SACRAMENTO STATE

April 29, 2018

DEPLOYABLE PROTOTYPE DOCUMENTATION

Automated Smart Wheelchair

Team Member

Quan Au
Benson Co
Timothy Mulvey
Kenneth Nguyen

Supervisor

Dr. Fethi Belkhouche
California State University of Sacramento

Abstract

This document is intended to be summarization our work process, in Fall 2017 and Spring 2018 semesters, in term of technical specifications of our project, Automated Smart Wheelchair for people with disabilities. The paper starts by explaining the societal problem and followed with the design idea as a proposed solution. In addition, we highlight the funding that was required to make this project possible. We also discuss the project milestones and their significance, as well as the work breakdown structure, which includes the risk assessment and mitigation plans, and task assignments for each team member. Included, we have a documentation for our design, which includes the hardware/software block diagram and schematics. We have also included any mechanical drawings that were used in our project. Additionally, we included our test plan along with the test result.

Keyword Index

Automated Smart wheelchair, Raspberry Pi, Max32, Arduino, automation, 2D mapping, collision avoidance, ultrasonic sensor, deployable prototype report, deliverables, budget, schedule.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

I. EXECUTIVE SUMMARY

Our senior design group designed and implemented an automated wheelchair that could navigate itself inside a building. The main components that allow the wheelchair to do self-driving are 2D mapping and localization through laser scan. By knowing its current location, the wheelchair will be able to create a loop of navigation commands and allow the wheelchair to move to the desired location.

II. INTRODUCTION

Our group was tasked with creating a solution for a certain societal problem. We chose a societal problem that we all feel would benefit society very much. Our societal problem focused with people with certain disabilities, specifically with motor control. When people lack the use of their legs and/or arms, this creates a great inconvenience in their daily lives. The most common solution to people who cannot use their legs and/or arms would be a wheelchair being operated by either the user themselves or by a caregiver. However, we want to create a solution where the user of the wheelchair can be as independent as possible. Our proposed solution is to have a wheelchair where it can autonomously navigate to various destinations inside a building without any assistance from the user or caregiver. This paper will highlight the fundamentals of our project and our progress along the way.

III. SOCIETAL PROBLEM

A. Problem Statement

The restrictive of people with disabilities' traveling distance has been one of the most problematic issue that people have been facing. We can see that many disabled people are using manual powered wheelchair that gives them a tough time moving to different locations. Many cases of disabled people need assistance from another person to move around, especially those with certain medical conditions such as spinal injuries that would paralyze the body from the shoulder down, or people with ALS. In these cases, even a regular motorized wheelchair would not be able to assist them in travelling from one place to another. Therefore, we decided to address this problem to find a solution that would help people with disability to move around more effectively.

B. Demand

There are many different medical conditions which leads to different types of disabilities and therefore require different needs from a wheelchair. Some of the most common medical conditions that would require a person to have a special design chairs are Lou Gehrig's disease, stroke, spinal cord injuries, multiple sclerosis, cerebral palsy, etc. Lou Gehrig's disease is a disease that attacks the motor neuron of the body. The disease will gradually deteriorate motor neurons which will cause the brain to lose its ability to control all voluntary movements of the body. Symptoms of the disease would be the inability to speak, eat, move and breath. It usually attacks people between the ages of 55 and 75 [1]. There are approximately 5,600 people in the U.S diagnosed with ALS each year. It is estimated that as many as 30,000 Americans may have the disease at any given time [2]. The leading cause of paralysis is caused by stroke. The frequency of a stroke happening with someone in the U.S is about a stroke every 40 seconds. Every year, more than 795,000 people in the U.S will have a stroke. Only about 2/3 of these individuals survive and will have long life disabilities. Strokes are a leading cause of long term disabilities because it reduces the mobility in half of stroke survivors of age 65 and up [3]. According to Figure 1, out of all the survivors of stroke, only 10% will completely recover from stroke, and the other 90% will be living with some form of disabilities for the rest of their lives.

The next leading cause of body paralysis is physical injuries such as spinal cord injury and traumatic brain injury. In the U.S, there are 282,000 individuals with spinal cord injuries in the year of 2016. With a population of 314 million people, the approximation of spinal cord injury is 54 cases per million population. There are nearly 17,000 new spinal cord injury cases each year. There are less than 1% of people who experience complete recovery from spinal cord injury. Since 2010, 45% people who have spinal cord injury experience incomplete tetraplegia, 21.3% incomplete paraplegia, 20% complete paraplegia, 13.3% complete tetraplegia and only 0.4% is completely recovered [4]. Similarly, traumatic brain injuries also leave a person with disabilities. There are about 235,000 cases of traumatic brain injury each year and about 80,000 to 90,000 people will experience long term disabilities every year [6].

Lastly, there are many other causes that can lead to body paralysis such as neurofibromatosis, cerebral palsy and post-polio syndrome. With these causes combined, the total U.S population that suffers

from paralysis is more than 5 million people, representing about 1.9 percent of the entire population [7].

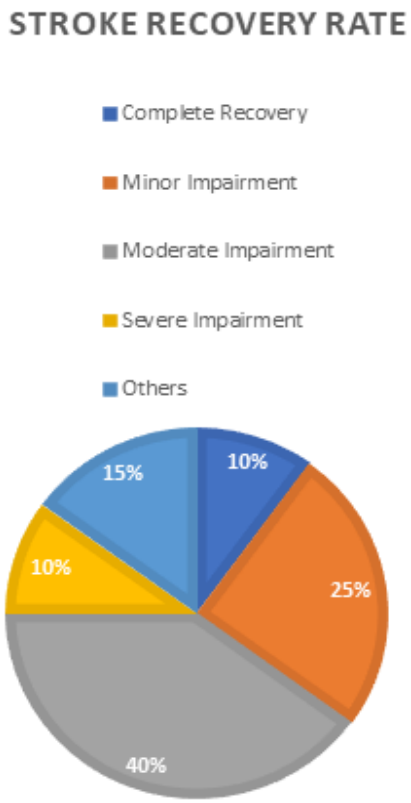


Figure 1. Percentage of recovery in stroke patients [5]

C. Shortcoming of Traditional Wheelchair Design

The traditional wheelchair design is just a chair on wheels in which requires the user to use their arms and push the wheels for it to be able to move. The thing about manual wheelchairs is that the user ultimately becomes the limiting factor when it comes to where he or she can go. This can vary from not enough strength to go up a hill or lacking the experience to make rapid left and right turns when needed. This is a huge problem in general because many people would like to continue their lives prior an accident or an unfortunate medical condition. These limitations can be overcome by practice, however there are users that, no matter the amount of time spent practicing, still will not be able to operate a wheelchair efficiently. These may be people with more serious conditions, such as ALS or an impairment which greatly affects the user's ability to use a motorized vehicle. In these cases, a basic wheelchair will not be the answer to their needs.

A more advanced design of wheelchair is the powered wheelchairs with motorized wheel that could run using batteries and controlled by a joystick. Powered wheelchairs are also prone to user error. An example of this would be when someone becomes unconscious while continuing to accelerate their wheelchair towards a crowd of people. The wheelchair has no ability to sense that it is headed towards a crowd and will eventually crash into the crowd. Not only does this put others in danger, but it also puts the user in great danger as well. This scenario can be easily avoided with a collision avoidance system. Also, users with diseases like cerebral palsy will not be able to use the traditional motorized wheelchairs.

IV. DESIGN IDEA CONTRACT

Our main idea to help with the societal problem that we discussed earlier is to design and build an Automated Wheelchair for people with Disabilities. The wheelchair will have three main features that are mapping, autonomous navigation, and collision avoidance. Speaking about the feature, mapping is a process of creating a virtual 2D map of surrounding. Using the Kinect camera, we convert the distance information into a 2D laser scan to create the map. Next, using the information taken from the mapping feature, the wheelchair will be able to navigate itself, using ROS (Robot Operating System), to a designated location. Finally, using the ultrasonic sensors, our wheelchair will be able to detect obstacles that are in its way and stop before any collision happen to avoid injuries or damage to our user and product.

Besides the main features that we already mentioned, our wheelchair is also have a battery management system to ensure that the battery does not become discharged, to avoid damage to the battery. It provides an LCD display of the battery's stats, which including a battery voltage reading, battery percentage, and estimated runtime remaining. In addition, the wheelchair also has a charging unit that allow it to charge the battery at a regulated rate and prevent overcharging/overheating. Finally, the battery will maintain the fully charged voltage until the user unplugs the power.

In the Fall semester, we focus on designing, putting the hardware together, and setting up ROS, as ROS is the main feature of our wheelchair, which differentiate ours to other product on the market. Before we came up with the idea of using ROS to control the wheelchair, our original idea was to make our wheelchair to follow lines to a designated location; however, after further research and make the wheelchair follow the lines, we agreed not to use the line following because it is not practical since it would take to. much effort and time for coding. During one of our early team meetings, we reached out to Dr. Jesse Leaman from University of Nevada, Reno, Department of Computer Science and Engineering. He is well known for his research interests on robotics and smart wheelchair for people with disability. Dr. Jesse recommended us to use ROS to make our wheelchair fully autonomous and that is why we finally us ROS as our main feature for our wheelchair.

In the Spring semester, we focused on testing the wheelchair on many different parts including each individual component as well as integrated components of the wheelchair. We also focused on improving the performance of ROS. In the fall semester, we successfully control the wheelchair remotely through Wi-Fi connection with ROS. In the fall semester, we worked on mapping and navigation of the wheelchair. The wheelchair is agreed to be able to navigate itself inside a building from one location to another location with a reasonable of obstacles.

FEATURE SET	
FEATURE	**DESCRIPTION**
Mapping	The wheelchair will be able to create a virtual 2D map of its environment using the Kinect camera. We are converting the Kinect distance information into a 2d laser scan to create the map.
Collision Avoidance	The wheelchair will stop if it detects a collision on its course using ultrasonic sensors. Using an Arduino Uno to detect obstacles. Once an obstacle is detected an interrupt signal is sent to our ChipKit MAX32 to stop motor control until the obstacle is removed.
Autonomous Navigation	Once the 2D map is created we will be using the ROS navigation stack to control the robot. The navigation stack sends a Twist message which consists of a linear velocity vector in meters per second and an angular velocity vector in radians per second.
Battery Management System	To ensure that the battery does not become discharged, to avoid damage to the battery. It provides an LCD display of the battery's stats, which including a battery voltage reading, battery percentage, and estimated runtime remaining. In addition, the wheelchair also has a charging unit that allow it to charge the battery at a regulated rate and prevent overcharging/overheating.

Table 1- Feature Set of the Wheelchair

BUDGET SUMMARY			
Part	**Quantity**	**Budget Cost**	**Actual Cost**
LEDs	10	$2.00	$7.00
Arduino Nano	1	$3.00	$3.00
Ambient Light Sensor	1	$7.00	$7.00
Ultrasonic Sensor	3	$10.00	$10.00
OLED Display	1	$10.00	$6.74
Kinect Adapter	2	$10.00	$8.99
Converter Step Down Regulator	1	$10.00	$8.99
Speaker	1	$15.00	$1.00
Micro SD	1	$15.00	$15.11
Wheelchair charger	1	$20.00	$21.19
PCB	1	$24.00	Do not need
Kangaroo	1	$30.00	$26.99
Raspberry Pi Board	1	$40.00	$40.00
Arduino Mega 2560	1	$40.00	$40.00
Encoders	2	$60.00	$55.89
Max32	1	$65.00	$65.00
Screen Monitor	1	$70.00	Do not need
Laptop mount	1	$70.00	$69.99
12 Volts Batteries	2	$80.00	$80.00
Kinect Camera	1	$85.00	$25.00
Motor controller	1	$150.00	$128.99
Laptop	1	$300.00	Donation
Base wheelchair	1	$450.00	Donation
Wires	2	Did not plan	$9.00
USB to TITL Serial Cables	1	Did not plan	$12.39
Emergency switch	1	Did not plan	$2.00
Laptop battery	1	Did not plan	$15.99
Other accessories	N/A	Did not plan	$50.00
	Total	**$1,566.00**	**$710.26**

Table 2. Budget Summary

V. BUDGET SUMMARY PROJECT FUNDING

This section discusses the costs associated with out project. All components were purchased within the group without any outside resources. The cost for our project is relatively high, this is due to the nature of projects requiring a lot of research and design. We test out components and then realize that it will not work, or we find something better. By the time we find a good combination of parts, we have already gone through a decent number of sensors or controllers to get there. There are also parts that are very delicate, such as the motor controller, which in our case, does not have reverse polarity protection. Components that lack safety or the robustness are considered high risk and we take extra care to make sure that we do not damage them during our testing. There is also the problem of purchasing parts last minute. A lot of the parts we purchased could've been purchased at wholesale pricing if we had additional time. However, due to time constraints, we had to purchase them with priority shipping which also raised the price. Luckily, we were all able to contribute to the cost of all components, making it much more manageable in the end when it came to finances.

The budgeted cost for the ASW project was set at $1,500. Table 2 is a broken-down cost of the financial plan in comparison with the actual expense for this project. Given the Budget Summary, our team was not only successful in meeting all its objectives and deliverables, but we also in progress of completing the project under budget. In addition, we have saved a lot money from donation. This allows us to spend our budget on other important components or tools that we did not plan at the initiation phrase.

VI. PROJECT MILESTONES

This section will cover the milestones that were crucial in the completion of our project. There were many smaller feats that we accomplished but we will list the more important ones.

A. Motor Controller

The motor controller is basically the heart of our project. Without being able to control the motors we would not be able to test any of the other features that we have. Initially, we did not have a motor controller. We were trying to reverse engineer the motor controller that came with the wheelchair and it was much more difficult than we had anticipated. On top of that, it only took in analog input which means that we would not be able to control it digitally with a micro controller. Therefore, we knew that we needed to find a motor controller that would be compatible with micro controllers as this would make programming much easier later on. We ended up choosing the SaberTooth 2x32A motor controller. This is one of SaberTooth's most advanced motor controller, which gives the user the option of analog input, serial, USB, or R/C. In our case, we will be using the serial input option.

When we got the motor controller, it was not without its issues. The motor controller needed to be configured and calibrated to an extent. We also did not know that we had to set the DIP pins to change the mode that it's in, which wasted a lot of time since the instructions were not very clear initially. However, once we were able to configure and calibrate the motor controller, we were able to basically control it however we wanted. Movements such as forward, backwards, left, right, and even forward while turning right at the same time was possible. Speed control was also not an issue once we had movement down. The motor controller prior to being installed is shown in the Figure 2.

In figure 3, we are shown a picture of the motor controller being configured and calibrated with DEScribe (program to configure motor controller). This process is relatively risky as during configuration it basically writes to the EPROM of the motor controller. If there is a power failure during this stage, it will render the unit useless, which would require us to acquire a replacement.

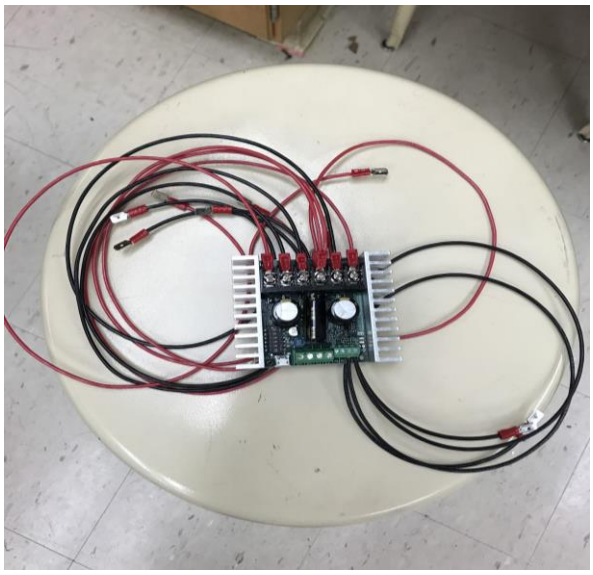


Figure 2. Motor Controller

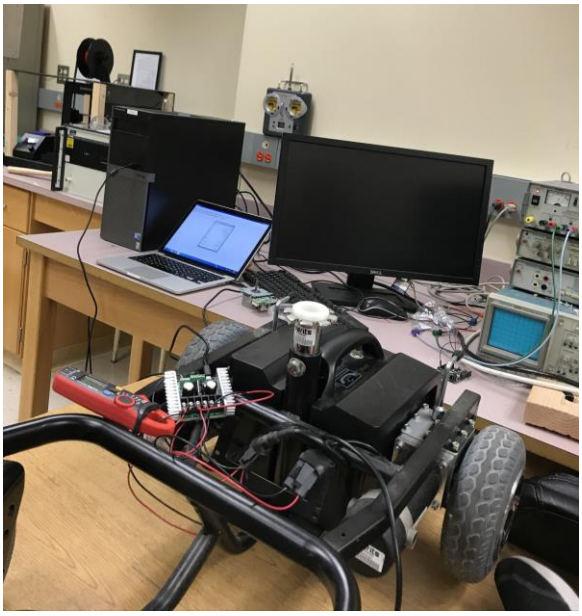


Figure 3. Configuration and Calibration of Motor Controller

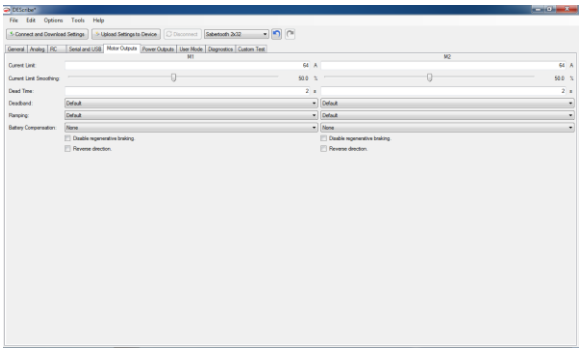


Figure 4. DEScribe Program

In the figure 4, we are shown a screenshot of the DEScribe program. This particular tab was for adjusting the current associated to each motor channel, which lets us set the upper limit to how much current is allowed for the motors. This is useful to prevent the motor driver from being overdriven as that could cause it to overheat. In the figure 5, we are shown a closer look at the DEScribe software.

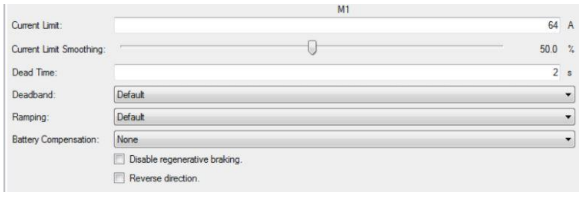


Figure 5. DEScribe Program Settings

B. *Encoders and Odometry*

Since we are using the 2D mapping incorporate with ROS, odometry is a major part of the project. In other words, the wheelchair needs to know how far it had traveled or else it would not be able to put out a virtual map with accurate information about the distances. For the encoder, we are using the modular incremental encoder on each wheel. Because of the lack of tools and resources, we are not able to secure the encoders right on the wheel. Therefore, we must create an angle bracket to hold the encoder in place on the outside of the wheel. In We also attach a rod on the center of the wheel and at the end of the rod is attached to the encoders. So, every time the wheel moves, the rod will also move, and the encoder will take information of the wheel from the movement of the rod. An illustration is shown in Figure 8 below. There sure are better ways of mounting the encoders, but because of time and resource constraints, this is the best solution to mount the encoder that we could come up with for the prototype of the wheelchair. Later in the fall semester, we changed the designed encoder attachments to something more secure and give more accurate information of the wheelchair's movement. We came up with the idea of 3D print a plate with a small rod in the middle, that way we can be certain that the rod is right in the middle of the wheelchair. We also made the angle bracket more secure using metal angle brackets to attach the pieces of wood together. An illustration of the improved system can be seen in Figure 9.

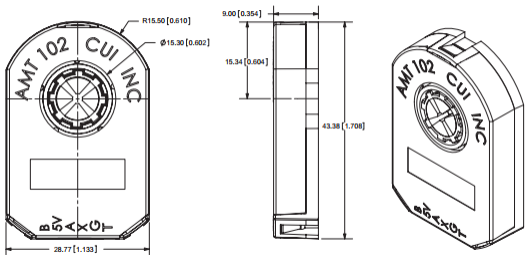


Figure 6. AMT 102 encoder [9]

After we tested and made sure that the encoders worked as we wanted, we needed to make the wheelchair somewhat go straight. We added a motion controller chip for the wheelchair. The motion controller that we used for this wheelchair is the Kangaroo Motion Controller. A picture of the Kangaroo is shown in Figure x. The way we mount the Kangaroo x2 is we directly hook it up to the Sabertooth motor controller. The serial input from

the Max32 will go straight into the Kangaroo and also receive feedback from the Kangaroo. After, the Kangaroo will send commands to the Sabertooth using serial communication. An illustration of the Kangaroo and how to mount it is shown below in Figure 7.

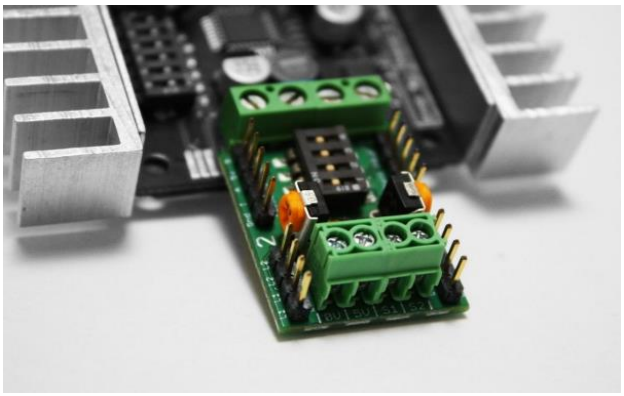


Figure 7. Kangaroo Motion controller [8]

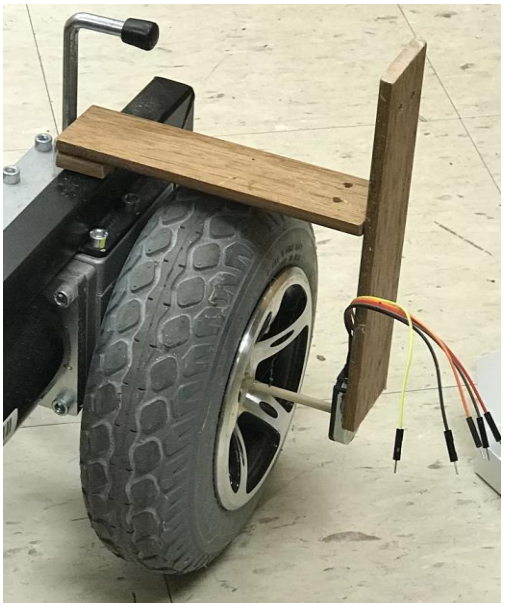


Figure 8. Mounting encoder onto wheelchair

Figure 9. New encoder mount

As shown in Figure 10, the two encoders will be connected directly onto the Kangaroo motion controller for information collection and feedback. At the end of the Kangaroo is the serial connection that is required to connect to the Max32 to control the wheelchair. There are two lines of communication which are called serial 1 and serial 2 on the motion controller. The serial 1 line connected to the transmit pin Tx on the Max32. The serial 2 is connected to the receive pin Rx on the Max32. The serial 1 will be taking commands from the Max32 and the serial 2 will be sending feedback to the Max32. The DIP switch on the Kangaroo is to select between different controlling modes of the Kangaroo such as mixed/independent mode, analog/digital input, analog/quadrature feedback, velocity/position control. The two potentiometers on the side of the motion controller is for controlling maximum and minimum speed of the wheelchair. If it is put into position control mode, the potentiometer will determine the maximum and minimum speed if no input of speed is present. After odometry is done, we can start working on the 2D mapping.

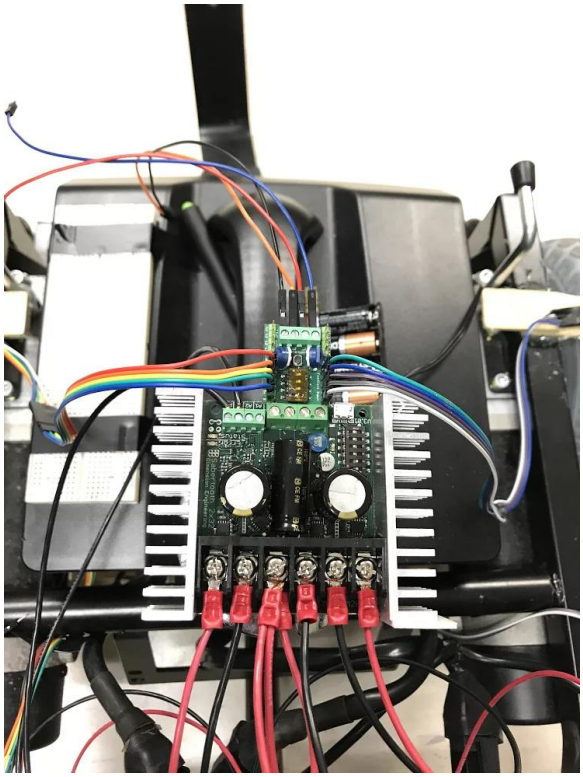


Figure 10. Attach the Kangaroo onto Sabertooth

C. *ROS*

ROS was a major challenge as none of us had any experience with it. Getting ROS installed, and running was the first major milestone for software part. It took a long time to reach the next milestone as there was a very large learning curve with ROS. A lot of time was spent doing tutorials and becoming familiar with ROS before more progress could be made.

There were many milestones along the way as we tried to get all the different parts of the mapping system to work. We started with sending basic custom messages through ROS that could be used to turn the motors on and off. Eventually we were able to use ROS twist messages to accurately control both linear and angular velocity. The twist messages are what will be used by the ROS navigation stack to control movement once we have that setup. Being able to use twist messages to control movement was a big milestone. Publishing odometry, depth information from the Kinect, and all the other needed information was also a big milestone.

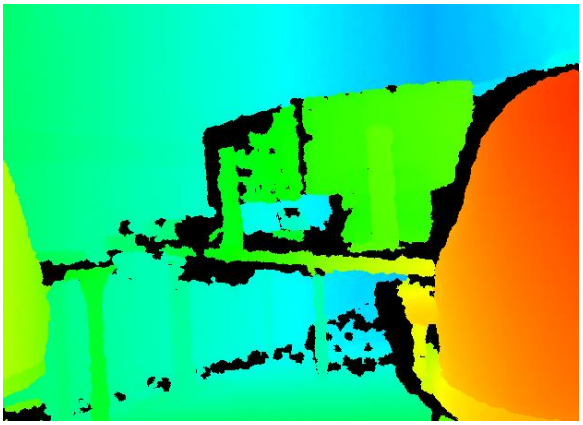


Figure 11. Kinect Point Cloud

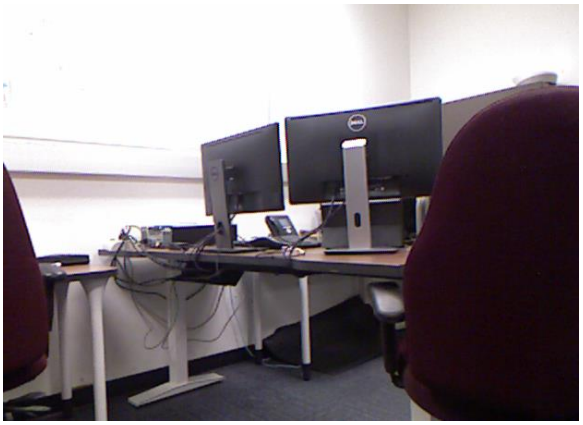


Figure 12. Kinect Picture

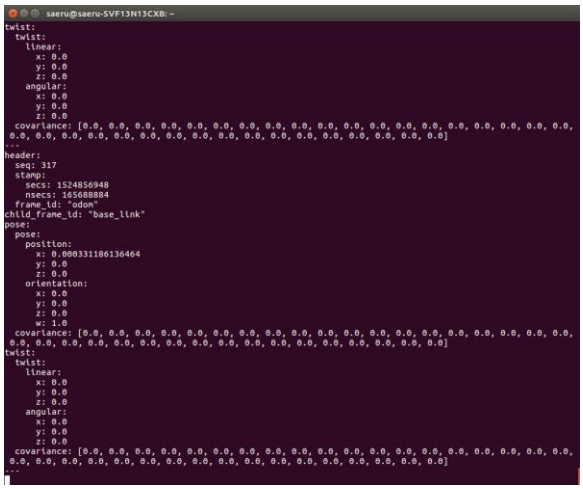


Figure 13. ROS Odometry

Connecting the laptop for the user interface and for monitoring the ROS system was the next milestone. We connected the laptop the raspberry Pi with an ethernet cable directly. This required a custom network setup using static IP addresses. ROS also had to be configured to use IP addresses instead of host names. The laptop was also configured to use the remote ROS core of the raspberry pi.

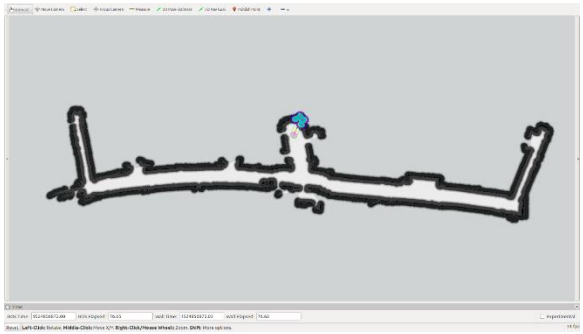


Figure 14. RVIZ user interface

After the odometry and sensor information was working we had to get all the transforms and configurations tuned for the mapping process. This included a transform for from the base of the robot to the location of the Kinect. The point cloud coming from the connect also needed to be converted to a laser scan. The next major milestone was producing the map. This also required tuning the odometry information to make it more accurate.

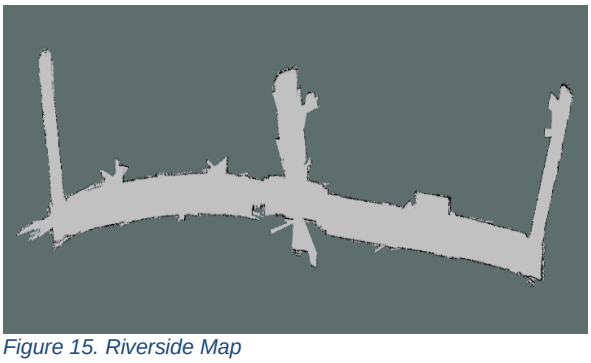


Figure 15. Riverside Map

The navigation stack was the last major milestone for ROS. This was probably one of the hardest because we keep running into limits with the hardware we were using. First the navigation stack would not even run on the raspberry pi. We had to use the laptop which was originally just for the user interface to run the navigation stack. Next the three devices we overloading their connections. We had to increase the serial speeds used by the Arduino, motor controller, and the raspberry pi. The ethernet connection was still overloaded because of all the visual information being sent from the Kinect to the navigation stack. The Kinect was then moved directly to the laptop, but this overloaded the CPU of the laptop. Next a personal laptop was used. We had to remove Windows and install Linux. Finally, after more configuration the navigation stack finally worked.

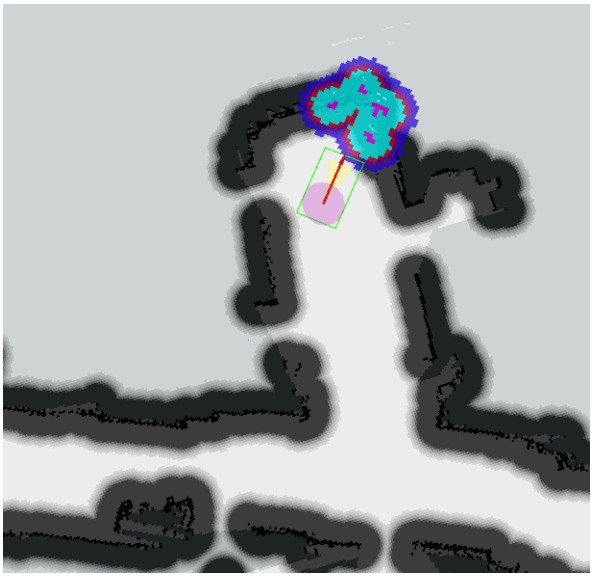


Figure 16. ROS Navigation

VII. WORK BREAKDOWN STRUCTURE AND TASK ASSIGNMENTS

A. *Work Breakdown Structure*

The following table is an overview of our project's work breakdown structure. The primary features of project are shown in the first column as Level 1, following the secondary features as Level 2, and finally broken down to smaller components as Level 3.

WORK BREAKDOWN STRUCTURE
Team 1 - Automated Smart Wheelchair

Level 1	Level 2	Level 3
Battery Management System	Battery Monitoring	Voltage Measurement
		Voltage Percentage
	Charging System	DC CC/CV Power Supply
Motor Control	DC Motor Controller	Encoder Feedback
		Kangaroo x2 motion controller
Navigation	2D Mapping	Kinect Camera
	Foor to Floor	Altitude Sensor
	Room to Room	ROS
	User Interface	Rviz
Safety Feature	Alert System	Speaker
	Collision Avoidance	Ultrasonic Sensors
	Lightning System	LED Strip
		Light Sensor

Table 3. Project WBS chart

B. *Task Assignments*

This section will cover the task assignment of each individual team member and what they worked on throughout two semesters. It will include hours worked per task, hours per team member, and what their task was. In the table 4, we calculated all hours that were spent on each category. Note that these hours do not include the team meeting time spent, we only accounted for the time spent on the actual project.

TASK ASSIGNMENT
Team 1 - Automated Smart Wheelchair

Task	Benson	Kenneth	Quan	Tim	Group
2D Mapping	10	11	10	73	104
Battery Charching System	26	0	0	0	26
Battery Monitering System	34	0	0	0	34
Documentation	92	94	101	90	377
Encoder Setup	12	39	36	9	96
Hardware Configuration	156	152	154	41	503
Motor Controller Calibration	28	33	26	12	99
Navigation	10	21	13	85	129
Safety Feature	0	0	40	0	40
Software Coding	104	106	91	130	431
Testing	150	150	150	150	600
User Interface	0	15	0	29	44
Grand Total	622	621	621	619	2,483

Table 4. Task Assignment

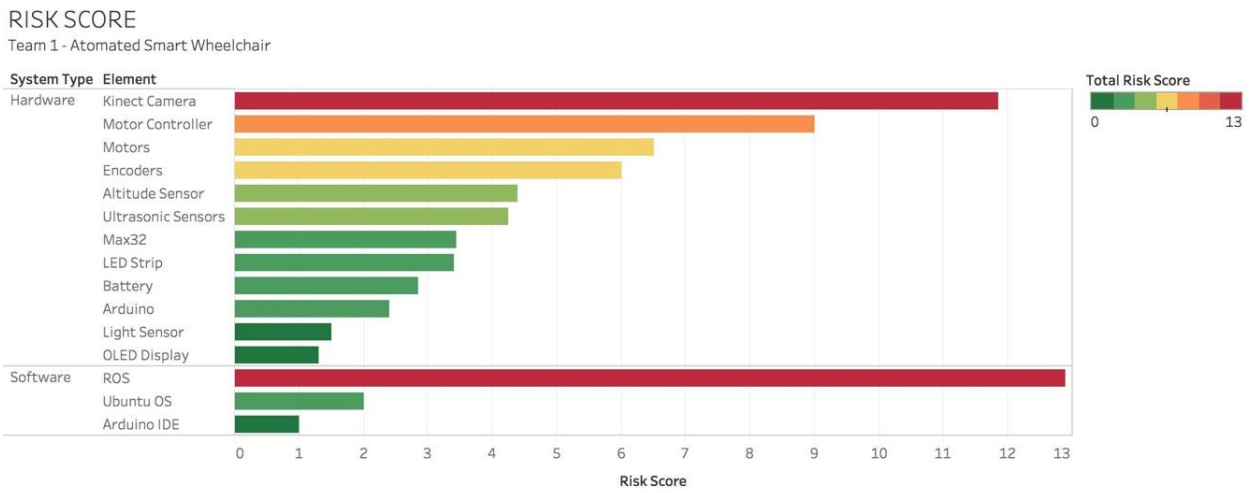


Figure 17. Risk Score

VIII. RISK ASSESSMENT AND MITIGATION

Using the work breakdown structure that we discussed in the previous section, we identified all the risks that can happen to each of the component. After the risk identification, we analyze the risk based on their likelihood of failure and the consequence of them. We rate the likelihood of failure on the scale from 1 to 5, which is the failure of element is likely within from 10% up to around 90%. In addition, we rate the consequence of failure in based on Safety and Operations. In Safety, we concern about Public Safety and User Safety. On the other hand, Operational Impact is the failure of element may impact to meeting the deadlines. We also rate them on the scale from 1 to 5, where 1 is no impact at all, up to 5 is the most impact.

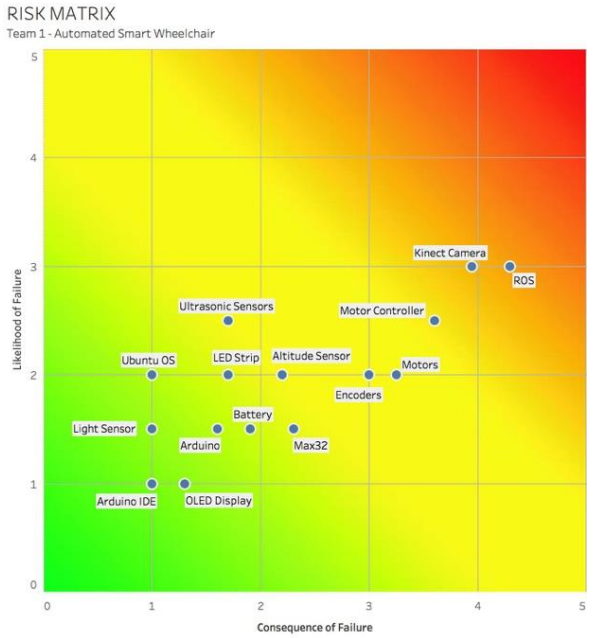


Figure 18. Risk Matrix

Next, we created a scoring table to record of the score of each element that we discussed earlier. We also added a total risk score column, which is equal the impact score multiplies with the probability score. We used this column to put together a Risk Score chart that is shown in figure X.

The following figure is the Risk Matrix as a visualization for our risk management. The x-axis is the consequence of failure and the y-axis is the likelihood of failure, which is the probability of the risk that may happen. As expected, Kinect Camera and ROS are among the highest risk for our project, followed are encoders, motor, motor controller, etc.

A. Hardware

1. Kinect Camera

Navigation and 2D Mapping are two of the most important feature that defines our project. The Kinect camera is what made them possible. Without the Kinect camera working properly, it would highly impact the rest of the project in term of delivering the project on time and keeping it safe for our users as well as other surrounding people and properties. We bought our Kinect camera used and have modified its power cable so it can connect to our power source; therefore, it has higher risk of stop working. Given these points, the Kinect Camera has the highest risk score in comparison with other hardware on this project.

If the Kinect camera stop working for some reason. First, we will troubleshoot the problem and try to fix it. If that is not the case, we can always purchase another one to replace it.

2. Motor Controller

The motor controller is an important component of our project. Our motor controller does not have reverse polarity protection; therefore it is possible to destroy it by just connecting the power source backwards. Another potential issue would be something shorting out or overheating. This would most likely damage the MOSFETs due to excessive current draw or heat, which would also render the motor controller useless afterwards. Lastly, there is also the risk of the motor controller simply not working one day. This could be caused by either loose components or a failure of a component(s) on the board. Altogether, it is understandable to see the mother controller is among the ones with highest risk score.

If the motor controller is damaged, we have considered several mitigation plans. First, we will attempt to repair it. If that is not possible, we can use our backup motor controller; however, it does not have the brake feature. Lastly, we can always purchase a new motor controller if the first two options are not possible.

3. Motors

Motors are another crucial component of our project. They are still new, so we do not expect them to be broken anytime soon. However, it is still possible to damage the motors from improper usage. For instance, if the motor controller does not disengage the brake and tells the motors to turn with them engaged; this will damage the braking system, which could cause damage the actual motor, since they are connected to one another. For another example, the motors could lock up due to a foreign object while it is still being fed power; this would cause the motor to overheat and damage the coil that produces the magnetic field which would destroy the motor. Given these points, the motors have medium to high risk score.

If a motor becomes damaged or malfunctions, we will have to purchase new ones. Motors for wheelchairs are not cheap, a new motor would cost around $200. They are very robust and are engineered specifically for a certain wheelchair. Based on our budget plant, we have more than enough funds to purchase one new motor without worrying too much.

4. Encoders

Encoders are another integral part of this project. If they are not working properly, the wheelchair will be lost and cannot move or started to move in unexpected direction; which may cause damage to the users and people around them. There are several risks that are associated with the encoders. For example, they are attached to the rear wheel of the wheelchair. People may not see it and kick it by accident. If an encoder is damaged or malfunctions, we can always try to fix it first. If not, the only other option is purchasing a new one.

5. Microcontroller

The microcontroller can be considered of the brain of our project. It could malfunction or become damaged through user error. In addition, the frame of the wheelchair is made of steel. Therefore, we could short out the microcontroller without noticing if we misplace the microcontroller, while going back and forth between programming the microcontroller and testing it with the wheelchair.

If the microcontroller Max32 or the Arduino stop working, we must try to debug the problem first. However, if the problem is too complicated and unable to fix, we can replace the microcontroller with a different one.

6. Sensors

We use a variety of different sensor for our project; namely, ultrasonic sensors, altitude sensor, and light sensor. The light sensor has the least risk score, because it is a stand-alone feature and does not have much impact on the other components. On the other hand, the altitude sensor has higher risk score, because it helps our product to navigate through each floor. Lastly, the ultrasonic sensor has the highest risk score out of the three's, because it keeps safety for our users and the surrounding people. All of our sensors are quite cheap. Therefore, we could replace the sensors with a new one if one gets damage.

B. Software

As has been said, navigation is the most important feature of our project; it works

because of ROS. Therefore, ROS in one of the most crucial component of our project . It is complicated and a risky investment because we did not have any experience on it before. Because of that, ROS has the highest risk in our analysis. In order to help mitigate the risk for getting ROS working, we have a backup plan of using line following for our navigation.

IX. DESIGN OVERVIEW

We wanted our design to be something that would be beneficial to people with disabilities. We wanted the wheelchair to be fully autonomous as our overall goal. However, due to time and hardware limitations, we were only able to achieve about 80% autonomy.

Our philosophy was to remove the user interaction and have everything as autonomous as possible. We did some research for this and found out about ROS (Robot Operating System), it is widely used in pretty much all autonomous projects by individuals or larger companies. Once we decided on ROS, we needed to pick out our hardware. Due to us being college students, we couldn't afford the best hardware therefore we had to make compromises, choose components that were compatible, but at the same time, good enough so that we would get good results. For the 2D mapping, we decided on the Xbox 360 Kinect, this camera is widely used in the hobbyist world and has a decent amount of documentation on it so we decided that would be a good choice. The kinect will be the main source of information for our whole project. It will scan the environment and create a virtual 2D map in our database, which can then be referenced later on to navigate through its environment. However, one downfall to 2D mapping is that it needs odometry information in order to keep track of where it's going, this normally would not be an issue with 3D mapping. We initially planned on 3D mapping but we later found out that we did not have enough processing power, since we are using the Raspberry Pi 3. Nonetheless, to solve the issue with odometry, we acquired encoders which will keep track of how far the wheelchair has gone at any given time. Given this information, within ROS, it will be able to locate where the wheelchair is at based on odometry information on the 2D map. This will allow the wheelchair to know if it has reached its location or not.

Next was choosing suitable motor controllers for our wheelchair. Because we are using microcontrollers for pretty much everything in our project, we needed something that could communicate with them on a digital level and not an analog one. Communicating with analog inputs would be very restrictive, limiting us on what we could tell the wheelchair to do. With digital inputs, we can send much more accurate commands to the wheelchair as well as receive information back from the motor controller. We ultimately decided on the Sabertooth 2x32A motor driver. It had all the features we needed, which we will go more in detail in the hardware appendix section. But the most important feature was the option to use serial input from a microcontroller. This was exactly what we were looking for as this simplifies everything tremendously.

Our working prototype is a compilation of various components that we worked on individually throughout the semester, combined. To give an idea on how everything goes together, we have the kinect connected via USB to the Raspberry Pi 3. The Raspberry Pi 3, which is the main processor for ROS, is then connected to the max32 microcontroller which communicates to the motor controller. Through encounters with the low process power of the Raspberry Pi, we completely take out the Raspberry pi and do all the processing on a laptop with much stronger CPU. The motor controller also has a feedback controller connected to it (SaberTooth Kangaroox2) which provides adjustments necessary from the encoders to keep the wheelchair going in a straight line. There are still 2 components that we still want to improve upon in term of mounting hardware if we have more time in this semester, which would be the location of the encoders and the kinect. They work fine as of now, but we want to make them more effective and more hidden.

We are currently able to select a destination on our 2D map of the 3rd floor of Riverside. Once selected, the wheelchair will be able to navigate to that destination autonomously. In the future, if we had more time we would also have done a better user interface. Currently, we have a start button which will run all of the initially commands needed to start and initialize ROS. All the user has to do is select their destination and the wheelchair will do the rest. Although there are many sub-categories for this project that are needed to make the overall design functional, we will leave the finer details in the designated sections.

X. DEPLOYABLE PROTOTYPE STATUS

The current status of our deployable prototype is functional. We successfully met our goals that we agreed upon at the beginning of our project. The wheelchair is able to navigate autonomously, it can detect what floor it's currently on, there is obstacle avoidance and emergency stop. The wheelchair performs very well given our current hardware. For example, our Kinect camera has a very limited field of view, of roughly 60-degrees on either side. Whereas, a true laser scanner made for navigating autonomously would have a complete 360-degree field of view. Which would be much more ideal in our project. Our deployable prototype meets essentially all of our desired goals that we discussed in the beginning of the semester. One of the most important features that we accomplished was 2D mapping a floor and using that 2D map to navigate autonomously inside that map.

XI. DEPLOYABLE PROTOTYPE MARKETABILITY FORECAST

To make our deployable prototype marketable we need to polish up a few features. One of the major things that we need to do is to use a laser scanner instead of using a Kinect camera. This will provide a complete 360-degree field of view for ROS. We also need to use a more powerful laptop as ROS requires a lot of processing power. We also need to create a better mount for our laptop holder as currently it is just barely enough to hold it in place, we did it that way for testing purposes. In the final design we would like to weld together a frame that would be able to fully support the weight of the laptop and prevent any unwanted vibrations that may be introduced into the system during operation. Another thing that we need to do is to create an enclosure so that we can place our microcontrollers and motor controller in it. This will provide a weather proof design so that the user can operate the wheelchair either indoors or outdoors. We also need to mount our encoders in a more optimal position. Currently the encoders are in a vulnerable location and can be damaged if it hits a wall or gets hit by someone or something.

XII. CONCLUSION

In conclusion, after working on our project for the span of 2 semesters. We were able to meet our goal of having the wheelchair navigate autonomously within a given environment. This project presented us with many challenges and problems. While we were able to solve most of the problems, the ones we weren't able to solve we used an alternative method. By completing this project, we were able to present the world a new technology that can help the lives of many people with disabilities.

XIII. REFERENCES

[1] Office of Communications and Public Liaison, "Amyotrophic Lateral Sclerosis (ALS) Fact Sheet," *National Institute of Neurological Disorders and Stroke*. [Online]. Available: https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Amyotrophic-Lateral-Sclerosis-ALS-Fact-Sheet

[2] "Incidence of ALS," *Incidence Rate of ALS - ALS Facts & Statistics | UC San Diego*. [Online]. Available: http://als.ucsd.edu/about-als/Pages/incidence.aspx.

[3] "Stroke Facts," *Centers for Disease Control and Prevention*, 06-Sep-2017. [Online]. Available: https://www.cdc.gov/stroke/facts.htm. [Accessed: 10-Sep-2017].

[4] National SCI Statistical Center, "Spinal Cord Injury (SCI) Facts and Figures at a Glance," *NSCISC*, 2016. [Online]. Available: https://www.nscisc.uab.edu/Public/Facts%202016.pdf.

[5] K. Nguyen, *Stroke Recovery Rate*. .

[6]] AANS , "Traumatic Brain Injury," American Association of Neurological Surgeons. [Online]. Available: http://www.aans.org/Patients/Neurosurgical-Conditions-and-Treatments/Traumatic-Brain-Injury. [Accessed: 10-Sep-2017].

[7] R. C. Rabin, "Study Raises Estimate of Paralyzed Americans," *The New York Times*, 20-Apr-2009. [Online]. Available: http://www.nytimes.com/2009/04/21/health/21para.html?mcubz=0. [Accessed: 10-Sep-2017].

[8] Dimension Engineering, "Kangaroo x2 Motion Controller," *Dimension Engineering*, 2013. [Online]. Available: https://www.dimensionengineering.com/datasheets/KangarooManual.pdf.

[9] CUI Inc, "MODULAR INCREMENTAL ENCODER," *CUI Inc*, 04-Sep-2015. [Online]. Available: http://www.cui.com/product/resource/amt10-v.pdf.

XIV. GLOSSARY

ROS: Robot Operating System (ROS) is robotics middleware (i.e. collection of software frameworks for robot software development). Even though ROS is not an operating system, it provides services designed for heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management.

rviz: (ROS visualization) is a 3D visualizer for displaying sensor data and state information from ROS. Using rviz, you can visualize Baxter's current configuration on a virtual model of the robot. You can also display live representations of sensor values coming over ROS Topics including camera data, infrared distance measurements, sonar data, and more.

gmapping: The gmapping package provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called slam_gmapping. Using slam_gmapping, you can create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

APPENDIX A. USER MANUAL

Boot up the system: open system's terminal and start up the software by double clicking the play icon application.

The following rviz window should pops up.

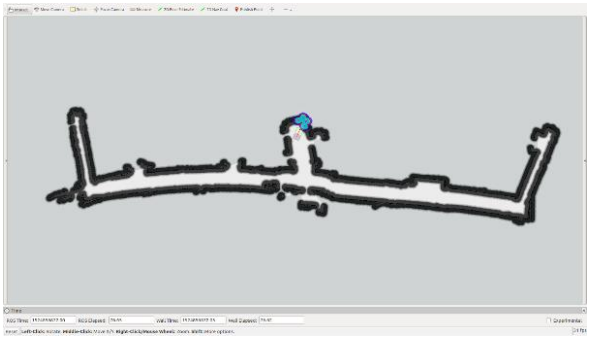
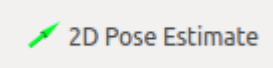


Figure 17. 2D Map

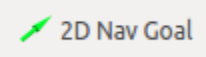
Assign current location and final destination:

Assign current location: click 2D Pose Estimate



Locate your current location on the virtual map. Click and drag the arrow to the direction the wheelchair is facing.

Assign final destination: click 2D Nav Goal



Locate your desired destination on the virtual map. Click and drag the arrow to the direction you want the wheelchair to face.

Quit the program: After finish using the wheelchair, close program by clicking exit.

APPENDIX B. HARDWARE

This section will cover how each individual part is connected and wired at both the block and component level. Therefore, if maintenance is needed in the future, this will serve as a very important resource.

A. *Block Diagram*

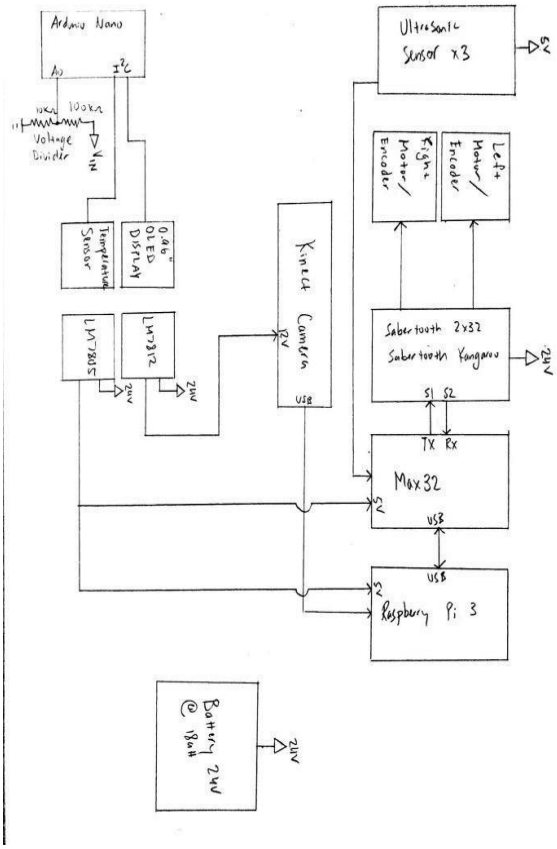


Figure 18. Block Diagram of Hardware

Shown in Figures in Appendix B, we can see the overall layout of the components and their interconnections. This is a fairly simplified version as the actual components have many more connections that need to be made within a single block. However, for basic troubleshooting, the block diagram will be more than sufficient to guide the user to diagnosing problems within the system.

B. *Schematic Diagram*

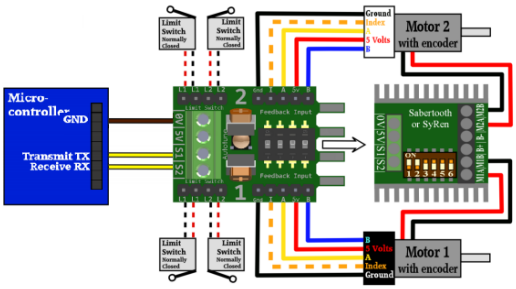


Figure 19. MotorController and Encoder Diagram

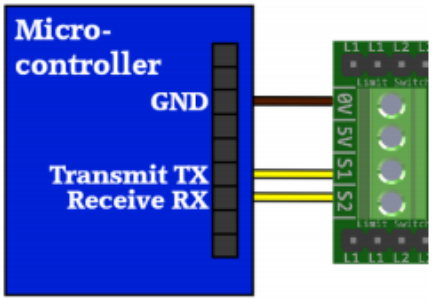


Figure 20. Wiring Between MotorController and Kangaroo

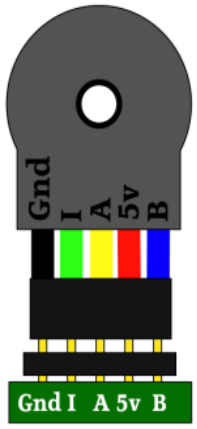


Figure 21. Encoder Pinout

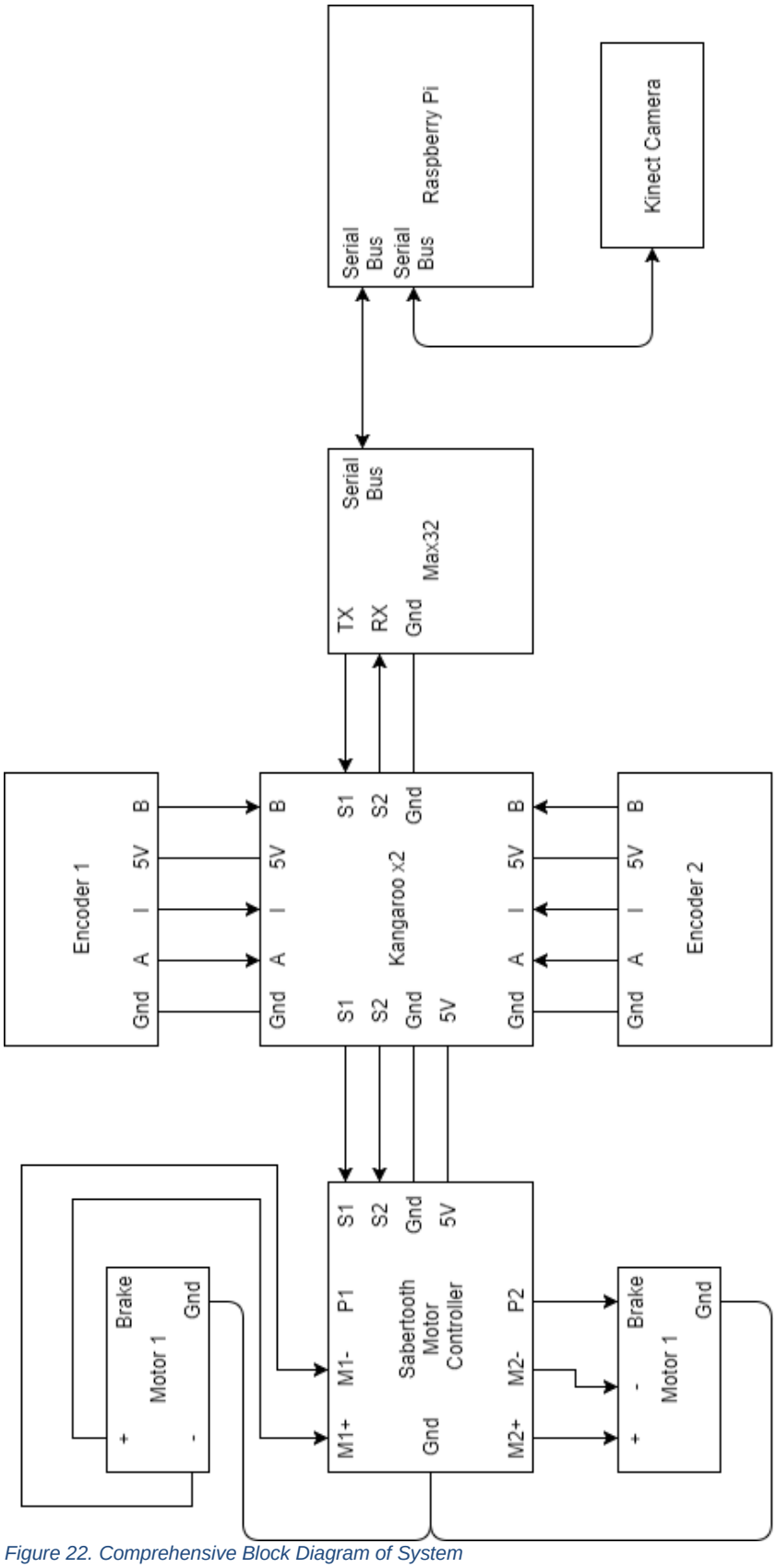


Figure 22. Comprehensive Block Diagram of System

C. Device Test Plan

1. Component test

- Ultrasonic Sensors:

Since the ultrasonic sensor used in this project are cheaply made, testing for functionality of individual sensors are required. The testing for the ultrasonic is to send a simple trigger code and measure the traveling time of sound relative to the testing distance. The measured distance using a ruler should be close to the distance measured by the ultrasonic sensor, or else it could not be used in the project. Example code for testing the ultrasonic sensor can be found in the following URL.

http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

- Speakers

To make sure the speaker that we installed for the Obstacle Avoidance system work properly, we first connected it to audio source and verify there is sound come out of it.

- LEDs

Like testing the speakers, the procedure to test the LEDs should be simple. We connected all the LEDs that we are going to use to a microcontroller, such as the Arduino Uno. Then, we write a basic code to turn on and of the LEDs at a certain rate. These LEDs are expected to turn on and off at the rate that we coded.

- Motors

Since the motor is the basis of the wheelchair, it is needed to test for it when we first bring it into the lab. Because there is an electronic braking system installed into the motor itself, it is important that the brake is given enough voltage to disengage before testing the functionality of the motors. We add a low voltage to the motor first and gradually increase it to make sure that we do not damage the motor while testing it. After we see that it moves, we reverse the polarity of the input voltage into the motor to test for the reverse functionality of the motor.

- Wheelchair Frame

For the wheelchair frame, we are going to test its ability to handle large weight on it. It is expected for the wheelchair frame to handle up to 300 lbs. Therefore, we are going to put weight on it, then increase the weight slowly up to 300 lbs. to see if it can handle the weight.

- Kangaroo for Motor Controller

The kangaroo is an important component that is needed for ROS since it provides odometry information. Odometry is important since it allows ROS to calculate where our wheelchair is within the 2D map. Therefore, the functionality of the kangaroo is vital for accuracy. To test this, we will make sure that the kangaroo is not giving out any errors upon startup. The kangaroo requires a onetime calibration and then that calibration is saved. Although, a recalibration is needed every time something major is changed within the setup. Such as, new motors, significant weight changes, and so forth. The kangaroo implements a very useful status L.E.D. which flashes at certain rates to tell the user if there is an error, and what that error is.

2. Integration Test

- Motors and motor controller/Kangaroo

Testing the Kangaroo with the motors and motor controller will be done in the Riverside Building hallway. Since the hallway is long, we can test long distance traveling of the wheelchair. If the PID of the Kangaroo, motors and motor controller works, it should go on a relatively straight line.

- Microcontrollers and sensors

To ensure that the sensors are communicating properly with the microcontroller and vice versa. We will first ensure that the sensor themselves are not faulty and that the wiring is correct. Once we know that there is nothing wrong on the hardware side, we will test the software side. This can range from verifying the code and pulling data from the sensor to ensure that we are acquiring an accurate reading.

- Arduino and Max32

To test to see if the Arduino can send signals over to the Max32, we can write a simple test code that turns on an L.E.D. on the Max32, when the Arduino sends a signal over. This will allow us to easily verify that the communication is present and working.

- Max32 and Raspberry Pi

This is probably one of the more important connections as this is where most of the information is transferred. Such as odometry, motor controls, and obstacle avoidance data. Luckily, the connection

between these two devices can be easily tested by sending various commands to the Max32 and observing the output. Most of the communication being done would be the Raspberry Pi sending information to the Max32, such as motor controls.

- Raspberry Pi and Laptop running ROS node

These two components will be important as the laptop will be our eyes into ROS. The laptop will ultimately be our user interface. To ensure that they are communicating properly we can check to see if information is being received and sent to either devices. Which is simple as we would not be able to see anything if ROS did not recognize either device. The laptop will be hardwired via an ethernet cable so dropped connections should not be an issue, unless there is a problem with the ethernet cable itself, which in that case, can be easily fixed with a replacement cable.

D. Testing Result

1. Kinect Camera

- Test date: 02/05/2018
- Tool: Kinect camera, a computer with ROS and frrenect_famera ROS package installed
- Expected result: the Kinect camera should be able to show image on the computer when run command "roslaunch freenect_launch freenect.launch"
- Test result: the Kinect camera successfully show image on the computer
- Number of bugs/failures: 1
- Additional comment: The Kinect did not turn on; the reason for this the power supply of the camera is broken. The Kinect is working well

Ultrasonic Sensors

- Test date: 02/06/2018
- Tool: Ultrasonic sensors, Arduino microcontroller, a computer with Arduino IDE installed, wires.
- Expected result: The Ultrasonic should be able to send a ping signal; then, we will measure the traveling time of sound relative to a certain distance. This distance should be close to the distance measure by the ultrasonic sensor.
    - Test result: the measured distance using the ultrasonic sensors is within the acceptable boundary of the real distance.
- Number of bugs/failures: 1
- Additional comment: Should increase numbers of ultrasonic sensors around the chair to increase safety.

2. Speakers

- Test date: 02/06/2018
- Tool: Speakers, Arduino microcontroller, a computer with Arduino IDE installed, wires.
- Expected result: The speakers should be able to produce sound when connect to an audio source.
- Test result: The speakers successfully make sound when connected to an audio source.
- Number of bugs/failures: 0
- Additional comment: none

3. LEDs

- Test date: 02/06/2018
- Tool: LEDs, Arduino microcontroller, a computer with Arduino IDE installed, wires.
- Expected result: The LEDs should be able to turn on and off at a certain rate.
- Test result: The LEDs strip successfully turn on and off using a sample code.
- Number of bugs/failures: 0
- Additional comment: none

4. Motors

- Test date: 02/07/2018
- Tool: Motors, Voltage source
- Expected result: The motor should be able to move forward and backward when there is a voltage add to it.
- Test result: After slowly increase the voltage, the motors successfully moved forward. We then reversed the polarity of the input voltage, the motors moved backward.
- Number of bugs/failures: 0
- Additional comment: None

5. Wheelchair Frame

- Test date: 02/07/2018
- Tool: Wheelchair, Weights
Expected result: The wheelchair frame should be able to handle up to 300lbs.

- Test result: We slowly increase the weights that we put on the wheelchair to 300lbs. It is confirmed that the wheelchair frame can handle up to 300lbs.
- Number of bugs/failures: 0
- Additional comment: None

6. Kangaroo

- Test date: 02/07/2018
- Tool: Kangaroo, Motor controller
- Expected result: The Kangaroo should not give out any errors upon startup.
- Test result: The Kangaroo's error light did not light up.
- Number of bugs/failures: 2

Additional comment: The Kangaroo needs to recalibrate everything there is a major change in setting up. Therefore, it is critical to setup everything correctly then recalibrate the Kangaroo.

7. Motors and Motor Controller/Kangaroo

- Test date: 02/14/2018 - 02/15/2018
- Tool: Motors, motor controller, Kangaroo, power source, a controlled testing location, wires, cables.
- Expected result: The kangaroo should be able to help the motors to move the wheelchair in a straight line.
- Test result: The wheelchair went in a straight as expected
- Number of bugs/failures: 0
- Additional comment: None

8. Microcontrollers and sensors

- Test date: 02/15/2018 - 02/16/2018
- Tool: Microcontrollers, sensors, wires, cables.
- Expected result: The microcontrollers should be able to reads the data that sensors collected.
- Test result: All microcontrollers can read the data from all the sensors.
- Number of bugs/failures: 0
- Additional comment: None

9. Arduino and Max32

- Test date: 02/19/2018 - 02/20/2018
- Tool: Arduino, Max32, a computer with Arduino IDE, LEDs, wires, cables,
- Expected result: Arduino should be able to send signal to Max32. In our case, Arduino should turn on an LED that connected to the Max32.
- Test result: The LED that connected to the Max32 is turned on using a command from Arduino, which confirmed the integration between the Arduino and Max32
- Number of bugs/failures: 0
- Additional comment: None

10. Max32 and Raspberry Pi

- Test date: 02/20/2018 - 02/21/2018
- Tool: Max32, Raspberry Pi, power source, wires, cables.
- Expected result: using various commands, Raspberry Pi should be able to send information to Max32 and Max32 should be able to reads it.
- Test result: Max 32 is able to reads the information that was sent from Raspberry Pi
- Number of bugs/failures: 0
- Additional comment: None

11. Raspberry Pi and Laptop running ROS node

- Test date: 02/21/2018 - 02/22/2018
- Tool: Raspberry Pi, a computer with ROS installed, power source, wires, cables.
- Expected result: ROS should be able to recognize Raspberry Pi. Then, Raspberry Pi should be able to send information to ROS and ROS should be able to reads it.
- Test result: ROS recognized Raspberry Pi.
- Number of bugs/failures: 0
- Additional comment: None

12. User Interface and ROS

- Test date: 02/22/2018 - 02/23/2018
- Tool: a computer with ROS installed, user interface.
- Expected result: the user interface should be able to send message to ROS to control the wheelchair.
- Test result: User interface is unable to send any signal to ROS.
- Number of bugs/failures: 1
- Additional comment: we need more time to integrate the connection between ROS and our user interface.

APPENDIX C. SOFTWARE

A. *Referencing Material*

ROS Install
http://wiki.ros.org/kinetic/Installation/Ubuntu

ROS Odometry
http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom

ROS gmapping
http://wiki.ros.org/gmapping

ROS teleop
http://wiki.ros.org/teleop_twist_keyboard

ROS navigation
http://wiki.ros.org/navigation

ROS rviz
http://wiki.ros.org/rviz

ROS costmap_2d
http://wiki.ros.org/costmap_2d

ROS pointcloud_to_laserscan
http://wiki.ros.org/pointcloud_to_laserscan

ROS transforms
http://wiki.ros.org/tf

ROS static transform_publisher
http://wiki.ros.org/tf#static_transform_publisher

ROS move_base
http://wiki.ros.org/move_base

ROS rosserial
http://wiki.ros.org/rosserial

ROS freenect_launch
http://wiki.ros.org/freenect_launch

ROS map_server
http://wiki.ros.org/map_server

ROS roslaunch
http://wiki.ros.org/roslaunch

ROS amcl
http://wiki.ros.org/amcl

ROS base_local_planner
http://wiki.ros.org/base_local_planner

B. *Block Diagram*

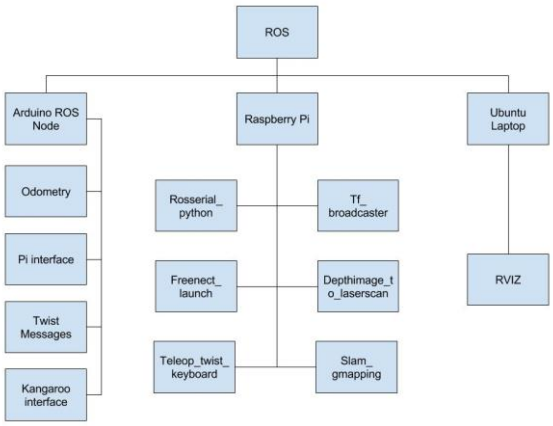


Figure 23. Software Block Diagram

This project uses ROS for the primary software system. ROS is running on three devices. The core ROS software is running on a Raspberry Pi. The Pi runs most of the ROS nodes and handles passing messages between the other devices. An Arduino is used to interface with the Kangaroo X2 Motion controller, handle Twist messages, and broadcast odometry messages. The laptop is used to run RVIZ to visualize ROS messages and data.

The Arduino ROS Node publishes Odometry using ROS messages. The current angle is read from the Kangaroo X2 and converted from ticks to radians. The x and y coordinates are estimated using the angle and the linear velocity with the amount of time that has passed. Twist messages are accepted and used to control the linear and angular velocities. The Twist messages are made of two vectors. One for the linear and one for the angular velocities. Library functions for the Kangaroo are used to send volocity messages over a serial interface to the Kangaroo. The Raspberry Pi interfaces with the Arduino with a serial connection over a USB cable. ROS messages are both sent and received over this connection.

ROS Core runs on the Raspberry Pi and handles all the messages between nodes. Rosserial_python is a node that handles the serial communication with the Arduino. Tf_broadcaster is a custom written ROS node that is used to translate the laser scan frame to the base frame. This is so the mapping knows how far the incoming sensor information is offset from the center of movement. Freenect_launch controls and broadcasts all the sensor information from the Kinect. Depthimage_to_laserscan translates the depth image from the Kinect into a 2D laser scan for gmapping. Slam_gmapping is used for Simultaneous Localization and Mapping. This creates a map from the incoming laser scans and also helps identify where the robot is on the map. Teleop_twist_keyboard is a temporary node used for testing. It creates twist messages to control the robot movement from keyboard commands. Once we get the full navigation stack running it won't be needed anymore.

The third device is an Ubuntu laptop. The laptop is being setup to run rviz as a remote node to read ROS messages and display mapping data. It may be used as the primary interface for controlling the system in the future.

C. *Flowcharts*

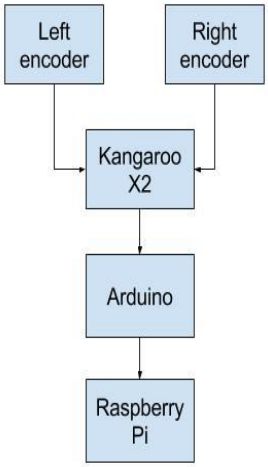


Figure 24. Odometry flow

The Kangaroo X2 reads the encoder information directly. The Arduino requests the current position information from the Kangaroo using library calls and translate this information into a ROS message that is then sent to the Raspberry Pi where it can be accessed by any of the other ROS nodes that need it.

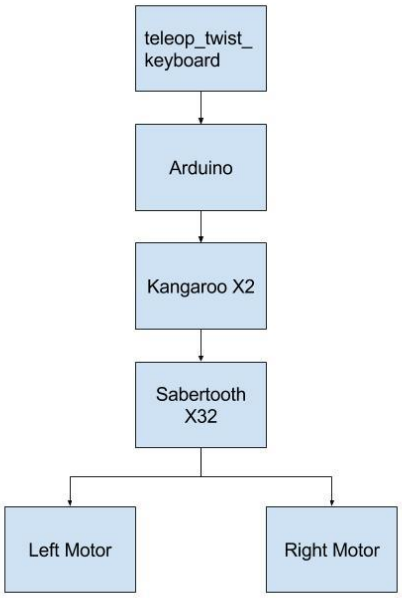


Figure 25. Twist flow

Twist messages are created by the teleop_twist_keyboard node and are sent to the Arduino. The Arduino translates the two vectors into two function class to the Kangaroo to set the linear and angular velocities. The Kangaroo combines the velocities with feedback information from the encodes to control the Sabertooth. The Sabertooth then controls the left and right motors directly.

D. *Device Test Plan*

1. Component Test

- Kinect camera:

To make sure that the camera is fully functional, the Kinect camera is required to connect to the computer or microcontroller such as the pi and run a test run. This will require the computer to have ROS already installed as well as the freenect_camera ROS package. We then will run the following command:

```
roslaunch freenect_launch freenect.launch
```

If the Kinect camera is working properly, after entering the command, there should be an output from the camera will be as illustrated in Figure 1.

Figure 26. Freenect

- ROS

Each part of ROS will be tested using RVIZ. RVIZ is a tool for visualizing different messages in the ROS system. A laptop has been setup and connect with the ROS core running on the Raspberry Pi. Now Odometry and all the steps involved in mapping and navigation can be analyzed and checked individually.

2. Integration Test

- ROS and Kinect camera

The ROS messages providing information from the Kinect camera has already had basic testing done. We still need to fine tune the settings for the Kinect to remove unneeded information and make sure the incoming data is responsive enough for the navigation system to make quick decisions.

- ROS and Ultrasonic Sensors

The Ultrasonic Sensors are handled outside of ROS and send an emergency stop signal to the ROS node that handles motor control. This still requires additional debugging and testing to make sure the motors stop without causing issues in the ROS Core.

- ROS and Encoders

The encoders are primarily handled by hardware on the Kangaroo. The Max32 reads the distance information for the Kangaroo and translates it into odometry information for ROS. The hardware has already been confirmed to work as expected. There is currently a bug in the translation from the encoder distance information to the ROS odometry information. This is a matter of formatting the information correctly, but will still require more software testing after the bug is resolved.

- User interface and ROS

The user interface will be the input of the whole system. To test it, we first need to have a ROS node that could send messages from the laptop to move to different location. We then will use a user-friendly interface with simple directions to send those command by just a simple gesture. This one gesture will allow the laptop to send a message to the system through ROS and control wheelchair and perform its navigation feature.

E. Test Result

3. Kinect Camera

- Test date: 02/05/2018
- Tool: Kinect camera, a computer with ROS and frrenect_famera ROS package installed
- Expected result: the Kinect camera should be able to show image on the computer when run command "roslaunch freenect_launch freenect.launch"
- Test result: the Kinect camera successfully show image on the computer
- Number of bugs/failures: 1
- Additional comment: The Kinect did not turn on; the reason for this the power supply of the camera is broken. The Kinect is working well now after we fixed the power supply.

4. ROS

- Test date: 02/05/2018
- Tool: a computer with ROS and RVIZ installed
- Expected result: in ROS, RVIZ should show visual messages regards odometry, mapping, and navigation
- Test result: RVIZ successfully show messages in ROS, which confirmed that ROS is working.
- Number of bugs/failures: 0
  Additional comment: None

5. ROS and Kinect camera

- Test date: 02/08/2018 - 02/09/2018
- Tool: Kinect camera, a computer with ROS installed, power source, cables.
- Expected result: ROS should be able to communicate with the Kinect camera and the camera should be able to send back information to ROS.
- Test result: the Kinect camera successfully sent image to ROS to create 2D map
- Number of bugs/failures: 0
- Additional comment: None

6. ROS and Ultrasonic sensors

- Test date: 02/12/2018 - 02/13/2018

- Tool: Ultrasonic sensors, a computer with ROS installed, Arduino microcontroller, wires, cables.
- Expected result: The Ultrasonic sensors should be able to send an emergency stop signal to ROS node, then ROS should be able to stop the motors
- Test result: The Ultrasonic sensors is able to send a signal when they detect obstacles. However, ROS still not be able to reads this signal, to stop the motors.
- Number of bugs/failures: 1
- Additional comment: We need to spend more time to integrate ROS and the Ultrasonic sensors.

7. ROS and Encoders

- Test date: 02/13/2018 - 02/14/2018
- Tool: Kangaroo, Max32, Encoders, a computer with ROS installed, power source, wires, cables.
- Expected result: the encoders should be able to send the data it collects from spinning the wheel to the Kangaroo. Then, the Max32 reads the distance information for the Kangaroo and translates it into odometry information. Finally, ROS should be able to reads this information.
- Test result: After all time and effort of debugging the Odometry function, ROS communicate with Encoders.
- Number of bugs/failures: 0
- Additional comment: None

8. User Interface and ROS

- Test date: 02/22/2018 - 02/23/2018
- Tool: a computer with ROS installed, user interface.
- Expected result: the user interface should be able to send message to ROS to control the wheelchair.
- Test result: User interface is unable to send any signal to ROS.
- Number of bugs/failures: 1
    - Additional comment: we need more time to integrate the connection between ROS and our user interface.

APPENDIX D. MECHANICAL

This section will cover any mechanical drawings that we have done for our project. One thing that we did have to design was an adapter plate that went over the wheels on the wheelchair which would serve as a connection to our encoders. This was necessary as we did not have any other option to mount the encoders, since the shaft of the wheelchair motor was too large. We also did not want to use optical encoders as we do not think they would be accurate enough. Therefore, we designed something purely for testing purposes but it ended up being used in our prototype as it worked better than we had initially anticipated.

This is a CAD image of our mount for the encoder.

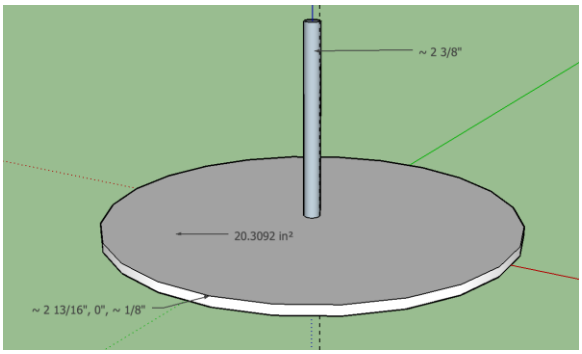


Figure 27.CAD Drawing of Encoder Adapter

Image of the encoder adapter mounted on the wheelchair's wheel.

Figure 30. Encoder Adapter Mounted

The encoder adapter was 3D printed. Due to the fact that it was 3D printed, it was not 100% to scale, it was actually 1-2mm off measurement in comparison to our CAD drawings. Although, once mounted it performed better than we thought it would and we ended up using them throughout our testings. The encoders have no problem taking a reading from them and moves in a very straight line when told to do so based on the feedback given from the encoders. We plan to improve this design to make it more robust and hidden, as currently it does stick out quite a bit which limits the agility of our wheelchair.

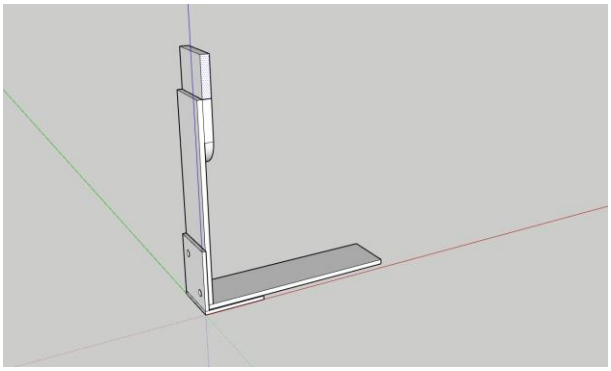


Figure 28. Angle bracket for encoder back view

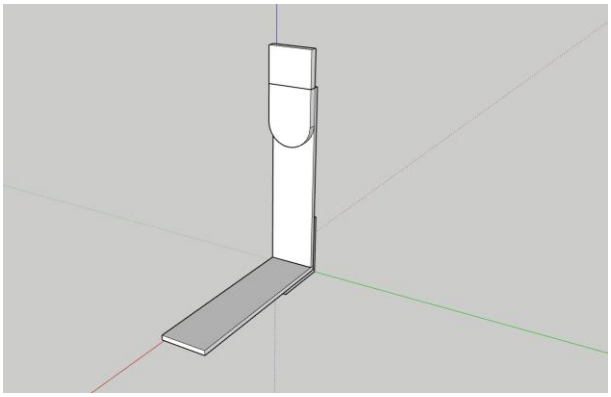


Figure 29 Angle bracket for encoder front view

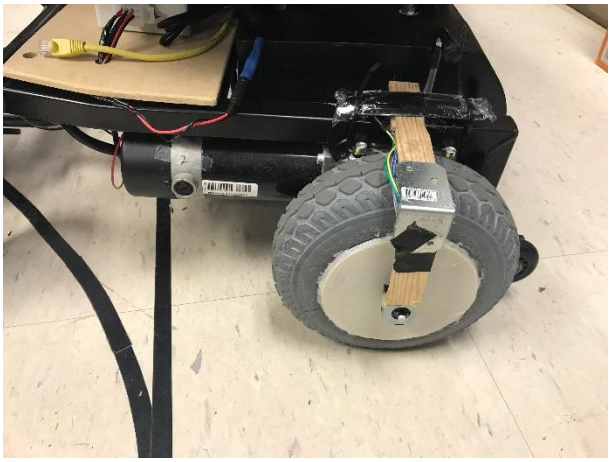


Figure 30. Encoder mounted

As you can see, the mount fit quite well onto the wheelchair.

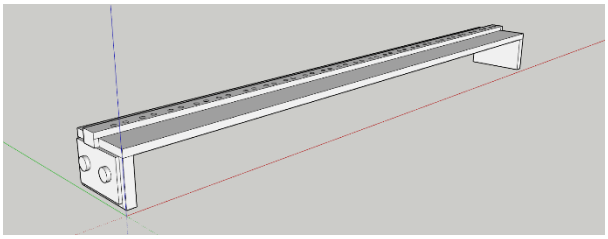


Figure 31. LED mount

Figure 32. LED strip mounted

Figure 33. Laptop mount on wheelchair

Figure 34. LED strip mounted onto laptop mount

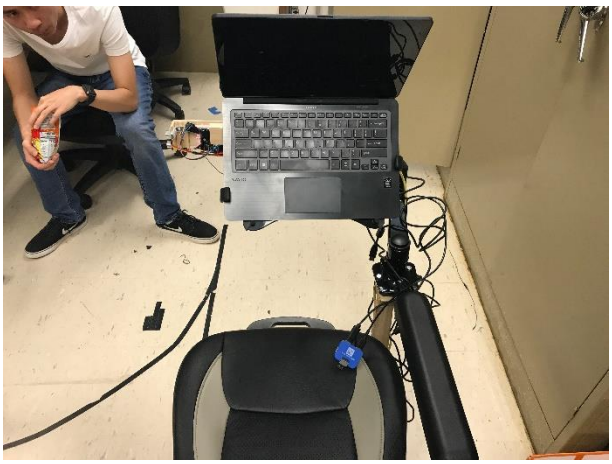


Figure 35. Laptop on laptop mount

APPENDIX E. RISK ACCESSMENT

CONSEQUENCE of FAILURE (Impact)							
Category	Criteria	Definition/Primary Project Objective	1 Minimum	2 Low	3 Medium	4 High	5 Very High
SAFETY	**Public Safety**	Failure of element may damage property or injures public and other people	No damage or No injuries	Minor property damage and/or injuries	Major property damage and/or injuries	Serious property damage and/or injuries	Catastrophic property damage and/or loss of life
	Users Safety	Failure of element may result in injury to users	No injuries	Minor injury	Injury that need medical attention]	Serious Injury, possible Loss of Life	Loss of Life likely
OPERATIONS	**Operational Impact**	Failure of element may impact to meeting the deadlines	No impact	Barely meeting the deadlines	Unable to meet deadline	Unable to meet deadline and effect other element to meet theirs deadlines	Unable to meet deadline and make it impossible for other elements to meet theirs deadlines
LIKELIHOOD of FAILURE (Probability)							
Category	Criteria	Definition/Primary Project Objective	1 Not Likely	2 Low	3 Likely	4 Highly Likely	5 Near Certainly
CHANCE OF FAILURE	**End of Useful Life**	Failure of element is likely within:	10%	30%	50%	70%	90%

Table 5. Scoring Scale

Feature Information			Impact			Impact	Probability	Total
			35%	35%	30%	Score	Score	
System Type	Element	Cost	Public Safety	User Safety	Operational Impact	Consequence of Failure	Likelihood of Failure	Risk Score
Hardware	Motor Controller	$ 130.00	3	3	5	3.60	2.50	9.00
	Motors	$ 400.00	2	3	5	3.25	2.00	6.50
	Max32	$ 50.00	2	2	3	2.30	1.50	3.45
	Arduino	$ 22.00	1	1	3	1.60	1.50	2.40
	Ultrasonic Sensors	$ 4.00	2	2	1	1.70	2.50	4.25
	Altitude Sensor	$ 2.00	1	1	5	2.20	2.00	4.40
	Light Sensor	$ 2.00	1	1	1	1.00	1.50	1.50
	LED Strip	$ 10.00	2	2	1	1.70	2.00	3.40
	Kinect Camera	$ 70.00	4	3	5	3.95	3.00	11.85
	Encoders	$ 56.00	3	3	3	3.00	2.00	6.00
	Battery	$ 83.00	1	1	4	1.90	1.50	2.85
	OLED Display	$ 7.00	1	1	2	1.30	1.00	1.30
Software	ROS	$ -	4	4	5	4.30	3.00	12.90
	Ubuntu OS	$ -	1	1	1	1.00	2.00	2.00
	Arduino IDE	$ -	1	1	1	1.00	1.00	1.00

Table 6. Scoring Table

APPENDIX F. RESUMES

Benson Co

Summary

I am currently in my final semester majoring in Electrical and Electronics Engineering at Sacramento State. I am seeking an entry level position as an electrical engineer. I am a very quick learner and have many hours of hands on experience with electronics outside of class. I also work very well with others in a group environment and usually take initiative.

Core Strengths

- Strategic and Analytical approach to R&D projects
- Mechanical and Assembly Experience
- Experience with Component level troubleshooting
- Experience with many electrical testing equipment
- Experience with C/C++, Python, Verilog, and Java
- Experience with Microcontrollers and use of various sensors
- PSpice, Multisim, Arduino IDE, Visual Studio, and Matlab
- Experience with Printed Circuit Board (PCB) Design

Education

California State University, Sacramento - Major: 3.63 GPA — Graduating Spring 2018

- B.S. Degree, Electrical and Electronics Engineering - Concentration: Analog and Digital Hardware

Professional Experience

Sales Associate - Staples — June 2016 - June 2017

- Assisted customers with questions and finalizing their purchases
- Kept inventory full and shelves stocked
- Organized store and filing required paperwork

IT Student Assistant - CalRecycle — March 2018 - Present

- Troubleshooting computer, network, and software related problems
- Assistance on all programs used by the department
- Maintain and quality assure IT equipment

Affiliations

Power Engineering Society - CSUS — November 2016 - Present

Institute of Electrical and Electronics Engineers - Officer Position — May 2017 - Present

Academic Projects

Arduino Powered Lithium Battery Charger

- Designed, built, and coded a Lithium battery charger that can charge up to a 3s cell setup, with the ability to expand up to 5s. Charger has the ability to detect what type of cell configuration the battery is once connected and will apply the correct voltage to charge the battery. Features also include live voltage monitoring and what kind of battery configuration is currently connected.

LED VU/Volt Meter

- Designed and built a LED VU/Volt meter based off the LM3915 Display Driver. The circuit is purely analog therefore it has a very quick response time. In the volt meter mode, it can measure instantaneous voltage drops due to its analog nature, versus a digital multimeter where it takes time to process the signal first before being displayed. In VU mode, it will display the amplitude of an AC input signal in a needle like fashion in 3db increments. I have since re-made this project using the ATmega328 microcontroller.

Kenneth Nguyen

• •

OBJECTIVE: A career in electronic and electrical engineering

EDUCATION:

In progress: **BS, Electronic and Electrical Engineering** • CSU Sacramento • GPA 3.66 • May 2018

Courses:

Electronic 1&2	Digital Signal Processing	Applied Electromagnetism
Advanced Logic Design	Machine Vision	Modern Communication Systems
Computer Hardware Design	Digital Control System	Signal and Systems

PROJECT EXPERIENCE:

Senior Project Design – Autonomous Smart Wheelchair:
Member of a four-person team that is developing an automated wheelchair that could navigate itself through an indoor environment. The wheelchair run on Ubuntu Mate operating system and use ROS as the main controller for the wheelchair. It takes live feed input from Kinect camera as well as encoder information for localization and navigation. The user will be able to select and estimated current location and final destination; and the wheelchair will do the rest to take them to the desired destination. We worked together with Dr. Jesse Leaman to improve his design of iChair as well as his currently use wheelchair

Beam and Ball Balancing:
Led a two-person team through the design, development and implementation phases of a video image processing and PID control system. The system will take live feed from a web-cam to a Raspberry Pi and do image analysis to detect the ball's current location. The current location will be transmitted to an Arduino, where PID will be implement and control a servo to move the beam and self-correct the ball's position to a desired set-point.

Multi-Drive-Mode RC Car:
Individual project to implement a project involving micro-controller in a short amount of time. The RC car has two separate drive modes: automatic and remote control. The project is programmed on an octa-core micro-processor. In automatic mode, the car will be able to avoid obstacle using ultrasonic sensor. In manual mode, the car will be controlled with a wireless remote. Additional feature: turn on headlight when dark, speaker alert, turning light signal.

KNOWLEDGE AND SKILLS:

Programming Languages: C • C++ • Verilog • Python • VHDL • x86 Assembly

Operating Systems: Windows XP • Windows 7 • Linux • Ubuntu

Software: Visual Studio • Multisim • PSPICE • Pycharm • Matlab • Xilinx ISE • MS Office

Tools: Oscilloscope • Function Generator • Logic Analyzer • Multimeter

Organizational and Communication Skills:

- Strong analytical and problem-solving skills acquired through the completion of hardware and software projects and computer labs.
- Excellent written and oral communication skills developed through lab reports and group presentations.
- Highly organized having managed multiple projects simultaneously.
- Self-motivated and dependable; always complete projects before deadlines.

WORK EXPERIENCE:

Computer/Math Tutor October 2015 – May 201

- Worked with children from elementary to middle school.
- Review material learned in school and learns material for the following day.

Hardware Troubleshooter ***Self-Employed*** May 2014 – Present

ACCOMPLISHMENTS AND AFFILIATION:

- Dean's Honor List
- Power and Energy Society - CSUS
- Institute of Electrical and Electronics Engineers
- Tau Beta Pi California Upsilon- CSUS

Quan Ky Au

OBJECTIVE

Challenging position in Electrical/Electronic Engineering and achieve my best with satisfactory feedback

EDUCATION

Bachelor of Science, Electrical/Electronic Engineering — California State University
Expected May 2018, 3.1 GPA — Sacramento, CA

ACADEMIC PROJECTS

Autonomous Smart Wheelchair (Senior Project)
Designed and built, in a team of four members, an Autonomous Smart Wheelchair to navigate people with disabilities, who cannot use their limbs, safety from point A to point B indoor. The product is consisted of three main features: 2D Mapping, Autonomous Navigation, and Collision Avoidance. It has a Battery Monitor system that would allow user to see how much power the battery has left and a Graphic User Interface that would allow user to control the wheelchair. In addition, the wheelchair will have smaller components like headlights to improve visibility for the user and speakers for alerting.

Camera Speedometer System
Designed and built, in a team of two members, a Camera Speedometer System to measure the speed of a moving object in real time. The whole system will be consisted of one single camera in order to capture images of the moving object. The Camera Speedometer uses machine vision to learn and memorize the object so it manages to recognize and capture it amongst other different objects. A camera will capture the object at an initial time and again at a final time. The system will use this information to calculate the average velocity of the captured object.

Air Cooling System
Designed and built, in a team of two members, an Air Cooling System, using a PID controller, to maintain a desired input temperature level of a confined space. The system is able to lower the rising temperature inside a confined space. By using the existing temperature inside the confined space as the threshold temperature, a disturbance is created that will cause the temperature to rise with the introduction of additional heat. The system is consisted of a microcontroller and a temperature sensor to monitor the changes above the threshold. As the sensor detects changes above the threshold, it will output the information to the microcontroller unit (MCU). The MCU will output a signal to the fan system that will control the speed of the fan motor, ultimately reducing and maintaining the threshold temperature. As the change in temperature approaches the threshold, the new temperature readings will be inputted back into the MCU. The MCU will determine the new signal output to send to the fan and maintain the proper temperature speed approach to the threshold target.

Vending Machine Controller
Designed and laid out a digital controller circuit for a vending machine in a team of two members, using Cadence Virtuoso. This controller keeps track of how much money has been deposited by the customer and dispenses the product once it reaches a certain amount. The controller also outputs a number at all times to indicate how much money has been deposited so far. The machine has three separate sensors for each of the three coins. Once both the product and any change have been dispensed, the controller will be reset to the initial condition.

Mini-Robot
Designed and built a robot in a team of four members, using Parallax Propeller Board. The robot has full functionality in self-driving as well as a manually controlled. Self-driving allows the robot to study the surrounding using the PING sensor. The robot is able to avoid objects and take a different route if necessary. Manually controlled by a remote uses an IR receiver where the robot is controlled by the buttons pressed on the remote. Besides the driving capabilities, the robot will also be able to play music and audio during these processes.

KEY SKILLS

Software & tools: SolidWorks, AutoCAD, Tableau, MATLAB, Multisim, Oscilloscope, Multimeter, Signal Generator, MS Office
Languages: English, Vietnamese, MATLAB, Access SQL, C, Java, Verilog, VHDL

WORK EXPERIENCE

California Department of Water Resources August 2017 – Present
Student Worker/Office Assistant Sacramento, CA

- Assisting Asset Management engineers and analysts in the collection, handling, and presentation of technical engineering information related to assets.
- Assisting with maintenance and enhancement to existing database sources.
- Improving and implementing data collection and entry processes from several engineering sources throughout the State Water Project.
- Reviewing and examining engineering projects in asset management.
- Checking asset classification and grouping to aid in engineering analysis.
- Advancing data management strategies.
- Assisting in geospatial presentation of condition assessment program data.
- Supporting engineering project managers with current information including scope, schedule and budget tracking and reporting.
- Monitoring correspondence, researching Project Manager inquires, and attending meetings regarding various engineering studies and/or planned construction activities.
- Preparing dimensional drawings, installation drawings and more complex drawings related to electrical products
- Revising and updating existing drawings reflecting design changes to ensure consistent documentation of the project.
- Reviewing blueprints, plans, specifications and other documentation
- Writing White Paper

California Department of Social Services June 2014 – July 2017
Student Worker/Office Assistant Sacramento, CA

- Audited County Expense Claims (CEC) which includes independently reviewed, analyzed, and solved problems relating to the claim expenditures to ensure accuracy of the claims
- Prepared claim letters, filing documents, and provided quality assurance
- Prepared spreadsheets and templates
- Responded to inquiries from the counties related to CEC.

ACTIVITIES & ACCOMPLISHMENTS

Vietnamese Student Association 2010 – present
Member/Volunteer Sacramento, CA

- Received a recognition award from the Senate Republican Leader, Jean Fuller, California Senate 16th District
- Fundraising for the annually Vietnamese American Scholastic Achievement Awards Program
- Fundraising for the December 2016 Central and South-Central Vietnam Flood
- Fundraising for the September 2015 Valley Fire in southern Lake County
- Fundraising for the April 2015 Nepal Earthquake (Gorkha Earthquake)
- Fundraising for the March 2011 Japan Earthquake and Tsunami

Music Entertainment Service 2012 – present
DJ/Producer Sacramento, CA

- DJ-ing for music events, weddings, parties, etc.
- Recording and producing music

Timothy Mulvey

SKILLS:

- **Software Engineer** - Formally and self educated
- **Languages**: C++, Java, C, Verilog, Assembly x86, Python, JavaScript, Powershell, Bash, PHP, C#, ASP
- **Tools**: Visual Studio, Eclipse, jGrasp, Git, Vim

EDUCATION:

Bachelor of Science, Computer Engineering

California State University, Sacramento	May 2018 Expected Graduation
Tau Beta Pi Engineering Honor Society	CSUS 3.52 GPA, 3.37 Cumulative

Associate of Arts: University Studies - Engineering

Shasta College	May 2015
Graduated with Honors	Phi Theta Kappa Honor Society

Related Courses:

Introduction to Computer Architecture	Java Programming
Web Design using Dreamweaver	Intro System Program Unix
Data Structures and Algorithm Analysis	C++ Programming
Programming Concepts and Methodology II	

Independent Study:

Algorithms (JavaScript) - khanacademy.org
Artificial Intelligence (Python) - courses.edx.org

PROJECTS:

Virtual Micromouse - C++ - https://github.com/TimothyCM/Virtual-Micro-Mouse

- Built a virtual version of the micromouse competition
- Used skills learned in Algorithms to generate random mazes
- Applied techniques from edX course in Artificial Intelligence to solve the maze
- Extensive test code and debugging

Remote Controlled Vehicle - C - https://youtu.be/yDlgdmrFdNw

- Used multicore programming techniques on a Propeller Microcontroller
- Wrote code for motor control and ultrasonic sensors to detect objects and prevent collision
- Worked as a group to complete the project, present it, and prepare the report resulting in an A