Worcester Polytechnic Institute

A Major Qualifying Project

WALRUS Rover

Water and Land Remote Unmanned Search Rover

Submitted By:
Brian Eccles, Computer Science
Brendan McLeod, Robotics and Mechanical Engineering
Tim Murcko, Robotics and Electrical & Computer Engineering
TJ Watson, Robotics and Mechanical Engineering
Mitchell Wills, Robotics Engineering and Computer Science

Advised By:
Taşkin Padir, Assistant Professor
Robotics Engineering, Electrical & Computer Engineering
Kenneth Stafford, Associate Teaching Professor
Robotics Engineering, Mechanical Engineering
Sonia Chernova, Assistant Professor
Robotics Engineering, Computer Science

Abstract

Search and discovery rovers have evolved to play an essential part in disaster relief and humanitarian aid. With features such as supervised autonomy, high-definition vision systems, and enhanced mobility, the goal of this project is to create an amphibious rover to aid in the search and discovery of persons during disasters. Such a rover would not only give relief teams more information about the situation at hand, but also eliminate the danger of sending search parties into harsh and high-risk environments. The addition of water mobility allows teams to deploy the Water and Land Remote Unmanned Search (WALRUS) Rover after disasters such as tsunamis or hurricanes. A rover with amphibious capabilities exceeds the maneuverability of current commercial platforms and enables a wider range of mission profiles. The end result of this project is a fully functional prototype of an amphibious rover guided by our design requirements. The WALRUS Rover floats in water, can drive up stairs and over large debris, and supports accessory payloads. This document outlines the project requirements and design work done to build and test the WALRUS Rover.

Acknowledgments

In addition to our project advisers we would like to thank David Cyganski, Dean of Engineering; Karen Oates, Dean of Arts and Sciences; and Arthur Heinricher, Dean of Undergraduate Studies as well as Velin Dimitrov, Tracey Coetzee, Joseph St. Germain, Jon Anderson, Michael Fagan, Matt Brekken, and Andrew Bennett for their advice and support throughout this project. We would also like to thank our sponsors, for without their donations this project would not be possible: WPI Athletics Department, Electro-Mechanical Technology Company, SolidWorks, Bren-Tronics, Inc., Ubiquiti Networks, Vicor, Anderson Power Products, RIVeR Lab, Maxon Motor, BRECOflex CO., L.L.C., Action Bearing, APEM, Barnstorm Cycles, Hydro-Cutter, Digi-Key, Pololu, Molex, Boston Gear, Igus, and Intel.

Contents

List of Figures

List of Tables

1 Introduction

According to the Centre for Research on the Epidemiology of Disasters (CRED), over one million people were killed in natural and technological disasters worldwide since 2000 [2], and countless others were seriously injured. This number does not include acts of terrorism such as 9/11. The dangers of sending humans into the hazardous or unknown environments created by such disasters make teleoperated and semi-autonomous robotics a growing field. Commercial robotic platforms like the iRobot PackBot and the QinetiQ Talon are used in a variety of mission scenarios including search and rescue, reconnaissance, and simple remote environment manipulation. There are robots that can be tossed into buildings to act as scouts [3], pull out injured persons [4], and disable bombs [5].

Search and discovery rovers have evolved to play an essential part in law enforcement and disaster relief efforts. With features such as supervised autonomy, high-definition vision systems, and increased mobility, this project aimed to create an expandable amphibious rover to aid in the search and discovery of persons in distress at a competitive cost. Such a rover would not only give relief teams more information about the situation at hand, but also eliminate the danger of sending search parties into harsh and high-risk environments. The addition of water mobility allows teams to deploy the Water and Land Remote Unmanned Search (WALRUS) Rover after disasters such as tsunamis or hurricanes. Land-only rovers commercially available today are not capable of these missions. A rover with amphibious capabilities exceeds the maneuverability of current commercial platforms and enables a wider range of mission profiles without a significant increase in cost. In many cases, there may be victims isolated inside homes or buildings with no method of contacting a rescue team. Typical search methods, such as helicopter teams, may not be able to see an individual in need of assistance in these scenarios.

One class of common use cases for rovers involves indoor exploration and manipulation of the environment. No matter where you are, being able to negotiate common indoor obstacles is essential. The interior of homes was designed to make life easier for human inhabitants, not robots. Robots like Boston Dynamics' Atlas [6] show promise for investigation of human environments, but are still in development. In the mean time, robots are needed that can easily overcome obstacles like stairs and tight hallways. Most of the rovers available on the market today have poor vision systems and do not provide enough situational awareness for the driver to discern their location. A more sophisticated rover would keep track of its location and be capable of retro-traversal.

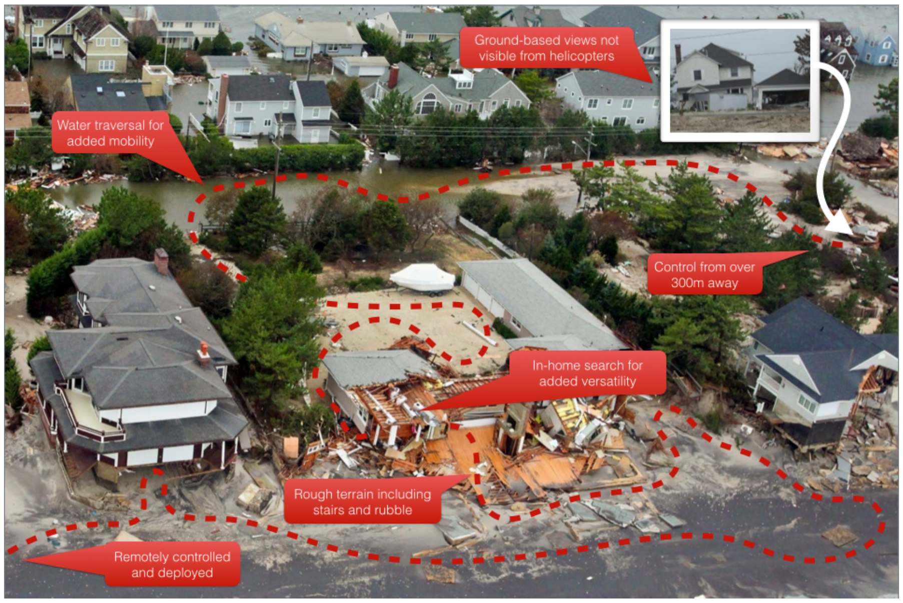

Ground-based views not visible from helicopters
Water traversal for added mobility
Control from over 300m away
In-home search for added versatility
Rough terrain including stairs and rubble
Remotely controlled and deployed

Figure 1: WALRUS Rover Example Mission.[7][8]

Figure 1 shows an example mission scenario the WALRUS Rover is capable of completing. This image shows the aftermath of Hurricane Katrina and portrays a variety of challenging terrains including deep water, large rubble, and indoor navigation. The WALRUS Rover is able to traverse all of these obstacles from a distance using the rover's situational awareness and communication systems.

Though this project addresses the primary use case of the WALRUS Rover for search, discovery and reconnaissance, the rover is designed as an expandable mobile platform. Additional sensors, manipulators, or other actuators can be mounted to the WALRUS Rover's open source platform for the successful completion of different missions. This opens up doors to new tasks the WALRUS Rover could be capable of completing. An example includes responding to infectious diseases such as the Ebola outbreaks to perform tasks including decontamination in rural areas. It can be very difficult for medical workers to avoid contamination even with proper safety protocols. Remote operation of the WALRUS Rover with the addition of a bleach sprayer for decontamination and a robotic arm to deliver medicine and supplies to patients would make it much safer for medical workers to aid those in need.

2 Motivation and Background

This chapter is dedicated to reviewing past work done in search and rescue/discovery robotics as well as the need for such systems. We will begin by looking at common mission scenarios where these type of robots are useful and then focus on a discussion of existing rovers with similar capabilities to the WALRUS Rover.

2.1 Mission Scenarios

Search and rescue robots have been in use since the September 11, 2001 attacks on the World Trade Center where rescue teams deployed them to search for victims, identify hazardous materials, and provide human crews with advanced knowledge of the dangerous environment [9]. Among the robots used were the QinetiQ Talon, the Foster-Miller Solem and the iRobot PackBot. Teams deployed these robots to find voids in the rubble and gain access to basement levels in search of survivors.

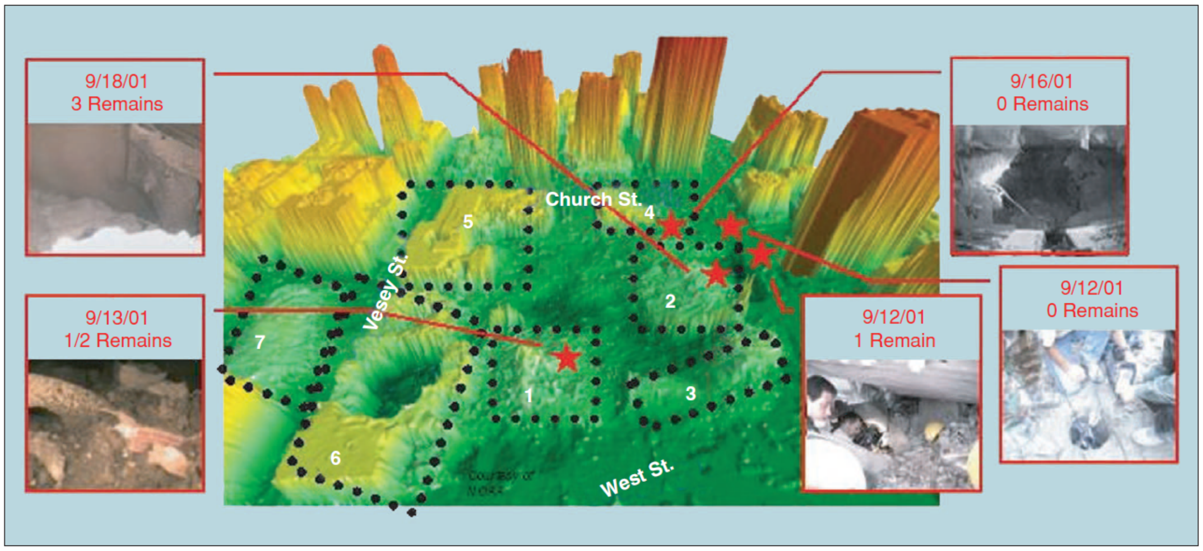


Figure 2: Location of robot deployments in and around Ground Zero[9]

From this disaster, rescue teams learned that robots could be instrumental in safely finding survivors in collapsed buildings [9]. This section outlines the various mission scenarios these robots are designed for and the advantages the WALRUS Rover brings to each.

2.1.1 Hurricanes and Tsunamis

Rescue robots were deployed after hurricanes and tsunamis including hurricanes Katrina, Ike, and Sandy; and the 2011 tsunami in Japan. These situations presented an additional challenge for rescue teams because of pervasive flooding. During these disasters, teams had to deploy underwater search robots to find remains and inspect damage to underwater structures [10]. The WALRUS Rover's amphibious capabilities provide a platform for search and discovery in flooded areas not currently available on the market. The WALRUS Rover has the ability to traverse areas filled with contaminated flood waters and potentially dangerous debris as well as search damaged buildings for trapped survivors. Tasks that were previously difficult to accomplish with pure land and air robots

2.1.2 Geophysical Disasters

Robots have been sent in after geophysical disasters such as the 2011 Tohoku earthquake when the Quince rover was deployed [11]. Geophysical disasters present a difficult land mobility challenge because the rubble piled up after these disasters is usually very large and tough to move over. In addition, a well developed situational awareness system becomes more important as camera views may be obstructed by large debris. Robots that can walk and climb like humans usually would perform much better than treaded or wheeled robots, however these robots are still very much under development because they are significantly more complicated. Therefore, treaded robots like the PackBot are usually sent into these scenarios. With only two flippers and a simple situational awareness system however, the PackBot's mobility can be seriously hindered. The WALRUS Rover overcomes this limitation with advanced mobility features and a more sophisticated situational awareness system, allowing for a larger set of mission profiles.

2.1.3 Medical Epidemics

Medical epidemics such as the current Ebola crisis are a huge problem in rural areas because of poor infrastructure and a lack of medical personnel and hospitals. Usually medical workers are sent to these areas and stationed there for an extended period of time to aid those who become infected. This can be very dangerous for the workers because if they become infected they too have limited access to quality medical care. Even when following safety protocols, workers can become contaminated when removing their gear if not careful. While there are limited examples of robots that have been sent on these missions, this continues to be a growing problem, which is why many Universities like WPI are beginning extensive research and development in this area [12]. One actual example is the deployment of Vecna's VGo telepresence robot to West Africa to help treat people infected with Ebola [13]. Robots are sometimes used in these scenarios to assist the medical workers, but are completely limited to the payloads a company has developed for their particular robot, or are designed in a very focused manner. The WALRUS Rover bridges this gap by allowing complete open source expandability. If a company has a need such as remote medical assistance, they can develop a payload specifically for that need. This expandability makes the WALRUS Rover a single solution to almost any mission scenario - a feature limited to very few commercially available robots.

2.2 Current Solutions

The design of the WALRUS Rover follows from a careful review of current commercially available search and rescue robots. This will give us motivation for improvements for robots in this field to allow our rover to compete with these existing platforms. Two of the leading companies in this field are iRobot and QinetiQ. Therefore the iRobot Packbot and Kobra, and the QinetiQ Talon will be discussed.

2.2.1 PackBot

Arguably the most popular robot for search, rescue and reconnaissance is the 510 PackBot from iRobot. The PackBot was not only sent into Ground Zero during the 9/11 attacks but was also the first robot to enter the Fukushima nuclear disaster zone [14]. Weighing in at 30 lbs including batteries, the Packbot is designed to be one soldier portable and rugged [5] for mainly military applications. The PackBot is a relatively small waterproof treaded rover that has two front flippers for added mobility over challenging terrain. PackBot can operate for up to four hours in ideal conditions, drive up to 5.8 mph, and traverse grades up to 60°. It should be noted that the PackBot must use its arm to change its center of gravity (CoG) to align itself so that it can maintain adequate traction when ascending steep grades. This is shown in Figure 3 as a PackBot attempts to climb stairs.

Additionally, the PackBot can accommodate flippers on the rear of the robot. This allows the CoG to remain within the bounds of the flippers to prevent the robot from tipping [16]. Often times, an incline of about 35° (common staircase) causes the robot to loose traction. To overcome this, PackBot has ridges on its treads that allow the robot to hook onto stairs and adds a force to support it. Although very simple, a treaded climber needs to pay close attention to its CoG when climbing to ensure that proper forces are applied.

As seen in Figure 4, PackBot is capable of actuating its flippers to better achieve optimal positioning of the robot. Turning the flippers down allows the chassis to be elevated.

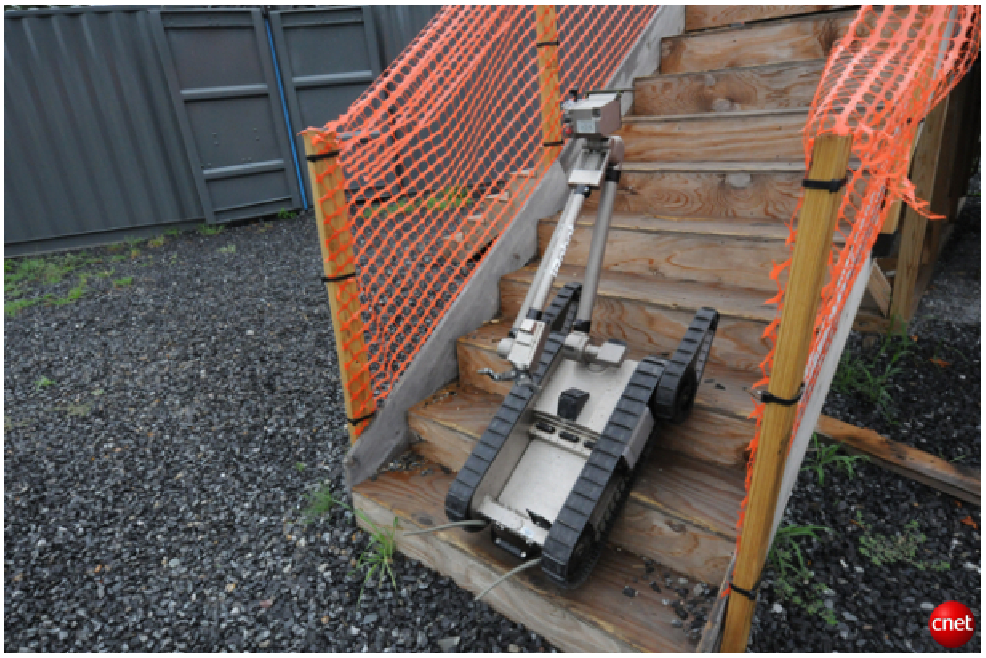
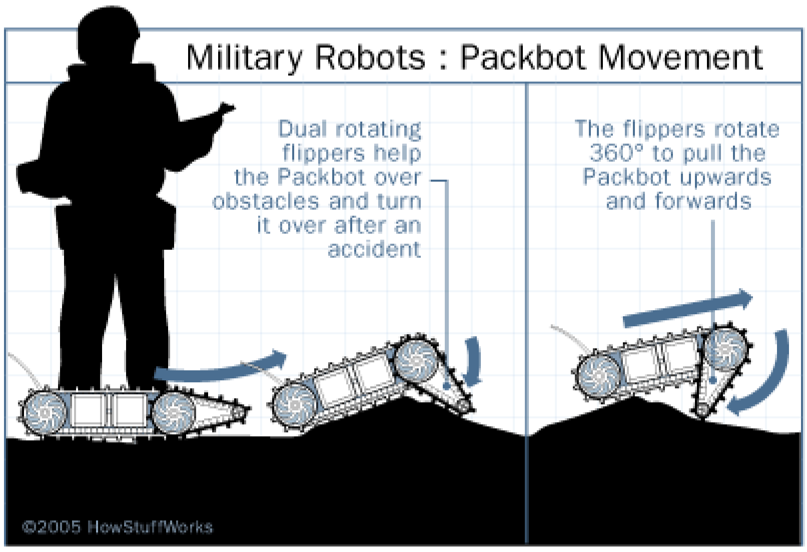


Figure 3: PackBot using its arm forward to change its CoG.[15]


Military Robots : Packbot Movement
Dual rotating flippers help the Packbot over obstacles and turn it over after an accident
The flippers rotate 360° to pull the Packbot upwards and forwards
©2005 HowStuffWorks
Figure 4: PackBot flipper positions.[17]

Flippers with treads also help with stair climbing by allowing the robot to contact the stairs at more points. Additionally, the use of flippers helps the robot right itself [17].

For communications, PackBot has a range of about 800 m line of sight (LOS) [5]. For increased range a mesh network can be setup. Mesh networking is a communications topology that allows for data to be relayed or hopped from node to node within a specific network. This means that data can traverse longer distances by relaying data between nodes over shorter distances. This is a way to extend communication range without changing hardware or software. All of iRobot's military robots (and their control boards) can be used as nodes in a mesh network to relay messages to other robots and operators.

Generally speaking, most current commercial search rovers exhibit very limited autonomous capabilities to assist users in rugged environments. The PackBot is one notable exception as a user can purchase an optional User Assist Package (UAP). This package upgrades the PackBot's software and hardware to support a number of autonomous features including retro-traversing, self-righting, and heading-hold [5]. Retro-traversing is a unique feature that combines environmental mapping and autonomous driving to retrace a robot's path if communications are lost. This removes the need for a human to enter an unsafe environment to retrieve the lost robot. Heading-hold enables the robot to stabilize itself and correct its direction of travel when traversing rough terrain that tends to bump the rover away from the driver's intended heading. These features make the robot easier and more natural to drive.

PackBot also has payload expandability for manipulators and other sensors. However, the payload interface is not open for third-party technologies. Some example payloads designed by iRobot include the Enhanced Awareness Payload (EAP), Manipulator 1.0 (3-Link Arm), and the HazMat Detection Kit.

2.2.2 Kobra

The iRobot 710 Kobra is a step up from PackBot. Weighing in at 370 lbs, the Kobra is designed to handle much more aggressive and large scale missions than PackBot. It too can support iRobot specific payloads, has mesh networking capabilities, and uses front treads for added mobility. Figure 5 shows the Kobra inspecting a truck using its added arm payload.

Kobra can drive up to 8 mph, go up 60° grades, lift 330 lbs, drive for 6 to 10 hours, and communicate up to 800 m LOS [19]. While as agile as PackBot, Kobra is about 5 times larger, being mainly designed for outdoor use. It would be near impossible to navigate Kobra around in tight corridors, but it has a mobility advantage when dealing with much larger debris. Unlike the PackBot, Kobra is not fully waterproof, as it can only be sub-

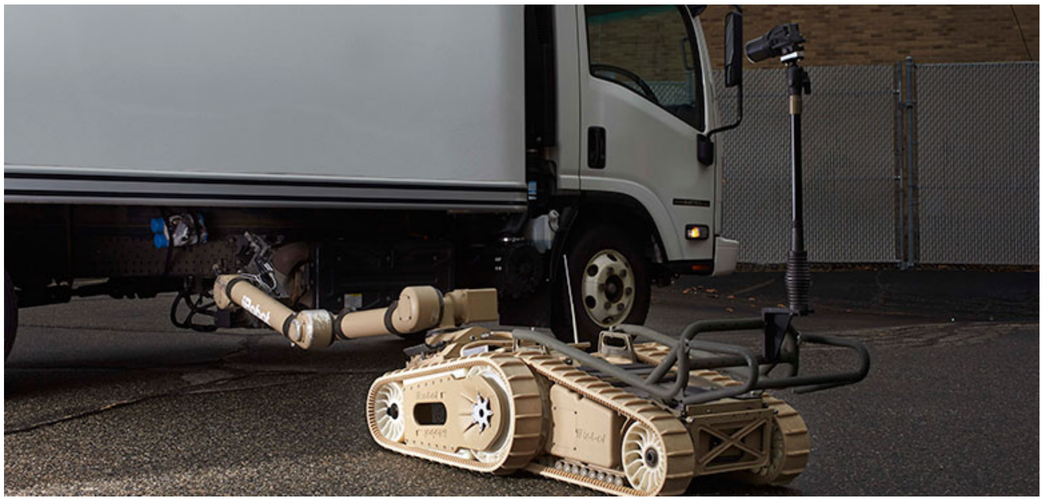


Figure 5: Kobra Inspecting a Suspicious Truck.[18]

merged partially up to 18 in. It also has limited autonomous features and does not support the UAP.

2.2.3 TALON

The QinetiQ TALON is another popular search and discovery robot mainly used for military applications. They are widely deployed for improvised explosive device (IED) and explosive ordnance disposal (EOD), reconnaissance, communications, CBRNE threats (Chemical, Biological, Radiological, Nuclear, Explosive), security, heavy lift, defense and rescue missions. There are currently two different models of the TALON - the C-TALON and the TALON V. The C-TALON shown in Figure 6 is a submersible crawling robot designed for use in rivers, surf zone, and limited access harbor areas.

The C-TALON also uses a treaded system like PackBot and Kobra, but without flippers. Specific to the C-TALON is its large-area treads to crawl along the bottom of a body of water with ease [20]. The robot carries a variety of sensors on board such as a compass, pressure sensor, odometer, GPS, and hi-res imaging sonar. It does not have the ability to add additional payloads easily like the Packbot and Kobra, and has no autonomous features.

The TALON V shown in Figure 7 is the most advanced TALON made by QinetiQ. Weighing in at 124 lbs, the TALON V can drive up to 6.75 mph, go up 48° grades, and operate for up to four hours [21].

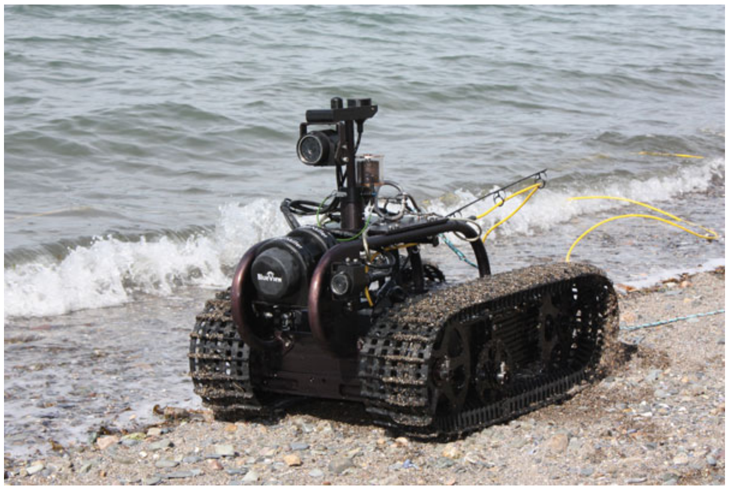


Figure 6: C-TALON on position to enter the water.[20]

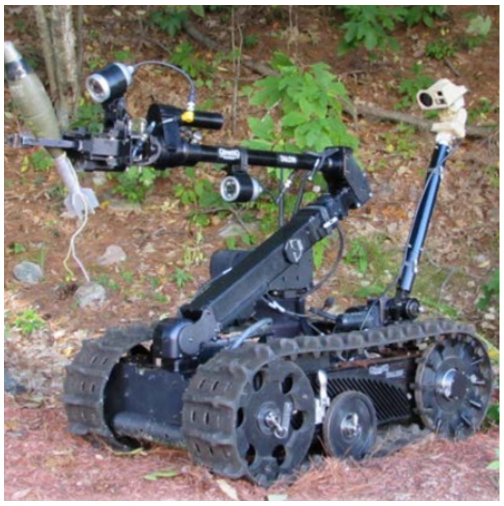


Figure 7: TALON V.[20]

One of the more advanced features of the TALON V is that its electronics and software are designed around open architectures and system level modularity. The robot provides 16 I/O ports including Interoperability Profile (IOP) A and B connectors and the software supports plug and play discovery of IOP devices. IOP refers to a military standard for open architecture and interfaces. It also has ports for USB, Ethernet, CAN, RS-232, and 12 V power, making it far more expandable than most other rovers on the market.

3 Design Requirements

This chapter will discuss the technical requirements of the WALRUS Rover that guided the design of the physical robot, its electrical systems and its software. These specifications were carefully chosen to represent the minimum capabilities required of a robot designed to complete the goals of this project, while keeping practicality in mind. The reasoning behind all requirements and how we planned to test them are elaborated upon in this chapter. A summary of this project's requirements is shown in Table 1.

Design Requirements	**Value**
Maximum land speed	$\geq$6.5 ft/s
Maximum water speed	$\geq$1 knot (1.69 ft/s)
Maximum stability limit	$\geq$45° pitch; $\geq$30° roll
Stair capability	$\geq$37°
Maximum total weight	80 lbs
Maximum size	32 in diameter circle
Maximum width	28 in
Maximum battery life	$\geq$1.5 hours
Minimum control distance	975 ft (Line of sight)
Payload Capacity	At least capable of holding light sensor packages for amphibious missions ($\geq$1 lb) or heavier packages like first-aid kits ($\geq$10 lbs) for landlocked missions
Operator feedback	Camera Views (Mast, Lower Front, Lower Rear) Orientation, Temperatures, Battery life, Self diagnostics, Environmental information
Shared control	Easy transition between land and water, Assisted stair climbing, Heading hold on land

Table 1: Summary of Design Requirements

3.1 Size

To facilitate indoor locomotion, the robot must be capable of turning in tight spots. According to Massachusetts (location of development) building code, hallways must be a minimum of 36 in wide [22]. For the robot to reliably turn in place, it must fit within a 36 in diameter cylinder projected vertically from the floor. We have elected to make our rover fit within a 32 in diameter cylinder to give the operator ample clearance when turning in hallways. In addition, the rover must be capable of going through doorways. The minimum size of a doorway is 30 in wide [22]. Our rover will be no more than 28 in wide to give the operator clearance when going through doors. The maximum height of the rover is not a large concern, though we intend to make it as short as possible for stability and fitting under fallen debris.

3.2 Stability

The robot will be climbing stairs, so it makes sense for it to be statically stable above the maximum 37° incline. To facilitate this and incorporate a safety factor, the robot must be statically stable (when powered on) at a pitch of 45°. In the roll direction, the rover must be stable at ±30° from vertical as well. This will allow the rover to turn on and horizontally traverse a reasonable incline without falling or slipping. Pitch and roll stability will be tested by tipping the robot on flat ground and measuring the angle at which it reaches static equilibrium and will no longer fall back to its resting orientation.

3.3 Weight

The vehicle must be buoyant in fresh water such that its means of propulsion are not hindered. This determines the vehicle's allowed weight based on its displacement. It should also not exceed the National Institute for Occupational Health and Safety material handling safety index of 2.0 for one-man carry. This means that the rover must not exceed a weight of 80 lbs. This is in contrast to the aforementioned PackBot's 24 lbs (without batteries) and Talon V's 124 lbs. WALRUS strikes a balance between the two, gaining added mobility while maintaining buoyancy.

3.4 Speed

The vehicle must be capable of 6.5 ft/s on land, and 1 knot (1.69 ft/s) in the water. These speeds are on the higher end of mobility for conventional vehicles and respective domains, as the WALRUS Rover must be highly capable in both land and water domains. The speed of the rover will be validated by driving straight over a distance of 32.5 ft in 5 seconds or under. The speed in water will be validated by driving the rover through still water over a distance of 15 feet in 10 seconds or under. Our land speed (about 4.4 mph) contrasts with PackBot's 5.8 mph and Talon V's 6.5 mph.

3.5 Stairs

The robot will be capable of driving up a common household set of stairs. To satisfy this condition, the robot must be able to climb a staircase with an incline of 37° from horizontal. Although this requirement will not guarantee success on all staircases, it will allow the rover to perform in a high percentage of mission scenarios [23]. The rover will be tested with staircases of different grades to determine at what steepness it begins to fail.

3.6 Payload

The robot will have a payload capacity for the connection of additional sensors. On water the robot must be capable of a capacity $\geq$1 lb, while on land a larger payload of $\geq$10 lbs must be accommodated. The rover must contain accessible data and power ports for payloads. All other requirements must be met with the maximum permitted payload weight added.

3.7 Power

Battery life is very important for a search and discovery robot. After estimating the power consumptions of actuators and electronics systems, we decided that a 1.5 hours operating time would be an reasonable goal and inline with the operating time of the PackBot. This requirement will be tested by subjecting the rover to continuous average use on fully charged batteries and noting how long it takes for the batteries to die.

3.8 Operator Interface

The operator interface must be capable of controlling the entire rover remotely. It will allow the operator to individually manipulate all actuators on the robot (in order to give them maximum flexibility). The operator interface must also provide real-time feedback to the operator to give them situational awareness of the rover's environment. This will include the ability to view all of the rover's cameras, a display of the rover's environmental sensors, and self diagnostic information including remaining battery charge and internal component temperatures.

3.9 Shared Control

In addition to providing manual control of all actuators, the operator should be able to control the rover at a higher level and leave low level control to the rover's software. This allows the operator to focus on interpreting the rover feedback instead of controlling every degree of freedom that it has. More complex actions will be executable from the operator interface including assisted stair climbing. The rover will be able to automatically detect the presence of a staircase in front of itself and offer the operator the choice to enter stair climbing mode. At this point the operator can command the rover forward and it will ascend the stairs, reconfiguring if needed.

3.10 Communication

A wireless communication system must be used to communicate with the robot. After doing research on available systems and existing capabilities, we decided a communication range of 975 ft line of sight (LOS) would be reasonable. More signal attenuation is expected when operating the robot in an urban environment and is not incorporated into this requirement. While the video bit rate at this distance is not specified it should be enough for high definition video streaming at a reasonable frame rate. This requirement is mainly determined based on existing search and rescue robots such as FirstLook, PackBot, and the TALON. The WALRUS Rover should be in a similar range as these commercially available robots. LOS range will be tested by driving the rover away from the operator and noting the distance at which communications are lost or the operator can no longer reliably control the robot through on-board vision systems.

3.11 Computer Vision

To provide the user with a full view of the environment, the WALRUS Rover must be equipped with multiple cameras. A collapsible camera mast must be fitted with a high-definition camera for driving and inspecting objects. At least two additional cameras must be fitted to the rover to provide the operator with additional views to avoid obstacles. Supervised autonomy features like stair detection may also use these cameras.

3.12 Other Sensors

In order to monitor the health of the WALRUS Rover and to collect environmental data during operation, additional sensors are required. Temperature sensors located at different points on the inside of the rover will ensure that the entire system is running below critical temperatures at all times. Additionally, water sensors must be incorporated to alert the operator of leaks before water damage can occur.

Environmental sensors are also required to provide rescue teams with information about potentially hazardous environments. These include sensors for temperature, audio, pressure, and dangerous gasses. Custom hardware design is required to interface these sensors with the main computer. Other sensors such as IMUs, encoders, and potentiometers will be used as needed.

4 Design and Analysis

This chapter will review the design decisions and analysis completed to build a professional-quality prototype and meet the requirements set in the previous chapter. The design methodologies will be broken up into six main categories: mobility, thermal and flow analysis, electrical systems, software architecture, user interface, and payload capabilities.

While each of these main categories have their own respective requirements, the design of each was conceived in the context of the larger system in order to properly interface everything together. With such a complicated system, robustness and hierarchical management are key in creating a reliable system. We had to ensure we completed enough research and prototyping to conclude we could meet our requirements before proceeding into design and analysis. Once fully designed; we completed manufacturing, assembly, and system integration. An external view of the WALRUS Rover can be seen in Figure 8.

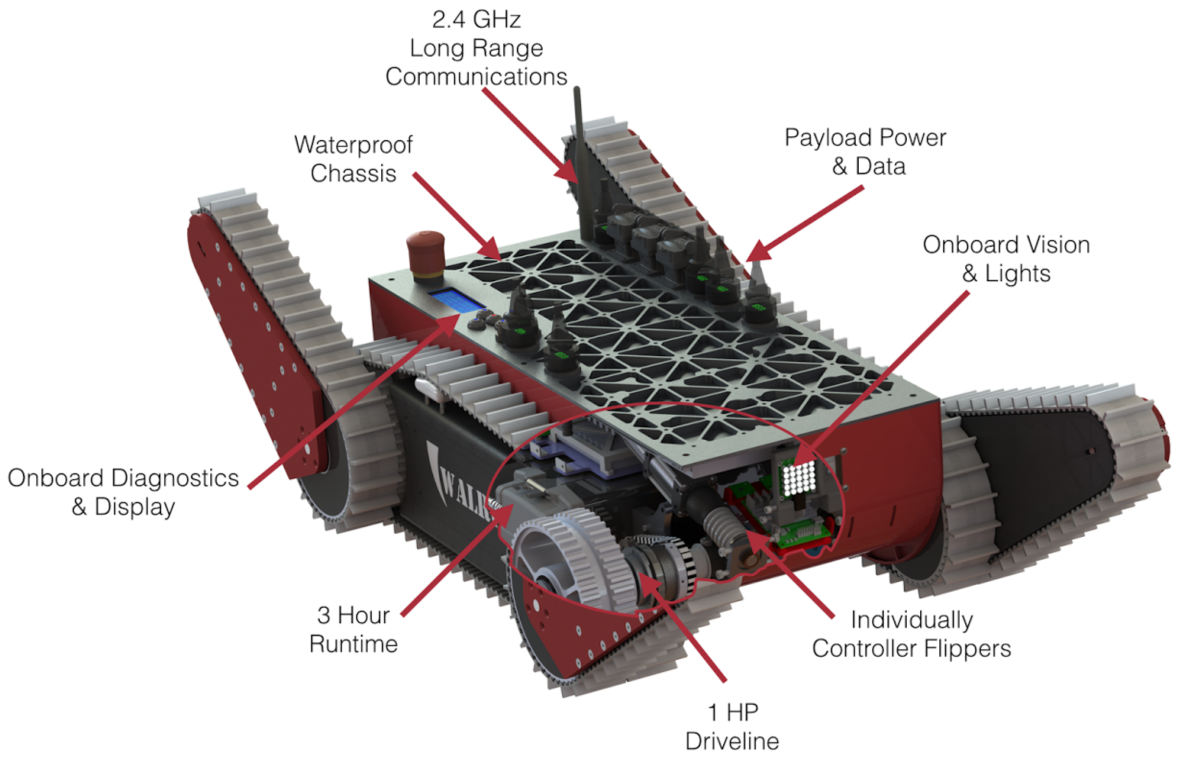


Figure 8: Full WALRUS Rover System Render.

4.1 Mobility

This section covers aspects related to the mobility of the WALRUS Rover including chassis design, main drive design, and flipper design.

4.1.1 Chassis

As the main structural component of the WALRUS Rover, the chassis shown in Figure 9 is a welded 6061-T6 frame which offers support and mounting points for the entire rover. On the sides of the rover are 1/4 in thick plates with lightening pockets down to 1/16 in. In the center of the rover there is a 1x0.5 in C-channel that is intended to prevent the side plates from bending as well as support the chassis during welding. To allow for a smooth radius for the bore seal on the top plate, there are two webbing plates on the top of the chassis. Lastly, the inside of the chassis is ribbed with four 1/4 in ribs that span the length of the chassis. These ribs not only provide rigidity to the 1/16 in under body, but also allow mounting of electronics and motors.

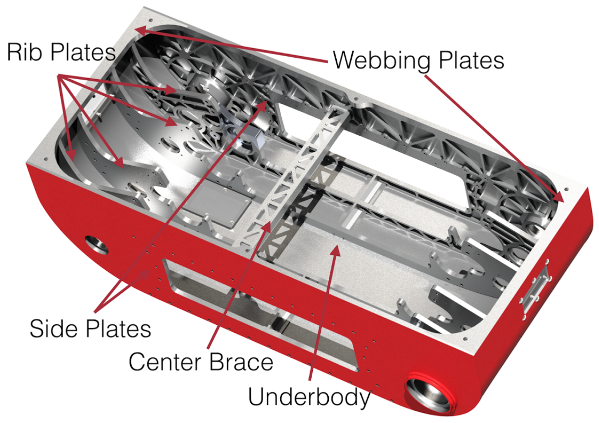


Figure 9: The WALRUS Rover Chassis.

For increased corrosion resistance as well as aesthetics, the exterior of the chassis is finished with a textured powdercoat. Additionally, the interior of the chassis went through a chromate conversion to prevent corrosion. In an effort to avoid alignment issues and shifting due to welding, we milled the chassis both before and after welding. All features were milled into the components pre-welding; however, all aligning features were left under-size so they could be located afterwards. This allowed the components in the rover to shift by over 1/16 in and still allow all bearings and press-fits to align. Examples of features that required this kind of treatment included the bore seal surface for the top plate and the bushing holes for the pod shafts. See Figure 28 in Section 4.4 for more details.

In the center of the chassis is a polycarbonate tube that acts as a bulkhead for the battery compartment. This tube is sealed with marine sealant to the sides of the chassis such that if the battery bay doors are open, the battery compartment can flood with water and the main chassis will remain sealed. The battery system uses waterproof batteries and sealed connectors so a leak in this section would not be a concern electrically. This was done because the batteries must be removed from the system to be charged, and the seals may not perform under repeated cycling. The battery bay walls are face sealed to the sides of the chassis with 1 in bolt spacing to maintain proper pressure, and then the battery bay doors are bore sealed and bolted in place.

Finally, there are three camera views in the chassis, and as such there are three port holes built into the underbody. A polycarbonate shield is sealed to each porthole using a soft rubber gasket, and as a secondary measure each assembly has marine sealant around it as well.

4.1.2 Main Drive and Shaft Sealing

Power transmission in the WALRUS Rover is achieved through two dual-inline shafts located in the front-left and back-right corners of the robot. The outermost shaft powers the treads for driving and the innermost shaft controls pod orientation. The other corners have only pod orientation shafting.

Powering each of the drivelines are Maxon EC45 motors. These motors must be able to propel the robot up a stair case of 35° at a speed of 2.5 ft/s. That means that each motor must be able to provide at least 150 W of mechanical power on the output. Given that shaft sealing will be used, we estimate about a 60% efficient system, meaning each motor must provide 250 W output before gearing. Each of these EC45 motors provides a nominal 250 W but can source up to 1000 W if proper precautions are taken. In the case of the WALRUS Rover each motor is current and speed limited so they will never exceed 500 W. These limits prevent the motor from overheating and causing damage.

In order to ensure a waterproof chassis, the WALRUS Rover uses a combination of radial shaft seals and o-ring seals to prevent water from entering. All six rotating shafts (2 for drive and 4 for flipper actuation) use spring-loaded neoprene shaft seals. These seals are lubricated using a marine-grade, water-repelling grease.

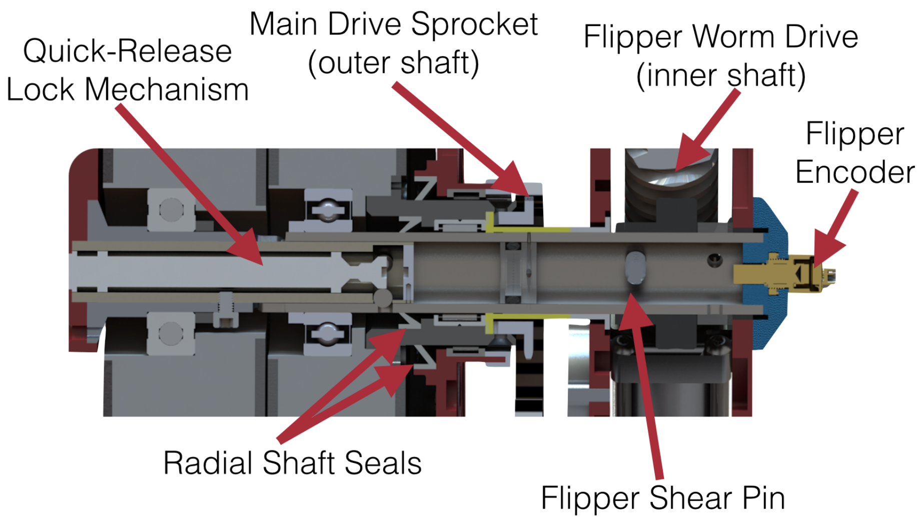

Quick-Release Lock Mechanism
Main Drive Sprocket (outer shaft)
Flipper Worm Drive (inner shaft)
Flipper Encoder
Radial Shaft Seals
Flipper Shear Pin

Figure 10: Integrated Sealing in dual inline shafts.

In order to allow both drive and flipper rotation to occur along the same shaft, the dual-inline shaft uses a combination of shaft seals and bearings to allow the main drive pulleys to rotate around the flipper drive shaft. As seen in Figure 10 the inner shaft (which is sealed with an internal plug) is supported by the dual needle bearings and a bushing. Surrounding it is one shaft seal which seals between the main drive shafts and the flipper drive shafts. All of the support for the main drive comes from rotation around the flipper shaft, as well as one needle bearing between the drive coupler and the chassis. There is one last seal that occurs between the chassis and the main drive. Although there are three inline bushings/bearings supporting the flipper shafts, it is important to note that the center bushing is only proving support for thrust loads, and does not provide any support for axially loads. This is done to prevent any alignment issues by having three bushings inline.

The interlocking mechanism for the flippers extends and supports the pod-driving shaft. This connects directly to the outer pod plate, which secures the idler pulley and therefore controls the orientation of the pod. A bearing pressed into the driving pod pulley supports one side of the pulley, the other being supported by the driving pulley pinned to it.

4.1.3 Flippers

The WALRUS Rover's flippers are intended to be the main source of its advanced mobility. Each flipper is comprised of a main drive pulley, idler pulley, tread, locking mechanism, inner support and outer structure plate. Figure 11 shows each of these components as they would appear in a disassembled flipper.

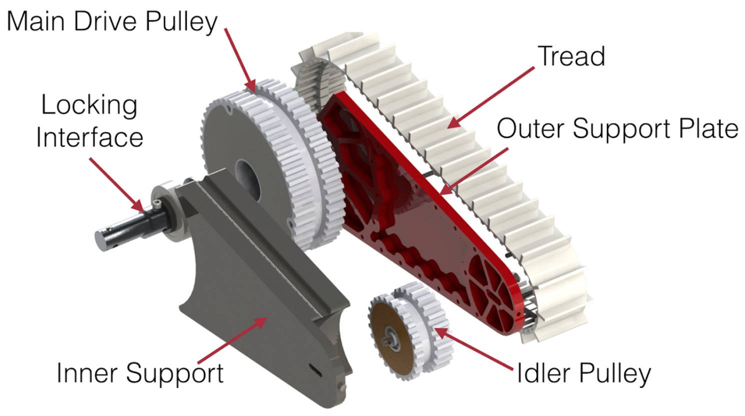


Figure 11: Disassembled Flipper.

The locking interface is a unique feature of the flipper subsystems. The flippers are non-backdrivable, and as such when the robot is powered off they cannot be moved. When moving the robot from place to place, it is convenient to be able to make the robot smaller and lighter for transport, and we wanted to do so toolessly. The resulting design is a standard ball-lock built into the flipper shaft as shown in figure 12. When the button on the end is pushed, it compresses a spring at the opposite end and the reduced section of the shaft allows the balls to fall out of the locking position, clearing the pod-driving shaft. When the button is released, the shaft springs back and puts force on the balls, pushing them out in an attempt to lock. If properly aligned with the pod-driving shaft, the balls fit into holes on the driving shaft and create an interference which holds the pods in place.

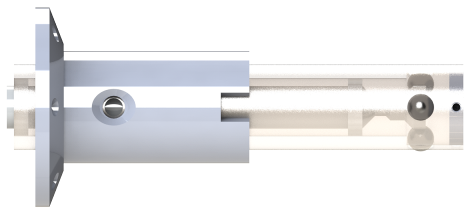


Figure 12: Locking Interface (shaft made clear for internal view).

4.1.3.1 Sizing and Design

As this rover will be used in a wide variety of scenarios, we decided that each flipper should be interchangeable. In an effort to allow the robot to do self-lifting maneuvers, each flipper must be long enough to pass the center of gravity of the robot. Measured as the maximum length between the two pulleys each flipper is slightly over 15 inches, which facilitates lifting. Figure 13 shows that each pod is capable of passing center.

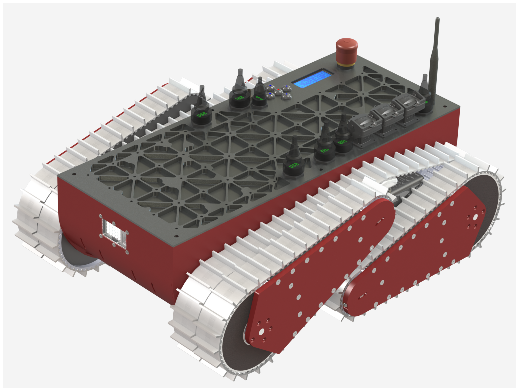


Figure 13: WALRUS Flippers passing center.

In addition to allowing the rover to do self-lifting maneuvers, the use of front and rear flippers allow the WALRUS Rover to climb stairs. As explored earlier, many robots need to shift their center of gravity (CoG) forward to provide stability while climbing stairs. Without any additional payloads or internal moving systems, the WALRUS Rover does not have the ability to shift its center of gravity. This decision was largely driven by the additional complexities required to add a separate system that adds this degree of freedom. Additionally, using some sort of manipulator or payload to climb stairs not only limits the mobility of the base system itself, but also limits the user's ability to add additional payloads in conjunction to the CoG shifting system. Instead of shifting the CoG, the rover will rotate its rear flippers behind itself to extend its drive base by 50%. By doing this, the robot is long enough that its CoG will always be in between two contacting steps. Although the robot is capable of being stable of a pitch of up to 80° and most staircases are much less than this, during the climbing process, if the CoG of the robot is not in between two steps, the robot is performing an action more similar to a bump traversal rather than incline traversal. This will cause the robot to tip past its stability if precautions are not taken.

4.1.3.2 Power Requirements

In order to allow this high grade of mobility, each flipper's drive must be capable of providing a significant amount of torque. Knowing that we wanted approximately 5 RPM on the output, we used the calculations below to find the output power and torque needed for this system. After further analysis we discovered that self-lift maneuvers were not the most high-torque scenario for the rover's pods. To determine this we created a MATLAB simulation of the forces on pods at different times, and discovered that the worst-case scenario actually occurred when all four flippers are lifting the robot from the extended position. This simulation assumed a coefficient of static friction to be 2 between the pods and ground (A high coefficient of friction is likely on compliant ground such as sand, grass or deep carpets). In Figure 14 the results of this simulation are shown. Figure 14 shows the flipper positions as the X and Y axis, while the resulted torque exerted on the rear flippers is shown as the depth (Z) axis. Due to the assumption that the center of gravity of the robot is at the X/Y center point, the graph for the front flippers should be a mirror image, and therefore is not calculated. The function used in Figure 14 is derived from basic free-body diagrams using the robots' weight, orientation, and forces due to dragging the treaded flippers along the ground.

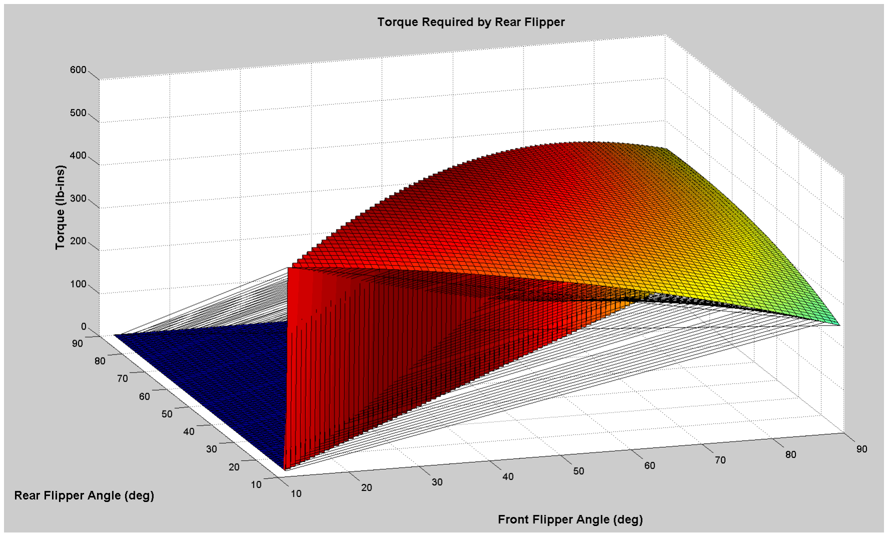


Figure 14: 3D Surface of Forces Exerted on Flippers.

Using Figure 14 we know the maximum static load that each flipper will see is about $540 in * lbs$ assuming that the robot is level in roll and the CG is centered in the chassis. In an effort to prevent the flippers from being under-powered, the system is intended to sustain loads of up to $880 in * lbs$. At 5 RPM, this corresponds to a 52 Watt draw. We needed to find a motor which output this power with a high enough efficiency that it did not draw more than 10 Amps at 12 Volts. The reasoning behind this comes from a desire to not trip the current limiting features of our main batteries. With all of this in mind, we chose the RS550 motor, easily acquired from Banebots. Although it is an air cooled motor, it would see no more than 10 Amps through the windings in operation and spin at 16000 RPM when loaded, both of which are excellent operating conditions for a motor running intermittently.

4.1.3.3 Power Requirements

As the flipper subsystems are involved directly in interacting with the environment, special care was placed in designing them to be rugged and durable. We will look at the safety mechanisms for the flippers as we work our way out from the inside. The internal driving mechanism for the flippers is paired with a 256:1 planetary reduction and through a unique 20:1 worm gear reduction for a total anti-backdrive reduction of 5120:1 and a total efficiency of 31%. This gearing was chosen to keep the operating conditions of the motor within 10% of free-speed operating conditions. This is the more energy efficient side of the motor curve and keeps the current draw low enough for multiple flippers to run and not trip our current limits. This gearing is unique because it is designed to be a compliant system. As shown in Figure 15 you can see that there are springs on either end of the worm separated by thrust washers, which allows for movement of the flippers without movement of the driving motor. This is designed to allow for impacts to the flippers and energy absorption during falls. The system is extremely rigid, and the effects are only seen during impacts.

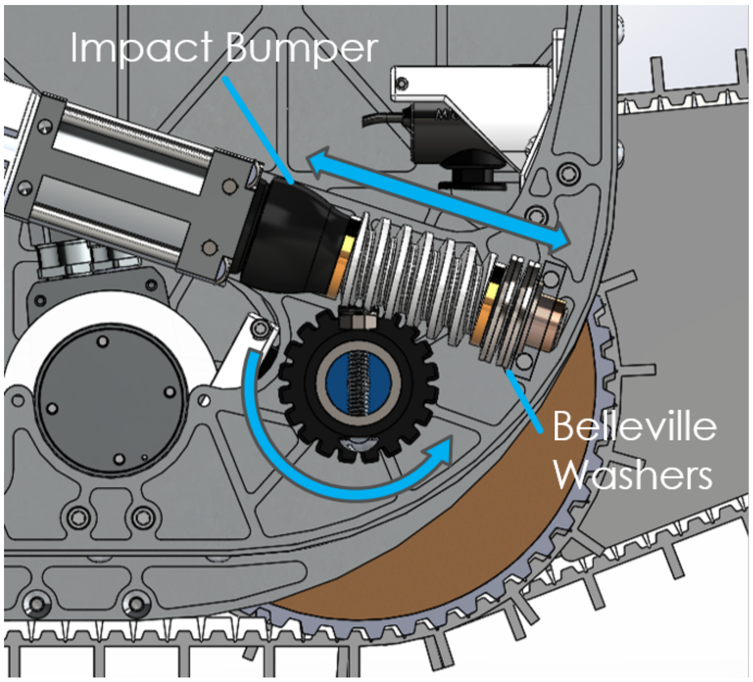


Figure 15: The compliant actuation system for the flippers.

After the worm is the worm gear. This part of the reduction was of particular interest, because it takes all of the torque applied to the flipper. The challenge was balancing the size of the reduction and the strength of the reducing components. We considered a number of different drive types, including spur, helical, planetary, bevel, and chain drives. To help with spacing, we preferred to have a gearing system capable of running perpendicular to the drive shafts, which is easily done with worm, bevel, and helical gears. A worm drive was eventually selected because the other drive types required such large pitches for the teeth to survive that only small reductions could fit between the pod shafts and the curvature of the underbody. To facilitate these decisions, Finite Element Analyses (FEA) were performed on various gearing types using the concept outlined in reference [24]. One gear was held fixed, and the second gear was rotated into the first. The results of these analyses can be seen in figures 16 through 18 below. Because the worm distributed the load out between more teeth than a standard spur gear, it could survive higher loads. We were able to verify these results by comparing output power limitations from gear designs catalogs offered by manufacturers. We also verified the results through our own testing by applying the maximum torque the set could see statically and comparing the results. This test actually far exceeded the values calculated by the FEA where no damage occurred to the gears even when approaching 1200 in-lbs. This is likely due to the nature of the Finite Element Method, which can have difficulty accurately solving for forces generated at sharp corners like the ones where teeth are cut out from gears.

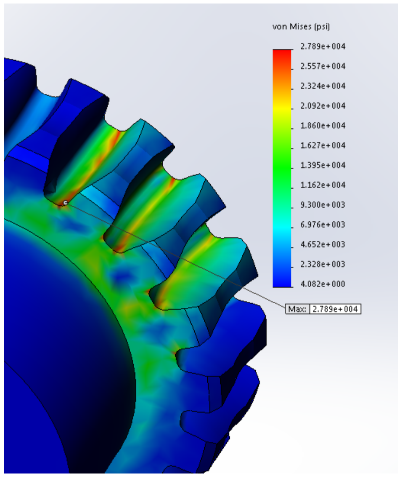


Figure 16: 12DP worm gear

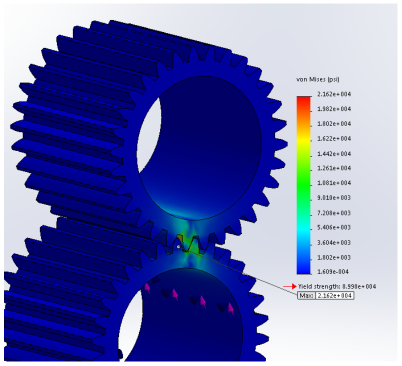


Figure 17: 20DP Spur Gears

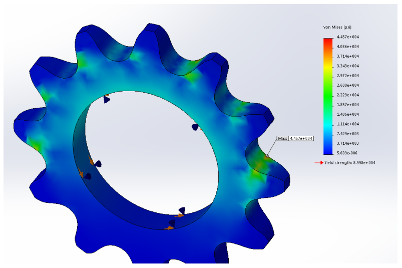


Figure 18: #35 chain

While the suspension served as a precaution for the average use-case, it is still possible that the system could experience a fall or jarring impact causing greater forces than the pod subsystem could handle. To prevent damage to the complex suspension system, a secondary safety measure was taken. A shear pin was designed into the connection between the worm gear and the flipper drive shaft. This pin was designed to shear at the same torque tested on the gears originally, 1200 in-lbs. The pin is in double shear, and after some calculations it was determined that an aluminum (6061 Alloy) pin with a diameter of 0.228" would make a perfect pin. This corresponds roughly to the minor diameter of a 1/4-20 bolt, and so an aluminum bolt can be found securing the rotation of the worm gear to the flipper drive shaft. A separate capture point holds the shaft securely in the bearings in case this bolt fails.

4.2 Top Plate Design and Analysis

The top plate is an important component of the WALRUS Rover and serves several purposes. It acts as the mounting point for additional payloads and houses all of the waterproof connectors. It also seals the main chassis, using a combination of a bore seal and a face seal to prevent water ingress. Finally, the anodized aluminum plate acts as the primary heat sink for thermal dissipation.

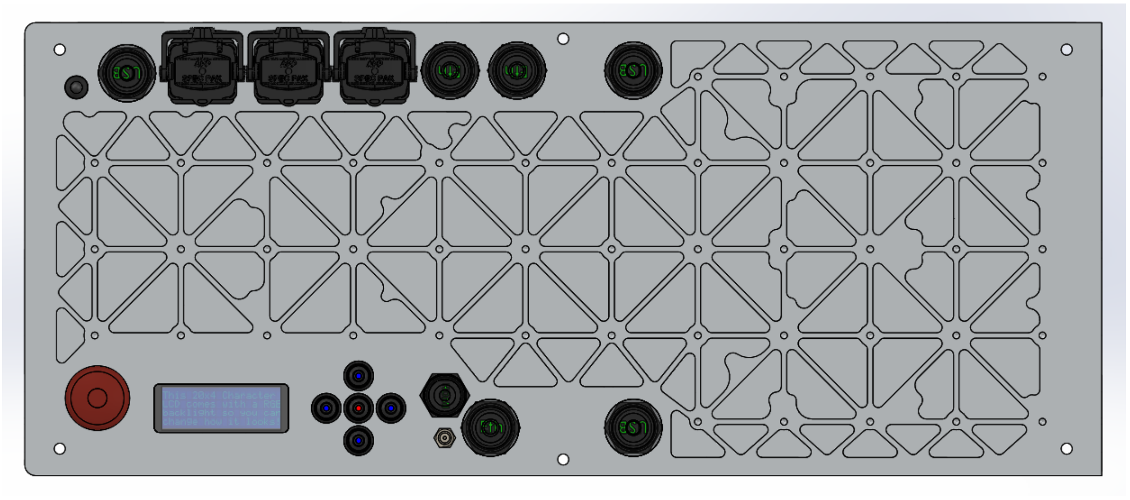


Figure 19: Top plate lightening patterns and interface.

One of the challenges associated with a sealed robot is how to appropriately cool the system. Motors are far more efficient heaters than rotary power transducers, and the onboard computers and power regulation systems produce heat during normal operation. The solutions for this problem are varied but the goal for the WALRUS Rover was to be able to cool the system entirely passively through the use of heat sinks and convective cooling. The top plate was designed with this in mind. Instead of lightening the bottom side of the top plate, the lightening patterns are exposed to the air and act as heat fins. The specific pattern they form is the result of not only where material is needed for bolting holes but also based on the required density of the heat fins for ideal heat dissipation.

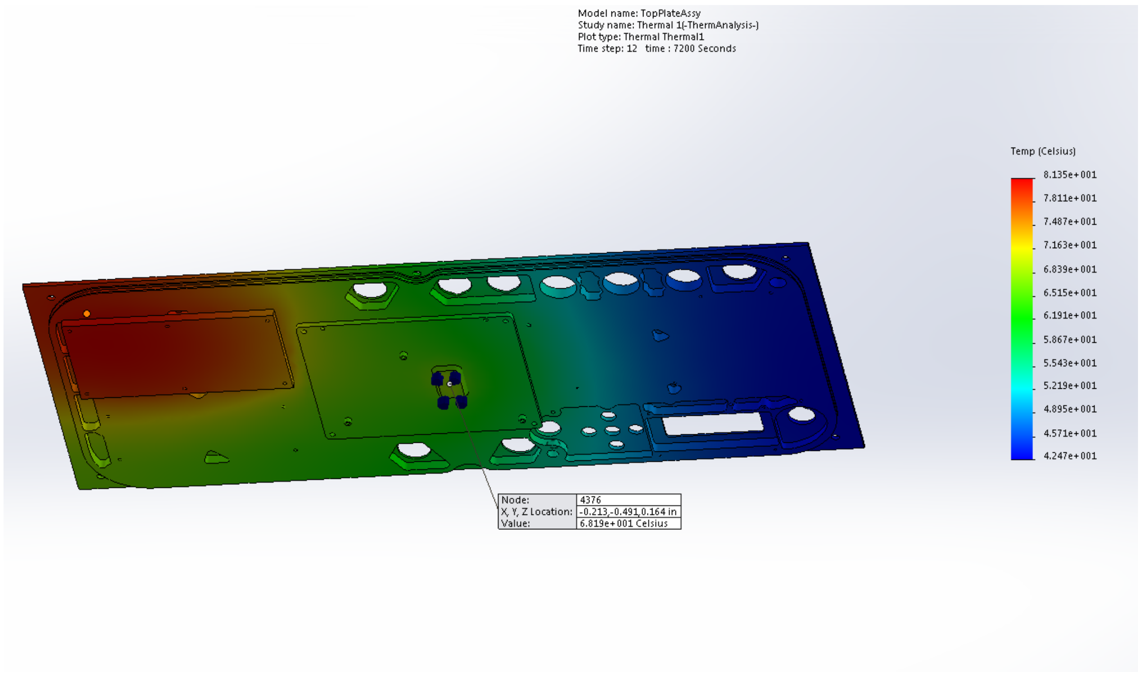


Figure 20: Top plate Thermal Simulation.

The results for this test can be seen in Figures 20 and 21. The temperature of the computer never exceeds 85 °C even when running for well over our maximum specified full-load run-time. This has since been verified during testing as we have never observed the computer's temperature to exceed 87 °C. We considered adding more fins to the top plate, however, filling the lightening patterns with fins only decreased the maximum CPU temperature by a few degrees, but would have added several pounds to the top plate. The additional weight added had to be compared to the potential benefit for cooling purposes, and the pattern was modified based on these trade-offs.

4.3 Flow Analysis

When designing a robot that is largely for use on land it is very difficult to maintain a proper body shape for use in water as well. To ensure the WALRUS Rover does not experience significant drag forces and back-currents, we completed many simulations as well as testing.

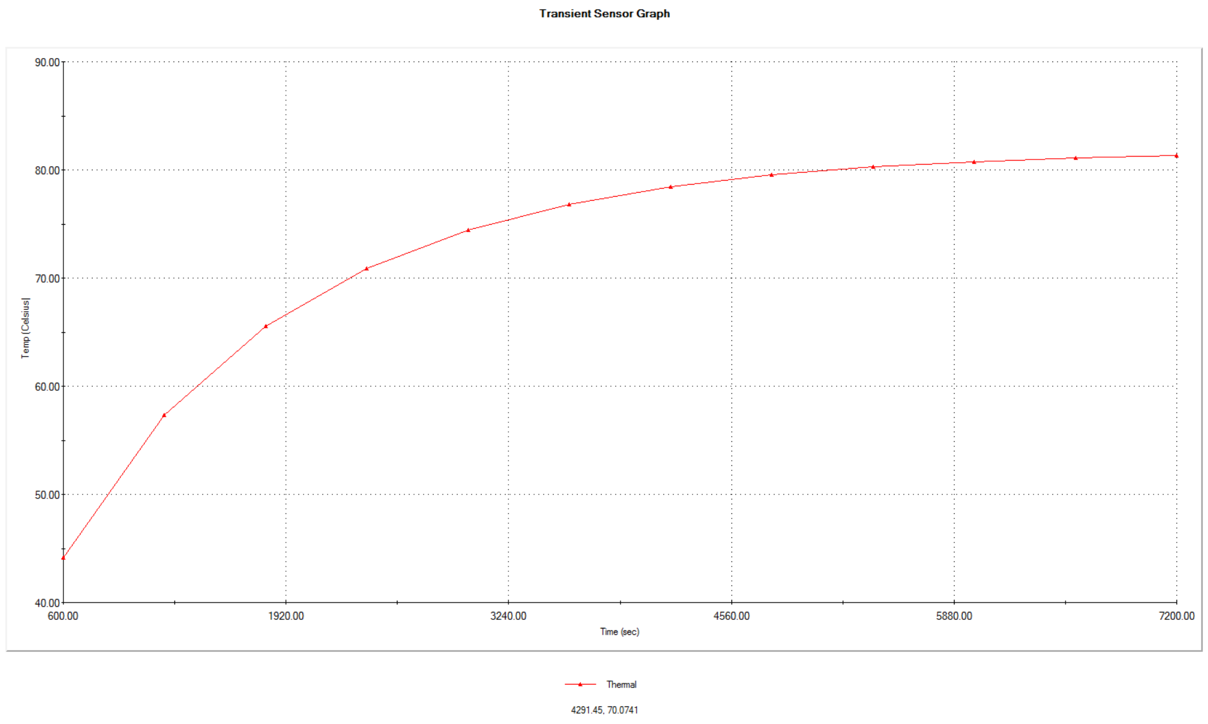


Figure 21: Computer temperature over time.

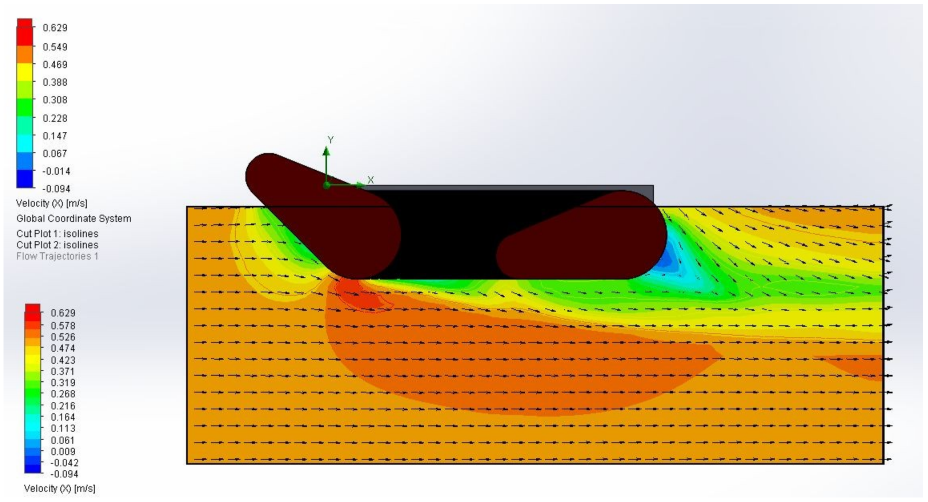


Figure 22: Solidworks Flow Simulation.

Using Solidworks simulation tools we were able to simulate a flow of water at 1.68 ft/s to see how the water will react around the robot. As seen in Figure 22 we were able to see where water will become still around the body and where we can make improvements. Also using this simulation we were able to estimate drag at 1.2 lbs.

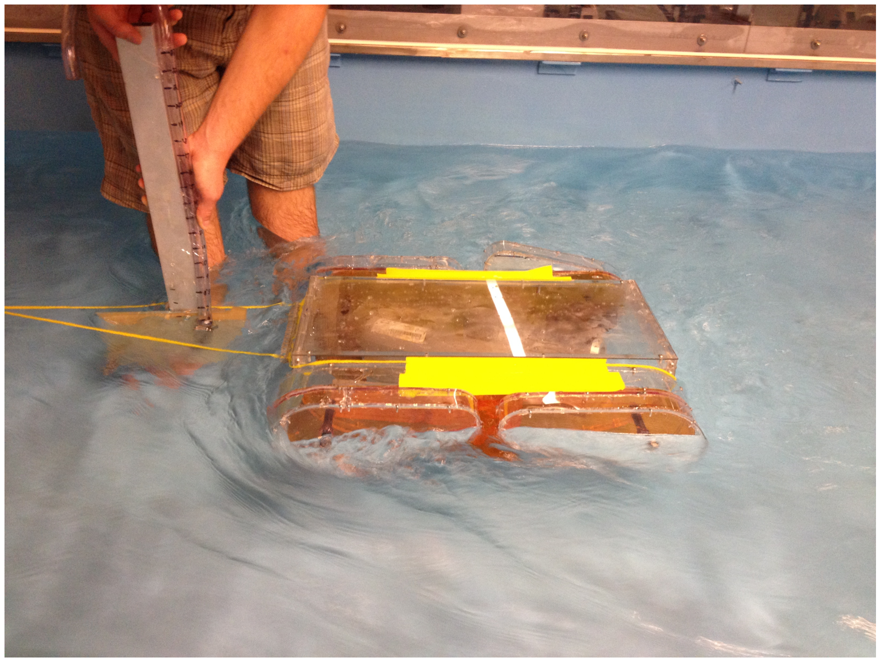


Figure 23: Verifying flow simulation results in the WPI Rowing Tank.

To verify the accuracy of the simulation, we created a mock robot that we placed in WPI's rowing tank (Figure 23). We then ran the water in the tank to simulate the robot moving at the desired speed. By putting a force gauge in between the anchor and the test robot we were able to measure the actual drag force that the robot experienced. Shown in Figure 25, the drag force on the robot varies greatly with flipper positions and is within range of the simulation.

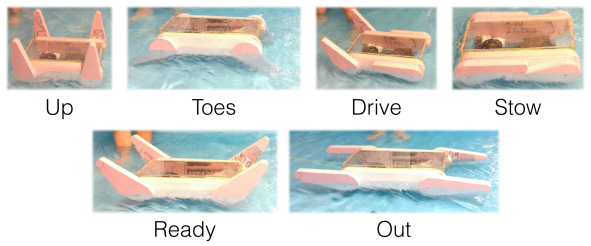


Figure 24: Drag Rig with Flipper Positions

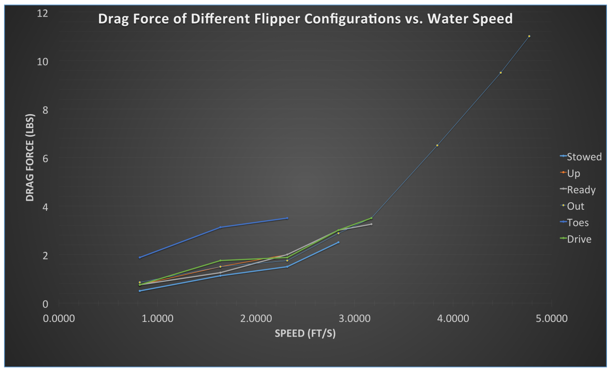


Figure 25: Drag Force by Speed and Flipper Position

In order to ensure that we could generate enough thrust to overcome the force of drag, we created a thrust test rig to measure the force generated by treads in water. Although these treads were not the final treads, they maintained similar speeds, density and size as the real treads. Figure 26 shows the thrust rig in the water. Through these tests we were able to show that our treads can produce around 1 lb of thrust per side. Using that we were able to proceed with our design of using treads as propulsion, as there was only between 1 to 2 lbs of drag on the robot, meaning our treads should be able to overcome this force.

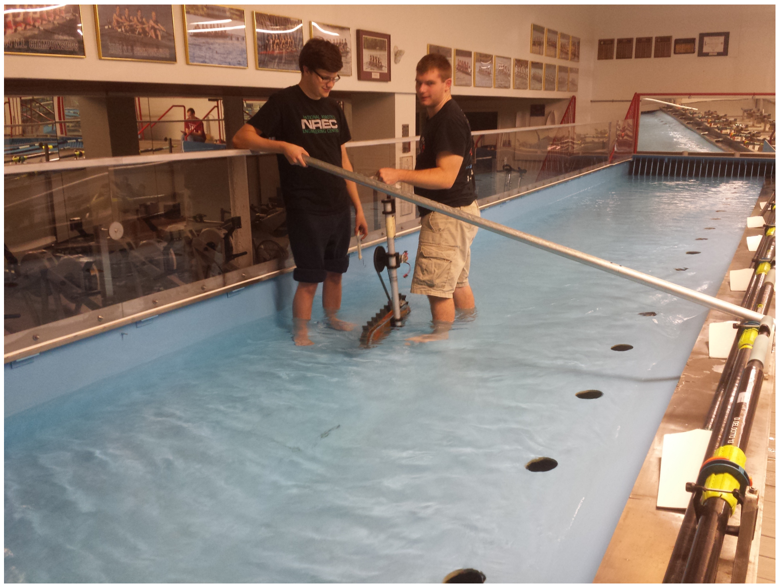


Figure 26: Thrust testing in the WPI Rowing Tanks.

4.4 Manufacturing

Through the use of the WPI Manufacturing Labs, the team completed nearly all of the manufacturing for the WALRUS Rover. Many parts for the rover were done on HAAS CNC milling machines. For the chassis and top-plates, we used a larger HAAS VM2 as shown in Figure 29, while the remainder of the aluminium and plastic parts were completed on a HAAS TM1 as shown in Figure 28. In addition to the CNC machines, we used manual machines (mills and lathes) to complete parts like those in Figure 27. Use of these machines allowed us to gain experience in Design For Manufacturing (DFM) as we were able to better understand the design and manufacturing trade offs and how part & feature counts have an effect on the final product.

Some additional parts inside the rover were manufactured through other means as well. For example, the mid electronics mounting board is made out of a 1/8 in thick piece of delrin

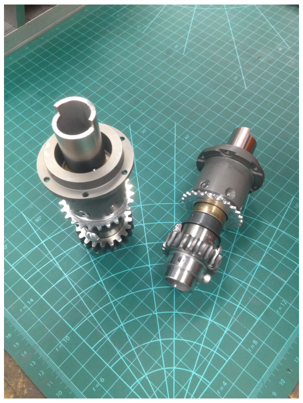


Figure 27: Inline Drives during assembly

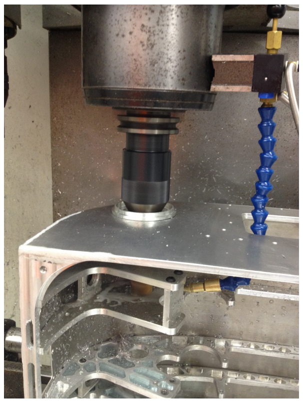


Figure 28: Post-processing the chassis

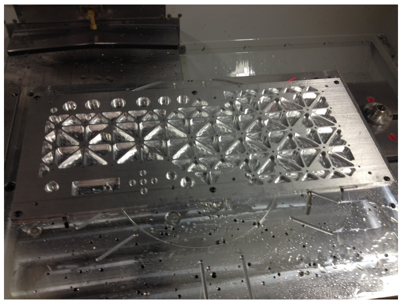


Figure 29: CNC Machining the top plate

that was manufactured using a VERSA 80 W Laser Cutter. Internal cameras and some other small parts were 3D printed on several different printers all using Fused Deposition Modeling (FDM) techniques. Using these methods of manufacturing greatly reduced some of the manufacturing time as these methods are less time-consuming compared to CNC Machining.

Many of the metal parts of the rover underwent various finishing techniques after manufacturing to prevent corrosion and wear through repeated use in water. Some of the processes used were hard coat anodizing, clear coat alodining/chromating, and powder coating. The chassis external and flipper outer plates went through powder coating instead of anodizing for aesthetics as well as concerns with welding pockets causing acid build-up inside the rover. Several parts, including the camera system payload, were alodined instead of anodized due to time and cost constraints. Many parts including all non-stainless steel shafting and the top plate were finished with a natural hard coat anodization. This allowed the top plate to have improved thermal dissipation as well as coating to prevent wear and corrosion over time.

4.5 Electrical Systems

This section discusses the design and selection of the main electrical systems of the WALRUS Rover. This includes the main computer, communications module, batteries, backup power system, and power distribution. Figure 30 shows a high level system overview of the electrical hardware architecture. Figures 31 and 32 show the electronics layout of the chassis and top plate.

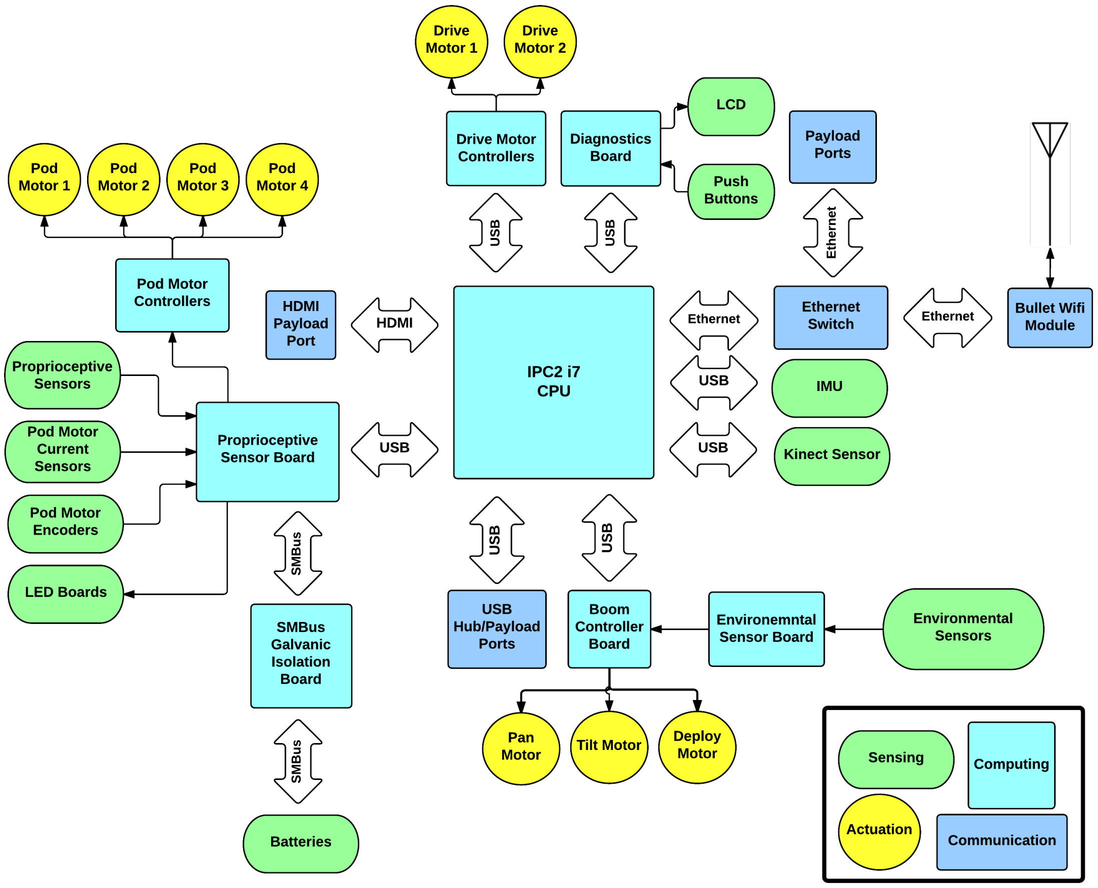


Figure 30: High Level Electrical Systems Diagram.

Figure 30 is broken up into subsystems, and the key is shown in the bottom right hand corner. In the center is the main computer which runs Linux. This computer handles our vision processing and main control loops, and communicates with a number of hardware controllers and custom boards over USB and Ethernet. The custom hardware designed include the proprioceptive sensor board, galvanic isolation board, diagnostics board, LED board, boom controller board, and the environmental sensor board. The remaining components are consumer products used for motor control, communications and sensing. Custom hardware allows us to interface a number of sensors and motors to the main computer which would otherwise be very difficult or impossible to interface with. In addition, there are payload data ports for USB, Ethernet, and HDMI. These ports are used to communicate with accessory payloads that can be added to the rover.

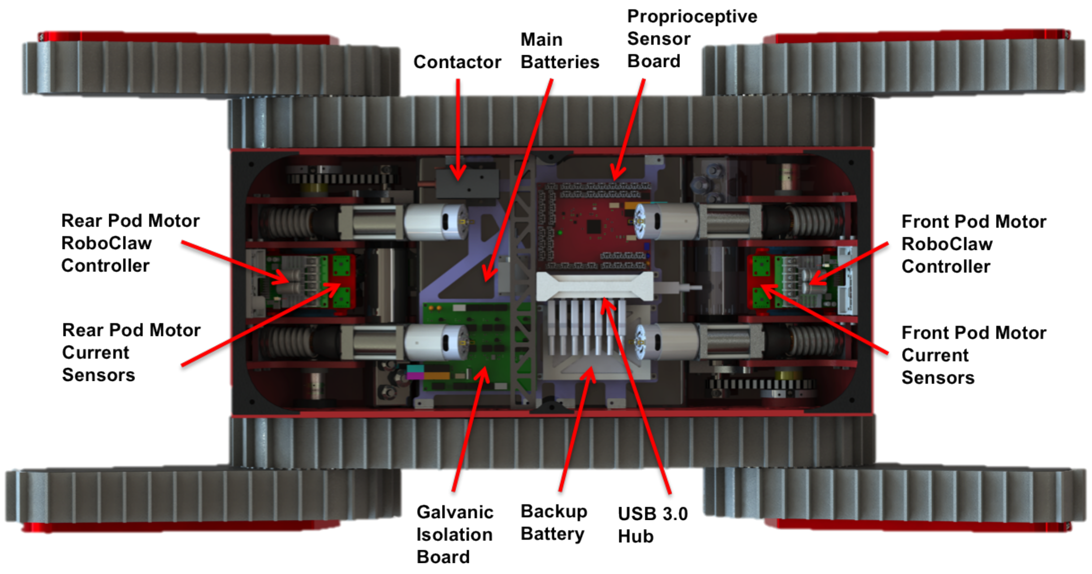

Contactor
Main Batteries
Proprioceptive Sensor Board
Rear Pod Motor RoboClaw Controller
Rear Pod Motor Current Sensors
Front Pod Motor RoboClaw Controller
Front Pod Motor Current Sensors
Galvanic Isolation Board
Backup Battery
USB 3.0 Hub

Figure 31: High Level Chassis Electrical Systems Layout.

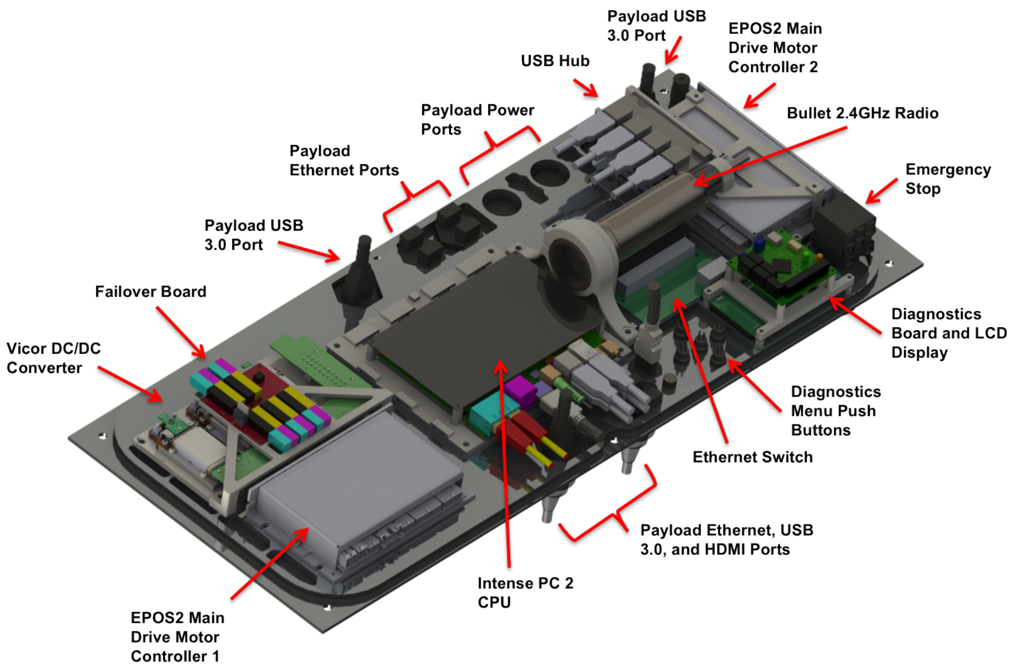


Figure 32: High Level Top Plate Electrical Systems Layout.

4.5.1 Main Computing

When searching for a main computing platform we considered five main factors: performance, efficiency, price, size, and architecture. The WALRUS Rover is intended to be a capable and expandable platform for the development of supervised autonomy. Therefore its main computer must be powerful enough to meet the needs of CPU intensive algorithms yet small and light enough to fit inside the robot and efficient enough to be passively cooled by our chassis. Through previous experience we felt that a system based on a mobile Intel x86 processor would best fit these needs as anything more powerful would require too much space. In addition, the x86 architecture is capable of running Ubuntu Linux which is the most compatible operating system for developing software based on the Robot Operating System (ROS).

Given these considerations we found a number of small computing modules and single board computers (SBC) including CompuLab's line of FitPC products, Axiomtek's nano and pico SBC's, Intel's Next Unit of Computing (NUC) and a number of rugged computing modules for military applications. Though a ruggedized computer would be an ideal and safe option given the rover's application environment, they proved too expensive for the target price point of the rover and for the team to procure. We felt a consumer grade solution would be adequate given the rover's expected operating conditions so long as we used solid state instead of spinning disks. In the end, we chose to use the SBC version of CompuLab's IntensPC2 with a fourth generation Intel Core i7 processor. This SBC provides the required power at a reasonable price and was simple to heat sink to the robot chassis for passive cooling. We chose the most powerful processor available for the model due to its negligible affect on heat produced and acceptable price increase. A summary of the computer's specifications can be found in Table 2.

Processor	Intel Core i7-4600U
Clock Speed	2.1 GHz
RAM (User Installed)	8 GB DDR3 SDRAM
Storage (User Installed)	64 GB mSATA SSD
Graphics	Intel HD 4400
Networking	2x Gigabit LAN, WiFi 802.11ac
USB	2x USB 3.0, 2x USB 2.0
Display	2x HDMI, DisplayPort
PCI Express	Half size mini-PCIe slot, Full size mini-PCIe/mSATA slot
Audio	3.5 mm stereo audio jack, 3.5 mm stereo microphone jack
Other IO	3x RS232 mini serial, Micro SIM slot

Table 2: CompuLab Intense PC 2 Specifications.

4.5.2 Main Drive Motor Controllers

To control our Maxon drive motors we decided to go with Maxon's brushless EPOS2 motor controller shown in Figure 33. This controller is low profile and has connections for our drive motor's integrated encoder and hall effect sensors. While other brushless motor controllers would work, the EPOS2 have a number of advanced controls features such as automatic position PID calibration and velocity control that make them more desirable for our application.

Figure 33: EPOS2.[25]

The EPOS2 can handle 10 A continuous at a max voltage of 70 V, but you have the ability to adjust the controllers' overcurrent shutdown value to allow the EPOS2 to pull more than 10 A of current for a specific duration of time. They also have a CAN port which allows us to daischain the two EPOS controllers for the two drive motors together and communicate over USB to only one of them.

4.5.3 Pod Motor Controllers

To control the pod motors, DC motor controllers had to be selected. We needed dual channel controllers that could handle 32 V off the main battery line and about 6 A per channel. We therefore selected the dual channel 34 V max, 15 A continuous RoboClaw controller from Orion Robotics as shown in Figure 34. This motor control is commanded by the proprioceptive sensor board.

One problem with this setup is that quick changes in drive motor speed and direction can cause large voltage spikes on the battery line due to the natural inductance of the drive motors. An oscilloscope reading of this voltage spike shown in Figure 35 revealed an

Figure 34: RoboClaw.[26]

amplitude of about 70 V. This amplitude can easily damage the RoboClaws and therefore the controllers need some additional protection.

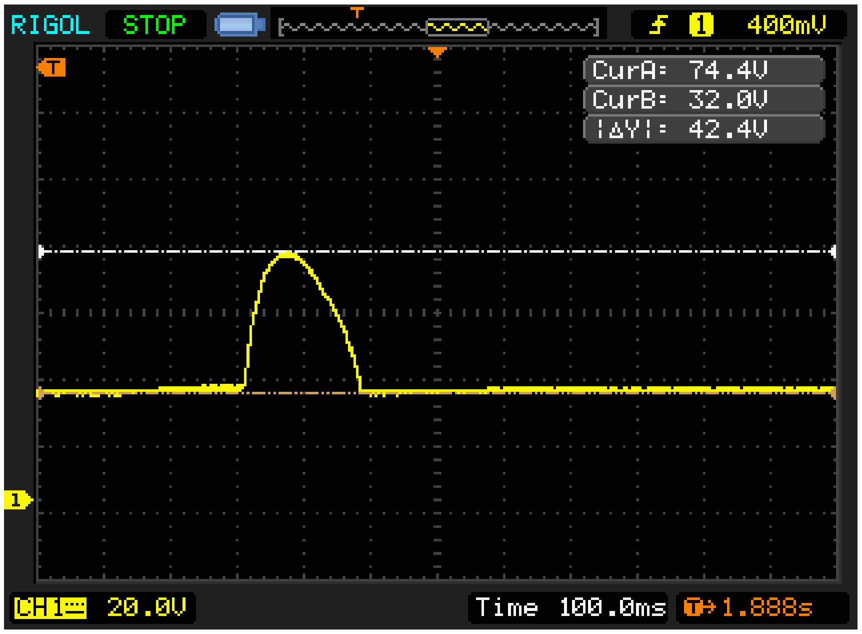


Figure 35: RoboClaw Voltage Spike.

Most robots like the iRobot Packbot use capacitor banks to absorb this energy but there is very little room inside of the WALRUS Rover to house a large bank of capacitors. Therefore, a simple software solution of slow deceleration is used so that the change in energy is much less aggressive, resulting in no voltage spikes. To determine the deceleration value required, we measured the battery voltage line with an oscilloscope while running the robot full speed and then stopping it. We then decreased the deceleration value until the voltage line remained flat.

4.5.4 Communications Hardware

In order to teleoperate and receive data from the rover, there must be a wireless communications link between the rover and control station. Initially the team thought about using a 4G network, but this communication would be limited in areas with poor reception or broadband infrastructure. Therefore, we determined that using a 2.4GHz 802.11 WiFi module was the better fit. Driven by the 975 ft LOS requirement, we selected the Bullet M2 WiFi radio from Ubiquiti as our communications hardware. Figure 36 shows a diagram of the Bullet.

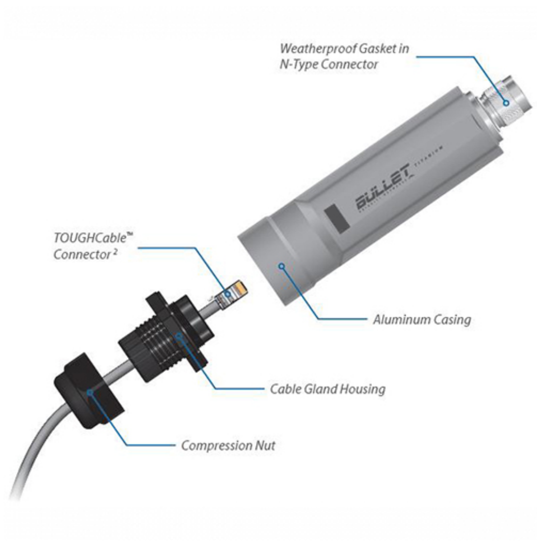


Figure 36: Bullet WiFi Radio.[27]

To select antennas and ensure that the Bullet module would give us the distance needed, the received power at the max distance required of 975 ft needed to be compared with the Bullet's worst case receive sensitivity. First, antennas were selected based on size and gain. The selected robot antenna has a gain of 3 dBi and the control station antenna has a gain of 8 dBi. The Bullet's output transmitting power, as given on the data sheet is 28 dBm and its worst case sensitivity is −96 dBm [28]. Assuming free space and line of sight, we can calculate the robot's received signal strength as follows. First we must calculate the loss in signal after the first meter.

$$P_0 = P_{tx} G_{rx} G_{tx} \left(\frac{\lambda}{4\pi}\right)^2 \tag{1}$$

$$P_0 = 28\,\text{dBm} \times 3\,\text{dBi} \times 8\,\text{dBi} \left(\frac{\frac{3 \times 10^6\,\text{m/s}}{2.4\,\text{GHz}}}{4\pi}\right)^2 \tag{2}$$

$$P_0 = 1.5 \times 10^{-4}\,\text{mW} = 10\log(1.5 \times 10^{-4}\,\text{mW}) = -38\,\text{dBm} \tag{3}$$

Received Signal Strength at 1 Meter

Next, we must calculate the total loss in signal strength at the 975 ft (300 m) requirement.

$$Prx = P_0 - 10\log(d) \tag{4}$$

$$Prx = -38\,\text{dBm} - 20\log(300) \tag{5}$$

$$Prx = -88\,\text{dBm} > -96\,\text{dBm} \tag{6}$$

Received Signal Strength at 300 Meters

Since the received signal strength is greater than the sensitivity required by the bullet of −96 dBm, this system of the Bullet radio coupled with the 3 dBi and 8 dBi antennas should allow us to communicate with our robot within the 975 ft requirement.

4.5.5 Battery Selection

When choosing a battery for the WALRUS Rover the most significant deciding factors were weight, size, capacity, and current draw rating. Since lithium ion batteries have the highest specific energy when compared to other types such as lead acid and NiMH, they typically make the most sense to use when trying to minimize weight and size.

Driven by the run time requirement and our selection of motors, we chose to use BB-2590 lithium-ion batteries. These batteries provide 8.7 Ah of charge at their nominal voltage of 32 V. It should be noted that the BB-2590 batteries contain two 16 V cells that can be wired in series to provide 32 V. This series configuration is used by the WALRUS Rover to provide adequate voltage for our 24 V drive motors. Table 3 provides estimates on power consumption and run time when operating with two batteries to meet our requirement.

System	**Average Draw (W)**	**Max Draw (W)**
Drive Motors	180	500
Pod Motors	30	120
Sensing and Vision	30	40
Wireless Communications	5	10
Main Computer	30	60
Total	**275**	**730**
Run Time (2 Batteries)	**1.5 Hours**	**0.5 Hours**

Table 3: Power Draw Estimates.

The BB-2590 batteries have a number of advantages over other lithium-ion batteries as well. They are military spec meaning they are more rugged and protected than a standard lithium-ion battery. They also already have an internal Battery Management System (BMS) and provide data over the system management bus (SMBus) such as charge, voltage, and health. In order to interface this data line to the robot, the galvanic isolation board is required. See Section 4.6.1.2 for more details. Lastly, the batteries have a shutdown feature in case more than 10 A of current is pulled continuously for a specific duration of time. All of these features allow for more data and protection, ultimately leading to increased reliability.

4.5.6 Power Distribution

The WALRUS Rover uses an 80 A contactor for switching main power. Power for the contactor's coil is routed directly from the batteries through an E-Stop switch and a solid state relay on the proprioceptive sensor board. Running coil power through both of these systems in series allows both the user and software to power off the robot. When the software issues a power off command to the proprioceptive sensor board all electronics including the computer will continue running on backup power. The ability to remotely power off the robot is required if our main batteries enter a shutdown state and load must be completely removed to bring them back online. Backup power is also routed through the E-Stop switch assuring all systems are powered down when it is pressed. From the contactor, main power is delivered to all motor controllers, an external payload port and the Vicor DC/DC converter which powers the remaining electronics and payload ports. Figure 37 shows a high level overview of power distribution within the WALRUS Rover.

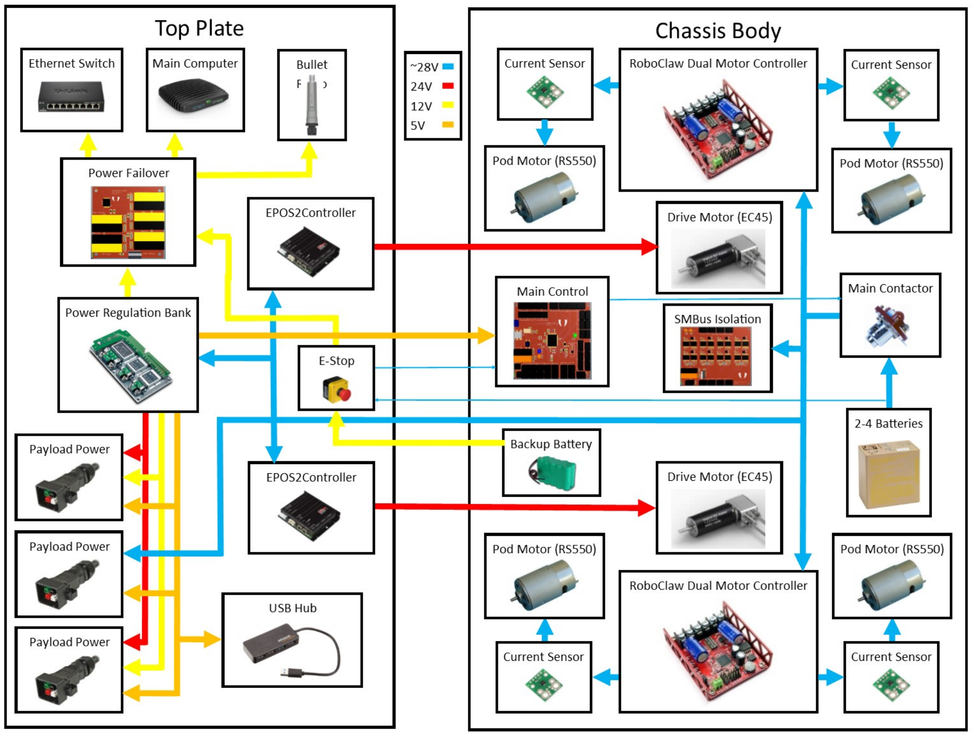


Figure 37: Power distribution inside the WALRUS Rover. Red lines represent 24 V power, yellow lines 12 V power, orange lines 5 V power and blue lines raw battery power. Ground wires are omitted for clarity.

4.5.7 Failover Board

Having a backup power system is useful as it keeps main electrical components up and running for a small duration of time in case of a power fault. This provides user feedback as to what happened. The most likely reasons for a power fault would be if the BB-2590 batteries shutdown from over current protection or if there was a lose connection somewhere on the main power line. The failover board can be seen in Figure 38.

The two inputs are for the main Vicor supply and a backup NiMH battery. The three outputs power the main computer, the Ethernet switch, and the Bullet WiFi radio. The output is connected to the Vicor supply as long as the main batteries are running, otherwise the output is swapped to the backup battery. This switching is done with a transistor and a diode as shown in Figure 39. The complete schematic can be seen in Figure 97 in Appendix B.

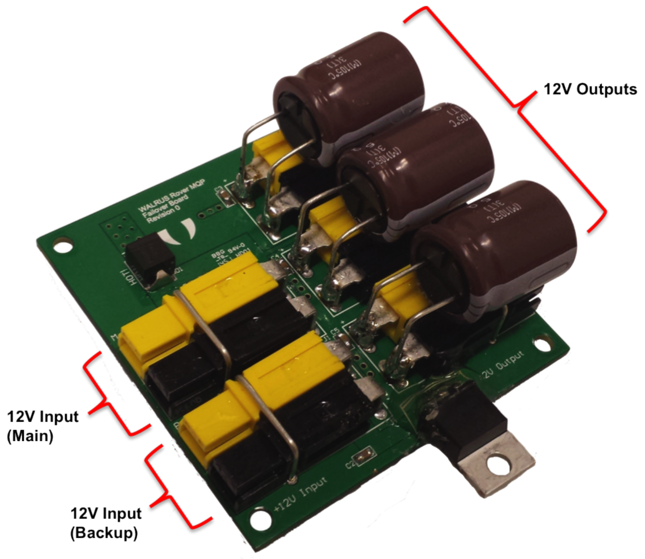


Figure 38: Failover Board.

If the main 12 V supply is active, the P-channel MOSFET is pulled high and turned off. No current will flow from the backup battery because of the open circuit, and the 12 V output will be powered off of the main supply through the diode. If the main supply fails the MOSFET will be pulled low by R_1 and turn on, allowing current to flow from the backup battery. The diode ensures that this current only flows to the output.

Capacitors C_1 - C_5 are standard decoupling caps to remove noise on the input and output lines, but C_6 - C_8 are used as bulk capacitors to help ensure the components this board powers stay on during a power source transition. These capacitors store a significant amount of energy, and discharge that energy during a transition to give some extra time for the backup battery to kick in. This is necessary for the main computer as it is very sensitive to quick changes in voltage. If the voltage on the computer drops too low too quickly it will reboot. In order to find a value for the capacitors we setup a simple experiment. With the computer running very little processing and thus drawing little current, the failover board was tested, and the output voltage was measured on an oscilloscope. This gave us the $\frac{\mathrm{d}V}{\mathrm{d}t}$ the computer could handle without rebooting. Estimating the total current draw from the failover board to be 6 A with a conservative safety factor allows for the calculation of the capacitors as follows.

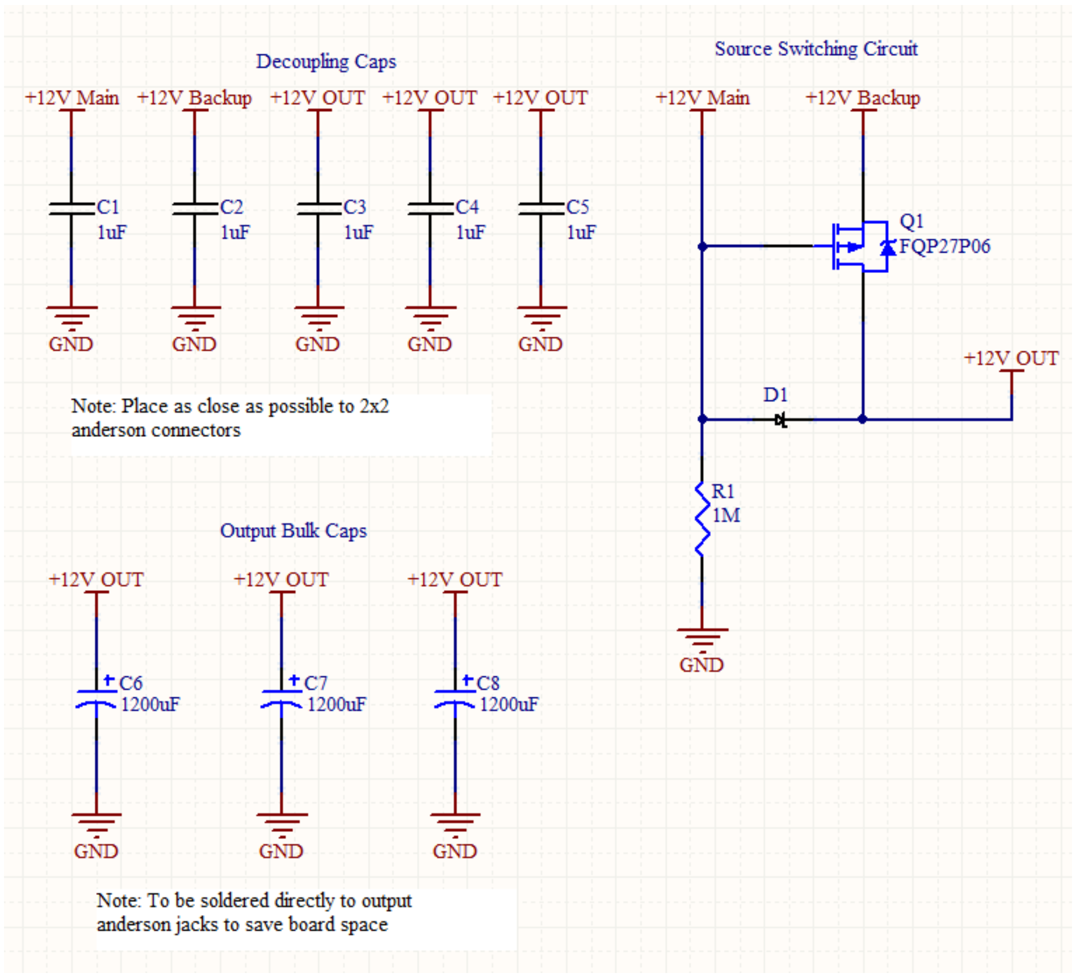


Figure 39: Failover Board Schematic.

$$I_{max} = C_{total}\frac{\mathrm{d}V}{\mathrm{d}t} \tag{7}$$

$$6\,\mathrm{A} = C_{total}(1740\,\mathrm{V/s}) \tag{8}$$

$$C_{total} \approx 3500\,\mu\mathrm{F} \tag{9}$$

Failover Board Bulk Capacitance

With three outputs, this total capacitance could be divided by three to find the individual capacitance values of each to be roughly 1200 µF with an additional safety factor.

4.6 Sensing and System Monitoring

This section discusses all of the sensors and custom hardware designed for both proprioceptive and exteroceptive sensing. Proprioceptive sensing consists of temperature, humidity, battery data, water leaks, flipper position, and flipper motor current. Exteroceptive sensing includes harmful gases, temperature, and humidity.

4.6.1 Proprioceptive Sensing

Proprioceptive sensing is crucial to ensure that both the rover and the user are aware of the rover's overall health and state. Monitoring all major electrical components lessens the likelihood of damaging a significant system and makes it possible to perform autonomous tasks such as pod position syncing which would be impossible otherwise.

4.6.1.1 Proprioceptive Sensor Board

The heart of the internal sensor system is the proprioceptive sensor board. An annotated diagram of this board is shown in Figure 40. This is the main breakout board for all internal sensors in the robot and interfaces these sensors which communicate over a wide range of protocols such as I^2C, 1-wire, and analog, to the main computer over USB. Table 4 summarizes the sensors that are broken out from this board.

Sensor	**Part Number**	**Protocol**	**Quantity**
Component Temperature	DS18S20	One-Wire	10
Ambient Temperature	SI7020	I^2C	1
Ambient Humidity	SI7020	I^2C	1
Pod Encoder	MA3	Analog	4
Pod Current	ACS714	Analog	4
Water	SEN11304P	Digital	6
Battery	BB-2590	SMBus	4
Pressure	MPL3115A2	I^2C	1

Table 4: Internal Sensor Summary.

All of the sensors listed in this table are off board and connected using latching Molex SL connectors except for the on board SI7020 ambient temperature/humidity sensor. The complete schematic can be seen in Figure 98 in Appendix B. The major parts of this board which will be reviewed are the microprocessor circuitry and PCB layout, the USB circuitry and PCB layout, and the on-board temperature/humidity sensor circuitry and PCB layout.

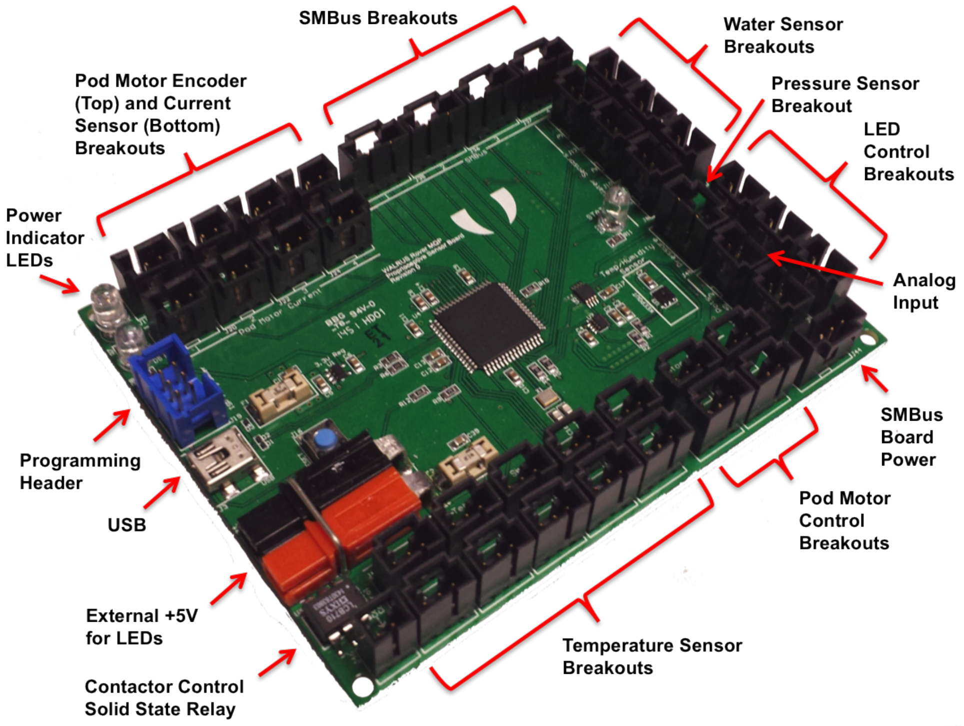


Figure 40: Proprioceptive Sensor Board.

The chosen MCU for this and all other custom boards is the Atmel AT90USB1286 MCU. The AT90USB1286 is a 5 V powered 64 pin MCU that communicates over USB. It has a total of 46 I/O, built in 8 channel ADC, as well as SPI and I^2C lines. We seleced this specific MCU for the proprioceptive sensor board because it has enough GPIO and data protocols for our sensing needs, communicates over USB making it easy to interface with our main computer, and is used on the Arduino Teensy++ 2.0 making it simple to get up and running because of its open source hardware and software resources available. Figure 41 shows the MCU circuitry schematic. While this MCU is overkill for other boards designed, it made more sense to use the same MCU for everything as it simplifies designs and keeps software more consistent between boards.

Like any standard MCU design, decoupling capacitors C_8 - C_{14} are used on all voltage pins to eliminate noise which could hinder the performance of the IC. Note that UCAP is the output of the internal 3.3 V USB regulator, and AREF is the ADC voltage reference. R_3 and R_6 are used for USB line termination and will be reviewed later. The reset button J_{16} does not require a pull-up resistor because there is an internal pull-up. This specific MCU

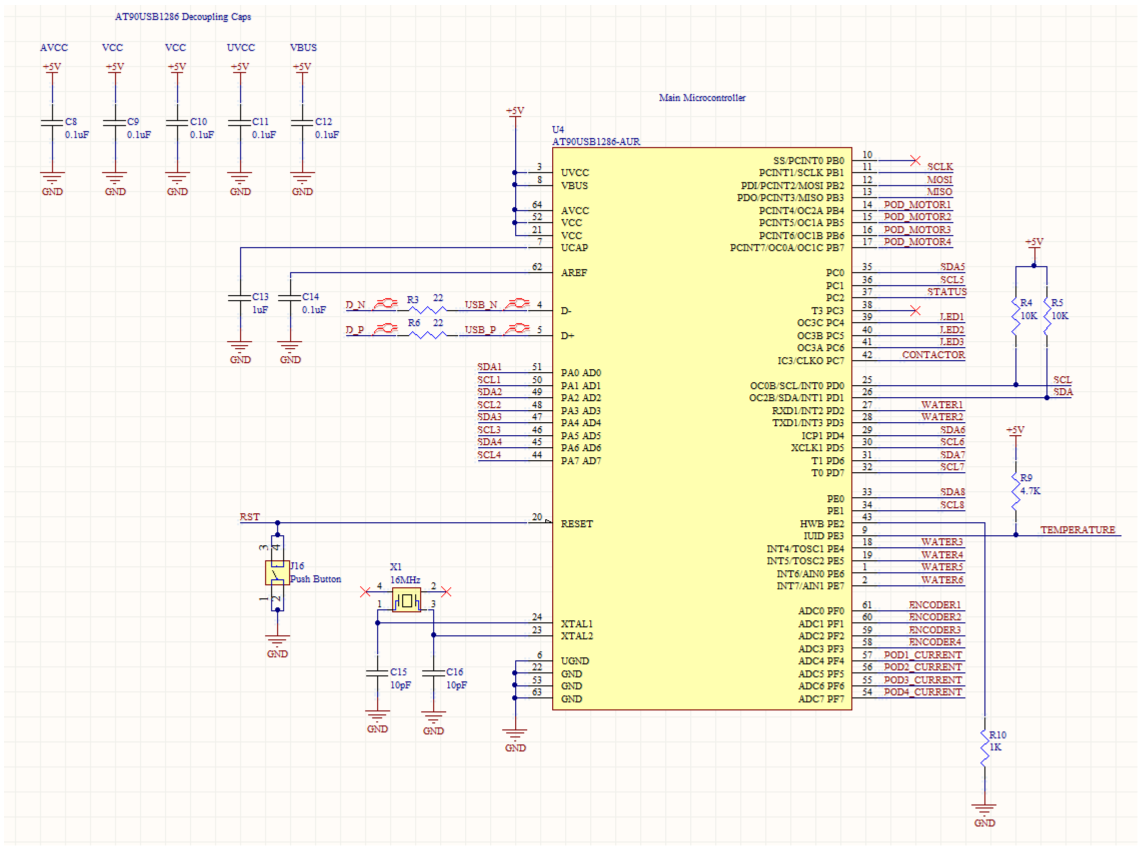


Figure 41: AT90USB1286 Microcontroller Schematic.

operates off of a 16 MHz crystal, and C_{15} and C_{16} are used for decoupling of the crystal X_1's signal. R_4 and R_5 are the I^2C line's pull-up resistors, and R_9 is the pull-up resistor for the temp sensors 1-wire bus. R_{10} pulls the HWB pin down, allowing the MCU to execute the boot loader after reset.

The remainder of this design is just a breakout of all the sensors going to their respective protocols. For example, the analog encoders connect to ADC pins PF0 - PF3 and the 1-wire temp sensors connect to pin PE3. To ensure ideal performance, many steps had to be taken on the PCB layout design for this MCU, which are carried through every other board that utilizes the MCU. Figure 42 shows the entire board layout. Red represents the top copper layer and blue represents the bottom copper layer. The top layer also acts as a ground plane, and the bottom layer acts as the 5 V power plane to help reduce noise. Figure 43 shows a zoomed in view of just the MCU layout.

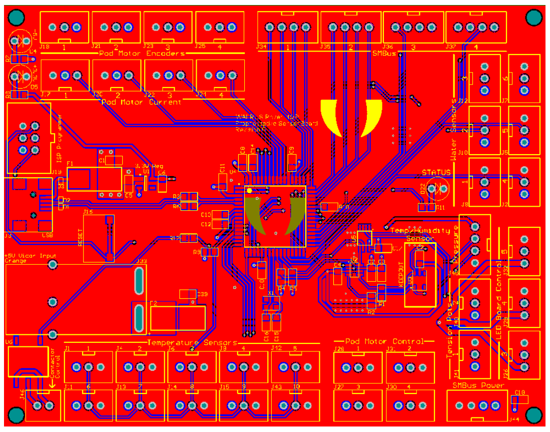


Figure 42: Complete Proprioceptive Sensor Board Layout

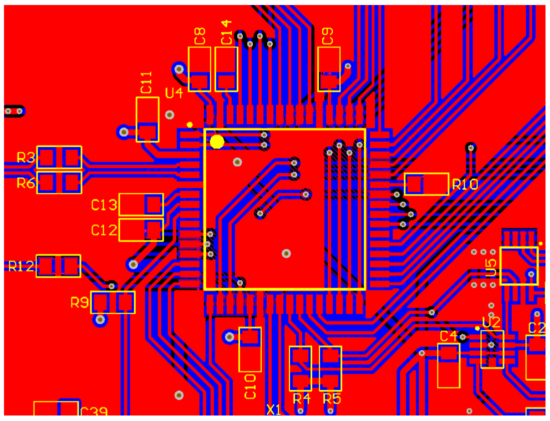


Figure 43: MCU Layout

The two biggest design considerations for this board's layout are the physical placement of decoupling capacitors and the ground plane. The decoupling caps are placed as close as possible to the MCU to help shunt noise to ground. If the caps are placed farther away, then more noise can be superimposed from the traces between the cap and MCU. Having a solid ground plane is always important for ICs, especially something as complicated as an MCU. While the top ground plane does connect underneath the IC in the four corners, it is a very poor connecton. Therefore stitching vias are used on the bottom layer to connect this inner square of ground plane to the rest of the ground plane outside the MCU layout.

Figure 44 shows the USB circuit designed for this board. To help ensure hardware is not damaged for power faults off the main 5 V line, the USB circuitry has both overvoltage and overcurrent protection built in. Bidirectional zener diodes D_1-D_3 clamp the data and VBUS lines to 5 V, and a 500 mA fuse F_1 breaks if the USB ever attempts to pull more than 500 mA of current which is the max for USB 2.0 spec. C_1 is used for decoupling any high frequency noise which could affect digital ICs.

Similar to the MCU design of this board, there were a few design considerations on the PCB layout as well. High speed data lines such as USB have specific design rules which must be followed to ensure that the digital signals do not become too degraded due to noise. USB differential lines D- and D+ must avoid sharp corners, be minimized in length, stay parallel, and most importantly be impedance matched to 90 Ω, which is the standard for a USB cable. Impedance matching is very important as differences in impedance for high speed signals can result in reflections. There are many models which have been

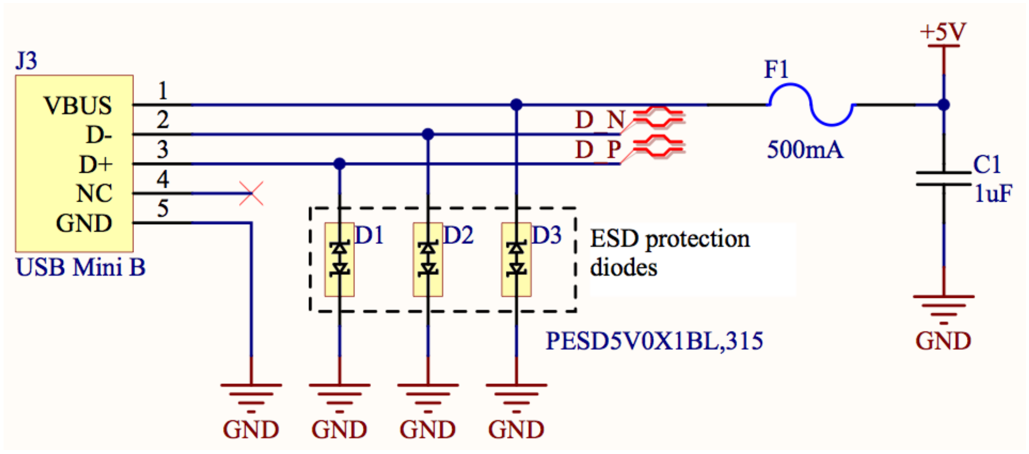


Figure 44: USB Schematic.

developed to calculate the impedance of a differential pair. Equation 10 taken from the "Design Guide for High-Speed Controlled Impedance Circuit Boards" [29] is one method to calculate differential impedance. ϵ_r is the relative dielectric constant of the board material, and w, d, h, and t are geometric dimensions as defined by Figure 45.

$$Z_d = \frac{174}{\sqrt{\epsilon_r + 1.41}} \ln\left(\frac{5.98h}{0.8w + t}\right)\left[1 - 0.48e^{-0.96\frac{s}{h}}\right] \tag{10}$$

Differential Impedance

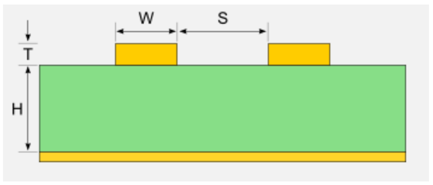


Figure 45: Differential Impedance Calculation Dimensions.[30]

Assuming $\epsilon_r = 4.2$, $h = 4.7$ mils, and $t = 1.4$ mils, all of which are standard PCB manufacturing values, you can find a combination of trace width w and trace separation s that end up resulting in an impedance of $90\,\Omega$. Simply guessing and checking values resulted in $w = 8$ mils and $s = 11.7$ mils giving a differential impedance of $Z_d = 90.026\,\Omega$. Since USB is a fairly robust protocol, slight deviations in impedance will not significantly hinder the communication performance, and therefore this combination was deemed fit. Figure 46 shows a zoomed in view of the USB layout.

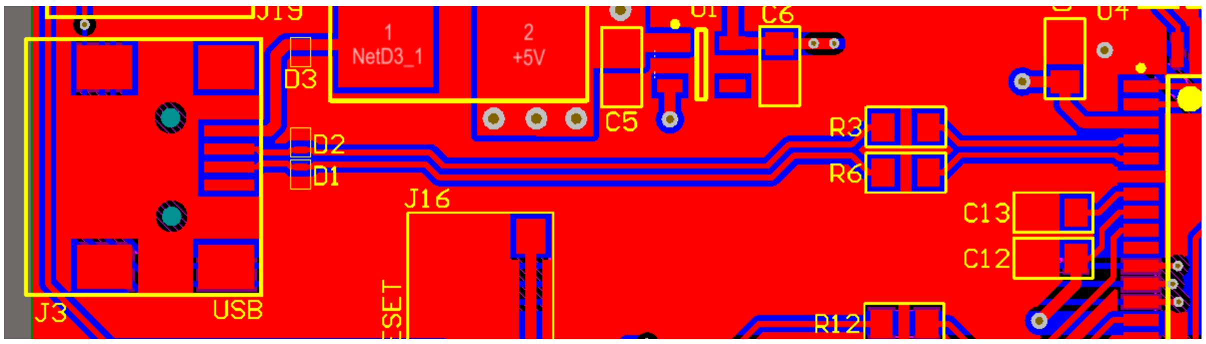


Figure 46: USB Layout.

From the figure it can be observed that trace lengths of D- and D+ are the same, they stay parallel, and they avoid sharp corners and vias. In addition, the ESD diodes D_1 and D_2 are kept as close as possible to the USB header. This ensures that if there was an ESD occurrence, the signal would be shunted very early on, rather than causing transient issues down the line. The last design consideration was with the addition of R_3 and R_6. These resistors act as line terminators which help prevent additional signal reflection. They can be compared to a damper in a spring mass damper system, and are very low value because a high value would cause too large of a voltage drop across them, resulting in a failure to communicate. This circuit and layout designed are the same for every other board using USB communication.

The last major part of this board is the ambient temperature/humidity sensor design. The schematic for this piece is shown in Figure 47.

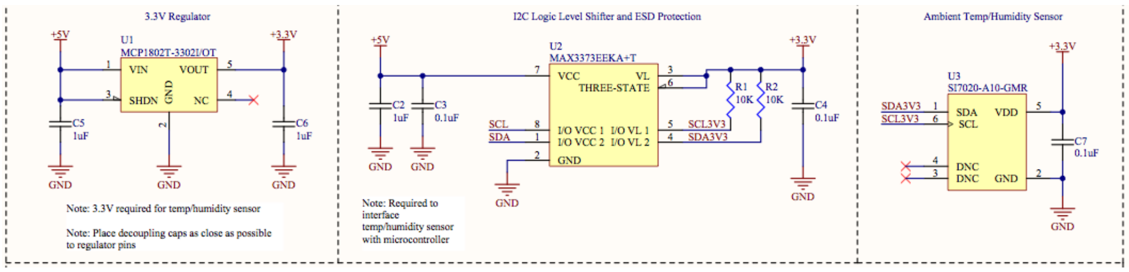

3.3V Regulator
+5V
U1
MCP1802T-3302I/OT
VIN
VOUT
SHDN
GND
NC
+3.3V
C5
1uF
C6
1uF
GND
Note: 3.3V required for temp/humidity sensor
Note: Place decoupling caps as close as possible to regulator pins
I2C Logic Level Shifter and ESD Protection
U2
MAX3373EEKA+T
VCC
VL
THREE-STATE
I/O VCC 1
I/O VCC 2
I/O VL 1
I/O VL 2
SCL
SDA
SCL3V3
SDA3V3
C2
1uF
C3
0.1uF
R1
10K
R2
10K
C4
0.1uF
Note: Required to interface temp/humidity sensor with microcontroller
Ambient Temp/Humidity Sensor
U3
SI7020-A10-GMR
SDA
SCL
VDD
DNC
GND
C7
0.1uF

Figure 47: Temperature/Humidity Sensor Schematic.

The sensor we chose was the SI7020 from Silicon Labs because it communicates of I^2C and takes care of both temperature and humidity in one package. The only downside is that it runs off of 3.3 V logic, meaning some additional circuitry is required to interface it with the 5 V MCU. A 3.3 V regulator U_1 is used to power the sensor, and a bidirectional logic level shifter U_2 is used to translate 5 V to 3.3 V digital logic signals and vice versa between the MCU and sensor. Additional pull-up resistors on the 3.3 V side must be used in addition to the pull-ups on the 5 V side, and decoupling capacitors must be used on all voltage pins for U_1, U_2, and U_3.

Since the SI7020 requires 3.3 V power, we took some additional design steps on the PCB layout. To reduce noise and keep traces neat, we used a technique called star routing. Star routing refers to breaking out multiple traces from a single point rather than relying on "daisy chaining" or other messy routing methods. This is shown in Figure 48. The 3.3 V power is star routed to various points from a single via circled in black next to C_4.

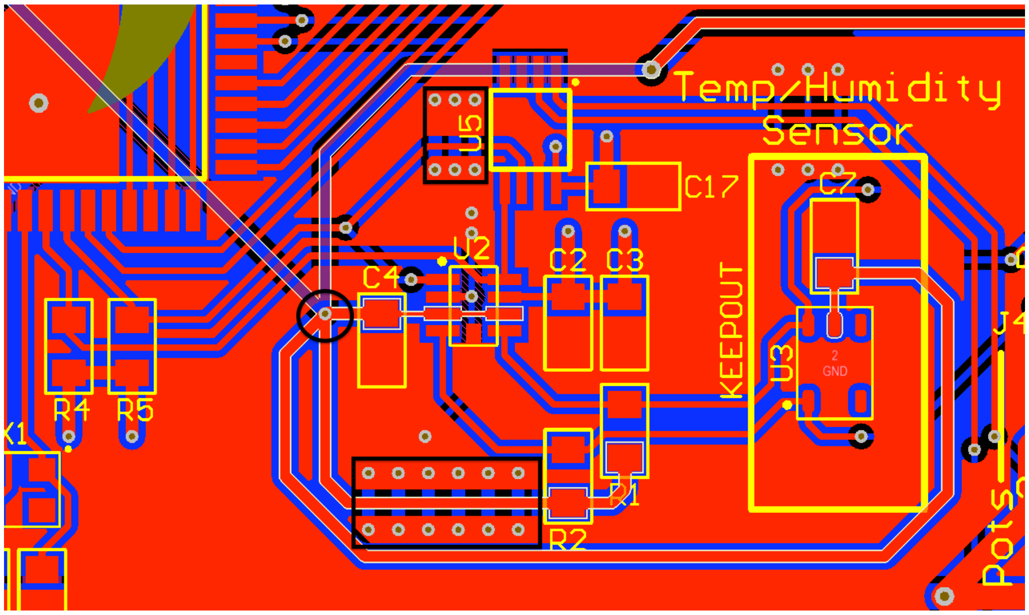


Figure 48: Star Routing of Temp/Humidity Sensor.

Another technique used is via stitching. Via stitching is a technique used to tie together larger copper areas on different layers which helps maintain a low impedance. The more surface area a power plane uses, the more effective it is at removing noise. When we made the ground plane of this board, there were small isolated islands that could not connect to the rest of the ground plane. Via stitching was used to connect these small islands to the plane and can be seen in Figure 48 in the boxed areas.

4.6.1.2 SMBus Galvanic Isolation

While the built in SMBus line on the BB-2590 batteries is very useful in determining the overall status of our batteries, it does require some external circuitry in order to interface it with the proprioceptive sensor board. Each battery consists of two separate 16 V packs, each with their own individual SMBus line for data and clock signals. When they are wired in series, the upper 16 V pack's data and clock lines become referenced to a ground 16 V higher than the lower pack's data and clock lines. Attempting to interface the upper pack's SMBus lines without any additional isolation circuitry would cause damage. Therefore, the SMBus galvanic isolation board is required, as shown in Figure 49.

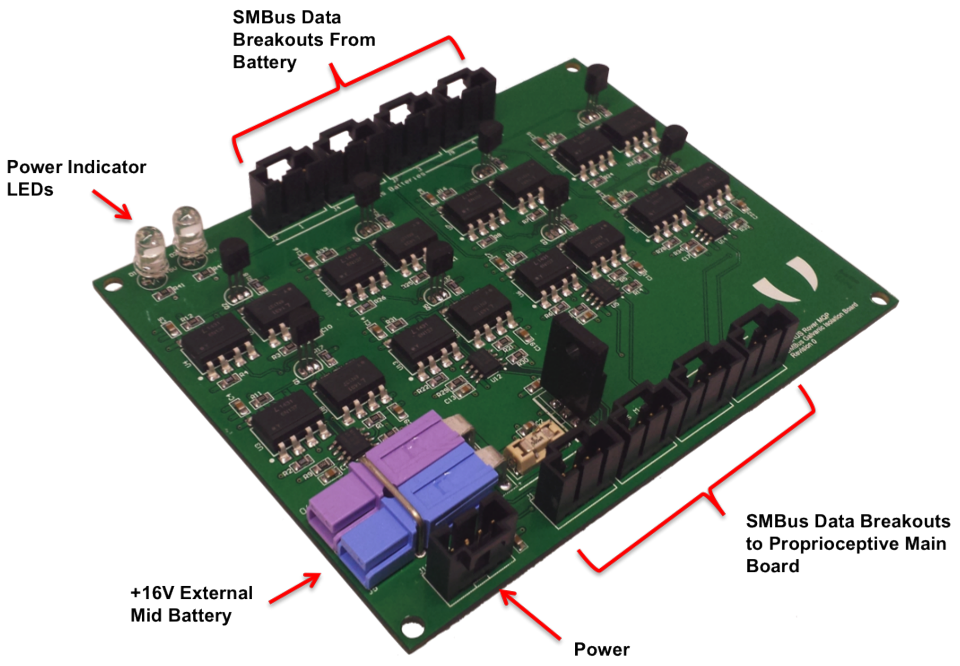

SMBus Data Breakouts From Battery
Power Indicator LEDs
SMBus Data Breakouts to Proprioceptive Main Board
+16V External Mid Battery
Power

Figure 49: SMBus Galvanic Isolation Board.

There are latching Molex SL connectors for the data (SDA) and clock (SCL) lines from each battery, as well as Molex SL connectors for the isolated 5 V outputs that connect to the propricoeptive sensor board. This board utilizes high speed optocouplers to isolate the two different potential levels. See Figure 99 in Appendix B for a complete schematic.

In order for the isolation circuitry to work, the higher referenced SMBus needs the midpoint ground reference point, which is taken externally. This power input is also protected with a 500 mA fuse in case of an overcurrent fault. The ICs are powered via the proprioceptive board with another Molex SL connector which is already fused on the proprioceptive sensor board.

Generally speaking this type of isolation is a solved problem. The most common solutions are by using transformers or optocouplers. NXP Semiconductors has a reference document with many different variants of galvanic isolation for I^{2}C data lines (which is similar enough to the SMBus protocol) depending on the required operational frequency of the circuit. One of the designs, which is what the galvanic isolation board is based on, is shown in Figure 50. Note that this schematic only shows the isolation of one SDA signal. An exact copy would be required for the SCL signal as well.

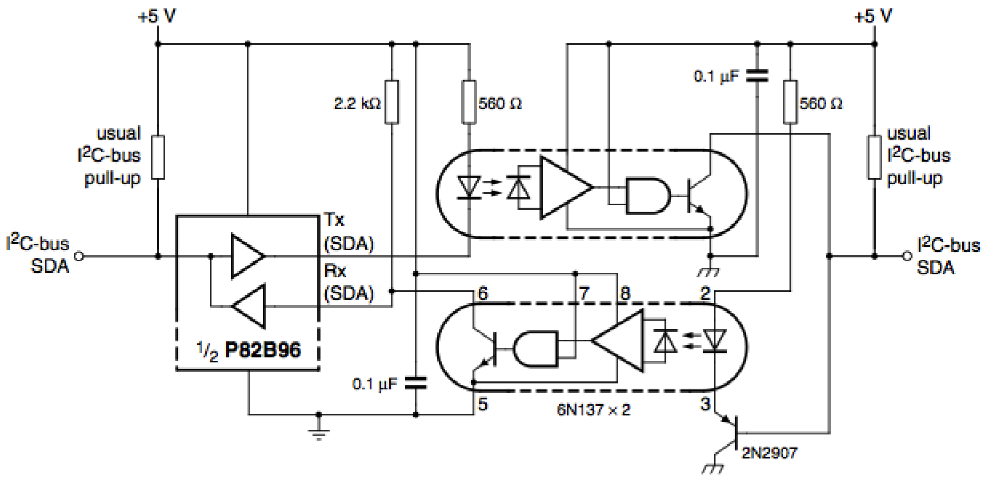


Figure 50: NXP Galvanic Isolation Schematic.[31]

This design utilizes high speed optocuplers to isolate between the two different potential levels. First, the SDA signal is fed into a signal splitter which separates the transmitting and receiving signals on this line. The split signals are sent to two optocouplers for isolation. There are two different 5 V and ground sources. The 5 V and ground symbol on the left comes from the proprioceptive sensor board, and the 5 V and ground symbol on the right comes from the battery side. The battery ground on the right is the midpoint ground of the battery, which is 12 V relative to the proprioceptive board ground. The 5 V on the right is relative to the right ground, which means it is at 17 V relative to the proprioceptive board's ground. The right 5 V can be achieved using a voltage regulator where the ground point of the regulator is attached to the right ground point. Standard pull-up resistors are used on the signal lines, decoupling capacitors are used on the power pins, and series current limiting resistors are used to protect the internal LEDs of the optocouplers.

The PCB layout of this board was fairly straightforward as no high speed or high power routing was required. Standard design practices of keeping decoupling capacitors close to power pins, avoiding sharp angles, and adding power planes were used. It should be noted that this is a four layer board since there are two different ground and 5 V power nets. While star routing or other PCB routing techniques could have been used, entire planes made routing much simpler.

4.6.1.3 Temperature and Water Leak Monitoring

Many of the components such as motors, the DC-DC converter, and the main computer dissipate significant amounts of heat. When run for long durations and in harsh environments these devices can overheat and become permanently damaged. Therefore, monitoring these components' temperature levels are critical to avoid damaging scenarios. The team decided to use the DS18S20 1-Wire temperature sensors from Maxim Integrated because of their accuracy, range, and use of a 1-wire interface which requires fewer wires to route. A total of ten sensors are placed throughout the robot to monitor the temperature of the four pod motors, the two pod motor controllers, the two main drive motors, the Vicor DC-DC converter, and the top plate. These locations are shown in Figure 51 with yellow dots. We also have the ability to access the main computer's internal temperature sensor. Each of these ten sensors route back to the proprioceptive sensor board which provides them with power. Lastly, the proprioceptive board contains an on board temperature sensor to monitor the ambient internal temperature of the rover.

4.6.1.4 Water Leak Monitoring

Whenever the WALRUS Rover is put into a body of water, the chance of electronics failure is significantly greater than when tested on land. If water were to come in contact with electronics components while running, shorts could occur, ultimately leading to catastrophic failure. Water sensors help provide immediate feedback as to whether the chassis is leaking. A total of six SEN11304P water sensors are placed strategically throughout the rover in the most likely spots where leaks could occur. These locations are shown in Figure 51 with blue dots. Note that the two sensors shown on the treads are actually underneath in the battery bay compartment. While this feedback is not too helpful in the case of a significant leak, it still provides an additional layer of protection against slower leaks.

4.6.1.5 Flipper Position and Motor Current Monitoring

Determining the position of the flippers makes it possible to sync their positions together, allows for predetermined driving positions to be executed, and makes the user aware of their positions for increased situational awareness. Measuring pod motor current is important for protection of the motors and their controllers. If the pods were to crash into each other,

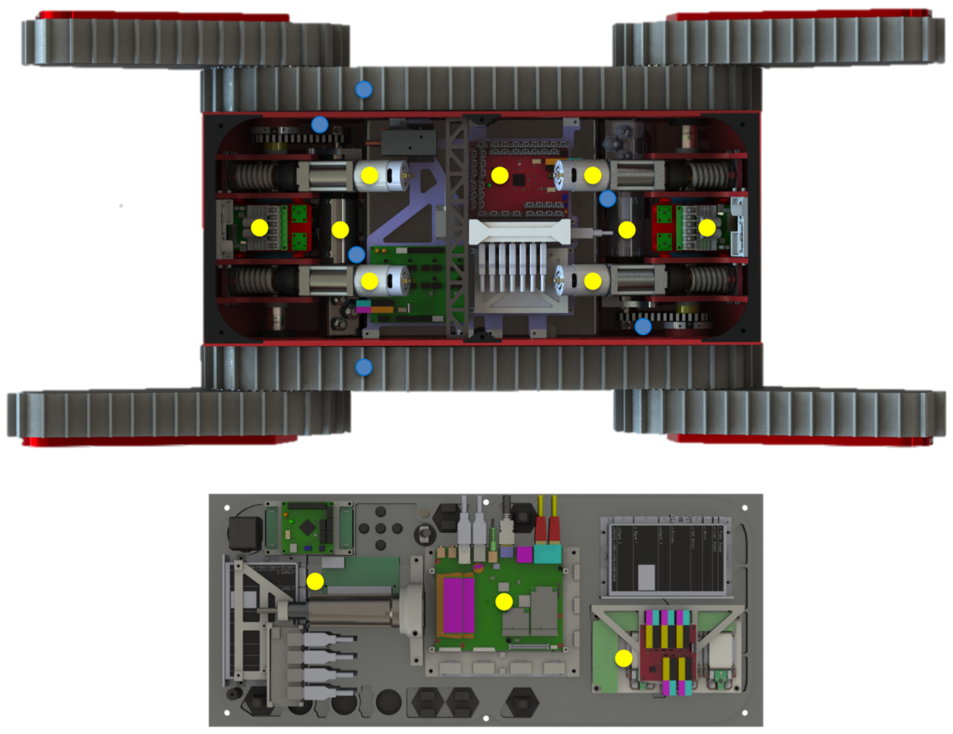


Figure 51: Chassis and Top Plate Temperature (Yellow) and Water (Blue) Sensor Locations.

or overexert themselves, they could damage themselves. Knowing the current of the motors allows for automatic shutdown in these cases.

We chose the MA3 absolute analog encoders from US Digital to monitor flipper position. They swing from 0 to 5 V as they turn from 0 to 360°. To measure pod motor current, we chose the ACS714 bidirectional current sensor from Pololu. This sensor has a range of -30 to 30 A and outputs an analog signal between 0-5 V with 2.5 V representing no current flow. It has a resolution of 66 mV/A which given our situation will suffice.

4.6.2 Exteroceptive Sensing

Exteroceptive sensing is particularly important for a search and rescue robot because it provides useful data on how dangerous the environment is to humans. Given the data, a human team could make a judgement call on how safe it would be to enter. Moreover, exteroceptive sensors allow for autonomous features that require data on the environment in order to properly manipulate it or navigate around.

4.6.2.1 Environmental Sensor Board

The WALRUS Rover has one custom sensor board on the boom payload which contains all of its environmental sensors. There are a significant number of possible sensors that could be used, but given the scope of this project we decided to limit the sensors to measure dangerous gas levels, temperature, and humidity. The gasses that can be sensed are carbon monoxide (CO), liquefied petroleum gas (LPG), compressed natural gas (CNG), and hydrogen (H). The environmental sensor board can be seen in Figure 52. This board can also be used as a breakout board for additional sensors if desired. I^2C and four generic I/O which can be used as analog inputs or digital inputs/outputs are broken out on the bottom. A complete schematic can be found in Appendix B with Figure 101.

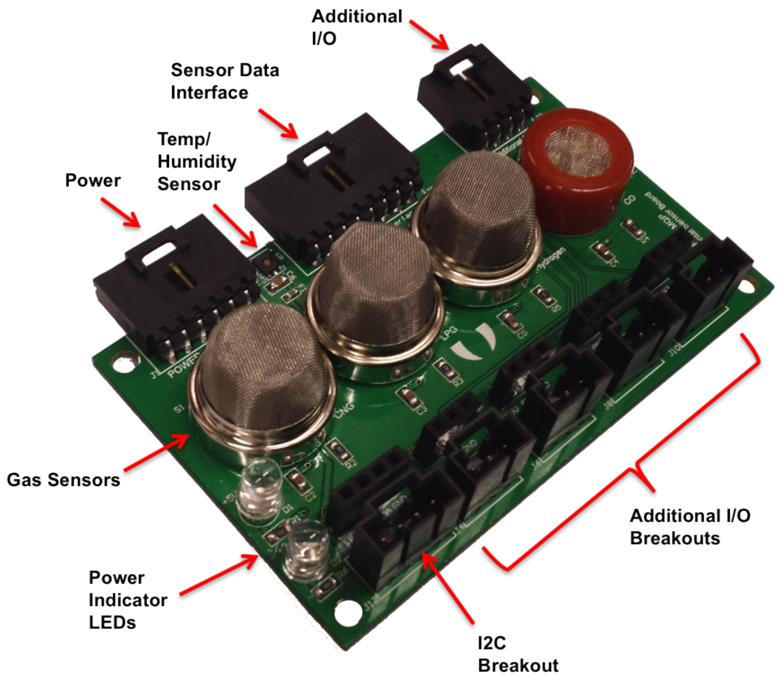


Figure 52: Environmental Sensor Board.

Since this board contains gas sensors it cannot be completely sealed off from water, but only made splash resistant. Therefore, water damage is a viable concern. To help prevent this board from being damaged, an inline fuse to the board's main power line is put in place. This fuse is a positive temperature coefficient (PTC) resettable fuse so that the board will automatically come back on once it's dry and the fuse has reset.

Another important design consideration was with the gas sensors. Their readings are very sensitive to temperature, and therefore have built in heaters to help heat the sensor up to a fairly constant temperature. This dissipation of heat could cause some bias in the temperature sensor's readings, and therefore the temperature and gas sensors are spaced away from each other. Furthermore, this dissipation of heat ends up drawing a few hundred milliamps of current when all four sensors are running, and therefore the sensors' voltage supply is drawn from the external supply of the main board, rather than the USB's supply.

4.6.3 Diagnostics Board

The addition of a diagnostics interface to the WALRUS Rover gives the ability for a user to obtain sensor data, put the rover into different demo modes or positions, and visualize any potential failures or warnings right from the robot. This interface is made possible with the diagnostics board, shown in Figure 53.

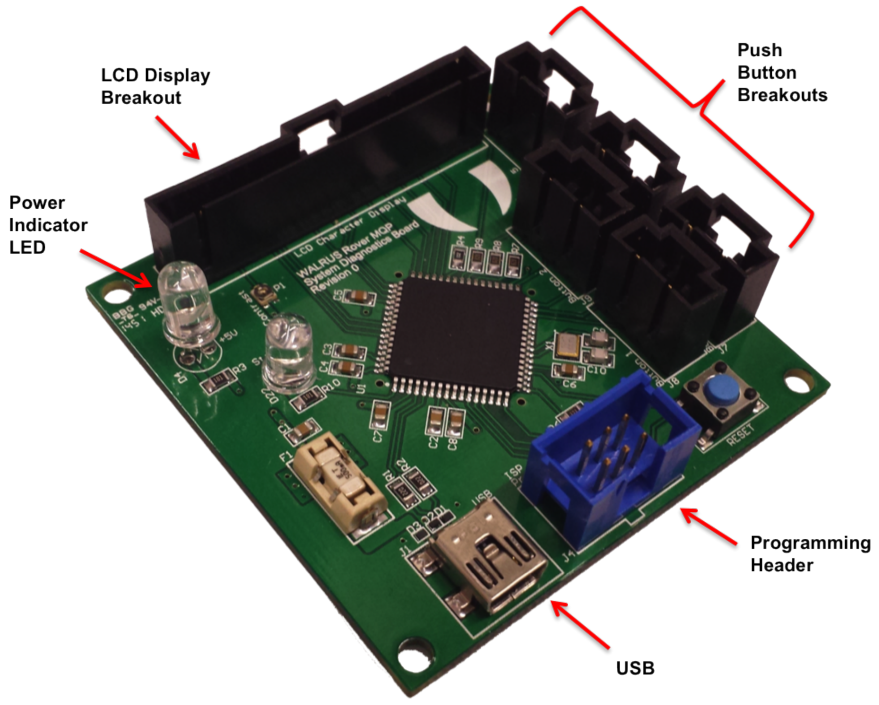


Figure 53: Diagnostics Board.

This board is a simple breakout for an LCD display and five push-buttons that can be used to navigate the display, and communicates over USB to the main computer. It contains the same Atmel AT90USB1286 MCU, USB, and overcurrent and overvoltage protection circuitry as the proprioceptive sensor board. The push-buttons also contain LEDs, which are connected to the PWM pins on the MCU. This means the button LEDs can light up different patterns depending on the state of the rover, giving an additional visual aid to the user. See Figure 100 in the Appendix for a complete schematic.

4.6.4 Position and Camera Feedback

There are three cameras onboard the main chassis; they point front, back, and down (but tilted slightly forward). Each of these looks through a port cut into the underbody with a polycarbonate panel sealed to the underbody with both a gasket and marine sealant. The forward and backward views are helpful in standard navigation, while the bottom view is primarily useful during complex maneuvers like self-lifting and also during aquatic search. The camera we chose was the Microsoft LifeCam HD camera.

We designed an LED board for increased situational awareness at night. These boards are placed with all cameras to illuminate dark areas and aid the vision of the cameras. The LED boards shown in Figure 54 use a 5x5 array of high power LEDs and are controlled with a PWM signal by the proprioceptive sensor board via latching Molex SL connectors. See Figure 102 for a complete schematic.

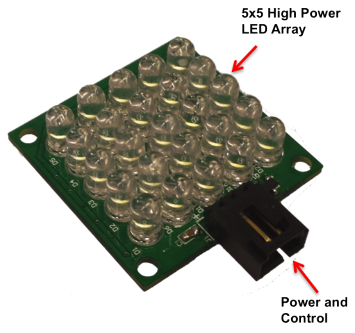


Figure 54: LED Board.

4.7 Software Architecture

In order to create an open architecture and build upon pre-written and tested software, all of our software is written on top of the Robot Operating System (ROS)[32]. While not a true operating system, ROS is a set of open source software libraries and tools that makes it easy to write reusable software modules for robotics applications. ROS systems are made up of a number of processes (nodes) that can run on the same computer or many different computers. Nodes communicate by exchanging messages through a multiple publisher, multiple subscriber system over TCP or UDP. Messages are defined in a language independent format that is compiled into language specific serialization and deserialization implementations. Stable ROS support exists for code written in C++, Python, or Common Lisp.

All of our code is available open source on GitHub [33], allowing us to share code with other teams at WPI and within the robotics community. We also employed continuous integration, using Travis CI, and pull requests on GitHub for quality assurance. ROS runs on the main computer which also hosts a web server to handle the user interface. While the main computer is currently the only computer inside the WALRUS Rover, ROS' networking capabilities make it easy to add additional computing resources or payloads that integrate seamlessly with the rest of the system. This section will describe the components that make up WALRUS Rover's software and how they operate with and communicate through ROS. A high level overview of the WALRUS Rover's software architecture can be seen in Figure 55.

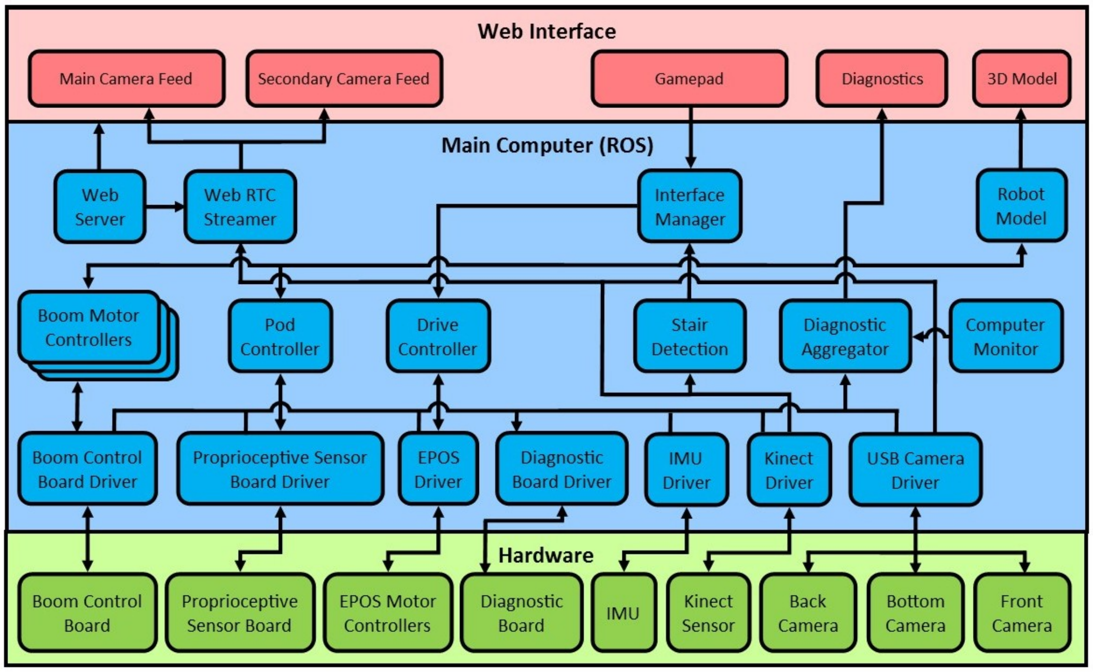


Figure 55: High level software overview.

4.7.1 Firmware

Our custom embedded control boards (the proprioceptive sensor board, diagnostic board, and boom board) are powered by Atmel AT90USB1286 microcontrollers (MCU). This MCU uses Atmel's own AVR instruction set architecture, the same used by the popular Arduino microcontrollers. The AT90USB1286 in particular is used by the PRJC Teensy 2.0++

microcontroller development system which is programmed in an Arduino environment. Using this MCU allows us to use PRJC's existing Arduino hardware core for the platform with only limited modifications. This enables us to use existing Arduino Libraries, greatly speeding up development of embedded software. In addition, we loaded each board with a bootloader packaged with the Lightweight USB Framework for AVR's (LUFA) library developed by Dean Camera enabling us to program the boards over USB.

The software on each embedded board uses ROS Serial to communicate with ROS on the main computer. ROS Serial is a collection of ROS packages that allow code running outside of the ROS environment to publish and subscribe to ROS topics by communicating with a server through serial or network sockets. Fortunately, a package (`rosserial_arduino`) already exists for using ROS Serial on Arduino compatible platforms. Though the ability to directly publish and subscribe to topics is extremely powerful, a number of factors had to be considered when developing communication protocols for each board. First, the MCUs have very limited memory, meaning they can only support a finite number of publishers and subscribers. In addition, care must be taken when using message types with unbounded arrays, such as `DiagnosticArray`, that can easily outgrow the MCU's memory capacity. Space concerns coupled with the significant overhead of sending and receiving messages led us to to develop custom messages for each board for communicating with a driver running on the main computer, which would then communicate with the rest of the ROS system.

4.7.1.1 Proprioceptive Sensor Board Firmware

The proprioceptive sensor board has multiple functions that require different data rates. In order to reduce the overhead of sending messages, data with similar rate requirements are published together in a single message. The board operates on two types of data: periodic data including motor control, motor feedback and sensor data; and aperiodic data including LED control, power control, and status and error reports. Motor control and feedback (encoder positions and current readings) are required to operate at a rate of 50 Hz, while sensor data primarily intended for human consumption can be transmitted as slow as 1 Hz. Three messages are used for these periodic communications: one for motor control from the PC to the board, one for motor feedback from the board to the PC and one for sensor data from the board to the PC. For aperiodic communications only a single additional message type is used with two topics for bidirectional data. The communication flow between the proprioceptive sensor board and PC is visualized in Figure 56.

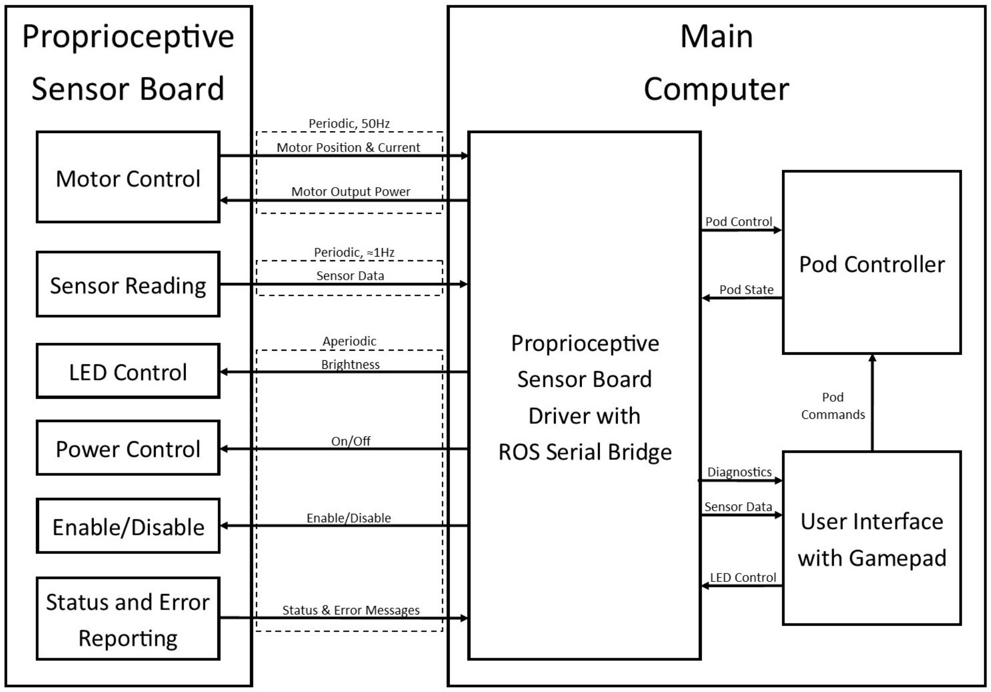


Figure 56: Flow of data between the proprioceptive sensor board and the main computer.

The most significant task for scheduling proprioceptive sensor board operations is pod motor control because it requires speed and strict timing. Every 20 ms (50 Hz) the board must receive motor control data, issue motor control pulse changes, read current and position for each pod, and send a feedback message. In order to meet this requirement all other operations had to be scheduled between these high speed data updates. Since the MCU is not powerful enough to permit the use of a real-time operating system (RTOS), scheduling is done statically. In addition to high speed motor operations, the board also has to process control messages and send sensor feedback. Attempting to send sensor feedback at the same rate as motor feedback is impractical as reading all of the sensors controlled by the main board far exceeds 20 ms. Instead, a state machine is used to stagger sensor readings between high speed data updates. Between each update, a limited set of sensors is read. After all sets are read, the data is published together. Each set of sensor readings must be fast enough to run between high speed updates. If a control messages is received between updates the message is processed and the scheduled sensor reading is postponed until the next update. This process assures non-deterministic aperiodic control messages do not cause the board to miss high speed updates. In addition, errors and status changes also block sensor readings to make time to send the appropriate data in a control message to the main PC. A flowchart showing the flow of operations in the main board can be seen in Figure 57

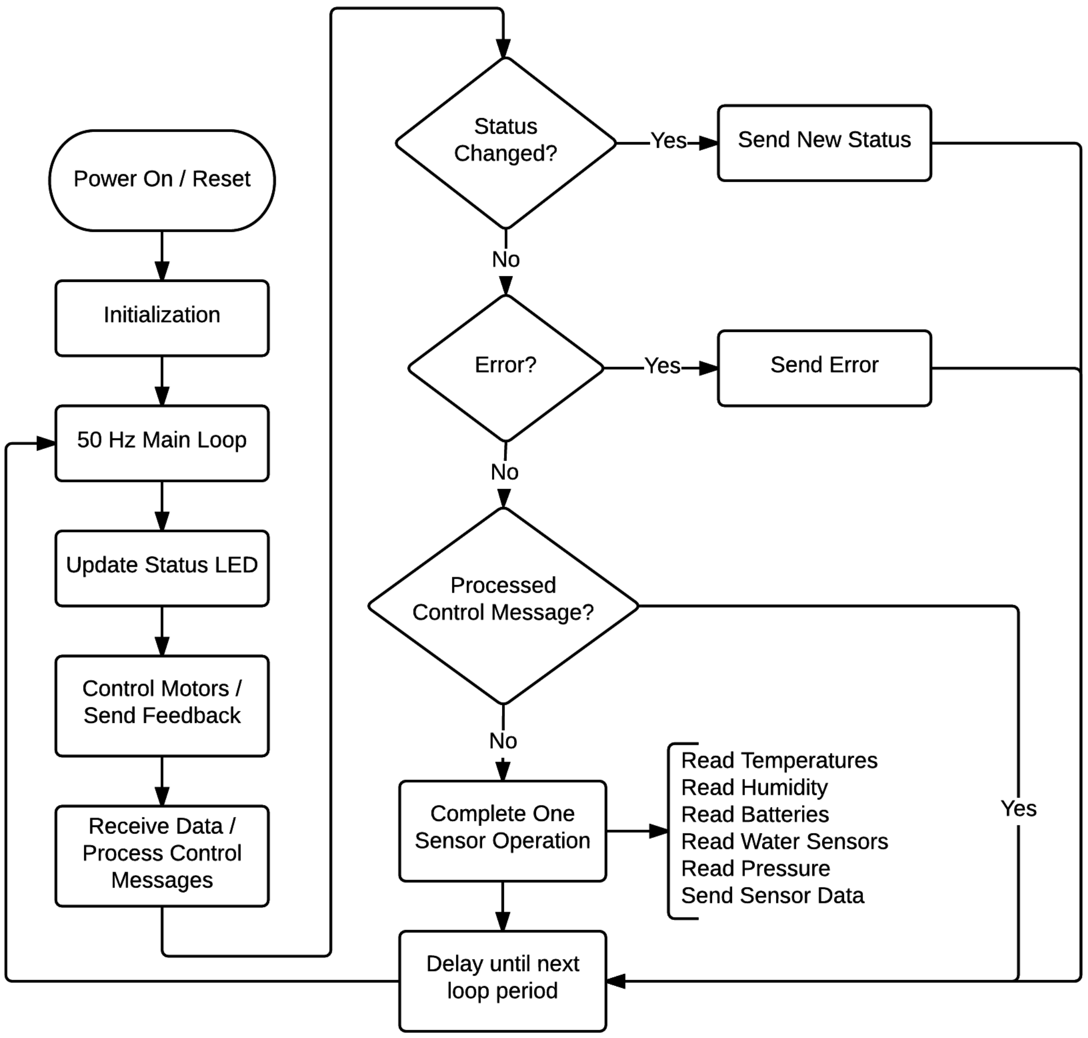


Figure 57: Flow of control within the proprioceptive sensor board.

4.7.1.2 Diagnostic Board Firmware

The diagnostic board software is the simplest of our three boards. The only operations running on the board's processor are printing to the screen and debouncing input buttons. When a button is pressed (or held for a brief period) a `DiagnosticButton` message is published indicating which button was pressed and how. The driver running on the main computer responds by sending a `DiagnosticScreen` message containing what can be thought of as a frame buffer (a simple character string) to be displayed on the screen.

All of the screen's displays and menus are created and processed on the main computer. This structure makes programming much simpler because complex custom messages do not need to be created to request and send specific data between the board and main computer. Additionally, it is difficult for the MCU to process larges amounts of diagnostic data, such as messages from the diagnostic aggregator, with its limited speed and memory. Offloading the processing to the main computer affords us and future developers greater flexibility when implementing and upgrading diagnostic screen features. These benefits come at the cost of greater latency between button press and screen refresh. A visualization of the diagnostic board's functionality is shown in Figure 58.

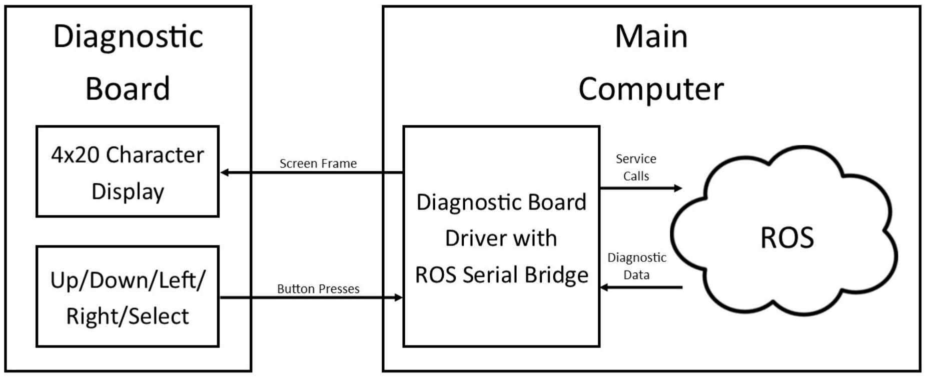


Figure 58: Overview of diagnostic board operations.

4.7.1.3 Boom Board Firmware

The boom control board operates in the exact same manner as the proprioceptive sensor board. It uses similar messages for communications with its respective driver on the main PC as well as using the same scheduling process. A communications diagram can be seen in Figure 59. The only significant difference between the two boards is the speed of publishing sensor data. Since the boom board boasts less sensors that also take less time to read, sensor data from the boom board is published much faster at a rate of 10 Hz.

4.7.2 Joint Controls

Control of actuators and associated feedback sensors is managed inside ROS through a set of packages collectively called ROS Control. These packages allow programmers to write modular interfaces and controllers for manipulating robot hardware. The ROS Control systems revolves around a robot description file. This file, written in the Universal Robot Description Format (URDF), describes how parts of the robot, like arms and wheels, move with respect to each other. It also contains links to meshes that together form a dynamic

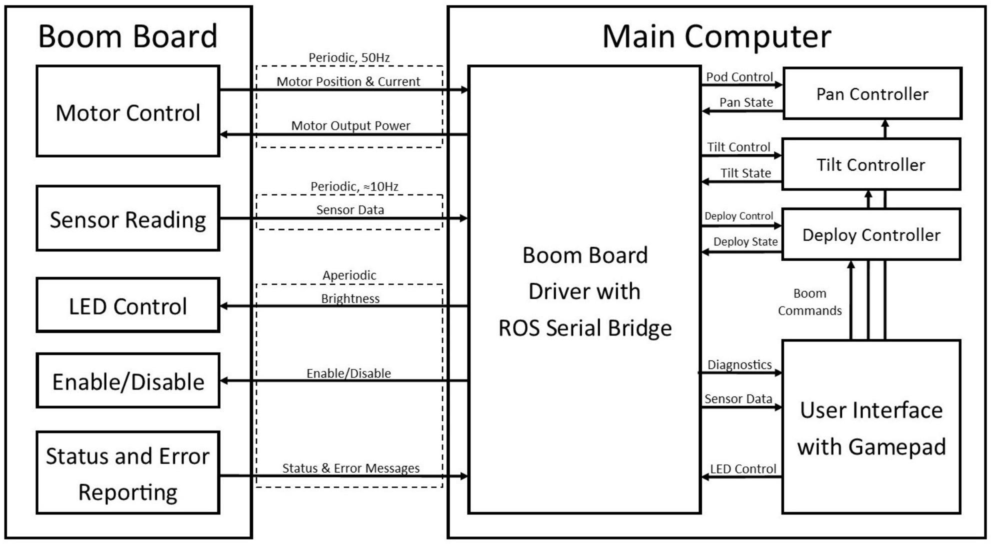


Figure 59: Flow of data between the boom board and the main computer.

3D model of the robot. Robot 'joints', as they are called in URDF, represent interfaces between two parts of a robot. Actuators can be attached to these joints and configured to load custom hardware drivers and controller plugins for driving them through ROS. This section describes the controllers used by the WALRUS Rover for controlling its actuators. The main benefit of ROS control is that it exposes the physical state of the robot in a standard way that allows different packages to interact with hardware regardless of the specific hardware driver or controller used. An overview of ROS Control can be seen in Figure 60.

4.7.2.1 Drive

The main drive motors are controlled by Maxon EPOS/2 7010 motor controllers. These controllers are very intelligent and can perform PID control on-board removing the need for a complex controller running under ROS. A such, the main drive actuators are described as using 'Velocity' joint interfaces and simple controllers that pass velocity commands directly from the user interface to hardware.

4.7.2.2 Pods

The WALRUS Rover uses custom controllers for driving pods. In order to properly synchronize pod movements and prevent collisions, all pods are driven by the same controller. The pod controller allows user interface and supervised autonomy software to control both the position of the pods and their raw output power when required. Since the main board

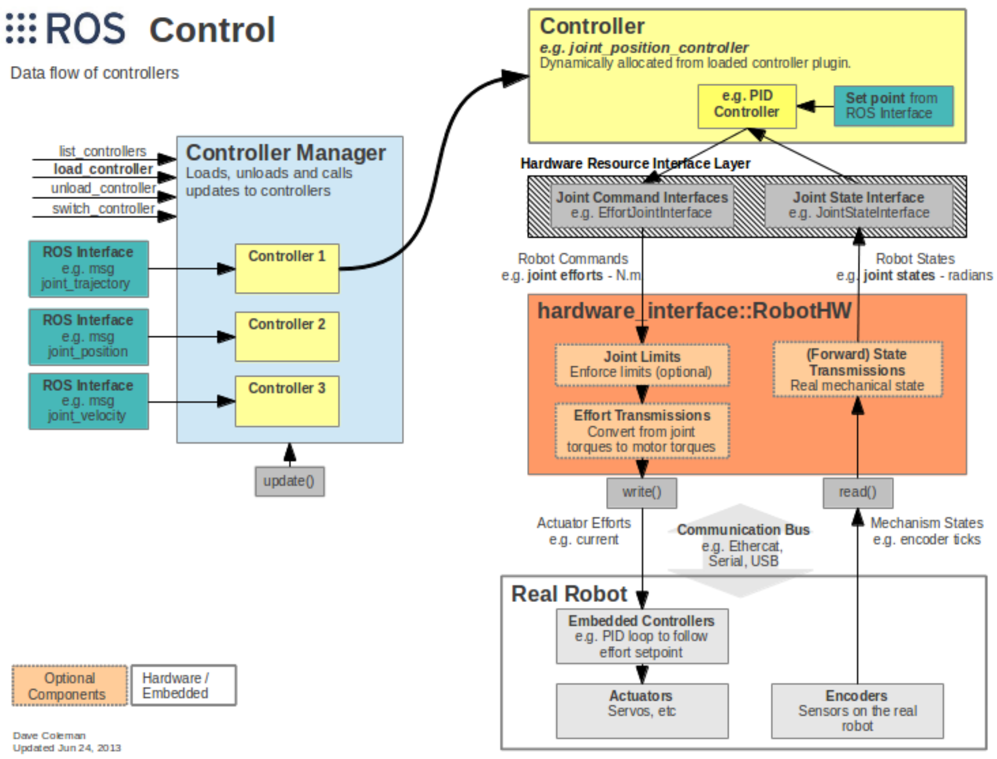

ROS Control
Data flow of controllers
list_controllers
load_controller
unload_controller
switch_controller
Controller Manager
Loads, unloads and calls updates to controllers
ROS Interface e.g. msg joint_trajectory
ROS Interface e.g. msg joint_position
ROS Interface e.g. msg joint_velocity
Controller 1
Controller 2
Controller 3
update()
Controller
e.g. joint_position_controller
Dynamically allocated from loaded controller plugin.
e.g. PID Controller
Set point from ROS Interface
Hardware Resource Interface Layer
Joint Command Interfaces e.g. EffortJointInterface
Joint State Interface e.g. JointStateInterface
Robot Commands e.g. joint efforts - N.m
Robot States e.g. joint states - radians
hardware_interface::RobotHW
Joint Limits Enforce limits (optional)
(Forward) State Transmissions Real mechanical state
Effort Transmissions Convert from joint torques to motor torques
write()
read()
Actuator Efforts e.g. current
Communication Bus e.g. Ethercat, Serial, USB
Mechanism States e.g. encoder ticks
Real Robot
Embedded Controllers e.g. PID loop to follow effort setpoint
Actuators Servos, etc
Encoders Sensors on the real robot
Optional Components
Hardware / Embedded
Dave Coleman
Updated Jun 24, 2013

Figure 60: Overview of ROS Control[1].

is only capable of controlling the pod motors' raw output level, the pods actuators are described as 'Effort' joint interfaces. When position control is required, a PID loop is run inside the pod controller on the main computer. The pod controller sends motor output power through ROS Control to the main board driver which communicates with the main control board through ROS Serial. The main board responds in kind with pod position and motor current feedback.

4.7.2.3 Boom

Custom controllers are also used for controlling boom deploy, pan and tilt actuators. These auctions operate in a similar way to the pods as software might want to control both the raw output power of the motors or the joint positions. These actuators also are described as 'Effort' joint interfaces and each use a separate instance of a custom `EffortPositionController` which supports both forms of control. Boom motor output control is likewise sent through ROS Serial to the boom board which responds with position and current feedback.

4.7.3 Operator Controls and Feedback

We designed and built the operator interface using modern web technologies in order to create a cross platform and cross browser application. We used Rosbridge the primary communication channel between the robot and the browser, while WebRTC was used to allow for performance, real-time video and audio streaming. We built the interface using modern CSS3 for layout and the AngularJS MVC framework for simplifying application code.

4.7.3.1 Rosbridge and roslibjs

Rosbridge[34] is a collection of tools and protocols that allow for easy communication between a ROS system and a web browser or system without ROS support. It comprises of a server that accepts WebSocket or TCP connections and then proxies ROS topics and services over these connections. Rosbridge defines a protocol for use over these connections that is made up of JSON messages that convey a client subscribing to a topic, publishing or receiving a message, or calling a service. Message data is represented as simple JSON objects so messages can be easily interpreted or created. Rosbridge also provides a Javascript library, called roslibjs[35], that can be used to easily communicate with a Rosbridge server over WebSockets. This allows us to communicate with ROS nodes from a web browser as if it was another node. We use Rosbridge to publish control information from a joystick and other options on the UI as well as subscribe to robot state and diagnostics information. Other libraries are also provided for rendering 2D and 3D content ROS content on the web; these are used to display a 3D view of the robot model with very little work. While Rosbridge allows for easy communication with the robot, it does not scale well with large amounts of information such as audio and video streams in real-time over a variable bandwidth link.

4.7.3.2 WebRTC

In order to reliably stream audio and video in real-time we used an emerging technology named WebRTC[36] to stream video from the robot and stream audio bidirectionally. WebRTC provides a platform that allows for bidirectional audio and video streams with variable bitrate encoding as well as access to a system's audio (mic and speakers) and video (webcam) devices from Javascript code without the use of a plugin. It currently is supported out of the box by Chrome, Firefox and Opera on both personal computers, tablets and smart phones. WebRTC uses a number of different codecs to stream media over UDP (default) or TCP directly between peers even if there is a server responsible for serving the webpage. This reduces latency as the media is sent directly between computers instead of being forwarded by a server. However, to set up the initial connection a server that both computers can talk to must exchange some configuration information between the two hosts over what is called the signaling channel as seen in Figure 61. While the connection between peers is defined in the WebRTC specification, the implementation of the signaling channel is left up to the application developer.

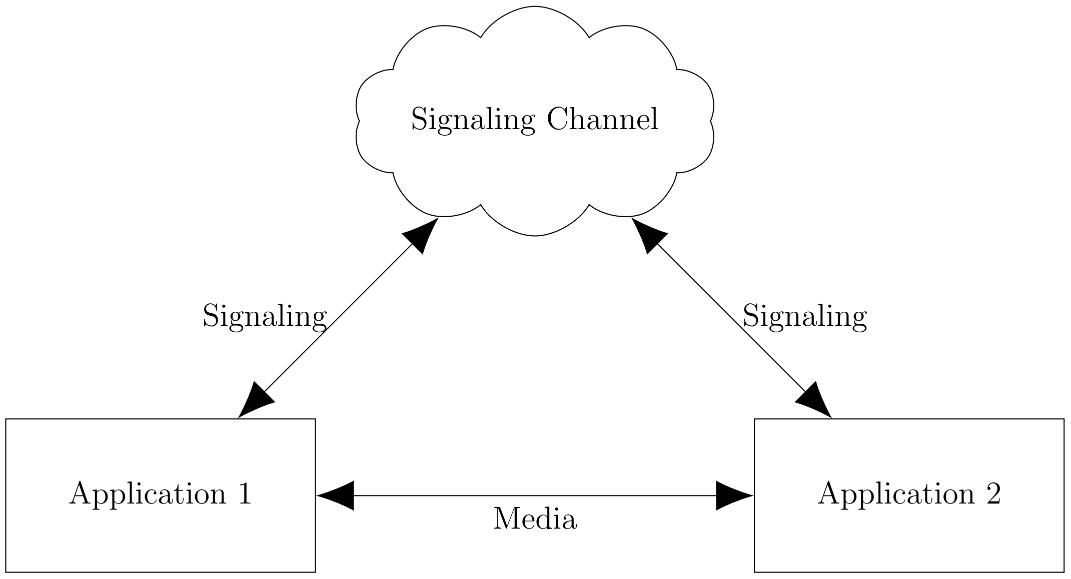


Figure 61: An overview of the connections involved in a WebRTC communication.

Although WebRTC peers are normally applications in a web browser using Javascript, the C++ API that the Chrome and Firefox implementation are built on top of is also available. This API can be used to develop native applications that communicate using WebRTC and also provides a Java API for use on Android and an iOS API. In order to use WebRTC to stream media from ROS to the our web interface we had to implement a bridge that would subscribe to ROS image topics and send the frames over WebRTC. In addition to acting as a peer, we also needed a program to setup a signaling channel between peers. We decided to combine the server and one of the robot peers into a single program. The program provides a web server that listens for WebSocket connections from a web browser. When the WebSocket is established it is used as the WebRTC signaling channel to send additional configuration information (such as what topic to subscribe to); the web server then calls functions on the robot WebRTC peer directly. This allows for a browser to subscribe to a video topic and take advantage of WebRTC. We publicly released the ROS node and supporting code as a ROS package called `webrtc_ros`[37].

4.7.3.3 AngularJS

AngularJS[38] is a web Model View Controller (MVC) framework that allows for quickly building web application with minimal code. It allows easy two way data binding of HTML content to models defined in Javascript, reducing the amount of work required to build an interactive application. AngularJS also provides facilities for building reusable code and contains a full dependency injection framework. The most common component that is used is a controller. Controllers are used to define properties and functions that can be called from the application view definition (the HTML content). Properties of a controller can be bound to text in a HTML document or the value of a form control, while functions can be defined as to events generated from the view, such as clicking a button. Services are singleton objects that can be injected into controllers and used throughout an application. One example of a built-in service is the `$http` service, which allows for making AJAX requests. While this could be easily done through browser APIs, the use of modularized components allows for easier testing using one of the many Javascript testing frameworks that AnuglarJS supports. The use of modularized code has also led to a number of open source libraries that are available online. We developed much of the code used by the operator interface and test applications as individual components in order to reduce code repetition and development time.

4.7.3.4 Angular Material

Angular Material[39] is a library that provides a web implementation of Google's Material Design Guidelines[40]. Material design provides a number of responsive, accessible components for use in web applications. This allows for applications built on top of the framework to be useful on a variety of form factors and platforms, including laptops, tablets, and phones. This allows almost any modern device that can connect to a Wi-Fi network to be used to control the WALRUS Rover. By building on top of AngularJS, Angular Material is able to provide components that can be easily used like normal HTML elements; this allows for the provided components to be used without any additional work.

4.7.3.5 Operator Interface Application

The operator interface uses the components described above in order to provide a full featured control interface inside of a web browser, as shown in Figure 62. It features two configurable video streams that can show any of the cameras connected to the robot, an interactive 3D model that shows the robot's configuration, and diagnostic information from many of the robot's sensors. It is built around AngularJS with a number of services that were used to provide abstractions over different hardware and communication channels. We developed services to interact with the gamepad API and the robot's ROS system. These allowed for developing modularized controllers for the UI while only having a single connection to the robot and gamepad. In addition to a service that allows ROS message exchange, services were built on top of it in order to provide simple to use interfaces for some ROS components, such as diagnostics and video streams.

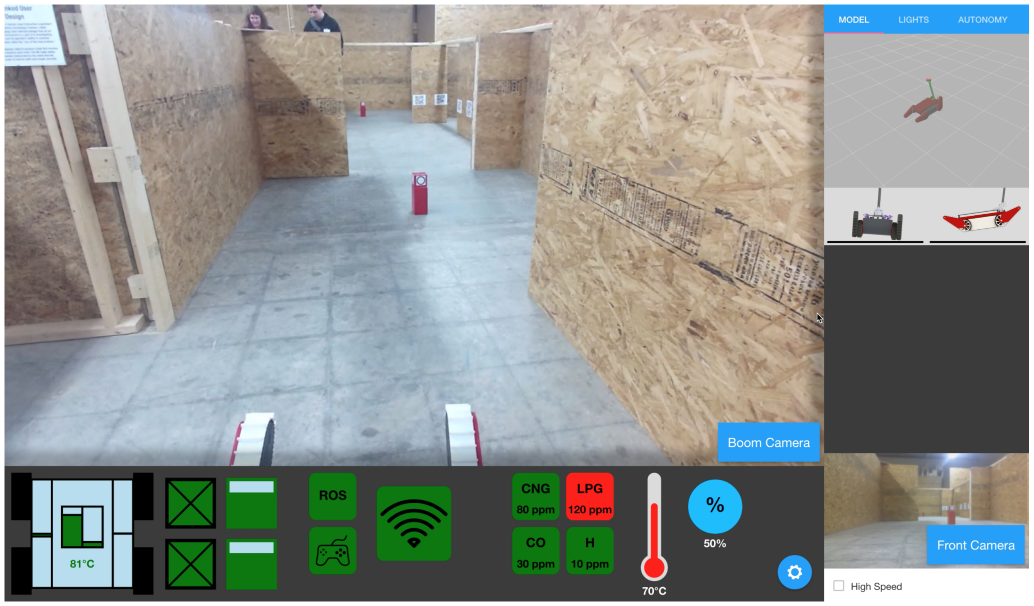
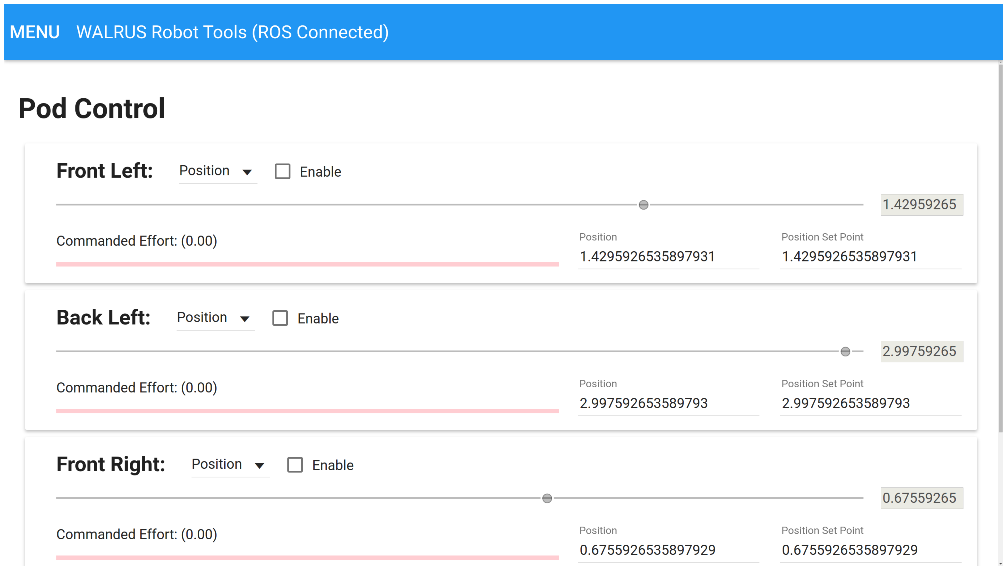


Figure 62: The normal operator interface web app layout.

4.7.3.6 System Test Application

In addition to developing an operator interface web application, we also built a test application to allow for easy testing and debugging of the system. It provides lower level interaction with the actuators and sensors than is available through the standard interface. It is made up of a number of small mini applications that communicate with different components of the robot. Figure 63 shows the interface that allows the user to control the pods directly using either position control or effort control. It also provides detailed feedback about the position controller. The test application also allows for interacting with the underlying ROS system directly. Figure 64 shows an interface to allow for subscribing to any message that is published. Other interfaces also exist for viewing the ROS log. We designed the the test application to be extended with additional functionality as it is needed.



MENU WALRUS Robot Tools (ROS Connected)

Pod Control

Front Left: Position Enable 1.42959265

Commanded Effort: (0.00)

Position 1.4295926535897931

Position Set Point 1.4295926535897931

Back Left: Position Enable 2.99759265

Commanded Effort: (0.00)

Position 2.997592653589793

Position Set Point 2.997592653589793

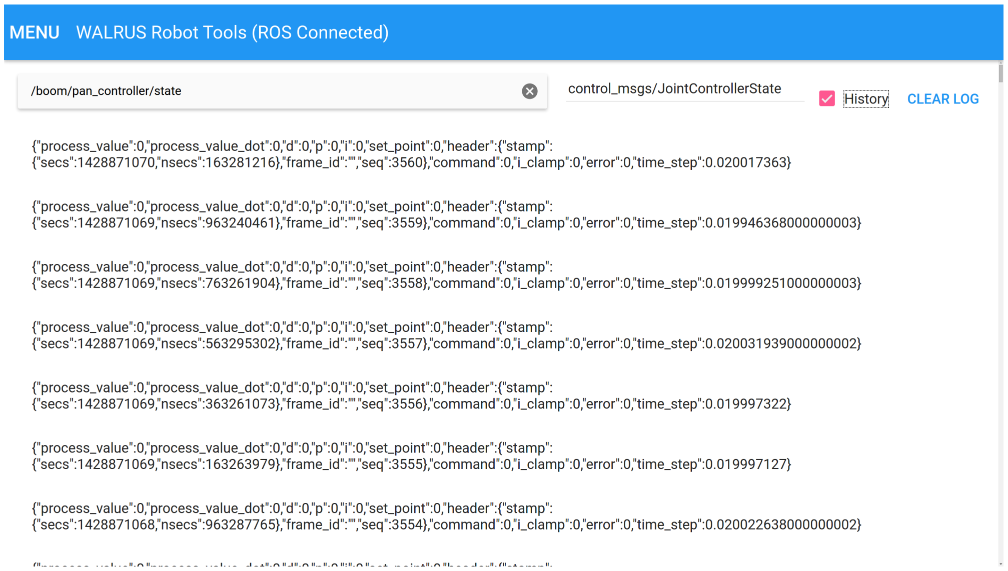
Front Right: Position Enable 0.67559265

Commanded Effort: (0.00)

Position 0.6755926535897929

Position Set Point 0.6755926535897929

Figure 63: The test web interface that allows for individual control of the pods.



MENU WALRUS Robot Tools (ROS Connected)

/boom/pan_controller/state

control_msgs/JointControllerState

History CLEAR LOG

```
{"process_value":0,"process_value_dot":0,"d":0,"p":0,"i":0,"set_point":0,"header":{"stamp":
{"secs":1428871070,"nsecs":163281216},"frame_id":"","seq":3560},"command":0,"i_clamp":0,"error":0,"time_step":0.020017363}

{"process_value":0,"process_value_dot":0,"d":0,"p":0,"i":0,"set_point":0,"header":{"stamp":
{"secs":1428871069,"nsecs":963240461},"frame_id":"","seq":3559},"command":0,"i_clamp":0,"error":0,"time_step":0.019946368000000003}

{"process_value":0,"process_value_dot":0,"d":0,"p":0,"i":0,"set_point":0,"header":{"stamp":
{"secs":1428871069,"nsecs":763261904},"frame_id":"","seq":3558},"command":0,"i_clamp":0,"error":0,"time_step":0.019999251000000003}

{"process_value":0,"process_value_dot":0,"d":0,"p":0,"i":0,"set_point":0,"header":{"stamp":
{"secs":1428871069,"nsecs":563295302},"frame_id":"","seq":3557},"command":0,"i_clamp":0,"error":0,"time_step":0.020031939000000002}

{"process_value":0,"process_value_dot":0,"d":0,"p":0,"i":0,"set_point":0,"header":{"stamp":
{"secs":1428871069,"nsecs":363261073},"frame_id":"","seq":3556},"command":0,"i_clamp":0,"error":0,"time_step":0.019997322}

{"process_value":0,"process_value_dot":0,"d":0,"p":0,"i":0,"set_point":0,"header":{"stamp":
{"secs":1428871069,"nsecs":163263979},"frame_id":"","seq":3555},"command":0,"i_clamp":0,"error":0,"time_step":0.019997127}

{"process_value":0,"process_value_dot":0,"d":0,"p":0,"i":0,"set_point":0,"header":{"stamp":
{"secs":1428871068,"nsecs":963287765},"frame_id":"","seq":3554},"command":0,"i_clamp":0,"error":0,"time_step":0.020022638000000002}
```

Figure 64: The test web interface that allows for the display of any ROS message.

4.8 Supervised Autonomy

The implementation of supervised autonomy features allows the the operator to focus less on maneuvering the rover and more on the task they are trying to accomplish. One way the WALRUS Rover is able to do this is by assisting the operator in climbing stairs; it does this by keeping the rover in the center of a staircase while climbing it. First we developed a stair detection algorithm [41] that extracts planes from a point cloud using a region growing method and then applies a number of successive RANSACs to estimate the stair model parameters. The RANSAC operations look at the distance between and dimensions of detected planes in order to estimate the rise, run, width, height and location of the stair case. We implemented this using the Kinect v2 sensor on the sensing boom payload, described in Section 4.9.1.

4.8.1 Stair Detection

In this section we present a description of our algorithm for stair model extraction, which is summarized in Figure 65. As with other works[42, 43, 44], the algorithm begins by segmenting planes from the raw point cloud. In order to simplify the estimation process, we compute a set of stair-relative axes from the extracted planes, shown in Figure 66. Distances are computed along these axes (rise along vertical, run along direction, and width along horizontal); this reduces ambiguity when the detected planes are not exactly parallel or the stair risers are not perpendicular to the treads. The algorithm focuses on detecting the stair risers, which are normally more visible than the treads from a low perspective. Horizontal planes are only used to refine the vertical vector estimation.

4.8.1.1 Initial Point Cloud Processing and Plane Segmentation

The first step in the detection of a flight of stairs is to segment the point cloud into different planes. In order to improve the efficiency of the algorithm, we down-sample the raw point cloud using the PCL VoxelGrid, which segments the point cloud into regions and replaces all of the points in the region with a single point that is an average them all. Additionally, we filter out points that are too far away or not in the center of the scene. Down-sampling and filtering the point cloud reduces the computation required for plane segmentation. Once the point cloud is down-sampled, we segment it into planes using the standard PCL normal estimation and region growing based on the point normals. PCL normal estimation works by fitting a plane to points surrounding the point in question for every point in the cloud. The region growing algorithm uses the extracted normals to segment the point cloud

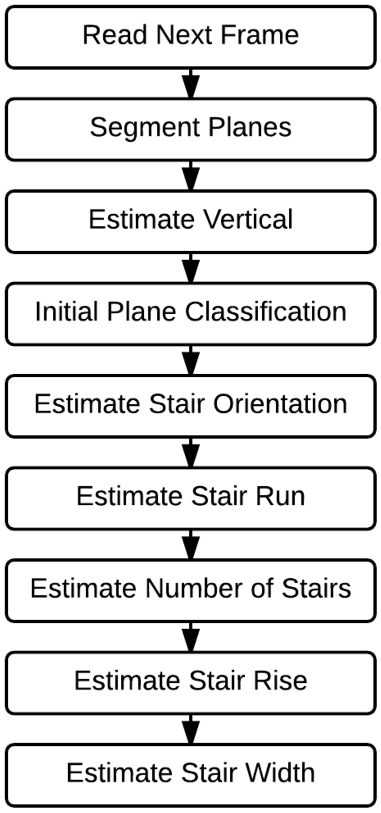


Figure 65: An overview of the process used to estimate the stair model

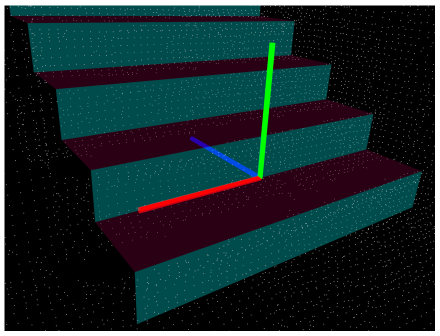


Figure 66: The axes used to describe elements that are on the stairs: horizontal (red), vertical (green), direction (blue)

into planes by attempting to find smooth regions by growing a region from a seed point until all the surrounding points exceed a smoothness threshold[45].

4.8.1.2 Vertical Estimation

In the next step of the process we estimate the vertical axis which will be used in the plane classification step to estimate which planes are horizontal or vertical in the scene. To estimate the vertical axis, we first seed the model with an initial estimate of the vertical vector based on the gravity vector from an IMU. We then analyze the 3D image and select planes that are within a threshold angle of the initial vertical vector based on the assumption that the majority of the planes near the estimated vertical are the ground plane, or parallel to the ground plane. Finally, we apply a RANSAC algorithm to the normals of the selected planes to estimate the normal of the ground plane (the vertical).

4.8.1.3 Initial Plane Classification

We bin the planes in order to improve the success rate of model estimation RANSAC algorithms at later stages. Using the vertical vector computed in the previous step, we perform an initial classification of the detected planes as horizontal, vertical, or neither. Horizontal planes are those whose normals are within a given threshold angle of the vertical. Vertical planes are those whose normal is approximately perpendicular to the vertical. The remaining planes are classified as neither.

Next, we compute the size of the vertical planes in order to perform further classification. For each plane, axes are defined as the vertical vector and the normalized cross product of the vertical and the plane's normal (the horizontal axis). We project each point in the plane onto the 2D coordinate system defined by these axes and the minimum and maximum values are extracted for each axis. The width and height of each plane is defined as the distance between the minimum and maximum coordinates, where the height is along the vertical and the width is along the horizontal axis. A center point is also defined as the average of the minimum and maximum values on each axis and is projected back into the sensors 3D reference frame. If the width of the plane is greater than the height by a factor greater than 1 (it is wider than it is tall) we classify it as a potential stair riser. After the initial classification process is complete the planes are classified as horizontal, potential riser, vertical that is not a riser, or unknown as shown in Figure 67.

4.8.1.4 Stair Orientation

We estimate the direction of the stairs using the planes that are identified as potential risers. The normals of each of the potential risers is fed into a RANSAC algorithm, which estimates the direction that the stairs are oriented. The vertical component of the resultant

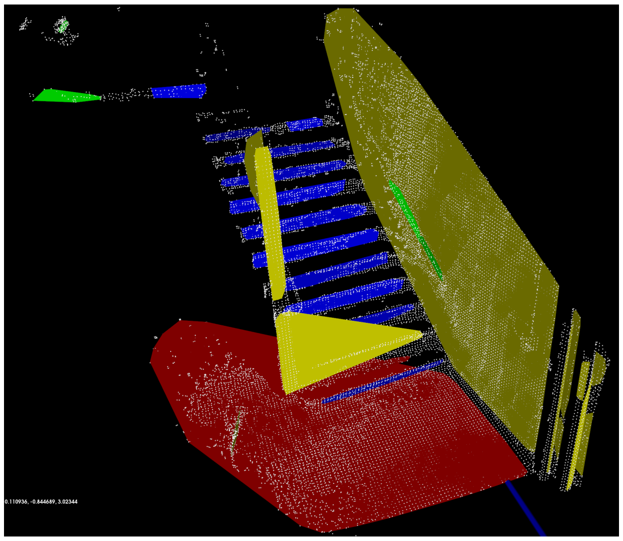


Figure 67: Initial classification of detected planes: riser (blue), tread (red), vertical (yellow), unknown (green)

vector is removed and the result is considered the direction of the stair case. The vertical, direction vector and the cross product of those two (the horizontal), as shown in Figure 66, are used as a coordinate system for the computation of the remaining stair properties. Additionally, any outliers of the RANSAC algorithm are removed as potential risers in order to reduce error in future stages.

4.8.1.5 Compute Stair Properties

Once the direction of the stairs is computed, we estimate the stair parameters; we compute the run of the stairs from the risers. While the direction of the stairs is known, no reference point on the flight of stairs has been determined; an arbitrary reference point is picked because all properties are computed as the difference between points (for simplicity the origin of the sensor's frame is used). We compute the distance from the reference point to the center of each potential riser plane along the stair direction vector. Next, we build a list of the distance (along the direction vector) between each plane, representing the potential distances between risers. Values that are larger or smaller than a minimum and maximum run are ignored to reduce unnecessary computation. Sometimes, not all riser planes are detected or are incorrectly filtered in previous steps, leading to gaps that do not fit in the defined bounds. In order to account for this occurrence, a certain number of potential missing intermediate risers are accounted for. We achieve this by dividing larger distances by each number of potential missing risers plus one and adding the result to the list if it falls withing the required bounds.

We build a histogram from the list of potential stair runs by adding a normal distribution with a mean of each value to each bucket in the histogram. This histogram is processed to find the largest bin. We use the value of that bin as a first estimate of the stair run. We next use a RANSAC algorithm to better refine the estimate and remove outlier planes. A random plane is chosen to be a known riser and the distance of the other planes from the chosen one is computed. The distances are divided by the estimated stair run from the histogram in order to estimate the index of the stair the potential riser may be part of. The model is computed as a linear fit of the estimated stair index and the actual distance offset. We remove the outliers from the RANSAC algorithm from being potential risers. The slope of the resultant linear fit is used as the stair model's run. We group the remaining riser planes by estimating the stair index in order to merge all of the planes that make up a specific stair riser as in Figure 68. For each riser we compute a new width, height and center, similar to the initial plane classification.

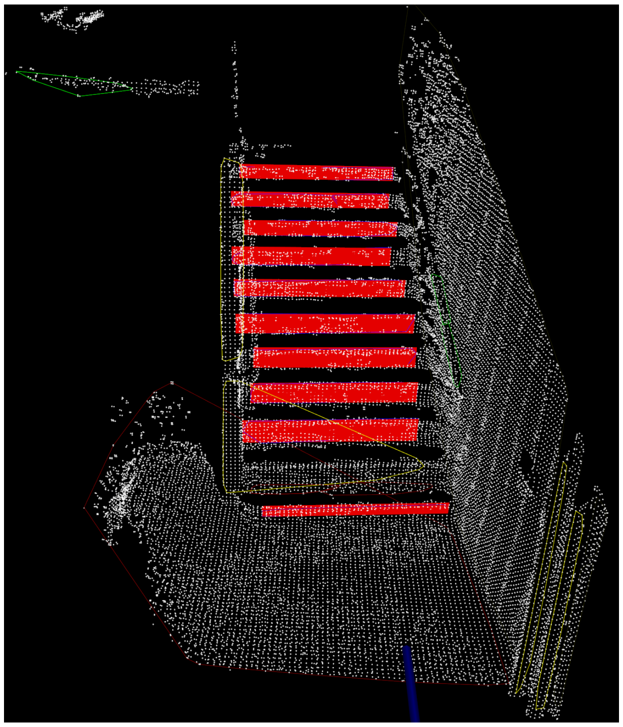


Figure 68: An example of multiple planes merged into risers (red rectangles) with the original planes outlined

Once the planes are grouped into risers, we estimate the number of stairs by iterating through the risers in order, checking for discontinuity. Starting at the first riser, we iterate through the planes until the last plane is reached or the separation between two risers is greater that the maximum skipped planes parameter. The number of stairs detected is the range in indices from the remaining risers.

We use the risers to estimate the rise of each stair; a RANSAC algorithm is performed to fit a linear model to the stair index verse the center position of the riser projected into the vertical vector. The rise of each stair is the slope of this model.

Finally, we estimate the width of the stair using the computed riser bounds. We compute the horizontal minimum and maximum values relative to an arbitrary point (in this case the sensor origin) along the horizontal axis. Because some detected planes do not always extend to the edge of the stair, a RANSAC algorithm is performed on both all of the minimum values and all of the maximum values to compute a linear fit model for each. It is necessary to not only compute the offset, but also the slope because some stair cases do not have consistent widths throughout the staircase. The value of each model at the bottom stair is used as the bounds of either side of the stairs. If the computation of one of the models fails we use only the value from the first stair. The width of the stairs is the distance between the minimum and maximum values. The center of the stairs is the average of these two values.

We use the slope and offsets from the previously computed rise and run models as well as the center of the stairs to estimate the base of the stairs: the center, lower point on the first stair's riser. Using these values the estimate of the stair case is complete, as shown in Figure 69. If at any point during the process, except for calculating stair width, a model fit does not reach a consensus the algorithm will indicate that no stair was detected in the scene.

4.8.2 Stair Climbing

In order to assist the user when climbing stairs the rover will bias the user's control so that the robot's direction is parallel to the direction of the stairs and is centered between the two sides of the stair case. When climbing stairs the robot is most stable when the cleats on the treads are parallel to the edges of the stairs and is least likely to collide with the side of the stairs if it remains in the center. In order to make use of this the operator must first

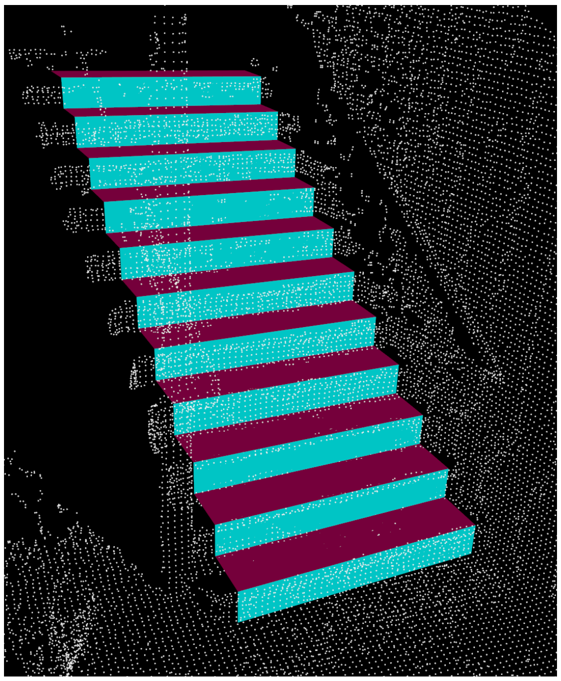


Figure 69: A completed model of the stairs overlaid on the original point cloud

enter a stair climbing mode; the stair detection is computationally expensive so it should only be run when needed. Once in stair climbing mode the operator only has control of the robot's linear speed. The robot's angular velocity is then computed to be proportional to the error in direction of travel as well as the distance from the center of the stair case.

4.9 Payload Interface

This section outlines the payload interface designed into the WALRUS Rover which allows for additional actuators and sensors to be mounted for specific missions. This interface is designed to be more open than current platforms by using standard data ports such as USB 3.0 and power levels such as regulated 5 V and 12 V. This allows anyone to design a payload for the WALRUS Rover to meet their needs. We also discuss our example boom payload which incorporates both the Kinect camera for advanced vision processing features and the environmental sensor board for additional sensing capabilities.

4.9.1 Payload Hardware and Software Interface

Since there is no standard for rover payload interfaces, we elected to design our own. This meant determining the physical mounting as well as the hardware and power ports users would connect their payloads to. Furthermore, this entire interface had to be designed in such a way that kept the rover sealed from water. We decided to use a consistent 2 in square #10 tapped bolt hole array for mounting. Figure 70 shows a top view of the rover with the data and power ports annotated. For power, regulated 5 V, 12 V, and 24 V is available and unregulated 32 V is available. For data, the interface provides USB 3.0, Ethernet, and HDMI ports.

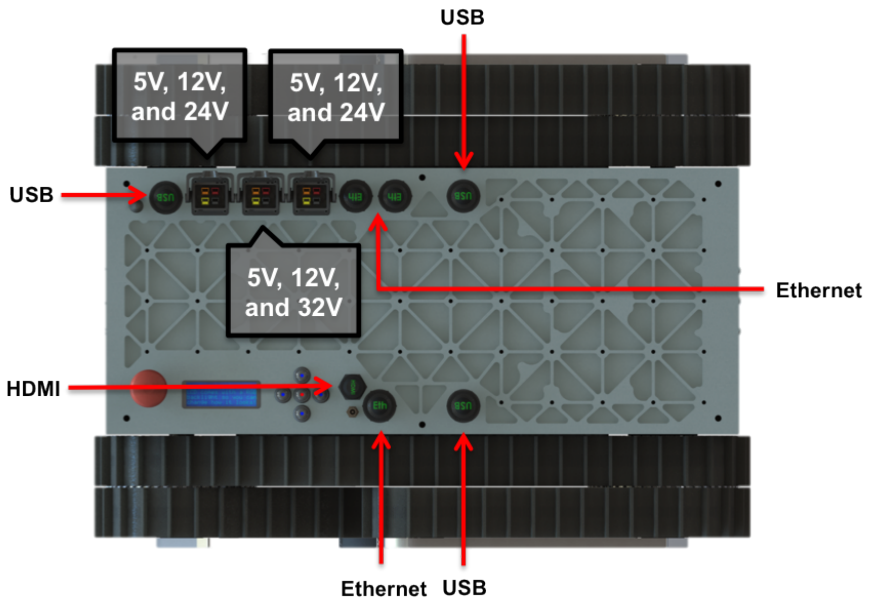


Figure 70: Power and data payload ports.

All of the connectors are IP67 rated, and therefore can be fully submerged in water up to three feet deep without being damaged. Example payloads that can be added to the WALRUS Rover include an arm manipulator and a screen payload for telepresence in medical applications.

In order to show one way the sensing capabilities of the rover can be extended, we built a deployable sensor boom (Figure 71). It offers both environmental sensors, such as gas and temperature, as well as advanced vision capabilities through the Kinect v2. In addition to providing the operator with more information and a better vantage point, the boom allows for the development of more advanced on-board intelligence to help the operator, such as supervised stair climbing.

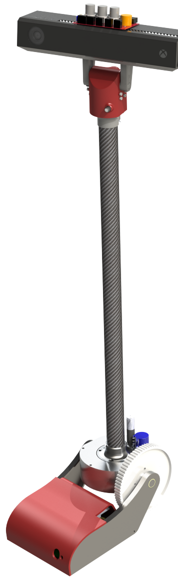
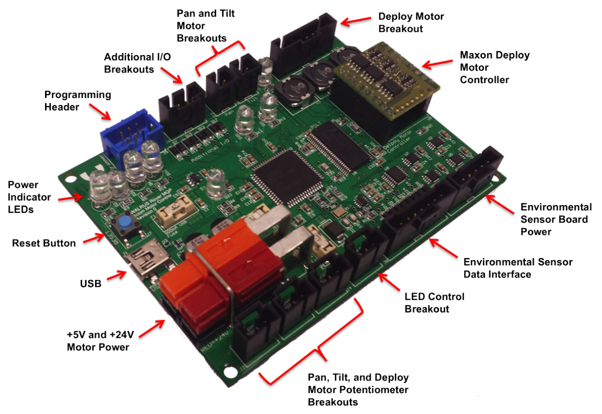


Figure 71: Render of the Sensing Boom


Pan and Tilt Motor Breakouts
Deploy Motor Breakout
Additional I/O Breakouts
Maxon Deploy Motor Controller
Programming Header
Power Indicator LEDs
Environmental Sensor Board Power
Reset Button
Environmental Sensor Data Interface
USB
LED Control Breakout
+5V and +24V Motor Power
Pan, Tilt, and Deploy Motor Potentiometer Breakouts
Figure 72: Boom Board

4.9.1.1 Boom Board

In order to actuate the boom motors, monitor the position of the motors, and interface the environmental sensor board to the main computer, we designed the boom board shown in Figure 72. The boom board is a four layer PCB that can control two brushed DC motors, one brushless DC motor, and interfaces all of the environmental board sensors to the main computer. The complete schematic can be found in Appendix B with Figures 103 and 104. It has an H-bridge IC to control the two 5 V pan and tilt motors, and a Maxon brushless motor controller breakout to operate the deploy motor. The remainder of the board contains breakouts for motor position potentiometers and all of the digital and analog sensor breakouts from the environmental sensor board.

This board's MCU circuit has additional ESD protection on the I^2C and ADC pins as shown in Figure 73. This is because these pins are broken out to the environmental sensor board which will be exposed to unknown and potentially dangerous environments. If that board is ever exposed to an over voltage scenario, the harmful signal can be sent down to the boom board and damage additional hardware.

The other major parts of this board that will be reviewed are the brushless deploy motor driver circuitry and PCB layout, the brushed pan and tilt motor driver circuitry and PCB

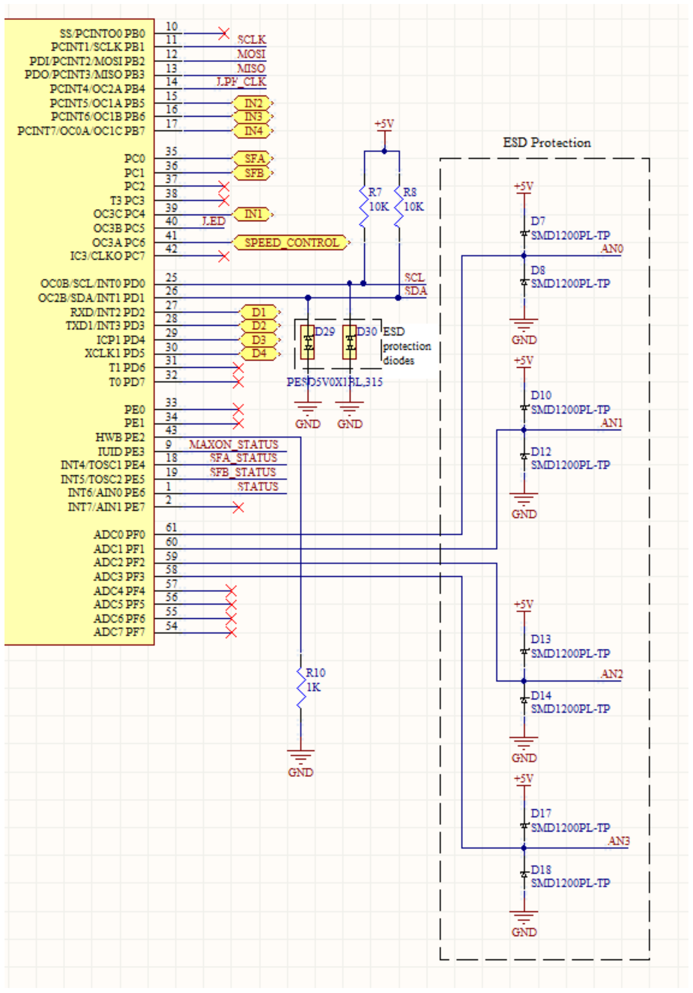


Figure 73: Boom Board MCU ESD Protection.

layout, and the environmental sensor board analog signal conditioning circuitry. Figure 74 shows the motor power schematic used for both the brushless and brushed motors.

The pan and tilt motors use 5 V power, and the deploy motor uses 24 V power which are both received from the Vicor DC-DC converter. Fuses F_2 and F_3 are overcurrent protection fuses which protect the motor drivers, not the motors themselves. C_{26} and C_{27} are large bulk capacitors for the pan and tilt motors. They provide both protection and an immediate source of charge in case of small power supply glitches. To help ensure the capacitors respond quickly and do not heat up due to power dissipation, capacitors with low equivalent series resistance (ESR) were sourced. D_{25} is an ESD protection diode for the brushless motor driver, and C_{28} is another bulk capacitor for the brushless motor.

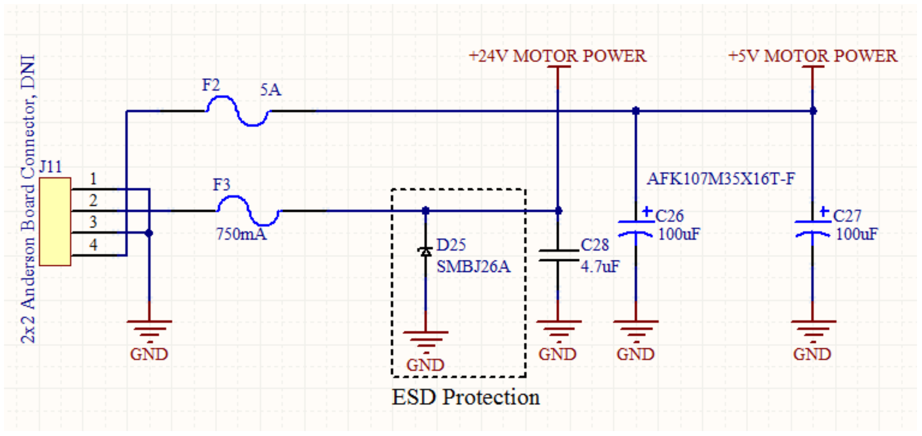


Figure 74: Boom Board Motor Power Schematic.

For reliable performance, additional steps were taken on the PCB layout design of this power circuit. Figure 75 shows the entire boom board layout, and Figure 76 shows just the zoomed in motor power circuit layout.

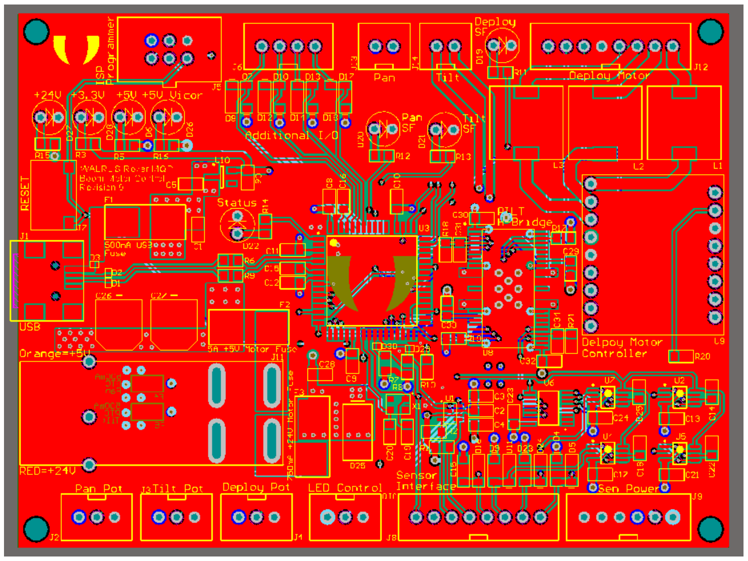


Figure 75: Boom board layout.

This board's layer stack from top to bottom is the ground plane, 24 V plane, motor 5 V plane, and then USB 5 V plane. In addition, signal traces are used on every plane because of how little room between components there is due to the size constraint of the board.

For reduced noise, wide low impedance traces are used for the 5 V and 24 V lines and then via stitched to their respective inner power planes. All fuses, bulk capacitors, and zener

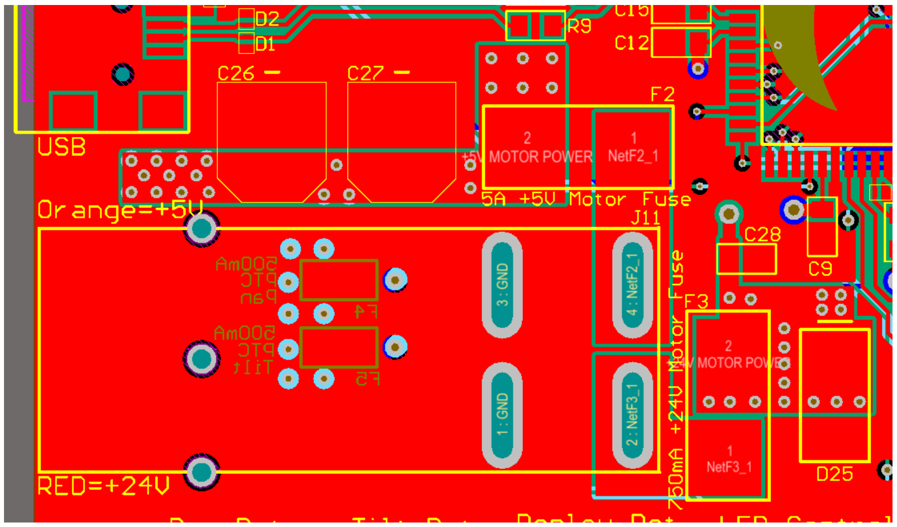


Figure 76: Boom Power Layout.

diodes are placed very close to the power Anderson connectors to ensure any damaging transients become shunted as quickly as possible.

Since the deploy motor is from Maxon, it has a standalone motor driver board that can be used to control it. Rather than making this from scratch, we decided to make a header breakout for the premade driver board. Figure 77 shows the remainder of the deploy motor driver schematic.

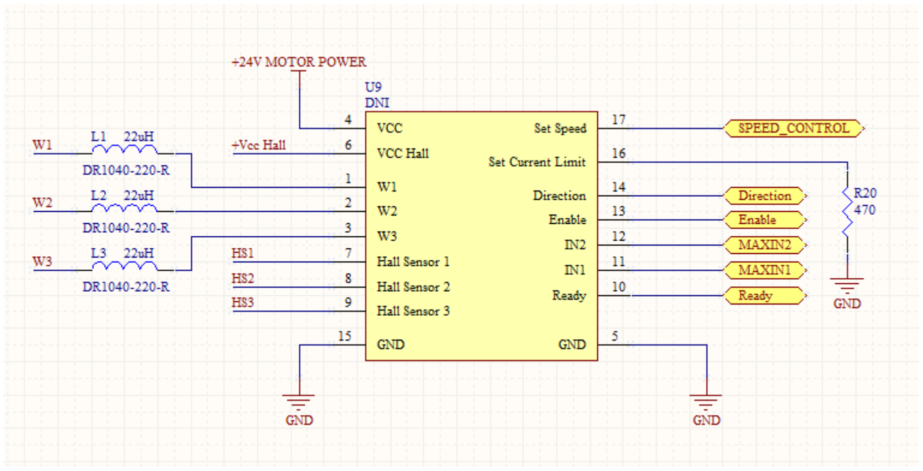


Figure 77: Boom Board Brushless Deploy Motor Schematic.

The driver is the 1-Q-EC Amplifier DEC Module 24/2 and it can handle up to 3 A continuous at 24 V. Digital pins 11-14 are used to control speed mode, enable, and direction, and digital pin 10 is a status flag. A motor current limit to protect the motor is set to 500 mA using R_{20} and the speed of the motor is controlled with a PWM signal to pin 17. The remainder of the pins are used for the three inductor windings control, as well as the position hall effect sensor, and power. To prevent high current ripple in the motor's windings, choke inductors L_1 - L_3 are used. To calculate their minimum inductance required, an equation given on the motor driver data sheet was used as follows. L_{Phase} is the external inductance per phase, V_{CC} is the supply voltage, f_{PWM} is the generated control frequency which is given as 46,800Hz, I_N is the motor's nominal current given on its datasheet, and L_{Motor} is the phase to phase inductance of the motor which is also given on the motor's datasheet.

$$L_{Phase} \geq \left(\frac{V_{CC}}{6f_{PWM}I_N} - 0.3L_{Motor}\right) \tag{11}$$

$$L_{Phase} \geq \left[\frac{24\,\mathrm{V}}{6(46\,800\,\mathrm{Hz})(1.4\,\mathrm{A})} - 0.3(0.182\,\mathrm{mH})\right] \tag{12}$$

$$L_{Phase} \geq 3.2\,\mu\mathrm{H} \tag{13}$$

Deploy Motor Choke Inductor Calculation

Therefore, we chose an electromagnetically shielded 22 μH inductor with high saturation and continuous current, as recommended on the motor driver datasheet. The next major part of this board is the pan and tilt (PILT) motor driver circuitry. An H-bridge is one of the simplest and most common methods of DC motor control. H-bridges are power efficient, easy to control with an MCU, and readily available in many packages and specifications. Rather than design an H-bridge from discrete transistors, it made more sense to spec an H-bridge IC. The H-bridge IC we chose was the MC33932 from Freescale Semiconductor which can handle 5 A continuous at 28 V. The schematic for the MC33932 H-bridge is shown in Figure 78. Aside from meeting our current and voltage specs, the MC33932 also has a variety of additional features making it a great fit for our needs. First, it has overcurrent and thermal shutdown protection. Secondly, it has status flags which can communicate with the MCU to tell whether or not the IC has gone into a shutdown mode. Lastly, it has current feedback for both motors.

The top and bottom halves divided by the dashed line are exactly the same and each control one of the two PILT motors. Pins IN1, IN2, D1, and D2 control the direction, speed, and

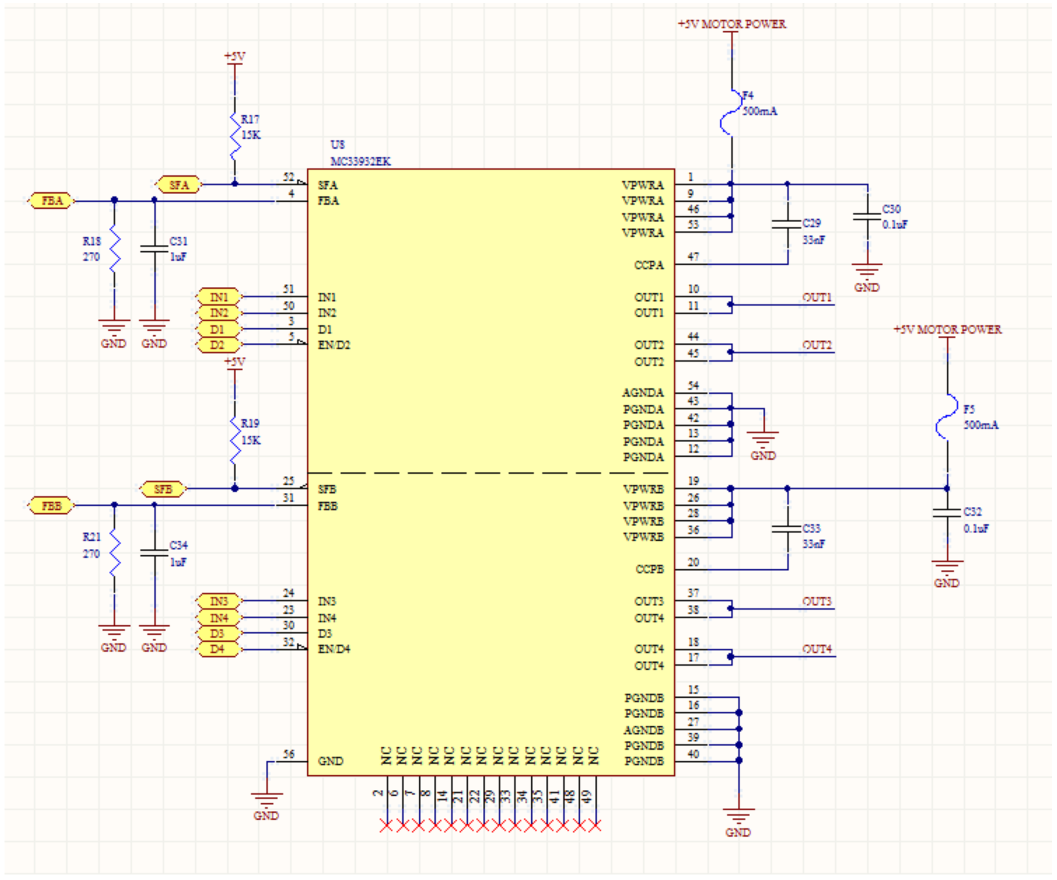


Figure 78: Boom Board Brushed Pan and Tilt Motor Schematic.

mode of the motor. FBA is the current feedback line, and is converted to a voltage through R_{18} before being sent to the MCU's ADC. C_{31} is just a filtering capacitor for this feedback. SFA is the motor status flag which is pulled high with R_{17}. VPWRA is the motor power, which is decoupled, and also protected with a PTC fuse. If the motor ever stalls, the PTC will trip but then reset in a few seconds. This protects the motor without needing to go in and replace a blown fuse. OUT1 and OUT2 are the positive and negative leads of the motor.

The reference documentation for this IC had a number of design considerations for the PCB layout as well which were required for proper performance. The H-bridge layout is shown in Figure 79.

As usual, decoupling capacitors are placed as close as possible to the IC. The more important design considerations were the heatsinking, and the routing of motor power, and

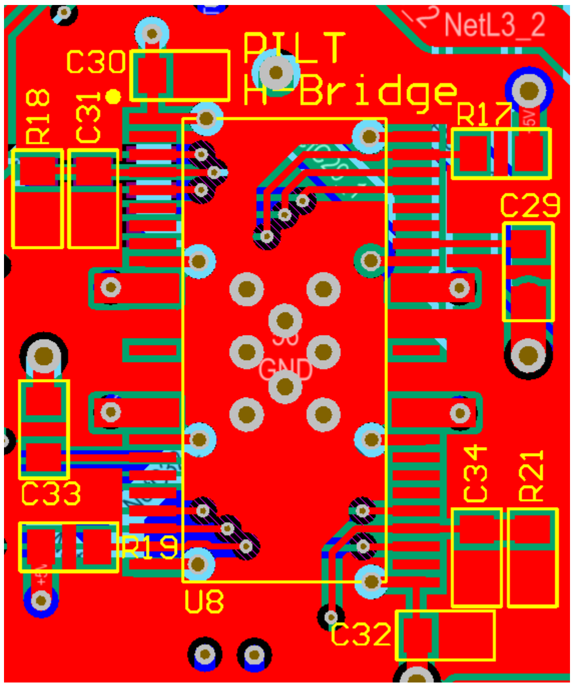


Figure 79: Pan/Tilt H-Bridge Layout.

output motor PWM signals. The MC33932 has a metal plate on the underside of the IC which is used for heatsinking. One of the easier methods to heatsink surface mount components is by attaching them to the ground plane since it is a solid piece of copper. Since the ground plane is the top layer of the board, it was trivial to make a good connection. For increased heat dissipation, we placed vias right beneath the IC's heatsink but were not electrically connected to any other layer. These vias act as "heat pipes" and provide additional convection flow for the heat through the board.

To ensure traces are not damaged from overcurrents and noise does not interfer with the performance of the IC or the motor, wide low impedance traces are used for both the IC's motor power lines and the motor output PWM signals. The PWM signal pins are connected with rectangular pads on either side, and the motor power pins are connected with star routing, as shown in Figure 80. It should be noted that the motor PTC fuses are placed on the left hand underside of the board because of space considerations, and are via stitched to the inner 5 V motor power plane.

The last significant design of this board was with the gas sensors' analog signal conditioning circuitry. The gas sensors produce an analog signal between 0 and 5 V to represent the

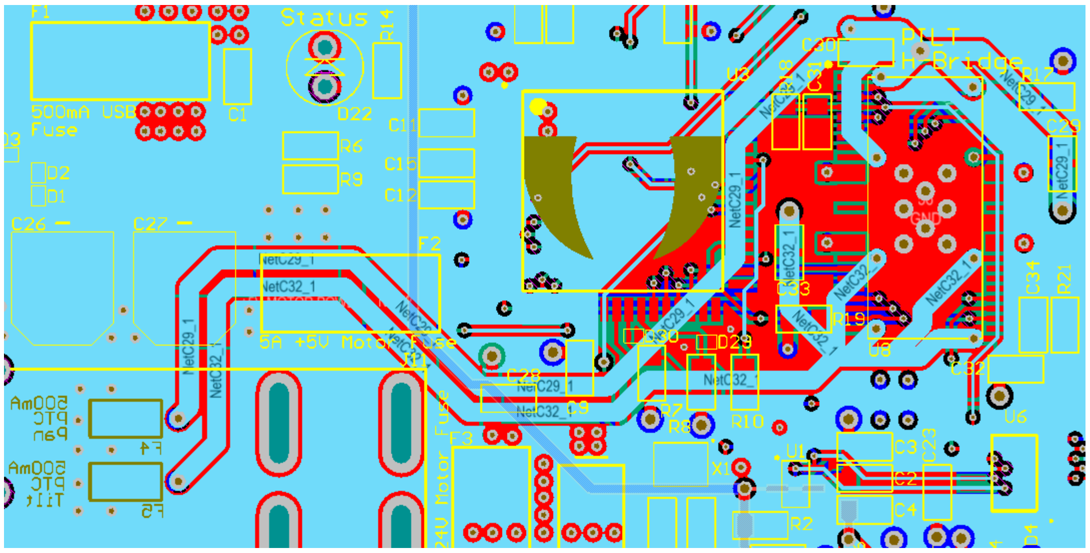


Figure 80: Pan/Tilt H-Bridge Star Routing.

PPM level of a specific gas. Since these sensors will be placed in unknown conditions, and are sensitive to heat, they will likely be subjected to high frequency noise. Therefore, low pass filters for each gas sensor signal should be used in order to clean up with signal. The challenge is that since these environments are unknown, there is no good way to predict what type of filter and how much filtering is required. Therefore, rather than making simple first or second order filters with a fixed cutoff frequency, dynamic 5^{th} order low pass filter ICs from Maxim were speced. The schematic for one of the four gas sensor filters is shown in Figure 81. The exact same circuit for the other three is repeated.

The LPF IC selected is the MAX7408CUA+. It is very simple to use with only an input analog signal, the filtered output, external decoupling pins, and the cutoff frequency controlling input. The filter's cutoff frequency is adjusted by changing the frequency of LPF_CLK. The equation given on the datasheet is $f_c = \frac{f_{clk}}{100}$. D_9 and D_{11} are ESD protection diodes for the IC, since the signals come from the environmental sensor board. This IC makes it easy to determine a suitable cutoff frequency, without having to also worry about overall response since it's a high performance 5^{th} order filter. Chances are that the cutoff frequency would not need to be varied based on the environment, but this IC does give that flexibility if desired as well.

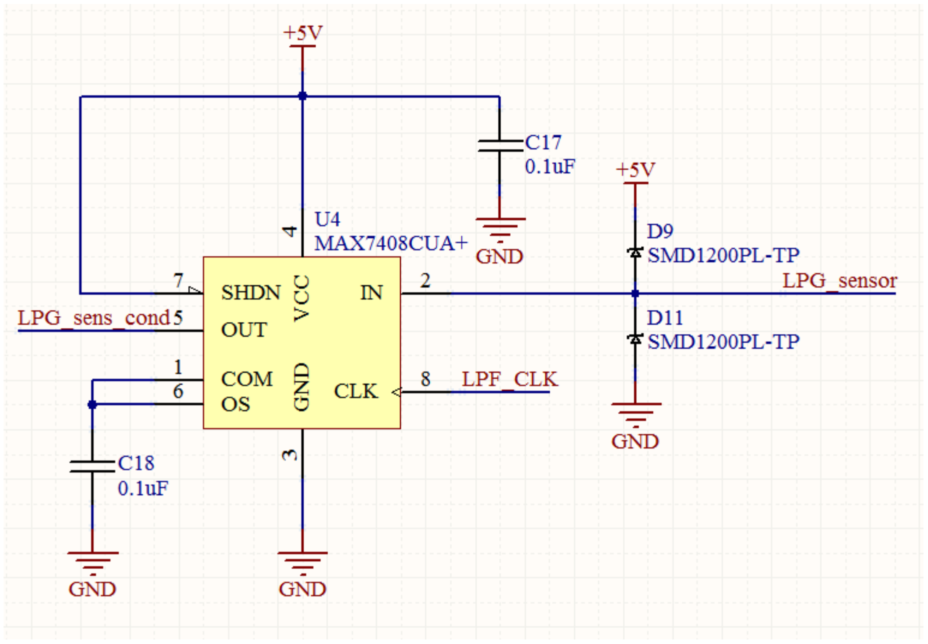


Figure 81: Gas Sensor Low Pass Filter.

5 Performance Evaluation

In order to evaluate the performance of the system we conducted the testing outlined in our requirements chapter (Chapter 3). Many of these tests were done with the intention of better understanding the limits of our platform. To test mobility, we used the facilities at the NERVE Center at UMASS Lowell [46]. During the NERVE Center testing, all of the diagnostics data was logged so that current draw, battery charge level, and temperatures could be reviewed later.

5.1 Physical Properties

When the pods are stowed or upright, the WALRUS Rover fits (Figure 83) within a 32 in vertically projected cylinder, and is only 19.5 in wide. These dimensions ensure easy navigation in tight spaces. The robot is very stable with a pitch stability of 80° (Figure 84), and roll stability of 60° (Figure 82). The robot currently weighs 82 lbs including two batteries.

5.2 Land Mobility Testing

We tested mobility by driving the robot on various terrains and observing how the robot performed including speed of travel and ease of control. Tested terrains included features

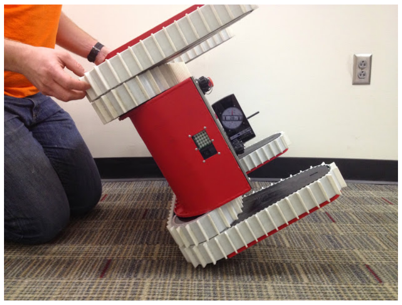


Figure 82: 60° Roll stability

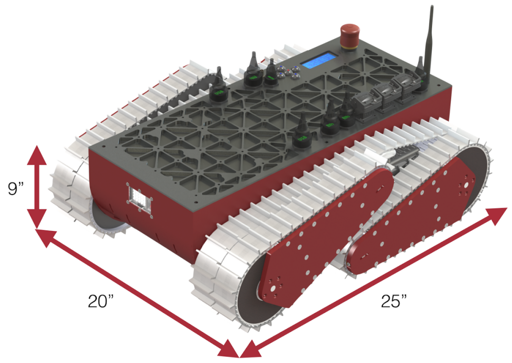


Figure 83: Robot dimensions

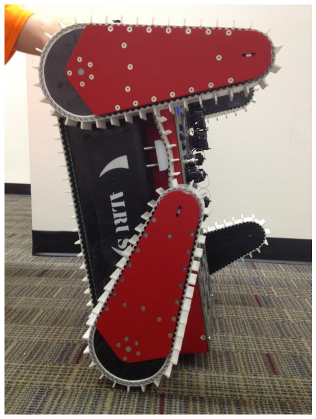


Figure 84: 80° Pitch stability

that would be found in an urban environment as well as those that would have resulted from destruction of the environment.

5.2.1 Flat Land Speed

In order to measure the land speed we commanded the rover straight down a flat hallway for a distance of 32 ft at the maximum safe velocity of the motors, 7 ft/s. Once accelerated the robot took approximately 4.6 s to travel the 32 ft distance. This exceeds the project requirement of 6.5 ft/s. Similar results were computed from the drive wheel encoders, which measured the commanded speed of 7 ft/s. The wheel encoders were earlier checked for accuracy using a tachometer.

5.2.2 Stairs

To test stair traversal we used a number of different staircases of varying slope and material. The robot can successfully ascend staircases with slopes up to 35° with no assistance. We tested this by starting the robot on the floor in front of a staircase and commanding the robot forward at 1.6 ft/s straight up the staircase. The front pods were positioned at approximately a 45° angle from the ground in order to allow for an easy transition from the floor to the staircase. Once on the stair case the robot will ascend normally without needing to stop. The most important aspects of climbing stairs are staying centered on the staircase and avoiding turning significantly. Figures 85 and 86 show the WALRUS Rover climbing different staircases.

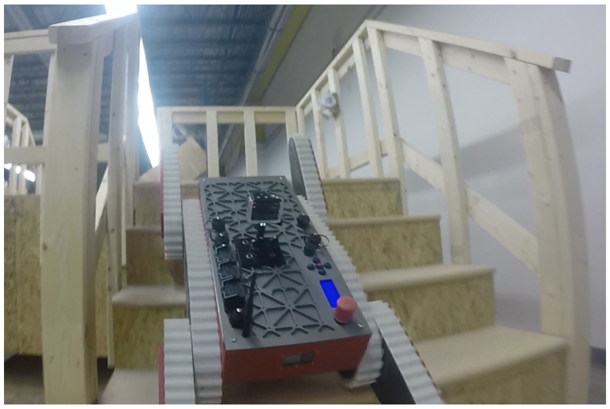


Figure 85: WALRUS Rover Climbing 35° NIST Stairs

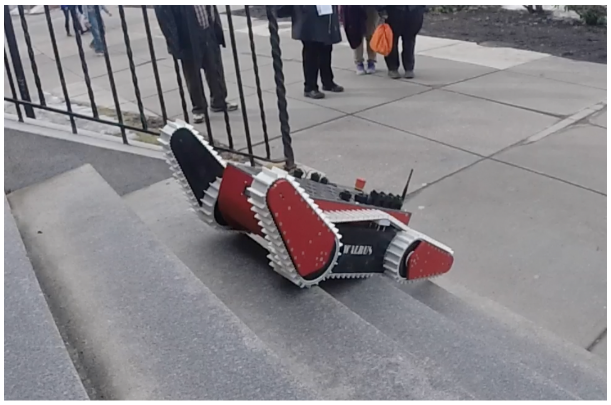


Figure 86: WALRUS Climbing Outdoor 30° Concrete Stairs

The material of the tread of a staircase can have an affect on the ease that the robot can ascend it. Staircases with more slippery materials, such as smooth stone, and steep inclines can lead to increased slipping of the drive tread cleats on the lip of the stairs. Increased slipping of the treads can decrease the stability of the robot as it can cause sudden turns, which can be hard to compensate for when the robot is driving quickly. By driving slower the robot is still able to ascend these kinds of stairs and remain stable. Stairs made of a less slippery material, such as rubber, pose no issue and can be ascended quickly.

The largest limitation of climbing steep stairs is the available current. With two batteries installed the robot is able to successfully climb a 34° stair case while the computer is at a medium load; however, if while climbing the pods are actuated, the current spike will exceed the batteries' limits and they will enter a protective shutdown. This may be solved through the use of four batteries instead of two.

5.2.3 Rough Terrain

The WALRUS Rover excels at traversing rough landscapes. Using its four pods the rover is able to easily traverse many larger obstacles. Two examples of difficult terrain are the NIST Crossing Ramps and Symmetric Stepfield scenarios.

The NIST Crossing Ramps scenario involves traversing a series of alternating 15° ramps perpendicular to the direction of travel, as shown in Figure 87. In this scenario the robot is able to traverse the terrain in most places with the pods fully extended. When driving over the high point where two ramps meet it will sometimes bottom out; however, this is easily remedied by actuating the pods to lift the robot body above the ramps. The robot is then able to easily proceed. As this course is relatively consistent and the robot does not pitch up and down significantly while traversing it, it is possible to navigate this course using the chassis cameras, although the use of the boom camera would provide a much better view of the surrounding area.

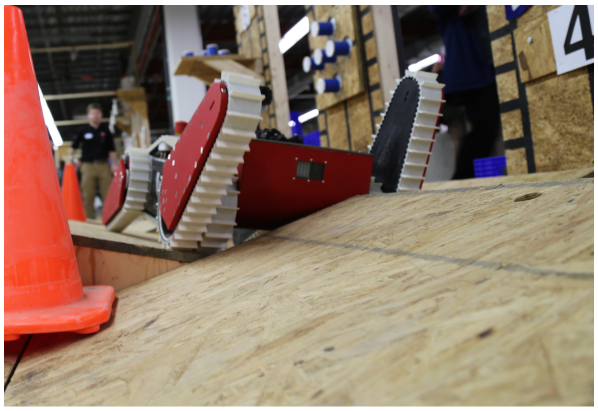


Figure 87: The WALRUS Rover traversing a NIST Crossing Ramps course

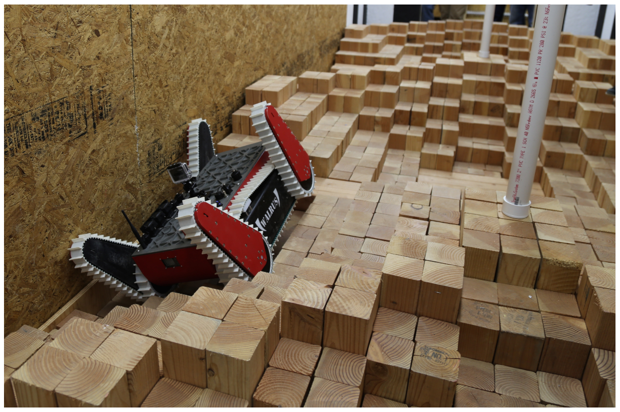


Figure 88: The WALRUS Rover traversing a NIST Symmetric Stepfield

The NIST Symmetric Stepfield, shown in Figure 88, is a much more complex terrain that requires more attention when traversing. It consists of a number of 4x4 wooden posts of varying heights. This course proves a significant challenge for many robots and was performed while watching the robot. The course emphasised the usefulness of having four pods as opposed to two when traversing larger terrains as they make it much easier to get over higher points where the robot would normally bottom out. While the robot pitched significantly more in these tests, rendering the chassis cameras mostly useless, the use of a boom camera would provide a good view of the area to allow the operator to traverse it remotely.

In addition to being able to traverse this terrain, it is important that the WALRUS Rover can pass the terrain without quickly reaching the limits of the system. When traversing rough terrain the motor temperatures increase as the motors can draw up to 25 A continuously. However, even in the tough terrain presented at the NERVE Center, the exterior motor temperatures never exceeded 60 °C as seen in Figure 89. Although this is the exterior motor temperature, it still indicates that the winding temperature should be less than the max winding temperature of 125 °C.

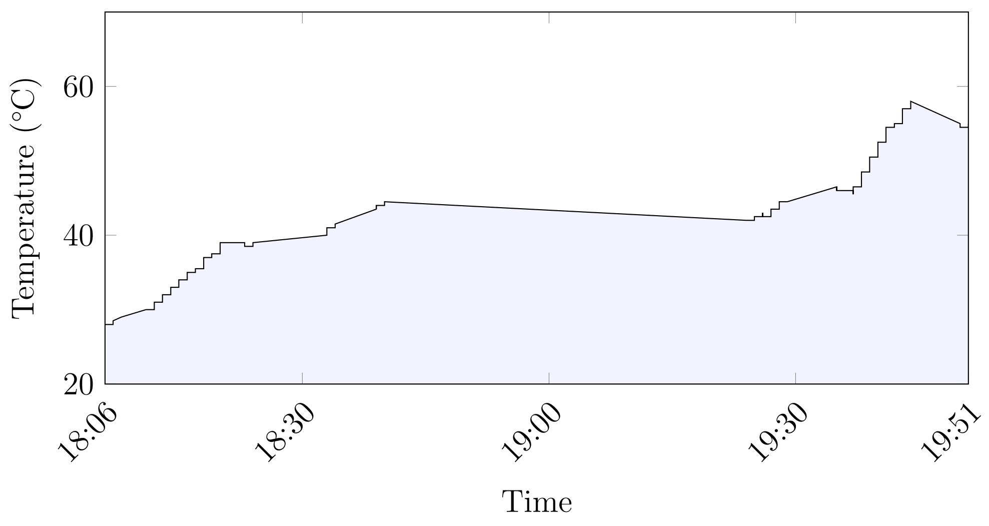


Figure 89: Motor temperature during testing at NERVE.

5.3 Water Mobility

We performed a test of the robot's ability to traverse water by placing the robot in a body of water and attempting to propel it forward using the treads (Figure 90). This proved more difficult than originally expected as the robot exceeded the weight limit that was deemed ideal causing it to sit lower in the water. This led to reverse thrust being generated by water being lifted over the top of the treads and thrown forward (Figure 91). Although this is detrimental to the maximum achievable speed in water, a speed of 1 ft/s was still achievable. Figures 90 and 91 show walrus traveling through water at the NERVE Center.

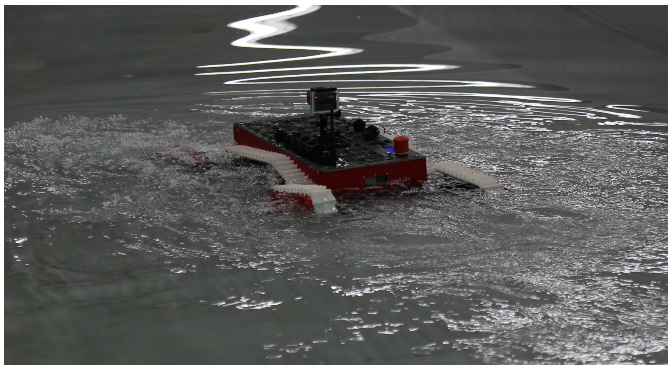


Figure 90: WALRUS Rover in water

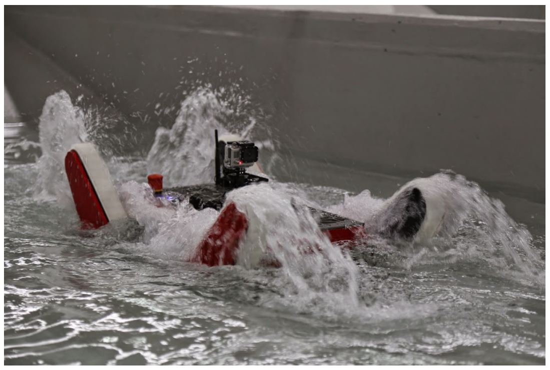


Figure 91: Rover fording

5.3.1 Sealing

Although the robot is able to traverse water for periods of 20-30 minutes without encountering any leaks, water will eventually leak through the drive shaft seals. This is more than long enough for traversing short bodies of water and may be fixed through a number of improvements outlined in Chapter 7, Future Work. In the event of a leak the operator is provided with a notification that a leak has occurred. In all cases tested thus far, the max total volume of water accrued in the chassis formed a bubble the size of a dime.

5.4 Communications

In order to test the communication capabilities, we operated the rover outside from a distance through an area sparsely filled with trees as shown in Figure 92. The robot was still easily controllable at a distance over 1000 ft line of sight with a single camera feed as operator feedback, satisfying the project requirement of 975 ft. In order to further explore the capabilities, the robot was driven further beyond a small one story building and continued to be controllable up to a distance of approximately 1200 ft.

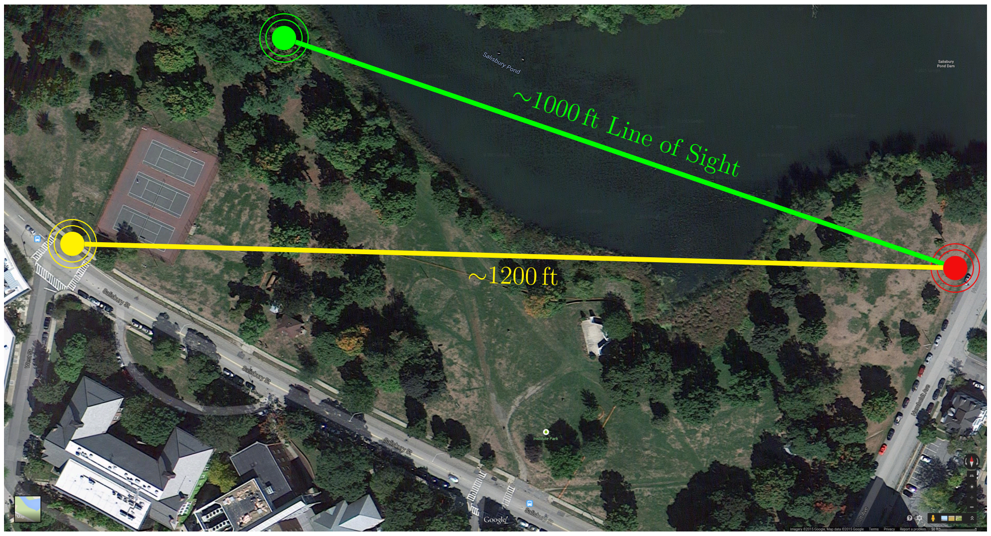


Figure 92: Communications distance testing map.

In addition to full range outdoor testing, we drove the robot around in a multi-story building to test robustness against signal interference and attenuation. We performed the testing in an area that contained a number of other strong WiFi access points, which could interfere with communications. These devices appeared to cause no issues as the robot was still easily controllable over one hundred feet away when traversing a different floor than the operator was on.

5.5 Power Consumption

We monitored the power consumption throughout testing using the SMBus and galvanic isolation board in order to evaluate the expected battery life. During the testing performed at the NERVE Center the remaining battery was monitored in order to determine how traversing the demanding terrain impacted battery life. Figure 93 shows the battery level throughout much of the testing that occurred. During the one and a half hour testing window, the battery level of the two installed batteries decreased from 90% to 49% as reported by the battery management system. The testing at the NERVE Center represented a continuous mix of demanding and light driving. Ultimately, the max battery life at an average battery consumption well exceeds the requirement of one and a half hours.

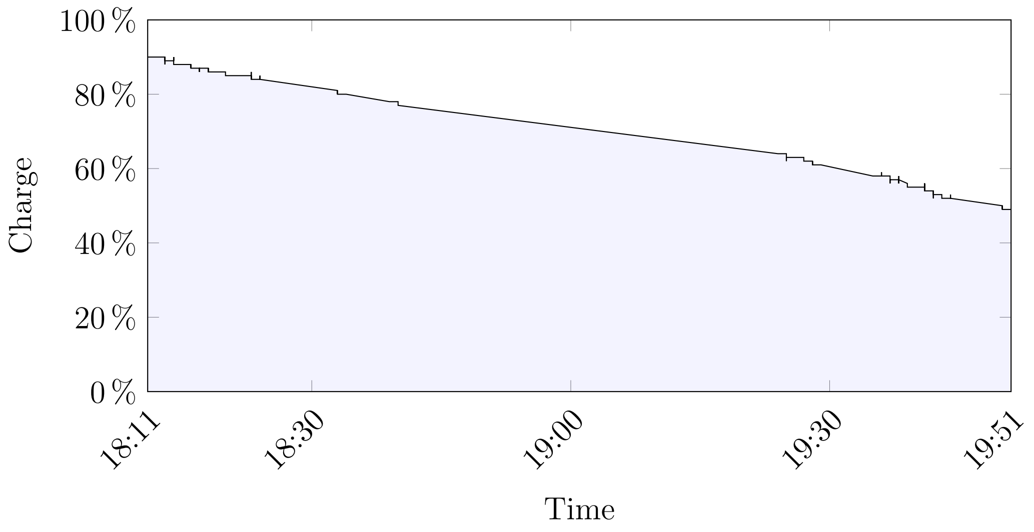


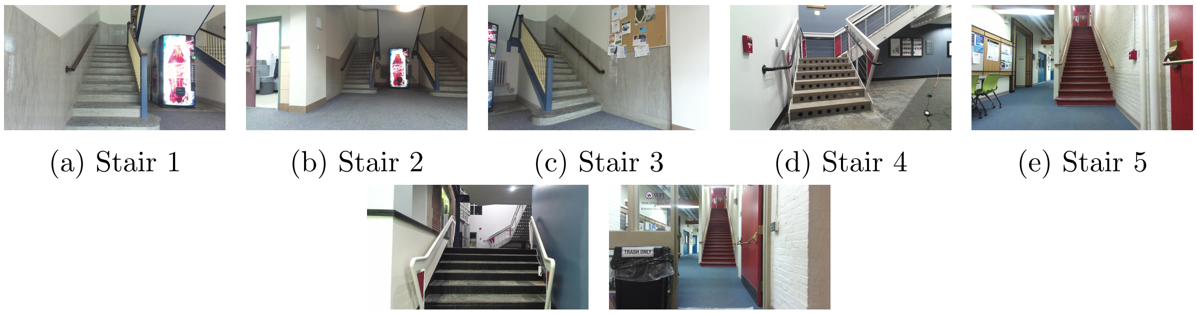
Figure 93: Power consumption history while testing at NERVE.

5.6 Operation

Operation of the robot is made easy through a web interface that is available on an machine using a modern web browser that supports WebRTC, Websockets, Flexbox and the Gamepad API (Chrome and Firefox) without the requirement of installing or configuring any additional software.

5.6.1 Situational Awareness

The operator has the ability to view up to two camera views at any given time so that they are able to better understand the environment around then. In order to visualize the robot state, a full 3D model of the robot is displayed so that the operator understands how


(a) Stair 1 (b) Stair 2 (c) Stair 3 (d) Stair 4 (e) Stair 5

(f) Stair 6 (g) Stair 7

Figure 94: The stair case perspectives used to test the algorithm

the robot is configured. The robot's orientation relative to gravity is also displayed so that the operator is able to understand the surface that the robot is resting on. The operator interface, shown in Figure 62, was used during the mobility tests and allowed the operator to maneuver easily in many different environments.

5.6.2 Supervised Autonomy

The implemented stair detection algorithm was evaluated on 7 different datasets with varying perspectives and stair measurements. The tested stair cases are shown in Figure 94 and the computed results are shown in Table 6 in Appendix C. In two of these datasets the stairs were not detected because the stair planes were too small to be correctly detected; however, in the other sets the stairs were correctly detected in 49 out of 50 frames. The computed rise and run were always within 2 cm of the ground truth, but within 1 cm in the majority of cases. While the computed width normally varied significantly more (up to 38 cm in a few cases), it was within 5 cm in most cases; despite the variance in width, the center of the stair case remained well within the sides, which allowed it to still be used as a target position when climbing stairs. More details on the results can be found in [41].

Using the detected stairs the operator can then instruct the robot to climb up the stairs while attempting to stay centered. In order to stay centered the robot's turning is biased towards the center of the stairs. Although this drives the robot towards the center it results in oscillation around the center. In order to avoid this the control output of the robot's turning velocity is also biased to drive the robots orientation to be the same as the direction of the stairs. This both dampens the oscillations and results in the robot traveling straight up the stairs. While this works well, the stair detection is computationally expensive so it must be activated by the operator.

6 Project Logistics

This chapter briefly outlines the important logistical aspects of the project. It begins with a timeline used to keep the team on pace throughout the academic year and is followed by a budget analysis.

6.1 Timeline

This project required a full academic year of work to complete. It began with research, then proceeded to design and prototyping, assembly, and finally testing and validation. With such an ambitious project, a strict timeline was developed in order to keep the team on track. This timeline is shown in Appendix D.

6.2 Budget

In an effort to project the cost of development for the WALRUS Rover, we created our expected budget at the beginning of the project. Given that a rover such as this can be created many different ways, we created two separate budgets, a minimal cost to produce the rover at minimal requirements and an optimal cost budget that was a best-case-scenario budget. This budget allowed us to purchase high-end computing, actuation and vision systems. The initial budgets are shown in Appendix E. In the end, we gained more sponsorship and support than anticipated. This extra support allowed us to make the robot smarter and more powerful, with high end components. Our face costs are shown in Figure 5. The rover is still very cost effective, as other robots on the market such as Packbot and TALON can cost over $150,000. Even with the increase in cost of the rover, the project has resulted in no cost to the students.

Chassis & Actuation	**Cost**	**Electronics**	**Cost**
Motors	$2,750	Main Computer	$1,000
Motor Controllers	$2,000	Custom PCBs	$2,000
Power Transmission	$3,430	Wiring & Connectors	$1,500
Treads	$2,800	Sensors	$1,500
Raw Materials	$1,500	Batteries & Charger	$4,400
Hardware	$1,000	Misc Electronics	$1,600
Total	**$13,480**	**Total**	**$13,000**
Grand Total	$26,480		

Table 5: Robot Costs

7 Future Work and Recommendations

Although the WALRUS Rover achieved many of its goals and is overall successful, as with any project there are improvements to be made. In this section we will cover some of the more pressing issues with the rover and how they could be improved in the future.

Robot Weight Weighing in at about 80 lbs, the WALRUS Rover is almost 20 lbs over its intended weight. This causes issues with motor current draw as well as with the robot's buoyancy in water. One way to alleviate these problems would be to consider using different aluminium alloys with better strength/weight ratios and removing material from inside the robot. Another would be to do more investigation into the internal ribbing structure and find ways to remove parts of them. Many of the electronics are very heavy. Use of a custom mount for the Vicor DC-DC converter could save weight. Lastly some of the stainless steel shafting may be replaced with anodized aluminium alloy without removing strength, but losing weight.

Chassis Welding One of the largest issues with the WALRUS Rover's body is caused by significant shifting during welding. Many of these shifts were caused by improperly designed fixtures and a lack of support from bowing. Use of weldment slots or deeper grooves would have allowed the welds to better hold the chassis and prevent several spots where weldments were milled off during finishing operations.

Chassis Sealing During extended use in water some of the shaft seals inside the robot allow for micro leaking to occur. In these leaks, less than 0.1 cubic inches will build up over a duration of about 30 minutes. To prevent these leaks, a re-design of the shafting inside the robot should include double sealed shafts, as well as bi-directional seals. This would not only allow the internals of the robot to be pressurized to help detect leaks, but would prevent all seals from failing due to alignment or wear. Additionally, a re-design of the robot's top plate bore seal should no longer have the seal route around the center bolts. This would allow for a much better seal as well as greatly improve the installation process.

Motor Selection Having chose to use Maxon motors and motor controllers as our primary drive motors left a significant lack in feature set in the use of our pod motors. As future development, either the pod motor and motor controller should be replaced, or a more intelligent and safer motor controller should be selected. An easy fix would be to replace the RoboClaw motor controllers with RoboteQ motor controllers and control them over a serial interface. This would prevent issues with the current motor controllers which do not automatically stop once a signal is invalid or the controller lose connection.

SMBus Isolation Currently the design of the SMBus isolation board routes all of the signals through one board to isolate the voltages, then relays the signals to the main board for processing. An improved version of this would have a single board per battery that is stored inside of the battery connector. The signal would then be processed on an internal board an sent back to a main controller for transmission to the PC. This would allow for much better wire routing. Another improvement to the SMBus would be to isolate out improper signals. Although not fully diagnosed, the current system continually reads bad values when the robot's pods are moving at all.

Electronics Layout Overall a re-routing of all of the robots electrical components should be thought through. Currently some items are located far away from their mating components and unnecessary or complicated wire routing is necessary. Several components that are located on the mid-electronics board should be moved to the top plate to remove extra routing.

Energy Dissipation One of the largest problems with the WALRUS Rover is the excessive energy dissipation due to back EMF when the robot slows down. Currently the robot has a low deceleration factor to prevent the robot from stopping too soon, but if this number is increased, there will be a large voltage spike on the main power line which may cause issues with some electronics. To fix this, there should be ample capacitor banking to absorb and slowly dissipate the extra energy.

Additional Autonomy Features To give the operator better control, several autonomous features should be developed. Automated lane centering would center the robot between the two primary obstacles on either side of it to prevent accidental crashing. Auto-pod positioning would straighten out the pods at high speeds to prevent the robot from turning too easily and damaging itself. Significant testing would need to be completed to ensure that the robot is reliable enough to avoid unpredictable behavior.

Additional Payloads/Sensing A waterproof vision and sensing payload should be developed to replace the non-waterproof one which exists now. Another useful payload would be a waterproof 2 degree-of-freedom arm. This would allow the robot to deliver tools and supplies to patients or people in need. Lastly, another system would be a waterproof cellular 4G adapter for using the robot on long-range missions.

Water Speed With the robot not reaching its target speed, one possible improvement would be to add cowlings over the tread to prevent excessive water movement in the opposing direction. Additionally, removing weight or adding buoyancy would also help increase water speed.

Treads To improve the tread drive on the WALRUS Rover, a dynamic tensioner should be added. This will allow the tread to ingest small debris without causing jams to occur. Also, the ends of each cleat should be rounded to reduce resistance when turning.

Thermal Dissipation While the robot is running difficult computation and aggressive driving, it is possible for the computer to reach unsafe use temperatures. A future improvement would be to add additional cooling to the top plate in the form active fans. This would allow for forced convection to occur on the top plate to lower the operating temperature.

Software Improvements Some work should be done on the WALRUS Rover's main software. These changes include: adding more features to the on-board diagnostics display, adding better and more configurable browser support, and adding an RTC channel for commands.

8 Conclusion

The goal of this project was to create an expandable amphibious rover to aid in the search and discovery of persons in distress. To do this, we researched the capabilities of rovers on the market today and created a list of requirements for the WALRUS Rover that would make it competitive with current technology. Using these requirements, we designed a system that would not only challenge us academically but also provide a new solution to the problem at hand. Using resources available to us on WPI's campus, we were able to manufacture all parts of the rover ourselves. Through collaboration with external sponsors we designed and had custom PCBs manufactured. Additionally, we not only developed our own software for both embedded and web-based systems, but also released tools and software packages into the community for use by others.

This project was successful in reaching nearly all of our design requirements and in many cases exceeding our design specifications. In one academic year this team designed, built, and thoroughly field tested a rover prototype. The end result is a fully functional rover of professional quality and standards. The WALRUS Rover is capable of traversing water, climbing stairs, and driving over large obstacles all from a distance with help from its sophisticated situational awareness and long range communications system. Overall the team is proud to have created a rover of product level quality. We can compare our rover to existing platforms such as the iRobot PackBot (Figure 95) and QinetiQ TALON, and in some cases feel we have advanced specific features of these rovers. With an open source expandable payload interface, the WALRUS Rover can be converted into an all purpose rover for a broad set of mission scenarios. We believe that this type of system will allow future students, developers and companies to build upon our work and improve the field of search and discovery robotics.

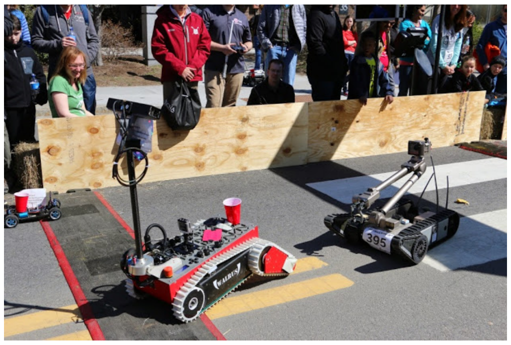


Figure 95: WALRUS Rover and Packbot working together.

References

[1] D. Coleman, “Ros control overview,” 2013.

[2] CRED, “EM-DAT: The OFDA/CRED International Disaster Database.,” online, CRED, 2015. http://www.emdat.be/.

[3] iRobot, “110 FirstLook Specifications,” online, iRobot, 2014. http://www.irobot.com/~/media/Files/Robots/Defense/FirstLook/iRobot-110-FirstLook-Specs.pdf.

[4] Vecna, “Battlefield Extraction-Assist Robot,” online, Vecna, 2009. http://www.theinvestigativefund.org/files/managed/BEAR.pdf.

[5] iRobot, “510 PackBot Specifications,” online, iRobot, 2014. http://www.irobot.com/~/media/Files/Robots/Defense/PackBot/iRobot-510-PackBot-Specs.pdf.

[6] B. Dynamics, “Atlas - The Agile Anthropomorphic Robot,” online, Boston Dynamics, 2013. http://www.bostondynamics.com/robot_Atlas.html.

[7] M. S. M. C. Olsen, “121030-f-al508-081c aerial views during an army search and rescue mission show damage from hurricane sandy to the new jersey coast, oct. 30, 2012,” 2012.

[8] P. O’Connor, “Sandy damage,” 2012.

[9] R. R. Murphy, “Trial by Fire: Activities of the Rescue Robots at the World Trade Center from 1121 September 2001,” article, 2004. https://www.student.cs.uwaterloo.ca/~cs492/papers/trial.pdf.

[10] M. MacRae, “Rescue Robots Aid Japanese Recovery,” online, American Society of Mechanical Engineers, 2011.

[11] E. Guizzo, “Japan Earthquake: Robots Help Search For Survivors,” online, IEEE, 2011. http://spectrum.ieee.org/automaton/robotics/industrial-robots/japan-earthquake-robots-help-search-for-survivors.

[12] E. Ackerman, “Real Robots to Help Fight Ebola,” online, IEEE, 2014. http://spectrum.ieee.org/automaton/robotics/medical-robots/real-robots-to-help-fight-ebola.

[13] S. Gaudin, “First robot, networked tablets head to West Africa to fight Ebola,” online, Computer World, 2014. http://www.computerworld.com/article/2852680/robot-networked-tablets-head-to-west-africa-to-fight-ebola.html.

[14] E. Guizzo, “Robots Enter Fukushima Reactors, Detect High Radiation,” online, IEEE, 2011. http://spectrum.ieee.org/automaton/robotics/industrial-robots/robots-enter-fukushima-reactors-detect-high-radiation.

[15] CNET, "From living rooms to battlefields at irobot," 2010.

[16] iRobot, "510 PackBot CBRNe," online, iRobot, 2013. http://www.irobot.com/~/media/Files/Robots/Defense/PackBot/iRobot_510_PackBot_HazMat_CBRNe.pdf.

[17] E. Grabianowski, "How Military Robots Work," online, 2010. http://science.howstuffworks.com/military-robot3.htm.

[18] iRobot, "iRobot 710 Kobra," online, iRobot, 2015. http://www.irobot.com/For-Defense-and-Security/Robots/710-Kobra.aspx#PublicSafety.

[19] iRobot, "710 Kobra Specifications," online, 2014. http://www.irobot.com/~/media/Files/Robots/Defense/710/710_Specs.pdf.

[20] C. Authors, "C-Talon Unmanned Robot," online, 2014. https://www.qinetiq-na.com/products/unmanned-systems/c-talon/.

[21] QinetiQ, "TALON Specifications," online, 2014. https://www.qinetiq-na.com/wp-content/uploads/brochure_talon.pdf.

[22] S. of Massachusetts, "Building Planning Foe Signal and Two-Family Dwellings," online, State of Massachusetts, 2008. http://www.mass.gov/eopss/docs/dps/780-cmr/780053c.pdf.

[23] B. Bradley, "Stair Design From The Ground Up," online, Builder Bill, 2012. http://www.builderbill-diy-help.com/stair-design.html.

[24] A. M. . T. P. R. J. Vivek Karaveer, "Modeling and Finite Element Analysis of Spur Gear," journal, VIT University, Chennai Tamilnadu India, 2013.

[25] Electromate, "Maxon Motor AG," online, 2015. http://www.electromate.com/products/series.php?&series_id=105269.

[26] Pololu, "RoboClaw 2x15A Motor Controller with USB (V4)," online, 2015. https://www.pololu.com/product/2392.

[27] Ubiquiti, "Bullet M," online, Ubiquiti, 2015.

[28] Ubiquiti, "BulletM Titanium Datasheet," tech. rep., 2015. http://dl.ubnt.com/datasheets/bulletm/BulletM_Ti_DS.pdf.

[29] IPC, "Design Guide for High-Speed Controlled Impedance Circuit Boards," online, IPC, 2004. http://www.jrdpcb.com/userfiles/files/IPC-2141A-2004_%E9%98%BB%E6%8A%97%E5%8F%97%E6%8E%A7%E9%AB%98%E9%80%9F%E7%94%B5%E8%B7%AF%E6%9D%BF%E8%AE%BE%E8%AE%A1%E6%8C%87%E5%8D%97.pdf.

[30] E. Web, "Edge Coupled Microstrip Impedance," online, EE Web, 2015. http://www.eeweb.com/toolbox/edge-coupled-microstrip-impedance.

[31] NXP, "Opto-electrical isolation of the I^{2}C-bus," tech. rep., 2015. `http://www.nxp.com/documents/application_note/AN10364.pdf`.

[32] "ROS.org — Powering the world's robots." `http://www.ros.org/`.

[33] "RIVeR-Lab/walrus." `https://github.com/RIVeR-Lab/walrus`.

[34] "rosbridge_suite." `http://wiki.ros.org/rosbridge_suite`.

[35] "roslibjs." `http://wiki.ros.org/roslibjs`.

[36] "Webrtc." `http://www.webrtc.org/`.

[37] "webrtc_ros." `https://github.com/RobotWebTools/webrtc_ros`.

[38] "Angularjs." `https://angularjs.org/`.

[39] "Angular material design." `https://material.angularjs.org/`.

[40] "Introduction - material design." `http://www.google.com/design/spec/material-design/introduction.html`.

[41] M. Wills, S. Chernova, and T. Padır, "Stair detection for mobile robots using a depth camera," in *IROS*, 2015. (Under Review).

[42] S. Oßwald, J.-S. Gutmann, A. Hornung, and M. Bennewitz, "From 3d point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pp. 93–98, IEEE, 2011.

[43] T. J. J. Tang, W. L. D. Lui, and W. H. Li, "Plane-based detection of staircases using inverse depth," in *2012 Australasian Conference on Robotics and Automation (Dale Carnegie 3 December 2012 to 5 December 2012)*, pp. 1–10, Australian Robotics and Automation Association, 2012.

[44] M. Elmasry, "Stairs Detection and Modelling Using RGB-D Images," Master's thesis, Rheinische Friedrich-Wilhelms-Universitaet Bonn, 2013.

[45] T. Rabbani, F. van den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 5, pp. 248–253, 2006.

[46] "NERVE Center @ UMASS Lowell." `http://nerve.uml.edu/`.

A CAD Exploded View

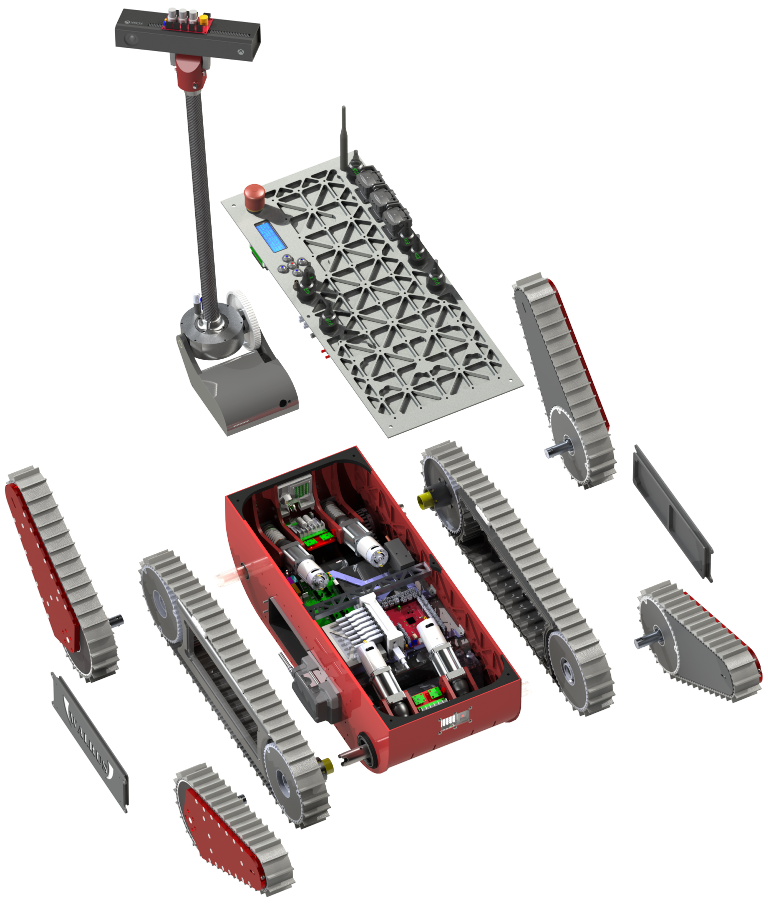


Figure 96: CAD Exploded Rendering

B Custom PCB Schematics

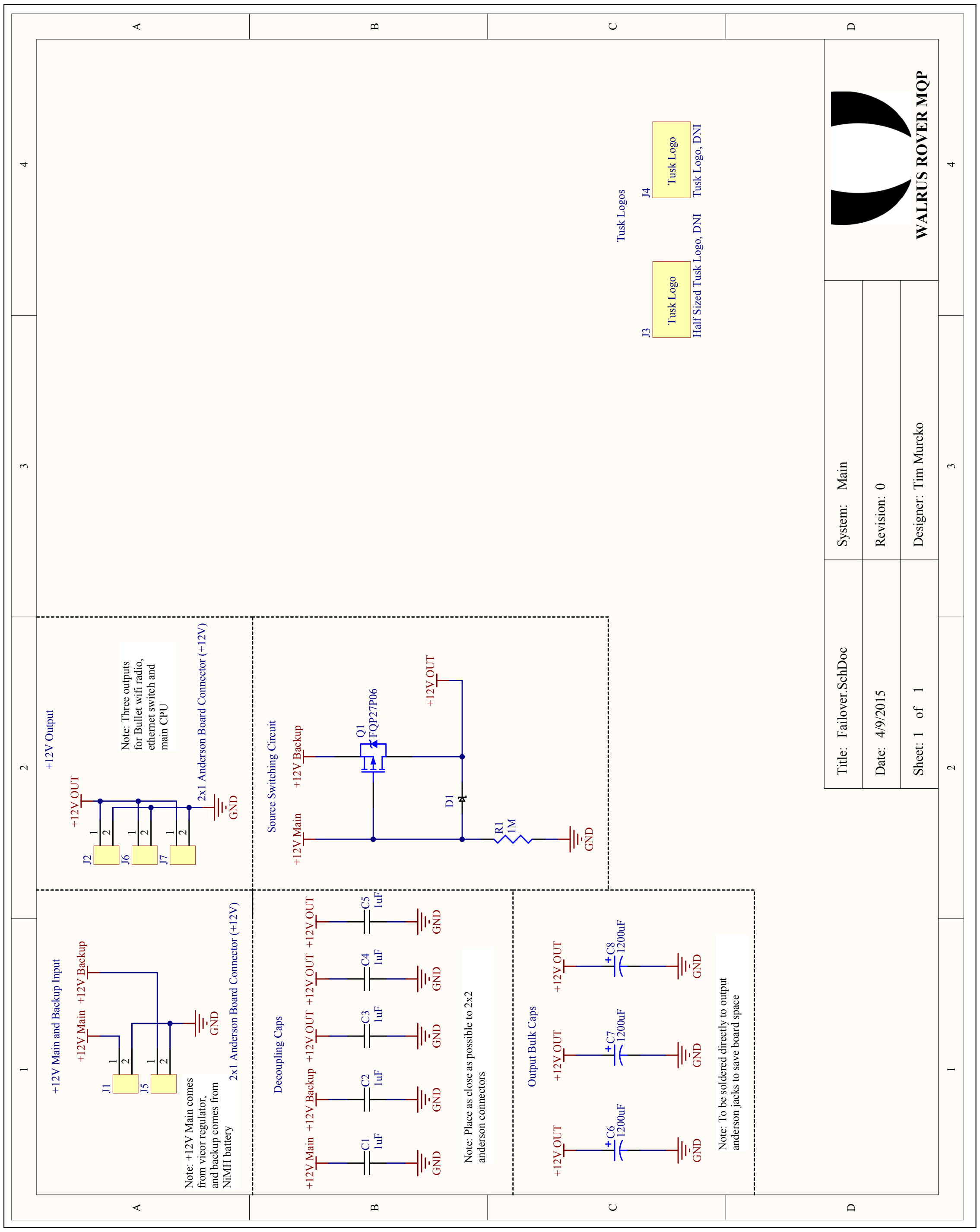

+12V Main and Backup Input
+12V Main
+12V Backup
J1
J5
GND
2x1 Anderson Board Connector (+12V)
Note: +12V Main comes from vicor regulator, and backup comes from NiMH battery
+12V Output
+12V OUT
J2
J6
J7
Note: Three outputs for Bullet wifi radio, ethernet switch and main CPU
2x1 Anderson Board Connector (+12V)
Source Switching Circuit
Q1
FQP27P06
D1
R1
1M
Decoupling Caps
C1 1uF
C2 1uF
C3 1uF
C4 1uF
C5 1uF
Note: Place as close as possible to 2x2 anderson connectors
Output Bulk Caps
C6 1200uF
C7 1200uF
C8 1200uF
Note: To be soldered directly to output anderson jacks to save board space
Tusk Logos
J3 Tusk Logo
Half Sized Tusk Logo, DNI
J4 Tusk Logo
Tusk Logo, DNI
WALRUS ROVER MQP
Title: Failover.SchDoc
System: Main
Date: 4/9/2015
Revision: 0
Sheet: 1 of 1
Designer: Tim Murcko

Figure 97: Failover Board Schematic

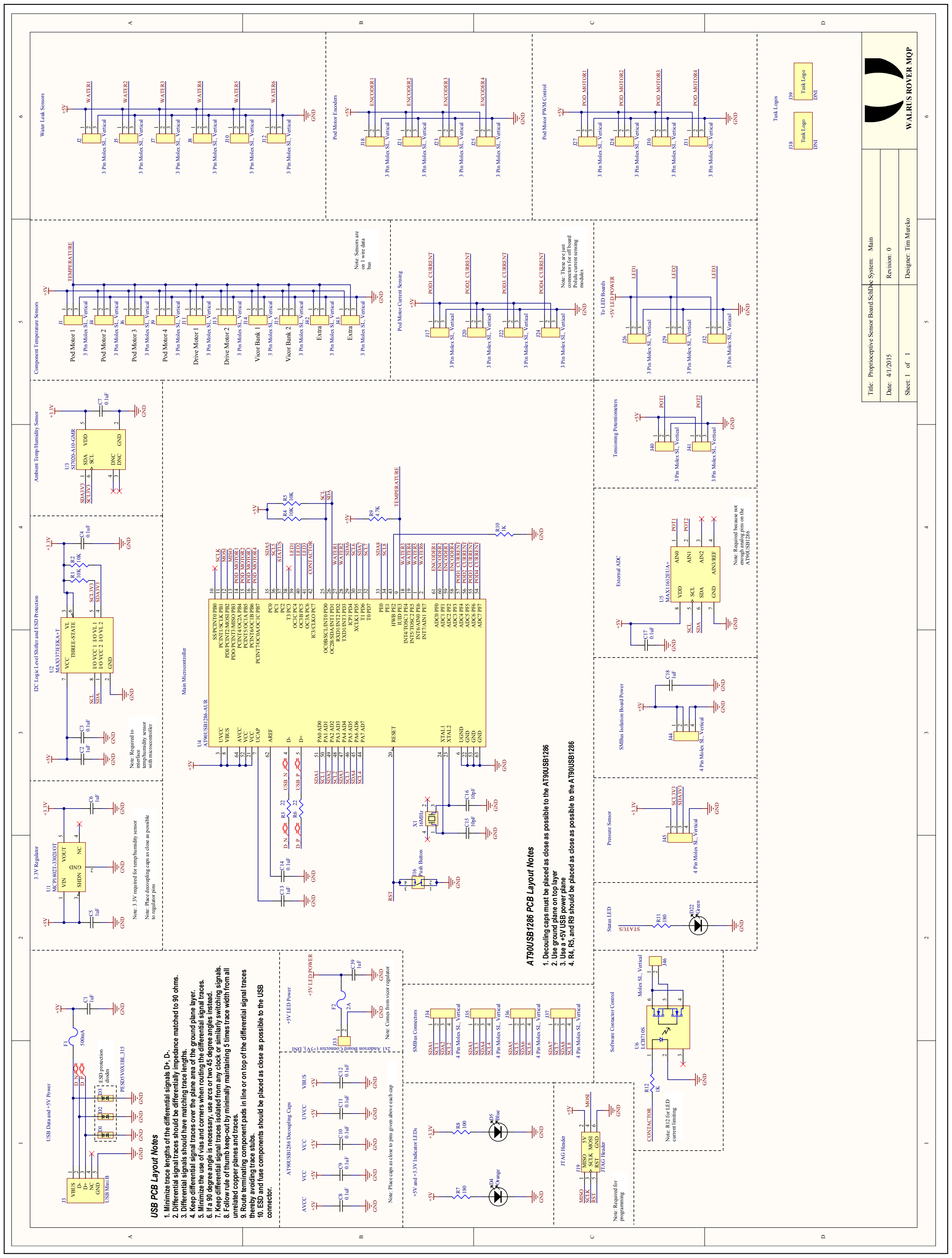

Title: Proprioceptive Sensor Board.SchDoc System: Main
Date: 4/1/2015
Revision: 0
Sheet: 1 of 1
Designer: Tim Marcko
WALRUS ROVER MQP
Water Leak Sensors
Pod Motor Encoders
Pod Motor PWM Control
Tank Logos
Component Temperature Sensors
Pod Motor Current Sensing
To LED Boards
Ambient Temp/Humidity Sensor
Tensioning Potentiometers
I2C Logic Level Shifter and ESD Protection
Main Microcontroller
External ADC
SMBus Isolation Board Power
3.3V Regulator
Pressure Sensor
Status LED
USB Data and +5V Power
+5V LED Power
SMBus Connectors
Software Contactor Control
AT90USB1286 Decoupling Caps
+5V and +3.3V Indicator LEDs
JTAG Header
USB PCB Layout Notes
1. Minimize trace lengths of the differential signals D+, D-.
2. Differential signal traces should be differentially impedance matched to 90 ohms.
3. Differential signals should have matching trace lengths.
4. Keep differential signal traces over the plane area of the ground plane layer.
5. Minimize the use of vias and corners when routing the differential signal traces.
6. If a 90 degree angle is necessary, use arcs or two 45 degree angles instead.
7. Keep differential signal traces isolated from any clock or similarly switching signals.
8. Follow rule of thumb keep-out by minimally maintaining 5 times trace width from all unrelated copper planes and traces.
9. Route terminating component pads in line or on top of the differential signal traces thereby avoiding trace stubs.
10. ESD and fuse components should be placed as close as possible to the USB connector.
AT90USB1286 PCB Layout Notes
1. Decoupling caps must be placed as close as possible to the AT90USB1286
2. Use ground plane on top layer
3. Use a +5V USB power plane
4. R4, R5, and R9 should be placed as close as possible to the AT90USB1286

Figure 98: Proprioceptive Sensor Board Schematic

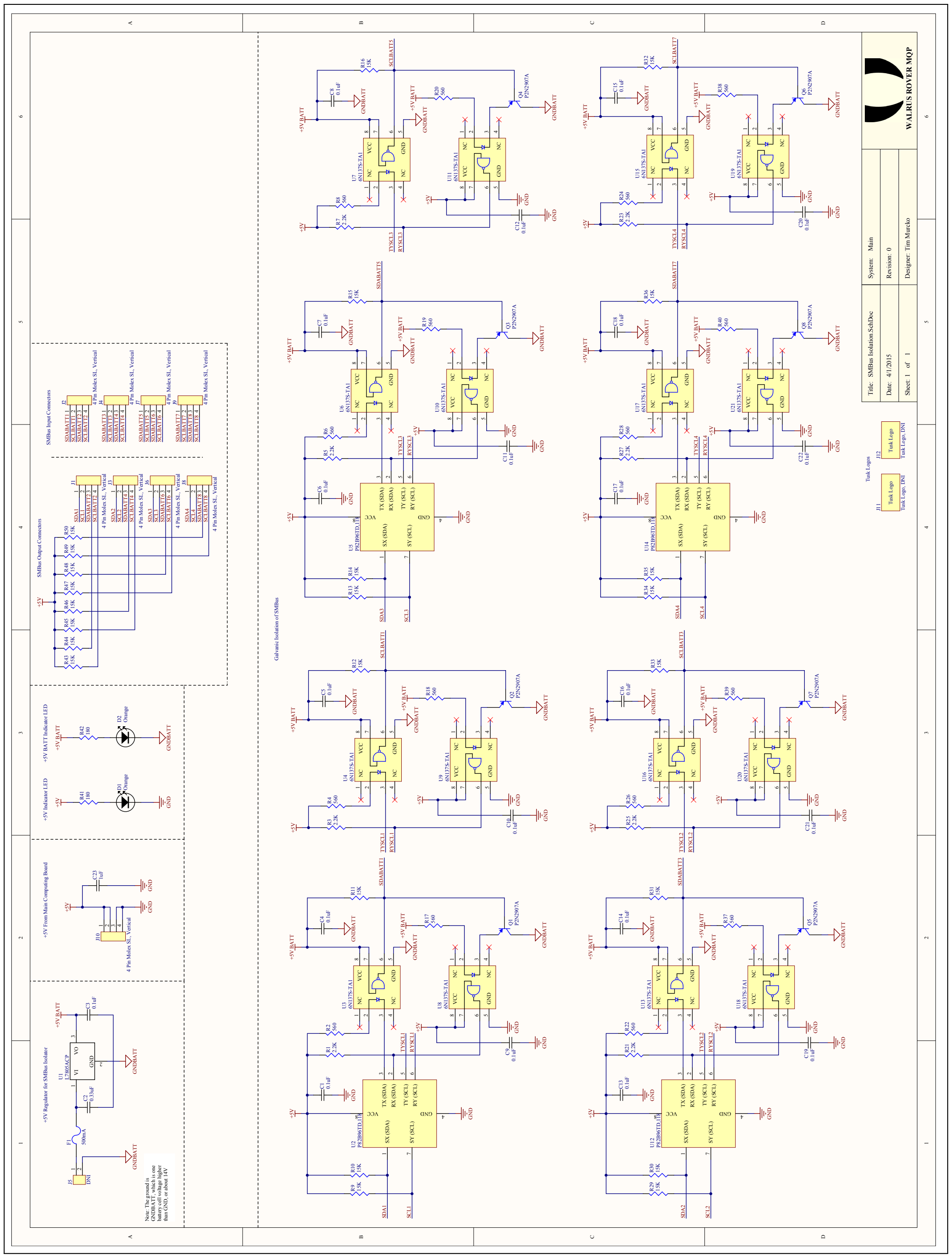
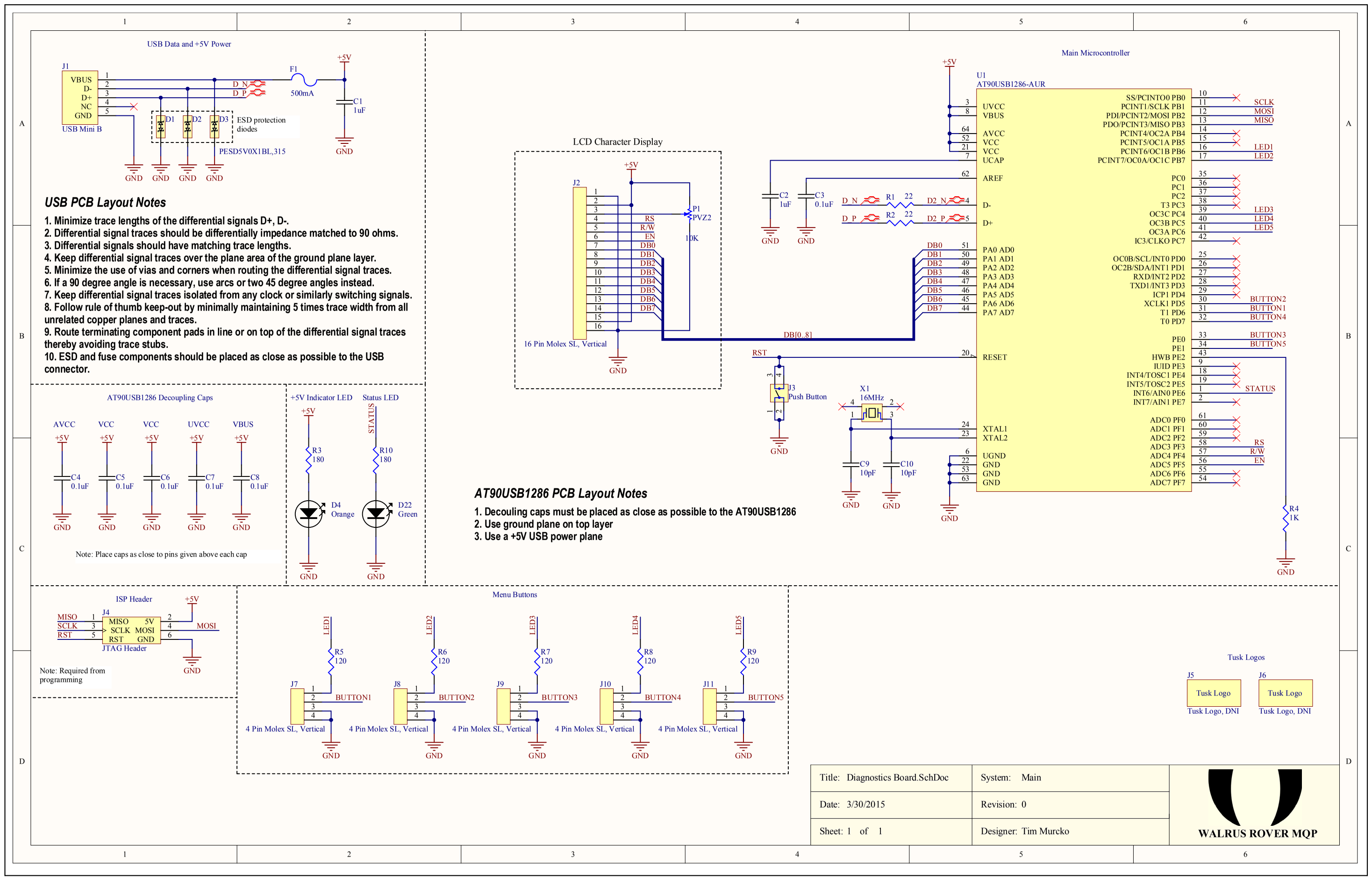


Figure 99: SMBus Galvanic Isolation Board Schematic



USB PCB Layout Notes

1. Minimize trace lengths of the differential signals D+, D-.
2. Differential signal traces should be differentially impedance matched to 90 ohms.
3. Differential signals should have matching trace lengths.
4. Keep differential signal traces over the plane area of the ground plane layer.
5. Minimize the use of vias and corners when routing the differential signal traces.
6. If a 90 degree angle is necessary, use arcs or two 45 degree angles instead.
7. Keep differential signal traces isolated from any clock or similarly switching signals.
8. Follow rule of thumb keep-out by minimally maintaining 5 times trace width from all unrelated copper planes and traces.
9. Route terminating component pads in line or on top of the differential signal traces thereby avoiding trace stubs.
10. ESD and fuse components should be placed as close as possible to the USB connector.

AT90USB1286 PCB Layout Notes

1. Decouling caps must be placed as close as possible to the AT90USB1286
2. Use ground plane on top layer
3. Use a +5V USB power plane

Figure 100: Diagnostics Board Schematic

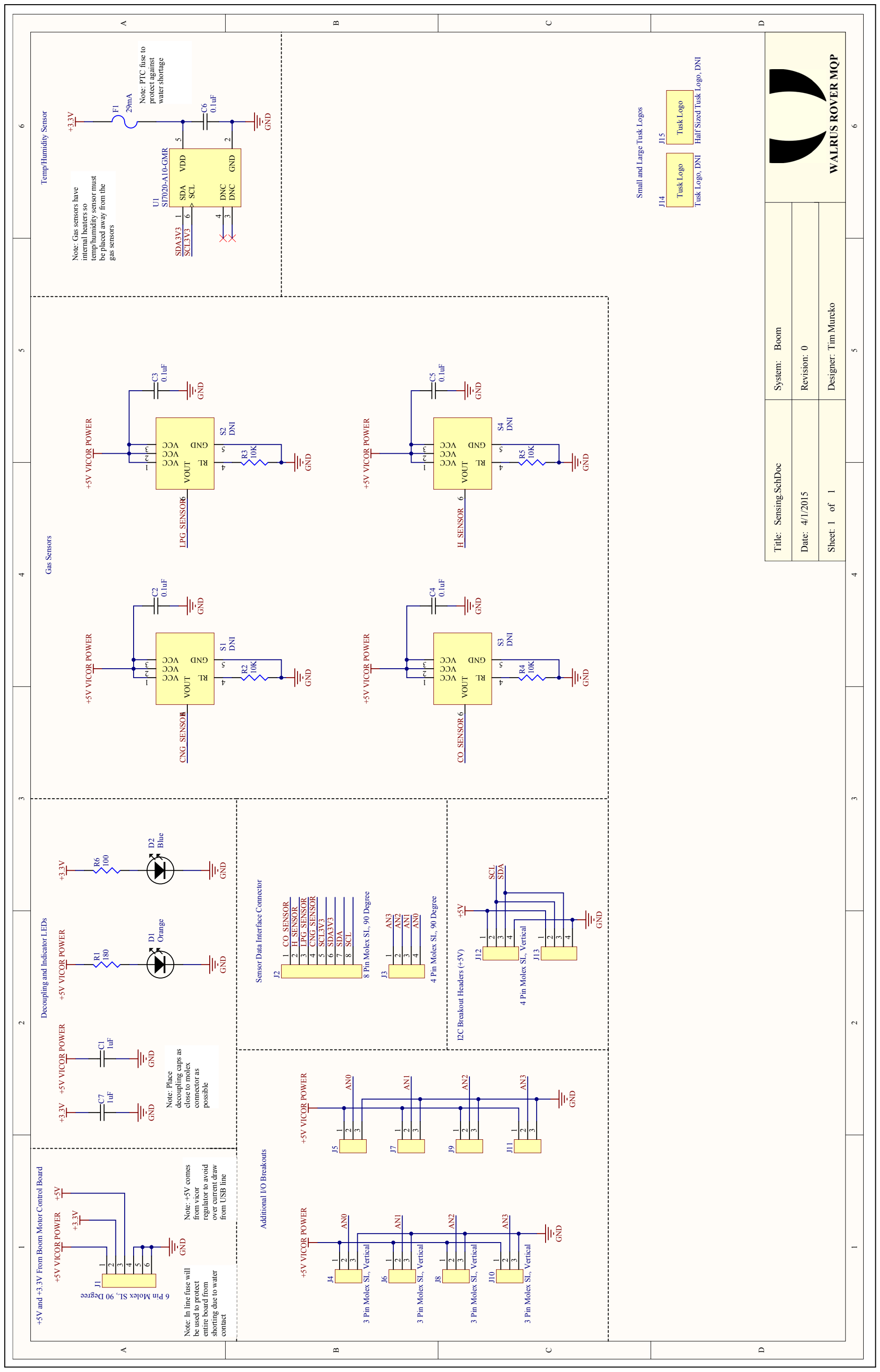


Figure 101: Environmental Sensor Board Schematic

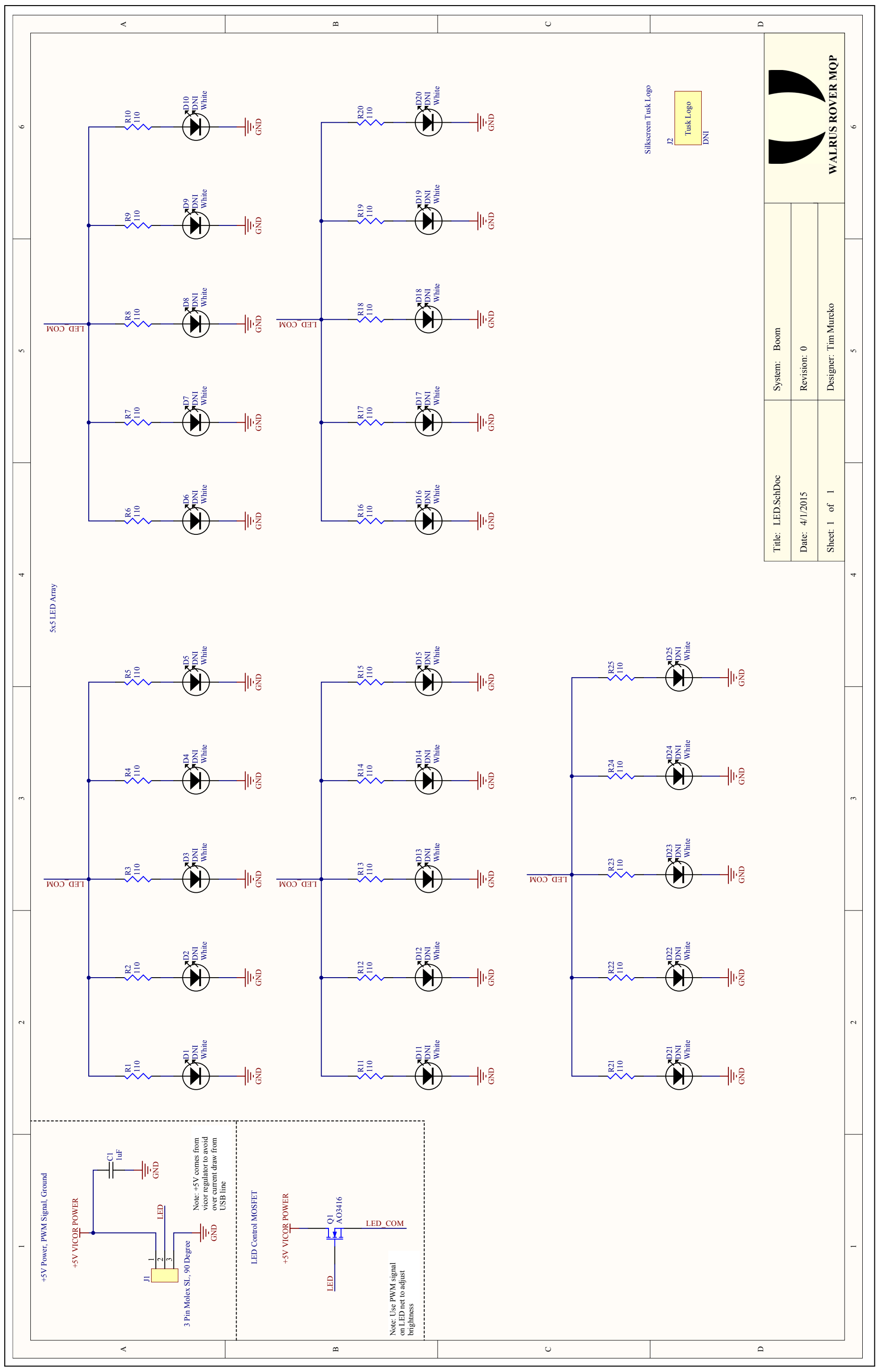


Figure 102: Diagnostics Board Schematic

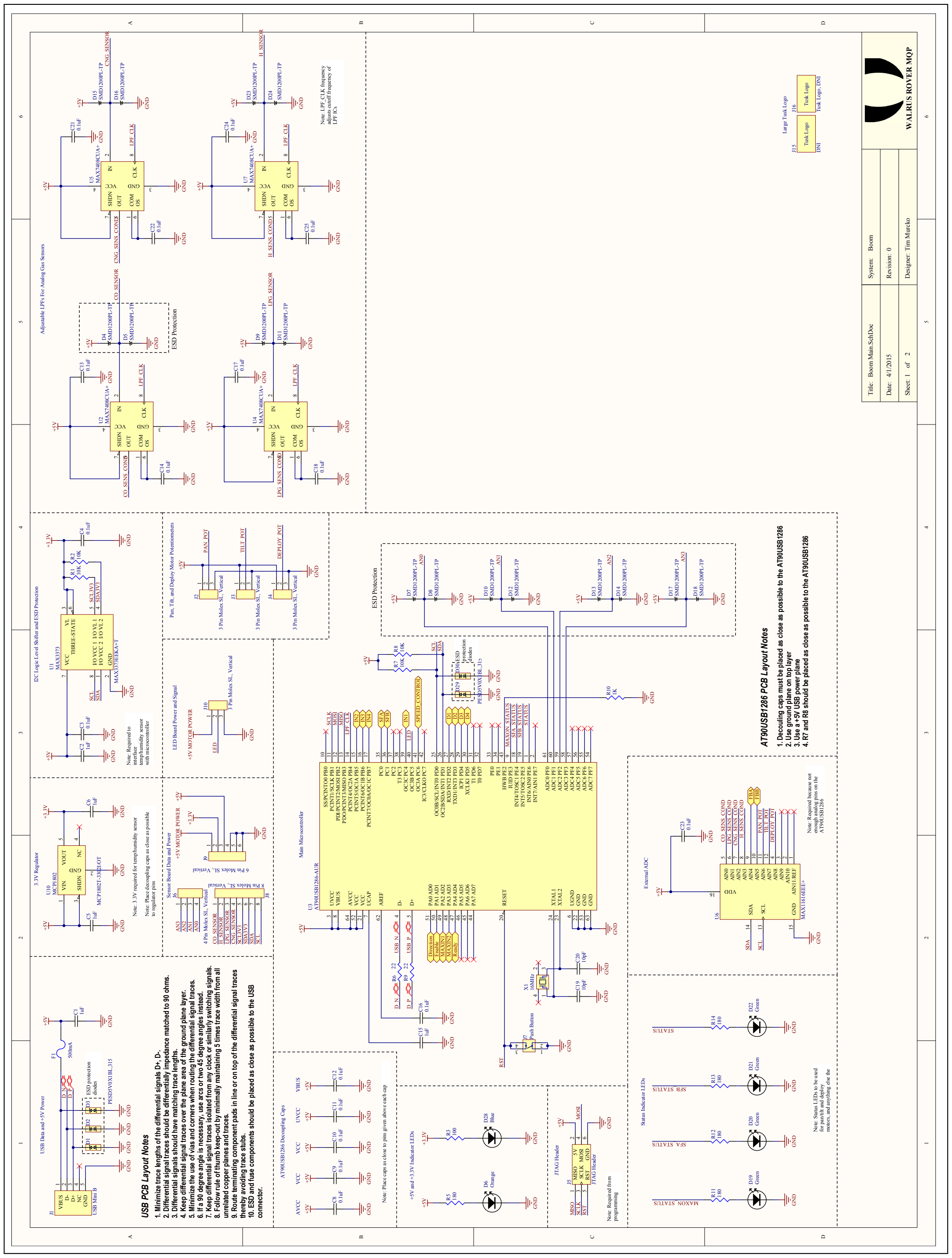


Figure 103: Boom Motor Control Board Schematic (Page 1)

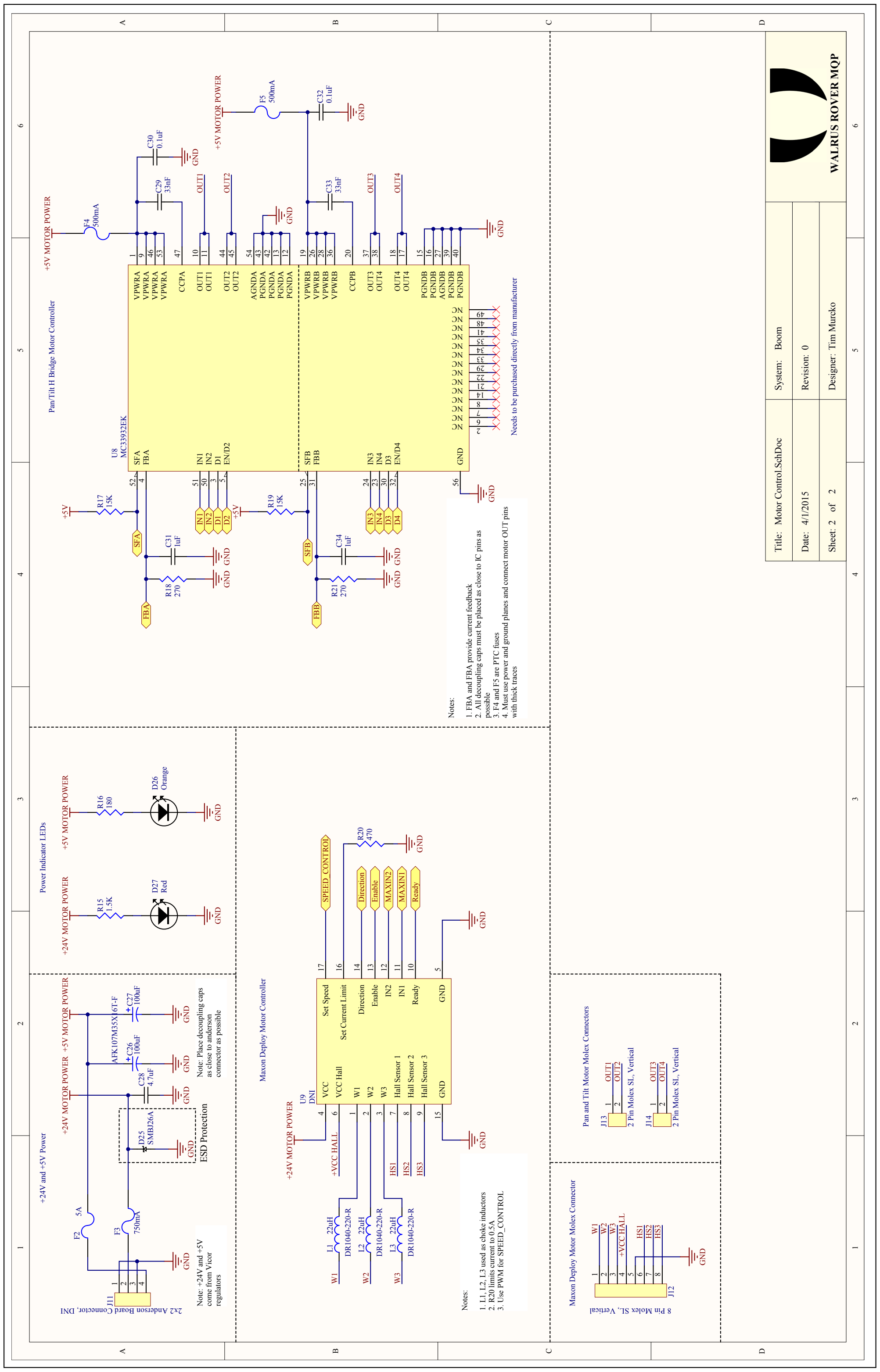


Figure 104: Boom Motor Control Board Schematic (Page 2)

C Stair Detection Results

	Stair 1	Stair 2	Stair 3	Stair 4	Stair 5	Stair 6	Stair 7
Estimated Rise	17.5 ± 0.8 cm	18.2 ± 0.2 cm	18.5 ± 0.1 cm	16.5 ± 0.6 cm	17.3 ± 0.2 cm	n/a	n/a
Estimated Run	27.8 ± 0.7 cm	26.8 ± 0.3 cm	27.3 ± 0.2 cm	28.2 ± 0.9 cm	28.2 ± 0.3 cm	n/a	n/a
Estimated Width	104.0 ± 2.1 cm	108.0 ± 6.3 cm	73.6 ± 37.8 cm	98.8 ± 23.0 cm	129.0 ± 2.1 cm	n/a	n/a
Ground Truth Rise	18.5 cm	18.5 cm	18.5 cm	17.8 cm	17.8 cm	15.4 cm	17.8 cm
Ground Truth Run	26.5 cm	26.5 cm	26.5 cm	28.0 cm	28.0 cm	28.6 cm	28.0 cm
Ground Truth Width	117 cm	117 cm	117 cm	137 cm	148 cm	189 cm	148 cm
Rise Error	-1.0 ± 0.8 cm	-0.3 ± 0.2 cm	0.0 ± 0.1 cm	-1.3 ± 0.6 cm	-0.5 ± 0.2 cm	n/a	n/a
Run Error	1.3 ± 0.7 cm	0.3 ± 0.3 cm	0.8 ± 0.2 cm	0.2 ± 0.9 cm	0.2 ± 0.3 cm	n/a	n/a
Width Error	-13.0 ± 2.1 cm	-9.0 ± 6.3 cm	-43.4 ± 37.8 cm	-38.2 ± 23.0 cm	-19.0 ± 2.1 cm	n/a	n/a
False Negatives	0/10	0/9	0/10	0/11	0/10	7/7	10/10
Incorrect Detections	0/10	0/9	0/10	0/11	1/10	0/7	0/10
Plane Segmentation Runtime	124 ± 7 ms	270 ± 8 ms	333 ± 6 ms	289 ± 5 ms	219 ± 3 ms	129 ± 5 ms	140 ± 6 ms
Parameter Estimation Runtime	7.8 ± 1.2 ms	7.4 ± 0.4 ms	8.8 ± 0.6 ms	8.9 ± 0.4 ms	10.7 ± 1.8 ms	3.1 ± 0.5 ms	5.5 ± 1.3 ms

Table 6: Results of algorithm on different stair datasets

D Project Timeline

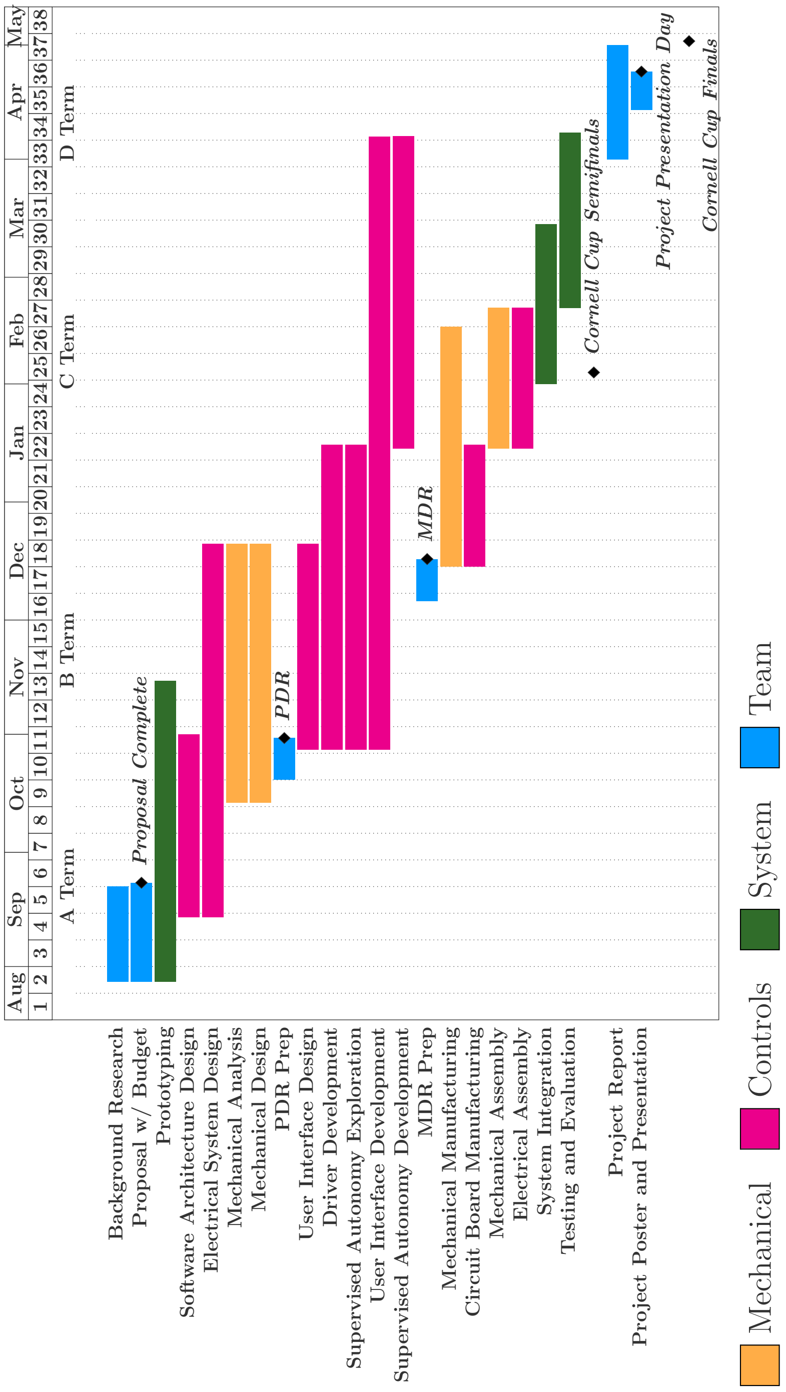


Figure 105: Project Timeline

E Budget

Sub-System	**Optimal Cost**	**Minimal Cost**
Actuation	\$3,300	\$2,000
Sensing & Vision	\$2,000	\$450
Computing	\$1,500	\$800
Electrical & Battery	\$6,500	\$2,000
Hardware	\$6,300	\$3,000
Total	\$19,900	\$8,250

Table 7: Budget