when the user is done shopping two handles can be loosened to release the rotary breaks.

3.4 Onboard Electronics

The onboard electrical system of PARbot runs off a single 24 volt battery. Onboard DC-DC converters are employed to provide the lower voltages necessary to run the onboard computer, network router, and various sensors for navigation in interaction with the environment. USB cables relay digital sensor data to the onboard computer for processing. The Simultaneous Localization and Mapping (SLAM) algorithm utilizes a point cloud of data to recognize obstacles and generate a map of showing the safe locations for the robot in its current environment. Various types of cliff detection

sensors are used to offset the flaws inherent to a single sensor type, providing more accurate data to prevent accidents.

3.4.1 Battery Selection

Battery selection is a very important part of building any mobile robot. The battery has to have a good ratio of power capacity to weight. A mobile robot needs a large amount of power available in order to operate for extended periods of time while staying light enough to move itself and power any extra systems on the robot. Weight is critical when a robot can brought with someone by putting it in their car. Normal high capacity car batteries are typically lead-acid; these batteries are very heavy. Lithium Polymer batteries are a new, lighter technology which can be high capacity and are substantially lighter than their Lead-acid counterparts. The lighter weight is attributed to the use lithium and polymer to hold the electrical charge, rather than of lead.

The PARbot team decided to use a Lithium-Polymer battery because of the lower weight to power ratio that the batteries have. The battery selected was purchased from the AA Portable Power Corp and has a rated voltage of 25.6 volts and a 20Amp Hour capacity. The total weight of the battery is 9lbs and 11.8 Oz; a comparable lead acid battery would weigh around 25 Lbs. [40]

3.4.2 Motor Selection

When buying electric motors, there are two choices on types of motors, brushed or brushless motors. A brushed motor has two or more carbon "brushes" on the inside of it; these complete the circuit and cause the motor to spin. A brushless motor instead uses a rotating magnetic field to generate the spin of the motor. A table of advantages and disadvantages is shown in Table 1.

TABLE 1: BRUSHED VS BRUSHLESS

Brushed		**Brushless**	
Pros	***Cons***	***Pros***	***Cons***
Easy two wire control	Maintenance required due to brushes	Less maintenance due to absence of brushes	Higher cost
Brushes can be replaced	Poor heat dissipation	High Efficiency	Control is complex due to the use of Hall sensors
Low cost	Lower speed range due to brush friction	High power out to size ratio	Controller is expensive
	High noise	Low noise	

The original plan for the robot was to find a brushless motor that would have all the required torque. After some research it was determined that an off the shelf brushless motor was not an option. Brushless motors with numbers that meet the project specifications are only available through custom manufacturing. Therefore it was decided to move to a brushed motor, the ease of acquiring and ease of control were the main factors that pushed towards the change. The change to a brushed motor means that the robot will periodically need brushes changed, but the lower motor cost would offset the future costs.

3.4.3 Voltage Converters

To support any future modular additions to the robot, several different voltages will be available to use. The decided upon voltages are 3.3, 5, and 12 volts, these are common voltages among sensors and motors. These voltages will be provided by the main 24V battery through different transformers. The 3.3 volt transformer will output 3.3 volts at 5 Amps. The 5 volt transformer will support 3A out, and the 12 volts will be supplied at a maximum of 20A. The full spec sheets for these transformers will be available in the appendix.

3.5 Onboard Computer

The onboard computer is responsible for handling all of the processing, control, and interaction functions of the robot. It must be able to interpolate all of the sensor data coming in, perform SLAM, path planning, user tracking, keep track of user commands, and any other functions that future modules may add. Additional processing power is also budgeted for unforeseen additional components or processing needs in the future. The first option that was considered was an Intel Atom D2550 based system provided by the Cornell Cup.

The Cornell Cup rules require that a specific computer system be used. This system has an Intel Atom D2550 and an Atmel FPGA onboard. Due to the sensors being used, the FPGA cannot be used as a sensor interface without the addition of a complex daughter board. Rather, an Arduino board (described previously) is used for analog sensor interfacing and USB connections are used for digital communications. This sensor connection and programming plan requires that the processing be done by the Atom. Tests of the Atom's processing power show that it delivers about 350 Megaflops of performance. Other tests show that streaming data from the PrimeSenses and processing the data is not possible with the limited processing power of the Atom board. Additionally, when running the six processes needed to use the sensors and convert the data to lasers scans the Atom was limited to about 75% total CPU usage. This stems from the single threaded implantation of the depthimage processing libraries. Although an openMP implementation was investigated the underlying memory management was not suited to openMP's shared memory model resulting in unresolved stability issues. More over the laser scan data produced by the Atom was delayed by approximately half a second under ideal condition, reduced depth image size and a high percentage of frame drops. This meant that even if the all of the other code needed for the robots operation could run on a single thread core, since the Atom contains two Hyper-Threaded physical cores, its decisions would be out of sync with the real world. Such a delay could cause the robot to cover two body lengths before even getting data informing it of an obstacle. A likely outcome of this combined with the PrimeSenses dead zone, an area about 30-40 centimeters in front of each sensors that cannot be reliably observed, is that the robot would run into any static first observed from less than a meter away. Dynamic obstacles such as people would move at high enough speeds relative to the robot to that there could be no certainty of avoidance.

The designed onboard computer uses an Intel core-i7-4770K Haswell processor with Intel's Z87 chipset. An ASUS Impact VI mini-ITX motherboard is used to provide improved voltage control and power efficiency. Voltage control is necessary to ensure that fulminations in the supply voltage due to components due to other components turning on or off do not brown out or damage the processor. Eight GB of 2400MHz DDR3 ram is used by the system which provides enough space and speed for the

system. A Zalman Reserator 3 Max CPU cooler was considered to keep the robot watertight, but when design changes allowed for sufficient airflow the stock cooling system was left in place. Tests show that the i7-4770K reached approximately 200 gigaflops when running Intel's Linpack test or about 570 times faster than the Atom processor. Through testing, this system has shown that it is capable of processing the necessary vision sensor data as well without consuming significant processing resources.

3.5.1 Communication

On the PARbot system, there are several communication networks in used. A Local Area Network (LAN) is hosted by an onboard router for communication between the main system computer and a Human Robot Interface, and can provide additional connectivity as required. Sensors and motor controllers will communicate with the main processor over USB, allowing for simple, standard connections. Analog sensors will communicate over USB via an Arduino which serves as an Analog to Digital Converter. Figure 11 shows a diagram of the data connections included in the robot.

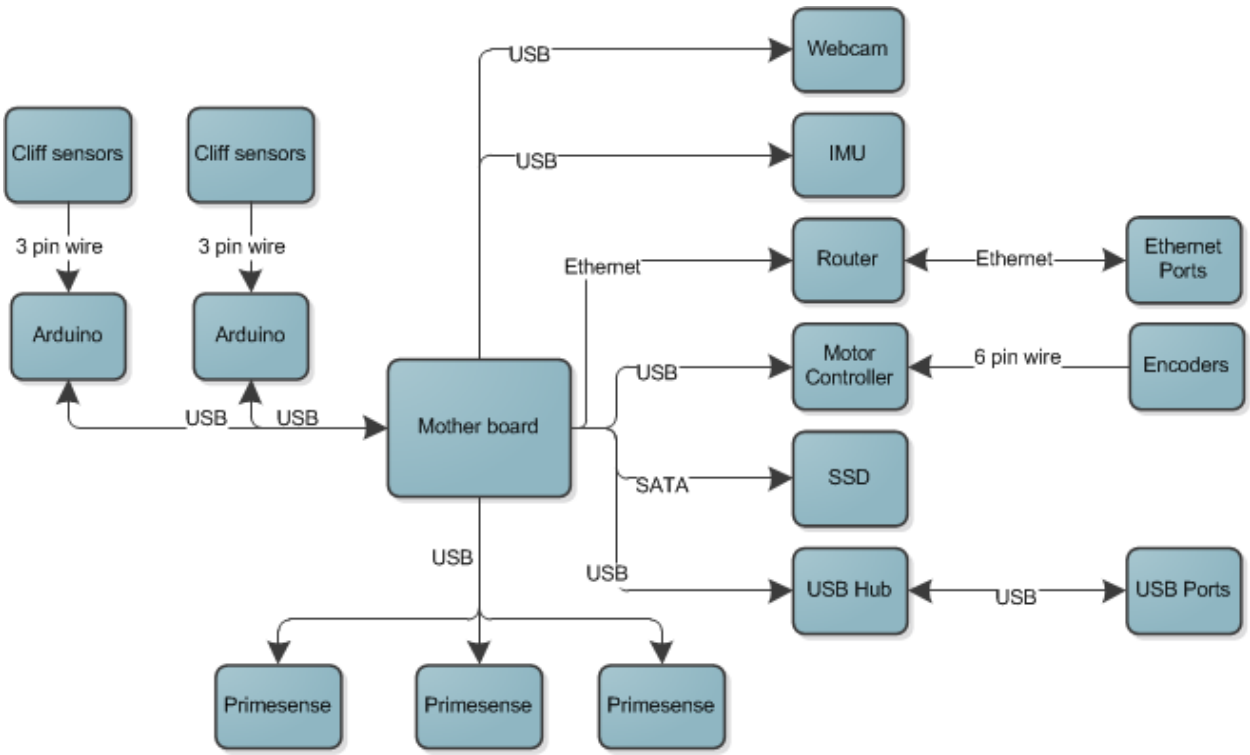

FIGURE 11: DATA CONNECTIONS MAP

As shown in Figure 11, USB connections make up the backbone of PARbot's data communications system. USB connections allow for simple addition and removal of components with standardized, reverse compatible jacks and cables. Further, USB connections are digital, which provides a small level of resistance to signal interference and can provide a limited amount of power to the device that is connected, allowing some modules to be added though the connection of only a USB cable. To ensure that there is not a data bottleneck, several USB buses are used to allow critical data to flow more quickly.

In addition to the USB connections which are prevalent on the robot, several device specific connections are present on the robot. A six pin wire is connected to the motor controller to provide it with data from the two wheel encoders that are equipped to the robot. Three pin wires are used to connect the analog interfaces of the cliff detection sensors (Power, Signal, and Ground) to an Analog to Digital Converter on the Arduino Galileo boards. Ultimately, this information is communicated to the computer via PARbot's USB connections. Finally, Ethernet connection are made available for connections to other computers and modules.

3.5.2 Networking

In the PARbot project, the inclusion on an onboard Ethernet network was needed to allow for onboard communication for current features as well as future modules. Further, a wireless network needed to be broadcast to allow for communication with the robot during testing without a tether. This wireless network could later be used for incorporating external modules as well. Router selection for the project was based on available gigabit Ethernet ports and availability of high speed, multi-bandwidth Wi-Fi support. The router selected was a Western Digital My Net N900. The router allows for the use of multiple computers and the connection of the onboard user interface. It also allows for data to be streamed from the robot to external computers for debugging and monitoring.

3.5.3 Analog to Digital Conversion

While some sensors communicate over the robots' network via Ethernet or USB, other sensors output an analog voltage containing the data. In order to interpret and use the information from these sensors, a system to transform these analog signals to a digital format is needed. To achieve this Analog to Digital Conversion (ADC), an Arduino is in use on the robot. The Arduino is a small microprocessor with many analog input ports and an easy to use programming library. Each analog sensor will be connected to one of the Arduino's analog inputs and the digital values are output over the USB connection to the main processor. There is also a ROS stack made for interfacing with the Arduino over USB, removing any potential communication issues. With these features it is simple to add additional sensors to the robot without the need for additional conversion hardware.

3.5.4 Sensors: Navigation

In order to ensure that PARbot's navigation ability operates at the necessary level of accuracy to ensure safe and reliable operation, several sensor configurations were considered. These sensors are necessary to ensure the robot's navigation is both robust and safe. The operating environment for PARbot means that the robot will be around numerous humans in addition to the user and the ability to avoid them as well as static obstacles while performing SLAM is addressed through these sensors.

3.5.5 Inertial Measurement Unit (IMU)

The 9DOF Razor Inertial Measurement Unit (IMU) operates at 3.3VDC and incorporates three sensors - an ITG-3200 (MEMS triple-axis gyro), ADXL345 (triple-axis accelerometer), and HMC5883L (triple-axis magnetometer) that provides nine degrees of inertial measurement. The outputs of all sensors are processed by an on-board ATmega328 chip and output over a serial interface. This enables the 9DOF Razor to be used as a very powerful control mechanism for autonomous vehicles, UAVs and image stabilization systems. The board comes programmed with the 8MHz Arduino boot loader (stk500v1) and firmware that demos the outputs of all the sensors. The serial TX and RX pins are

connected with a 3.3V FTDI Basic Breakout which transfers data serially at 57600bps. The Arduino IDE used to program your firmware code onto the 9DOF using the Arduino Pro or Pro Mini (3.3v, 8 MHz) w/ATmega328' as the board. ROS C++ and Python packages are available online that publish the yaw, pitch and roll data from the sensor to the terminal through a USB connection and perform simulation presenting a graphical representation of the orientation of the IMU. [39]

The IMU unit is used on PARbot to supplement the odometry data received from the wheel encoders. The IMW will also provide data on any slope the robot is climbing, which is necessary for computing the transformations used to generate a map of the operating environment.

3.5.6 Cliff Detection

One of the problems that a robot faces in an environment made for humans are cliffs. These could be actual cliffs or much simpler cases such as a curb or stair. If the robot is not given appropriate sensors it will not be able to interpret its environment. To make sure that the robot will not miss a set of stairs a series of sensors are added, these are called cliff detection sensors. To have very accurate data two different types of sensors are to be used. The first sensor is an Infrared Sensor (IR), this sensor uses infrared light to determine distance. Because this sensor uses light it is susceptible to problems due to reflectivity of surfaces. To counteract this problem the IR sensor will be used in conjunction with an ultrasonic range finder. This sensor uses ultra-sonic sound to determine distance. By using both of these together the robot will have a much more accurate indication of its surroundings.

The cliff detection system on PARbot uses eight sensors arranged in pairs, one infrared paired with one ultrasonic sensor. By arranging the sensors in pairs, the likelihood of a sensor not recognizing a risk is mitigated through the inclusion of a sensor with different features. A distance of 1.5 inches separates the leading and trailing pairs of sensors to prevent gaps, such as those found at the entrance to elevator doors, from registering as a cliff.

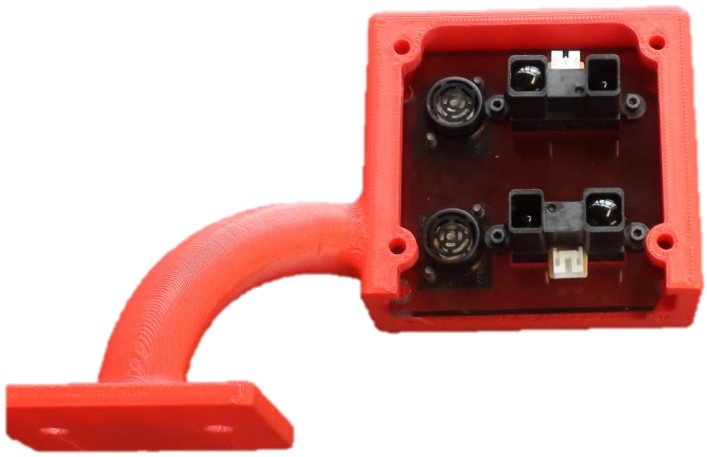

FIGURE 12: CLIFF SENSOR MOUNT

Figure 12 shows an individual cliff sensor mount with the pairs of cliff sensors mounted. One of these assemblies is mounted in front of each of the main drive wheels. In Figure 13 provides a theoretical overview of the complete sensor arrangement at the front of the robot with the distances between sensors labeled.

<table>
<tr><td>Infrared Sensor Set#3</td><td>Ultrasonic Sensor Set#3</td><td rowspan="4"></td><td>Ultrasonic Sensor Set#1</td><td>Infrared Sensor Set#1</td></tr>
<tr><td colspan="2">(Distance of 1.5 inches between the two sets)</td><td colspan="2">(Distance of 1.5 inches between the two sets)</td></tr>
<tr><td>Infrared Sensor Set#4</td><td>Ultrasonic Sensor Set#4</td><td>Ultrasonic Sensor Set#2</td><td>Infrared Sensor Set#2</td></tr>
<tr><td colspan="2">(Distance of 2 inches from the front plate of the robot base or 7 inches from the center of the wheel)</td><td colspan="2">(Distance of 2 inches from the front plate of the robot plate or 7 inches from the center of the wheel)</td></tr>
<tr><td colspan="5">Robot Base</td></tr>
</table>

FIGURE 13: CLIFF SENSOR ARRANGEMENT

Before the sensors can be used they must be properly calibrated. The Arduino program maps the value from the analog voltage to an appropriate value for the sensor connected; this value is then returned in inches. IR mapping is accomplished through the testing of various physical conditions to collect sensor values for different readings. A best fit equation is then applied to the data. The resulting equation is:

$$Distance\ (in) = 6202.3 * Sensor\ Value^{-1.056}$$

This equation has an R^2 value of .9914 which means that the equation represents the actual data with a 99.4% accuracy rate. The data used to compute the equation can be seen in Table 2. The actual and best fit data are plotted together for comparison in Figure 14. The equation is implemented in the Arduino code. The data is transmitted via a USB connection to the computer where the ROS node uses this data.

TABLE 2: RECORDED CLIFF DETECTION DATA

Sensor Data	Distance: Measured (Inches)	Distance: Best Fit (Inches)
502	8	8.72187069
434	10	10.17099907
379	12	11.73572065
329	14	13.62680059
283	16	15.9759353
257	18	17.68737528
229	20	19.9786666
213	22	21.56671045
199	24	23.17202136
139	36	33.84769818
91	48	52.94257476

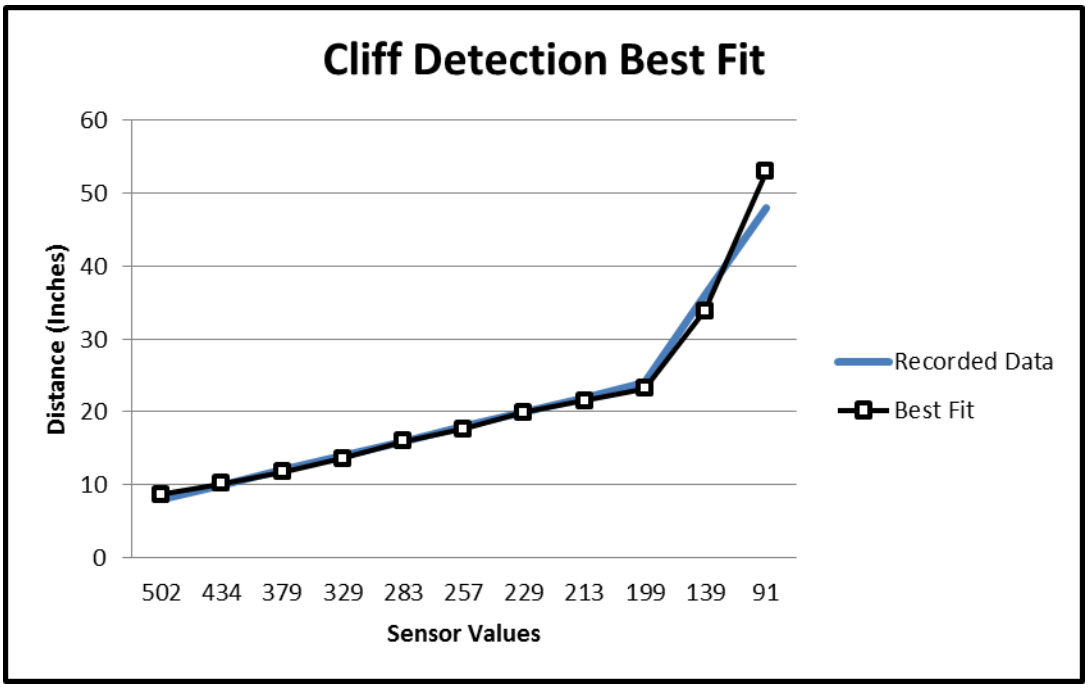

FIGURE 14: CLIFF DETECTION CALIBRATION BEST FIT GRAPH

The cliff_detection node publishes one of four states that are derived from the values of the four Infrared sensors and four Ultrasonic sensors: safe, slowdown, stop, and arbitrary. The logic for the program is such that if either one of the sensors from a set of parallel combination of an Infrared sensor and an Ultrasonic sensor gets triggered then that portion of the robot or that set would put the robot in a state of possible cliff or edge. Figure 13 represents the configuration of the sensors with respect to the robot when seen from a top view with the sensors being in from of the robot.

The minimum distance for a reading to be considered as a cliff or an edge is 12 inches vertically. The minimum value for cliff distance was chosen both analytically and by conducting research. The cliff mounts are mounted at a height of 10 inches from the ground and therefore differences in cliff reading of 2 inches or more would be treated as a cliff or edge. The program makes use of a switch case to interpret cliff states. Table 3 is a truth table for the output from each pair of sensors and Table 4 shows the overall robot state as determined by the combination of the 4 pairs of sensors.

TABLE 3: CLIFF DETECTION PAIR VALUE COMPARISON

Infrared Sensor	Ultrasonic Sensor	Set Value
0	0	0
0	1	1
1	0	1
1	1	1

Note: A value of 0 signifies that the sensor has not been triggered and a cliff/edge has not been detected yet. A value of 1 signifies that the sensor has been triggered and a cliff/edge has been detected.

TABLE 4: CLIFF DETECTION STATES

Set 4	Set 3	Set 2	Set 1	Total	State
Left-Rear	Left-Forward	Right-Rear	Right-Forward		
0	0	0	0	0	Safe
0	0	0	1	1	Slowdown
0	0	1	0	2	Arbitrary
0	0	1	1	3	Stop
0	1	0	0	4	Slowdown
0	1	0	1	5	Arbitrary
0	1	1	0	6	Arbitrary
0	1	1	1	7	Stop
1	0	0	0	8	Arbitrary
1	0	0	1	9	Arbitrary
1	0	1	0	10	Arbitrary
1	0	1	1	11	Stop
1	1	0	0	12	Stop
1	1	0	1	13	Stop
1	1	1	0	15	Stop
1	1	1	1	16	Stop

In order to draw specifications on the mounting plate for the cliff sensors and get the minimum distance needed for the robot to stop, a test mount was developed and used, as shown in Figure 15. The sensors were placed at different mounting points in the series on the mounting plate and an appropriate stopping distance was obtained and recorded by using the odometry data from the odometry package while the sensors were triggered (when a depth of the cliff distance value of a minimum of 12 inches or greater was recorded). The odometry data was obtained by first supplying a stop signal to the motor on a continuously flat surface to make sure that the robot doesn't tip over a set of staircases when first testing it and the stop distance in this case was recorded to be 7 inches. This minimum stopping distance value was then tested on a cliff/edge and the value was validated to be 7 inches from the center of the wheel. This value was used a base line for the mounting points and therefore the rear mounts were mounted at a distance of 7 inches from the center of the wheel while the front set of sensors were an additional 1.5 inches from the rear sensor in order to account for any surfaces that bear similar resemblance to the gorge between the floor of a building and the building elevator while entering or exiting an elevator which would be one of the most common scenarios associated with following ADA Guidelines.

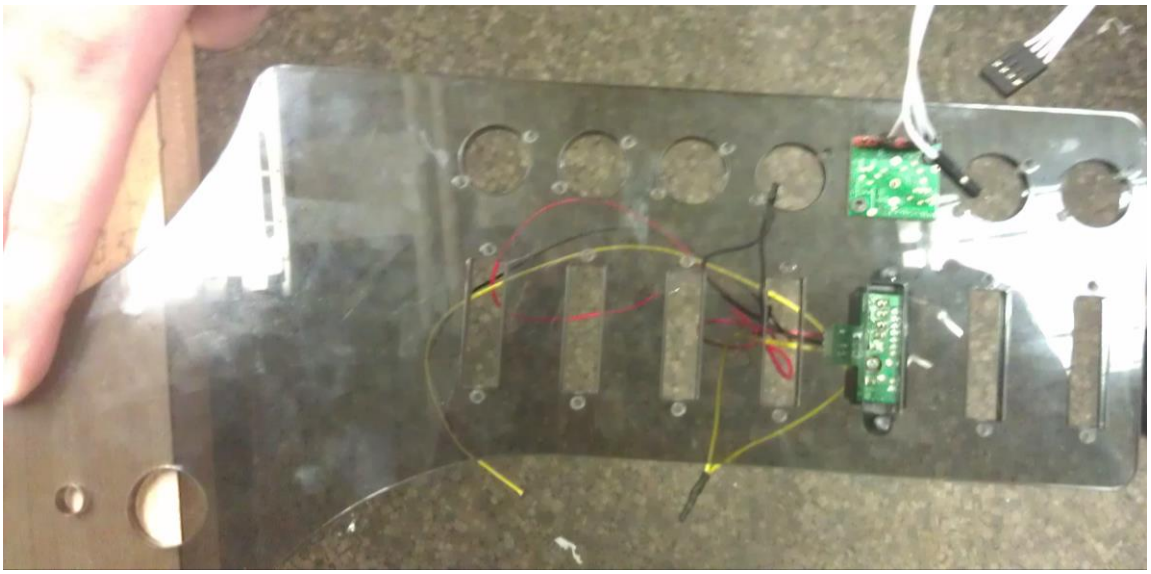

FIGURE 15: TOP DOWN VIEW OF CLIFF DETECTION TEST PLATE

TABLE 5: CLIFF DETECTION TEST PLATE WIRING

Sensor – Wire Color	Pin Configuration
Infrared – Red	5V on Galileo
Infrared – Black	GROUND on Galileo
Infrared Sensor 1 –Yellow	Analog Input pin A0 on Galileo
Infrared Sensor 2 –Yellow	Analog Input pin A1 on Galileo
Ultrasonic – Red	5V on Galileo
Ultrasonic – Black	GROUND on Galileo
Ultrasonic Sensor 1 –Yellow	Analog Input pin A2 on Galileo
Ultrasonic Sensor 2 –Yellow	Analog Input pin A3 on Galileo

Table 5 contains a table reference for the connector wiring for this sensor assembly. The electrical connections on each of the two Galileo's is such that all the 5V supply wires/ red wires for the two Infrared and two Ultrasonic sensors are soldered into one connection to the 5V supply port on the Galileo and all the GROUND wires/ black wires from the two Infrared and the two Ultrasonic sensors are soldered into one connection to the GND port on the Galileo.

The sensors were mounted to the system and tested for proper functionality. The robot passed all the tests as expected. The cliff readings for various pre-recorded surfaces were accurately depicted by the readings published by the cliff_detection node on the terminal and the robot followed the routine for safe motion, slowdown, stop and arbitrary states as expected. Final design of the sensor mounts is shown in Figure 16.

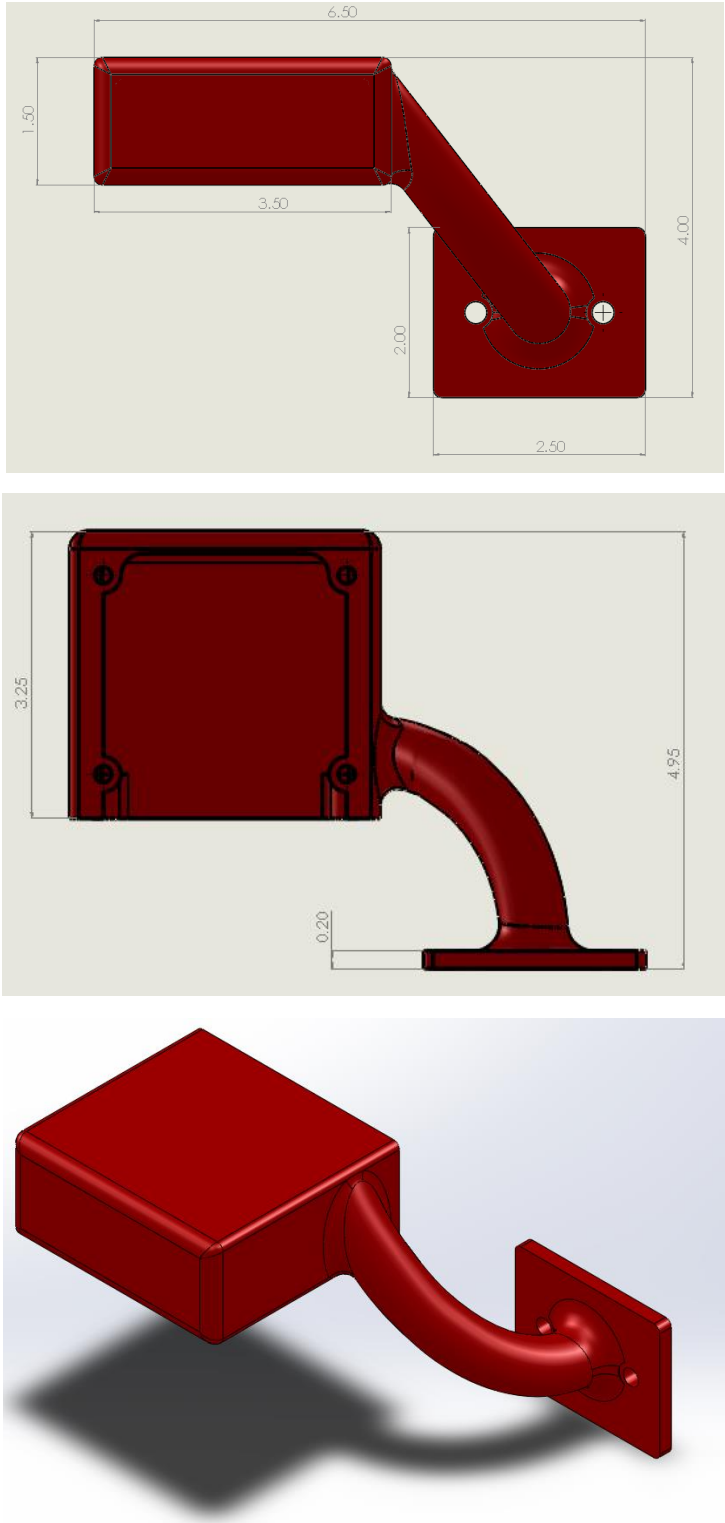

FIGURE 16: CLIFF DETECTION SENSOR MOUNTS

3.5.7 Motor Controller

A motor controller is a piece of hardware that manages the amount of current going to each motor. This in turn controls the speed and torque that the motor outputs. For safety reasons having a robust motor controller is imperative on this project. If the motor controller is not properly configured or wired, it can have disastrous effects. The motor controller chosen for our application is the Roboteq MDC 2230, pictured in Figure 17.

FIGURE 17: MDC2230 MOTOR CONTROLLER

This controller was chosen for several main reasons. The first reason was because of this controller's ability to handle two channels of motor voltage. This saves on space inside the robot because one motor controller can handle the two motors of the robot. Secondly was this controller's ability to automatically process Encoder data and perform the math for a PID (Proportional Integral Derivative) loop control. This simplifies the programming of the motion of the robot. Thirdly the controller had readily available software that interfaces with ROS from a previous project. And finally, the controller is rated to work within the currents and voltages of the chosen motors.

3.4.11 Wiring diagram

The robot is powered off of its single 25.6 volt battery. This voltage is fed into several transformers to create the appropriate voltages for powering the system. These different voltages are passed into a modular DIN rail voltage distribution system. After that they are sent to specific components. Figure 18 is a diagram of where each voltage is used.

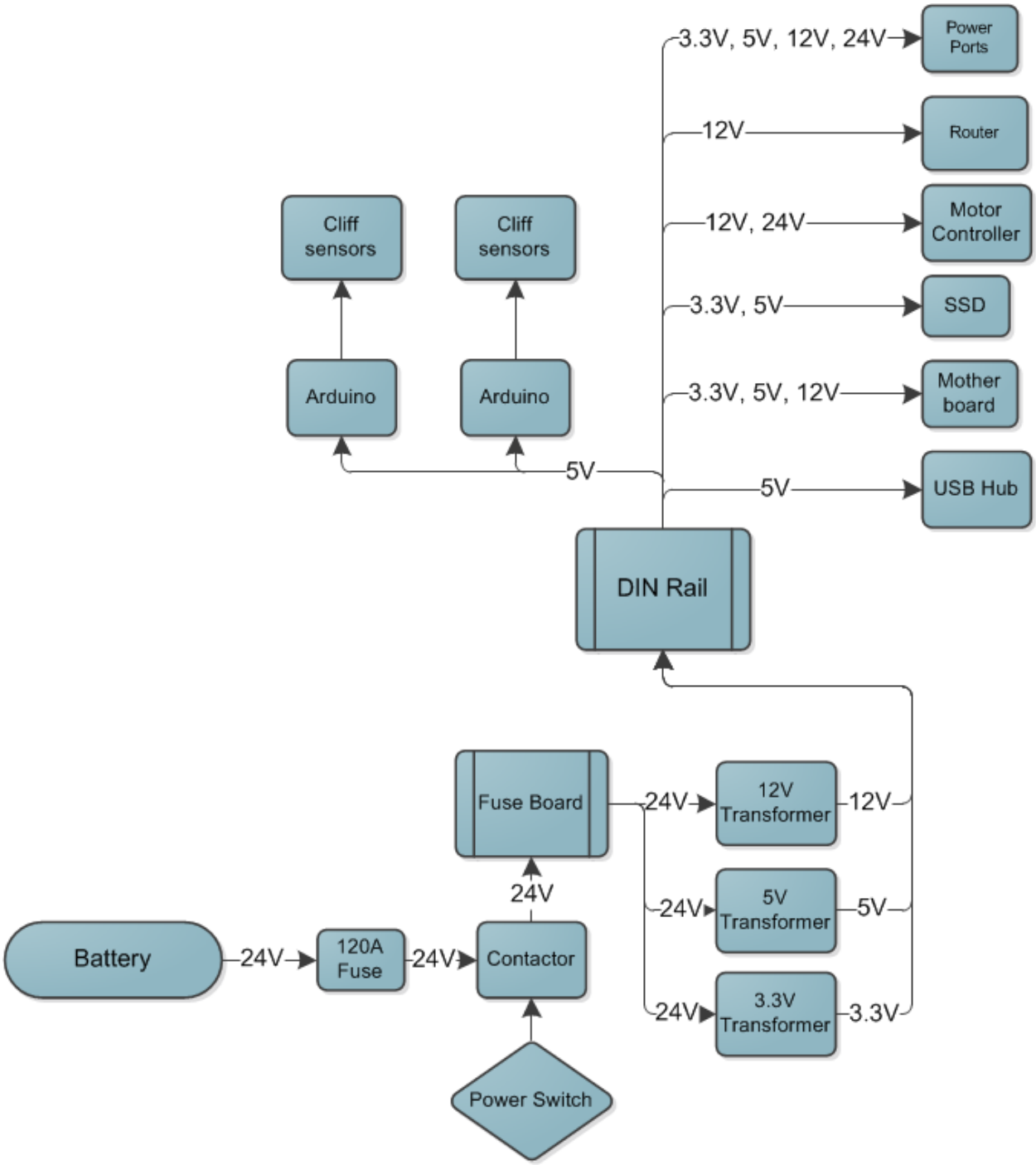

FIGURE 18: POWER CONNECTIONS MAP

3.4.12 Safety features

The components within the robot need to be correctly protected to ensure the safety of the user and to prevent damage to the robot. The first step in the overall safety system of the robot is a large 120A switch fuse, shown in Figure 19. This switch will trigger a fuse to break connection when over 120 amps is drawn through it.

FIGURE 19: 120 AMP SWITCH

The second step comes from the internal circuitry of the battery. There is a circuit that will cut off current if 40 amps is pulled from the battery. The next component is an emergency contactor switch. This switch is designed to prevent high current arcing inside of a switch. It achieves this through a vacuum, this vacuum means that are no particles for electricity to arc through. This switch is in place for a worst case scenario where no fuses are triggered and is shown in Figure 20.

FIGURE 20: GIGAVAC CONTACTOR SWITCH

After passing through the contactor switch, the 24 volts from the battery is passed into a fuse board. This board has several slots for fuses, these fuses lead to the voltage transformers. These fuses were designed to safeguard the robot in case of an overdraw of current. If a regulator draws more current than the fuse allows for the connection will break. To minimize work required by users the fuses are auto resetting. After around 10 seconds of rest the fuses will reset themselves and be ready to conduct again. To prevent damage to the motor controller a Transient Suppressing Diode (TVS) is

installed in the circuitry. A TVS prevents a large voltage spike from damaging components by absorbing the energy contained in that spike.

These components are all in place to prevent damage to the robot internally, but if there is a problem with programming which causes the robot to drive erratically there has to be a way to quickly stop it to prevent potential damage and injury. To remedy this situation the robot has a large emergency stop built in, as shown in Figure 21.

FIGURE 21: EMERGENCY STOP BUTTON

This button, when hit, will shut off all power on the robot. To re enable the robot one must twist the button to release it. It is easy to access and will quickly stop the robot in case of the aforementioned situations.

3.6 Graphical User Interface (GUI)

A GUI was developed using Java to allow for real time interaction between the user and the PARbot system. This GUI provides the user with several functionalities, including a software emergency stop, a power button option, a button to activate a tele-operation mode and a button to pause and resume autonomous follow mode. Figure 22 shows the completed visualization with the buttons in their default state.

FIGURE 22: TOUCHSCREEN GUI

The GUI is built as a Java application that accomplishes the specified function by calling a batch file designed to send the appropriate commands to the PARbot system to interpret the user input and react accordingly.

3.7 Localization and Mapping: Navigation Sensor Selection

In order to perform Localization and Mapping, data needs to be recorded to form an image or map of the environment. The most common way to get this environmental data is to use vision sensors to get laser scan data which tells the sensor about the objects around it. This information can then be used to form a map of the surrounding environment.

Vision sensors for robotic bases can be separated into two main categories: cameras and Light Detection and Ranging (LiDAR). Each system has positive and negative attributes. Cameras provide an image that can be used to generate a three dimensional array of points, where each point has a horizontal and vertical position as well as a distance value (measured from the camera). LiDAR, by comparison, provides a single row of data points evenly spaced based on the angular resolution of the selected unit. Data collected from a LiDAR unit consists of an array of distances. To compute the horizontal location of the point, the angular resolution of the unit is multiplied by the point's position in the array. The vertical position of the data points is fixed based upon the placement of the scanner on the robot. A point cloud similar to that of a camera can be generated using a LiDAR unit placed on tilt unit.

The two units considered for use in the PARbot vision system were the PrimeSense and a Hokuyo LiDAR unit. The Hokuyo LiDAR's main benefit is that it directly provides a laser scan and requires no extra computations to provide usable data. However, the downfall of the Hokuyo LiDAR is that it only provides data on a single level. This means that certain obstacles such as tables are very hard to detect.

The PrimeSense, however, provides both a picture and a depth image. There is built in ROS support allowing the provided depth image to be converted into laser scan data while taking into account obstacles at all visible levels of the camera.

The PrimeSense was selected to be used on the PARbot for Localization and Mapping for its increased versatility. Using the PrimeSense allows information from a wide range of vertical heights to be collected simultaneously, providing a cleared view of the robot's surroundings without the need for a pan-tilt unit. The image data can also be used for user tracking by detecting a body image within the array of data.

3.8 Tracking System

Tracking the user is critical to the functionality of PARbot. Color target and QR code tracking were considered by the team. Color tracking was implemented using a Logitech USB web cam, but this system had difficulties recognizing colors in certain lighting conditions. Florescent lights caused too much glare and most of the data in the image returned by the camera appeared to be white. This made tracking a color target, to square color blocks next to each other, extremely unreliable. For these reasons QR code tracking was pursued.

QR codes or quick recognition codes are a form of 2 dimensional barcode developed by Japanese car makers. These codes are designed to be read from any orientation without compromising the quality of the scan. This feature makes these codes perfect for this kind of application, its designed to stand out in most environments. Because QR codes are relatively easy to identify. QR codes have well defined proportions making it relatively simple to determine the size or location of the QR code given the other. This can be done using a two dimensional image, if the size of the QR code is known then the position and alignment markers can be used to determine the relative size of the code. Determine the relative size of the code can give the distance to the code, although cameras distort the image it can be corrected if the distortion is known. For this reason any camera used to do three dimensional QR tracking must be calibrated so that its distortion factors are known.

Tracking QR code tracking system was implemented using the Visp Vision library. This library was customized to allow for the use of multiple cameras. With the ability to use multiple cameras in place, a combination of Prime Sense and Logitech USB cameras could be used. The Logitech camera is capable of providing 1920X1080 video for QR tracking. This was not done because it images of that size could not be processed fast enough to be useful, instead a 1280X720 video stream is used. The QR code that is intended to be attached to the user’s wheelchair is approximately 8”X8”. The size of the QR code makes it easy to recognize while not being so large as to be cumbersome.

3.9 Software

The software system for PARbot runs on a Linux operating system: Ubuntu 12.04. This version of Ubuntu was selected for multiple reasons. First, this version is the latest long-term support release, meaning that it will continue to receive updates and bug fixes until the next long term support version. Ubuntu 12.04 is also very stable, with many of the bugs fixed and many resources pertaining to the Operating System (OS) are available online.

PARbot utilizes Robot Operating System (ROS) on top of the operating system. ROS has many built in packages of code to assist with the given development tasks. The built in ROS nodes execute code and publish information to "topics" which can then be accessed by other nodes and packages. This allows for easy communication between onboard systems. ROS also features launch files, which can start multiple "packages" and "nodes" with assigned parameters. Launch files allow for starting all the necessary code for PARbot using a single command. The most recent version of ROS, Hydro Medusa, a long term support version has been chosen for this project. ROS Hydro is recommended to run on Ubuntu 12.04.

3.9.1 Path Planning Software Development Process

As is the case with most software systems that designed and implemented, testing takes place in phases; adding to the software a little at a time allows for bugs to be found and dealt with earlier and more easily. This is no different with the path planning system used on PARbot.

Initially, the path planning software was written as a Python script. The use of Python allowed for simpler implementation than its equivalent in C++. This in turn allowed for the implementation's potential functionality to be tested quickly. Upon completion of a Dijsktra search algorithm and occupancy grid interpreter in Python, a simulated environment was used for testing. The use of Gazebo, a simulator in ROS, the provided Willow Garage world (a Gazebo environment representing one floor of an office building) and a simulated robot (the Clearpath Husky) allowed nearly all aspects of the path planning system to be tested in a virtual environment where more variables can be controlled. The Husky can be seen operating in the simulated Willow Garage environment on the right side if Figure 23. The left side of Figure 23 shows the resulting map from driving through the environment. The red lines represent the simulated laser scan data and the yellow arrows depict a planned path.

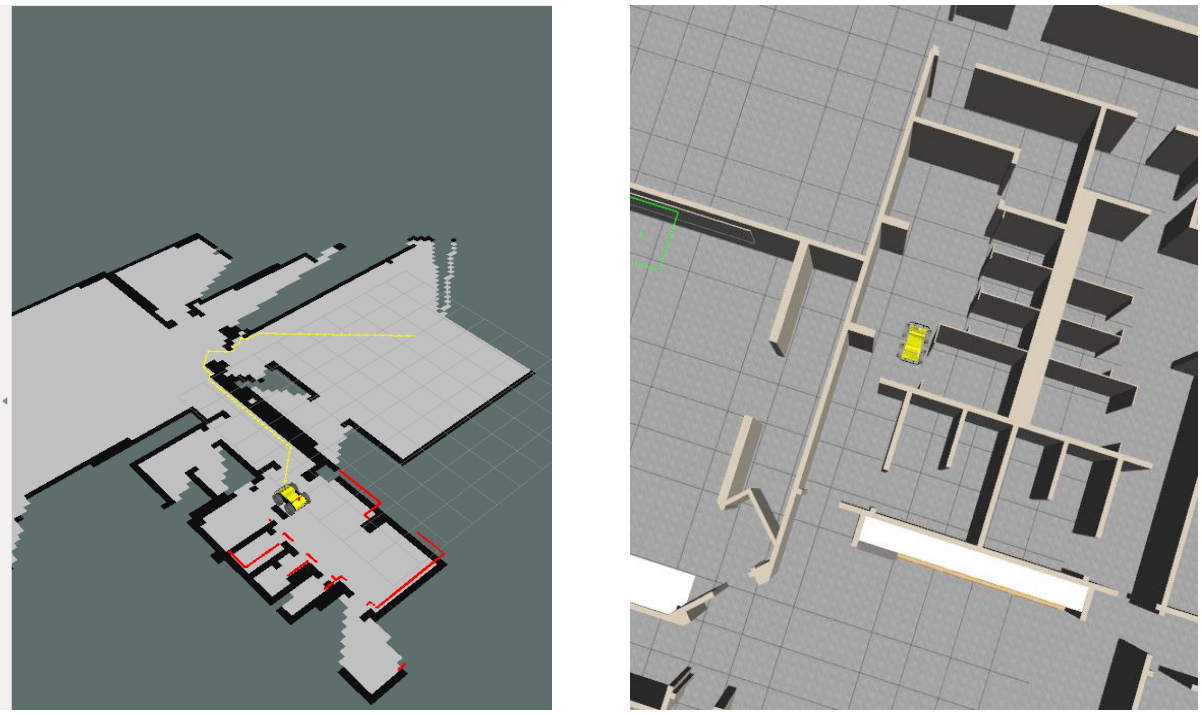

FIGURE 23: SIMULATED DEVELOPMENT ENVIRONMENT

The path planning system's results were as desired, however the performance characteristics of the software were not sufficient for a robot that would need to update its path in order to follow the user. The long execution times were exponentially related to the size of the map being navigated; with times reaching into minutes for execution of moderate sized maps (512 x 512 elements, where each cell had a resolution of 20cm x 20cm). Much of the delay was due to the memory access that Python uses to access the values of the imported occupancy grid and create a list of the searchable graph nodes. As a result of the long run times, it was decided that the code would be re-implemented using C++, a more efficient coding language with better retrieval and storage times for lists and vectors of data.

The same software structure was implemented using C++ to create a faster running path planning algorithm. This conversion saw a great improvement in the time taken to convert an occupancy grid: now about 3 seconds to convert a 512 x 512 element grid. However, the Dijsktra search algorithm execution time was not greatly impacted by change in execution language. Search times still ranged from 20 seconds to several minutes. As a result, an A* algorithm with a straight line heuristic function was implemented in C++ for comparison. The same node structure was using, due to the modular implementation of the code structure. When testing, the total run time for the path planning system remained under 4 seconds, a condition deemed acceptable by the PARbot team.

The use of simulated environment testing was used after successful implementation of the path planning system with a reasonable run time. In order to map the environment that the robot was in, Hector SLAM was utilized. This package, a freely available ROS software package would map and localize the robot in its environment. During simulated testing, the Hector SLAM software worked well, creating an accurate, clear map as well as publishing the location of the robot in the "map" frame for easy access. However, when testing with Hector SLAM began in a real environment, the node crashed whenever the robot would move and the laser scan data would not integrate properly with the SLAM unit. As a result, alternatives were considered.

Ultimately, the use of gmapping, a ROS mapping software package was found and included. Testing returned to simulation to determine its viability. The map generated by the gmapping node was not as crisp as that of hector slam, but the data provided was accurate enough to navigate with and was still quite similar to the actual environment. When used in the real world environment, the system did not crash and integrated well with the existing sensors, unlike Hector SLAM. Through tuning of the odometry system and the map resolution, a map representing the robot's actual environment could be generated reliably.

After successfully running gmapping in a real world environment, the path planning code was run on the map generated by gmapping. By setting fixed target and using the robot as the start location, the system was tested to see if how a path would be planned while the map was being built. The transition from simulation to real world environment was successful with the A* search and path creation operating as desired without need for any modifications. Table 6 shows the time required to process the occupancy grid and run a search algorithm in both Python and C++; the advantage associated with C++'s speed is evident.

TABLE 6: CODE RUNTIMES FOR 512X512 ELEMENT GRID

Programming Language	Occupancy Grid Generation	Dijkstra's Algorithm	A* Algorithm
Python	16 minutes	> 5 minutes	N/A
C++	3.5 seconds	> 5 minutes	< 3 seconds

3.9.2 ROS Software Structure

All code running on the PARbot system operates within the ROS-Hydro environment in the form of Nodes. Each Node is made up of one or more source files written in C++ or Python and serves a single purpose from planning a path to mapping the environment to simply publishing a target. This structure allows for simple communication between programs using a topic based system as well as a modular design that facilities changes in software and hardware. An overview of our software can be seen in Figure 24.

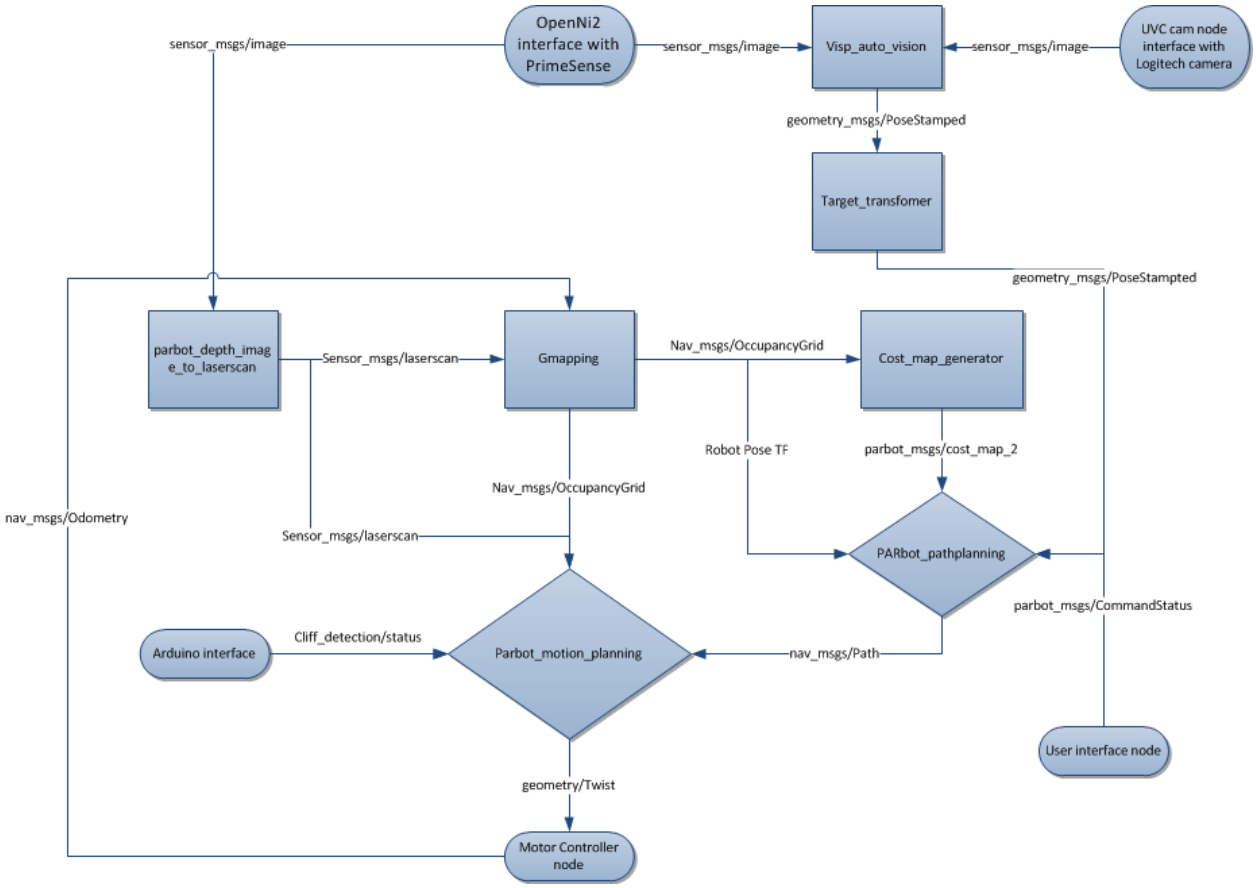

FIGURE 24: SOFTWARE FLOW OVERVIEW

3.9.2.1 Parbot_Vision.launch

The Parbot_vision node launches the three PrimeSense cameras and utilizes a modified version of depth-image-to-laserscan to take the three pixel clouds and turn them into laser scans. The node then merges the three laser scans into a single scan with transform data to be used by SLAM. Each camera is

mounted upside down to the tray surface of the robot, so when the scan data is received, it must be reversed before it can be used. Additionally, the side cameras are mounted at approximately 60 degree angles to provide roughly a 180 degree field of view. The final mounting configuration is shown in Figure 25.

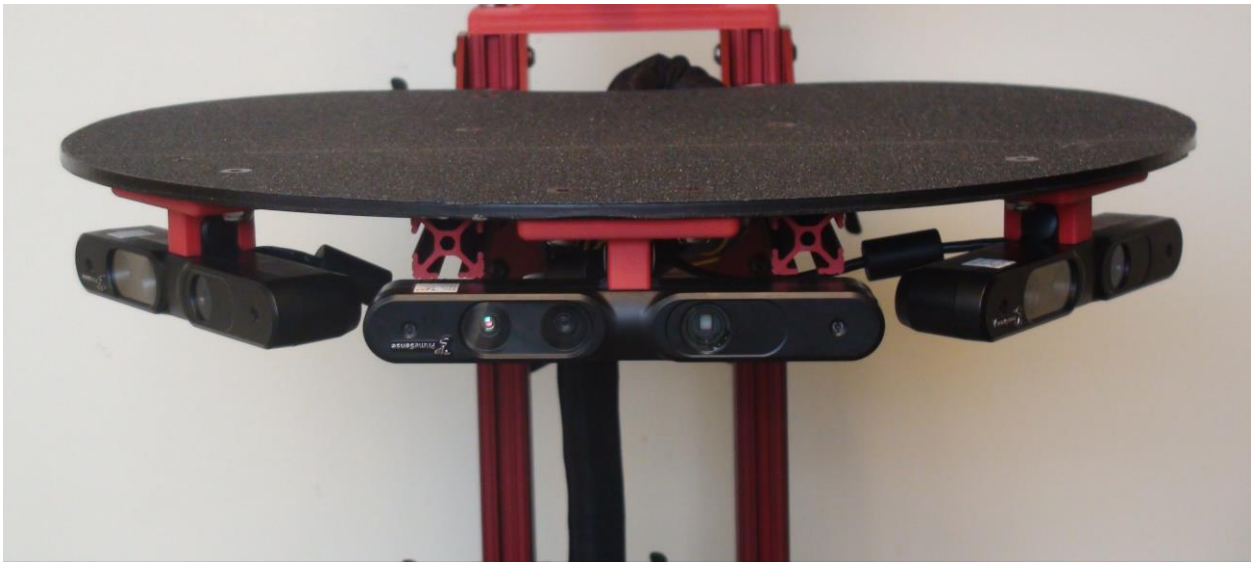

FIGURE 25: PRIMESENSES MOUNTED ON PARBOT

Additional transformations to the data are used in the ROS framework to put all the scan data in the same reference frame. The transformation data relationship as displayed in a virtual environment can be seen in the left side of Figure 26. The red lines represent the direction that each PrimeSense is facing. The red and green coordinate frame that appears behind the camera represents the base_frame of the robot. The right side of Figure 26 shows the implementation of this arrangement using a SolidWorks model, where the black rectangles represent the PrimeSenses.

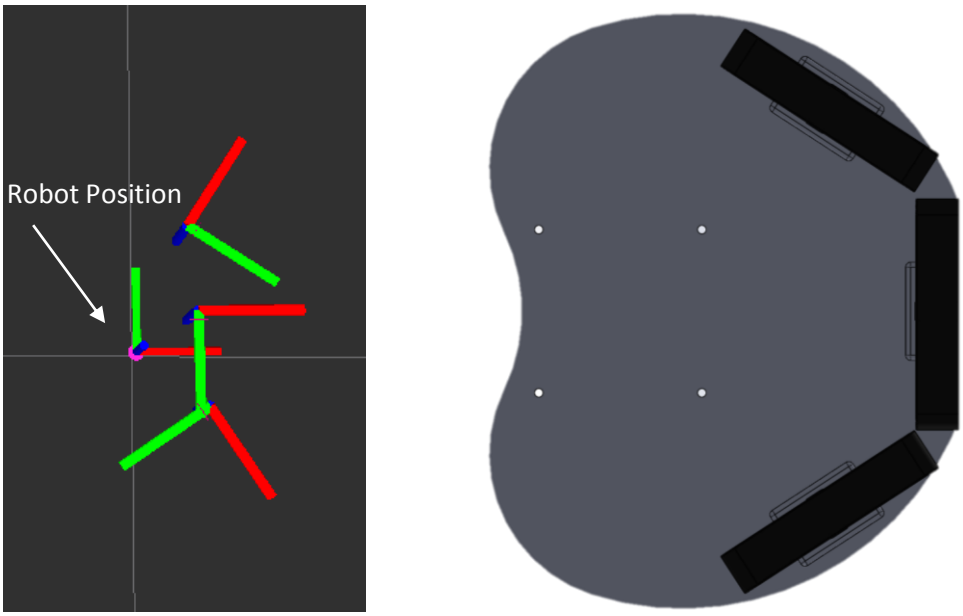

FIGURE 26: PRIMESENSE TRANSFORMATION DATA AND REPRESENTATION

After depth image data from the PrimeSenses is merged, the data can be used to visualize the operating environment by displaying the three dimensional point cloud of data that is being generated. An example of this data is shown in Figure 27. The colors associated with the data points correspond to how close the point is to the camera, with red being the closest and purple being the farthest away. In Figure 27, the series of red, green, and blue coordinate frames represent to robot's and camera's positions.

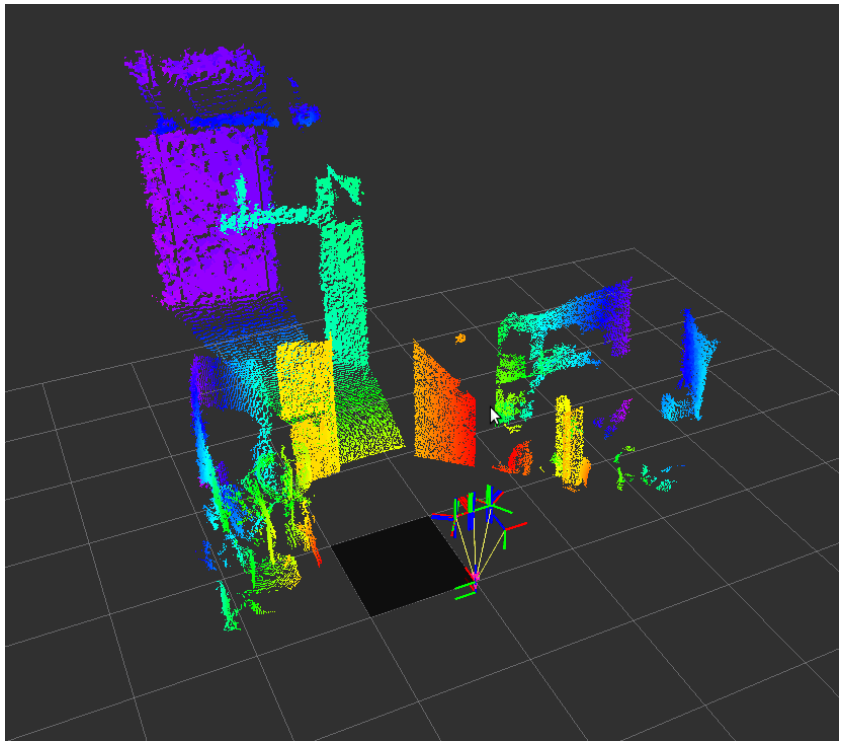

FIGURE 27: DEPTHIMAGE DATA PRODUCED BY PRIMESENSES

3.9.2.2 PARbot_pose_transformer

In order for the robot path planning and motion planning to operate as desired, the robot's current position must be made available in the "map" frame of reference. This transformation is accomplished in this node using a transform_listener. A "pose" (ROS type that contains a Cartesian point and a Quaternion orientation) with all fields equal to zero in the "base_footprint" frame is created to represent the location of the robot in its own local frame of reference – a location of all zeros relative to the robot centers the robot on itself. Next, the transform_listener is used to convert from the pose's current coordinate frame (base_footprint) to the map's coordinate frame where it can be used for navigation in a global sense. This pose is published to the "robot_pose" topic so it can be accessed by any node that needs it.

3.9.2.3 PARbot_search

Path Planning for the PARbot project is run in the PARbot_search node, implemented using C++. Here, an occupancy grid is read in from the map topic and processed into a series of GraphNodes representing the areas that are not marked as obstacles. A GraphNode is a custom class that contains information regarding points on the map, including their Cartesian coordinate location and the estimated cost to navigate there. The GraphNodes can be explored using a search algorithm; an A* search with a straight line distance heuristic is currently being used. The system is designed so a different algorithm can be implemented in the future if such a need arises.

To create GraphNodes, the X and Y coordinates of the node to be created are determined by using the map-metadata that accompanies the occupancy grid. This includes information such as the width, height (in number of cells), and resolution (meters per cell) of the map. By iterating over the map data, these are used to assign a coordinate in the map frame. This process occurs each time a new map "message" is published or as quickly as possible, depending on the current processor load. Care was taken to ensure that the code to convert an occupancy grid only relied on information gathered from the map itself so the code can be used with maps of varying sizes and resolutions.

Upon completion of generating a series of GraphNodes, the start and target locations (provided as X,Y points) are identified as specific GraphNodes for use in an A* search. The start and targets are variables specific to the PARbot_search code that are updated using ROS subscriber functions. For example, the start location is always received from the topic publishing the robot's current location and the target is the location of the user (with some built in personal space to ensure the robot does not hit the user). If either GraphNode is not defined, the closest GraphNode to the desired location that does exist is selected. Next, the complete list of GraphNodes as well as the Start and Target GraphNodes are passed to the search algorithm. The result is processed into a series of poses (a ROS type containing a coordinate for location and quaternion for orientation) which make up a Path message. This message contains the series of waypoints that the robot is intended to follow and is published to the parbotPath topic for use with motion planning.

3.9.2.4 Target_sim

In order to test the path planning in simulation and in the real world a target simulation node was required. In its final form the robot gets the target that it needs to navigate to from the target tracking node. To test the path planning separately, a mock target needs to be provided. This node

simply publishes a target pose message to the “Tracking/object_position” topic; the same topic that the target tracking node publishes to. For simplicity, this pose message is in the “map” frame so that no transforms are required.

3.9.2.5 PARbot_gmapping.launch

Generating a map of the operating environment is necessary to navigate safely to the target location. The ROS navigation stack contains a mapping tool called gmapping. This operation uses laser scan and odometry data to generate map of the environment. The map is made up of cells which are black (occupied / obstacles) or white (open space). This launch file opens the gmapping code with the parameters desired for the PARbot application; this includes setting the desired resolution of the map and the transform that represents the robot’s chassis. The gmapping system publishes and occupancy grid containing map data to the map topic as well as the “map to odom” transformation which relates the robot’s location to the map being generated.

3.9.2.6 Motion Planning

Motion planning for PARbot is intended to ensure that the robot moves safely along its intended path. This includes making the robot move smoothly even if the path is not smooth, while avoiding obstacles that may not have been added to the map. This means that motion planning must be fast and accurate when calculating PARbot’s path. To accomplish this task it was decided to look at how others had solved these problems. A paper from the University of the Armed Forces in Munich showed a plausible solution, tentacles [41].

Tentacles are paths that represented by circular arc segments, these segments can be followed

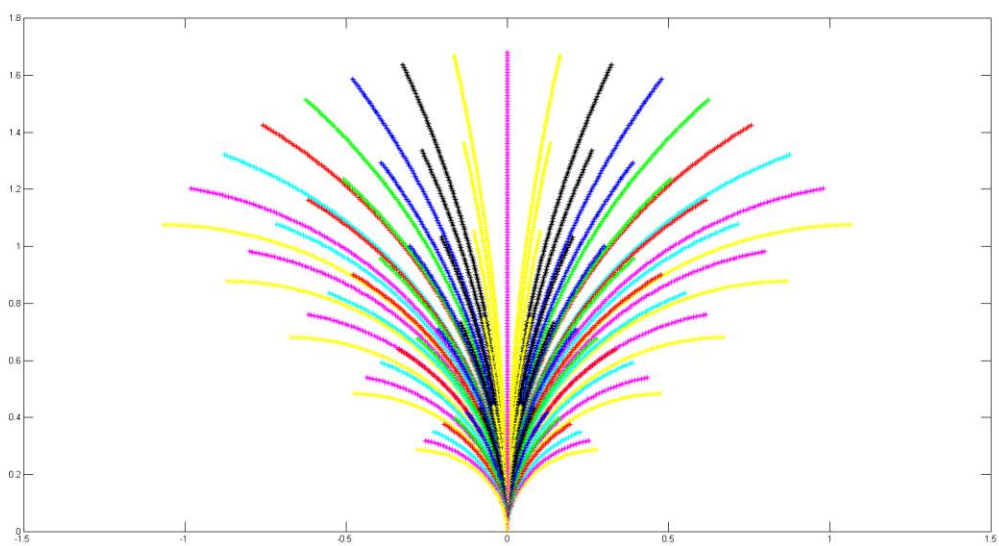

FIGURE 28: FIVE VELOCITY SETS WITH TEN TENTICLES EACH

by the robot by following constant linear and rotational velocity. For use with PARbot’s motion planning the tentacles would be calculated for the next second. Figure 28 shows a subset of the tentacles that PARbot will test to determine what immediate direction it should travel.

For each tentacle a collision matrix is generated. This matrix is based on a grid with an origin at the robots turning center. To generate these matrixes the paths had to be broken up into many steps, each the length of a grid cell. At each step the predicted orientation and position of the robot was used to project a representation of the robots footprint at that point. This meant that many points overlapped; additionally at small grid sizes this process is extremely computationally expensive. Because of these issues all of this was done during initialization. The advantages of this system are clear, since the robot only has to find one path pre-calculating many options makes the planning faster. Figure 29 shows the relationship between the calculated tentacles (grey), the desired path (red), and the path (blue) that the robot (here shown as a car) will take to reach its goal state [41].

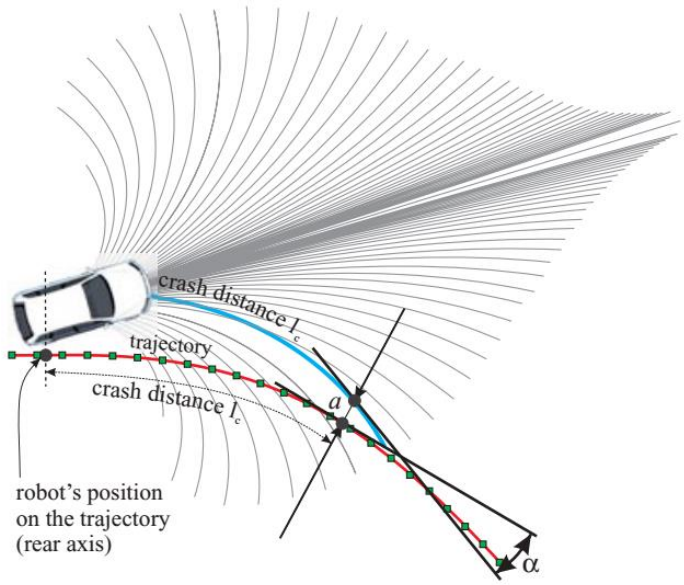

FIGURE 29: EXAMPLE OF TENTACLE BASED MOTION PLANNING

To test each tentacle both laserscan and map data would be used. This data needs to be converted to the same grid structure as the tentacles use. To do this handler objects were created to convert these external messages into internal data structures that can easily be used to find the best path. These handlers are designed to be reconfigurable if needed in other applications. The constructors use optional parameters so that only the parameters that need to be changed are given. This development method allows for easier reuse of code. A downside to this system is the increased complexity of building. The CMake for the package includes many libraries that must be compiled before the executables can be compiled.

Once the data has been converted to the correct format for motion planning it is used to determine which of the available paths is the best option. To do this is scores each path, or tentacle, that is safe on how closely it matches the path given by path planning, and how safe it is. Matching the given path is done by finding the straight line distance from points on the given path to points on tentacle. An example of selecting a tentacle to match a provided path can be seen in Figure 30.

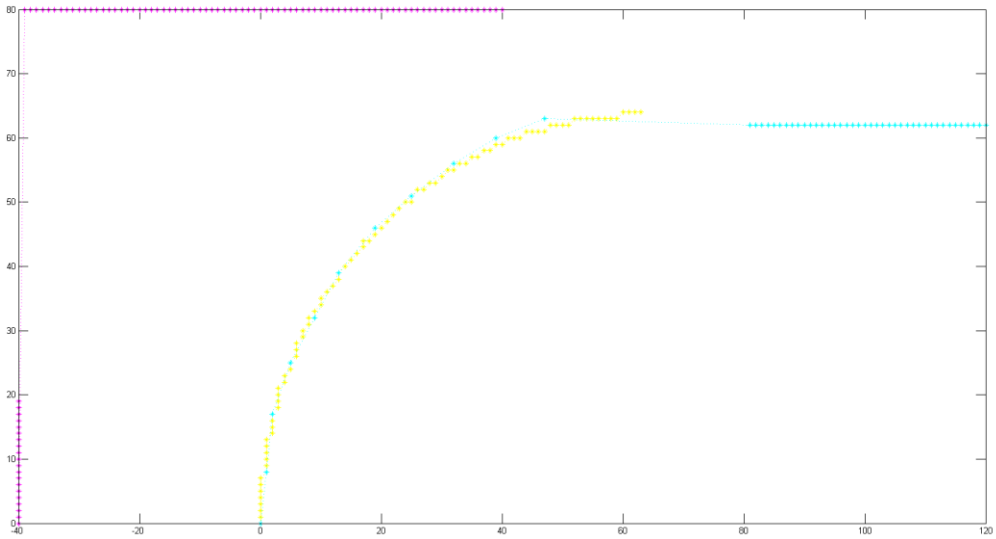

FIGURE 30: THE TENTACLE SELECTED BASED ON THE BLUE TARGET PATH AND THE PURPLE OBSTACLES

Determining safety is a more complex problem, in motion planning this is done by using and internal cost map. This is handled by the cost_mapper object, it takes a set of points relative to the robot and expands them using the curve shown in Figure 31.

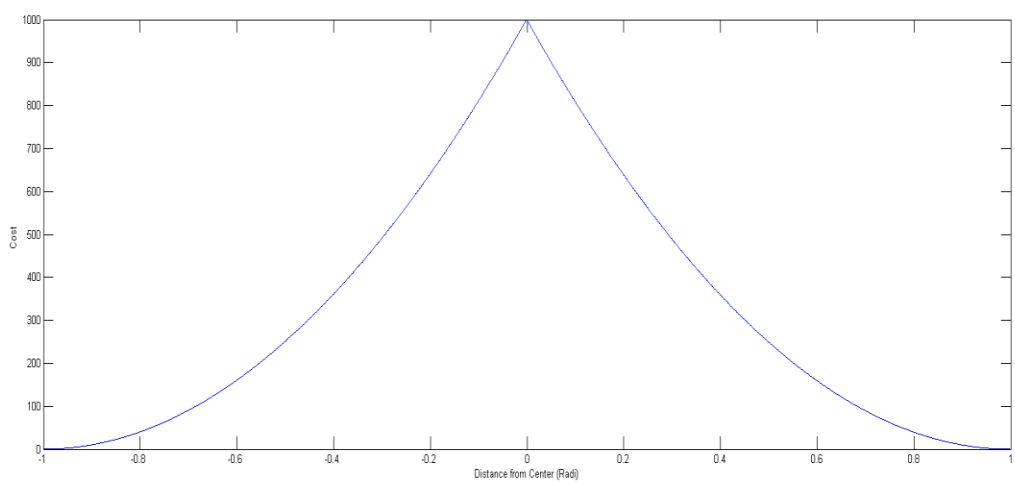

FIGURE 31: COST CROSS-SECTION

After cost expansion is complete the path can be scored. This score is divided by the number of points in the path, thus the score does not discriminate based on the length of the path. The impact of the cost expansion is that the robot can find the center of things such as hallways and doorways, and will keep a safe distance from any obstacles around it.

Motion planning is set to run at 10Hz, this means that each time a motion path is chosen it will be revaluated nine times before it's completed. This means that the robot has many opportunities to ensure that the path it is on is a safe one. Another consequence of this system is that the robot will reduce its speed as it approaches its destination. If PARbot is following a user the destination will be constantly moving, this means that the PARbot will attempt to stay one second behind the user at all times. This makes matching the users speed almost trivial.

3.9.2.7 User Tracking

In order for this system to follow a user and assist them PARbot must be able to locate the user and localize them on the map. Two different methods were considered to achieve this functionality: tracking a QR code using the visp_auto_tracker, a part of the visp_vison ROS package, and tracking a color pattern using a Pixy [42].

3.9.2.7.1 QR Code Tracking

The first method of tracking implemented was tracking the QR code. Originally, a PrimeSense was used for this, until it was realized that the camera in the PrimeSense had too low of a resolution to be effective. The resolution of 640 x 480 made tracking QR codes at longer distances impossible. A Logitech web camera, model C930e, with a max resolution of 1920 x 1080 was then utilized and proved to have much better results. The web camera is run at 1280 x 720 instead of full resolution, however, as running at full resolution quadruples the run time and only adds about 6-8 inches to the range. At 1280 x 720 resolution the approximate range is 8.2 feet.

In order to track the QR code the visp_auto_tracker package was utilized. This ROS package uses a camera to track a QR code by providing a ROS stamped pose message with the code's location. This stamped pose can then be transformed from the camera's frame of reference to the map's frame of reference for user tracking. In order to target the QR code and get the proper distance, a model that includes the dimensions of the code must be included. This information must be known to calculate the distance of the code based on its registered size. The tracking code is run every time the camera provides it a new image.

There are two main drawbacks to this QR based tracking system. One, that the user would be required to wear an unattractive QR code on their person, which to some users would be considered unacceptable. The second major drawback is that this system's range is highly dependent on the lighting in an area, and would not function at all in the dark.

3.9.2.7.2 Pixy Tracking

The second method of tracking used was tracking an object using a pattern of colors that can be easily detected by the Pixy vision sensor. Pixy can detect up to 7 different color signatures, which means that objects can be detected by the pixy that bear a pattern of all the seven signatures. For Pixy has a robust color filtering algorithm that enables it to easily identifying the color signatures and has a resolution of 320 x 200. In addition, Pixy features a variety of configuration features like adjusting the brightness, minimum saturation, hue spread, etc. that allows differentiating objects of similar color signatures easy.

In order to track a particular object, the signature resembling the object needs to be taught to pixy and this is done by installing PixyMon software package which provides the tools to achieve this task. The software has features that make it easier to teach pixy a particular object by simply taking a picture of the object in its surrounding and creating a boxed area around the edges of the object and pixy then detects the object in the frames from the continuous video feed of the pixy camera. The Pixy vision sensor is connected to the Arduino microcontroller, as seen in Figure 32, which is programmed to serially transmit all the detected blocks for the color signatures taught and their signature number, x, y, width and height values in pixels to the ROS node that further interprets this data to estimate the position of the object that needs to be detected.

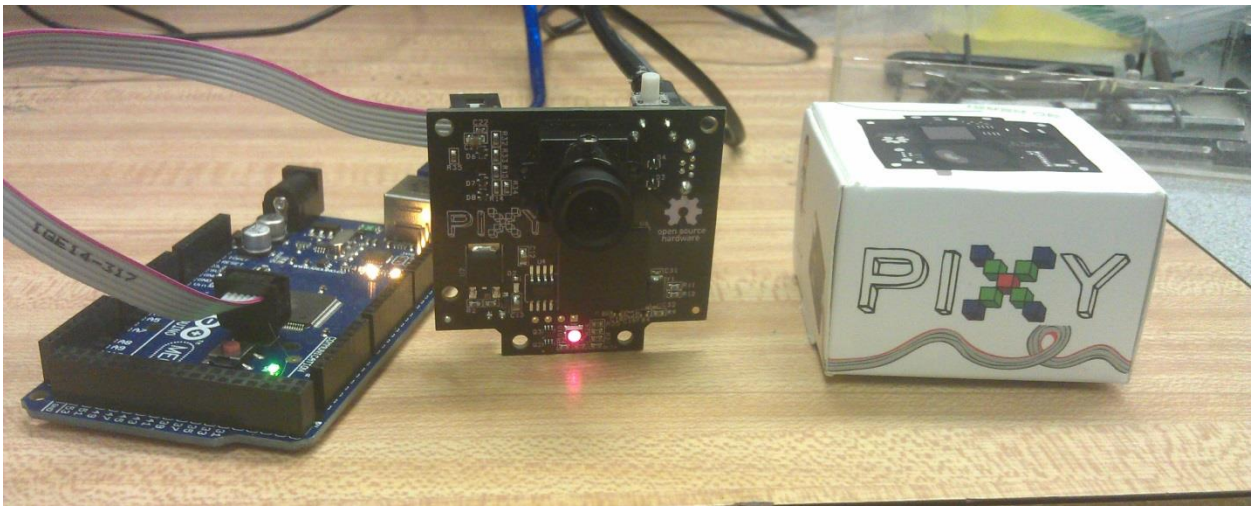

FIGURE 32: PIXY CONNECTED TO ARDUINO MICROCONTROLLER

One of the problems that is faced while teaching pixy different color signatures is that configuration parameters such as brightness cannot be applied to detecting every color signature and in order to overcome this problem, the ROS node has been makes use of an algorithm that uses position comparison of different blocks detected by the pixy to detect the set of blue and red block placed adjacent to each other that resembles our object as shown in Figure 33.

FIGURE 33: OBJECT TRACKING COLOR PATTERN

Practical experimentation resulted in the following relationship between X (horizontal), Y (vertical) and Z (distance to the object) axes; at a distance of x units from the object, the frame is 2x units wide and x units high and this was used to detect the x, y and z coordinates of the object in terms of distance from the center of the camera. After the object has been identified the average pixel values for the two color signatures or in certain cases one color signature is interpreted in terms of meters and a stamped pose message is published stating the position of the object with respect to the center of the camera and the axes following the right-hand thumb rule.

4 Results

The PARbot project took place over one academic year of study and resulted in the creation of a modular robotic base as well as a working code base with features such target tracking and path planning. The robot base itself is differential drive and is capable of travelling at speeds up to 2.5 meters per second, or 5.5 miles per hour. The footprint of the robot 24 inches long and 27 inches wide, with a low profile; the main component housing is 11 inches high. This small design allows the robot to have a non-threatening presence in its operating environment. Finally, the robot has a foldable design which allows it to be stored or transported in a small location.

The software developed runs in a ROS-Hydro environment. The code developed provides functionality for path planning and obstacle avoidance, with safety measures to prevent collisions. Further, the cameras mounted to the robot allow the robot to generate a map of its environment.

4.1 Foldable

To allow for users to easily store and transport the robot, a folding feature was built into the design. Figure 34 demonstrates the open configuration and closed configuration. Open is used for having the robot navigate and move. Closed configuration allows for the robot to be placed inside a vehicle. It also allows for the robot to be more easily stored in a home.

FIGURE 34: OPEN AND CLOSED CONFIGURATION

4.2 Modularity

The PARbot system features a modular design to allow for the robot to better meet the user's needs. Rather than over engineering a system that far surpasses the average user's needs in an effort to meet every potential user's needs, the PARbot system is designed as a base platform that can have individual modules added to fit the user, and only the user, needs. This is done through several design decisions.

4.2.1 Hardware

Firstly, the robot features power connectors for the most common standard and hobby voltages (3.3V, 5V, 12V, and 12V) to ensure that any new modules need to fit the user's needs can be powered onboard the robot. Figure 35 shows the pin out configuration of the waterproof power connection. The robot also features eight USB 2.0 and three gigabit Ethernet ports that connect to the main computer. With access to power and data adding new modules that communicate with the robot is simple.

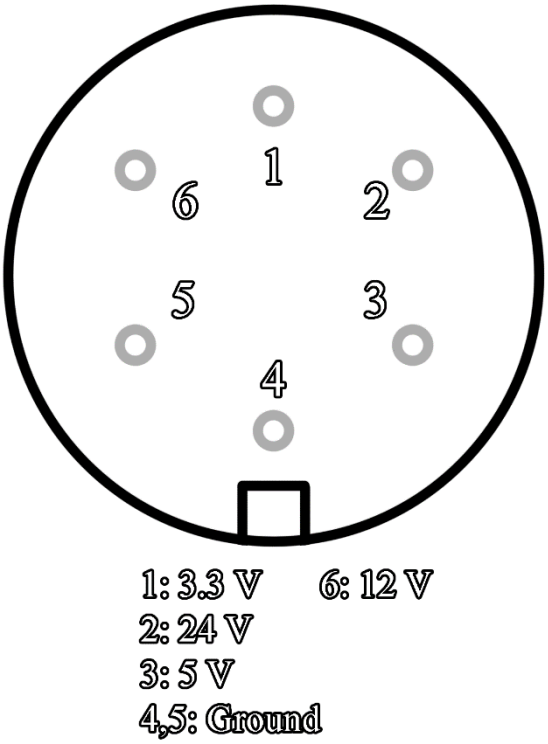

FIGURE 35: POWER CONNECTOR PIN OUT

In order to add these modules, though, they need to be able to be mounted the robot. To do this, a through-hole plate with holes forming a one inch grid, is attached to the robot as seen in Figure 36. In the figure, the plate is being used to mount the Intel Atom Development Board which is a part of the user interface module**.** In addition to a dedicated mounting area, the robot is made of extruded aluminum, which features channels that are easy to mount to. This is especially helpful for modules that may weigh a lot and would need more support.

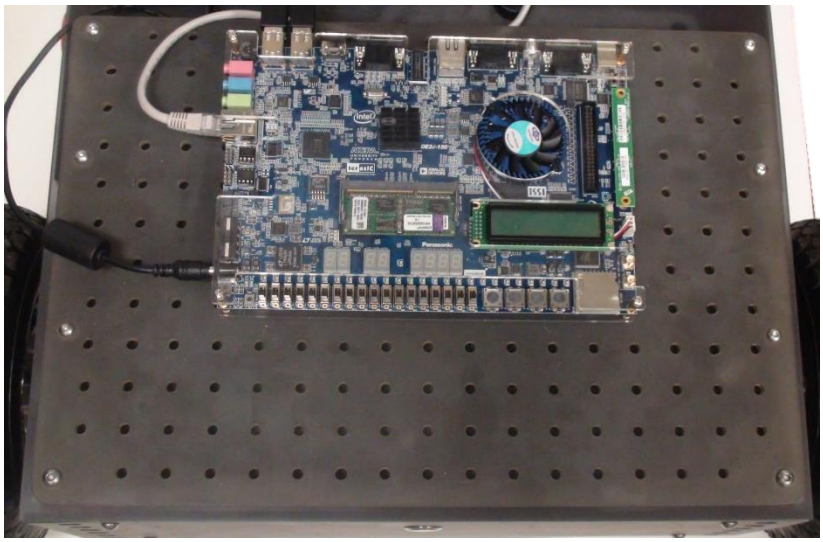

FIGURE 36: MOUNTING PLATE

There is another feature of the extruded aluminum that is used for the frame of the robot. Because of how the aluminum frame fits together it becomes possible to increase the vertical size of the case that houses the robot's internal electronics without changing the overall height of the robot. This allows the user to have more space inside the robot if internal modules are a necessity.

4.2.2 Software

In addition to the modular approach taken with the hardware aspects of the robot, software modularity is also an important aspect of the project. The use of ROS allows for the implementation of highly modular software. This is made possible by ROS's simplified inter-process communication system. This system is based on message passing. These messages are organized into "topics" which allow for quick and easy replacement of components called "nodes." In order to receive or make data available, all one needs to do is instantiate a publisher or subscriber, and in the case of a subscriber a callback that handles the data. Since ROS handles the inter-process communication future development can be focused on accomplishing new tasks without worrying about how data is being handled elsewhere.

ROS also makes it possible to change out sensors without making any changes to the existing code other than the code needed to interface with the sensor and provide the same data type. For example, the QR tracking code executes by finding the location of the QR code and publishing a stamped pose ROS message to make this information available to the Path Planning code. Switching from a QR based tracking system to a color based tracking system can occur with no changes to the software structure as long as the Pixy ROS node publishes the same type of message that the QR node published, a stamped pose message, and uses the same topic. In this way the software is just as modular as the hardware.

4.2.3 Graphical User Interface (GUI)

The simple and user friendly GUI allows for the operator to interact with the robot in real time, without needing any knowledge about the code needed to run the PARbot system. The GUI is designed to be simple to operate and to give basic users the ability to enable disable a few basic robot feature.

The basis of a Java application for the GUI allows for easy changes to be made to incorporate new features or to improve the currently existing ones. Currently, the GUI can enable and disable the following mode, enable and disable the teleoperation, provide a software emergency stop in addition to the hardware stop, and provide a shutdown command to the entire system.

4.3 Power Capacity and Consumption

To insure a usable operation time, the electrical system demands must be determined prior to battery selection. Since 4 voltages are to be provided on-board, the total draw must be calculated at the battery voltage, 24 Volts, for each transformer at peak draw. Table 7 shows the maximum current draw of the transformers used, accounting for their efficiency. The maximum draw of this system is 24.46A.

TABLE 7: TRANSFORMER POWER CALCULATIONS

Item	Voltage (V)	Current (A)	Quantity	Power (W)	Efficiency (%)	Current @ 24VDc
3.3V line	3.3	5	2	33	95	1.44
5V line	5	3	3	45	95	1.97
12V line	12	20	2	480	95	21.05
					Total Draw	24.46

The calculations in Table 7 do not include the motor current, as they operate at the battery voltage. Table 8 shows the motor's current consumption in various operational conditions. The stall current is the maximum draw by the motor when the output shaft is not allowed to rotate. The No Load current shows the current use when the motor is allowed to spin freely. The Rated voltage is the motor's current consumption when it is operating in its designed capacity. This is the value that will be used for computing the runtime of the robot.

TABLE 8: MOTOR POWER CALCULATIONS

Item	Voltage (V)	Current (A)	Quantity	Power (W)	Efficiency (%)	Current @ 24VDc
Motor (Stall)	24	20	2	960	95	42A
Motor (No Load)	24	0.9	2	43.2	95	1.89
Motor (Rated)	24	4.6	2	220.8	95	9.684

Based on the data presented in Tables 7 and 8 the maximum operating total current draw of the robot at 24V is 34.144. Since the battery is rated for 20 Amp Hours, this gives a run time of only 35 minutes. It is important to keep in mind, however, that this is under high draw operation. Although this current draw is over the 40 Amp limit of the internal battery safety circuit, this situation should not occur. In the case that the 40 Amp draw occurs the battery will shut the system down to protect itself.

Through real world testing and operation and repeated trials, the robot battery has been shown to last for 5 hours or more of operation. This included having the robot drive on a level surface for nearly half of the time and running both the Atom Module and the on-board Intel i7 computer. In real world

operation, the robot can be expected to exceed 3 hours in its current configuration. The additional power available in the system is intended for use by future modules.

4.4 Cliff Detection

The cliff sensing module accurately detects any cliff or edges that come along the path of the Personal Assistive Robot while it is operational. Any ranges that exceed the depth of 12 inches are registered as a possible cliff by the robot and the system control software accordingly decides whether the current state of the cliff modules depicts safe movement or whether the robot needs to slowdown or stop. As shown in the Figure 37, the terminal running the cliff detection node publishes an int32 value to represent the state of the cliff detection module with respect to the surface around the robot and in this case the node publishes a value of zero signifying safety in its operational movement as the robot continues to conduct its tasks. The terminal initiates the data stream by printing the states of the robot corresponding to their int32 values that would be parsed by the motion planning node for conducting its routine.

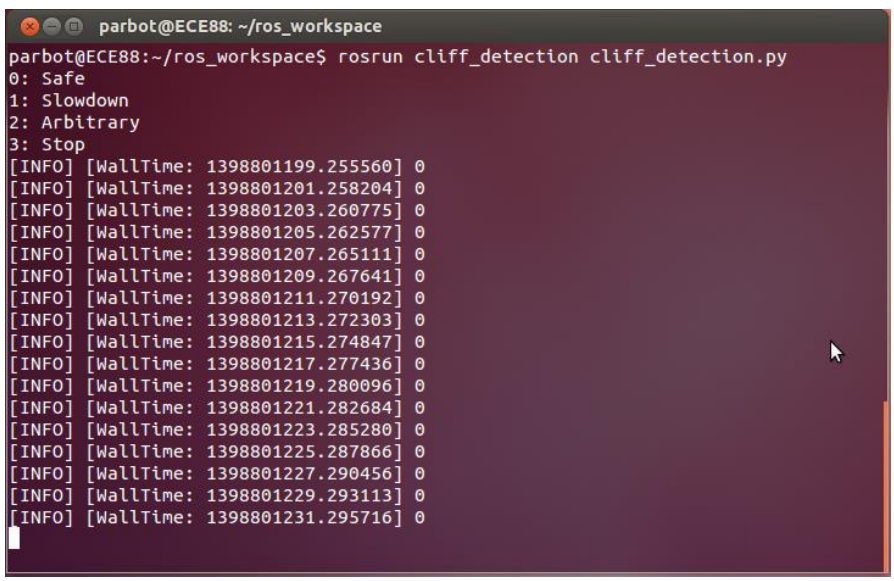

FIGURE 37: CLIFF DETECTION DATA STREAM

Currently, the cliff detection system is capable of recognizing a dangerous location and can set a flag to stop the robot; however this flag is not "listened to" in the robot's autonomous drive system. However, listening to this flag can be implemented in the future as it does not require the addition of any hardware or complex software.

4.5 Mapping and Path Planning

The PrimeSense cameras mounted to the underside of the robot tray system use a gmapping system to create a map and localize in its environment. Once the map is generated, it is interpreted by a cost map generator and used to determine the least dangerous path for the robot to take. Additionally the map is used in the A* path planning algorithm to provide a more long term goal for the robot to reach. The map shown in Figure 38 displays a map of the second floor of Atwater Kent Laboratories at

Worcester Polytechnic Institute, as generated by the gmapping system. The red line on the right side of the map shows a path planned by the A* system.

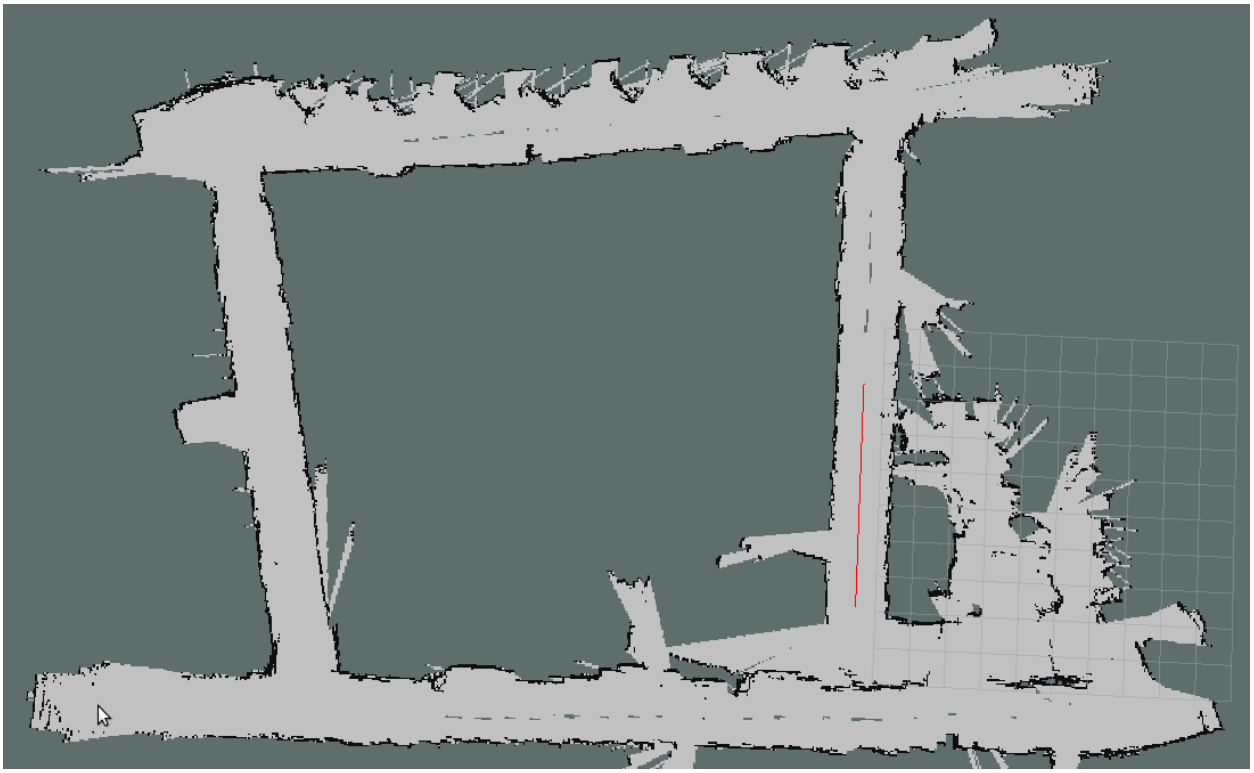

FIGURE 38: ATWATER KENT SECOND FLOOR MAP WITH PATH

4.6 User Tracking

Tracking the user is crucial to the functionality of PARbot; because of this multiple approaches were used. The first system implemented was the QR code based user tracking system. This system worked reliably and accurately. An offset was included it give user some personal space. Figure 39 shows the data collection process for a target using a QR code.

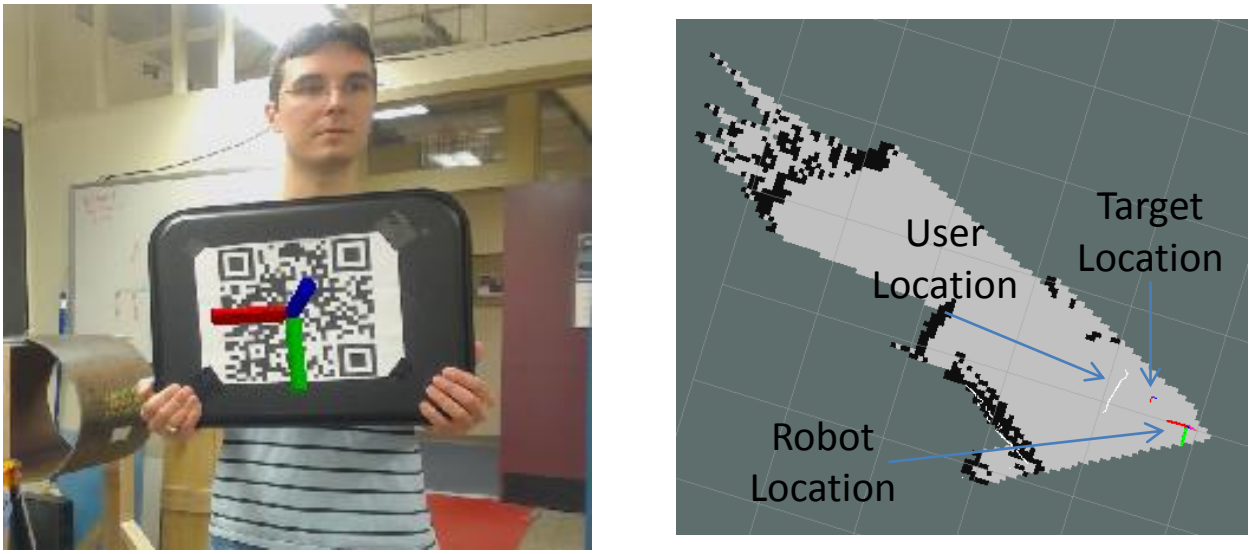

FIGURE 39: QR TRACKING TEST

The QR code is approximately 8”x8” and is shown in Figure 39. It is attached to the back of a cafeteria tray in the image to keep it flat during testing. The QR tracking system is sensitive to the surface the QR code is on. That is if the QR was on bent paper detecting the QR was much more difficult for the system. In addition to this issue the camera used was never designed to be used on a moving platform and there for has trouble focusing while the robot is in motion. This causes the robot to temporarily lose the user when the robots turns quickly. These issues proved to be less critical than initially thought, and in the end the system was able to track users at close range and low speeds. After tuning the map so that the user did not become an obstacle in the map, the system worked very well.

Pixy efficiently tracks the object in terms of the two color signatures as shown in Figure 40.

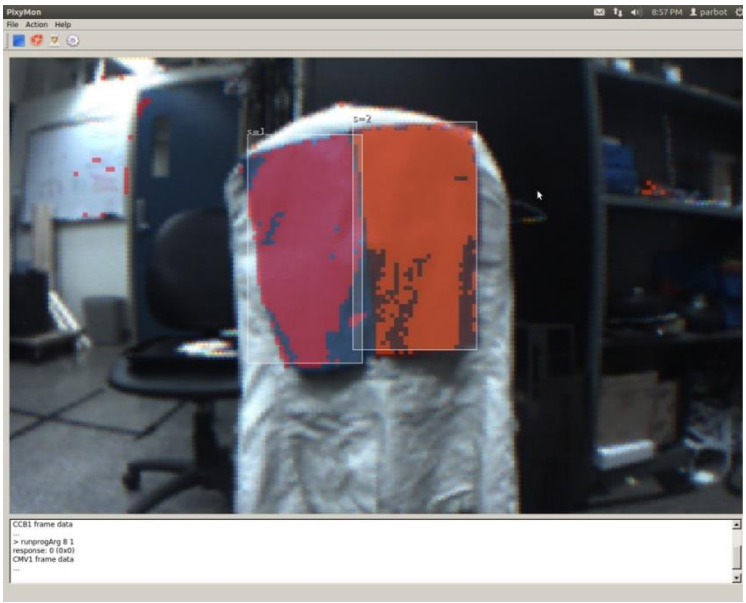

FIGURE 40: PIXY COLOR TRACKING

The following table shows the data obtained regarding the parameters of the block detected and the distance of the block from the camera. A relationship was drawn between the area of the block and its distance from the camera using regression testing. This could have very well been accomplished using the line fitting features that Matlab offers.

Width of Block (pixels)	Height of Block (pixels)	Area of Block (pixels)	Distance from Pixy (inches)
63	63	3969	24
58	58	3364	26
56	56	3136	28
51	51	2601	30
48	48	2304	32
46	46	2116	34
44	44	1936	36
42	42	1764	38
40	40	1600	40

The following equation, draws a relationship between the distance of the object to its area as detected by pixy in pixel counts.

Distance in meters = ((-0.006644502*Area) + 48.825)*0.0254

The position values are then published using a stamped pose message from the geometry messages package that is inbuilt into ROS. Figure 41 shows an image of the terminal showing the position of the object being published.

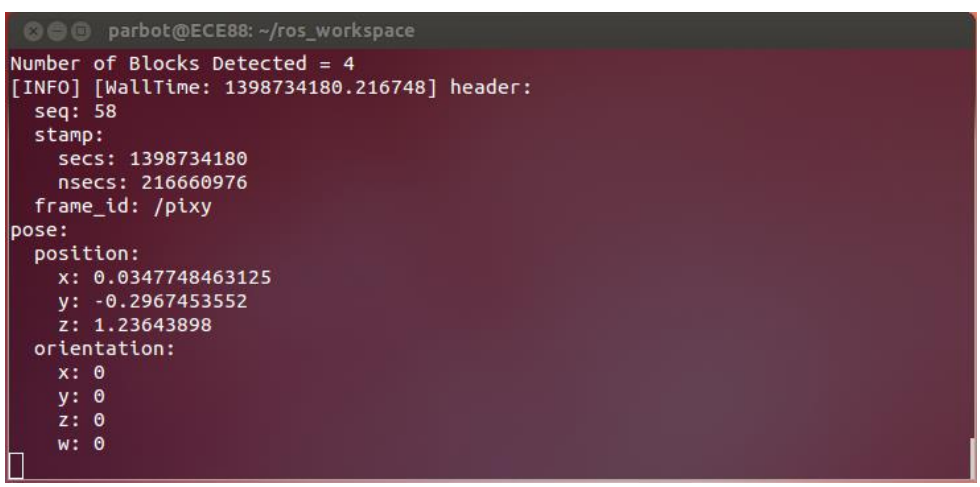

FIGURE 41: STAMPED POSE FROM PIXY

4.7 Motion Planning

For PARbot to perform any of the tasks required this year and future years require the robot to move quickly and safely while confronted with dynamic and complex obstacles. Motion planning

determines how to move the robot based on the data generated by the rest of the software system. This system required many of the other software systems before it could tested on the robot. For this reason a great deal of testing was done by simulating data from other systems. Once these problems were solved PARbot moved in a quick, safe, and smooth manor. PARbot is able to reliably navigate doorways that are only a few inches wider than itself. Hallways and other large spaces are handled with confidence and precision. Smooth motion is even achieved at high speeds, only sharp corners cause the robot to stumble. When the robot turns quickly it may pause to ensure that it is safe to proceed. This process takes one to three seconds and is mostly due to the slow map updates. Motion planning does what is intended to do, move quickly, safely, and smoothly to the user or another target.

5 Conclusions

The need for assistive care is projected to increase steadily over the coming decades. With this increasing demand for care will come with an increase in cost as the available supply decreases. In order to supply an alternative to personal care, the PARbot team has developed a Personal Assistive Robot that aims to increase a user's independence and decrease, if not eliminate, their need for a personal aide. PARbot's aim is to allow individuals with disability to maintain their lifestyle and independence for a greater amount of time.

The implementation of the PARbot system has helped to identify some of the potential challenges associated with developing an assistive care robot. Size and weight are the two largest challenges; the robot needs to be large enough to be useful while still remaining compact and non-threatening to a potential user. In complex systems where space is a premium, component selection and layout are crucial. Many of the space issues this project faced could have been solved through the inclusion of custom components best ordered in large quantities.

In addition to the engineering challenges previously identified, the importance of the systems engineering approach was made clear. While the PARbot team followed the approach loosely, a more critical approach would have made the project specifications and design decisions clearer and more easily supportable. The PARbot team was able gain a deeper understanding of systems engineering as the project progressed, however several large decisions had already been made and committed to. Through analysis of the decisions made, the choices could be supported, but better arguments could have been formed. In future years, a stricter application of the guidelines would lead to a faster, more concise decision making process.

PARbot has also shown features that would be desirable to include in future iterations of the PARbot project or other personal assistants. When speaking with members of the Massachusetts Assisted Living Facilities Association (Mass-ALFA) at their monthly meeting several possible features were mentioned. One of the most agreed upon features was the inclusion of personal calendar for reminding the user of events scheduled for their day as well as reminding the user of the current date. It was suggested that repetition is especially helpful for elderly users. Any way of relaying a call for help or an automated call for help is also ranked as highly desirable. Because of the modular approach to the PARbot project, it is possible to include these features in future version of the project.

Overall, PARbot has demonstrated the capabilities of a modular robotic platform and shown the areas where additional functionality would be the most beneficial. The current functionality includes navigation, mapping, localization, target tracking, driving to the target location and the ability to accept commands through a touch screen interface. Further, through the testing and development cycle, the PARbot team was impressed by the long battery life of the robot. While the expected battery life was only 2 or 3 hours and calculations previously presented suggested even less, under real world testing with intermittent driving, the robot was able to run for 5-6 hours under one charge. This shows that the overall system has excess power available for future modules, an important feature. Alternativly the battery capacity could be reduced to allow for a smaller battery to be placed in the frame, making it lighter and freeing internal space.

Overall, the PARbot project was also a great learning experience with a real engineering challenge to create a marketable product. The Cornell Cup as Presented by Intel helped to emphasize the design process and keep an emphasis on the end user.

6 Future Recommendations

The PABot MQP is designed to be active for 5 years, covering several iterations of the robot and its design. This section will highlight some of the possible future changes that could be implemented to improve the operation or increase the functionality of PARbot.

6.1 Ubuntu 14.04

This is a new version of the Linux based operating system currently used on the robot. This version of Ubuntu offers greater support for high-resolution display. The largest advantage to this system would be the integration of a tablet designed to work specifically with this operating system, and therefore would be compatible with ROS. The current implementation of the touchscreen interface uses an externally mounted Intel Atom Module running Windows 7, 32 bit. The application executes Batch files, which call bash files to make robot commands. While this does work, it is not the most elegant or simple interface and the use of a Linux based table could make the process far simpler [43].

6.2 Win_ROS

Win_ROS is a version of ROS that will run in a Windows 7, 32 bit environment. This would allow for the ROS meta-operating system to be more accessible to individuals unfamiliar with the Ubuntu operating system and broaden the number of components that can interface with the system. Until this time, a stable release of the Win_ROS platform has not been available, but has recently been released. Additionally, the use of a Windows operating system would resolve the complex touch screen interface as well, allowing for smoother operation [44].

6.3 Android

Android operating system will increase the possibilities for interface devices. The use of Android in the PARbot system would allow for integration of other devices, such as an Android tablet or smartphone. This would give the user more ways to customize PARbot to suit their needs. Based on suggestions from the meetings with Mass-ALFA, the inclusion of a personal calendar would be a useful feature that could be easily added though the use of an Android system. ROS supports Android with its ROSJAVA API.

7 References

1. "Age and Sex." - *The Older Population in the United States: 2011*. U.S. Department of Commerce, 28 Nov. 2012. Web. 21 Apr. 2014.

2. Donald Redfoot, Lynn Feinberg, and Ari Houser, "The Aging of the Baby Boom and the Growing Care Gap: A Look at Future Declines in the Availability of Family Caregivers," AARP, Washington DC, Issue 85, August 2013.

3. "Personal Robotics Lab." *Personal Robotics*. Carnegie Mellon, 11 Mar. 2013. Web. 21 Apr. 2014. <https://personalrobotics.ri.cmu.edu/projects.php>.

4. "Partner Robot Family." *Toyota Global Site*. Toyota, n.d. Web. 21 Apr. 2014. <http://www.toyota-global.com/innovation/partner_robot/family_2.html>.

5. "CONCEPT." *RIBA 公式ページ 理研-東海ゴム 人間共存ロボット連携センター*. RIKEN-TRI Collaboration Center for Human-Interactive Robot Research, n.d. Web. 21 Apr. 2014.

6. "Mobile Robot MP-500." *MP-500 - Neobotix*. Neobotix, 21 Apr. 2014. Web. 21 Apr. 2014. <http://www.neobotix-robots.com/mobile-robot-mp-500.html>.

7. "PeopleBot." *Robot Makes Human-Robot Interaction Research Affordable*. Adept, 21 Apr. 2014. Web. 21 Apr. 2014. <http://www.mobilerobots.com/researchrobots/peoplebot.aspx>.

8. "Pioneer P3-DX." *Adept MobileRobots Pioneer 3-DX (P3DX) Differential Drive Robot for Research and Education*. Adept, 21 Apr. 2014. Web. 21 Apr. 2014. <http://www.mobilerobots.com/researchrobots/pioneerp3dx.aspx>.

9. "United States Access Board." *ADAAG*. United States Access Board, 21 Apr. 2014. Web. 21 Apr. 2014. <http://www.access-board.gov/guidelines-and-standards/buildings-and-sites/about-the-ada-standards/background/adaag#purpose>.

10. "Which Wheelchair Vans Are Right For Me?" *Choosing a Wheelchair Van, Handicap Van, or Accessible Vehicle*. BraunAbility, 21 Apr. 2014. Web. 21 Apr. 2014. <http://www.braunability.com/which-minivan.cfm>.

11. Linden, David; Reddy, Thomas B., ed. (2002). *Handbook Of Batteries* (3rd ed.). New York: McGraw-Hill. p. 23.5. ISBN 0-07-135978-8.

12. "What's the Best Battery?". Battery University. November 2006. Retrieved 10 September 2011.

13. GMB Liion Battery Individual Data Sheets. 2014. Guangnzhou Markyn Battery Co., Ltd. 21 April. 2014. <http://www.powerstream.com/lip/GMB052030.pdf>.

14. "DC Motor." *Wikipedia*. Wikimedia Foundation, 18 Apr. 2014. Web. 21 Apr. 2014. <http://en.wikipedia.org/wiki/DC_motor>.

15. "Brushed DC Electric Motor." *Wikipedia*. Wikimedia Foundation, 04 Nov. 2014. Web. 21 Apr. 2014. <http://en.wikipedia.org/wiki/Brushed_DC_electric_motor>.

16. "Brushless DC Motor." *Wikipedia*. Wikimedia Foundation, 04 Feb. 2014. Web. 21 Apr. 2014. <http://en.wikipedia.org/wiki/Brushless_DC_motor>.

17. "Stepper Motor." *Wikipedia*. Wikimedia Foundation, 21 Apr. 2014. Web. 21 Apr. 2014. <http://en.wikipedia.org/wiki/Stepper_motor>.

18. "Servo Motor." *Wikipedia*. Wikimedia Foundation, 20 Apr. 2014. Web. 21 Apr. 2014. <http://en.wikipedia.org/wiki/Servo_motor>.

19. "Accelerometer." *Wikipedia*. Wikimedia Foundation, 18 Apr. 2014. Web. 21 Apr. 2014. <http://en.wikipedia.org/wiki/Accelerometer>.

20. "Inertial Measurement Unit." *Wikipedia*. Wikimedia Foundation, 15 Apr. 2014. Web. 21 Apr. 2014. <http://en.wikipedia.org/wiki/Inertial_measurement_unit>.

21. "Gyroscope." *Wikipedia*. Wikimedia Foundation, 21 Apr. 2014. Web. 21 Apr. 2014. <http://en.wikipedia.org/wiki/Gyroscope>.

22. "Ultrasonic Sensor." *Wikipedia*. Wikimedia Foundation, 21 Apr. 2014. Web. 21 Apr. 2014. <http://en.wikipedia.org/wiki/Ultrasonic_sensor>.

23. "Robot App Store." *Robot App Store*. Robot App Store, 21 Apr. 2014. Web. 21 Apr. 2014. <http://www.robotappstore.com/Robopedia/Ultrasonic%2520Sensor>.

24. "Robot App Store." *Robot App Store*. Robot App Store, 21 Apr. 2014. Web. 21 Apr. 2014. <http://www.robotappstore.com/Robopedia/Cliff-Sensor>.

25. "Rotary Encoder." *Wikipedia*. Wikimedia Foundation, 19 Apr. 2014. Web. 21 Apr. 2014. <http://en.wikipedia.org/wiki/Rotary_encoder>.

26. "Robot App Store." *Robot App Store*. Robot App Store, 21 Apr. 2014. Web. 21 Apr. 2014. <https://www.robotappstore.com/Robopedia/Microphone>.

27. "Robot App Store." *Robot App Store*. Robot App Store, 21 Apr. 2014. Web. 21 Apr. 2014. <https://www.robotappstore.com/Robopedia/Speaker>

28. John MacCormick, John. "How Does the Kinect Work?" How Does the Kinect Work? 21 Mar. 2014. Web. 21 Mar. 2014. <http://users.dickinson.edu/~jmac/selected-talks/kinect.pdf>.

29. "Wiki." *Documentation*. Open Source Robotics Foundation, 17 Dec. 2013. Web. 21 Apr. 2014. <http://wiki.ros.org/>.

30. "Wiki." *Groovy*. Open Source Robotics Foundation, 17 Dec. 2013. Web. 21 Apr. 2014. <http://wiki.ros.org/groovy>.

31. "Wiki." *Hydro*. Open Source Robotics Foundation, 17 Dec. 2013. Web. 21 Apr. 2014. <http://wiki.ros.org/hydro>.

32. Rekleitsis, Ioannis. *A Particle Filter Tutorial For Mobile Robot Localization*. Centre For Intelligent Machines, 21 Apr. 2014. Web. 21 Apr. 2014. <http://www.cim.mcgill.ca/~yiannis/particletutorial.pdf>.

33. Welch, Greg, and Gary Bishop. *An Introduction to the Kalman Filter*. University of North Carolina at Chapel Hill, 24 July 2006. Web. 20 Mar. 2014. <http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf>.

34. Kuipers, Benjamin. "Occupancy Grids." University of Texas at Austin, Austin. Web. 04 Mar. 2014. <http://www.cs.utexas.edu/~kuipers/slides/L13-occupancy-grids.pdf>.

35. Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. p.590

36. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 22.3: Depth-first search, pp. 540–549.

37. Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths".*IEEE Transactions on Systems Science and Cybernetics SSC4* **4** (2): 100–107. doi:10.1109/TSSC.1968.300136

38. "United States Access Board." *ADAAG - ADA Accessibility Guidlines*. U.S. Government, Sept. 2002. Web. 21 Apr. 2014. <http://www.access-board.gov/guidelines-and-standards/buildings-and-sites/about-the-ada-standards/background/adaag#purpose>.

39. *Sparkfun.* (2013). Retrieved from Sparkfun : https://www.sparkfun.com/products/10736

40. *Taico Lead Acid Battery 24V 20AH* (2013): Retrieved from Alibaba: http://www.alibaba.com/product-gs/970365407/Taico_lead_acid_battery_24V_20AH.html?s=p

41. Von Hundelshausen, Felix, Michael Himmelsbach, Falk Hecker, Andre Mueller, and Hans-Joachim Wuensche. "Driving with Tentacles: Integral Structures for Sensing and Motion." *Journal Of Field Robotics* (2008): n. pag. Web. 10 Mar. 2014. <www.interscience.wiley.com>. Novotny, Filip.

42. "Wiki." *Visp_auto_tracker.* N.p., n.d. Web. 28 Apr. 2014.

43. "Ubuntu 14.04 Desktop: Trusted OS for Consumers and Business." *Ubuntu Insights*. N.p., 16 Apr. 2014. Web. 29 Apr. 2014.

44. Stonier, Daniel, and Yeong-Il Choe. "Win_ros." *ROS.org*. N.p., n.d. Web. 29 Apr. 2014.

45. QR Code. Digital image. Barcodes India. N.p., n.d. Web. 30 Apr. 2014. <http://www.barcode1.in/wp-content/uploads/QRcodeINDIA.gif>.

46. "Intel Galileo Development Board." *Intel Galileo.* Mouser, n.d. Web. 30 Apr. 2014. <http://www.mouser.com/new/Intel/intel-galileo-development-board/>.

47. "About." *Cornell Cup USA as Presented by Intel.* N.p., n.d. Web. 30 Apr. 2014. <http://www.systemseng.cornell.edu/engineering2/se/intel/about/>.

48. S. Thrun. Simultaneous localization and mapping. In M.E. Jefferies and W.-K. Yeap, editors, *Spatial Mapping Approaches in Robotic and Natural Mapping Systems.* Springer Tracts in Advanced Robotics, Berlin, 2006.

8 Appendix A: Personas

The following are the personas developed by the group to form a concept for potential users of the PARbot project. These were used to help define use cases for the robot, determine stakeholders in the project, as well as define the operating environment.

8.1 Persona 1:

name: Jim Henderson

age:71

Jim is a retired electrical engineer with severe rheumatoid arthritis. His arthritis limits his mobility but allows him to perform most daily tasks. He can drive himself around, but walking long distances and caring items can be very tiring and painful. Because of these limitations tasks such as moving things around his home, shopping, or getting things from his car can be very difficult. Jim uses a cane to help get around, but this can make tasks more difficult because it takes a hand to use. Due to his restricted income from retirement he cannot afford to live in an assisted care facility or have an aide come and help him. Having a relatively low cost solution that could be available on demand would help improve his day to day life.

Because of these limitations tasks such as moving things around his home, shopping, or getting things from his car can be very difficult. Jim uses a cane to help get around, but this can make tasks more difficult because it takes a hand to use. Even with the difficulties his illnesses cause him, Jim is determined to live at home.

8.2 Persona 2:

John

Age-75

John is 75 year old retired army personnel who besides his disability due to the age factor had a significant mobility hindering accident while on a army mission. He was shot in the back, and the doctors decided that he needed to have a few vertebrae fused. Although John can help himself to some of the daily tasks, he needs help with others. His wife has passed away and he doesn't see his children quite often either. From time to time he needs help with going to the grocery stores and needs someone to carry the heavy grocery bags for him. He is a real fanatic about gardening and wants to go the nursery pretty often to get new plants for his lawn but is limited due to his abilities and could certainly use a strong hand to carry all the plants and the heavy compost for his garden.

John is looking for or rather has a need for an intelligent electro-mechanical device that he could easily control and carry it around with him as his personal assistance machine to do all the lifting for him and any other features associated with the device that might be easy to use, give him a peace of mind and make his life better.

Besides that John could also use some help carrying trays containing food from the kitchen incase his old buddies from the army decide to come over and he could show off his new personal assistive robot.

8.3 Persona 3:

Mary Shelly

Age: 30

Injury/Disability: Partially paralyzed in a car crash

Mary was in a serious car crash last year that resulted in partial paralysis of the left side of her body. She is still able to move her arm and leg, however, she lost some of her range of motion and strength on that side of her body. Mary is able to travel short distances by walker though she mostly uses a wheelchair. She is able to carry light objects in her left hand though her grip is weak. Her right side functions fully with no loss of strength or mobility.

Mary works as a high school teacher and is able to navigate the school as it meets ADA requirements. She has trouble in the cafeteria and other high traffic areas and can't carry a lunch tray on her own. The students and staff are very helpful and understanding.

She misses her independence and hates that she needs help from family and friends for basic tasks like shopping. A personal robot would help her to carry items when out and about as well as in the grocery store. The robot would allow her to gain back some of her independence and freedom. The robot would also prove to be helpful in her home to help with general tasks such as carrying dinner from her kitchen to her dining room; a task which normally takes multiple trips.

8.4 Persona 4:

Paul: Age 40

Paul is a veteran who has served the past 18 years of his life in the army. After graduating college with a degree in mechanical engineering, he enlisted and was a part of several deployments. Near the end of his last tour of Iraq, he was shot in the leg which resulted in the amputation of the lower 30% of his left leg. Paul returned home and works in an office building but must move about with crutches or a wheelchair. He is able to carry out many everyday tasks such as cooking (his upper body was unharmed) and driving (with his other leg still intact) but tasks that require moving objects are difficult. When shopping he is limited by what he can carry in his lap while in the wheelchair and going to buffets require someone to carry the tray for him to prevent spills. Paul sees no need to move to an assisted living facility, but also does not want to become a nuisance to his family and friend who help him do his weekly shopping. Most importantly, Paul wants to maintain his independence.

As a result of his injury carrying items is difficult for Paul, but he still has a good deal of upper body strength from his time in the army. A personal assistant robot could be put to good use by Paul. When Paul needs to go shopping, the robot could be easily lifted and placed in the trunk of his car and then removed in when he arrives at the shopping center. The robot would attach to a shopping cart and follow Paul as he made his way around the store, placing items in the cart. When finished the groceries and the robot could be placed back in the trunk by Paul who could then drive home. Such robot could allow Paul to maintain his quality of life without another person needing to help him. Additionally this one time investment would keep Paul independent rather than have to pay for assistive care a few times a week.

8.5 Persona 5:

Name: Bethany Cerulean

Age: 32

Occupation:

Disability: Bethany is a veteran who was injured in a roadside bomb, which caused her to lose both her legs and become legally blind.

Bethany had a hard time coming to terms with not having the use of both her legs. She was very active and loved going for weekend hikes. She has been getting around using a wheelchair in her home and she has a set of prosthetic legs for going outside. In her home she can get around quite well. Because she knows the layout and her blindness really doesn't hurt her at home. But when she goes to the store it is hard for her to push the cart and pick things up because of the limited mobility of her prosthesis and her poor vision. By having the robot take care of the cart she would not have to worry about running into someone with the cart, Or just generally causing problems with it. With the future prospects of having the robot able to take things off the shelves it would make the entire process even easier.

Ways the robot could potentially help at home:

If the robot could go to the door and provide a video feed to a smartphone could make it easier for her to not have to move her wheelchair and provides the data to her immediately.

8.6 Persona 6:

Name-Mike Jones

Age-80

Mike Jones is an 80 year old retiree who has been trying to live independently for the last 15 years although his family considered having another person live with him to assist him. He has recently had a problem when he goes to stores where he has not been able to carry as much as he used to be able to. He has also lost some ability to push shopping carts with the loss of his upper body strength. He has been losing his upper body strength over the last few years as well as developing problems with walking and recently used either a walking cane or a walker. The main problem with Mike's health is from his old age and a small case of arthritis, giving him joint pain, trouble moving his arms and a loss of upper body strength. He is still able to drive as he is able to use his legs for this, however he encounters a problem when he attempts to shop for the food and other necessities he needs. This problem has been compounded by his recent reliance on his walker and cane, which removes one of his hands from being able to carry bags. A personal robot would be very helpful for Mike as it would be able to move a shopping cart around and follow Mike giving him a mobile shopping cart that he would not have to push. This would greatly increase his ability to shop for himself giving him the ability to live as independently as he would like to. This would ease the strain on his family and greatly improve his quality of life.

9 Appendix B: Development on PARbot

9.1 Connecting to PARbot Over the WiFi Bridge:

PARbot's onboard computer uses the static IP address 192.168.1.100. To connect to the onboard computer SSH is used. The group created a script called parbot_login.sh; this script calls the command *ssh –X parbot@192.168.1.100* which allows command line access to the onboard computer with X-forwarding. X-forwarding allows graphics to be sent over the network, meaning that programs like gedit or nautilus can be run on the robot but viewed on another computer. However, certain types of rendered views, such as Rviz, do not work with X-forwarding. To make this process easier a WiFi Bridge was setup.

A WiFi bridge consists of two or more wireless routers are connected to one another using the WiFi network in a manner that acts as an extension between them. In the case of PARbot, the onboard router is the host and the TP-LINK router is the client. When a device is connected to the TP-LINK router's Ethernet connection, it operates as if it were connected to the Ethernet ports of the onboard router. This link runs over the PARbot5G network on the 5GHz band, and is limited to 120Mb/s due to interference although this may be higher in locations other than WPI. The 5GHz band was selected for it lower traffic and stronger signal strength in the testing environment.

9.2 Moving files to and from PARbot

Although Secure Shell (SSH) has many features, file transferring is not one of them. To transfer files to and from PARbot, SFTP (SSH File Transfer Protocol) is used. This protocol is based on SSH but is specifically designed to handle file transfer. A script called sftp_login.sh was created to run the command *sftp parbot@192.168.1.100*. Once connected via SFTP the standard UNIX file commands still function. SFTP also has its own set of commands that are important to understand. The first of which is *get.* This command takes files from the system that has been logged into, in this case, PARbot, and sends it to the folder that SFTP was called from. The other SFTP command is *put.* The *put* command is essentially the opposite. It is important to note that *get* uses the local file path while *put* uses an absolute file path. Both commands can be called with the *–R* option which allows folders and their contents to be transferred recursively.