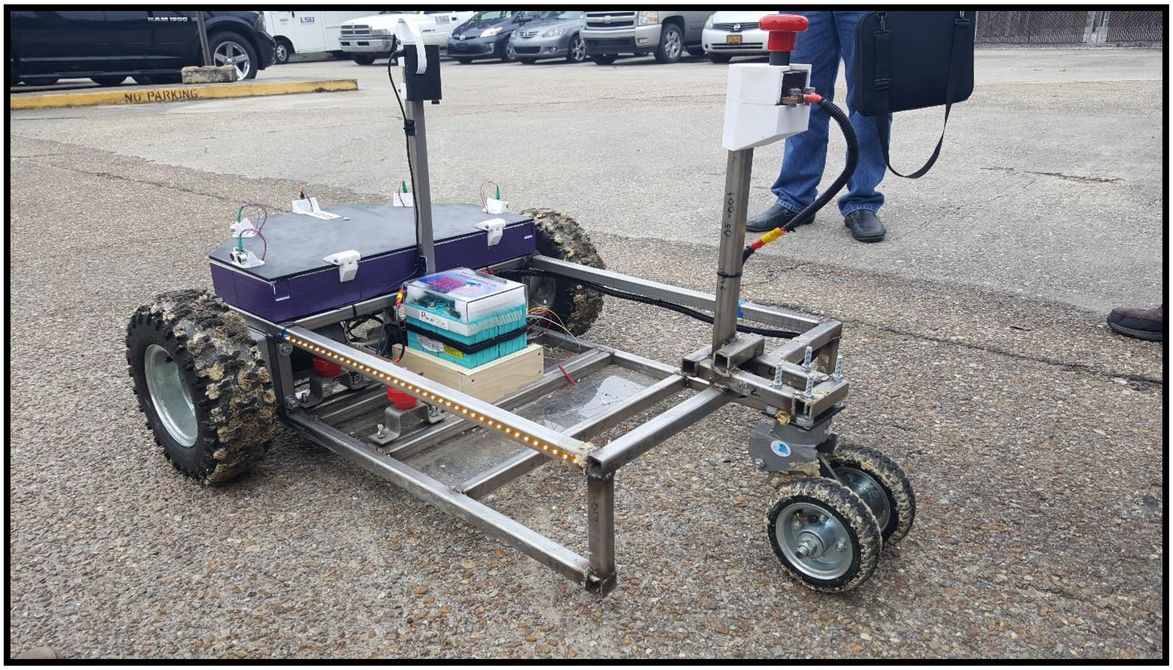
University Name: Louisiana State University
Vehicle/Team Name: Bengal Bot



Date Submitted: 5/15/16
Team Captain's Name and E-Mail: Robert Fletcher

Team Members Names and E-Mails

Team Member	Major	Email address
Aaron McCloud	Mechanical Eng.	aaronmccloud@outlook.com
Emerson Ashford	Computer Eng.	
Robby Fletcher	Computer Sc.	Mapping/Pathing
Tan Nguyen	Electrical Eng.	
Holden Chancey	Mechanical Eng.	Motors, Wheels

Faculty Advisors Name and Statement of Integrity: Sean King

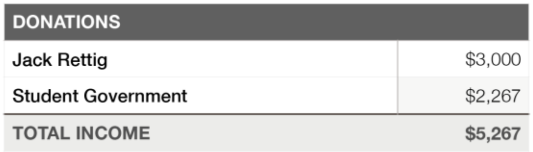
Introduction

The project undertaken by this team is the design and construction of an Intelligent Ground Vehicle. The Intelligent Ground Vehicle will be able to navigate autonomously through a course of obstacles to arrive at a predetermined goal location. There are constraints for the competition including vehicle size, speed, and safety considerations. This project was funded by our sponsor Jack Rettig, and was built at the LSU Mechanical Engineering shop

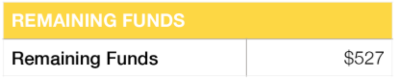
Budget

Initially we were donated $3000 to both build this robot and travel to Rochester, MI to compete in this competition. We realized that making this work would be extremely difficult. We reached out to Student Government and were able to secure funding for most of our travel (Airfare, Shipping the Robot, and a Rental Car), leaving the $3000 to pay for the robot, competition registration, and lodging.

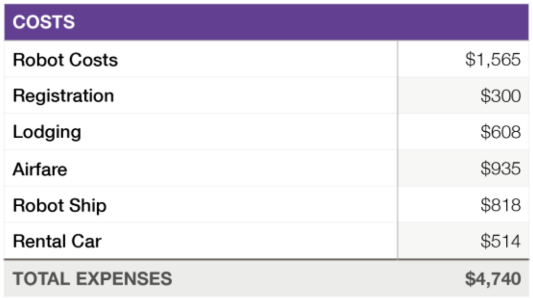
Below is a breakdown of how the money was actually spent.



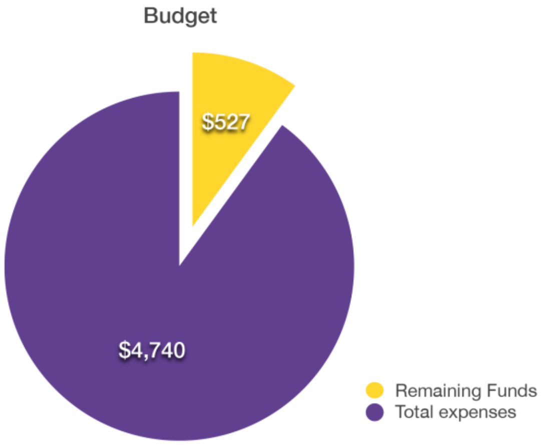
DONATIONS	
Jack Rettig	$3,000
Student Government	$2,267
TOTAL INCOME	**$5,267**



REMAINING FUNDS	
Remaining Funds	$527

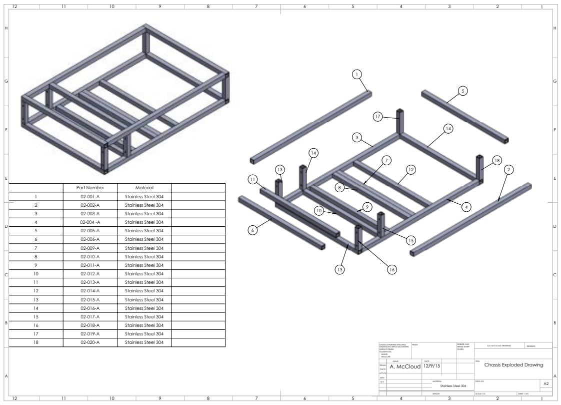


COSTS	
Robot Costs	$1,565
Registration	$300
Lodging	$608
Airfare	$935
Robot Ship	$818
Rental Car	$514
TOTAL EXPENSES	**$4,740**

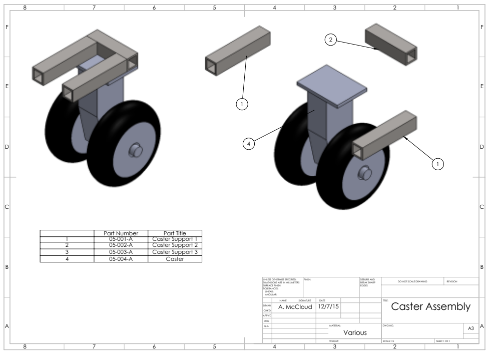


4. Description of Mechanical Design:

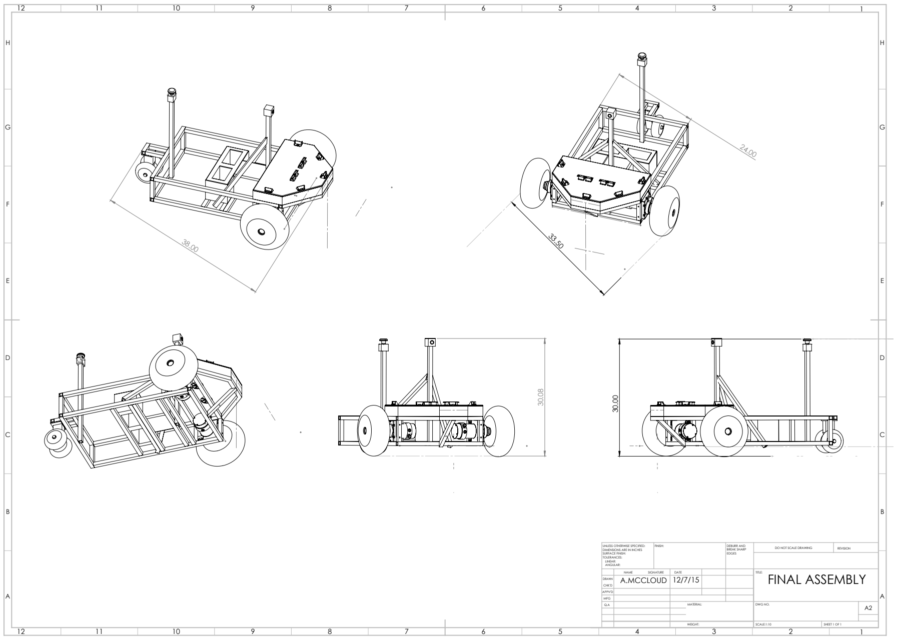
This robot consists of three subassemblies: chassis, electronics, and rear caster. The chassis is made of 304 Stainless Steel square tubing, welded together using MIG welding. To the rear of the frame is the caster assembly. This consists of two 5” wheels which rotate on an axis 360 degrees freely. Upon this system, the electronic subsystem is mounted. This includes two T-64 motors, the battery, cameras, and laptop assembly.



	Part Number	Material	
1	02-001-A	Stainless Steel 304	
2	02-002-A	Stainless Steel 304	
3	02-003-A	Stainless Steel 304	
4	02-004 -A	Stainless Steel 304	
5	02-005-A	Stainless Steel 304	
6	02-006-A	Stainless Steel 304	
7	02-009-A	Stainless Steel 304	
8	02-010-A	Stainless Steel 304	
9	02-011-A	Stainless Steel 304	
10	02-012-A	Stainless Steel 304	
11	02-013-A	Stainless Steel 304	
12	02-014-A	Stainless Steel 304	
13	02-015-A	Stainless Steel 304	
14	02-016-A	Stainless Steel 304	
15	02-017-A	Stainless Steel 304	
16	02-018-A	Stainless Steel 304	
17	02-019-A	Stainless Steel 304	
18	02-020-A	Stainless Steel 304	



	Part Number	Part Title
1	05-001-A	Caster Support 1
2	05-002-A	Caster Support 2
3	05-003-A	Caster Support 3
4	05-004-A	Caster



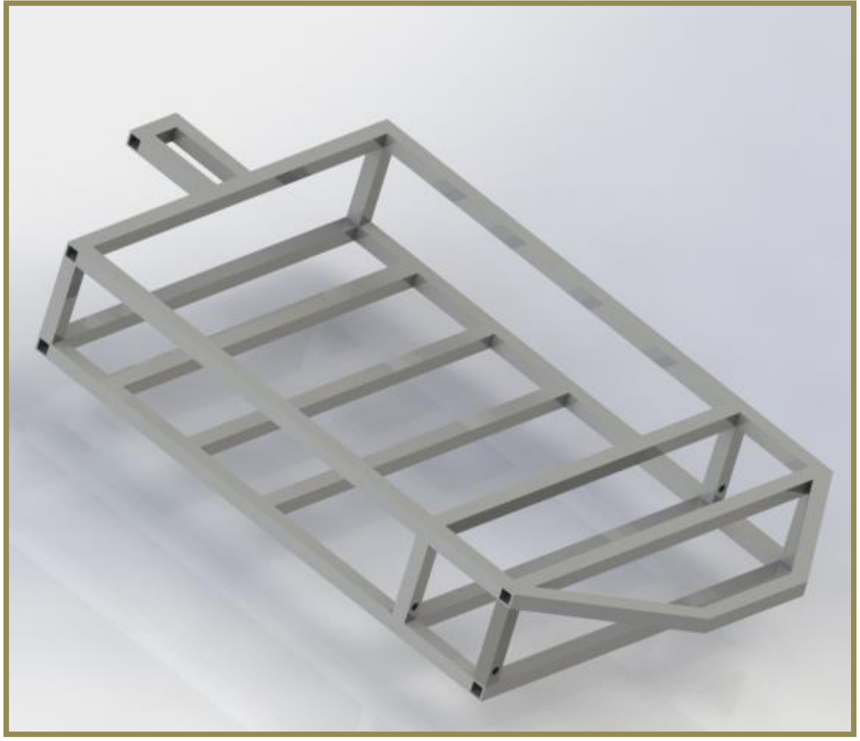
Decision on Frame Structure:

The following Quality Table was created to choose a material for our frame:

Desired Quality	Weight/100	Aluminum Score	Stainless Steel Score
Minimal Cost	20	20	10
Maximize Density	15	10	15
Maximum Yield Strength	15	15	10
Ease of Manufacture	20	10	18
Minimal Elasticity	30	10	30
Total Score:	100	65	83

Thus, Steel 304 was chosen due to its low elasticity, high weight, and ease of manufacturing. The issue of steels' high cost was balanced with the availability of resources donated to our team.

The chosen design for our chassis is as follows:



Data Acquisition & Processing

Sensor Array Configuration

We know the maximum speed of our robot is 5 miles per hour and that we would like to have at least three chances to spot an object before collision. Three was chosen as our safety factor because during testing we never had more than one false read in a row. We could have choses a safety factor of two but we wanted to maximize the amount of time our robot has to react. Using this information, we can determine a minimum data rate for our sensor array.

Since we are moving at a speed of 7.33 feet per second for we can define the distance moved during for any number of cycles, where a cycle is one read from each of the ultrasonic and serial communication, as follows.

$$D = (NC * CT) * 7.33$$

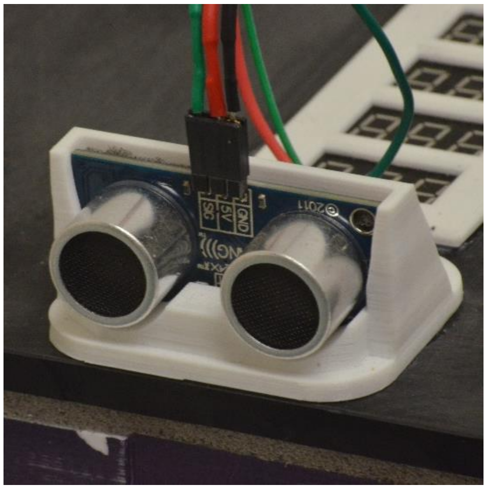
Where D is the total distance in feet, NC is the number of cycles, CT is the total cycle time, and 7.33 is the maximum speed of the robot in feet per second.
Since we know from the competition guidelines that the tightest point of the course will leave us 3 feet on either side we can set our maximum distance to 3 feet. Using our desired safety factor of 3 we arrive at CT = 136 milliseconds. Which gives us a minimum data rate of 1/136 milliseconds or 7.33Hz. This number gives the minimum data rate if the third read happens when the robot is literally touching the object. Clearly that isn’t want we want so we up the data rate to 10Hz or 100ms per cycle to get the following:

$$D = (3 * .1) * 7.33 = 2.19ft$$

This means that when the robot makes its third read it will have almost 10 inches of room left to stop. In the case of finding an object on the second read: 18 inches. First read: 27inches. For these reason we tuned our system to have a minimum data rate by limiting the maximum read distance.

To make sure that the sensor array was capable of maintaining this 10Hz data rate we tested the completed array in its worst case condition, no object in range forced sensor timeout. Using the NewPing library for Arduino we were able to set the maximum distance our sensors will check for by changing the timeout period. We decided the easiest way to find this maximum distance would be empirically. We started at the maximum range for our sensors, 10 meters, took 5 reads with it pointed up at the sky and checked the overall time from received command to finished command. We decreased the maximum read distance by 50cm each time until we found a distance that reliably gave us sub 100ms total cycle times. This distance is 200cm. Since 200cm is greater than the 3ft we used in the previous calculations we know that the robot will get at least three chances to see every object before possible collision.

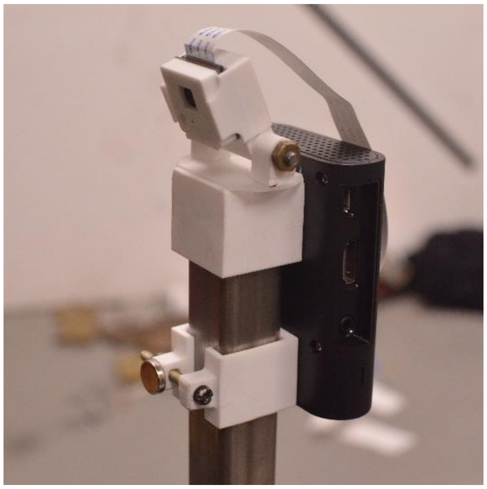
Parallax Ping))) Mount



This holder will hold the Parallax Ping))) by friction fit and keep it at a 90° angle to the sensor plate. It mounts the part upside down but since the sensor is not direction sensitive this will not affect performance. As you can see we have full access to the wiring pins from above and the part can easily be switched out in case of failure.

We ended up having to print the mounts about 1mm larger than we originally planned to deal with the difference dimensions between the Pings))) we received. We added a small block of foam to the rear of the sensor and that holds it in place quite snugly.

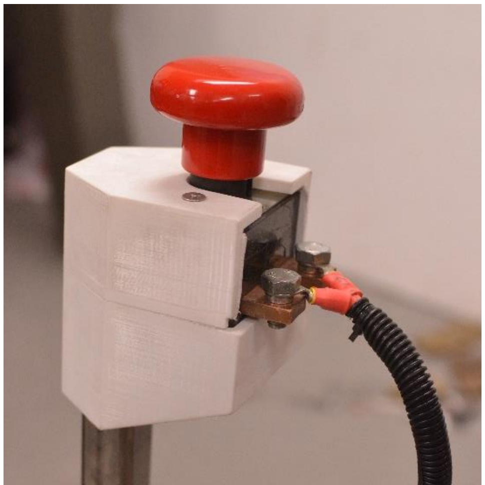
Raspberry PI 2 and Camera Mount



We needed to attach our Raspberry Pi and the Pi Camera to the frame while still maintaining the ability to adjust the angle of the camera. The mount shown to the right is a combination of a store bought Raspberry Pi case and some 3D printed parts.
The printed parts handle the attachment to the steel pole coming out of the frame in two different ways. The top cap is made to be a friction fit and must be hammered into place while the bottom one was printed with room for a tightening bolt. This allowed us to adjust the bottom to fi the store bought case exactly. The case is joined to the two frame members via CA glue.

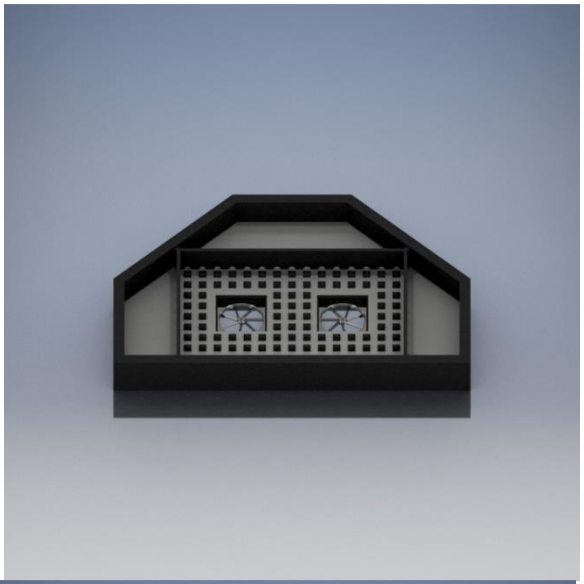
The camera mount is a modified version of user Frank26080115's PI camera mount from www.thingiverse.com. We added the angle adjustment mechanism located on the bottom of the camera mount as well as thickening the back for a stronger final product. We used the bolt to adjust the camera until we were satisfied with our field of view and then used a drop of CA glue to fix it more permanently in place.

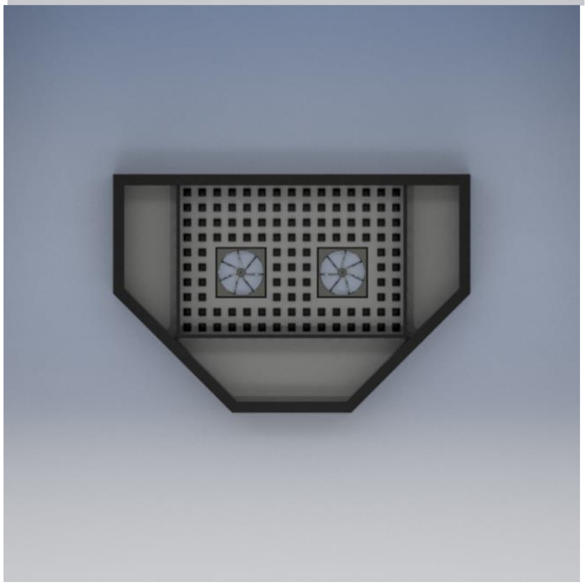
Emergency Button Mount



As per competition rules we need to have our emergency button mounted on the center rear of the robot. We decided to 3D print this mostly due to time constraints but it does come with a few benefits we did not immediately realize. First off once the PLA has been sanded it is actually quite smooth and very unlikely to cut you. During testing this button got a lot of use and due to the high speeds the robot travel at we ended up hurting ourselves far too often on the sharp steel of the mounting pole. Secondly the plastic insulates the emergency buttons exposed terminals from the frame. The way the button is made the back half actually has two exposed copper terminals that had shorted through the frame rendering the emergency button useless. The 5mm of PLA between the frame and the buttons terminals has guaranteed that this will never happen in the future.

Electronics Holder





We needed a way to hold all of our computing power. This holder needed to be:

- Water resistant
    - Protection from light rain and damp track conditions

- Shock resistant
    - Electronics should be safe when driving over rough terrain
- Cooled
    - Electronics should be kept well within their acceptable temperature ranges
- Sturdy
    - Strong enough to hold all electronics without breaking
    - Must be mountable on the frame
- Accurate
    - Must be square and symmetrical
    - Must hold sensors at correct angles

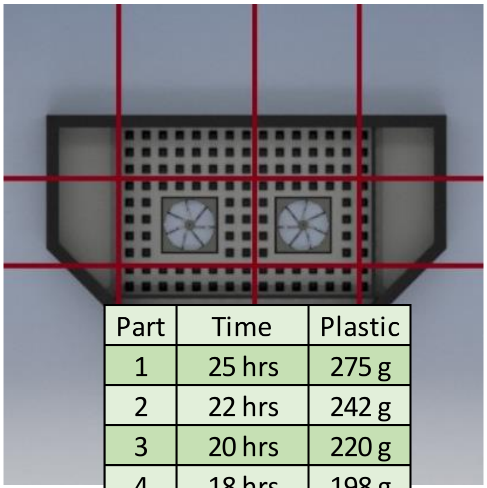
The design shown above is 3D printed in 12 parts in white PLA (Polylactic acid) plastic and constructed using CA (cyanoacrylic) glue. It holds both of our Arduino MEGAs, our Lenovo ThinkPad laptop and our power rectifier PCB (printed circuit board).

3D Printing Overview

We decided to use 3D printing instead of CNC milling for two main reasons: cost and experience. Using a 3D printer to make our parts meant that our parts would be hollow with an internal support grid. Being hollow this parts would use less plastic than a traditional solid one. This saves us both money, in materials, and reduces the overall weight of the sensor array. We were also much more familiar with 3D printing than we were with CNC milling. We wanted to make sure that out design would work the first time due to the high cost of plastic and 3D printing allowed us the ability to print scale models before committing to the real thing.

Total time and material bill

Due to the size constraints of the printers available to us we had to print the box in 12 parts as shown to the right.



The parts each took between 10 and 25 hours to print and used between 200 and 500 grams of PLA filament. The slicer estimates are shown in the chart below. These numbers are what the printer thinks will happen and as such are not 100% accurate.

Part	Time	Plastic
1	25 hrs	275 g
2	22 hrs	242 g
3	20 hrs	220 g
4	18 hrs	198 g
5	14 hrs	154 g
6	18 hrs	198 g
7	18 hrs	198 g
8	12 hrs	132 g
9	13 hrs	143 g
10	15 hrs	165 g
11	13 hrs	143 g
12	15 hrs	165 g
	203 hrs	2233 g

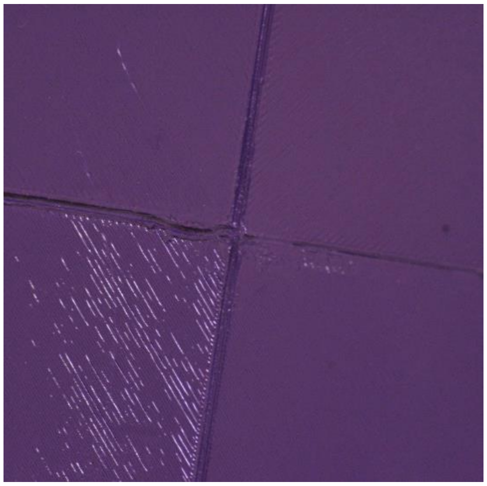
As you can see printing this box took 203 hours of combined printing time and 2.2Kg of filament. We purchased three 1Kg rolls of Hatchbox PLA for a combined cost of $66. If we had gone with CNC milling we would have needed to buy a solid 16X24X3 inch of plastic and our best estimate from McMasters was well into the $300 range.

Plastic Choice and Attachment

We needed to decide on what type of plastic to use for our printing initially we had planned to use ABS (Acrylonitrile butadiene styrene) so that we could solvent weld the individual components of the box together using acetone. We went as far as buying a single Kilogram of high quality ABS and doing some test prints. We learned that while our printers were capable of printing in ABS getting high quality warp-free parts was beyond our reach.

The only other plastic choice that we had was PLA (Polylactic acid) which is the most widely used printer filament on the market. PLA is very easy to print with due to its low melting point, resistances to thermal expansion, and great inter-layer adhesion. The only reason we didn’t want to use it in the first place is that it is immune to all the solvents we have access to making solvent welding impossible.

After doing some research online we found that the strongest bond we could get between two PLA parts with readily available adhesives was with CA glue.
Which is more commonly known as Superglue in retail settings. CA glue is effective due to its ability to bond with the surface of PLA parts unlike Epoxy which has a tendency to peel off plastic surfaces upon curing. Unfortunately, even CA glue isn’t as strong as the original plan of solvent welding so to add additional strength we “welded” all of the seams in our box after gluing.



This was done using the same Hatchbox PLA filament and a soldering iron with a flat tip. The process involves setting the soldering iron right above the melting point of PLA (200c) and steadily running it through across the seam while carefully feeding in filament. It took a few tries to get used to but the end result was a set of seams that are water tight and very strong. The results can be seen above.

Electronics Box Heat

The TDP (Thermal Design Properties) is a measure, in watts, of how much heat the CPU outputs at maximum. Ours a 4th generation Intel Core i7-4702MQ has a TDP of 33w which we will scale to 40W just to be on the safe side and to include our other smaller systems like the Arduinos and the rectifier PCB. Next we can convert to BTUs/hr. using the conversion factor of 3.41giving us 136.42 BTU/hr. Using the heat removal method we can solve for the amount of air flow needed on a 90° day to keep our box at 50° which is a pretty standard operating temperature for mobile processors.

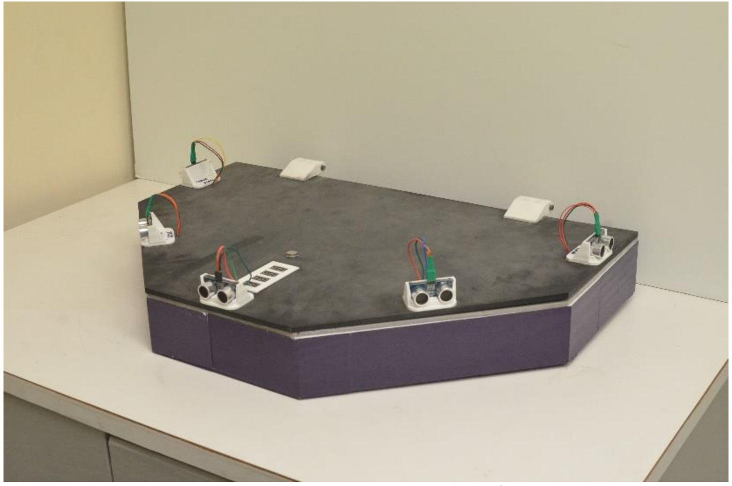
$$CMF = \frac{BTU/hr}{1.08 * (External - Desired)} = \frac{136.42}{1.08 * (90 - 50)} = 3.2CFM$$

As you can see we only need about 3 CFM to keep our box running at 50°. Using our dual 80mm fans, each rated at 35CFM, we can have 70CFM available. This gives a safety factor of:

$$\frac{70}{3.2} = 21.8$$

Even with the fact that we are using standard computer fans which do not deal with high pressure situations very well we feel confident that even with sub optimal performance our parts will not overheat over the course of a 10 minute run. During testing we have had zero issues with component heat inside the box.

Full Sensor Array Major Design Changes



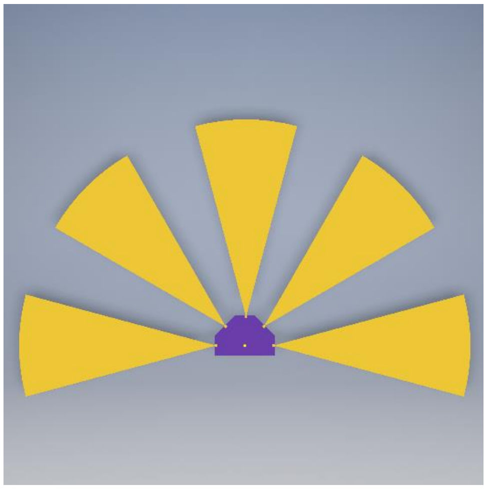
The original sensor array design called for 4 LIDAR Lite V2 and 5 Parallax Ping))) but due to the aforementioned supply problems we have been left with only our 5 Parallax Pings)))'s. This hasn't changed the overall shape of our sensor array but it has changed the method by which read the sensor.

Our previous design had the Arduino moving the servos, calling one Ping))) and then all four LIDAR each cycle. The wait time for the servo to get into position was when the Ping)) took its read minimizing dead time on the system. However now we are simply calling all 5 Ping)))s clockwise around the array.

The top is now held on with a set of 3D printed hinges and held shut with four magnetic clasps. The entire array is attached to the frame via Velcro.

Blind Spots

With the loss of the LIDARs we lost all of our non-stationary sensors. Since all of our remaining sensors are at fixed angles this means that we have certain regions that will never get scanned. As you can see to the right all of our blind spots, areas not covered by gold, are at angles to the front of the array. Since our robot can only drive forward or make a zero point turn it isn't possible for an object to travel diagonally in relation to our robot. These blind spots are also 30 inches wide at their widest point and 10 inches at their narrowest point. The average size of an object on the course is around 18 inches wide so even it id did manage to sneak into the blind spot it would be detected before it could cause a collision.



Sensor Array Communication

Since we are a moving system the distance reads are only valid at the moment they are taken. The longer we wait to use them the closer the robot actually is compared to the recorded read. In order to maximize the freshness of the data we are using an ask first data collection scheme. The code snippet below handles gathering commands from serial and beginning the read cycle.

```
 if (Serial.available() > 0 )
  {
    input = Serial.readString();
  }
  if (input == "N" || input == "n")
  {
    start = millis();
    input = "S";
```

Once it has received the next command order the following for loop executes to gather all of the necessary data into an array called Distance.

```
for (int i=0; i<5; i++)
    {
      Distance[i] = Sonar[i].ping_cm();
    }
```

Now that we have all of our data we need to format it and send it back to the laptop. The format we have choses is (D1|D2| D3|D4|D5). The parentheses are our delimiter for each data set and the bars are a delimiter for each individual data type. The code below handles both the formatting and the sending.

```
Serial.print("|");
    for (int x=0; x<5; x++)
```

```
{
  Serial.print(Distance[x]);
  Serial.print("|");
}
```

Bandwidth

Since we are using UBS serial at a baud rate of 9600 bps we need to be sure that we won't ever exceed this. Each cycle of reads contains five 2 byte integers and six 1 byte characters bring our total cycle size to 16 bytes or 128 bits. At our maximum cycle rate of 40hz we will be sending 5120 bits per second which is still less than our 9600 maximum. If at any point in the future the sensor array is upgraded, we will simply need to re run these calculations and if needed raise the Baud rate to the next highest supported rate.

Servo analysis

To prove that 28oz/in is enough torque we need to find the moment of inertia for our part with the LIDAR attached. Since the weight will be relatively evenly distributed we can assume it is a cylinder whose moment of inertia is given by:

$$I = \frac{1}{2}MR^2$$

Which for us is equal to

$$I = \frac{.043kg}{2}.\left(\frac{.057m}{2}\right)^2 = .0000174663kgm^2$$

We can then plug that into newton's second law

$$\tau = I\alpha$$

With total torque as 28ozin ≈ 0.197Nm

$$0.197Nm = .0000174633kgm^2\ \alpha$$

Solving for angular acceleration we get $\alpha = 11322.2rads/s^2$
Plugging this into $\emptyset = \omega_0 t + \frac{1}{2}\alpha t^2$ with theta = 60° ≈ 1.05 radians and $\omega_0 = 0$ yields t = .01 seconds which is less than the 0.05 seconds to rotate 60° given by the manufacturer by a factor of 5. Since the theoretical speed is much faster than the printed maximum speed we know the torque will not limit our rotation speed so we will be able to use the servos max speed in our sensor array.

Furthermore given the time for 60° of rotation we can solve for the angular acceleration necessary to rotate 60° in 0.05 seconds and use that to find the torque applied by the servo.

$$1.05 = 0t + \frac{1}{2}\alpha t^2$$
$$\alpha = 840\, rads/s$$

From this angular acceleration and the moment of inertia we can find anticipated torque required to maintain maximum speed.

$$\tau = I\alpha = .0146Nm$$

Furthermore since servos are powered by a DC motor the relationship between output torque and current is approximately linear. From the datasheet we are given that at 0 load (almost 0 torque) it will draw 250ma and at max load (0.197Nm of torque) it draws 1600ma we get the line defined by the equation:

$$Current(ma) = 8096.45\, Torque\ (Nm) + 250ma$$

With this we can estimate that we will be using roughly 368ma per servo under normal operating conditions. While this is not a perfect estimate it should be good enough for preliminary design of the power supplies and regulator circuits. We will however need to accommodate spikes in current up to 1A per servo due to inrush current.

Ultrasonic Mount Strength Analysis

To determine if out Ultrasonic holder will be strong enough to withstand the forces applied due to acceleration we solved for the deflection of a beam and the stresses applied at the point of contact.

$$Max\ deflection = \frac{PL^3}{3EI}$$

Where P is the force applied to the end of the beam, L is the length of the beam, E is the young's modulus, and I is the moment of inertia.

For P we used the weight of the part (9grams) and the max acceleration it can experience (2.44m/s^2) for a force of 0.02196 N. We know that this is not the actual force but it is the easiest way to overestimate the max force. I is equal to 1.6*10^{-6}, L is 25mm and E was found to be between 1.4-3.2 we will be using 1.4. Using these values we have a total deflection of:

$$Max\ deflection = \frac{(.02196)(.25)^3}{(3)(1.4)(1.6 * 10^{-6})} = 1.3 * 10^{-18}\ Meters$$

Furthermore, we can solve for the stress at the base of this part using

$$\frac{WI}{Z} = \frac{\frac{WI}{hb^2}}{4} = \ .000014\ Pa$$

ABS plastic has a yield strength of 42.5MPagiving a safety factor of:

$$\frac{42500}{.000014} = 3{,}035{,}714{,}285$$

We feel that given we overestimated the force applied, used the lowest estimate we could find for young's modulus, and still have a safety factor of 3 billion that this part, and all subsequent sensors mounts, are sufficiently strong. We will not be showing analysis for the other sensor mounts for this reason.

Battery Operation and Charging Time Analysis

$$\text{operation time}_{\text{max}} = \frac{\text{battery rating (amps} * \text{hours)}}{\text{current draw (amps)}}$$

$$= \frac{\text{20Ah}}{\text{58A}} \approx \text{21 minutes}$$

Our battery is rated for 20Ah our estimation of max current draw for all components would be around 58 A. This gives us an operation time of around 21 minutes at the worst case scenario

$$\text{operation time}_{\text{avg}} = \frac{\text{battery rating (amps} * \text{hours)}}{\text{current draw (amps)}}$$

$$= \frac{\text{20Ah}}{\text{22.41A}} \approx \text{53 minutes}$$

Since we do not expect the components to operate at maximum current draw at all times, the expected operation time of the total system should be around 53 minutes. This would allow us to run multiple tests before having to charge again.

$$charging\ time = \frac{battery\ rating\ (amps * hours)}{current\ supplied\ (amps)}$$

$$= \frac{20Ah}{6A} \approx 3.33\ hours$$

The battery itself is comprised of 8 separate 3.2v 20Ah cells that was custom manufactured by batteryspace.com. The battery has a 6A intelligent charger and it would take approximately 3.33 hours to charge to full capacity.

Data Processing

This is our code that reads in the sensor data and maps our surroundings:

```
import threading
import serial
import numpy as np
# import astar


# Visualization Libraries
from bokeh.client import push_session
from bokeh.plotting import figure, curdoc


# Testing Libraries
import time
import random
import sys


print("Initializing..")




MAP_SIZE_X = 3084
MAP_SIZE_Y = 6168


origin = [MAP_SIZE_X, MAP_SIZE_Y]
location = origin


viz = 0


if (len(sys.argv) > 1):
    viz = int(sys.argv[-1])


ser = serial.Serial('/dev/tty.usbmodem1411', 9600)
time.sleep(1)


robotMap = np.zeros(shape=(MAP_SIZE_X * 2, MAP_SIZE_Y * 2))
```

```
trans = [[-1, -.7071, 0, .7071, 1], [ 0,  .7071, 1, .7071, 0]]


if (viz == 1):
    x = []
    y = []


    p = figure()
    c = p.circle(x, y, size=5, color="red", alpha=0.1)
    o = p.circle(origin[0], origin[1], size=25, color="purple")


    # open a session to keep our local document in sync with server
    session = push_session(curdoc())




def ultrasonic():
    ser.write(b'n')
    while (ser.inWaiting() == 0):
        continue
    s = ser.readline()
    if (viz == 0):
        print(s)
    s = s[1:-3].split('|')


    obstacles = []
    for i in range(5):
        obstacles.append([int(location[0] + int(s[i]) * trans[0][i]),
                          int(location[1] + int(s[i]) * trans[1][i])])


    for i in range(5):
        robotMap[obstacles[i][0]][obstacles[i][1]] += 1
        if (viz == 1):
            c.data_source.data["x"] = c.data_source.data["x"] + [obstacles[i][0]]
            c.data_source.data["y"] = c.data_source.data["y"] + [obstacles[i][1]]


    if (viz == 2):
        print(robotMap)
        print(chr(27) + "[2J")
```

```
# def camera():


print("Ready to go!")


if (viz == 1):
    curdoc().add_periodic_callback(ultrasonic, 5000)
    #
    session.show() # open the document in a browser
    #
    session.loop_until_closed() # run forever


while(1):
    ultrasonic()
```

This is our code that creates the path:
(Credit to Christian Careaga for basis for this code)

```
from heapq import *


import time


def heuristic(a, b):
    return (b[0] - a[0]) ** 2 + (b[1] - a[1]) ** 2


def astar(array, start, goal):


    neighbors = [(0,1),(0,-1),(1,0),(-1,0),(1,1),(1,-1),(-1,1),(-1,-1)]


    close_set = set()
    came_from = {}
    gscore = {start:0}
    fscore = {start:heuristic(start, goal)}
    oheap = []


    heappush(oheap, (fscore[start], start))
    while oheap:

        current = heappop(oheap)[1]

        if current == goal:
            data = []
            while current in came_from:
                data.append(current)
                current = came_from[current]
            return data

        close_set.add(current)
        for i, j in neighbors:
            neighbor = current[0] + i, current[1] + j
            tentative_g_score = gscore[current] + heuristic(current, neighbor)
            if 0 <= neighbor[0] < array.shape[0]:
                if 0 <= neighbor[1] < array.shape[1]:
                    if array[neighbor[0]][neighbor[1]] == 1:
                        continue
                else:
                    # array bound y walls
                    continue
            else:
                # array bound x walls
                continue

            if neighbor in close_set and tentative_g_score >= gscore.get(neighbor, 0):
                continue

            if  tentative_g_score < gscore.get(neighbor, 0) or neighbor not in [i[1]for
i in oheap]:
                came_from[neighbor] = current
                gscore[neighbor] = tentative_g_score
                fscore[neighbor] = tentative_g_score + heuristic(neighbor, goal)
                heappush(oheap, (fscore[neighbor], neighbor))

    return False
```

```
'''Here is an example of using my algo with a numpy array,
   astar(array, start, destination)
   astar function returns a list of points (shortest path)'''


nmap = numpy.array([
    [0,0,0,0,0,1,1,2,2,3,4,5,4,3,2,1,0,0,0,0],
    [0,0,0,0,0,0,0,1,1,2,3,4,5,4,3,2,1,0,0,0],
    [0,0,0,0,0,0,0,0,0,1,2,3,4,5,4,3,2,1,0,0],
    [0,0,0,0,0,0,0,0,0,0,1,2,3,4,5,4,3,2,1,0],
    [1,1,0,0,0,0,0,0,0,0,1,2,3,4,5,4,3,2,1,0],
    [2,2,1,0,0,0,0,0,0,0,0,1,2,3,4,5,4,3,2,1],
    [3,2,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3,2,1],
    [4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3,2,1],
    [5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3,2],
    [5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3,2],
    [5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3,2],
    [4,5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3],
    [4,5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3],
    [4,5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3],
    [4,5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3],
    [4,5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3],
    [4,5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3],
    [4,5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3],
    [4,5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3],
    [4,5,4,3,2,1,0,0,0,0,0,0,0,1,2,3,4,5,4,3],
    ])


a = astar(nmap, (19,9), (0,0))
```

Testing & Validation

Data Acquisition

Parallax Ping)) Speed Test

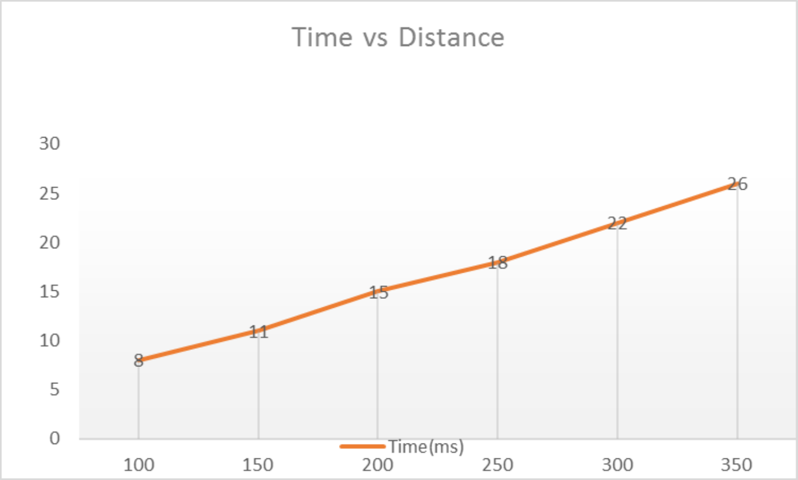
Objective

The purpose of this test is to determine at what distance the read time exceeds 20ms.

Procedure

A single Ping))) is set up 36 inches off the ground facing a piece of foam board at Xcm and a set of 10 readings is taken. Using the Arduino's built in millis() function at the begin and end of the loop allows us to compute the total time for 10 reads at distance X. the distance is reduced by 50 cm until the average per read time is below 20ms.

Results



We found that the point at which we exceed 20 ms is somewhere between 250cm and 300cm. This means that we can set the maximum read distance to 250cm and never have any total cycle read times greater than 100ms.
This does not account for serial communication times but we tested for that later and found that sending the 128bits of information took at most 4ms. For this reason, we have chosen to set the array maximum at 200cm so that our total time won't exceed 100ms even with serial read times.

Parallax Ping)) Accuracy Test

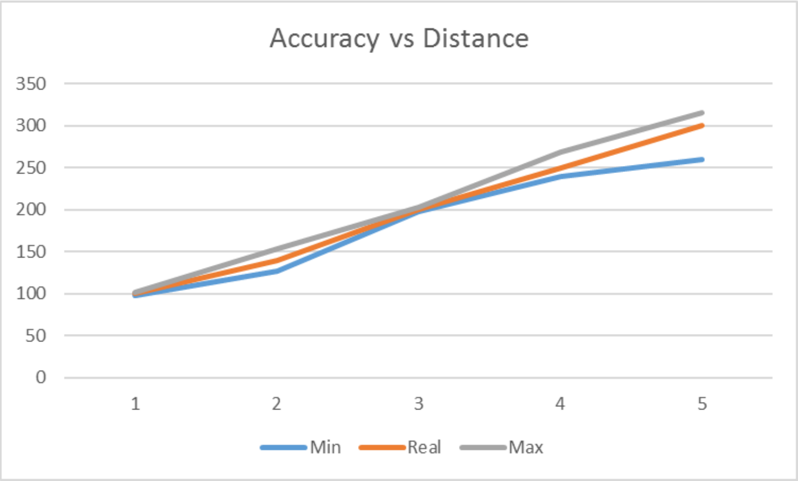
Objective

This test was used to determine if the ultrasonic sensors would be accurate out to our maximum distance of 200cm.

Procedure

Set up is the same as the speed test. 36inches off the ground pointed at a large piece of foam board. Each reading is taken 10 times at each distance and saved to a log file.

Results



The data reads are within tolerances all the way up to about 300cm where they start to diverge. This means that at our maximum distance of 200cm we don’t need to worry about the distance returned being wrong by more than 1 or 2 cm. Which should be taken care of by a properly weighted grid.

The second set of data is at 150cm and we discovered after testing that it was picking up a bit of trash at 127 cm from the start point which messed up our data at that point.

Full Array Speed Test

Objective

The purpose of this test is to further refine the data gathered in the individual speed test. Finding an average cycle time/rate for the full sensor array in a variety of real world conditions.

Procedure

The array is placed on the robot and a variety of obstacles (chairs, trashcans, etc.) are placed around the array at various distances. A set of reads is taken and times. The obstacles are moved to new locations and the test is repeated.

Results

Even in a wide open environment, i.e. nothing within 200cm, we never had our data rate drop below 11hz. The average cycle rate from our admittedly non-exhaustive testing was around 15hz. The maximum data rate we say was 40hz. We achieved this by setting an object within 10cm of each sensor. We don’t expect this to happen in the course but it did give us a nice upper bound for bandwidth calculations.

Graphs are missing because the data was stored on one of our teammates laptop which was destroyed due to a short caused by exposed wiring one day while working on the robot.

Movement

Two test types were performed to show the mechanical ability of our vehicle. The first was the speed test of our vehicle.

Speed Test

Testing Objective

Test that the robot can maintain a speed less than 5 MPH

Testing Protocols

1. Provide measured distance
2. Provide acceleration distance above seven inches
3. Provide and initiate robot speed command
4. Clock times as vehicle crosses distance

Instrumentation

Stop watch, Measuring Tape, Video Camera

Data Acquisition/Processing

Time was collected as the vehicle left the first spot and arrived at the second spot. This time was divided by the distance traveled to give the average speed of the bot over the distance.

Testing Results and Validation

	Distance (ft)	Time	MPH
Test 1	15	2.36	4.3
Test 2	15	2.46	4.2
Test 3	40	6	4.5
Test 4	40	6.02	4.5
Test 5	40	6.4	4.3

This test shows that the vehicle is able to maintain a speed below 5 MPH consistently.

Incline Test

Testing Objective

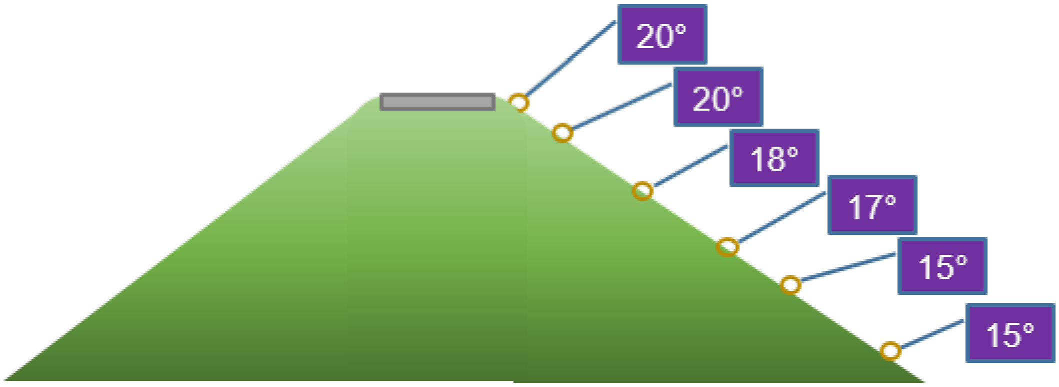
Verify the safety and ability of our vehicle to climb an incline of equal to or greater than 13.5 degrees.

Testing Protocols

1. Run vehicle on measured incline.
2. Adjust the center of mass by adding increments of weights to the front of the vehicle.
3. Test performance uphill and downhill incline

Levee Conditions:

- 15-20° slope
- Dry Grass
- Minimal Loose Grass
- Medium-Low Cut



Instrumentation

Weights, rope, camera, controller, hill, protractor

Data Acquisition/Processing

As the center of mass was adjusted using 10 LB weights to the front of the vehicle, the bucking of the vehicle was watched until failure occurred. The failure point was noted in the results file.

Testing Results and Validation

Predicted Required COM for 15-20 Degree Incline		Test #	COM (in)	Incline 15-20 Degrees	Bucking Performance
4.5 - 4.9 inch		Test Result 1	7.16	Uphill	Pass
		Test Result 2	7.16	Downhill	Pass
		Test Result 3	7.16	Uphill	Pass
		Test Result 4	7.16	Downhill	Pass
		Test Result 5	6.11	Uphill	Pass
		Test Result 6	6.11	Downhill	Pass
		Test Result 7	5.17	Uphill	Pass
		Test Result 8	5.17	Downhill	Pass
		Test Result 9	4.74	Uphill	Pass
		Test Result 10	4.74	Downhill	Pass
		Test Result 11	3.29	Uphill	Pass
		Test Result 12	3.29	Downhill	Buck
		Test Result 13	3.29	Uphill	Pass
		Test Result 14	3.29	Downhill	Buck

This test shows the center of mass of 7.2 inches from the axis or rations is safe for inclines under 20 degrees.

Safety

Data Acquisition & Processing

Ultrasonic Safety

The ultrasonic sensors we are using emit a burst at 40KHz which is well above the range of human hearing. Consulting OSHA guidelines we are allowed no more than an 115dB of output and since it is above hearing there is no weighted average for 8 hours given. The Parallax Ping))) outputs a 40KHz burst for no longer than 200us since we do not plan on modifying the Parallax Ping))) is any way it will be completely safe for the unprotected human ear.

Battery Overcharging

The battery is protected with a protective circuit module for over charge, over discharge, short circuit protection and balance function. The battery's charging max support current is 16A.

The balance function will only take place during charging without a load connected it. If a load is connected, no balancing will take place and the duration can vary from a minute to 30 minutes to several hours depending on the difference of voltage between cells.

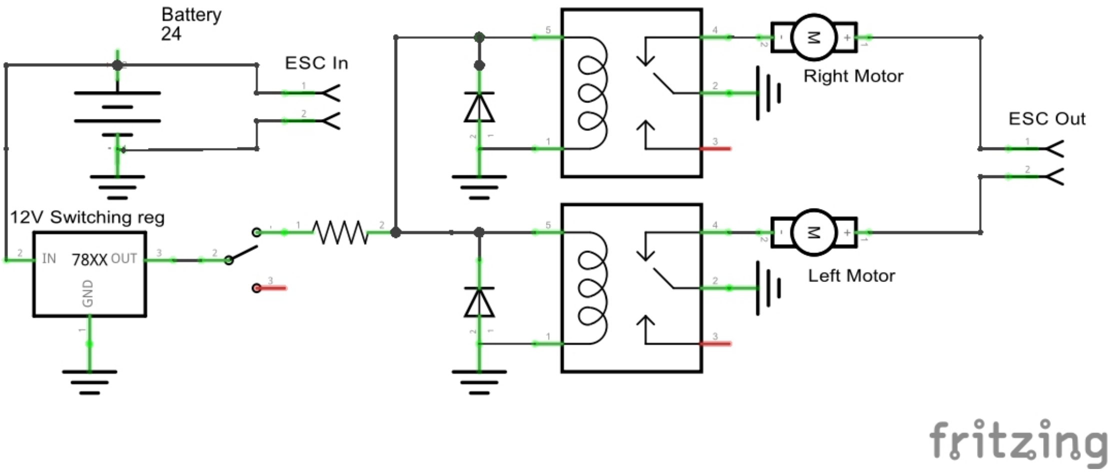
Electrical Hazards

According to OSHA, the main causes of Electrocution Fatalities that can apply to our project are due to contact with live circuits, poorly maintained extension cords, and defective power tools. Electrical Injuries are divided into two categories: Direct and Indirect. Direct electrical injuries include electrocution, electrical shock, and burns. Indirect electrical injuries include falls and fires.

The severity of the shock depends on the path, amount of current and duration of exposure to the body. OSHA requires special training while working on electrical equipment. This training teaches: safe work practices, isolation of electrical sources, test equipment, tools and PPE.

One preventative measure that should be taken with electrical circuits is releasing stored energy within capacitors and then testing them to see if they have been released of that energy. Cords should be visibly inspected before use for signs of fraying wires. They should not be placed in a high traffic area.

Safety Kill Switch



The contest rules require a wireless kill switch and a hard wired kill switch that can immediately immobilize the vehicle. These kill switches must not be programmed and are hardware based. The wired kill switch will be placed near the rear center of the vehicle with a 2” diameter. The wireless kill switch will need to be operated at a minimum range of 100 feet and has been designed to switch two 60A relays that are hardwired to the motors

Hazards while printing with ABS and Safe Soldering Practice

When printing with ABS plastic and soldering, precautions and safety measures should be taken in advance to prevent injury and exposure to fumes. The hazards one can encounter while printing with ABS or soldering include but are not limited to; exposure to fumes that can irritate the respiratory tract, skin, or eyes and cause headaches. Molten plastic/solder can also cause major burns.

There are preventative safety measures that are recommended by OSHA, IARC, NTP and ACGIA while handling ABS and soldering. An engineering control would be to print or solder in a well ventilated area. Safety glasses are recommended while printing or soldering to prevent the eyes from exposure to fumes and molten plastic. Heat resistant clothes and shoes are recommended to prevent burns. If printing in an area that could potentially expose oneself to high concentrations of the fumes, a NIOSH/MSHA approved air purifying filter is recommended.

There are also some first aid measures to take if one were to injure themselves while printing with ABS or soldering. It is recommended that one should flush their eyes with water continuously for 15 minutes if exposed to ABS or solder. Thermal burns should be exposed to cool water and medical attention should be sought out if burns are extreme. There are no known effects on ingestion of ABS plastic. One should move to a source of fresh air if exposed to fumes.

Movement

When looking into keeping the team and others that come into contact the vehicle safe, we need to exercise caution during the manufacturing, testing, and assembly phase of our vehicle. Due to some of the constraints set forth by IGVC, we will be required to exercise certain safety precautions. These include the mounted and wireless emergency stop button, along with the flashing LED lights that indicate when the vehicle is powered on or in autonomous mode. Below is a Risk Priority Number analysis of the different parts of our design that we believe should be made a priority when detecting issues. This analysis consists of determining the chance of that part having issues, the severity of those issues, and the ease of detecting the issues through observation. The part that we found to be most vital in the overall failing of our design is the batteries, totaling a score of 128. This coincides with our hypothesis since the battery is the power source for almost every other component on the vehicle.

RPN				
Part	**Severity**	**Occurrence**	**Detection**	**Total**
Frame	7	2	1	14
Motor	8	2	3	48
Tires	5	1	2	10
Sensors	6	4	4	96
Batteries	8	4	4	128
Wires	4	2	6	48
Welds	7	3	3	63
Laptop	4	4	4	64
Gears	6	2	7	84

While the vehicle itself has its' own safety precautions, we must also take steps in keeping ourselves safe during the manufacturing phase. One method of manufacturing that we will be using is welding. In order to maintain safety protocols, we will be following OSHA standard 1910.132 which highlights the protection of the eyes, the face, the hands, and head protection of the welder. We will be using OSHA approved gear such as facemasks, heat resistant gloves, and full welding coats to protect our skin from zsparks. Another OSHA standard that we will be considering for safety is OSHA standard 1910.303 which highlights electrical and wiring safety. We will follow these protocols in order to prevent harm during manufacturing and assembly.

There are preventative safety measures that are recommended by OSHA, IARC, NTP and ACGIA while handling ABS. An engineering control would be to print in a well ventilated area. Safety glasses are recommended while printing to prevent the eyes from exposure to fumes and molten plastic. Heat resistant clothes and shoes are recommended to prevent burns. If printing in an area that could potentially expose oneself to high concentrations of the fumes, a NIOSH/MSHA approved air purifying