Software Design Document

1/24/2018

Version 1.0

Keystone Robotics
Robot Assisted Tours

Project Sponsor:

Michael Leverington

Faculty Mentor:

Austin Sanders

Team Members:

Hailey Ginther, Shannon Washburn, Gabrielle Halopka, Falon Ortega

Table of Contents

1.0 - Introduction

Northern Arizona University's Engineering building is the most important stop of campus tours for future engineering and computer science students. The labs, project rooms, and lecture halls of the building are where they will be studying for the next 4 or more years. This means the impression groups get of the facilities during their time on a tour of the building is an essential contributing factor to attracting and retaining new students. Attracting the attention of these students with physical evidence of the work accomplished by seniors in the department will help persuade these new individuals to enroll.

Our client, Dr. Michael Leverington, is a professor at Northern Arizona University's School of Informatics and Computing. The professor has a Ph.D. in Education, Masters of Computer Science and Psychology, and a Bachelors in Physics. Additionally, he is an avid follower of developments in robotics and is eager to bring the multidisciplinary topic to his department. He has tasked our team with the initial planning, assembly, and programming of a robot potentially capable of giving tours of NAU's engineering.

The ultimate purpose of this project is to create a robot capable of autonomously giving tours of the engineering building, fulfilling the need for a captivating introduction to the projects NAU students can accomplish thanks to or in conjunction with their coursework. Our goal as the starting force behind this project is to put what we have identified as the basic level of hardware and software this robot needs into practice.

For the scope of this capstone project, the client's goal is to have a fully assembled robot that can be moved with user inputs. Other major goals during this project for this team are to perform tests that ensure the safety and stability of hardware and to leave behind detailed documentation for future students' use during follow-up projects. The basic system our team has outlined to solve our problems involves using microcontrollers that send signals to motors through motor controller circuits. The design must meet certain requirements as delineated by our client, such as the ability to handle its own weight of at least 100 pounds and be driven by high-voltage batteries. Additionally, this robot must be able to construct a map of a given space and be able to successfully navigate any controlled area with user input, with the stretch goal of the project being autonomous navigation. Autonomous navigation would include the robots ability to navigate a section/floor of the building with just the destination given to it, so no user input would be necessary.

The requirements are based on the goal that the team will be able to manually control the robot. The requirements take into account who the end user is, environmental factors, and safety. The basic requirements include: the robot will be capable of basic navigation, the robot will be expandable to future projects, the robot will operate safely, and the robot will be usable for a technical end user. In order to meet safety and navigation requirements the robot will be able to navigate a path between two points on a single floor, be able to avoid hazards, be manually controllable, and will have an emergency stop mechanism which will disconnect the motors from the power supply. Some other technical requirements for the robot are it's velocity will be at least 1.5 m/s while moving at least 100 lbs, it will be able to operate continuously for at least 2 hours. The environmental requirements for this project are budget, scale, housing, and location. For budget, the parts must be cost effective. For scale, the project is defined as a large-scale robot or about four to four and a half feet tall. The components for the robot must be placed in a 30-gallon barrel. For location, the robot only needs to operate indoors.

2.0 - Implementation Overview

As previously stated, there are two components to this solution: the hardware of the robot and the software that will control it. The hardware solution includes the physical components of the robot as well as the circuitry that powers it and aids in the processing of information. The software includes the programs and libraries used by the microcontrollers that facilitate the hardware functions. For the minimum viable product of this the project's robot, the solution will be a combination of integrated hardware and software components

2.1 - Hardware Solution Overview

One part of the hardware design will be the physical robot which includes the barrel, motors, wheels, and frame. The robot's build is focused around stability, ease of access, and safety. To ensure stability a custom steel mount is used to hold the two wheelchair motors in place. Removable shelving is placed inside the barrel to make access easier. The frame is secured with machine and wood screws and structurally tested for safety.

The other part of the hardware design is the the electrical portion of the build. The power circuit must be able to turn the device on and power it efficiently for a few hours at a minimum. When the device's charging line is plugged into a wall and/or an off switch is pulled, the device must with a switch to a charging circuit and all power to the motors should be cut off. The electrical and physical designs will allow the robot to move with the help of the software components.

2.3 - Software Solution Overview

The software should be able to take input and make decisions for the robot based on that input. The software will use a series of sensors in order to collect the necessary data and connect to the motors in order to control them. Message passing between the Raspberry Pi and Arduino Uno is handled by ROS. A suite of ROS packages are used to translate and send sensor data and give the motor instructions. These motor instructions are generated via two methods: the navigation stack for autonomous movement, and the joy package for manual control.

3.0 - Architectural Overview

In the overall architecture of the project, both software and hardware solutions must work together to in order to create a fully functional robot. A more detailed outline of the hardware, software, and overall solution is discussed in this section.
The block diagram of the entire system is shown below.

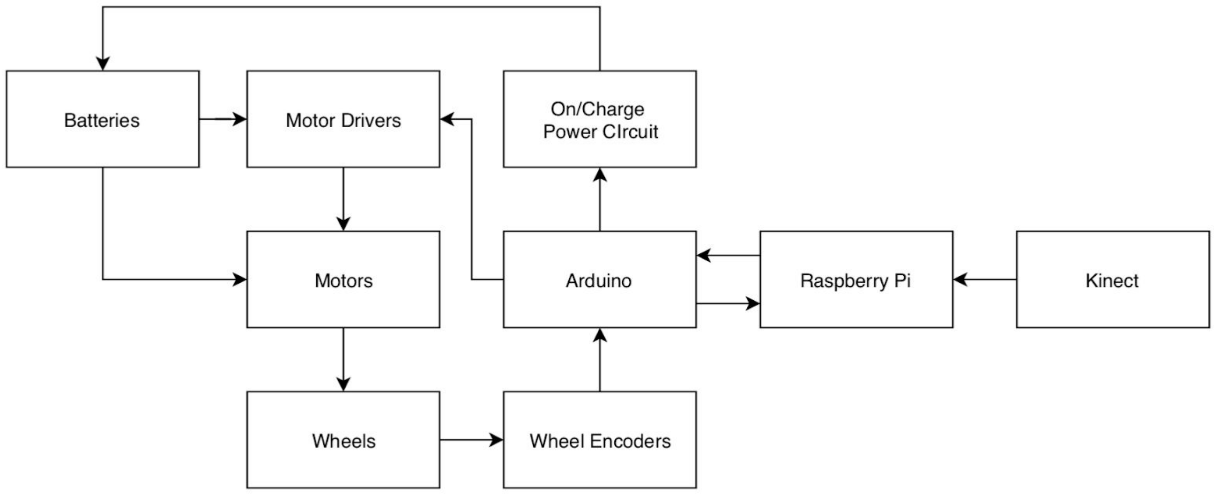


Figure 1: An architectural diagram of the entire system

The Raspberry Pi in the system serves as the 'brain' it takes inputs from the Arduino and other sensors in order to determine how the robot will move.

3.1 - Architecture: Hardware

The hardware components of this system are the batteries, motors, wheels, base, barrel, and other circuitry. Some of the circuitry in the system will be connected to the Arduino and Raspberry Pi, where the software design will be done. The wheels and motors together are a purely physical system and connect directly to each other, and then interface into the electrical system. The on/charge circuit (also known as the power circuit) acts as a switch that either connects the batteries to the motors or flips to a different circuit where the batteries will be recharged. In the on or 'high' position, the on/charge circuit will connect the batteries to the motors so that the robot will be able to move. In the off or 'low' position, the batteries will be cut off from the main system and switched to a charging circuit, so no power will reach the motors themselves. The on/charge circuit will 'check' if there is a signal from the Arduino, or if the robot is on, to see if it is safe to connect the batteries to the main circuit. A block diagram of the system is shown in Figure 2.

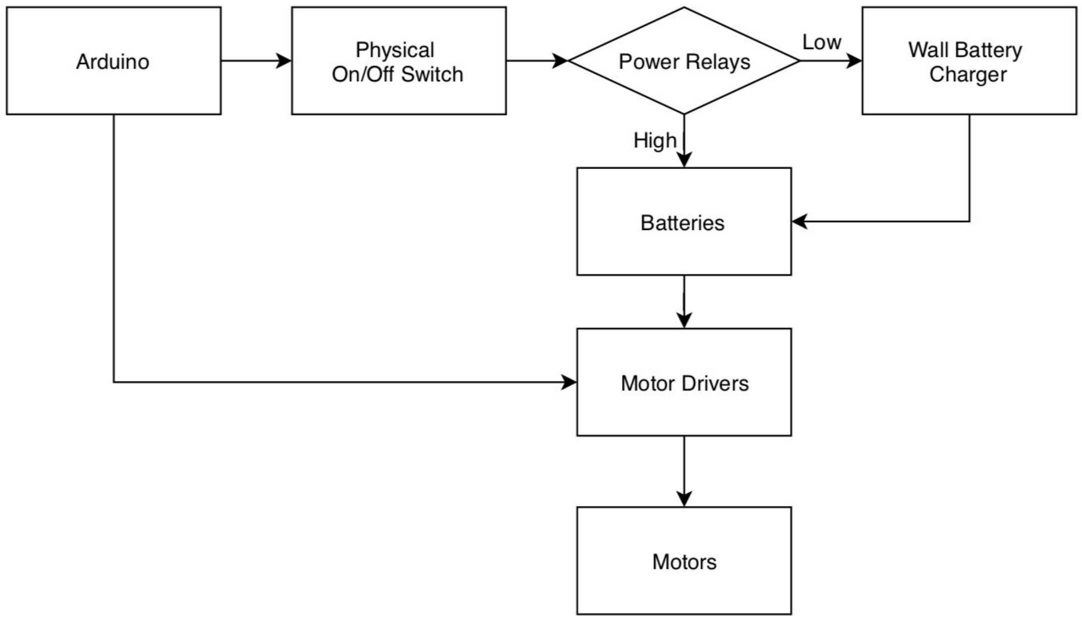


Figure 2: Block diagram of hardware components

3.2 - Architecture: Software

The software piece of this project focuses on the software running on a Raspberry Pi and Arduino Uno. These microcontrollers take sensor-dependant inputs from sensors placed on the motors and exterior of the robot's barrel, interfacing via ROS (Robot Operating system). Encoders, the sensors placed on the motors, report the RPM to the Arduino, which in turn reports the information to the Raspberry Pi. A Microsoft Kinect mounted to the top of the barrel sends laser scan data to the Pi as well. The Pi can then make use of the data received from these two sources in tandem with a set of pre-recorded map data to facilitate path-making decisions. These paths are then transformed into velocity commands and sent to the Arduino, which interprets the commands and changes the speed of the robot's two motors accordingly. User control is accomplished via a USB controller that connects to the Raspberry Pi and sends motor control messages to the Arduino via a set of ROS joystick libraries.

4.0 - Module and Interface Descriptions

This section covers the different modules in more detail and how they work on an individual basis. The section has been divided into hardware, circuitry, and software subsections. Physical construction covers the physical construction of the robot's base and motor mounts as well as the barrel's mounting and interior shelving structure for holding the components described in the other two subsections. The hardware describes the electrical system contained within the barrel of the robot and is responsible for powering the motors. Finally, the software describes the system facilitated by ROS that runs on a set of microcontrollers that receive sensor input and send messages between the different components.

4.1 - Physical Construction

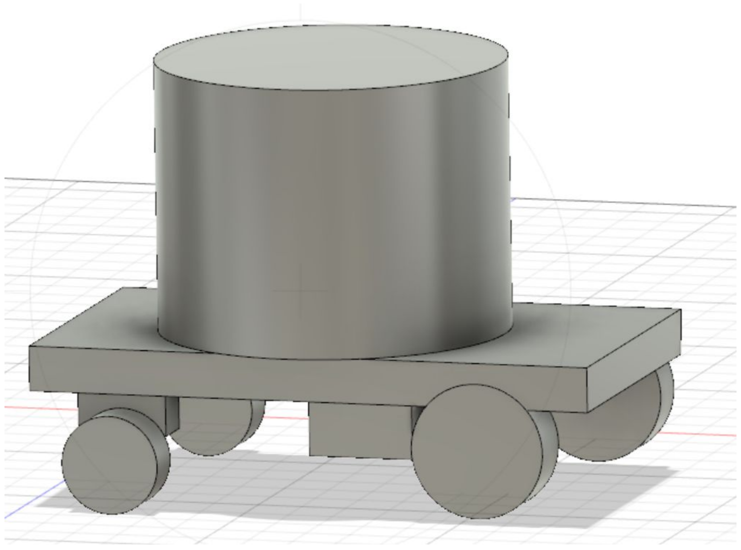


Figure 3: A rough diagram of the physical robot

The base of the robot is derived from a wooden moving dolly which was partially dismantled. A custom-ordered steel mount is attached with mechanical screws at one end of the dolly, replacing the dolly's original back caster wheels. The motors are secured to this mount via 6 bolts, 3 on each side. The back wheels are 13" in diameter and fit firmly onto the shafts of the motors. At the front end of the base are two of the moving dolly's original caster wheels which are attached to the main body of the dolly via two 7" lengths of 4" x 4"'s. The wooden frame provides stability and support for the motor and wheels, and the use of two caster wheels with two motor-powered wheels allows for rear-wheel differential driving. An image of the base of the robot is shown in figure 4.

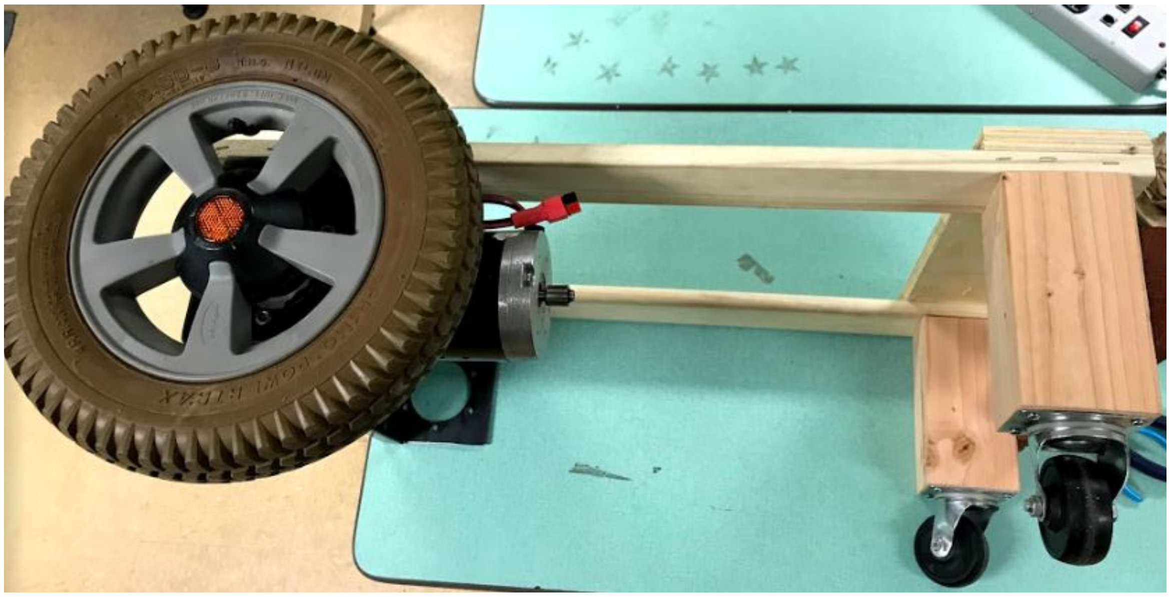


Figure 4: An image of the semi-completed base

4.2 - Hardware

The schematic below depicts the on/charge circuit.

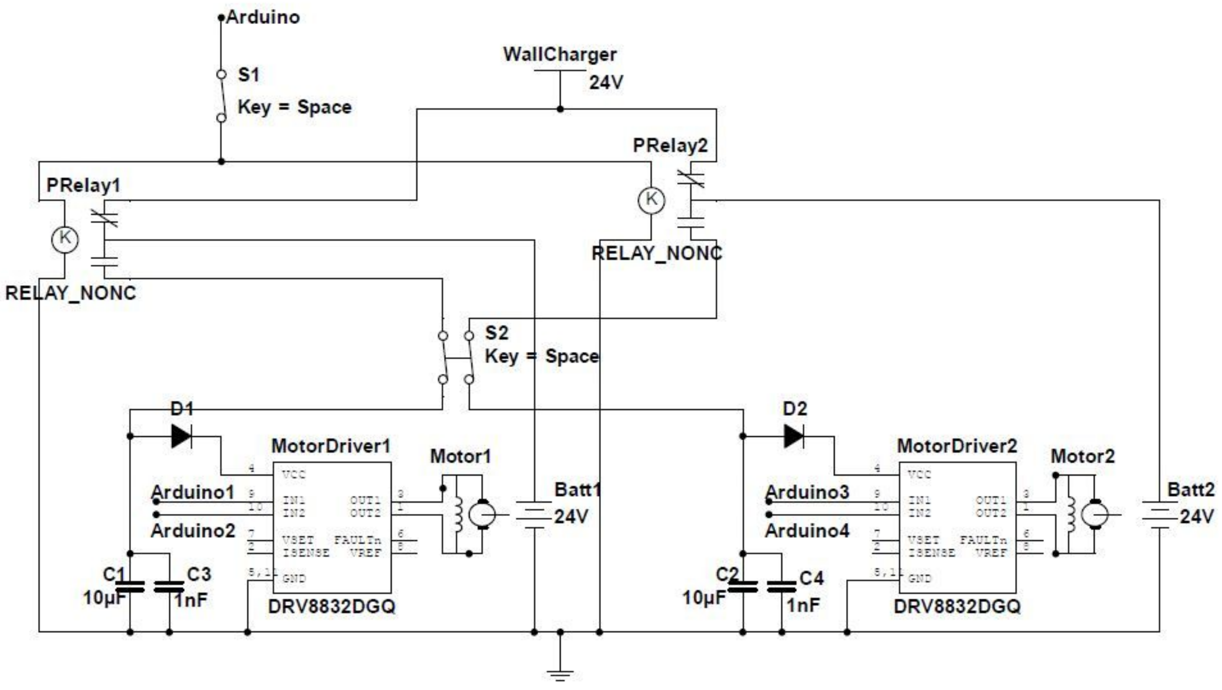


Figure 5: The schematic for the on/charge circuit

There are two relays, which connect to both the wall charger and the motor driver circuits. The relays chosen are single pull, double throw. When a signal is received on the ‘pull line’, the coils on the other side of the relay move and flip the position of two switches. Different relays can have the internal switches default to positions so that both switches open or close on a high signal. For the purposes of switching between the charging circuit, a relay that has one open and one close when a high signal is sent on the pull line makes the most sense, to cut the two circuits off from each other. The signal line on the relays comes from the Arduino. If the device is on and the switch is flipped to on, then the batteries can send power to the motors and then the wheels may spin. When the switch is flipped to off, the Arduino can’t send a signal to the relays. This switch's primary purpose is a physical switch the client can press. In the on position, the top lines of the relays open and disconnect the wall charger, and the bottom lines close, connecting the batteries to the motors.

The capacitors are used to ‘clean up’ the signal coming from the battery to the motor driver. It is standard practice to put capacitors on the ground end of a power source to prevent noise from affecting any of the logic or processing in the circuit. The diode will prevent any backward current flow into the motor driver, which could damage the board and any electrical components on it. The motor drivers themselves will take in signals from the Arduino, on the IN1 and IN2 ports, that can change the direction the motors spin so that the robot can move backwards without having to turn around. When the batteries are switched to the charging circuit, the relay cuts the connection to the main circuit and connects to the wall charger. This way, the robot can be charged without having high or backward current passing through the entire system. Finally, the entire system will share one large ground connection at the bottom of the barrel. Heat sinks will be attached to make sure nothing gets too hot and smokes or melts. By sharing a ground, we can prevent having any floating grounds or large differences between grounds, which would negatively affect our circuit’s performance.

4.3 - Software

The diagram below shows the basic flow for autonomous navigation using ROS.

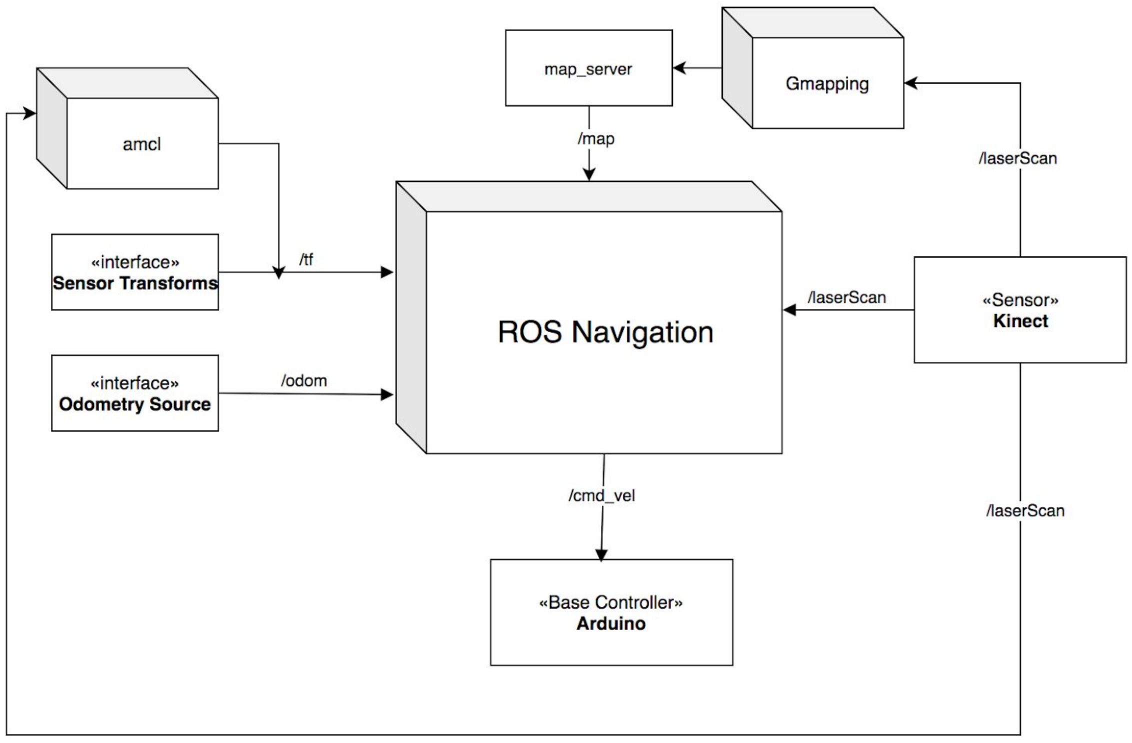


Figure 6: Basic Autonomous Navigation Flow

The first step for autonomous navigation is to create a map of the area in which the robot will be navigating. For this, ROS provides a package called *gmapping* which uses the laser scan data which we will be providing using the Microsoft Kinect. This package uses the SLAM (Simultaneous Localization and Mapping) algorithm as a ROS node to create a 2D occupancy grid. This map data can then be used by the *map_server* package which will publish the map data as a ROS topic called /map. After the map data has been created and published to the system, the robot then needs to be able to localize itself with a given space. The *amcl* package provided by ROS uses the map and laser scan data and applies the adaptive Monte Carlo localization algorithm to calculate an array of pose estimates. As the robot moves throughout the space, the estimates become more accurate until the most accurate pose is determined. For the ROS navigation stack to function, we will need to publish the transform data for the sensors using the *tf* package. This transform data includes where the sensors such as the laser scanner are located in relation to the base. Lastly, the calculated odometry of our robot needs to be published to the /odom topic. Once these components are in place, we will need a way for the velocity commands calculated by the navigation stack to be interpreted by the microcontroller attached to the motors. In order for the robot to be able to move at accurate velocities, it needs a base controller where to motor speeds can be monitored, set, and published to the system. Our base controller is in the form of an Arduino program which can communicate with the ROS system running on the Raspberry Pi. The motor speeds will be calculated using optical wheel encoders attached to each motor which send signals to the Arduino when they are running. These signals must be interpreted to calculate the velocity of each motor. Once an accurate velocity is calculated, we will use a PID (proportional-integral-differential) control loop to adjust the speed of the motors to the desired speed.

The next diagram shows the basic flow diagram for manual control.

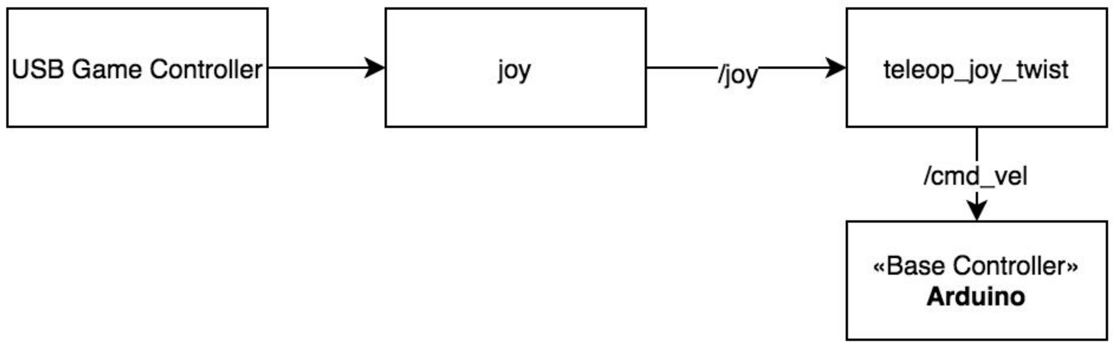


Figure 7: Manual Control Flow

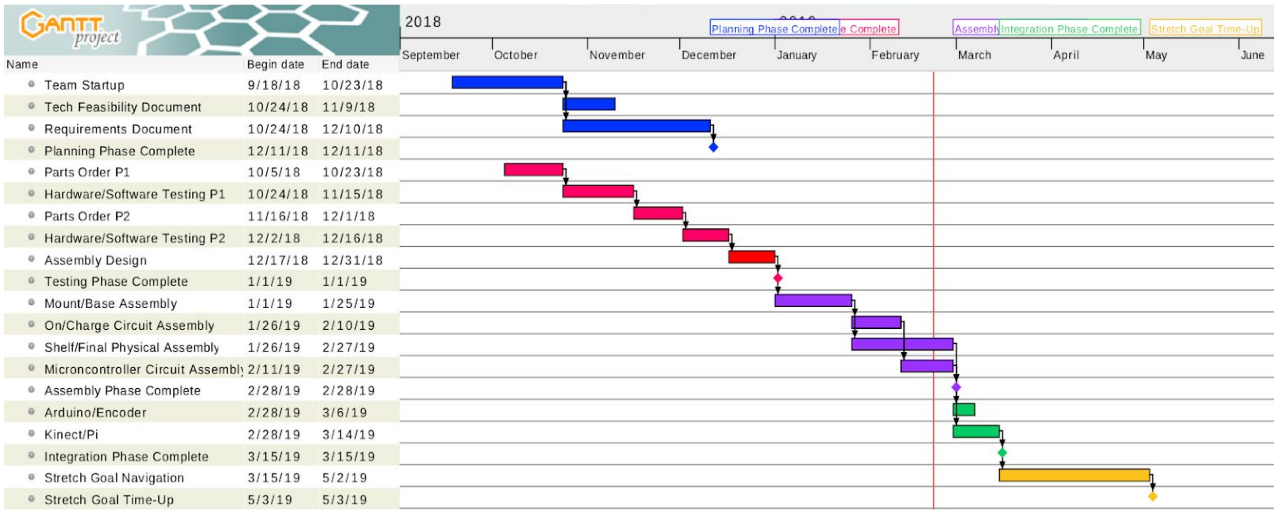
The joy node provided by ROS publishes USB joystick data on the /joy topic. Then the teleop_joy_twist node converts the joystick data into a twist message on the /cmd_vel topic. Our base controller will be subscribed to that topic and use the data to control the motors.

5.0 - Implementation Plan

The following section presents our most current design schedule. The Gantt chart provides a general outline of each major milestone in our development process. Section 5.2 presents a table breaking down the distribution of each task between members of the team.

5.1 - Gantt Chart

The chart is broken into five phases: planning, testing, assembly, integration, and stretch-goal implementation. Each phase is represented by its respective color, and the phase milestone is represented by the milestone of the same color. The red, vertical line represents the current point the team is at in the project. The previous Gantt chart was able to be broken down into more finite parts once the assembly phase began because the team could create more realistic goals and timelines for each portion of the project as the work was delegated to specific team members.



Name	Begin date	End date
Team Startup	9/18/18	10/23/18
Tech Feasibility Document	10/24/18	11/9/18
Requirements Document	10/24/18	12/10/18
Planning Phase Complete	12/11/18	12/11/18
Parts Order P1	10/5/18	10/23/18
Hardware/Software Testing P1	10/24/18	11/15/18
Parts Order P2	11/16/18	12/1/18
Hardware/Software Testing P2	12/2/18	12/16/18
Assembly Design	12/17/18	12/31/18
Testing Phase Complete	1/1/19	1/1/19
Mount/Base Assembly	1/1/19	1/25/19
On/Charge Circuit Assembly	1/26/19	2/10/19
Shelf/Final Physical Assembly	1/26/19	2/27/19
Microncontroller Circuit Assembl	2/11/19	2/27/19
Assembly Phase Complete	2/28/19	2/28/19
Arduino/Encoder	2/28/19	3/6/19
Kinect/Pi	2/28/19	3/14/19
Integration Phase Complete	3/15/19	3/15/19
Stretch Goal Navigation	3/15/19	5/2/19
Stretch Goal Time-Up	5/3/19	5/3/19

The planning phase consisted mainly of figuring what parts would be needed for the build as well as planning and sketching designs, this is shown as the blue part of the graph. Testing occurs in the red portion where each individual piece is tested. This includes testing the individual parts, like wheels and motors, to more complex systems such as the entire stop/ charge circuit. Assembly phase is where individual components are constructed. This is when the base of the robot and circuitry is constructed as well as completing a majority of the coding is completed for independent systems. Integration occurs when the whole system is put together to form a semi-complete robot. The stretch goal portion is for any stretch goals of the project.

6.0 - Conclusion

Tours of the Engineering building are time-consuming and largely uninformative. Professors must take time out of their day to assist with these tours, and the students who give them are often not engineering students. Our client, a computer science professor with a physics and engineering background, wants a device that can take over this role. By automating this process with a robot-assisted tour, faculty can save time, and visitors to the building can be shown the capability of NAU's engineering and computer science students.

The "30-gallon robot" consists of a software 'brain' made of a Raspberry Pi and possibly several Arduino. The 'body' of the robot is made of two large battery-motor pairs, a barrel, wheels, and a base to hold it all. Together, these components work together to create a functional robot. The scope of this project is limited to building a solid and expandable base for other students to work on for future capstones, where the minimum viable product will be a robot that must be able to navigate a floor of the engineering building successfully at walking speed. This navigation will definitely be able to be done based on user input, or if spare time is had, the team may implement autonomous navigation modules. Other requirements our client had were that the specifications and design process for this robot should be well documented and the final product must be usable by a computer science professor. This means the design must be expandable for future use by other capstone teams and students to contribute to the project's end goal and to be used by our client at the end of this capstone project.