DESIGN AND FABRICATION OF AN AUTONOMOUS SEARCH AND RESCUE ROBOT

Seven Nation Army: Rocky
MAE 322: Mechanical Design

Due Date: May 13, 2014

Course Project by

FIYINFOLUWA AKINLAWON
DAVID BECK
ANNIE CARDINAL
ALEX CREELY
DAVID HARRIS
ERIC MATERNIAK
VICTOR PRATO

2014
PRINCETON UNIVERSITY

Contents

List of Figures

List of Tables

1 Executive Summary

Imagine this: A devastating earthquake hits, and dozens of people are injured and trapped inside buildings. It is too dangerous for rescue personnel to enter, and time is running out for the people inside. They need medicine, and fast. Enter Rocky, the Search and Rescue Robot (SaRR). With Rocky's ability to carry a medical kit, traverse rubble and obstacles, and autonomously deliver the kit to the target, the lives of the trapped victims can be saved.

Our SaRR, Rocky, is a 45 pound robot with a 20" by 10" chassis that is able to deliver medicine to a victim in approximately two minutes. It is capable of retrieving a medical kit, navigating an obstacle course, and autonomously traversing two 6 inch steps followed by a foot-high drop and navigating through a chute without hitting the walls. It then detects a flashlight, mimicking a light beacon waved by an injured person, and autonomously delivers the medical kit to the victim.

Rocky is designed to be robust and simple. The drivetrain is used for both general travel and wall traversal, and the arm requires only one motor as the result of a passive claw. The first highlight of Rocky's design is the tri-lobe front wheels, which operate in sync with each other and are passive until they are used to climb the wall. While driving, only one of the tri-lobes is in contact with the ground because it ensures better weight distribution over the back wheels.

The second highlight is Rocky's passive claw, which reduces the number of necessary motors and makes Rocky simple and robust. Two prongs hook under the handle of the medical kit and lift it up into a safe storage position.

The rear wheels are bicycle-style, providing traction and steering control on the floor and a boost to climb over the wall. A low gear ratio provides control of maneuverability. When traversing the wall, the robot's center of gravity is just past the top of the wall, causing it to slide over instead of breaching. Two metal antenna projections on the front of the robot safely prevent the robot from tipping over when descending the 12" drop.

In order to navigate the chute and detect the light autonomously, two light sensors and four proximity sensors were chosen to allow accurate autonomous navigation.

Rocky is a simple and robust SaRR and is reliable at completing the task at hand.

2 Introduction

2.1 Objectives of the Design

Rocky's objective is to retrieve a medical kit and deliver it to a basket at the end of an obstacle course. This basket models a person, trapped under rubble and in need of assistance, holding a flashlight. In an attempt to achieve this goal, the robot has to first retrieve the medical kit, navigate an obstacle course, traverse a wall barrier, navigate through a chute, and then deliver the medical kit. For each of the sub tasks within the main objective, the robot must adhere to certain criteria, which will be elaborated upon below:

2.1.1 Retrieving the Medical Kit

The robot starts in a designated area and will be manually driven for about 10ft. before the med kit can be manually retrieved. The med kit is modeled as a 2.5lb 4" cube with a 4" handle shown in Figure 1, and starts in a fixed position on the floor. The retrieval mechanism must be able to ensure security of the kit through the rest of the course.

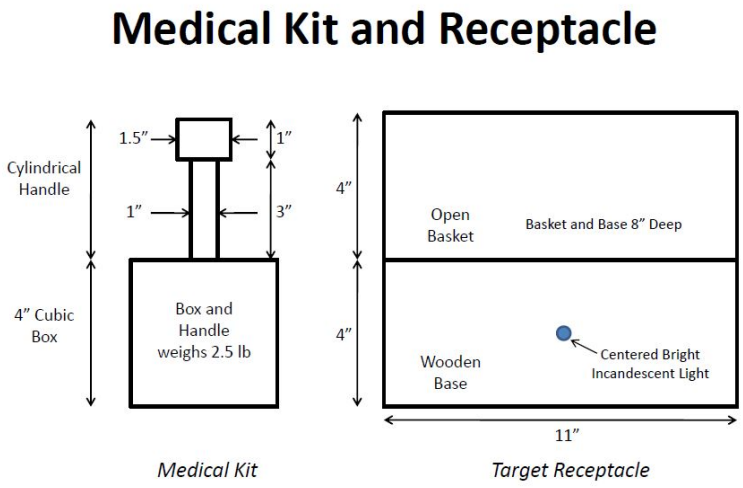


Fig. 1: Schematic of the Medical Kit and Receptacle [NOTE: not to scale]. The med kit is a 4" cubic box, with a 3" cylindrical handle with a 1" diameter and a 1.5" topper. This creates a design constraint for the passive gripper that it must be x wide, where $1" < x < 1.5"$. The front view of target receptacle shown, with the light source beneath the open basket. The med kit is to be delivered to the receptacle.

2.1.2 Navigating the Obstacle Course

After retrieving the med kit, the robot must manually navigate an obstacle course that ends in a demarcated area. The obstacle course Rocky will travel through involves turns and curves marked out by cones. In addition to the cones, there is a fair amount of debris along the course through which the robot should navigate smoothly in an attempt to display its agility and maneuverability. A schematic of the course is shown in Figure 4.

2.1.3 Breaching the Wall

On arriving at the area prior to the wall, the robot must be in autonomous mode and must autonomously traverse the wall and arrive on the other side with all parts and the med kit intact. The wall has two wooden steps each 6" high, the first step is 6" wide and the second step is 1" wide. In addition, the wall has a metal bar fixed 36" above the second step that may also be used for traversing if necessary. Figure 2 shows the wall in more detail.

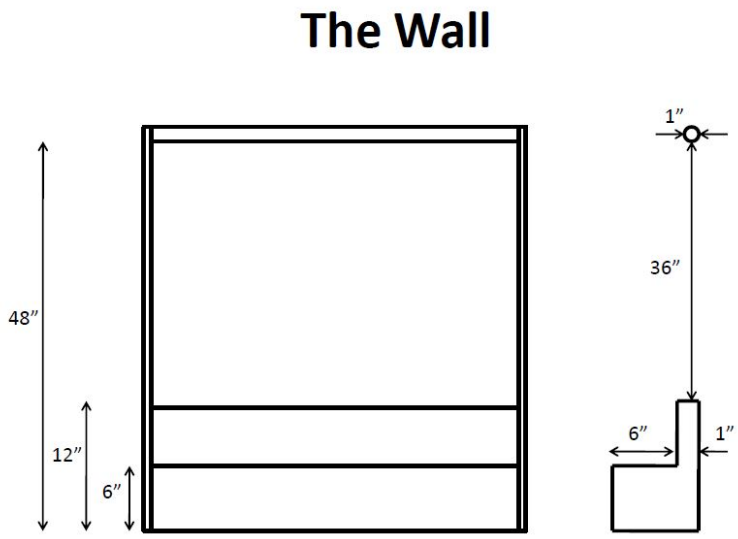


Fig. 2: Schematic of the Wall [NOTE: not to scale]. The front view shows the wall with two 6" steps and a bar 48" above the base. The side view shows the lower step with a width of 6" and the other step with a width of 1". The horizontal bar is 1" in diameter

2.1.4 Navigating the Chute

As the robot descends the wall it must autonomously navigate through the chute. The chute shown in Figure 3 is a 15' long, 3' wide pathway created using unpainted 12" high, $\frac{3}{4}$" thick plywood. This autonomous navigation of the robot should involve the use of proximity sensors to allow navigation through the chute.

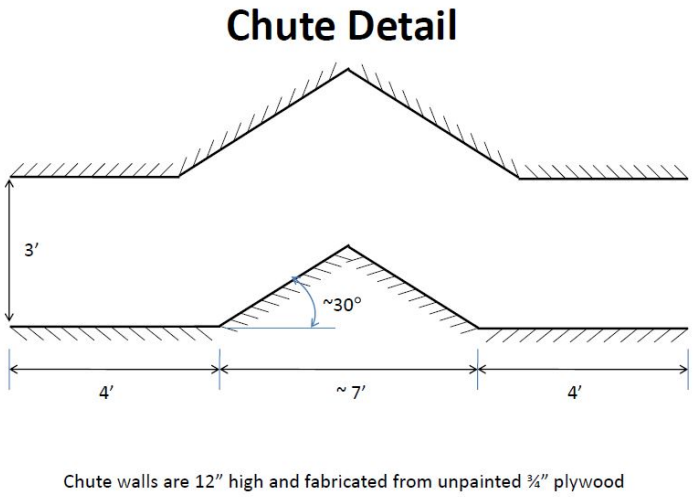


Fig. 3: Schematic of the Chute [NOTE: not to scale]. Chute is 15' long and 3' wide, with two 30^o bends.

2.1.5 Delivering the Medical Kit

The robot autonomously delivers the med kit to a basket attached to a light source (Figure 1). After navigating the chute, the robot must autonomously find the light source and navigate to the source. On arrival, the robot must deliver the med kit to the basket. The distance from the end of the chute to the light source is about 20'. A schematic of the entire course can be seen in Figure 4.

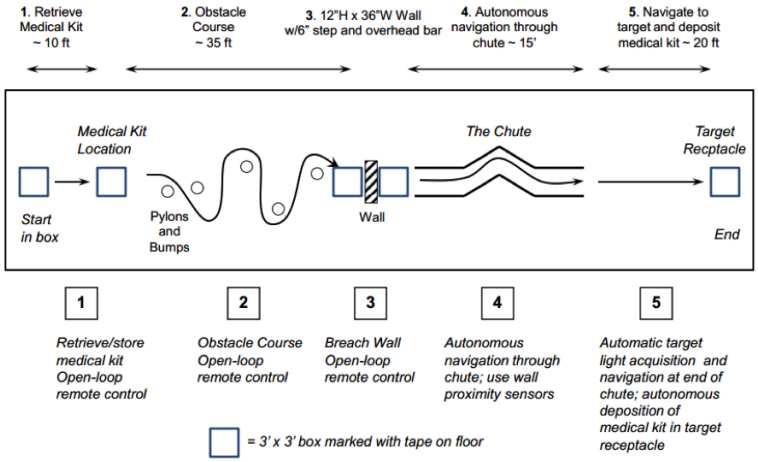


Fig. 4: Schematic of the Course [NOTE: not to scale]. The figure shows the entire course from start to finish, showing the relative positions of the various obstacles in the course and the approximate length of each challenge and obstacle.

2.2 Preliminary Research

Search and Rescue Robots must be robust in function and ability in order to respond effectively in a variety of situations. In an attempt to encourage such robustness, the Intelligent Systems Division of the National Institute of Standards and Technology (NIST) set criteria for urban search and rescue robots to achieve. They are:[4]

- Mobility: They must be able to traverse an array of uneven surfaces with drops and steps and sharp turns and take advantage of unconventional routes.
- Sensing: They must be able to sense the change in environmental factors like light and surface geometry. In addition it should be able to sense the presence of victims manifested in multiple ways including: acoustic, thermal and visual.
- Knowledge Representation: It should be able to build up a map of the site as it traverses it and using knowledge determine victim and hazard locations and potential quick exit routes.
- Planning: Should be able to plan the optimal route for exploring the given area, taking into account the time-critical nature of the work and the maximizing the number of victims rescued.
- Autonomy: Should be able to operate autonomously, as the radio controllers may be out of reach. In such a scenario, the robots must be able to continually update their plans and objectives.
- Collaboration: In the scenario where there are multiple robots exploring a similar site (which is usually the case), the robots should be able to talk to each other to optimize travel and reduce redundancy.

These rigorous criteria were set for a course pre designed by the NIST, but can be applicable to just about any ground search and rescue robot.

Using these criteria as a benchmark, a standard has been set for the efficacy of search and rescue robots. This has led to an array of concepts put forward in theory and in practice. Drawing inspiration from nature, a group of Japanese engineers developed a snake-like rescue robot, built by connecting multiple crawler vehicles in series. It has the main advantages of climbing over obstacles, moving through narrow spaces and adapting to irregular surfaces. [5]The concepts behind this promising robot lie true to the fundamentals of the search and rescue robots. However, more work has been proposed on the design, actuation and control of these robots in order to make them truly

Fig. 5: Amoeba robot

deployable.[6] In addition to the snake robot,
other nature inspired crawler robots are in the design phase including: lizard-type robots that adhere to walls [6] and amoeba-like robots (Figure 5) that use their entire outer skin as a means of propulsion.[7] It posses the advantage of being able to squeeze into areas that many other robots could not, but still maintain the integrity of its functions. Dennis Hong, the researcher in charge of the project expresses that the robot is like a 3-D tank tread [7], alluding to the idea that tank treads are preferable for search and rescue robotic purposes.

Fig. 6: A robot used in the search and rescue effors of 9/11

Hong's tank tread assertion for a conceptual design is supported by robots that have been actually used in disasters. Natural and man-made disasters have been the major propellers for search and rescue robot advancement, but they have not been the only platform. International competitions aid in that regard as well.[5] An array of ground robots have been used in disasters and they support the use of tank treads and multiple actuators for gripper, arm and body mobility.[6] Although these robots were mostly successful in their missions including: the World Trade Center (Figure 6), La Conchita Mudslide and the Sago Mine Disaster, there are design lessons to be learned: [6]

- Robots should have a safety rope or if wireless should have strong enough communications to ensure constant contact.
- Robots should be invertible in order to aid mobility and agility
- Robots should have enough water proofing to increase the range of applications
- Imaging is generally desirable
- Autonomic functions in the robot are desirable

2.3 Design Concept

By understanding the objectives of the test course, using the criteria set by the NIST, drawing inspiration from conceptual design robots and learning from the search and rescue robots that have been used in practice, an iterative design process for Rocky began.

Traversing the wall is arguably the most significant challenge in the course and was the focal point of the design. In order to maintain the focus on simplicity and to minimize the number of moving parts, any design involving the use of the overhead bar on the wall was ruled out, mainly because it would involve a mechanism different from the drivetrains which creates more avenues for potential failure. Therefore the design shifted focus to using the drive train to get over the wall (Figure 2).

Tank treads seemed to be a good choice for the drivetrain because they provide great traction and maneuverability that make them especially agile. This is why they are popular in industrial search and rescue robots as explained earlier. However, the use of tank treads was rapidly ruled out because the treads are expensive [8]and would have stretched our $500 budget thin. In addition the tank treads fell against the primary goal of simplicity and minimizing the number of moving parts, and would have been difficult to implement.

Tires were chosen as an alternative to tank treads for the drive train. Tires are conventionally used for driven robots outside and within the search and rescue field, so they made a good fit for the robot given the objectives it has to perform. However a decision had to be made on what size of tires to use on the robots. To get over the 6" step, a wheel with a radius > 6" was required, therefore a 14" wheel was chosen.

In an attempt to ensure wall traversal capabilities, step-climbing mechanisms were considered for the front wheels. Inspiration was drawn from stair-climbing robots [9] and incorporated into the design for the drivetrain. By so doing, Rocky is design to traverse the wall by climbing over it with its front wheel and driving over it with its back wheels. While driving around the course, Rocky will be rear wheel driven to ensure better manuerability and proximity sensing, however when getting over the wall, the tri-lobe step climbing front wheels will move in sync to climb over the wall.

The other main component of the course is picking up and delivering the med kit. As explained earlier, actuated grippers and arms are the norm for search and rescue robots because they guarantee security of the package. However, in an attempt to stay true to a design philosophy of simplicity, a passive gripper with and motorized arm was designed as the piece to pick and deliver the med kit. The kit slides into the crevice of the gripper as the robot drives forward, and rests on the arm as the kit is lifted onto the chassis of the robot. The arm then rests on the robot's chassis (Figure 14).

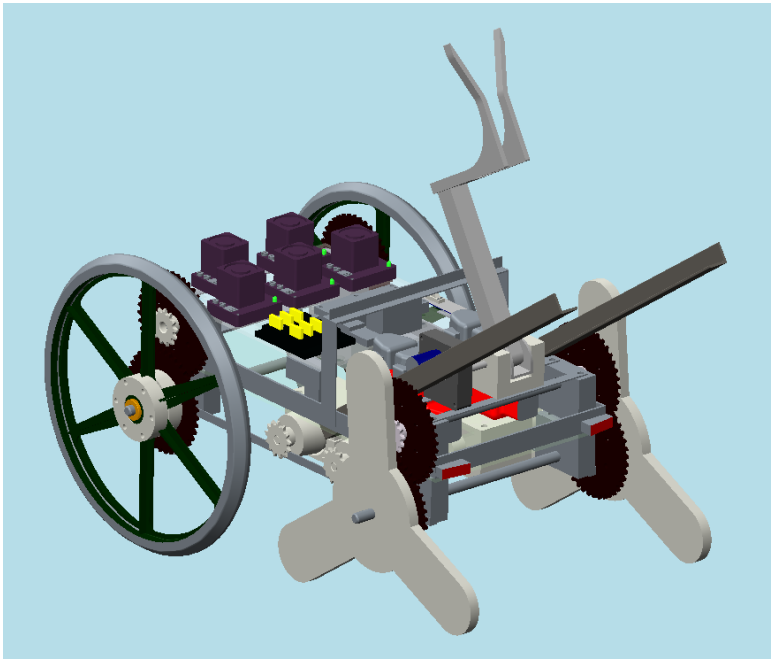


Fig. 7: Schematic of the Complete SaRR

3 Detailed Design and Analysis

3.1 Design Process

In order to determine the dimensions for the chassis, we did several experiments using foam board to model the chassis and wheels climbing over the wall. From these experiments, we determined that a chassis length of 20" would be ideal. This length was chosen so the two sets of wheels would not overlap, and that the center of mass of the robot would be far enough forward to ease the descent of the robot from the top of the wall. The 10" width of the chassis was determined based on the amount of space needed to store the four batteries and motor control system.

To prevent Rocky from flipping upside down when it descends the 12" drop on the back side of the wall, two antennas were added to the robot. These antenna were made from steel, and were set at a $\approx 24^o$ angle with respect to the chassis. The two antenna are set to a length of 22". Their purpose is to make the robot slide forward and prevent the robot from flipping when it is in a nearly vertical position with the rear tires still on top of the wall.

3.2 Electric DC Motors

The actuator of choice for this robot were brushed permanent magnet DC motors. They are simple, cheap, and sufficiently robust for the task objectives. The robot was designed based on the specs of the motors that were freely provided because purchasing more would be prohibitively expensive. Therefore, Rocky is equipped with four Low Voltage Johnson DC Motors (Model HC613LM) to actuate each wheel and 1 Molon Permanent DC magnet gearmotor to actuate the arm. Each Johnson motor weighs about 0.8 lbs while the Molon motor weighs approximately 1 lb. The Johnson Motors were procured from LDX112PK Black & Decker Drills with the gearbox and clutch still attached. The Molon gearmotor was a pre-existing item in the shop. The Johnson Motor was shown to be have a stall torque of 0.8 lb-ft via a testing station in the shop. The Molon gearmotor is rated with a stall torque of 40 in-lb.

Both motors required a 12 V power supply which was satisfied by four 12 Volt Lithium ion batteries procured from the same LDX112PK Drills as the Johnson motors. The current levels for the motors were not known, and documentation on the specific model was not available. However, information for a similar Johnson Model (HC613G) was obtained that indicated a Stall Current of 88 Amps and a No Load Current of 3.40 Amps. There were multiple instances where a chain issue caused a stall current that overloaded the 20 Amp rated fuses in our circuit. The Molon Motor had a continuous current rating of 0.68 Amps. From these values the Peak and Average Power Ratings were calculated for use in analyzing power requirements. The motor temperature was not a concern since their operational timeframe is relatively low.

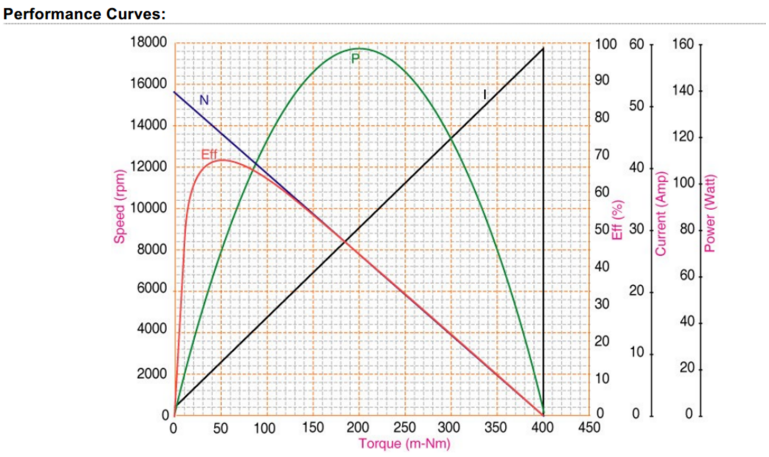


Fig. 8: Graph shows DC motor performance curves. The black curve is current against torque, the red is efficiency against torque, green is power against torque and blue is speed against torque.

Based on the constraints of the motors provided we designed the robot accordingly. The design constraint was that we had to ensure that the front wheels had enough lifting torque to traverse the wall obstacle. Therefore we designed a chain drive that increased the torque of the motors well above the required value for wall traversal, allowing for 25% losses before the stall torque was reached.

The gearboxes of these motors were not analyzed as the stall torque was the main parameter we needed.

The motors are controlled by the Arduino Servos library by writing the pulsewidth desired to the motors, the range is 1000 to 2000 with 1515 being the stall pulsewidth.

3.3 Power Requirements and Power Regulation

An analysis was done to determine the battery life based on the respective power draws of each major electrical component. The components included were the two different motor controllers, Sharp Sensors, Photo sensors, Johnson Motors, Molon Motor, and RAMB T3 Microcontroller. The fact that the RAMB T3 Microcontroller is in reality powered separately by six NiMH batteries was ignored to produce a worst case scenario. The peak or instantaneous power spikes were not provided in documentation for the Victor 884 series, the sensors, or the Molon gearmotor. The batteries were modeled as 4 packs wired in parallel with each supplying a current of 1750 mAh. An online calculator (Figure 9) was used to produce the results, and necessary parameters were divided into motor requirements, electronics requirements, and battery setup.

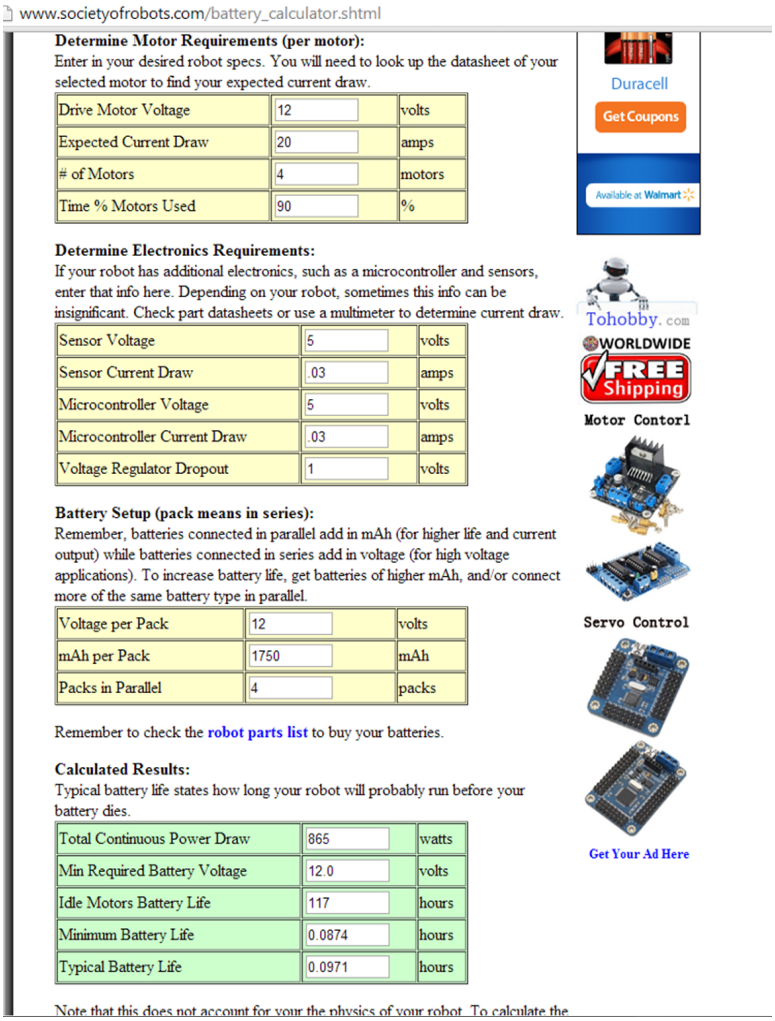


Fig. 9: Online calculator used to determine power requirements. The results are in the bottom box.

The overall power consumption of these components is on average 3.8 kW and at peak times at least 10.2 kW. Based on the robot's time of operation, the average power consumption is at a rate of 346.68 kWhr.

3.4 Electronics Layout

Rocky's electrical components are neatly laid out on a laser-cut piece of acrylic with holes for wires and screws. This keeps everything organized and safely attached. In addition, all key electrical components and acrylic sheet were damped with rubber tubing to add extra support and cushioning.

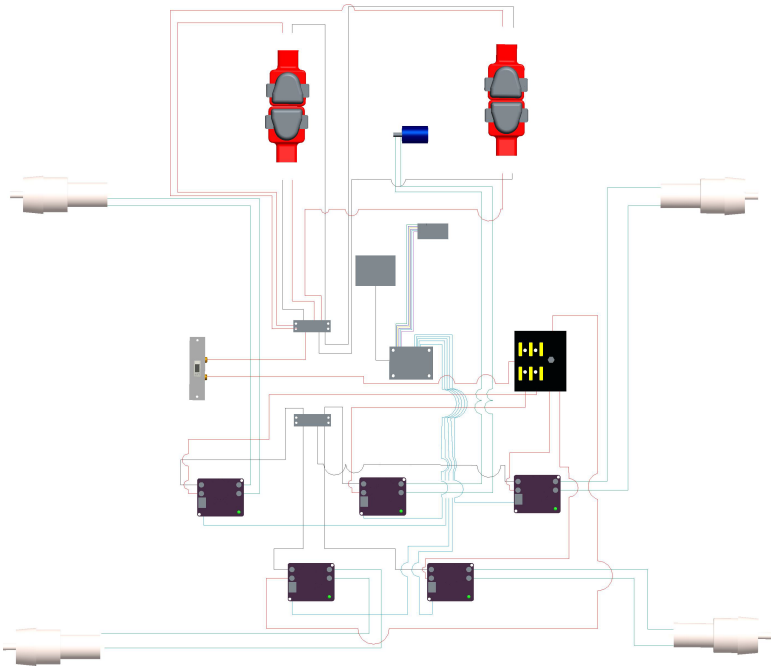


Fig. 10: Detailed wiring diagram showing the relative position and wiring of different components on the electronics layer. There are 5 motor controllers, 4 drill motors, 1 geared down motor and four battery packs. In addition to the connection blocks and arduino.

3.5 Infrared Proximity Sensors

Rocky uses four proximity sensors, made by Sharp, for autonomous navigation and operation. The sensors use infrared technology to obtain a measurement ranging from 0 to 1023, and these measurements correspond to various distances, as shown in the test plots below. The relationship is not linear, and results smaller than 3 inches should be discarded. From 0 to 3 inches, there is a sharp linear increase up until about 900, and after 3 inches, the points follow a gentle curve for about two feet. A higher value corresponds to a small distance. As the plots below convey, the sensors are fairly consistent with each other and follow nearly the same contour.

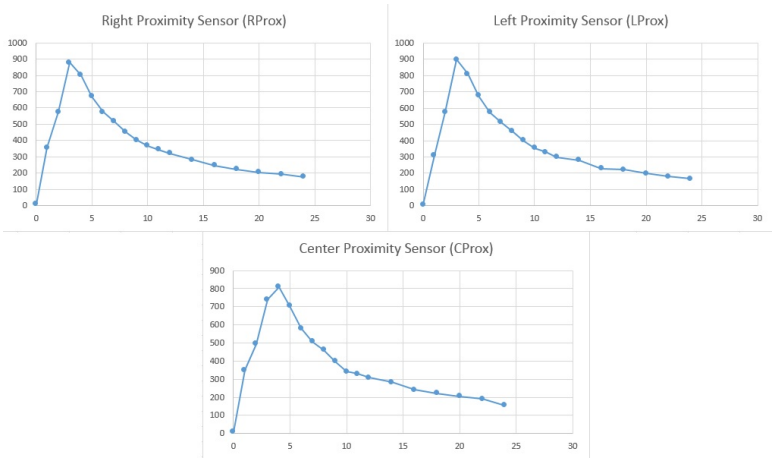


Fig. 11: The graphs show sensor reading vs. distance (in inches) from the sensor. The three sensors have the same generic curve which is expected as they are all Sharp sensors of similar models.

3.6 Photoresistive Light Sensors

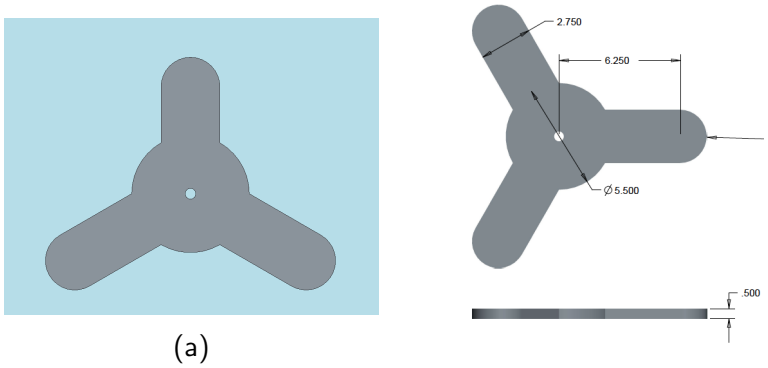
Rocky uses two photoresistors to gather data about light. Photoresistors change in resistance depending on the amount of light exposure. A large amount of light results in a low readout value, and a small amount of light results in a high readout value. The two sensors are used to guide Rocky toward the light on the receptacle at the end of the course. By pivoting, Rocky compares the values of the two sensors and determines whether to turn left or right with the goal of matching the light sensor values. If the left sensor returns a smaller value than the right sensor, then Rocky turns toward the left, as the robot is pointed toward the right. If both sensors are sufficiently bright and within a threshold of each other, then Rocky drives forward toward the light.

3.7 Wheels and Tires

When designing the wheels, getting the robot over the wall was the first and most important objective. In order to provide enough force to drive over the wall, the wheels needed to have a greater radius than the 6" height of the step. We chose 14" diameter rear wheels so they could provide a force great enough to help push the vehicle over the wall. After observing the limited traction on the smooth linoleum floor, therabands were wrapped around the rear wheels to increase traction.

The front wheels are the primary mechanism of climbing over the wall. Our research showed that tri-lobe wheels were good for step climbing. Originally, the tri-lobe wheels had a seven inch lobe radius in order to be uniform with the rear wheels. Looking at the geometry of how the wheels turned as the robot went over each step led us to make one of the lobes shorter than the other two. This would allow the groove between the circular section and lobe to fit snugly against the top of the step. This design is shown in Figure 12a.

After testing this design, it was determined that the lobe length needed to be much longer in order to push our robot over the wall, and that three symmetric lobes would perform better



(a) (b)

Fig. 12: (a) shows the first front wheel design iteration, (b) shows the final design.

than the one shorter lobe. The final tri-lobe wheel design has lobes of 7.75 inch radius. This allows the center of mass of the robot to be far enough forward to tip the robot over the wall. This final tri-lobe wheels design is shown in Figure 12.

In order to ensure that the tri-lobe wheels could support the entire weight of the robot, we decided to use half-inch thick HDPE for the material. This thickness was chosen so that one lobe on each wheel could support Rocky's weight, which would only be necessary immediately after Rocky traverses the wall. From stress analyses in Creo, the tri-lobe wheels could support a static load of $25lbf$ on each wheel with a safety factor of between two and three.

3.8 Arm and Claw

We had to design an arm to retrieve the med kit from the driving surface, safely and securely store the med kit on the robot through the obstacle course, and then deposit the med kit accurately in the light receptacle. We looked at different ways to make an arm and a claw to pick up and drop off the med kit. We decided that for the arm, minimal moving parts and maximum simplicity of design were key for this component.

We considered having an arm with a joint, or having a claw that would open and close on one half of the claw and be static on the other half, but eventually we decided that it would be much simpler and more foolproof if we had only one motor, which would actuate the arm, and a static claw design that would simultaneously pick up the med kit and store it when the arm was moved to an upright position.

We decided on a 1" square aluminum tube and $\frac{1}{8}$" aluminum sheet for the static claw. We used a geared down motor from the scrap supply and an aluminum rod and shaft collars were used to attach the motor to the arm. In order to pin the arm to the collar we made a HDPE plug for the end of the arm and drilled through the arm for the motor shaft. Then perpendicular to that we drilled a hole through the arm, the plastic plug and the motor shaft and put a spring pin in to constrain the unit.

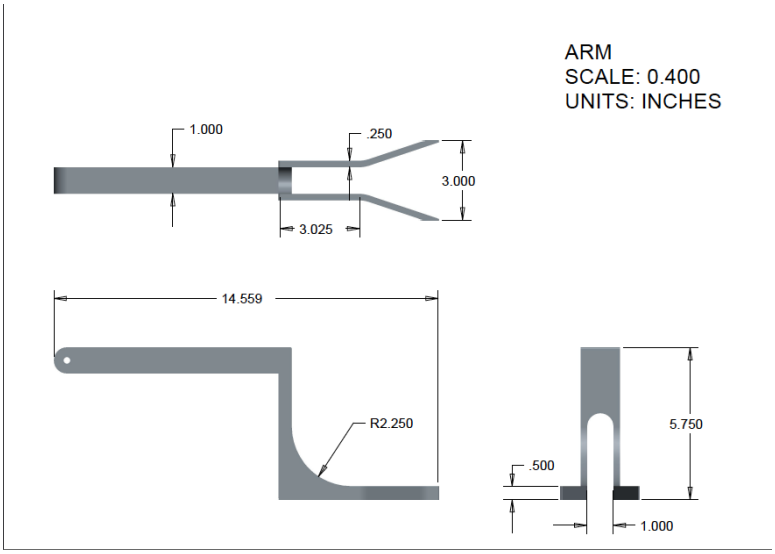


Fig. 13: Creo rendering of arm and claw assembly

There were multiple ways we could have bent the metal sheet to create the basic shape we wanted but in the end we went with cutting two mirrored parts and bolting them on either side of the arm, just bending the bottom of the claw so that it would flare outward for retrieval of the med kit. The final design can be seen in Figure 13. With the robot mostly assembled, we used the med kit and the basket we were supposed to deposit it into in order to choose final geometry. This worked well for us, because the geometry of the arm was dependant on the height and attitude of the chassis, the medkit and the basket, and nothing else, so there wasnt much that could change after we designed the arm.

We chose 12 inches for the arm, with the motor attached a half inch into the bottom of the arm and the claw attached a half inch in from the top. The claw was cut based off of the height of the med kit from the ground and the width of the medkit.

A huge advantage of our design is that during storage the medkit is solidly held in place, because we used eighth inch aluminum sheet and made it about a half inch wide. We were considering adding a cradle on the arm to receive the med kit and hold it in place during wall traversal, but it turns out that the claw and the design already in place holds the medkit securely.

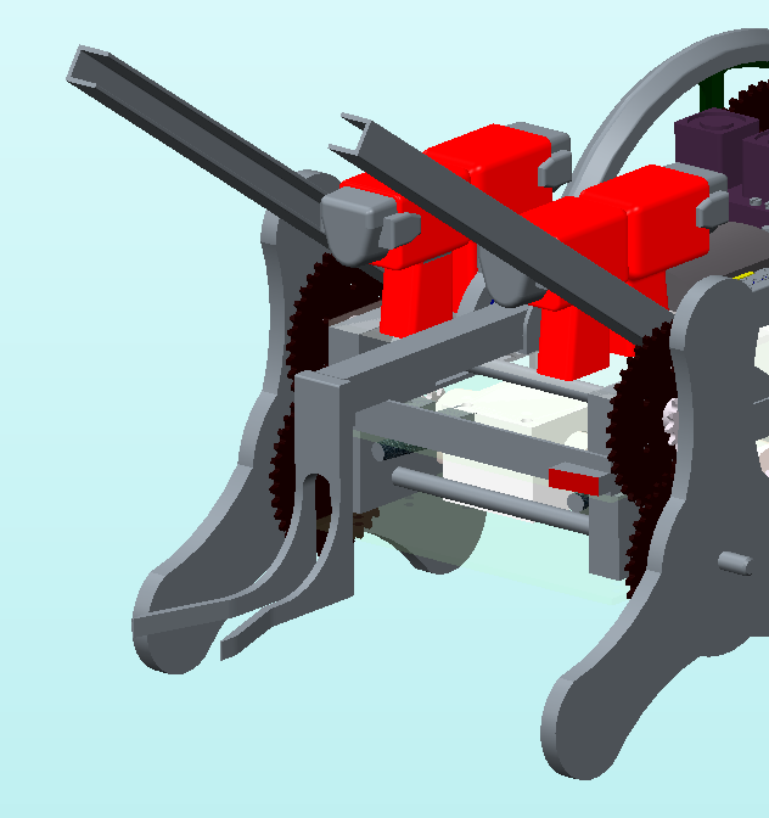


Fig. 14: Schematic of Gripper and Arm

3.9 Static and Dynamic Analysis and Design Loads

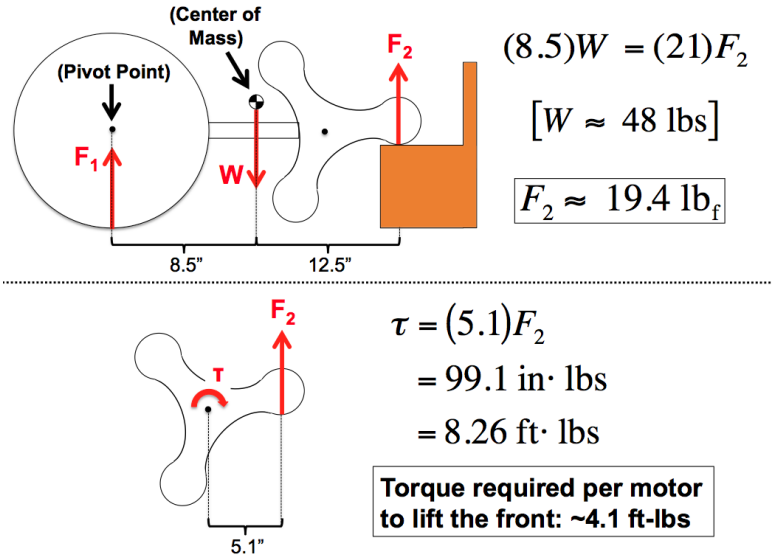


Fig. 15: Free Body Diagram of Lifting Torque [NOTE: not to scale]. Shows how much force will be required from the front motors to lift the robot.

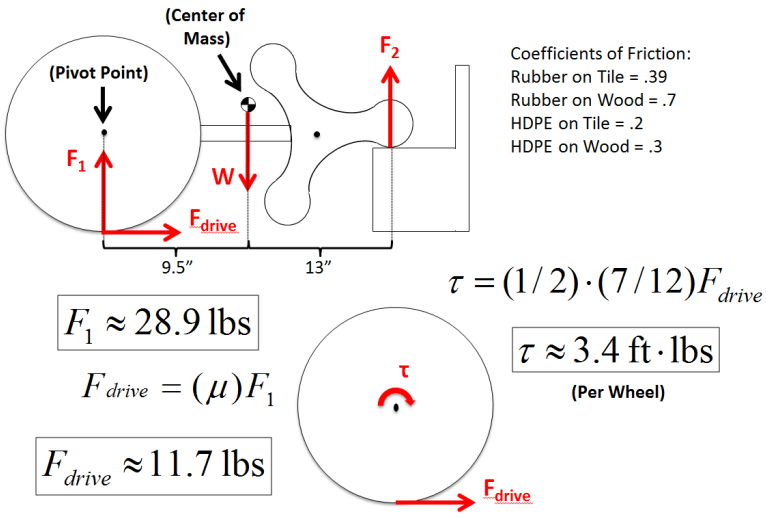
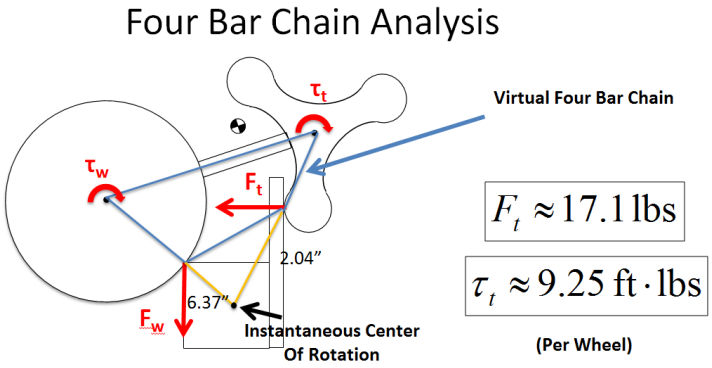


Fig. 16: Free Body Diagram of Drive Torque [NOTE: not to scale]. Shows the drive torque required of the rear motor prior to gearing.



$$\Sigma M = 0 = 2(F_w + 2(1.61\cdot\tau\cdot\mu))(6.37/12) + 2(12/7)(2.04/12)\cdot\tau$$

Fig. 17: Free Body Diagram of Chain Analysis [NOTE: not to scale]. Shows the maximum torque required of the front wheels at the top of the wall.

Each of our four wheels will has an associated drill motor. The front wheels are pinned to the axle, so the axles spins as well. The drill motors we have available to use output approximately 0.8 ft-lb of torque. In order to climb the wall, a torque of 9.25 ft-lb is required at the point of tipping. This requires a gear ratio of 11.56:1. Using two sets of gears in order to achieve this ratio. A small gear will be attached to the drill motor and a gear 5 times larger will be connected to it with a chain. This large gear will sit on an axle with another small gear the same size as the gear connected to the drill motor. This gear will be connected to a gear 4 times the size of the smaller gear via another chain. The second large gear will be connected to the axle of the wheel. In total, this will provide a 20:1 gear ratio,

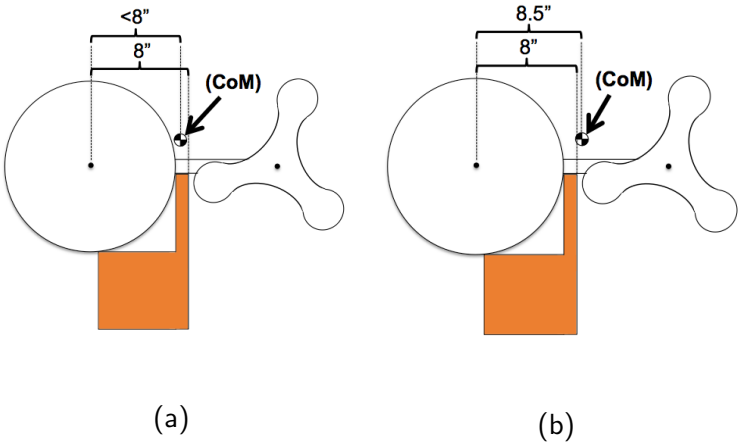


Fig. 18: (a) shows the position of the center of mass with the original configuration of the motor mounts, the robot could not make it over the wall. Moving the motor mounts forward, the center of mass moves beyond the wall as seen in (b), which forces the robot over the wall.

which will supply a torque of 16 ft-lb. This is much greater than the minimum required to traverse the wall, and sufficiently allows for 25% losses from the motor to the wheel.

In order to allow for the necessary torque to climb the wall, we chose chains as opposed to belts to connect our gears. This decision was made based on a torque test that we did in lab. When attaching a metal bar to the wheel of the test robot, the drill motor with a 7.2:1 gear ratio only output about 2.5 ft-lb of torque before the belt began to skip. Chains will keep tension on the gears at all times and ensure that a sufficient torque can be supplied to the wheels. Further chain analysis is shown in the appendix.

3.10 Control Systems

3.10.1 Sensor Placement

The front and rear proximity sensors were placed centrally so as to provide the best average distance between the front or rear of the robot and whatever is sensed. It was initially decided to place the side sensors on the front of the robot so as to detect the position of the front of the robot, which is most relevant for navigating the chute. However, geometric constraints precluded this option. The sensors could not be placed too far forward, or else they would hit the wall during the wall traversal. This constraint prevented sufficient angling to detect the side walls if the sensors were placed at the front. For this reason, both sensors were placed on the sides of the robot, but angled forward so as to better approximate the position of the front of the robot, which would be what is turning during pivots.

The light sensors were both placed at the front of the robot which would be closest to the light during the medical kit delivery. As described below, the difference between the signals from the two sensors is used to determine the position of the light. For this reason

the two sensors have a slight spatial offset and are also offset in angle. They were initially both pointed outward, in order to detect light from both sides, but this caused the robot to lose sight of the light when it was too close to the drop location. Therefore, the two sensors were switched to angle inward, so that the robot can sense the light even when very close. The signals from the two sensors were then corrected for the angle change, such that the signal from the left sensor reads the right light input and vice versa.

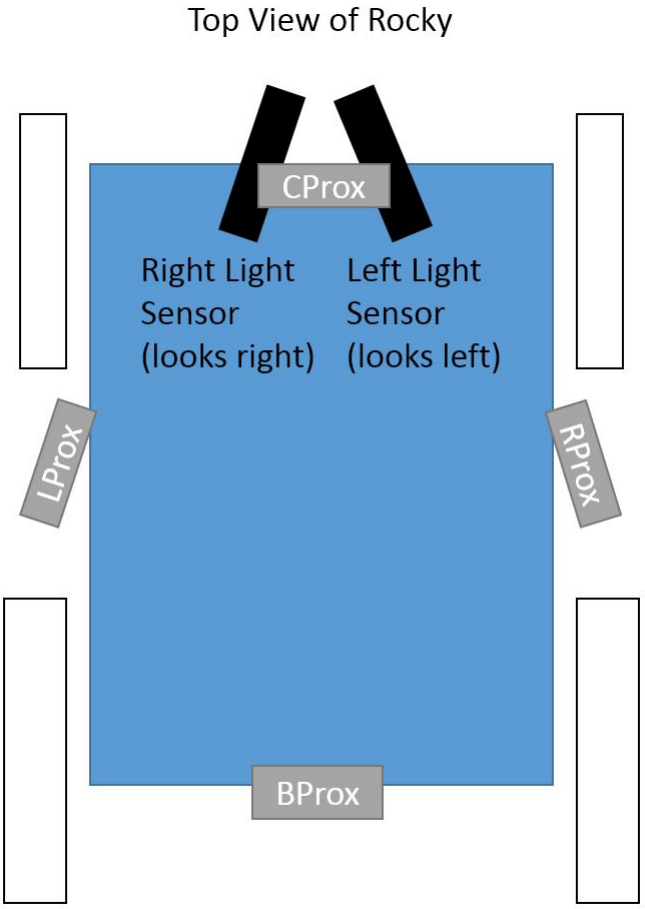
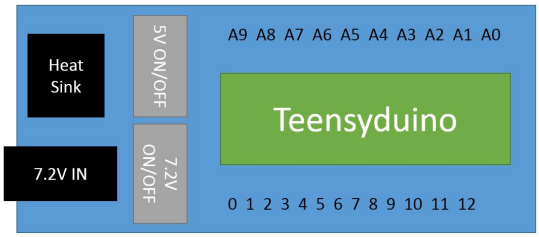


Fig. 19: Diagram of sensor placement. CProx is center proximity sensor, LProx is left proximity sensor, RProx is right proximity sensor, BProx is back proximity sensor.



Analog Sensor Inputs
A0: Right Light Sensor
A1: Left Light Sensor
A2: Left Proximity Sensor (LProx)
A3: Center Proximity Sensor (CProx)
A4: Right Proximity Sensor (RProx)
A5: Back Proximity Sensor (BProx)

Digital I/O
12: Ch6 Receiver
11: Ch5 Receiver
10: Ch4 Receiver
9: Ch3 Receiver (unused)
8: Ch2 Receiver
7: Ch1 Receiver

Digital I/O (cont)
6: unused
5: unused
4: Arm Motor
3: Left Tri-Lobe Motor
2: Right Tri-Lobe Motor
1: Left Drive Motor
0: Right Drive Motor

Fig. 20: Block diagram of Teensyduino inputs and outputs.

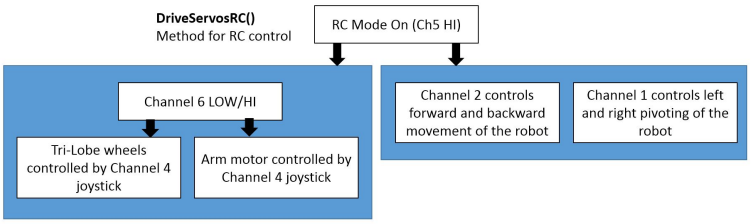


Fig. 21: Outline of control algorithm for open-loop.

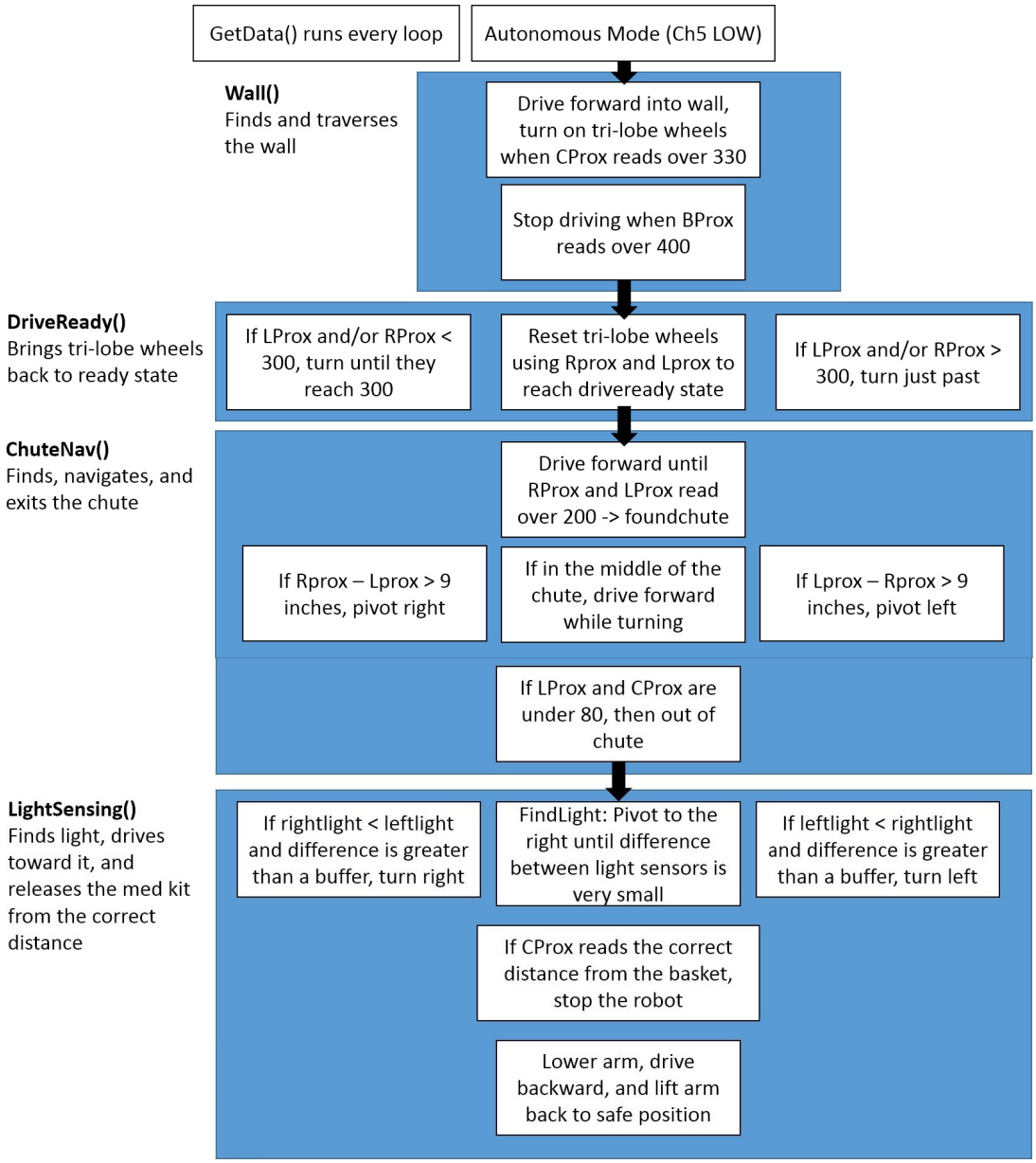


Fig. 22: Outline of control algorithm for closed-loop.

3.10.2 Code Structure Overview and Design Process

The overall structure of our code consists of a series of **else if** statements that call internal functions for the autonomous methods. This facilitates debugging, as each method can be tested separately, and makes the final combination of the autonomous methods simpler. Starting with the final method of finding the light and stopping the robot, the **if** statements work backwards and end with wall traversal. The following code is the bulk of our Autonomous loop, aside from a data-collecting function and a time delay.

```
if (stopbot) {
  Halt();
  HaltTri();
} else if (LightSense) {
  LightSensing();
} else if (driveready) {
  ChuteNav();
} else if (OverWall) {
  DriveReady();
} else if (!OverWall) {
  Wall();
}
```

When writing the code, we began with overall pseduocode of the autonomous process. This ensured that we thought through all of the necessary components and maneuvers Rocky needs to complete. The autonomous code is outlined in the later section on Autonomous Mode.

3.10.3 RC Mode

Rocky uses RC control for the obstacle course portion of the challenge. Channel 2 controls forward and backward motion and Channel 1 controls left and right pivoting. Channel 5 is a switch that triggers either RC mode or the Autonomous loop. Channel 6 is another switch that triggers either RC use of the tri-lobe wheels or the arm, which are both controlled by Channel 4. The arm and tri-lobe wheels never need to be RC controlled simultaneously. Channel 3 has been finicky and giving Rocky issues in RC mode, so we chose to avoid using it.

- **DriveServoRC()**: The DriveServosRC() function utilizes five of the six receiver channels on the FlySkyCT6B remote controller to write microsecond pulses to Rockys 5 motors based on the actuation of the joysticks. The neutral pulse width for this method is 1550 microseconds. There are three modes devoted to RC control, the tri-lobe wheel mode, back wheel mode, and arm mode. The pivot point steering is controlled by subtracting a variable called CalcHold from neutral for the motor that is turning backward, while the forward motor gets the raw pulse value. CalcHold is determined by the absolute value of the difference between the neutral value and the value written to the turning motor.

3.10.4 Autonomous Mode

- **GetData()**: GetData() function is responsible for reading data from four proximity sensors, and 2 light sensors. It is called approximately every 20-40 milliseconds depending on the navigational method that is running. Every data variable is set to zero before calling the analogRead function of its corresponding pin on the RAMB T3. Each data variable comes from the average of a hundred sampled values for improved

consistency, and generates the difference variables for use in their respective navigational modes. A number of print statements are included in this method that write to the serial monitor for debugging purposes.

- **Autonomous Drive Functions**: The autonomous drive functions consist command specific microsecond pulses to the right and left DC motors for the back wheels in order to drive Rocky through autonomous tasks. There is an optional input for specifying the amount of delay time after the command is executed to constrain the distance that Rocky travels. The microsecond pulse lengths were determined by trial and error and are considered specific to our DC motors and drivetrain. This is because the motors did not respond to equal pulse lengths with equal power output. There were two autonomous drive functions that deserve further explanation, namely DriveReady() and DepositLeave(). DriveReady() wrote pulses to the Front Tri-lobe Wheels for the purpose of rotating them to a point where one lobe was perpendicular to the ground, which is the most efficient driving configuration for Rocky. The right prox sensors ability to sense a sharp decrease in distance each time tri-lobe units passed in front of it was utilized to implement a delay in the ForwardTri() drive function such that Rocky consistently stops in the ready position every time. The DepositLeave() function runs series of commands that deposit the medkit in the basket, backs up until the prongs are clear of the medkit, and then lifts of the arm again to prepare to go save another victim.

- **LightSensing()**: The LightSensing() function is responsible for implementing the light following algorithm and calling the DepositLeave() drive method when the center proximity sensor reads a distance smaller than 10 inches away. Rocky checks approximately every 40 milliseconds to see if conditions for darkness or light are met, then proceeds to drive toward the light source using differential steering in response to sensor values from the left and right light sensors. The correction factor for turning Rocky is the difference between the two light sensor readings with the default response being going forward at a speed less than 1 ft/s.

- **ChuteNav()**: The ChuteNav() function is responsible for getting Rocky through an area that has continuous walls on either side. This is mainly because the proximity sensors have a very small field of view and were not mounted with an actuation mechanism for scanning. The minimum distance to trigger chute navigation is 10 inches from the left or right proximity sensors while the distance to trigger light sensing must be greater than approximately 2.5 feet. When inside the chute the robot employs differential steering based on the ratio of the left and right prox sensors to keep Rocky at an equal distance from both walls.

- **DriveReady()**: The DriveReady() function solves the problem we were initially having regarding insufficient traction on the back wheels. The cause was that having two points of contact with the ground reduced the traction on the rear wheels as the tri-lobe wheels were not quite even. Rocky drives much better with only one point of contact from each tri-lobe wheel, and with the touching lobe angled slightly forward. We decided to affix an optical encoder to the front shaft to accurately relay the position

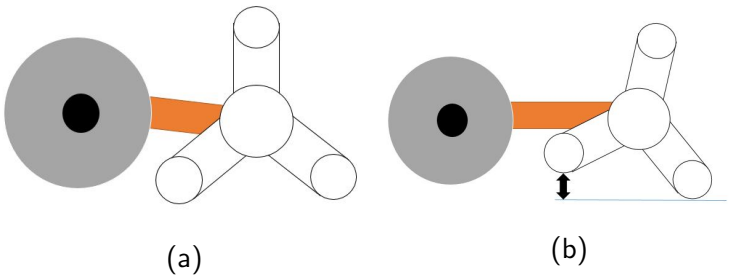


Fig. 23: (a) shows the front wheel orientation of Rocky when it is not ready to drive. (b) shows the orientation when it is ready to drive.

in order for us to position the wheels autonomously for the best drive configuration before and after going over the wall. The Vex Robotics optical encoder, however, was inconsistently sending out data and did not provide for a robust solution. Instead, the placement of the side prox sensors, which were unintentionally attached so that the tri-lobe wheels sometimes blocked them, was utilized to sense the rotation point of the tri-lobe wheels and was able to halt them in the desired position on a consistent position. (Figure 23)

- **WallTraversal()**: The wall traversal code is responsible for finding the wall using the center prox, then driving the Tri-lobe wheels until the back proximity sensor detects an object that is greater than approximately 9 inches away. This segment of the code is relatively simple and utilizes the fact that the Tri-lobe Wheels are mechanically linked allowing us to assume that the robot gets over the wall without changing its orientation with regards to the back sensor.

4 Specifications

4.1 Drive Train Specifications

The chassis dimensions were chosen to accomodate all the electronics required to power the drive train and arm. In addition the length was chosen to allow both sets of wheels to sit without overlap or locking. The width was chosen to have the motor mounts fit on the same axis without overlap.

After deciding on using the drivetrain to get over the wall, a wheel with radius greater than the height of the step was required to feasibly drive the robot over the wall. A 14" diameter back wheel was chosen because it matched the design constraint and was available for free in the shop. The front wheel diameter was chosen to ensure the robot made contact with the step and was long enough to push it past the center of mass.

After geometrical considerations for the position of the robot as it landed after traversing the wall, 22" antennae were added at $\approx 24^o$ to provided extra support on collision with the ground and prevent the robot from tipping over.

The derivation of the other drive train specifications are shown in the appendix.

Parameters	Value
Max Width (in)	20
Max Length (in)	30
Max Height (in)	26
Front Wheel Radius (in)	7.75
Back Wheel Diameter (in)	14
Antenna Length (in)	22
Weight (lb)	41.19
Delivery Time (s)	135
Gear Ratio	20.00
Wheel Torque (ft-lb)	16.00
Max Velocity (ft/s)	1.99
Lifting Torque (ft-lbs)	11.64

Tab. 1: Drive Train Specifications

4.2 Arm Specifications

Length was chosen to accomodate the medkit being fit in the receptacle, but short enough to be below the maximum torque provided by the motor of $40in - lbs$. Figure 1 shows that the passive gripper has to be between 1" and 1.25" wide. The prongs on the gripper were therefore designed to fit the design requirement with clearance on both ends of the requirement.

The stowing angle was chosen to ensure the arm was high enough off the electronics board that it did not interfere with the electronics, but low enough that it did not interfere with traversing the wall.

The pick up angle is for the most part adjustable, but is $\approx 90^o$. The prongs on the arm were designed to be deep enough to allow easy pick up at $\approx 90^o$. The arm was designed to be as light as possible to reduce the overall weight of the robot. The other parameters are derived in the appendix.

Parameters	Value
Length (in)	11
Prong Separation (in)	1.1
Stowing Angle (^{o})	15
Pickup Angle (^{o})	90
Arm Weight (lb)	0.8
Medkit Weight (lb)	1
Motor Torque (in-lb)	40
Arm Torque (in-lb)	11
Drop Time (s)	5
Raise Time (s)	6.5

Tab. 2: Arm Specifications

4.3 Power Specifications

Figure 9 shows the online calculator used to find the power requirements based on the inputs of the DC motors. Assumed motor current is calculated as an average of the current required for the motor controllers and motors.

Parameters	Value
Assumed Motor Current Draw (A)	20
Continuous Battery Draw (W)	865
Min. Required Battery Voltage (V)	12
Idle Motors Battery Life (hrs)	117
Min. Battery Life	5min14s
Typical Battery Life	5min49s

Tab. 3: Power Specifications

4.4 Operational and Navigational Modes

4.4.1 RC/Manual Mode

When navigating the course in manual mode, Rocky will drive with one of the lobes on the tri-lobe wheels in contact with the ground. This allows better weight distribution over the back wheel and consequently provides better torque and increases steering controllability. In addition to the weight distribution, the large gear ratio also contributes to the steering control but is at the expense of a greater top speed. This control should facilitate easier movement through the obstacle course, saving time by preventing unnecessary turning corrections.

Although Rocky has a large chassis, the tri-lobe wheels provide a way of getting over unexpected bumps. Turning the tri-lobe wheels moves the robot forward and over a bump, and the back wheels can drive over easily. This will only be implemented if necessary,

however, as we would prefer to keep the tri-lobe wheels passive until they are activated to climb the wall.

Therefore the medkit will be picked up manually and be driven through the obstacle course in RC mode and the robot will remain in RC mode until reaching the wall, where it is switched to autonomous in line with the course specifications.

4.4.2 Autonomous Mode

Once Rocky reaches the wall, it will be switched into autonomous mode. Using the center proximity sensor, Rockys lobe wheels will begin turning once Rocky is the correct distance from the wall. The wheels will turn at a pre-prescribed speed, and the tri-lobe wheels will rotate until Rocky is on the other side of the wall. The antennae prevent Rocky from flipping over and a proximity sensor at the back of the robot lets it know when it has traversed the wall.

The tri-lobe wheels will then be set into a home position using limit switches to ensure proper sliding. Once Rocky conquers the wall, the two side proximity sensors, mounted in the front of the robot, will ensure that Rocky's flawless navigation through the chute. The robot is rear wheel drive and the sensors are in the front, meaning that Rocky will know where it is at all times. Even though Rocky has a large chassis, the low gearing will allow for controlled turning in the chute.

Finally, Rocky will spin in a circle, use two light sensors to triangulate the light source, and navigate to the source until he is a pre-determined distance away as determined by the proximity sensor. Then, the arm will lower the medical kit into the basket, and Rocky will drive away, leaving the kit in the basket.

5 Testing Methods and Results

5.1 Testing Methods

When developing Rocky, we tested every component thoroughly before proceeding to the next step, but also worked on machining and coding in parallel to increase our productivity. Prototyping was a key component of the testing process, such that most components had been tested in some form or another before their final manufacturing. For example, we made a prototype medical kit retrieval arm such that we could test the operation of the code and the motor while we were still manufacturing the final version of the arm. This prototype was also helpful in determining the final geometry of the arm. In addition, the gear transition components, which connected the intermediate sprockets, were fabricated twice: once as a prototype, and then again, taking into account some of the stability issues that were discovered the first time. A similar process was also used for the electronics guard. Coding the sensors started with separate test methods that interpreted and output the data to the Serial Monitor. Once the sensor input was read in correctly, a method was added to our larger program and was called either by the Autonomous or RC loop for testing in context. We worked in reverse order, developing the light sensing algorithm, then the arm code, followed by the chute, and finally concluding with autonomous wall traversal. Keeping each component isolated was crucial in debugging and developing the code.

Mechanically, we performed multiple drop tests of Rocky, and realized that we should add shock absorbers to the electronics board to prevent the acrylic base from cracking and protect the sensitive components. However, the drop tests did not phase Rocky, and he proved to be very robust. After many successful wall traversal tests, both open-looped and autonomously, Rocky has begun to fatigue and the tri-lobe wheels are looser on the shaft than when they were first manufactured. They now exhibit some roll from side to side, which is problematic when attempting to pivot. We tested Rocky on the vinyl floor of the Civil Engineering Wing two days before Deans Date and noticed that the tri-lobe wheels were getting caught and the back wheels had very poor traction. Cleaning the tri-lobe wheels to decrease friction and adding rubber traction to the back wheels improved Rockys maneuverability greatly.

5.2 Testing Results

Total course traversal time, excluding obstacle course, is approximately 2 minutes.

5.2.1 RC Retrieval of Med Kit

Collection time depends on accuracy of driving up to medical kit. Typically takes approximately 20 seconds. Approximately 7 seconds to lift arm once medical kit has been collected.

5.2.2 Wall Traversal

Total time to breach the wall is approximately 8 seconds. This includes 3 seconds of approach, 4 seconds to climb, and 1 to stop. This has been very consistent throughout approximately 50 trials.

5.2.3 Chute Navigation

Total time to navigate the chute is approximately 35 seconds. This depends somewhat on the level of traction that is achieved by the wheels and thus the turning times.

5.2.4 Light Sensing

Seeking the light varies somewhat with the level of traction possible with the rear wheels. This varies from approximately 30 to 45 seconds. The exact position of the front lobe wheels influences the traction significantly.

5.2.5 Dropping Off Med Kit

Total time to deposit the medical kit is approximately 11 seconds. This includes 5 seconds to lower the kit into the basket, 1 second to reverse, and 5 seconds to lift the arm back into position.

6 Conclusions and Future Work

As explained in the executive summary, our robot design is not intended to perform well in a wide variety of terrain situations. Taking into consideration the time and budgetary constraints for the project, we decided to make a robot that would consistently perform well on the predetermined course rather than one that would be able to handle any given terrain. We debated a number of designs including tank treads, deployable mobile bridges, active claws, and wheels big enough to just drive over the wall with speed. We ended up choosing a design that would be simple enough to perform predictably and consistently, but also could be put together comfortably in the allotted time.

On completion and extensive testing of the robot, we are very pleased with our design. The front wheels lift the robot over the wall very reliably, and their solid, single-piece construction puts them at a very low risk of breaking. Additionally, rotating the front wheels allows us to change the height of the front of the robot as well as the distribution of weight between the front and rear wheels. The pneumatic rear wheels work well apart from the fact that they are prone to occasional skidding, so the ability to transfer weight to the back of the robot allows us to eliminate skidding by increasing friction. Because the coefficient of friction of the rear wheels on linoleum is lower than we anticipated, we stretched a sheet of high friction latex across the surface of the wheels to increase friction. This results in more consistent turns and generally better performance. With more time we would choose a more permanent solution to this issue - perhaps wheels that are made of a material that has a significantly higher coefficient of friction on linoleum.

In total we used six sensors: one proximity sensor for each side of the robot and two light sensors for the front. This arrangement of sensors enabled our robot to perform consistently on the predetermined obstacle course, but the arrangement and type of sensors could certainly limit the performance of the robot in other circumstances. The robot has blind spots on all four corners, so it would not be well suited for sensing small obstacles that are in any of these locations. Additionally, because the light sensors on the front have a narrow range of vision, they would not be well suited for guiding the robot on uneven terrain. A future iteration of our robot could include sensors that are better suited for a wide range of conditions such as cameras, GPS, and laser scanners. Furthermore, our robot could be made lighter and more robust.

The arm was designed to be as simple and solid as possible, and we are comfortable that our arm will perform without any problems. Because the design is simple and involves minimal moving parts, if the med kit retrieval or deposit fails it will almost certainly be due to a failure of the code or sensors rather than the arm itself.

We are comfortable that our robot will be able to navigate the course with around a 90 percent chance of complete success on the first run. Our uncertainty is mostly due to the frictional issues that we realised only when testing on the linoleum floor after the robot was fully constructed.

References

[1] *Stair Climbing Robot : Engineering.* Extreme Projects, 14 Mar. 2012. Web. 27 Mar.

2014.
http : //www.engineering.xtremeprojects.net/2012/01/*mechanical − engineering − project − stair − climbing − robot.html*

[2] *"Urbie" Urban Robot during Vision-guided, Autonomous Stair Climbing.* Robotics. NASA and JPL, n.d. Web. 27 Mar. 2014.
https : //www − robotics.jpl.nasa.gov/systems/urbie$_i$*mage.html*

[3] *Princeton Robot Race - Final Project from MAE 322 (Mechanical Design), Spring 2013.* YouTube. YouTube, 20 Dec. 2013. Web. 27 Mar. 2014.
*http : //www.youtube.com/watch?v = AHvUBQvBC*0*o*

[4] *Measuring the Performance and Intelligence of Systems: Proceeding of the 2000 PerMIS Workshop.* August 14-16, 2000. Edited by A.M. Meystel and E.R.Messina
*http : //books.google.com/books?hl = en&lr = &id = yAhRAAAAMAAJ&oi = fnd&pg = PA*253*&dq = search + and + rescue + robot&ots = 3bX*$_I$*0f*1*iV*&sig =*$_Q$ *tk*5*hj*5*mtAWSNG*7*FScJaeIKRac#v = onepage&q = search*

[5] T. Kamegawaa, T. Yamasaki, H. Igarashi and F. Matsuno *Development of The Snake-like Rescue Robot "KOHGA"* ICRA (April 2004) 5081-5086.
http : //ieeexplore.ieee.org/stamp/stamp.jsp?tp = &arnumber = 1302523*&tag =* 1

[6] *Springer Handbook of Robotics* Springer 2008. Edited by Bruno Siciliano, Oussama Khatib. 1158 - 1163
http : //searchit.princeton.edu/primo$_l$*ibrary/libweb/action/dlDisplay.do?docId = PRN*$_V$*OYAGER*5558993*&vid = PRINCETON&institution = PRN*

[7] Duncan Graham-Rowe *Amoebalike Robots for Search and Rescue* MIT Technology Review, March 29, 2007.
*http : //www.technologyreview.com/news/*407603/*amoebalike−robots−for−search−and − rescue/*

[8] *Robot Mesh .*
http : //www.robotmesh.com/rover − 5 − tank − kit − powered − by − arduino − and − bluetooth?gclid = COyexMHQ$_L$*0CFaVQOgod*9*hAAog*

[9] *Stair climbing robot - mechanical engineering project* YouTube. YouTube, 24 May. 2012. Web. 7 May. 2014.
*https : //www.youtube.com/watch?v = cl*27*YZ*1*JYjI*

A Milestones

Milestone	Expected Date	Achieved Date
Demostrate Drive Train (OL)	3/28/2014	3/28/2014
Autonomous Navigation to Light Source (CL)	4/4/2014	4/4/2014
Traverse Wall (OL)	4/11/2014	4/16/2014
Object Retrieval and Placement (OL)	4/25/2014	4/25/2014
Object Retrieval and Placement (CL)	5/2/2014	5/2/2014
Final Competition	5/13/2014	5/13/2014

Tab. 4: Milestones

B Budget

Item	Quantity	Individual Price	Total
Rigid HDPE Polyethelyne, 1/2" Thick, 24" X 36"	1	$58.95	$58.95
Rigid HDPE Polyethelyne Rectangular Bar, 1" X 1" X 1'	1	$2.94	$2.94
Optically Clear Cast Acrylic Sheet, 3/16" X 12" X 24"	3	$20.14	$60.42
Multipurpose 6061 Aluminum Sheet, .125" X 12" X 12"	1	$28.34	$28.34
Black-Oxide Steel Set Screw Rigid Shaft Coupling, 5/16	1	$7.45	$7.45
Set Screw Shaft Collar, 5/16"	1	$0.98	$0.98
Multipurpose 6061 Aluminum Rod, 5/16" D X 6'	1	$9.08	$9.08
Multipurpose 6061 Aluminum Rectangular Tube, 1/16'	1	$23.38	$23.38
Rubber Bands	1	$1.78	$1.78
Abrasion Resistant Reinforced Rubber Washers	1	$17.95	$17.95
			$211.27

Tab. 5: Budget

C Timecard

Week	Annie	Alex	Victor	David H	David B	Tumi	Eric	Total for Week	Goal/Estimated
Up to 3/14, including spring break and classes up until now	48	50	30	38.5	30	37	45	278.5	500
Week 7 (ends 3/28)									
Midterm Report	4	4	1	1	1.5	6	4	21.5	
Midterm PowerPoint	2	2	1	5	3	1	1	15	
Miscellaneous (including class/lab)	9	8	6	8	6	6	6	49	
Deliverable: Midterm Reports (sum)	15	14	8	14	10.5	13	11	85.5	80
Week 8 (ends 4/4)									
Order Preliminary Materials	1.5	2	2	1.5	0	0.5	0	7.5	
Build Chassis	0	1	2	0	0	6	0	9	
Autonomous Light Navigation	10	2	1	10.5	2.5	0.5	3	29.5	
Miscellaneous (including class/lab)	2	6	5	0	9	6	8	36	
Deliverables: Light following, materials list, chassis	13.5	11	10	12	11.5	13	11	82	80
Week 9 (ends 4/11)									
Drive Train + Drive Code	8	14	8	2		8	6.5	46.5	
Robustness testing			1	0				1	
Finalize weighting			0	0				0	
Fabricate wheels	3		0	0			4	7	
Design Specifics of Arm			5	0				5	
Miscellaneous (including class/lab)	6	5	4	9	11	6	8	49	
Deliverable: Wall traversal	17	19	18	11	11	14	18.5	108.5	100
Week 10 (ends 4/18)									
Build arm			6	0	6			12	
Drive Train + Drive Code	3	12	1	7		4	2	29	
Sensor placement			0	0				0	
Autonomous wall climb			0	0				0	
Miscellaneous (including class/lab)	10	4	5	3	5	10	11	48	
Deliverable: Finalized Creo	13	16	12	10	11	14	13	89	80
Week 11 (ends 4/25)									
Arm Code/Motor etc	5	8	1			3		17	
Test/tweak drive code		1	0	2				3	
Proximity sensor code	2.5		0	4				6.5	
Miscellaneous (including class/lab)	1.5	3	6	6	7	8	7	38.5	
Deliverable: OL arm demo	9	12	7	12	7	11	7	65	80
Week 12 (ends 5/2)									
Autonomous Arm Code	3.5	1	0	1				5.5	
Autonomous Drive Code (chute, wall, etc)	3.5	3	0	6				12.5	
Miscellaneous (including class/lab)	3	5	6	5	8	9	6	42	
Deliverable: CL arm demo	10	9	6	12	8	9	6	60	80
Week 13 (ends 5/9)									
Obstacle course final testing	6	1	1	2			6.5	16.5	
Optimize code	8	3	0	8	2	8	3	32	
Report draft			3	5	2			10	
Miscellaneous		8	6	2	11			27	
Deliverable: Final code + report draft	14	12	10	17	15	8	9.5	85.5	100
Dean's Date (5/13)									
Final Report	2.5	2	10	8	9.5	11	1	44	
Final Robot Testing	13	13	3	3	12	5	8.5	57.5	
Final PowerPoint	2		0	0				2	
Final Week Total	17.5	15	13	11	21.5	16	9.5	103.5	80
Project Total	157	158	114	137.5	126.5	135	130.5	958.5	1180

Tab. 6: Timecard with total time of 958.5hrs

D Material List

#	Mechanical Component	Material	Stock Dimensions	Part Dimensions	Source	Manufacturing	Amount	Value
1	Tri-Lobe Front Wheel	Rigid HDPE	1/2" Thick; 24" x 24"	See Drawing	MC#8619k472	CNC, Mill, Bandsaw	2	$46.06
2	Front Wheel Traction	Super-Grip Non Skid Coating	11-oz Aerosol	--	Plastidip International	Spray Can	1	$1.00
3	Back Wheels	Bicycle Tires	14" Diameter	16" Diameter	Morgan / Skyway	Lube	2	$40.00
4	Back Wheel Hub	HDPE	4" Diameter, 1" Length	3" Diameter, 0,75" thick	MC#8624k611	Lathe	2	$14.38
5	Wheel Sprocket Hub	HDPE	4" Diameter, 1" Length	3" Diameter, 0,75" thick	MC#8624k611	Lathe	2	Repeat
6	Back Wheel Bearing	Graphite SAE 841 Bronze Flange Bearing	1/2" Shaft Diameter, 3/4" OD, 1" Length	No Change	MC#9440t25	Lathe	4	$3.86
7	Top Axle	Aluminum 6061 Rod	1/2" Diameter; 6' Length	11.125" long	MC-#8974k28	Lathe	2	$14.50
8	Bottom Axle	Aluminum 6061 Rod, Steel Rod	1/2" Diameter; 6' Length	15.875" long	MC-#8974k28	Lathe	2	$14.50
9	Axle Mount	HDPE	1" Thick, 3" Width	2.25" x 1" x 3"	MC#8671k63	Mill	8	$7.05
10	Front Chain 1	ANSI Roller Chain #35	10'	19.5", 52 Links	MC-#6261k172	Chain Break	2	$39.00
11	Back Chain 1	ANSI Roller Chain #35	10'	27.0"	MC-#6261k172	Chain Break	2	Repeat
12	Chain 2	ANSI Roller Chain #35	10'	17.25"	MC-#6261k172	Chain Break	4	Repeat
13	Bottom Skid Support	Aluminum	90 Degree Angle, 1/16" Thick, 1/2" x 1/2" Legs	19" Lg. 1/3" Wd.	MC#8982k54	Bandsaw	2	$2.29
14	Sprocket_2	Steel	#35 Chain, 3/8" Pitch, 45 Teeth, 19/32" min Bore	--	MC#2299k319	Half-Links	4	$24.42
15	Sprocket_2Trans	Steel	#35 Chain, 3/8" Pitch, 45 Teeth, 19/32" min Bore	--	MC#2299k319	Half-Links	4	$24.42
16	Sprocket_2Wheel	Steel	#35 Chain, 3/8" Pitch, 45 Teeth, 19/32" min Bore	--	MC#2299k319	Half-Links	2	$24.42
17	Sprocket_1	Nylon Machinable-Bore Sprocket	#35 Chain, 3/8" Pitch, 9 Teeth, 1/4" min Bore	--	MC#60425k173	Half-Links	4	$11.75
18	Sprocket_1Trans	Nylon Machinable-Bore Sprocket	#35 Chain, 3/8" Pitch, 9 Teeth, 1/4" min Bore	--	MC#60425k173	Half-Links	4	$11.75
19	Chassis Frame	Aluminum 6061 L-Beam	1/16" Thick , 8 ft long	Bend to 19.25" x 10.1875"	MC-#8982k39	Bender	1	$7.99
20	Base Plate	Optically Clear Cast Acrylic Sheet	3/16" Thick, 12" x 24"	20" x 10"	MC-#8560k214	Laser Cutter	1	$20.14
21	Computer Board	Optically Clear Cast Acrylic Sheet	3/16" Thick, 12" x 24"	11.125" x 9.5"	MC-#8560k214	Laser Cutter	1	Repeat
22	Motor Mount	HDPE	1-1/2" Thick, 4" Width, 1 ft long	3.5" x 3.5" x 1.25"	MC#8671k77	CNC, Mill, Bandsaw	4	$18.28
23	Antenna U-beam	Highly Corrosion-Resistant 6063 Al. U-Channel	1/8" Thick,1" Base, 1" Leg, 4 ft long	21.125" long	MC-#9001k52	Bandsaw	2	$16.26
24	Lateral Support	Multipurpose 6061 Aluminum T-Beam	1/2" x 1/2" ; 2' Length	19" long	McMaster	Bandsaw	4	$2.80
25	Battery Holder	Plastic	Custom Made by Black & Decker	--	B&D Model: LDX112PK	Hacksaw	4	Free
26	Arm Hinge	Multipurpose 6061 Aluminum Bar	1/2" x 1/2"; 2' Length	--	MC-#9008K81	Mill, Bandsaw	1	Repeat
27	Arm Lever	Multipurpose 6061 Aluminum Bar	1/2" x 1/2"; 2' Length	12.5" long	MC-#9008K81	Bandsaw	1	$2.33
28	Arm Fork	Multipurpose 6061 Aluminum Bar	1" x 1", 1' Length	--	MC-#9008k14	Bender, Bandsaw	1	$6.91
29	Socket Screws	Nylon	8-32 Thread, 1" Length, 100-pack	--	MC#95868a349	Screwdriver	1	$5.88
30	Plastic Screws	Nylon	5/16" Diameter, 5" length	--	--	Bandsaw	8	Repeat
31	Socket Head Cap Screw	Nylon	6-32 thread, 1" Length	--	--	Screwdriver	22	$1.76
32	Shock-Mount Rubber Tubing	Silicon Foam	Tube, 3/4" OD, 1/4" ID, 36" Long, 75A Durometer	1/16" Thick	MC#8637k111	Scissors	1	$2.61
	Total							**$364.36**

Tab. 7: Pricing of Mechanical Parts

	Electrical Component	Mounting	Info	Source	Amount	Value (each)
1	Victor 884	#6-32 threaded studs	http://team358.org/files/electrical/V883UsersManual.pdf http://team358.org/files/electrical/V883InstallationInfo.pdf http://team358.org/files/electrical/Victor883FETirl2203n.pdf	Innovation First	3	$89.99
2	Victor 888	#6-32 threaded studs	http://www.vexrobotics.com/217-2769.html	Vex Pro	2	$69.99
3	Bussmann ATC/ATO 6 Position Fuse Panel	#10-32 threaded studs	http://www.ebay.com/itm/Bussmann-ATC-ATO-6-Position-Fu	Cooper Bussmann	1	$13.95
4	On/Off Toggle Switch	#6-32 threaded studs		Home Depot	1	$10.00
5	Robodyssey RAMB T3 Motherboard w/ Teensy 3.0	#8-32 threaded screws	http://www.robodyssey.com/ramb-t3-for-the-teensy-3-1/	Robodyssey Systems LLC	1	$49.95
6	4-position Dual-Row Terminal Strip	#6-32 threaded studs	http://www.vetco.net/catalog/product_info.php?products_id=6	All Electronic Corp	2	$1.35
7	Fly Sky CT6B OEM Version Exceed RC 6-Ch 2.4GHz Radio Transmitter w/ Receiver	External to Robot	http://www.xheli.com/exceedrc-24ghz.html	Robodyssey Systems LLC	1	$50.00
8	5 Pack LittleFuse Electric Fuses Blade Type 20 Amp	In Fuse Panel	https://www.carltonbates.com/static/catalog/products/images	Littlefuse	1	$7.30
9	HC613LM Johnson PMDC M	Placed in Motor Mount	http://www.johnsonmotor.com/en/products/dc-motors/datashe http://www.johnsonmotor.com/en/resources-for-engineers/mo	Johnson Electric Motors	4	$30.80
10	Molon CHM PMDC GearMotor	Zip Ties	http://www.molon.com/standard_dc_motors.html http://www.molon.com/pdf/standard-dc_CHM.pdf	Molon Motor and Oil Corporation Distributer: McMaster MC#6409k13	1	$53.16
11	Black & Decker LB12 12V Lithium Ion Battery	Velcro, Tape	http://www.akku.net/Akku-Info/1.20.BUD.999.89.Akku-f%C3	B&D Model: LDX112PK	4	$48.00
12	Rechargeable NiMh 1,500 mAh AA Cells	In Battery Pack	----------	Robodyssey Systems LLC	6	$2.10
13	AA 6 Cell Battery Pack (without batteries)	Velcro	----------	Robodyssey Systems LLC	1	$4.99
14	Sharp Analog Distance Sens	Affix to Metal Plate	http://www.pololu.com/product/136	Pololu	4	$8.95
	Photocell w/pot	Affix to Metal Plate	http://oande.en.alibaba.com/product/595653509-213752728/	China	2	$1.50
15	Vex Shaft Encoder	??	http://www.vexrobotics.com/276-2156.html	Vex Robotics Inc.	1	$19.99
16	Wires (Female Terminals) 10 pk assort.3" w/ shrink tube	Tape	----------	Robodyssey Systems LLC	1	$2.24
17	PWM Cable	Tape	----------	Made in Shop	11	$1.00
18	24 inch Servo Extension Cable	Tape	----------	Robodyssey Systems LLC	1	$3.49
	Total					**$468.75**

Tab. 8: Pricing of Electrical Parts

E Chain Pitch

Parameter	Symbol	Values	Type
Chain Pitch (in)	p	0.375	Ind
Number of Sprocket1 Teeth	N1	9	Ind
Numer of Sprocket2 Teeth	N2	45	Ind
Sprocket Speed (rpm)	n	650	Ind
Center to Center Distance FCh3 (in)	C	5.188	Ind
Center to Center Distance FCh2 (in)	C	4	Ind
Pitch Angle (deg)	gamma	40	Dep
Sprocket Pitch Diameter (in)	D	0.411	Dep
Diameter (in)	d	0.17	Dep
Chain Velocity (ft/min)	V	182.81	Dep
Max Exit Velocity (ft/min)	v_max	69.90	Dep
Min Exit Velocity (ft/min)	v_min	28.52	Dep
Chordal Speed Variation	deltaV/V	0.23	Dep
Link Plate Limited Nominal Power (hp)	H1	0.08	Dep
Roller Limited Nominal Power (hp)	H2	21.56	Dep
Chain Length (in)	L3	21.39	Dep
Chain Length (in)	L2	19.28	Dep
Tooth Correction Factor Pre-extreme hp	K1	0.50	Dep
Tooth Correction Factor Post-extreme hp	K1	0.39	Ind
Strand Correction Factor	K2	1	Ind
Allowable Hosrepower	H_a	0.845	Dep
Required Horsepower	H_d		Dep

Tab. 9: Chain Pitch

F IR Sensor Data

Distance (in)	Right Prox	Distance (in)	Center Prox	Distance (in)	Left Prox
0	9	0	8	0	8
1	355	1	351	1	310
2	576	2	495	2	575
3	878	3	740	3	898
4	801	4	810	4	809
5	670	5	705	5	677
6	576	6	581	6	573
7	518	7	507	7	513
8	453	8	461	8	458
9	403	9	398	9	402
10	367	10	341	10	354
11	344	11	330	11	329
12	321	12	309	12	300
14	282	14	282	14	280
16	247	16	242	16	229
18	223	18	222	18	222
20	204	20	206	20	199

Tab. 10: IR Sensor Data

G Derivation Specifications

G.1 Drive Train Specifications

G.1.1 Delivery Time

From 4 we see that the minimal amount of distance covered by the robot is 80ft, 45ft under RC and 35 autonomous. For the autonomous portion of the course the robot will be moving at $\approx 0.5ft/s$ and under RC at $\approx 1ft/s$, the time taken to traverse the wall is approximated to 20s from trials.

$$T_{min} = \frac{45}{1} + \frac{35}{0.5} + 20$$
$$T_{min} = 135s = 2min15s$$

G.1.2 Gear Ratio

From 17 we can see that the maximum torque required for the rear wheel is approximately $9.25ft - lbs$. The drill motor has a specified torque of $0.8ft - lbs$ as tested in the shop. The

torque ratio is used to obtain the gear ratio:

$$R = \frac{T_{wheel}}{T_{motor}}$$
$$R = \frac{9.25}{0.8}$$
$$R = 11.6$$

A minimum gear ratio of 12:1 is required to get the minimum amout of torque required on the rear wheel. There was a surplus of 5:1 and 4:1 gears in the shop, so a 20:1 gear ratio was decided on, as it fit the necessary design constraint and reduced the total cost of the robot.

G.1.3 Wheel Torque

Given the gear ratio of 20:1, and drill motor torque of $0.8ft - lbs$, the wheel torque is calculated as:

$$T_{wheel} = R \times T_{motor}$$
$$T_{wheel} = 20 \times 0.8$$
$$T_{wheel} = 16ft - lbs$$

G.1.4 Maximum Velocity

Maximum velocity is calculated taking the revolutions per minute of the drill motor into account. The drill motors are specified at 650 rpm. The 20:1 gear ratio, put the wheels at $\frac{650}{20} = 32.5rpm$. The rear wheels have a diameter of 14".

$$Circumfrenceof Wheel = \pi \times D = 43.98''$$
$$V_{max} = \frac{32.5 \times 43.98}{60 \times 12}$$
$$V_{max} = 1.99ft/s$$

G.1.5 Lifting Torque

The lift torque is that which is required by the front tri-lobe wheels to lift Rocky up on the first step. From 15, we obtain the lifting force required to be $42.2lbs$

$$T_{lift} = \frac{F_{lift} \times radius}{12}$$
$$T_{lift} = 27.3ft - lbs$$

G.1.6 Pull Torque

The pull torque is that required by the front tri-lobe wheels to pull Rocky over the second step after it has engaged the step (17), we know the pull force to be $34.2lbs$

$$T_{pull} = \frac{F_{pull} \times radius}{12}$$
$$T_{pull} = 22.1ft - lbs$$

G.2 Derivation of Arm Parameters

G.2.1 Arm Torque

The arm torque is given by:

$$T_{arm} = W_{medkit} \times L_{arm}$$
$$T_{arm} = 11in - lbs$$

G.2.2 Drop Time

The time taken for the arm to drop the medkit is 5s, which is prescribed in the code.

G.2.3 Raise Time

The arm motor operates at 4rpm, therefore the drop time is calculated by:

$$t = \frac{\text{Revolution of Arm}}{4}$$
$$t = \frac{\frac{165}{360}}{4}$$
$$t = 0.1145min = 6.5s$$

H Code

```
#include <Servo.h> //Import the Servo Library

//Default Optional Inputs for Drive Methods
void BackUp(int delayLength = 0);
void Forward(int delayLength = 0);
void ForwardTri(int delayLength = 0);
void Halt(int delayLength = 0);
void HaltTri(int delayLength = 0);
void Right(int delayLength = 0);
void Left(int delayLength = 0);
void PivotRight(int delayLength = 0);
void PivotLeft(int delayLength = 0);
void Dropoff(int delayLength = 0);
void LiftUp(int delayLength = 0);

// Initialize Receiver Variables
int Ch1;  // Right Stick Y-axis
int Ch2;  // Right Stick X-axis
int Ch4;  // Left Stick Y-axis
int Ch5;  // Auto/RC
```

```
int Ch6;  // Arm/Tri-lobe wheel
int CalcHold;  //Variable to hold calculations for steering corrections

//Sensor Pins
const int leftlightpin = A0;  //Left Light Sensor
const int rightlightpin = A1; //Right Light Sensor
const int RProxpin = A2;      //Right Proximity Sensor
const int CProxpin = A3;      //Center Proximity Sensor
const int LProxpin = A4;      //Left Proximity Sensor
const int BProxpin = A5;      //Back Proximity Sensor

// Sensor Values
int leftlight;    // Left Light Sensor
int rightlight;   // Right Light Sensor
int LProx;   //Left Proximity Sensor
int CProx;   //Center Proximity Sensor
int RProx;   //Right Proximity Sensor
int BProx;   //Back Proximity Sensor

//Sensor Differentials
int lightdiff;    // difference btwn light sensors
int proxdiff;     // difference between proximity sensors

//Light Sensor Thresholds
int lightbuffer = 12;            // difference threshold for correction
int darkthreshold = 25;          // threshold for FindLight()

// Proximity Sensor Thresholds
int wallthreshold = 350;  // Threshold for avoiding walls
int dropthreshold = 350;  // Threshold for dropping medkit into box
int enterthreshold = 200; // Threshold for finding the chute
int exitthreshold = 80;   // Threshold for exiting the chute

// Initialize Decision Making Variables
boolean lowLight;
boolean FoundWall;
boolean OverWall;
boolean LightSense;
boolean WallTraverse;
boolean stopbot;
boolean driveready;
boolean foundchute;

// Create Servo Objects as defined in the Servo.h files
Servo R_DCMotor;     // Servo DC Motor Back Wheel
```

```
Servo L_DCMotor;      // Servo DC Motor Back Wheel
Servo R_DCMotor_tri; // Servo DC Motor Front Wheel
Servo L_DCMotor_tri; // Servo DC Motor Front Wheel
Servo Arm_Motor;      // Servo DC Motor Arm

////////////////////////////////SETUP/////////////////////////////////////
void setup() {
  // Set the pins with transmitter connections as Inputs to Receiver
  pinMode(12, INPUT); //To Chan1 of the Receiver
  pinMode(11, INPUT); //To Chan2 of the Receiver
  pinMode(9, INPUT);  //To Chan4 of the Receiver
  pinMode(8, INPUT);  //To Chan5 of the Receiver
  pinMode(7, INPUT);  //To Chan6 of the Receiver

  // Attach Speed Controller (servo) to the board
  R_DCMotor.attach(0);      //Pin 0
  L_DCMotor.attach(1);      //Pin 1
  R_DCMotor_tri.attach(2); //Pin 2
  L_DCMotor_tri.attach(3); //Pin 3
  Arm_Motor.attach(4);      //Pin 4
  Serial.begin(9600);  //Serial-to-USB at band rate of 9600
}

////////////////////////////////RC METHOD/////////////////////////////////////
void DriveServosRC() // Method for Remote Control Driving
{
  //Tri-Wheel Drive Code
  if (Ch6 <= 1600){ //Conrol Tri-Wheels with Ch4 if Ch6 is low
    Arm_Motor.writeMicroseconds(1515); //stop arm motor
    if (Ch4 < 1550 && Ch4 > 1450) { //stop lobe wheels
      HaltTri();
    }
    if (Ch4 > 1550) { //turn lobe wheels forward
      R_DCMotor_tri.writeMicroseconds(Ch4);
      L_DCMotor_tri.writeMicroseconds(Ch4);
    }
    if (Ch4 < 1450) { //turn lobe wheels backward
      R_DCMotor_tri.writeMicroseconds(Ch4);
      L_DCMotor_tri.writeMicroseconds(Ch4);
    }
  }
  //Drive Back Wheels
  if (Ch1 < 1550 && Ch1 > 1450) {//go forward or backward
    R_DCMotor.writeMicroseconds(Ch2);
    L_DCMotor.writeMicroseconds(Ch2);
```

```
  }
  if (Ch1 > 1550){ //turn left
    L_DCMotor.writeMicroseconds(1550-CalcHold);
    CalcHold = abs(1550-Ch1);
    R_DCMotor.writeMicroseconds(Ch1);
  }
  if (Ch1 < 1450){ //turn right
    L_DCMotor.writeMicroseconds(1550+CalcHold);
    CalcHold = abs(Ch1-1550);
    R_DCMotor.writeMicroseconds(Ch1);
  }
  //Drive Arm
  if (Ch6 > 1600){ //Control arm motor with Ch4 if Ch6 is high
    HaltTri();
    if (Ch4 < 1550 && Ch4 > 1450) { //stop arm motor
      Arm_Motor.writeMicroseconds(1515);
    }
    if (Ch4 > 1550) { //move arm up
      Arm_Motor.writeMicroseconds(Ch4);
    }
    if (Ch4 < 1450) { //move arm down
      Arm_Motor.writeMicroseconds(Ch4);
    }
  }
}

//////////////////////////AUTONOMOUS METHODS BEGIN HERE//////////////////////

void Forward(int delayLength){ //Drive Back Wheels Forward
  R_DCMotor.writeMicroseconds(1610);
  L_DCMotor.writeMicroseconds(1600);
  delay(delayLength);
}

void BackUp(int delayLength){ //Drive Back Wheels Backward
  R_DCMotor.writeMicroseconds(1400);
  L_DCMotor.writeMicroseconds(1400);
  delay(delayLength);
}

void ForwardTri(int delayLength){ //Drive Tri-Wheels Forward
  R_DCMotor_tri.writeMicroseconds(1650);
  L_DCMotor_tri.writeMicroseconds(1650);
  delay(delayLength);
}
```

```
void BackwardTri(int delayLength){ //Drive Tri-Wheels Backward
  R_DCMotor_tri.writeMicroseconds(1350);
  L_DCMotor_tri.writeMicroseconds(1350);
  delay(delayLength);
}

void Halt(int delayLength){ //Stop Back Wheels
  R_DCMotor.writeMicroseconds(1515);
  L_DCMotor.writeMicroseconds(1515);
  delay(delayLength);
}

void HaltTri(int delayLength){ // Halt Tri-Wheels
  R_DCMotor_tri.writeMicroseconds(1515);
  L_DCMotor_tri.writeMicroseconds(1515);
  delay(delayLength);
}

void Right(int delayLength){ //Turn Robot Right while going Forward
  R_DCMotor.writeMicroseconds(1600);
  L_DCMotor.writeMicroseconds(1690);
  delay(delayLength);
}

void Left(int delayLength){ //Turn Robot Left while going Forward
  R_DCMotor.writeMicroseconds(1650);
  L_DCMotor.writeMicroseconds(1600);
  delay(delayLength);
}

void PivotRight(int delayLength){ //Pivot Turn Robot to the Right
  R_DCMotor.writeMicroseconds(1390);
  L_DCMotor.writeMicroseconds(1610);
  delay(delayLength);
}

void PivotLeft(int delayLength){ //Pivot Turn Robot to the Left
  R_DCMotor.writeMicroseconds(1620); //1700
  L_DCMotor.writeMicroseconds(1400);
  delay(delayLength);
}

void Dropoff(int delayLength){//Lower arm
  Serial.println("Lowering arm!");
  Arm_Motor.writeMicroseconds(1710);
```

```
  delay(delayLength);
}

void LiftUp(int delayLength){//Raise arm
  Serial.println("Lifting arm!");
  Arm_Motor.writeMicroseconds(1110);
  delay(delayLength);
}

void DriveReady(){ //Rear up on Tri-Wheels for Autonomous Mode
  if ((RProx > 300 && LProx > 300) || (RProx > 300) || (LProx > 300)){
    //Set Angle for Tri Wheels
    BackwardTri(100);
    HaltTri(300);
    if ((RProx > 300 && LProx > 300) || (RProx > 300) || (LProx > 300)){
      BackwardTri(50);
      HaltTri(300);
      ForwardTri(100);
      HaltTri(300);
      driveready = true;
    } else {
      ForwardTri(100);
      HaltTri(300);
      driveready = true;
    }
  } else {
    BackwardTri(100);
    HaltTri(300);
  }
}

//////////////////////LIGHTSENSING()////////////////////////////////////
void LightSensing() {

 ////////////////DECISION MAKING/////////////
  if ((leftlight > darkthreshold) && (rightlight > darkthreshold)) {
    //Decide if dark or bright (relatively)
    lowLight = 1; // it is dark
  }
  else{
    lowLight = 0; // it is light
  }
  if (CProx > dropthreshold) { //run med kit release sequence
    DepositLeave();
  }
```

```
  else if (lowLight == 0){
    //If light then go toward it
    SeekLight(); // Seek Light Autonomously
  }
  else if ((lowLight == 1) && (CProx < dropthreshold)){
    //If its dark, pivot to find light
    FindLight(); //Find the Light
  }
  delay(20); // Wait for servos to catch up
  }

//////////////////////METHODS FOR LIGHTSENSING()////////////////////////////////
void FindLight(){
  PivotRight(); //pivot until it sees the light
}

void SeekLight()  // Method for Light Sensor Driving
{
  if(rightlight < leftlight && lightdiff > lightbuffer){
    Right();
  }
  else if(leftlight < rightlight && lightdiff > lightbuffer){
    Left();
  }
  else if(lightdiff < lightbuffer){
    Forward();
  }
  else{
    Forward();
  }
  Serial.println("Seek Light");
}

void DepositLeave(){ //function to deposit med kit
  Halt();  // Stop to run arm function
  Serial.println("Stopped!");
  Arm_Motor.writeMicroseconds(1710); //Lower arm
  delay(5000);
  Serial.println("Lowering arm!");
  Arm_Motor.writeMicroseconds(1515); //pause arm while backing up
  BackUp();  // Back Away from MedKit
  delay(1500);
  Halt();
  Arm_Motor.writeMicroseconds(1110); //Raise arm
  Serial.println("Lifting arm!");
```

```
  delay(4000);
  stopbot = true; //stop robot
}
/////////////////WALL TRAVERSAL CODE/////////////////////////////

void Wall() {
  if (!FoundWall) { //If it hasn't found the wall, then find it
    FindWall();

  }
  else if (WallTraverse && BProx > 400){ //Condition for over the wall
    OverWall = true;
    Halt();
    HaltTri(2000);
  }
  else if (FoundWall && !OverWall) { //Drive over the wall
    WallTraversal();
  }
}
void FindWall(){ //Find the wall
  if (CProx > 330) {
    FoundWall = true;
    Serial.println("Found Wall");
  } else {
    Forward();
  }
}
void WallTraversal(){
    ForwardTri(); //Drive Tri-Wheels Forward
    Forward();
    WallTraverse = true;
}
/////////////////CHUTE NAVIGATION CODE/////////////////////////////
void ChuteNav(){
  if (CProx < exitthreshold && LProx < exitthreshold && foundchute){
     //sense a very far distance once exiting the chute
     Halt(2000); //diagnostic halt
     Serial.println("Exited Chute!");
     Forward(5000);
     LightSense = true;
   } else if (foundchute) {
     WallFollowD(); //follow the chute
   }
   else if (!foundchute) { //conditin to find the chute
     Forward();
```

```
      if (RProx > enterthreshold && LProx > enterthreshold) {
        foundchute = true;
      }
    }
}
double LDist, RDist;
void WallFollowD() {//tries to stay toward the middle
  //LDist and RDist convert the proximity sensor readings to inches
  LDist = -6.608*pow(10,-13)*pow(LProx,5)+2.018*pow(10,-9)*pow(LProx,4)-2.441
  RDist = -9.593*pow(10,-13)*pow(RProx,5)+2.808*pow(10,-9)*pow(RProx,4)-3.248
  //Navigates through the chute, staying in the middle
  if (RDist-LDist > 6) {
    PivotRight();
    Serial.println("Pivot Right");
  }
  else if (RDist-LDist > 3) {
    Right();
    Serial.println("Right");
  }
  else if (LDist-RDist > 6) {
    PivotLeft();
    Serial.println("Pivot Left");
  }
  else if (LDist-RDist > 3) {
    Left();
    Serial.println("Left");
  }
  else {
    Forward();
    Serial.println("Going Forward");
  }
}

//////////////////DATA GATHERING///////////////////
void GetData() {
  //Read from Proximity Sensors
  CProx = 0; //Center Prox
  RProx = 0; //Right Prox
  LProx = 0; //Left Prox
  BProx = 0; //Back Prox
  //Average Data Sample
  for(int i = 0; i < 100; i++) {
    CProx += analogRead(CProxpin);
    RProx += analogRead(RProxpin);
    LProx += analogRead(LProxpin);
```

```
    BProx += analogRead(BProxpin);
  }
  CProx = CProx/100;
  RProx = RProx/100;
  LProx = LProx/100;
  BProx = BProx/100;
  proxdiff = abs(LProx - RProx);

  //Read from light sensors
  rightlight = 0; //light1
  leftlight = 0; //light0
  //Average Data Sample
  for(int i = 0; i < 100; i++) {
    leftlight  += analogRead(leftlightpin);
    rightlight += analogRead(rightlightpin);
  }
  leftlight = leftlight/100;
  rightlight = rightlight/100;
  lightdiff = abs(leftlight - rightlight);
  /////////TEMP FOR DEBUGGING////////
  Serial.print("Right Prox: ");
  Serial.println(RProx);
  Serial.print("Center Prox: ");
  Serial.println(CProx);
  Serial.print("Left Prox: ");
  Serial.println(LProx);
  Serial.print("Back Prox: ");
  Serial.println(BProx);
  Serial.print("Right Light: ");
  Serial.println(rightlight);
  Serial.print("Left Light: ");
  Serial.println(leftlight);
  Serial.println();
}
//////////////////PRINTING METHODS//////////////////////////////////////
void PrintRC(){ //Print RC Channel Values
  Serial.println(" RC Control Mode ");
  Serial.print("Value Ch1 = ");
  Serial.println(Ch1);
  Serial.print("Value Ch2 = ");
  Serial.println(Ch2);
  //Serial.print("Value Ch3 = ");
  //Serial.println(Ch3);
  Serial.print("Value Ch4 = ");
  Serial.println(Ch4);
```

```
  Serial.print("Control = ");
  Serial.println(Ch5);
  Serial.print("Value Ch6 = ");
  Serial.println(Ch6);
  Serial.println(lowLight);
  delay(500);
}

//////////////////////LOOP FOR OBSTACLE COURSE/////////////////////////////
void loop(){
  Ch5 = pulseIn(8, HIGH, 21000); //Read from Channel 5 to check mode
  ///////////Autonomous Mode///////////
  if(Ch5 <= 1600){

    GetData();
    //Decision tree for autonomous methods
    if (stopbot) {
      Halt();
      HaltTri();
    } else if (LightSense) {
      LightSensing();
    } else if (driveready) {
      ChuteNav();
    } else if (OverWall) {
      DriveReady();
    } else if (!OverWall) {
      Wall();
    }
     delay(20);
  }
  ///////////RC Control Mode///////////
  if(Ch5 > 1600) { //  Read all channels for input
    Ch1 = pulseIn(12, HIGH, 21000);
    Ch2 = pulseIn(11, HIGH, 21000);
    Ch4 = pulseIn(9, HIGH, 21000);
    Ch6 = pulseIn(7, HIGH, 21000);
    DriveServosRC(); //Drive Motors under RC control
  }
}
```

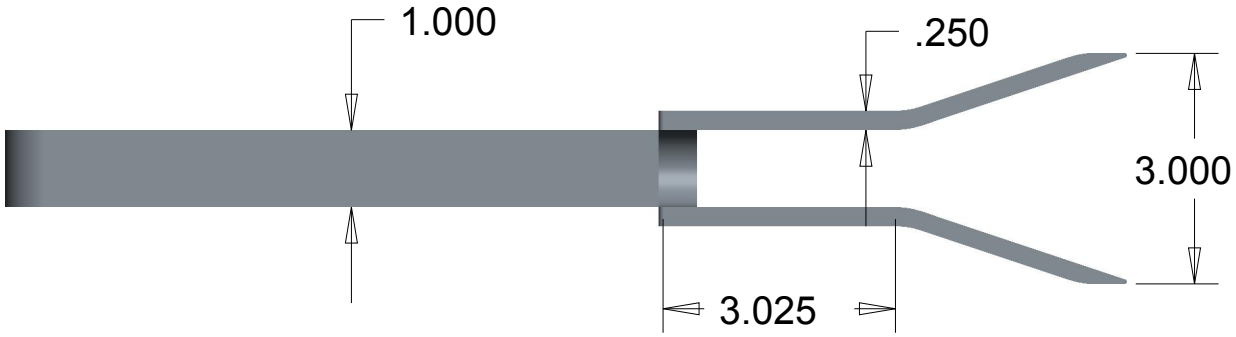
I Acknowledgements

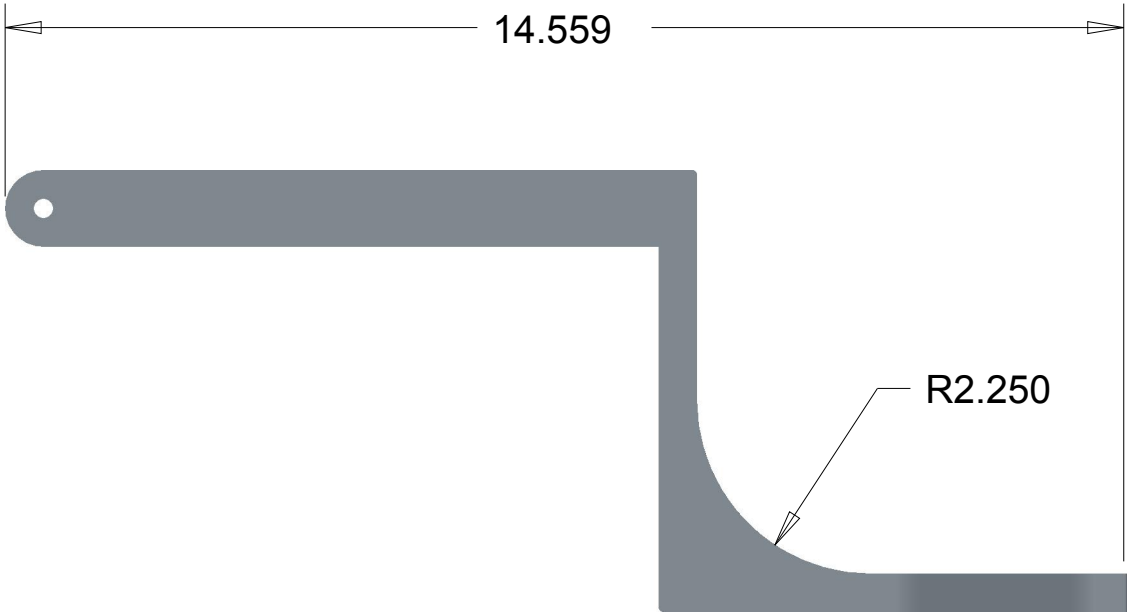
We thank Prof. Nosenchuck, Frank, Amy, Greg, Glenn, Chris, Yong and our classmates for making this an awesome learning experience.

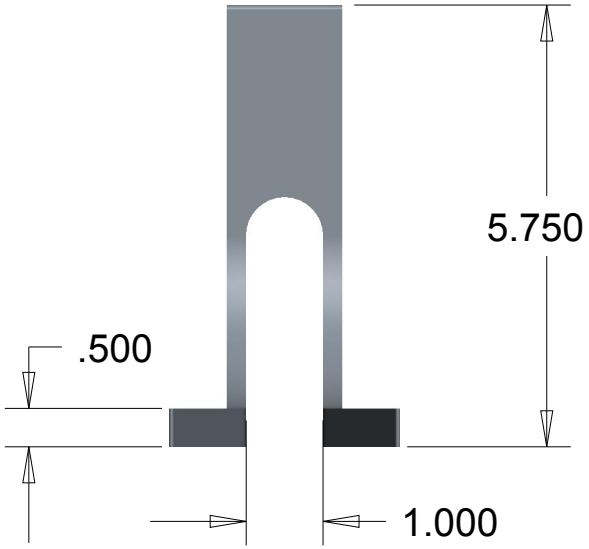
J Honor Code Pledge

We pledge our honors that we have completed this assignment in accordance with university rules and regulations. Signed, the members of team Seven Nation Army

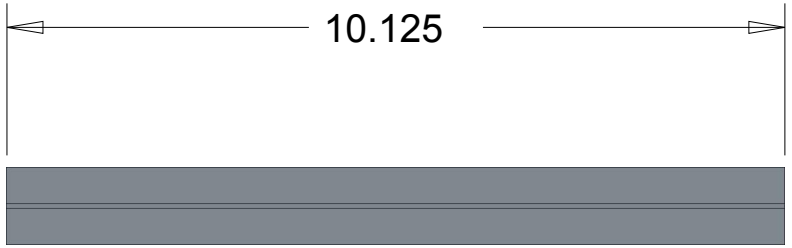
ARM
SCALE: 0.400
UNITS: INCHES

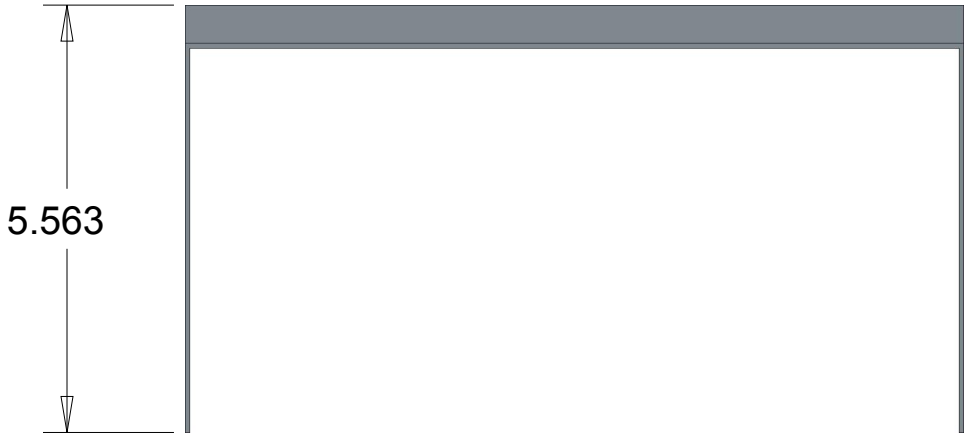


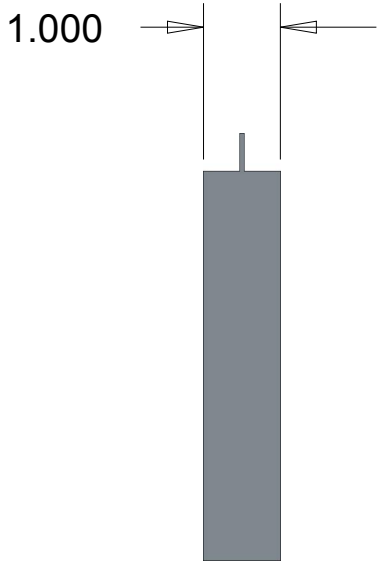


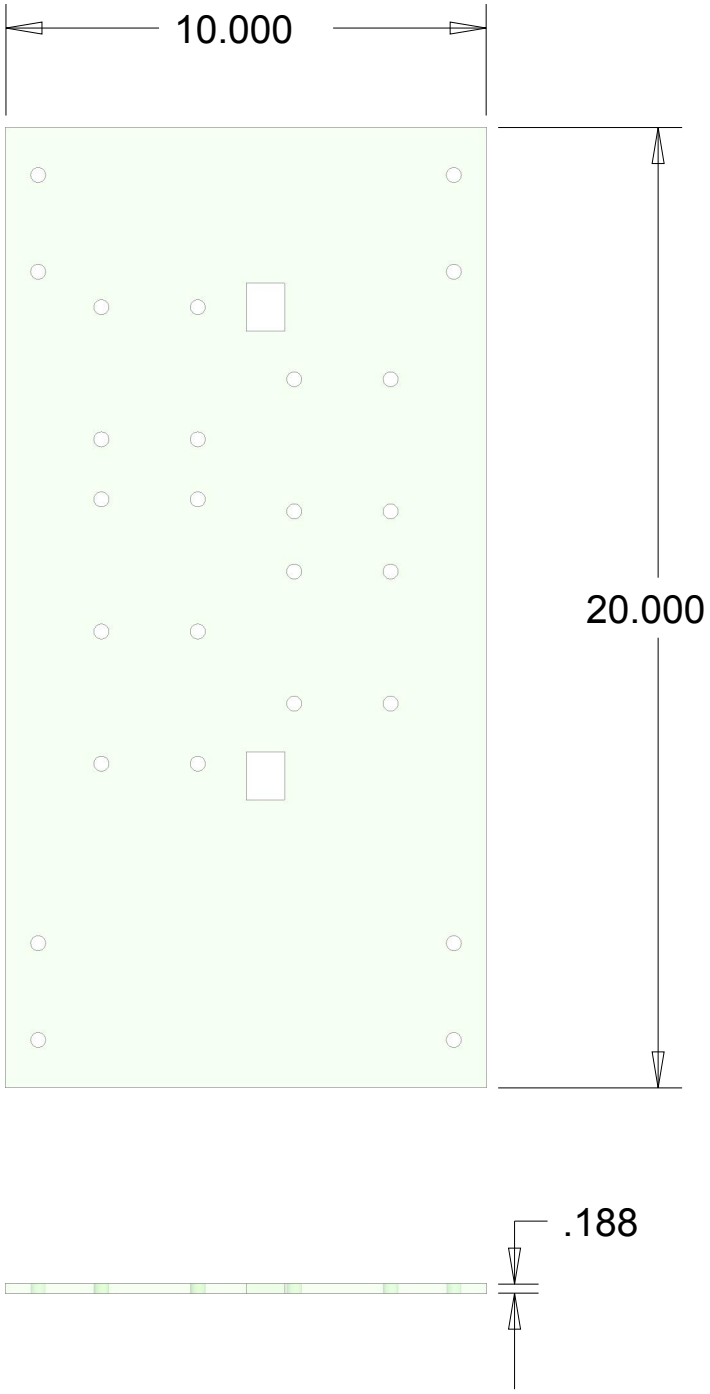


ARM STOP
SCALE: 0.400
UNITS: INCHES

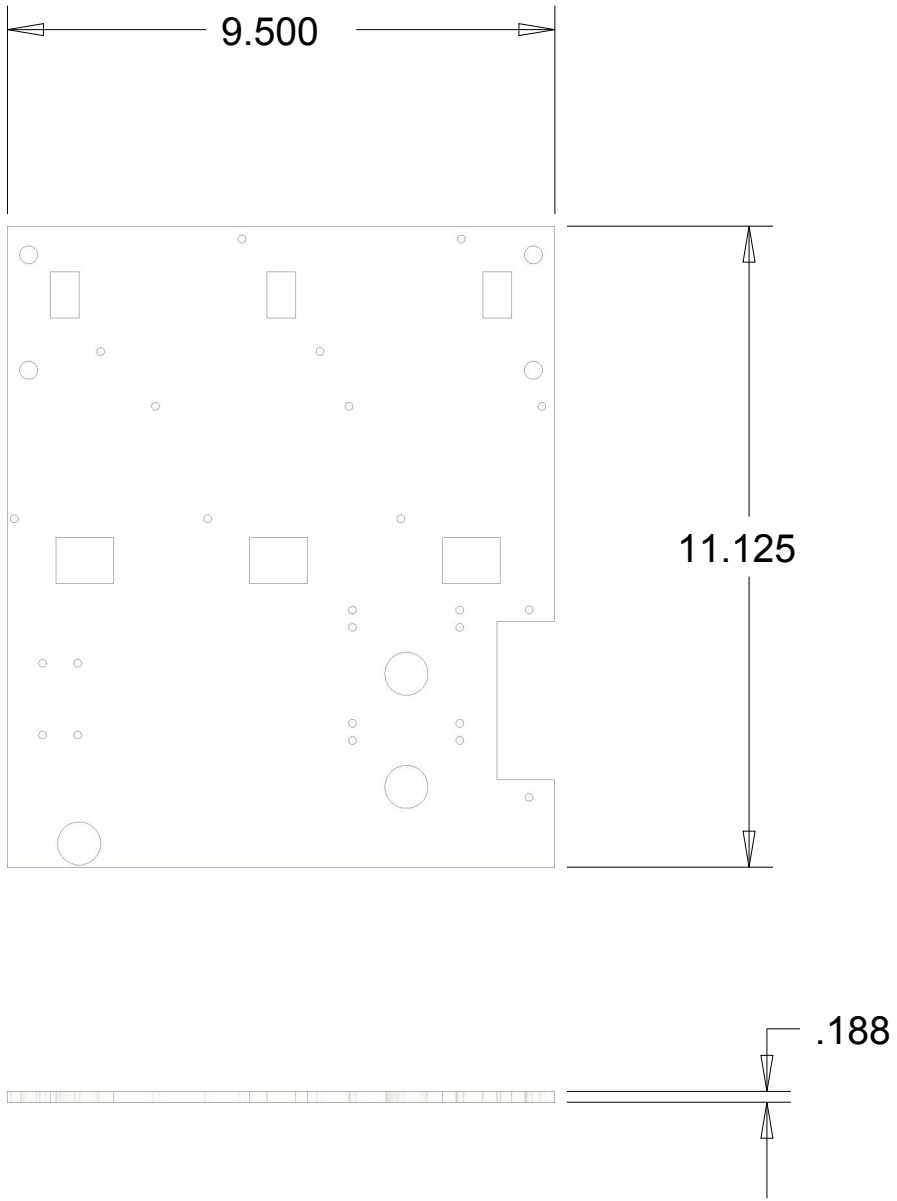






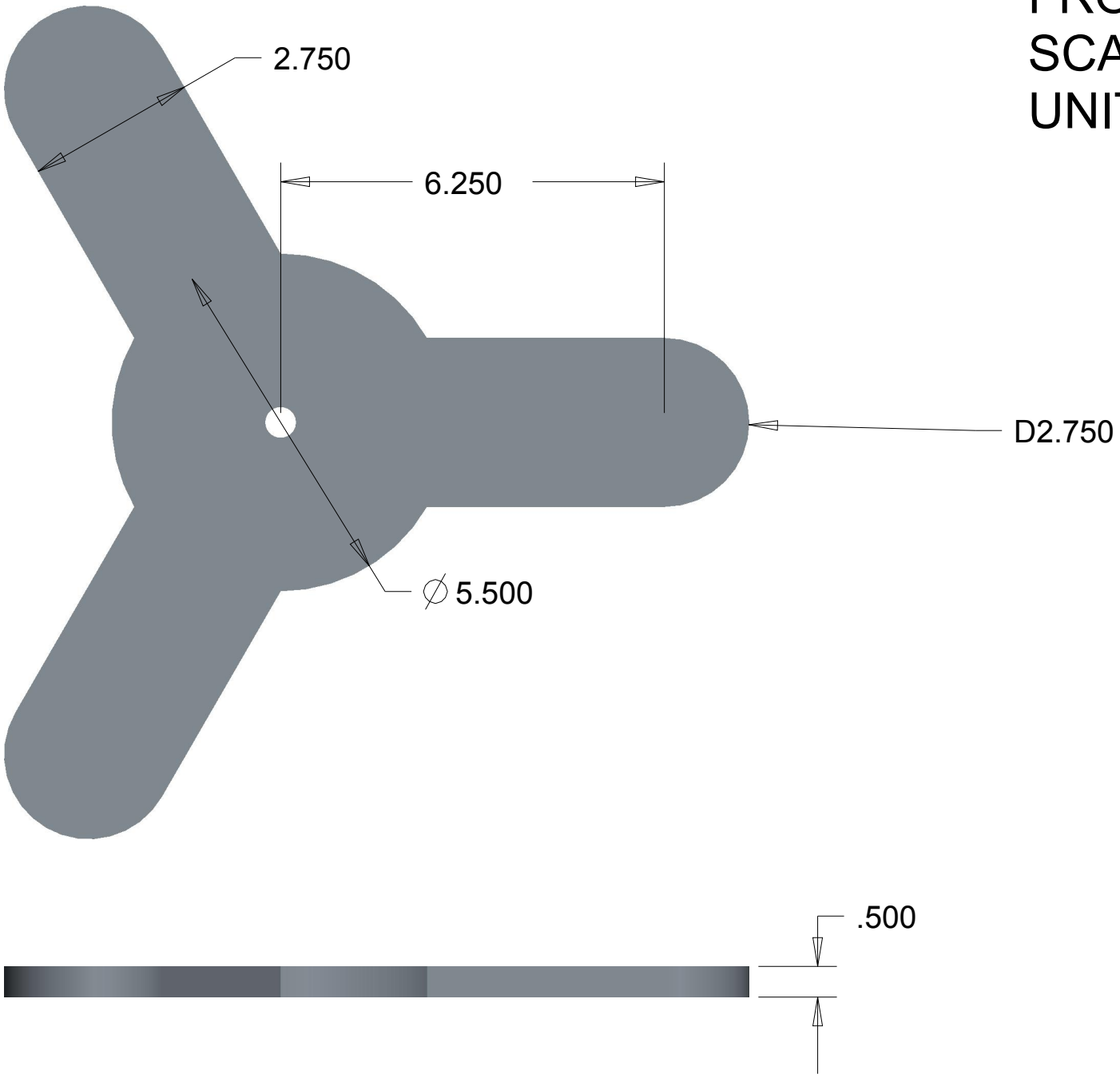


BASE PLATE
SCALE: 0.250
UNITS: INCHES

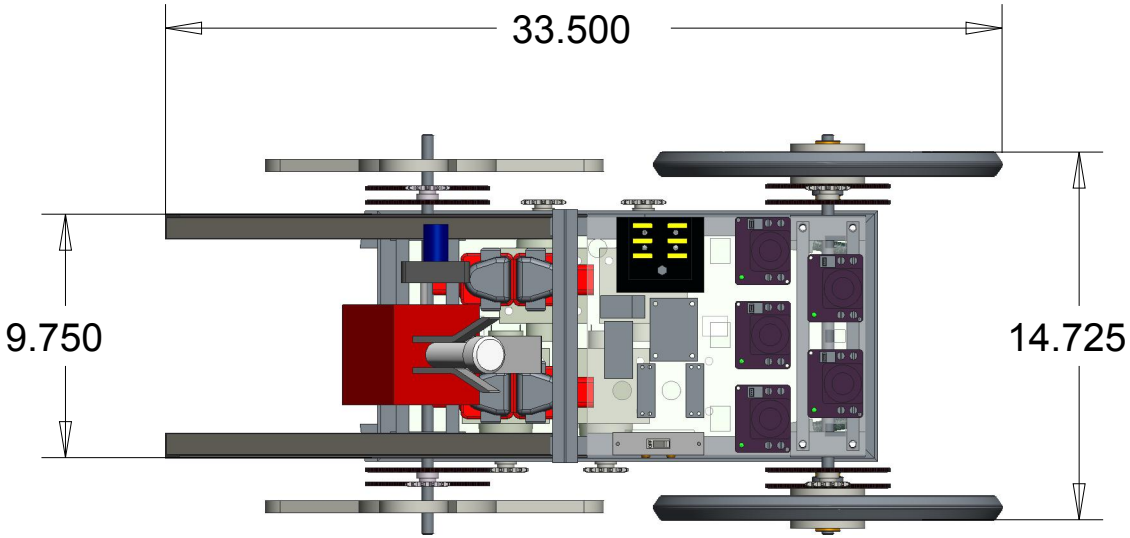


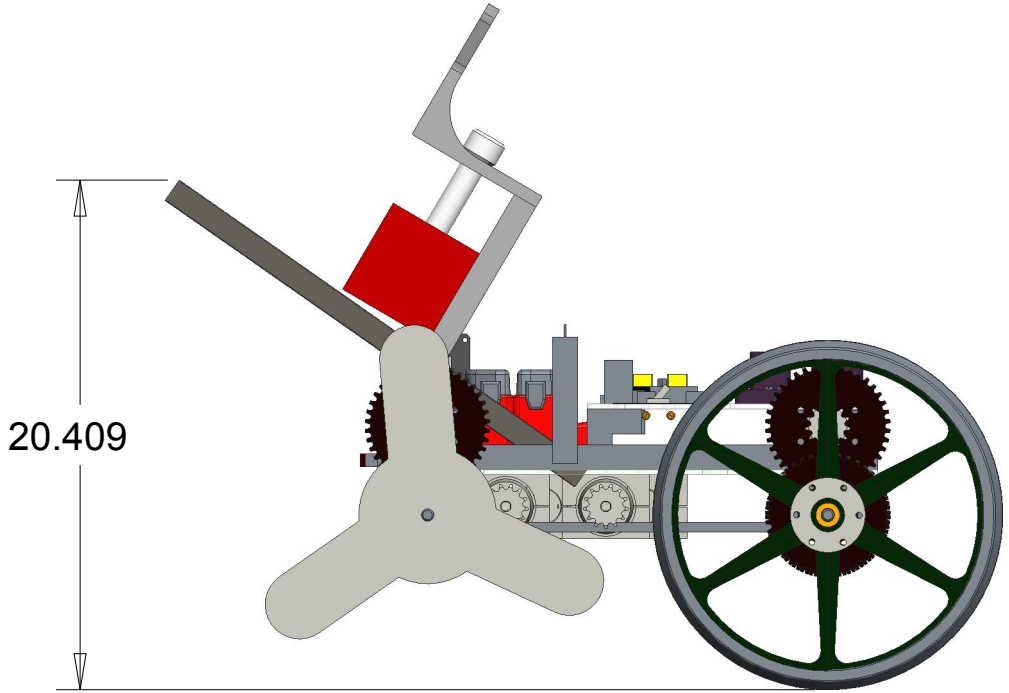
COMPUTER BOARD
SCALE: 0.300
UNITS: INCHES

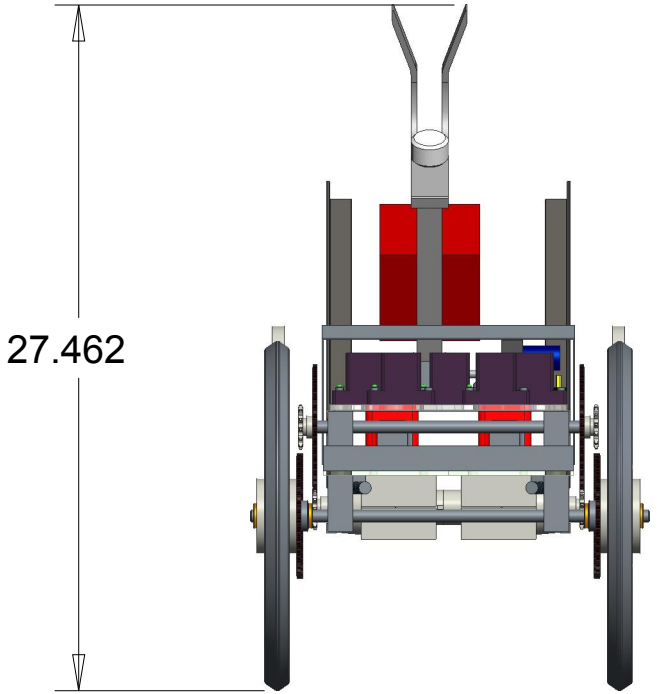
FRONT WHEEL
SCALE: 0.400
UNITS: INCHES



ROCKY
SCALE: 0.110
UNITS: INCHES







FRONT AND REAR SPROCKET
SCALE: 0.800
UNITS: INCHES