Assignment: Black Box

Team Name: LOST

Course: ENES100 - 0301

Submission Date: 12/14/18

Table of Contents

Approvals ……... 2
Executive Summary ……... 3
Introduction …… 3
Preliminary Design Shortcomings..…… 5
Final Design Details ..……. 6
Final Design Drawings ……..…….…... 6
Construction Details ……. 15
Final Bill of Materials …….. 16
Project Management ……..…….. 17
Disposal Plan ……... 18
Product Performance and Evaluation …… 18
Lessons Learned ……..… 20
Appendix ……. 23
Calculations…….…….... 23
Minutes ……..... 25
Gantt Chart ……..... 29
Final Code……..…….…… 30

Approvals

Shahed Bader

Preliminary Design Shortcomings, Final Design Drawings

Signature: _____

Atticus Cameron

Final Design Drawings Signature: _____

Anjali Dhamsania

Disposal Plan, Meeting Minutes, Final Design Details

Signature: _____

Bahaa Harraz

Final Design Details, Final Code Signature: _____

Nicole Kwok

Product Performance and Evaluation, Lessons Learned

Signature:_____

Vladimir Leung

Final Design Details, Final Code, Final Design Drawings

Signature:_____

William Mah

Final BOM, Project Management, Final Design Details

Signature:_____

Nicholas Ruei

Construction Details, Final Design Details Signature:_____

Julia Zhiteneva

Executive Summary, Introduction Signature:_____

Executive Summary

Our team was tasked with creating an over-sand vehicle (OSV) that can locate and retrieve an aircraft's Black Box from an undisclosed location within an obstacle area. The OSV and everything on it is required to cost no more than $350, have a mass of no more than 2.50 kilograms, fit within a 300 mm x 300 mm footprint, and run all systems at full power for at least 10 minutes without recharging. The base of the vehicle is a sturdy, rigid sheet of plywood. Attached to the chassis are four motors connected to rubber tires with reasonably deep treads. On top of the OSV are two batteries; one for the servos, and one for the rest of the electrical components (an Arduino Romeo microcontroller, three ultrasonic distance sensors, an IR sensor, and our mechanism to pick up the Black Box). The OSV is designed to maneuver to the center of the four quadrants, and use its IR sensor to detect the Black Box's infrared 50 ms pulse, four per second, at 38.0 kHz. Upon locating the Black Box, it will transmit its coordinates and then use its ultrasonic distance sensors to position itself in front the Black Box with the arms under it. The mechanism to pick up the Black Box is composed of four 3D printed arms, and four servos (two for each arm). The OSV will then move towards the box with its arms lowered, bring the second pair of arms closer together until secured firmly underneath the lip of the box, and lift. It will then maneuver itself back towards the original landing site, with the box in tow, to complete all mission objectives.

Introduction

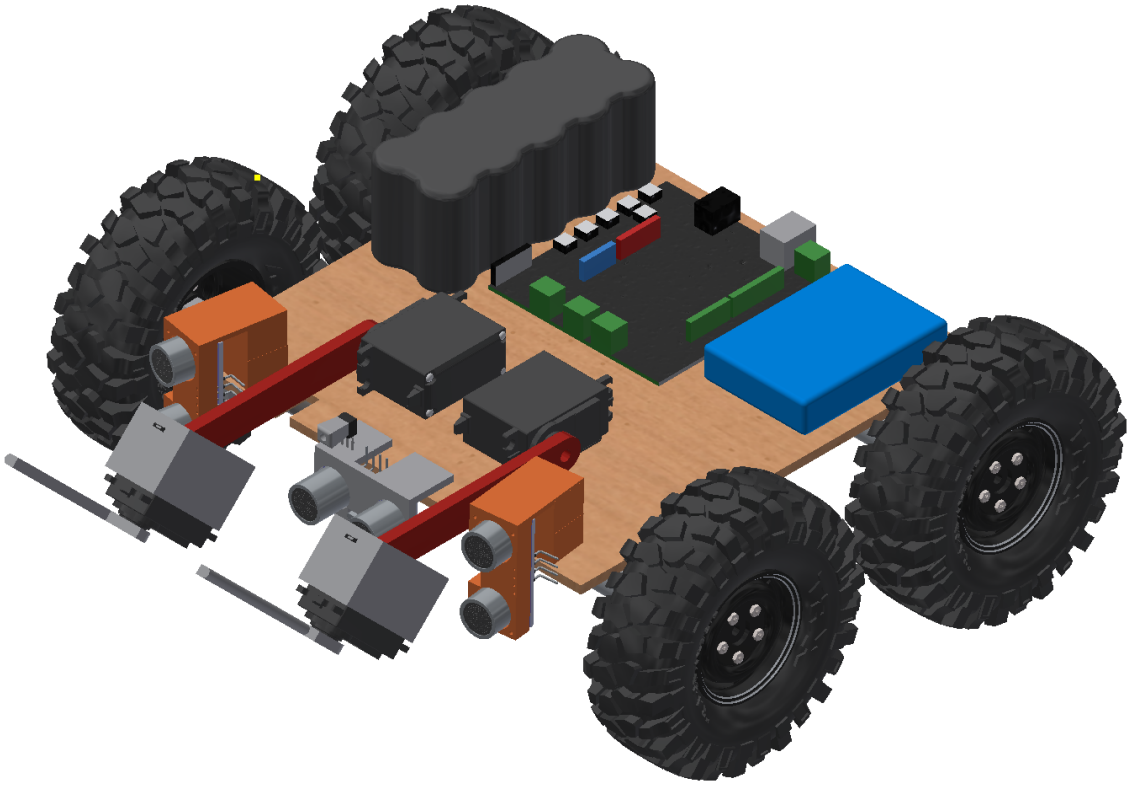
The goal of our engineering firm was to design an over-sand vehicle that could search for and transmit the coordinates of a Black Box on a "remote island." The vehicle needed to navigate autonomously from its starting point at the landing zone towards the Black Box. It needed to do so by traveling to the center of the four quadrants of the arena while avoiding obstacles with the help of its ultrasonic distance sensors. Upon reaching its destination, it would utilize its infrared sensors to detect the 50 ms pulse 38 kHz that the Black Box emits. Upon detection of the signal, the OSV would navigate towards the Black Box and transmit its coordinates to the command center, using its APC220 Radio Communication Module. In addition to its basic objectives, it will complete the advanced objectives of picking up the Black

Box and bringing it back to the landing zone. All of the objectives are to be completed within five minutes of starting the mission.

The design of our OSV is limited by a variety of constraints. The mass of the OSV cannot be more than 2.5 kg and the car itself must be able to fit in a 300 x 300 mm footprint. It cannot have purchased/pre-built assemblies that perform more than two vehicle functions, and cannot cost more than $350. The OSV cannot be powered by lithium or lead acid batteries, and the battery is expected to last at least 10 minutes at full power without recharging. Our OSV is a 4x4 wheel drive vehicle, with four drive motors connected to each of the four wheels. For the advanced mission, we constructed a mechanism composed of servos and 3D printed arms that, when in close enough contact to the Black Box, will lower the arms until lined up underneath the lip of the box, then move inwards, pressing against the sides of the box until the box is secured firmly.

The OSV must travel on sand, have motors that are powerful enough to provide the necessary torque to propel through rocky terrain, and navigate around obstacles. The tires we decided to use have deep treads which prevent the OSV from digging into the sand. For the motors, we determined appropriate calculations of torque and tractive effort to purchase a motor with a torque that fits in our desired range. For navigation, we used multiple ultrasonic sensors to detect obstacles randomly scattered around the arena. We used an IR sensor with a funnel to identify the location of the Black Box. After transmitting its coordinates using the APC220, the OSV would then use its arms to press against and pick up the Black Box, and then navigate back to the landing site.

The OSV faced a few shortcomings in terms of completing the advanced objectives. Due to the broad range of error with the pick-up mechanism, issues with constructing the mechanism itself, and the difficulty in writing a code for the pick-up mechanism to properly lift the box, our OSV was unable to pick up the Black Box from the ground and bring it back to the landing zone. Aside from that, the OSV was able to navigate the terrain, but was unable to avoid obstacles, due to problems in the code. The infrared detection of the Black Box did work, but we were unable to test it, because the vehicle could not pass the obstacles to get within a close enough range of the infrared signal.



Preliminary Design Shortcomings

Throughout the course of this semester, our team ran into some unexpected design shortcomings that set us back from completing the necessary tasks in a timely fashion. Some of our issues revolved around 3D printing. Our original design to pick up the Black Box was to 3D print arms that would attach to the servo and serve as the picking up mechanism. After much trial and error, we were not able to successfully 3D print arms that were able to complete this task effectively.

We also intended to 3D print the motor housing however it proved difficult to attain the perfect size. In order to resolve this issue we bought motor housing for all of the motors, which in turn took away from the budget that could have been used for other necessities. Another shortcoming was not having enough pins in the Romeo microcontroller to wire all of our servos. We were forced to resolve this issue by using an Arduino alongside the Romeo microcontroller to serve as an extra source of pins. One of the more challenging parts of this was being able to write code that would allow the Romeo to communicate with the Arduino.

We also originally intended the infrared sensor to be attached to a column without anything to hinder it, however we came to a realization that the sensor was picking up the frequency of the Black Box too early and the OSV would move at the wrong time, since the sensor had somewhat of a wide range. In order to narrow the range we made a funnel out of cardboard and placed it over the IR sensor so that it would only detect the frequency of the Black Box once it was directly in front of it. Finally, the last design shortcoming that our team ran into was the balance of the OSV.

Final Design Details

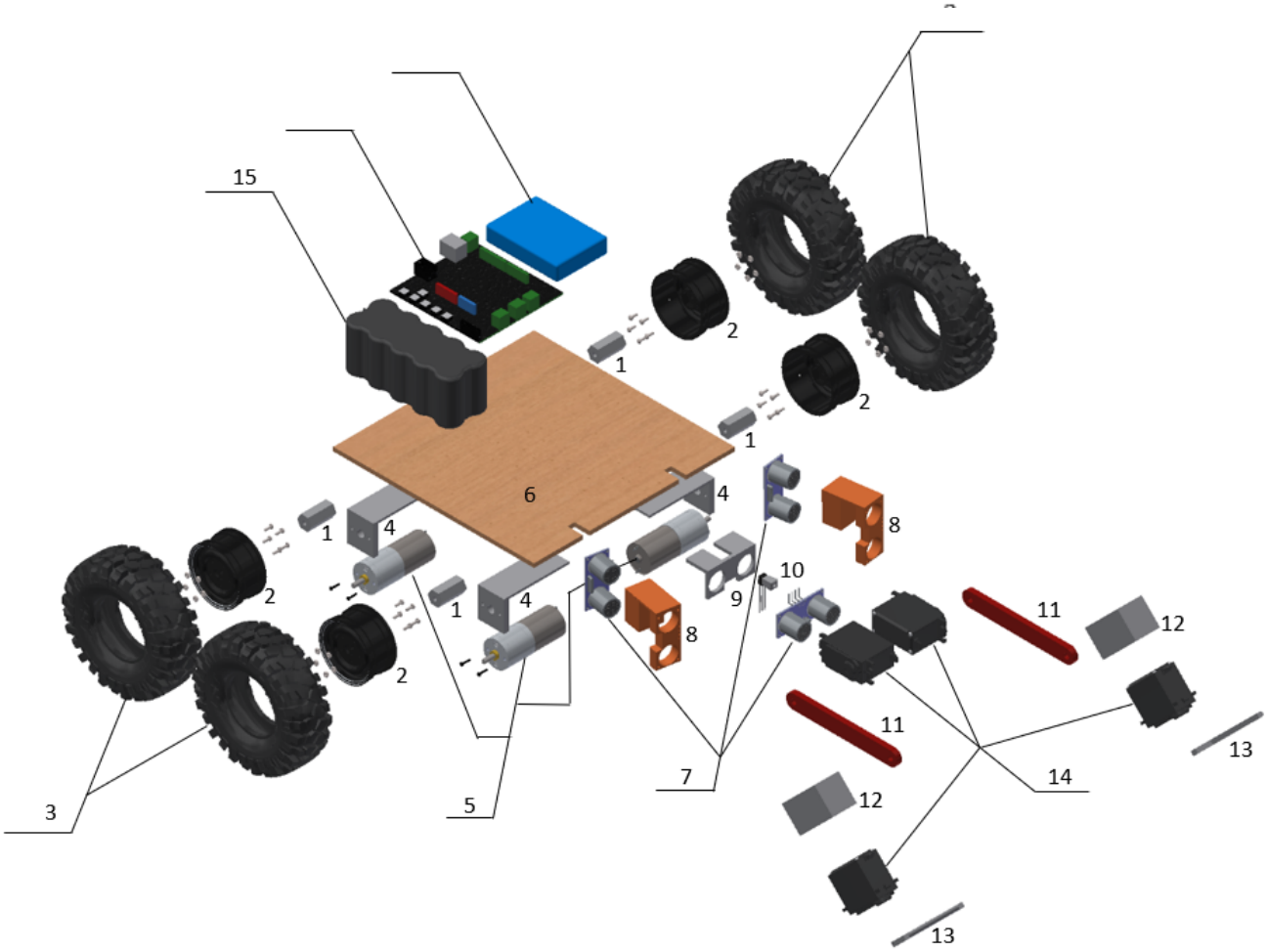
Structure

The OSV chassis is made out of a 210 x 210 mm piece of plywood that is 6.35 mm thick. This plywood was cut using a table saw during the first week of construction. We had originally wanted to use fiberglass, but opted for plywood due to its sturdiness and the difficulty we would have encountered in cutting the fiberglass. Other non-mission critical components on our OSV included a pencil sharpener that we fashioned to be a tower to mount the IR sensor onto, elevated platforms to attach the sensor mounts and servos, and a funnel which was intended to restrict the field of the view of the IR sensor so we could more accurately detect the Black Box within a few degrees of its actual position. We constructed platforms by using a handsaw to cut up rectangular blocks out of our leftover plywood, stacking them to reach a height of 19.05 mm for our servos and 12.7 mm for our ultrasonic sensors. The funnel was constructed out of hard cardboard by cutting it into pieces using scissors and gluing it back together.

To secure our Romeo microcontroller onto the chassis, we drilled a hole in the chassis and held it in place with multiple zip ties. Our motor and servo batteries, breadboard, and tower were held onto the chassis using duct tape. We also used hot glue to affix the ultrasonic sensor mounts, the wooden platforms propping them up, and the four motor mounts underneath. For extra reinforcement, after screwing the motors into the motor mounts we also drilled an additional hole near each motor so that we could zip tie them. All other components of the OSV were affixed to the chassis indirectly, including the motors, IR and ultrasonic sensors. The final

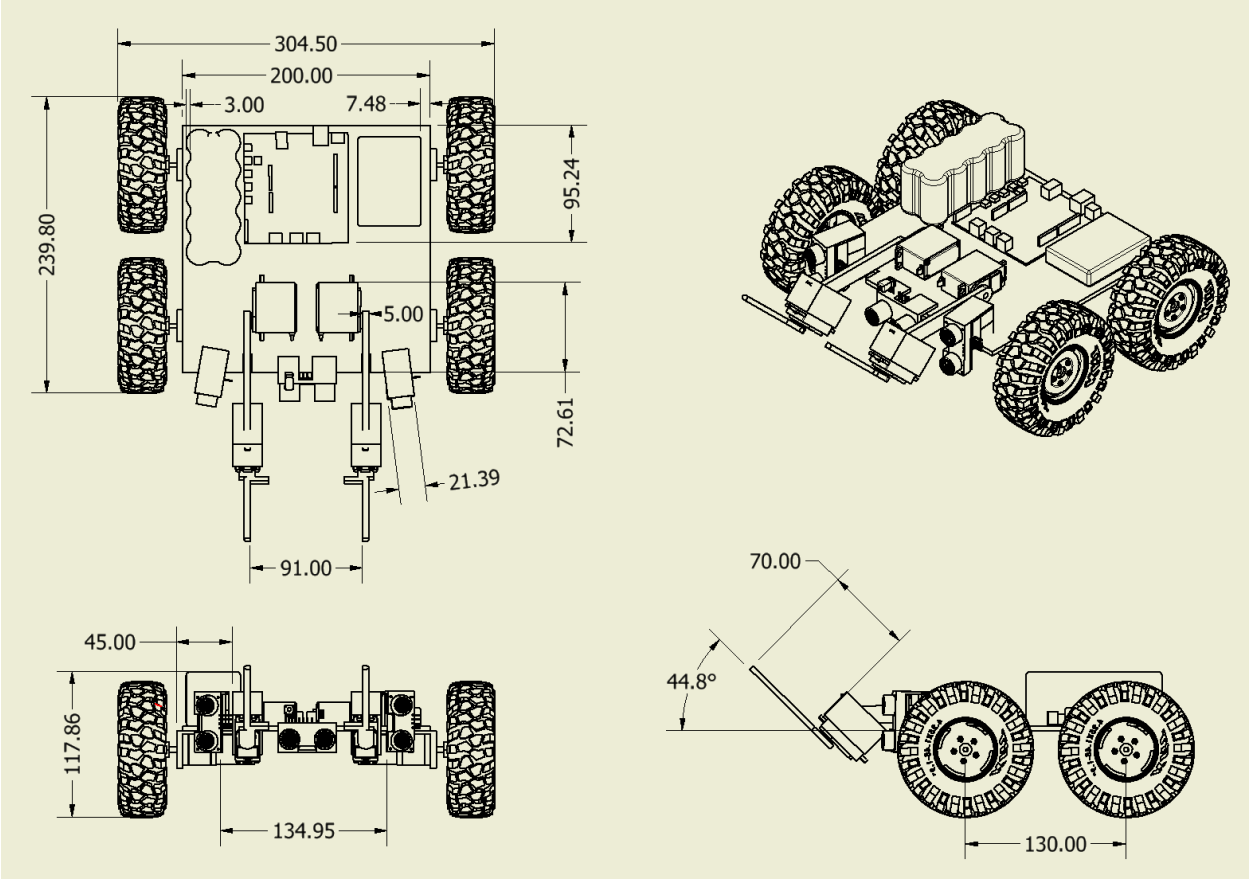
weight of our OSV with components attached was 1.7446 kg (see Bill of Materials for individual masses).

Exploded View



1	Motor Shaft to Wheel Hex Adapters	10	Infrared Sensor and Funnel
2	Tire Mounts	11	Servo Arms
3	Tires	12	Servo Holders
4	Motor Mounts	13	Clamping Arms
5	Motors	14	Servos
6	Base/Chassis	15	Main Battery
7	Ultrasonic Sensors	16	Romeo Microcontroller
8	Vertical Ultrasonic Sensor Holders	17	Servo Battery
9	Horizontal Ultrasonic Sensor Holder		

OSV Dimensions



Propulsion

Our team opted for malleable tires with deep treads to go over the rocky terrain and through the arena's sand. The selected tires had a diameter of 100 mm and a width of 36 mm. We then calculated a Tractive Effort (TE) range needed to overcome the resisting force of the sand, while not applying too much force so that the wheels begin to slip on flat ground ($2.35N < TE < 4.29N$). We then calculated the TE range at 35° as this is the resting angle for sand($3.66N < TE < 5.97N$). We do not expect the OSV to move along a 35° slope (as the sand will be pushed down from that angle before the wheel rises to it), but this means that every angle from 0° to 35° will be operable. From these ranges we combine the bounds creating the smallest range ($3.66N < TE < 4.29N$) [see Appendix for further calculations].

Moving away from motor calculations, we decided on a conservative distance and time that the OSV would take to travel (15m in 180s, or 60% of the mission time requirement). From this we obtained a desired linear velocity and angular velocity of 8.3 cm/s and 1.66 rad/s. From this, we obtained an acceleration to create a desired Tractive Effort, which fit inside our range ($3.73\ \text{N} * 5\text{cm} = 18.6\text{Ncm}$). Our mission desires a 1.66 rad/s angular velocity, so we required a motor with a high torque and a low no-load speed. We decided on a gear motor with a stall torque of 290 oz-in. As we built the OSV, we opted to place the motors on the underside of the plywood base, held up using motor brackets. We then connected the motors to hex wheel adapters that attached to the tires, which formed the basis of our propulsion system. This set up the edge of each wheel roughly 5mm from the plywood, giving ample distance for the wheel to spin while also maintaining a compact design.

The OSV's top speed varied based on the amount of charge left in the battery, but on average it travelled at $.036m\ x\ 36\ RPM\ x\ \frac{2\pi}{60sec} = .1357m/s$. Despite a relatively low velocity, our OSV was easily able to traverse the rocky terrain and travel across the arena with sufficient time remaining to carry out the mission. To steer our OSV, we mainly relied on our programming implementation to turn the OSV by turning either the right or left side of wheels clockwise, while turning the other side counterclockwise. This turned the OSV in place, and allowed us to gradually adjust for changing sensor data because of the delay implemented in between each turn call. Our speed and turning mechanism proved to be fairly robust as we successfully completed Milestone 5, which tested the OSV's ability to move to the opposite side of the arena and turn ninety degrees. Our propulsion system was also more than capable of completing Milestone 6, successfully fulfilling the second objective of navigating to the center, scanning for the infrared signal of the Black Box, and moving toward it. We were unable to complete the obstacle avoidance objective due to a failure of the navigation code, not our propulsion mechanism. In keeping with this theme, our propulsion system successfully got us over the rocky terrain and towards the center of the arena for Milestone 7.

OSV Mission

The objective for our engineering firm is to design an over sand vehicle that can search and transmit the coordinates of an airplane Black Box from an unknown location on a "remote island." Once landed on the Landing Zone (LZ), the vehicle is designed to travel to the center of the given four quadrants. It will utilize its IR sensor and ultrasonic distance sensors to navigate to within 250 mm of the Black Box, which will be emitting an infrared pulse of 50 ms at 38.0 kHz. When the OSV arrives at the Black Box location, it will measure and transmit the coordinates of the box to the command center, using its APC220 Radio Communication Module. In addition, as a part of its advanced objectives, the OSV is to retrieve the Black Box by lifting it up using its 3D printed arms and deliver it back to the given landing zone coordinates. Both the basic and advanced objectives are required to be completed within 5 minutes as soon as the vehicle lands on the island. While traveling to the mission site, the OSV is also expected to navigate through rocky terrain, as well as avoid various obstacles using the ultrasonic sensors mounted on the vehicle.

Our strategy for the OSV mission utilized four servos as mission specific actuators. We printed out two pairs of two arms, two of which moved along a vertical axis and two which moved horizontally. In order to pick up the Black Box, we would navigate to the center of the arena and begin turning 360 degrees to pinpoint the location of the box using the IR sensor mounted to the OSV. Once the box was found, we would travel within 80 mm of it. At this point, the servos would lower the vertical arms, allowing the L-shaped horizontal arms to grip under the lip of the box regardless of orientation. Then, the vertical arms would lift up and the OSV would begin moving backwards, returning the Black Box to the landing zone.

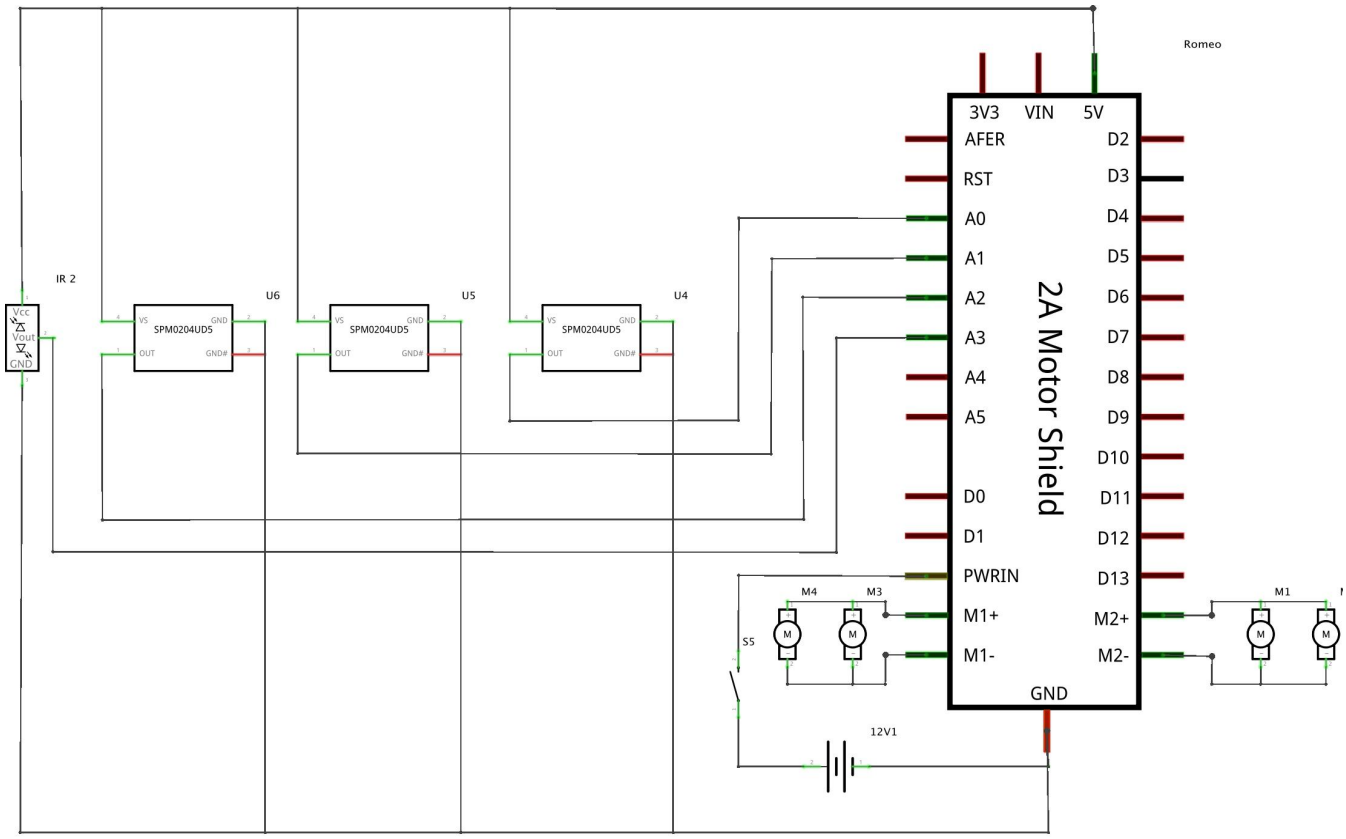
In practice, our OSV was not able to navigate to the center of the arena in order to begin our process for completing the Black Box objectives. This occurred due to an issue with our obstacle detection code. As a result, in our final runs, we were unable to test the ability of the OSV to pick up the Black Box.

Power

Power for the Romeo V2 microcontroller, the 4 motors, and all of the sensors were supplied through a 12V 2.8A rechargeable 3000mAh NiMH battery pack. The 4 servos had a

seperate power source, a 6V NiMH 2200mAh with a continuous discharge of 2.2A, connected to the Arduino as seen in Figure 3. As for our propulsion system, each wheel was powered by a DC motor (391:1 Gear Ratio, 36 rpm no load, 90mA no load current, 290 oz*in stall torque, 1.6A Stall Current). As shown in Figure 2, our team decided that it would be best to place each *pair* of motors in parallel with each other so that they operate at the same voltage and share the current. This also allowed for each of the side wheels to rotate in opposite directions, creating a mechanism for the OSV to turn. At 12V, the current draw of each motor was 161 mA, representing a current draw of 644 mA in total for all of the drive wheels. The sensors' max current draw of 200 mA in supplement to our motors also created a total current draw of 844 mA. We opted for a 2.8 A continuous and 5.6 A burst current draw, so that if three of our motors stalled out, the battery would not be fried. According to the data sheet of the battery we chose for our propulsions system, the real total capacity of the battery is 2948.4 mAh. This indicates that the OSV should be able to run for roughly 3.5 hours without recharging, which typically covered four or five typical test sessions in the lab.. However, this does not account for the areas of rocky terrain that the OSV would have to overcome which would result in a lower run time since the motors would need extra power to achieve the same torque. As for our mission specific actuators, the servo motors we chose drew 1048 mA for our two lifting servo motors, and 150 mA for our two gripping servos for a total current draw of roughly 1.34 A. The battery chosen for the servo continuously discharges 2.2A, which far exceeds our max power draw. Each input pin on the servo was intended to be directly connected to one of the PWM digital output pins.

Full Circuit Diagram



In practice, we did not complete the wiring of the servos or make use of the second battery on the OSV. Due to the relative brittleness of our 3D printed servo arms and a need to hurry construction due to time constraints, the arms did not stay attached on the OSV during our testing. Since Milestone 7 was run without a working servo mechanism, we were not able to test whether our power system would work correctly for the mission-specific actuators. However, the main power system worked to a satisfactory level. While low battery resulted in an obvious degradation in motor performance, when the battery was fully charged it delivered power to the motors as expected and typically went several lab sessions without needing a recharge.

Control Algorithm

The OSV is to be dropped off at the landing zone and begin to turn until it is facing the right ($\theta = 0$ radians). To take into account the vision system's location updating speed, there are

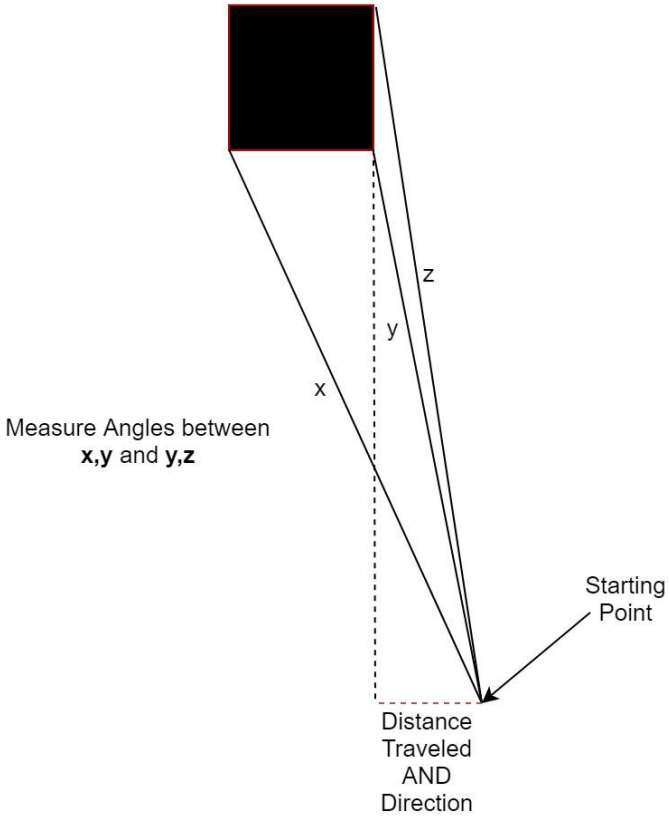
delays of .075 seconds and a tolerance of .07 radians built in. Once the OSV is pointing to the right, it will then begin moving until passed the rocky terrain. Once passed, the OSV will move forward with object detection enabled.

If an object is seen closer than fifteen centimeters away and it is above the center of the arena (y position > 1) then the OSV will turn down (θ = -1.57 radians), move forward for .05 seconds and turn back to the right and begin moving forward. If the OSV is below the center of the arena (y position < 1) then it will turn up (θ = 1.57 radians), move forward for .05 seconds and turn back to the right and begin moving forward. If the OSV has already moved up or down to move away from an object and still detects one when it turns back right, the OSV will turn the opposite way it just turned, move until .5m away from the wall turn to the right and attempt to move forward. If an object is still detected, the OSV will execute its normal object detection but move until it is three-quarters the way down or up the arena (y position = 1.5 or .5). This restarts the object detection in the other of the arena in case one side is unpassable.

Once the OSV has reached the desired x position (x = 1), the center of the arena, the OSV will move downward until reaching the center. Once at the center, the OSV will turn clockwise until the IR sensor detects the Black Box's IR signal, and will move forward until it is not detecting the IR signal, where it will begin to spin clockwise again. When an object is detected at 15 cm away, the OSV will stop and it will transmit the Black Box's location.

To orient to the Black Box, the OSV is designed back up for 20cm, turn clockwise until the center US sensor senses nothing, and turn in .05 radians counterclockwise increments to read how far away the Black Box is. After storing the right-most bound (z), the smallest distance (y), and the left-most bound (x), and the angles between each of the measurements, the OSV compares the two angles to find the smallest angle to determine if x = z, if x > z or if x < z. Then the OSV will calculate the angle needed to turn with the equation **1** (please refer to the Appendix) and calculate the distance needed to travel with equation **2** or if x < z then calculate the angle to turn with the equation **3** and calculate the distance to travel with equation **4**. Then move this distance and turn 90 degrees toward the Black Box and start the orienting to Black Box algorithm again, but if x does equal z then check if equation **5** and equation **6** are true. If so

use this as the output, but if it does not then use the x or z equation and follow the same steps as before.



To pick up the Black Box, the OSV will lower its arms to parallel to the sand, open the rotating arms, then move until 12 cm away from the Black Box. The arms will close and read back if it is encountering resistance as it closes on the Black Box. If so, the arms will stop trying to close, open slightly, then lower down .15 radians, and try again. Once the arms are parallel without any resistance, the arms will lift the box to .785 radians.

To return to the landing zone the OSV will then turn to the left (θ = -3.14 radians) and move forward with the same object detection mentioned above until it reaches the landing zone (x position < .3).

Construction Details

OSV Construction began shortly after the completion of MS3. Almost immediately, we decided to scrap our original fiberglass base due to its weight and the lack of tools to cut it, in favor of a more traditional 210mm x 210mm piece of plywood, which we sawed off of a larger plank. We also decided to forego the idea of gearboxes as they would be too difficult to set up in comparison to simply attaching the motors to the base of the vehicle. After testing individual motors to make sure that they would be able to turn the wheel once connected to the hex wheel adapters, we attached them to the bottom of the OSV. We first utilized hot glue to secure the gear bracket motor mounts which were bought online, and then screwed the motors into the mounts, using zip ties as additional reinforcements. After this, we taped our battery and breadboard onto the OSV, and began wiring the motors to the Romeo microcontroller which we temporarily left on top of the OSV. We then drilled a hole into the center of the base and secure the Romeo to it with zip ties before threading our wires through it. We then soldered the wires to the motors and cut them so that they could be wired into the breadboard.

Following this initial flurry of construction, we began slowing down, designing a 3D print for our ultrasonic sensor mounts as well beginning to design our two pairs of servo arms. After a few test prints of the sensor mounts, we were able to fit them onto the ultrasonic sensors with some minor filing. We then sawed off small portions of spare wood to create an elevated platform on the front of the OSV, on which we placed our sensors. Once again, we threaded the wires connecting the sensors to the Romeo through the center hole and this time, used electrical tape to further organize the wires running underneath the base board. With our ultrasonic sensors completed, we moved on to the infrared sensor, using duct tape on the front of the OSV to elevate it even further above the ultrasonic sensors. We also designed a funnel out of cardboard to reduce the field of view of the IR sensor for optimal precision. Afterwards, we wired up the IR sensor as well and again threaded it through the center hole in the base. From this point on, most of our efforts went towards programming while we continued working on the servo arm design.

Unfortunately, we ran into numerous 3D printing issues while creating the servo arms. The first pair of arms needed to be able to hold a second set of servos, so we designed a

box-shaped servo mount that would hold the second set. In order for that box to be parallel to the ground, we were forced to print the first set of arms at a forty-five degree angle relative to the second set of arms. Because of the way the majority of the arm was angled relative to the servo box at the end of the arm, we needed support structures to print the box. Due to issues with the extruder, filament, and automated support structure generation, the final set of servo arms took almost ten attempts to produce. After they were printed, we ran into a second issue. The screws which were supposed to attach the servo arms to the servo did not grip into the servo grooves well, and resulted in a very loose connection. With very little time left in the semester, we decided to hot glue our servo arms onto the servos themselves, using duct tape as a reinforcement mechanism. Afterwards, we cut two indentations in the front of the OSV such that (if projected forward) it would be aligned with the outside edge of the Black Box. Realizing the servos didn't quite align as anticipated, we again sawed a spare piece of wood to create raised platforms for the first set of servos to sit on. We then tried to attach the servo arms to them, letting them hang through the indentations such that the second set of servo arms were parallel to the ground. However, at this point we had no time left and needed to run Milestone 7. Rather than attempt it with barebones code and lacking circuitry, we opted simply not to attach the arms for the final run. This marked the completion of construction.

Final Bill of Materials

Items	Cost	Manufacturer	Description	Model Number	Mass (kg)
Wheels (4x)	$ 24.43	RC-Hub	Wheels for OSV		0.2
Infrared Sensor	$ 7.49	Vishay	Detects Black Box frequency	TSOP38238	0.005
Romeo Microcontroller	$ 31.99	DF Robot	Programmable microcontroller with h-bridges	DFR0004	0.0816
Ultrasonic Sensor	$ 16.14	SainSmart	Distance sensor	HC-SRO4	0.027
Battery (12V)	$ 24.99	Tetrix	Battery for motors/sensors	W39057	0.607
Forklift Servos (4x)	$ 31.80	RobotShop, Pololu Robotics	Regulates movement of forklift	1711MG	0.080
Forklift Arms	$ 8.00	3D printing	Enables forklift to pick up Black Box		0.05

(2x)					
Battery (6V)	$ 15.15	RobotShop	Battery for servos	RB-Pol-160	0.1410
Motors	$ 91.80	Pololu Robotics	290 oz*in stall torque, 1.6A Stall Current		0.1880
Wiring & Housing Units	$ 10.00	3D printing	Covers for wires		0.15
Wood Base Plywood	$ 10.48	Midwest Products	Wooden base for OSV		0.161
Hex Wheel Adapters	$ 7.90	Pololu Robotics	Screws to attach things to base		0.02
Gearmotor Bracket Pair	$ 14.90	Pololu Robotics	Mounting for gearmotors		0.034
Total:	$ 295.07				1.7446

Project Management

In the design phase of the semester, the team envisioned that it would take a total of three weeks to construct the major aspects of the OSV (See Appendix for Gantt Chart). These included the base and layout, the propulsion system, and the sensors, which were intended to be completed sequentially. The team initially planned for this to occur between October 15th and November 2nd. These advances would be made simultaneously with the coding. After the planned three week period of construction, the team hoped to test and refine the OSV for the remainder of the semester.

However, this plan was not adhered to. The major deviation from the plan was the division of labor. Once the team began working, we split up into different groups that focused on different aspects of the project. Every component that we planned out was intended to be worked on simultaneously, but it was rare that this actually occurred. Due to a knowledge gap between different members of the team and the unfortunate reality of multiple team members being sick at the same time, our team was forced to wait to complete one task before starting another. This resulted in an inefficient use of time, as there would typically be three or four members sick or simply focusing on other classes at any given time when we could have been completing two tasks simultaneously. Additionally, we failed to take into account that certain tasks would need to build on top of each other. For example, it took us much longer to figure out

how to get the propulsion system working properly once we started implementing the sensors. This is was because the use of the digital pins interfered with the Romeo's integrated h-bridges to function properly. We also had a week go by with no progress, as most of the group took time off while the people working on coding tried to fix our navigation, rather than beginning the next stage of construction.

As time went on, the team also began to become more lax in regards to weekly meetings, as team members became occupied with other classes and responsibilities. This began to create a culture where arriving late or simply not showing up at all was acceptable. While these minor absences likely did not have much of a tangible impact on the project as a whole, the environment that resulted from it contributed it to our team lacking of sense of urgency in working towards completing milestones.

Disposal Plan

The team plans on disposing of the OSV and its component parts by donating it and recycling it. The Romeo microcontroller will be given freely to team member Anjali's father, who owns a circuit board manufacturing business. Extra unused wires will be given to friends of teammates, particularly students who will be taking ENES 100 in the spring semester. The batteries, motors and wheels will be given to be reused by team member Nick's uncle, who has a hobby of making RC cars and will repurpose the parts for his own use. The sensors and servos will be given to an incoming ENES 100 student who will be taking the course in the spring semester, and may reuse these sensors for their own OSV design. Used wires will be thrown away in the trash. The wooden base will be recycled.

Product Performance and Evaluation

The OSV did well to stay within the product specification of being under 2.5kg and managed to squeeze into the 300x300mm footprint checker. The vehicle fit snugly into the dimension requirements and weighed ~1.75kg, which is under the maximum mass by ~ .75kg. Our budget was limited to $350, according to the product specifications, but we only had to spend $305 to buy all of our materials, including 6V and 12V NiMH batteries, two kill switches,

and a Romeo (Arduino compatible microcontroller). All of the milestones were completed except for Milestone 6 & 7, when it came to actually completing the specific Black Box mission objectives and obstacle detection. The OSV was able cross the rocky terrain without any issues, was able to maneuver through the sandy terrain and detect for any obstacles in its way but was unable to go around said obstacles. It was able to transmit information and send its coordinates to the command center through the RF transmission, but did not reach this point in the mission due to the aforementioned lack of working obstacle detection .

Some of the possible changes that could have been implemented were simply changing the code, better planning of the wiring layout, and creating a simpler pick up mechanism. Due to the time constraint and a lack of communication between our main programmers, we did not have a single working piece of code on test day and instead attempted to cobble together a working obstacle detection program from two different sources of code. By planning ahead before the deadline and encouraging more meetings between the programmers, it may have been possible simply to create working code to begin with, removing the need for any other changes. Another issue that could have been dealt with easily had we had more forethought was our wiring for the servos. Due to a miscalculation regarding the function and number of pins on the Romeo, we found it necessary to add an Arduino just to run all the servos and motors together at the same time. However, because we did not foresee this early on, much of the wiring was unnecessarily difficult and had to be redone. While hooking the Arduino in ended up being a good idea to increase the amount of pins we had to work with, the implementation could have been much smoother. Most of the issues we had in relation to the pickup mechanism stemmed from the difficulty of 3D printing a significant amount of material, and getting it to work with pre-bought servos. Had we opted for a forklift or a less complex mechanism, it is more likely that we would have been able to have it ready for Milestone 7, and we would have had more time to spend on coding and other tasks that were necessary just to put the OSV in a position to utilize the pickup mechanism. These three mistakes were all easily remediable and could have helped us create a more reliable OSV in less time.

Lessons Learned

1. Modeling the OSV required the team to do a lot of calculation to determine what is needed for the OSV and how to build it. We determined the wheels we needed through calculating the required torque and figuring out the dimensions the wheel needed, as well as the type of treads that would be needed to run on the sand. It also helped us determine the battery by calculating the required current draw to run the motors, sensors, and servos. Basing the materials needed on math and physics concepts simplified the choice of materials that needed to be bought and ensured a high reliability that the OSV would run. Without doing the calculations before hand, we probably would have faced significant issues even getting the OSV to move, since all components would be selected based on intuition.
2. Developing a prototype allows the team to test whether it is worth continuing on with the current design. While the prototype is being built, major design flaws or a failure to adhere to the guideline requirements can quickly become apparent, and this allows you to change what you need to change before you make the final product. It gives you insight as to see where the prototype is going wrong so that the final product can avoid the same pitfalls. Because a prototype allows for testing of individual components, it becomes apparent which parts are functioning properly and may help to localize a particular issue before building the final model. This also helps avoid the temptation to tie the subsystems to each other too quickly, as it ensures that each individual component works before they are all put together.
3. Troubleshooting is a very important process in the development of the OSV. When writing code, it is difficult to account for all the edge cases that could result in errors, especially in the case of pathfinding for the OSV. Thanks to the simulator, we were able to run a prototype of the code without ever having the physical OSV run. Based on the miscues in the simulator run, we were able to adjust variables so that the real OSV was able to perform better. Another way to test the different parts to the OSV was by individually running the code with the four sensors. To see if the infrared sensor was working properly, a test method was developed by hooking a LED with the circuit in the

breadboard and coding the LED to light up every time the sensor pick up a signal. The same was done for the ultrasound sensors by having an object placed in front or it and seeing if any change occurred in the serial monitor of the Arduino program. If a false positive was detected, we checked over the wiring and the code to see what was causing the issue. In the case of our infrared sensor, we determined that its wide field of view was causing the problem, and we solved it by creating a funnel that focused its vision to a much smaller range.

4. Teamwork is very important in this project. Our group had an overall good experience with teamwork, as everyone was fairly active in terms of communication. One key to our success in this area is simply having technology like GroupMe and mobile phones available to us. We were able to coordinate meetings, explain concepts, and report absences nearly instantaneously and in a way that everyone in the group could view on their own time. Because of that, we typically knew what we needed to work on and how it needed to be implemented. Our in-person meetings were also crucial to developing teamwork, as group members would often take turns teaching each other about certain concepts, and we often found ourselves getting dinner together after meetings. The atmosphere built from getting close with teammates allows for better communication because everyone feels free and able to speak their mind. As a result of this, our team did not suffer from groupthink, and tended to be able to come up with unique solutions when we needed to.
5. Project management skills are necessary for big projects like the OSV, which require a lot of work and time to create. A cost chart was made to track how much money would be needed for the OSV and to make sure that the bill was split evenly between the people in the group. The mass of OSV, as more and more materials were gathered, also required a chart to keep track of mass so that it did not go over the requirement. A schedule of the construction process was needed in order to keep track of what needed to be done by a certain time, so that the project was evenly spread out to give enough time to test and fix any problems that the team encountered, as well as time to meet the deadline of milestones. We also tried to implement subgroups, which would work independently of

the other subgroups so that everyone could work more efficiently. We learned that it is very important to chart all information gathered in a way that everyone can easily view it, as this aids the entire team in keeping track of everyone's progress. We also learned that it is important to have and adhere to strict guidelines for project-related functions such as meetings, as without a strong precedence in this area the team can quickly become very disorganized and dysfunctional.

Appendix

Propulsion Calculations

F_N = normal force
F_g = Force of gravity = (2.50)(9.81) = 24.5 N
C_{RR} = coefficient of rolling resistance
F_{RR} = force of rolling resistance
F_{sf} = force of static friction
TE = tractive effort
τ = torque
ω = angular velocity
r = radius of the wheels = 5.00 cm
w = width of the wheels = 3.60 cm
d = diameter of the wheels = 10.0 cm
μ = Coefficient of static friction = 0.700
h_{cg} = height of CoG (arbitrary) = 0.130 m
L_{cg} = length from CG to wheel = 0.100m
L_w = length from wheel to wheel = 0.200 m

Calculations at 35 Degrees:

$$F_{Nr} = \frac{mg(L_{cg}\cos(35)+h_{cg}\sin(35))}{L_w} = \frac{24.5(0.100\cos(35)+0.130\sin(35))}{0.200}\ x\ \left(\frac{1\ F_{Nr}}{2\ wheels}\right) = 8.53\ N\ per\ rear\ wheel$$

$$F_{Nf} = \frac{mg((L_w-L_{cg})\cos(35)-h_{cg}\sin(35))}{L_w} = \frac{24.5((0.200-0.100)\cos(35)-0.130\sin(35))}{0.200}\ x\ \left(\frac{1\ F_{Nf}}{2\ wheels}\right) = 1.50\ N\ per\ front\ wheel$$

$$C_{RRr} = \left[3.33\ x\ \left(\frac{8.53}{3.60(10.0)^2}\right)\right]^{1/3} = 0.429$$

$$C_{RRf} = \left[3.33\ x\ \left(\frac{1.50}{3.60(10.0)^2}\right)\right]^{1/3} = 0.241$$

$$F_{RRr} = C_{RR}\cdot F_N = (0.429)(8.53) = 3.66\ N$$

$$F_{sfr} = \mu\cdot F_N = (0.700)(8.53) = 5.97\ N$$

$$F_{RR} < TE < F_{sf} \rightarrow \boxed{3.66\ N < TE < 5.97\ N}$$

Calculations at 0 Degrees:

$$F_N = \frac{F_g}{4} = \frac{2.50(9.81)}{4}$$
$$= 6.13\ N\ on\ each\ wheel$$

$$C_{RR} = \left[3.33\ x\ \left(\frac{6.13}{3.60(10.0)^2}\right)\right]^{1/3} = .384$$

$$F_{RR} = C_{RR}\cdot F_N = (.384)(6.13) = 2.35\ N$$

$$F_{sf} = \mu\cdot F_N = (0.700)(6.13) = 4.29\ N$$

$$F_{RR} < TE < F_{sf} \rightarrow 2.35\ N < TE < 4.29\ N$$

Tractive Effort

@0°	$2.35N < TE < 4.29N$
@35°	$3.66N < TE < 5.97N$
Combined	$3.66N < TE < 4.29N$

$$d = \frac{1}{2}at^2 \rightarrow 15.0\,m = \frac{1}{2}(180.s\ x\ 180.s)a \rightarrow a = 0.001\frac{m}{s^2}$$

Desired Torque:

$TE\ x\ r = \tau = 3.73(5\ cm)$

$= 18.6\ N \cdot cm$

$$F_{N\ total} = 4(6.13) = 24.5\ N$$

$$C_{RR}L = \left[3.33\ x\ \left(\frac{24.5}{3.60(10.0)^2}\right)\right]^{1/3} \cdot 24.5 = 14.9\ N$$

$$TE - C_{RR}L = ma \rightarrow TE = 2.50(0.001) + 14.9 = 14.9N\ x\ \left(\frac{1}{4\ wheels}\right) = 3.73\ N\ TE\ for\ each\ wheel$$

$$v = \frac{d}{t}$$

$$v = \frac{15m}{180sec}$$

$$v = .083m/sec$$

$$\omega = \frac{v}{r_{tire}}$$

$$\omega = \frac{.083m/s}{.05m}$$

$$\omega_{desired} = 1.66\frac{rad}{s}$$

$$\tau = \left(\frac{-41.78kg \cdot cm}{1.885\ rad/s}\right)\omega + 41.78$$

$$(1.90\ kg \cdot cm) = (-22.16)\omega + 41.78$$

$$\omega = 1.800\ \frac{rad}{s} = .172\ kRPM$$

Black Box Orientation Equations

1. $90 - cos^{-1}[(x * ycos(\theta smallest))/black\ box\ side\ length]$
2. x*sin(θcalculated)
3. $90 - cos^{-1}[(z * ycos(\theta smallest))/black\ box\ side\ length]$
4. z*sin(θcalculated)
5. x*cos(θ) = y
6. z*cos(θ) = y

Meeting Minutes

Date	Members Present	Work Done Pre-Meeting	Work Planned for meeting	Work Done During Meeting	Work to be Done After Meeting
9/7/2018	Everyone	N/A	Discuss finances; buy APC; start working on plan for vehicle; potentially get a base sketch of the vehicle done	Brainstorm questions (is the Black Box buried, what are the dimensions of the Black Box, the weight of the Black Box, will the box be standing upright on the sand, etc.); research tires with deep treads; decided on 4 wheel drive; need flat board base for OSV; compared prices of APC220; considered lynxmotion motors; considered printing motor mounts and wheel adapters; realized we already have ultrasonic sensors in our	Find out blackbox dimensions; study a roomba's turning; finalize tires and purchase them; study mini claw mechanism; research light wood for OSV base

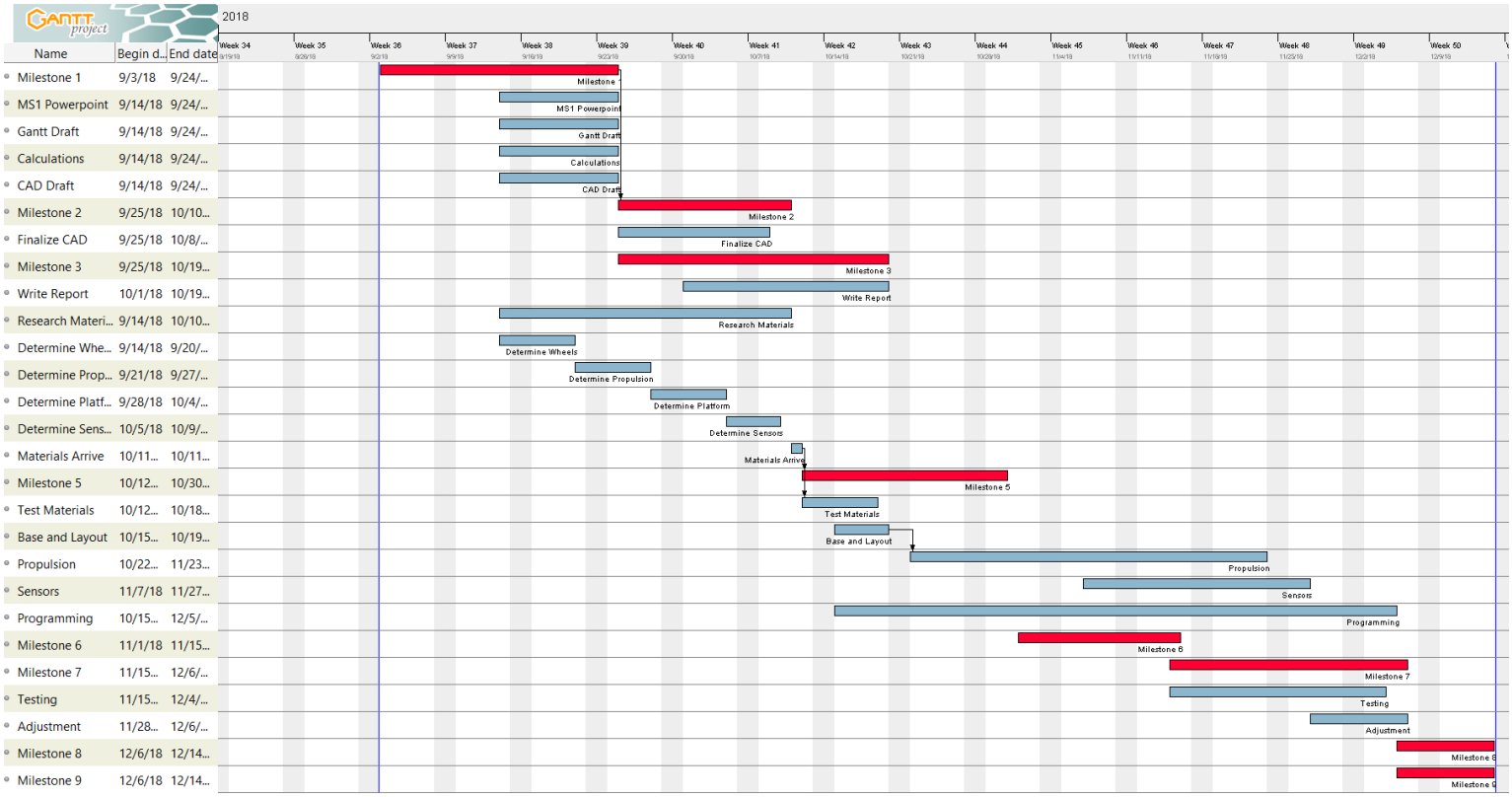
				possession; decided on rotating steering over skid steering	
9/14/2018	Everyone	Found dimensions of Black Box; found approx. size of potential tires and purchased them	Design what will be on the car, and what we need to put on it for the picking up mechanism	Drew up sketch of OSV and everything on it; found base for OSV; decided to use NiMH batteries; decided what will be on car (IR sensor, 3 ultrasonic sensors, romeo microcontroller, two batteries, four motors, servos and arms for picking up mechanism); decided for turning, the adjacent wheels will be synced; found potential batteries that we will compare with once we do power calculations	Figure out exact details of how we will "lift" the forklift-style arms of the picking up mechanism; work on torque and tractive effort calculations; work on MS1
9/21/2018	Everyone	Calculations; MS1	Finishing MS1	Finished MS1; decided who will present	Prepare for MS1 presentation

9/28/2018	Everyone except Atticus	MS1 presentation; calculations; CAD drawings	Go over MS1; work on fixing torque calculations and normal force	Divided team into groups to prepare for M2: CAD, project management, math, battery, servo, circuit diagrams; fixed normal force and tractive effort	Continue fixing torque calculations, find correct battery, find correct servos, do the detailing in the CAD
10/5/2018	Everyone except Atticus and Bahaa	Working with calculations	Work on M2: finish calculation, circuits, servo, sensors, CAD visuals, battery	Worked on a circuit diagram, solved for a desired tractive effort that fits within our TE range; work on MS2	Reconfirm the battery we are working with reconfirm the motors we purchased; finish MS2 due following week
10/12/2018	Everyone except Atticus	Finished and presented MS2; purchased final materials for OSV	Go over MS2	Work out logistics of servos and picking up mechanism; determine whether servos are on continuous draw; determine if gearbox is necessary	Work on MS3 and gathering all of our purchased materials for MS4
10/19/2018	Everyone	Gathered materials and completed MS4; started working on MS3	Finish MS3	Finished MS3, read over everyone's parts, printed and turned it in	Rest and study for Q1 and Q2; start working on building the base for OSV

10/26/2018	Everyone except Bahaa	Built base for OSV; bought and received the hex adapters for wheels	Attach wheels, motors and romeo microcontroller	Attached wheels and motors to OSV base; wired the circuit for the motors	Prepare for MS5 motor testing; write code for OSV to move forward to a specified location; write code for turning
11/2/2018	Everyone except Anjali, Bahaa, Nicole	Finished move forward and turning code; completed MS5	Work on IR sensor circuit and wiring	Integrated the IR sensor into the OSV breadboard; draw CAD for ultrasonic sensor mounts	Finish sensor mounts and attach it to OSV; write code for IR sensor detecting and moving to Black Box
11/9/2018	Everyone except Atticus and William	3D printed sensor mounts and attached them; wrote code for IR sensor	Work on integrating sensors into OSV and write code for obstacle avoidance	Wired up the sensors to the OSV; worked on sensors code; draw CAD for servo arms and test print	Continue working on sensor code; work on IR sensor testing for MS6
11/16/2018	Everyone	Completed second half of MS6	Continue working on sensors and servo arm printing	Continued servo arm test printing; found an issue with sensor code	Attempt to solve the problem with obstacle avoidance code and sensor integration
11/23/2018	Thanksgiving Recess	Thanksgiving Recess	Thanksgiving Recess	Thanksgiving Recess	Thanksgiving Recess
11/30/2018	Everyone except Shahed and Julia (illness)	Working on resolving issue with sensor code; made final servo arm prints	Fix sensor code and integrate IR code into one consolidated code; assemble servo motors to 3D servo arms	Wrote several codes to attempt fixing sensor code issue; did multiple tests of OSV obstacle avoidance (failed); integrated IR code into the	Try to fix obstacle avoidance code

				combined, final code	
12/7/2018	Everyone except Atticus	Fix obstacle avoidance code; write code for servo motors	Integrate sensors and IR codes and test it with the servo motors	Tried fixing sensors and final testing of MS7 (failed to integrate the servos properly into OSV)	Work on MS8 poster and MS9 report; try fixing sensor code again
12/10/2018	Everyone except Atticus	Worked on MS8 and MS9; worked on obstacle avoidance code	Fix obstacle avoidance code; test MS7	Failed to fix obstacle avoidance code; tested MS7 (only able to navigate past rocky terrain)	Work on and finish MS8 and MS9

Gantt Chart



Code

```
#include <Enes100.h>
#include <DFRTank.h>


#define ECHO_PIN_LEFT 8
#define TRIG_PIN_LEFT 9
#define ECHO_PIN_RIGHT 2
#define TRIG_PIN_RIGHT 3
#define ECHO_PIN_CENTER 11
#define TRIG_PIN_CENTER 10
#define MARKER_ID  9
#define RX_PIN 13
#define TX_PIN 12
#define GOAL_X 2
#define GOAL_Y 1
#define TOLERANCE 0.07 // May need to Calibrate
#define CLOSEST_DISTANCE .45 //change back to 50 //this is in cm, May need to Calibrate
#define UP 1.570796
#define DOWN -1.570796

Enes100 enes("Lost", BLACK_BOX, MARKER_ID, RX_PIN, TX_PIN);
DFRTank tank;


double cm_Center, cm_Left, cm_Right;
double TOO_CLOSE_TO_WALL = .5;
boolean arrivedAtCenter = false;
boolean baseObjectivesAchieved = false;
boolean pickedUpBlackBox = false;
int count = 0;
double averageTheta = 1;
double previousTheta = 0;
double currentTheta = 0;
boolean continueTurning = false;


int turn_cw(double newTheta);
int turn_ccw(double newTheta);

void setup() {
  tank.init();
  enes.println("Setting up");
```

```
  while (!enes.retrieveDestination());
  enes.println("Retreieving Destination");

  while (!enes.updateLocation());

  enes.println("Connected and Setup");
}

double DistanceMeasurement(int Trig_Pin,int Echo_Pin){
  double duration;
    // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  digitalWrite(Trig_Pin, LOW);
  delayMicroseconds(5);
  digitalWrite(Trig_Pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig_Pin, LOW);

  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  duration = pulseIn(Echo_Pin, HIGH);
  return (duration/2) / 29.1;

}

void TurnCW(){
  tank.setRightMotorPWM(255);
  tank.setLeftMotorPWM(-255);
  delay(750); // May need to Calibrate
  tank.turnOffMotors();

}
void TurnCCW(){
  tank.setRightMotorPWM(-255);
  tank.setLeftMotorPWM(255);
  delay(750); // May need to Calibrate
  tank.turnOffMotors();

}
void MoveForward(){
  tank.setRightMotorPWM(255);
  tank.setLeftMotorPWM(255);
  delay(50);
```

```
}

void TurnLeftOrRight(double cm, String Direction){
  while(cm <= CLOSEST_DISTANCE){

    enes.updateLocation();
    if ((enes.location.y - TOO_CLOSE_TO_WALL) < 0){
      while(!turn_ccw(UP));
      enes.println("turning left too close to wall");

    } else if ((enes.location.y + TOO_CLOSE_TO_WALL) > 4){ // only turns left if not close to left wall
      while(!turn_cw(DOWN));
      enes.println("turning right too close to wall");

    } else if (Direction.equals("Center")){ // turns toward center
      if (enes.location.y > 1){
        while(!turn_cw(DOWN));
      } else {
        while(!turn_ccw(UP));
      }

    } else if (Direction.equals("Right")){
      while(!turn_ccw(UP));

    } else {
      while(!turn_cw(DOWN));

    }

    // Center has priority, because after the OSV is done sensing the center, it should sense the objects
            with the side sensors
    // May need to calibrate how far/long the OSV turns when something is sensed by the CENTER
            sensor, because of its blind spots

    if (Direction.equals("Center")){
      cm = DistanceMeasurement(TRIG_PIN_CENTER,ECHO_PIN_CENTER);


    } else if (Direction.equals("Left")) {
      cm =  DistanceMeasurement(TRIG_PIN_LEFT,ECHO_PIN_LEFT);


    } else if (Direction.equals("Right")){
      cm = DistanceMeasurement(TRIG_PIN_RIGHT,ECHO_PIN_RIGHT);
```

```
   }
  }
}

//turns clockwise to theta, meant to be used in loop until 1 is returned
int turn_cw(double newTheta){
  enes.updateLocation();

  if(count == 1){
    previousTheta = enes.location.theta;

  } else if (count > 15){
    currentTheta = enes.location.theta;
    averageTheta = currentTheta - previousTheta;
    count = 0;

  } count ++;

  if (averageTheta == 0){
    tank.setRightMotorPWM(-123);
    tank.setLeftMotorPWM(-123);
    delay(750);
    averageTheta = 1;

  }
  if(enes.location.theta > newTheta + TOLERANCE) {
    //turn right:
    tank.setRightMotorPWM(-123); //change back, multiply by -1
    tank.setLeftMotorPWM(123); //change back, multiply by -1
    //enes.println("CW -- TRUE");
    return 0;

  } else if (enes.location.theta < newTheta - TOLERANCE) {
    //turn left:
    tank.setRightMotorPWM(123); //change back, multiply by -1
    tank.setLeftMotorPWM(-123); //change back, multiply by -1
    //enes.println("CW -- FALSE");
    return 0;

  } else {
    return 1;
  }
}
```

```
//turns couner clockwise to theta, meant to be used in loop until 1 is returned
int turn_ccw(double newTheta){
 enes.updateLocation();

 if(count == 1){
  previousTheta = enes.location.theta;

 } else if (count > 20){
  currentTheta = enes.location.theta;
  averageTheta = currentTheta - previousTheta;
  count = 0;

 } count ++;

 if (averageTheta == 0){
  tank.setRightMotorPWM(-123);
  tank.setLeftMotorPWM(-123);
  delay(500);
  averageTheta = 1;

 }
 if(enes.location.theta < newTheta - TOLERANCE) {
  //turn left:
  tank.setRightMotorPWM(123); //change back, multiply by -1
  tank.setLeftMotorPWM(-123); //change back, multiply by -1
  //enes.println("CCW -- TRUE");
  return 0;

 } else if (enes.location.theta > newTheta + TOLERANCE) {
  //turn right:
  tank.setRightMotorPWM(-123); //change back, multiply by -1
  tank.setLeftMotorPWM(123); //change back, multiply by -1
  //enes.println("CCW -- FALSE");
  return 0;

 } else {
  return 1;
 }
}


void ObjectDetection() {
 int i = 0;
 while(true){
  for(int i = 0; i < 6; i++){
```

```
    cm_Left = DistanceMeasurement(TRIG_PIN_LEFT,ECHO_PIN_LEFT); // Convert the time into a
            distance
    cm_Center = DistanceMeasurement(TRIG_PIN_CENTER, ECHO_PIN_CENTER);
    cm_Right = DistanceMeasurement(TRIG_PIN_RIGHT,ECHO_PIN_RIGHT);

    if(cm_Center <= CLOSEST_DISTANCE && i == 5){
      enes.println("Object Detected");
      TurnLeftOrRight(cm_Center, "Center");
      // MoveForward();

    } else if (cm_Left <= CLOSEST_DISTANCE && i == 5){
      TurnLeftOrRight(cm_Left, "Left");
      enes.println("Object Detected");
      //MoveForward();

    } else if (cm_Right <= CLOSEST_DISTANCE && i == 5){
      TurnLeftOrRight(cm_Right, "Right");
      enes.println("Object Detected");
      //MoveForward();

    }
   }
   return;
  }
}

void loop() {
  if(pickedUpBlackBox){
    //returning to LZ
    // turn to theta = pi then go straight
    //if object detected, turn away until not sensed and then move forward a SET distance
    //turn back toward LZ and continue forward

  } else if(baseObjectivesAchieved){
    //picking up Black Box
    pickedUpBlackBox = true;

  } else if(arrivedAtCenter){
     tank.setRightMotorPWM(-255);
     tank.setLeftMotorPWM(255);
     delay(100);
     tank.turnOffMotors();

    while (IR_Read()) {
     tank.setRightMotorPWM(255);
```

```
    tank.setLeftMotorPWM(255);

    }
    baseObjectivesAchieved = true; //when arrived at BB and transmitted coordinates

  } else if (enes.updateLocation()) {
   enes.print("x: ");
   enes.print(enes.location.x);
   enes.print("y: ");
   enes.println(enes.location.y);

   if(enes.location.x < (GOAL_X + TOLERANCE)){
    while(!turn_cw(0));
    ObjectDetection();
    for(int a = 0; a<20; a++){
     MoveForward();
    }

   } else if(enes.location.y > (GOAL_Y + TOLERANCE)){
    while(!turn_cw(DOWN));
    ObjectDetection();
    MoveForward();


   } else if(enes.location.y < (GOAL_Y - TOLERANCE)){
    while(!turn_ccw(UP));
    ObjectDetection();
    MoveForward();


   }else {
    enes.println("arrived at center.");
    tank.turnOffMotors();

    enes.navigated();
    Coordinate ret(enes.location.x, enes.location.y);
    enes.baseObjective(ret);
    arrivedAtCenter = true;

   }
  }

}
```