


TGTWAPTT Final Design Report

Advaith Bantval, Caitlin Lee, Carson Hoover, Gianna Moore, Jack Irvin, Katie Crawford, and
Sahar Farajun

University of Maryland

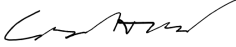
ENES100 - 0801 Black Box Mission

Signed:

Advaith Bantval -  Executive Summary, Introduction, Modeling, Construction
Details, Mission Objectives, Circuit Diagram, IR Shortcoming and Range drawings, Final Code

Caitlin Lee - Preliminary Design Shortcomings, Control Algorithm

Carson Hoover -  Project Management, Prototyping, Formatting

Gianna Moore - 

Jack Irvin -  Propulsion

Katie Crawford -  Executive Summary, Introduction, Preliminary Design

Sahar Farajun -  Control Algorithm, Programming, Circuitry, Troubleshooting

Table of Contents

Executive Summary	3
Preliminary Design Details (Introduction and Shortcomings)	4

TGTWAPTT	2
Final Design Details	7
Construction details	13
Final Bill of Materials	13
Project Management	13
Disposal Plan	15
Product Performance and Evaluation	15
Lessons Learned	16
Appendix A- Final Design Drawing	21
Appendix B- Minutes	23
Appendix C - Final Code	30

Executive Summary:

The black box assignment is a challenge where a team must design, build, and test an over sand vehicle (OSV) that can locate and travel to the source of a radio signal. The OSV must then transmit the coordinates of the signal's source, pick up the device sending the signal, and return to its starting area, avoiding obstacles in the mission area the whole time. The OSV must act completely autonomously and is tracked by a camera overhead which communicates with the OSV and tells it its coordinates and orientation.

Our Over-Sand Vehicle (OSV) was constructed from a plywood platform supported by four drive wheels, each powered by their own DC motor. We utilized a tank-steering mechanism and mounted most of our electronics and mechanisms on the top of our platform. On the front of the OSV are two ultrasonic sensors which act as the vehicle's vision system, detecting obstacles in the OSV's path. In addition, there is an infrared sensor with a tube-shaped casing mounted on a servo motor to accomplish the base mission objectives of the black box mission (locating the source of the IR Signal). In the rear is a lever-type contraption which is designed to pick up the black box when one corner is flush against the body of the OSV. To accomplish this, a servo motor will pull a string connected to the top of the device to pull it back and therefore secure the black box.

To begin accomplishing the objectives, once the OSV has navigated to the front of the area on the arena where the black box could reside, the servo motor in the front of the OSV will swivel 180° to attempt to detect an infrared signal. When a signal is detected, the OSV will turn 180 degrees and attempt to backup directly into the source of the signal. A limit switch on the rear of the OSV will be triggered when the source is hit. The OSV will then send the coordinates of the source to the computer and enact its black box retrieval operations. After this, the OSV will navigate back to the landing zone, or area where the OSV began its mission, to complete all mission objectives. When we tested our OSV's ability to accomplish all mission objectives, it performed worse than expected. It was only able to accomplish one of the primary objectives, which was driving over the rocky terrain. Despite the failure of our final design, our team was able to gain invaluable experience throughout the duration of this project.

Introduction

The central mission/main task for our team is to locate the black box, a 90mm square box somewhere in the arena that is emitting an IR frequency. The black box acquisition mission is unique from other missions in that our OSV must detect the location of our mission site, as opposed to it being provided for by the main computer and communicated to the OSV via a radio-frequency, or RF, module. To have a successful design, our OSV must navigate to within 250mm of the black box, determine the location of the black box to within 75mm of accuracy, and then transmit this location to the main computer via the RF module. The secondary objectives include picking up the black box completely off the ground and returning it to the landing zone.

There are several limiting factors for our OSV design. The total cost of the materials used to construct the OSV must not exceed \$350, the total mass of the OSV must not exceed 2500g, and the footprint size must not exceed 300mm by 300mm. Also, there are only 13 digital input/output pins available on a singular Arduino Microcontroller, so our group had to remain conscious of this while designing our vehicle.

Our OSV is a four-wheeled, Computer Numerical Control (CNC) cut, plywood chassis vehicle driven by four motors that are controlled by the Arduino Uno MicroController (Figure 1, 2). It is equipped with an infrared sensor (to locate the black box) that is mounted on a servo motor and two ultrasonic sensors (to successfully navigate around obstacles) that are mounted at the front of our OSV. Both sensors will communicate with the Arduino to allow our OSV to make educated decisions on how to operate the motors at any given moment throughout the mission (move forward, rotate, etc.). Our OSV is also equipped with a motor driver to control our motors with a tank-drive method, a limit switch to notify the Arduino when the black box has been reached, and a servo motor mounted on the top of the OSV to raise the lever mechanism once the black box is acquired. Our lever mechanism consists of a cradle with a 90 degree angle cut out that the black box fits into, and a drawbridge-type lever system that raises the cradle once the black box is nestled inside (Figure 1,2). This mechanism thus raises the black box off the ground via the lip that is between the top and bottom half of the black box.

Before our trials for Milestone 7, all of our components to the OSV worked independently. Unfortunately, due to several last-minute part failures and the lack of refinement of our system integration, our design failed to be successful in the arena. After several hours of out-of-class testing, we were able to accomplish the most basic of the mission objective for Milestone 7, which was our OSV navigating over the rocky terrain in the arena.

Preliminary Design Shortcomings

Navigation

Prior to any physical testing or experimentation, we planned to implement a mapping system to determine the locations of all obstacles within the sandbox. This would allow us to determine our ideal path to a predetermined waypoint prior to any movement, thereby increasing modularity and adaptability, and would theoretically decrease the amount of time necessary for the OSV to move to the destination. However, after writing the algorithm and constructing the ultrasonic system, we found that a major failure of the system was its inability to give accurate distances to a flat surface if the surface was not placed orthogonally to the ultrasonic signal (Figure 3a). While this made it nearly impossible to get an accurate map of the black box's location, the textured, rough faces of the rocks seemed to allow for near-accurate distance readings in most scenarios (i.e., different positions and orientations of the rock with respect to the ultrasonic sensor). With this knowledge, we modified our navigation plan. Instead of trying to map the entire sandbox, we would only map the front part (before $x=2.25$), as we knew that there would be obstacles in this area, but no black box. Then, once we reached $x=2.25$, then we would scan to determine which rear quadrant contained the box. To do this, we would attach two ultrasonic sensors to a single servo, such that one would be higher up than the other. The upper one would be mounted at a height taller than the black box, so that the only objects it would detect would be rocks, whereas the lower ultrasonic would detect both rocks and the box (Figure 3b). With this system, we would be able to receive data from both ultrasonics, and if at a certain angle only the lower ultrasonic receives a signal (not constrained to any maximum distance), then we would be able to determine that the box was in a certain quadrant. While this system would not be able to tell us the exact location of the box, we would be able to limit our search to

half of the possible area. However, we were unable to fully test and implement this idea due to time constraints.

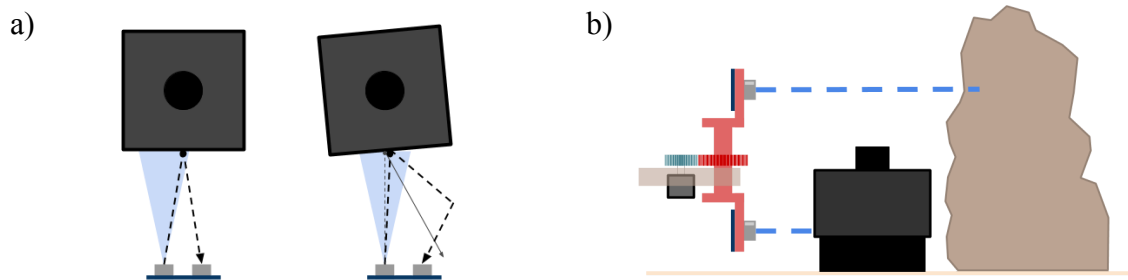


Figure 3. The problem with ultrasonic mapping, and a possible solution. a) An angled surface will cause the signal to not be reflected immediately back to the receiver resulting in inaccurate distance measurements. b) The top ultrasonic senses only rocks, whereas the bottom senses all objects.

Advanced Objectives

Another issue we faced was designing a sufficient way to pick up the black box. As a team, we discussed several options for picking up the black box and even prototyped a couple designs. We failed to narrow our design options to a single design until very late in the semester, as we continued testing new ideas. Our preliminary design included a collector-type system, intended to use a method found in tennis ball throwing machines, to trap the black box in the rear of the OSV. Rotating rubber wheels would create a high force of friction between the 3D printed material of the black box and the rubber wheels to draw the box in toward the center of the OSV to transport it back to the landing site. This design proved to be too ambitious within the time constraints we were given, and would have required lots of mechanical calculations, precision, and testing to be functional. We decided that it was more important to focus on meeting the primary objectives than creating this complicated design. We focused on meeting these advanced objective of picking up the black box and bringing it back to the landing site once our OSV was properly constructed and functioning. Because of the lower priority of these objectives, we opted for a much simpler design, electronically. Instead of two motors mounted vertically, our new design needed only one 20kg servo motor to operate successfully, and the design worked reliably and consistently.

Mission Objectives

We also ran into issues during the process of innovating a solution for the base mission objectives. Our initial idea to accomplish the base mission objectives was to use two IR sensor to triangulate the source of the signal since it was a tried and true method that had been used in the past to locate the black box, however, we searched for other more innovative ideas to accomplish the same end goal. Our next idea to accomplish the base mission objective was to mount a KY-022 Infrared Sensor on a stepper motor, have the motor swivel in the front of the OSV, and stop in the direction of the most powerful IR signal. We assumed the angle at which the IR signal was the strongest would be the angle at which the sensor was directed perpendicularly to the source (Figure 4). Fortunately, after repeated testing with the KY-022 IR receiver, it was discovered that rather than indicating the magnitude of the signal, it indicated the frequency of the signal. Testing with the IR Remote provided with the remote, we saw that each button on the remote emitted an IR signal with a distinct PWM and therefore indicated a different value to the receiver. The black box's PWM was constant, emitting 38 kHz pulse 4 times per second. Regardless of the angle at which the receiver was placed, the PWM read by the receiver would be the same and would essentially tell the OSV whether or not it was in the range of the black box's signal. As a result of this shortcoming, we began to re-explore triangulation, using a color sensor and ultrasonic sensor to detect the black box, and using 2 ultrasonic sensors, before ultimately settling on a solution to limit the scope of the KY-022 module and placing it on a servo motor. The final design to accomplish base mission objectives was somewhat similar to that of our initial method but it used a shield that prevented the receiver from detecting any IR signal unless it was positioned perpendicularly to the source.

Final Design Details

Structure

The base of our Over Sand Vehicle is made of $\frac{3}{8}$ " CNC cut plywood, measuring 160x140mm. Our constructed final design follows Figures 1 and 2. Components of the OSV such as sensors, motors, and controllers are attached to the base in via screws or hot glue. The two breadboards are attached via the adhesive that is manufactured onto the breadboard base. The

OSV base is supporting three main systems: the sensor array, the drive system, and the black box collector.

The sensor array is on the front half of the base and includes the arduino and both breadboards, two ultrasonic sensors, an infrared (IR) sensor on a servo motor, and a radio frequency (RF) module for communication to the main computer attached to the breadboard. The drive system is behind the sensor array and on the underside of the chassis and consists of one motor driver and four motors that each drive one wheel. The motor driver controls the motors such that it drives two wheels at the same speed and direction on each side of the OSV, employing a tank-drive system. This was done to reduce weight and simplify the navigation code.

The third and final system, the black box collector, consists of a 3D-printed cradle that is fastened to the OSV by a second 3D-printed part that serves as a backplate, which is attached to a hinge mounted on the OSV. The top of the backplate is fastened to a 20kg servo motor by a string, and when the servo motor rotates, it pulls the string in and out, thus raising and lowering the entire mechanism. The servo motor is mounted on a 3D-printed part that we did not end up using in our final design.

Our OSV was constrained by a mass limit of 2.5kg, and our final OSV weighed approximately 2.1kg (Table 1).

Component	Mass(g) per unit	Number of Units	Net Mass(g)
Chassis	600	1	600
Wheels	100	4	400
6V Motors	45	4	180
Servo Motors	80	2	160
Batteries	350	1	350
Arduino/Electronics	190	1	190
Blackbox Collector	90	1	90
3D Printed Parts	130	1	130

TOTAL	2100
--------------	-------------

Table 1. Estimated Mass Breakdown

Above the wiring and other components on the top of the chassis of our OSV, we constructed a cardboard platform that was supported by cardboard stilts. The purpose of this is to provide a secure location for the RF tile to be placed on our OSV. There is duct tape on this platform to provide a temporary adhesive so that the tile could be secured when need be, but removed when needed.

Propulsion

We made a decision matrix to decide between tracks and wheels for our OSV. Since wheels won, we researched the best type of wheels for use on sand and decided on large paddle wheels that are made for Traxxas RC cars. We originally planned to use a chain drive to run both wheels from a single motor on each side, but the design was too difficult to implement. To solve this problem we decided to directly drive each wheel, using a total of 4 motors. The 6V motors were purchased from another group who had completed this project last semester. We created adapters that allowed the motors to drive the wheels directly. The OSV uses its 4 wheel drive for tank-turning, allowing for rotation in place. The OSV has no suspension because we determined that it was not necessary to traverse the rocky terrain. The motors and wheels are mounted 3cm from each end of the rectangular chassis. The OSV was capable of traveling 0.3m/s or 30cm/s on flat sand.

a)

$$\begin{aligned}
4F_T &= 4F_{RR} \\
F_T &= F_{RR} \\
C_{RR} &= 0.3 \quad r = 0.05\text{m} \quad F_N = \frac{mg}{2} \quad \tau = F_T \times r \\
\tau &= \left[C_{RR} \times \frac{mg}{4} \right] r \\
\tau &= \left[0.3 \times \frac{(2.370\text{kg})(9.80 \frac{\text{m}}{\text{s}^2})}{4} \right] 0.05\text{m} \\
\tau &= 0.0696 \text{ Nm}
\end{aligned}$$

b)

$$\begin{aligned}
\Sigma F_{\text{parallel}} &= 2F_T - (2F_{RR} + mg \sin 35) \\
F_T &= F_{RR} + \frac{1}{2}mg \sin 35 \\
C_{RR} &= 0.3, \quad r = 0.05\text{m} \quad F_N = \frac{mg}{2} \quad \tau = F_T \times r \\
F_{RR} &= F_N \times C_{RR} = \frac{mg}{2} \times 0.3 \\
\tau &= (F_{RR} + \frac{1}{2}mg \sin 35) \times r \\
\tau &= \left[0.3 \times \frac{mg}{2} + \frac{1}{2}mg \sin 35 \right] r \\
\tau &= \left[0.3 \times \frac{(2.370\text{kg})(9.80 \frac{\text{m}}{\text{s}^2})}{2} - \frac{1}{2}(2.370\text{kg})(9.80 \frac{\text{m}}{\text{s}^2}) \sin 35 \right] 0.05\text{m} \\
\tau &= 0.422 \text{ Nm}
\end{aligned}$$

Figure 5 . Torque calculations. a) Calculated necessary torque that must be supplied by each of our four motors, used in motor selection process b) Calculated torque that our motors were able to supply, which was well over what we needed to overcome necessary torque.

OSV Mission

Our method to accomplish the primary mission objectives used a KY-022 Infrared Receiver and a limit switch. The mission objectives were to navigate to within 250 mm of the black box and transmit its coordinates to the central computer system. In order to achieve this, we placed a 10 cm long 3D printed tube over the IR sensor such that the angular range of the sensor was limited to signals only in its immediate front (Figure 6). This sensor system was mounted on a servo motor on the front of the OSV using a 3D printed adapter, and was programmed to sweep 180° in front of the OSV after navigating through the obstacles and reaching the waypoint y=2.25 m. The code also told the servo to stop immediately when a signal was detected. This system was accurate to within 75 mm. When a signal was detected, using the servo's angle θ (from 0° to 180°), the OSV would turn to the angle $\theta+180^\circ$ and drive backward toward the signal's source. A limit switch mounted on the OSV's rear would be triggered when it bumped into (theoretically) the signal's source, and the coordinates at which the limit switch was triggered would be the coordinates sent to the computer. In the case that the OSV drove too far without having its limit switch triggered, it would send a contingency code with all possible coordinates based on the IR reading to the computer.

To accomplish our advanced objectives, we decided on a cradle that attaches to the rear of our OSV that would lower down and be raised up on a “drawbridge”-type mechanism. This is powered by a 20kg servo motor. The cradle is raised all the way, such that when the OSV detected the black box with the IR sensors mounted to the front, it would turn around back up into it until a limit switch was triggered on the bottom of the cradle, meaning it had made contact with the black box. Then, the cradle would drop, the OSV would back up slightly more, and then the cradle would raise, lifting the black box by its lip. The black box is then secured on the OSV and ready to be taken back to the landing zone.

Power

We powered our motor driver with a 12V battery pack and our Arduino and sensors with a 9V battery. We had initially considered powering our motors using the Arduino 5V output pin, as our motor driver’s range of functioning voltage was 5-12V, however we quickly discovered that 5V was not nearly enough voltage to supply our wheels with the appropriate torque to move on sand and over the rocky terrain. We then tried powering the motor driver with a 9V battery pack, but this also proved to be too little power for our four motors, so we instead used a 12V battery pack. We used the 5V output pin on the motor driver to supply the RF with enough power to function properly. We powered our Arduino with 9V and used its 5V output pin to power our IR and ultrasonic sensors.

Control Algorithm

Overall, our OSV was intended to rely heavily on the vision system to determine our position at every moment in time, allowing for smooth, constant movement until the desired position or orientation is reached. However, due to difficulties with the RF module and the vision system updating speed, our final version of code was modified to turn and move in bursts (e.g., move for 500 ms, then stop and update the location) to allow for more time to receive and process data from the vision system. Pseudocode for this process is presented below.

Turn to face $y = 1$

//theta = $\pi/2$ if below $y=1$, theta = $-\pi/2$ if above $y = 1$

Move to $y = 1$

Turn to $\theta = 0$

Move across rocks (to $x = 1.25$)

While before possible black box area, i.e. $x < 2.25$ {

If only left ultrasonic senses object, turn right ($\theta = -\pi/2$), move 40 cm, turn to $\theta = 0$

If only right ultrasonic senses object, turn left ($\theta = \pi/2$), move 40 cm, turn to $\theta = 0$

If both ultrasonics sense object {

If below $y = 1$, turn left ($\theta = \pi/2$), move 40 cm, turn to $\theta = 0$

If above $y = 1$, turn right ($\theta = -\pi/2$), move 40 cm, turn to $\theta = 0$

}

Move forward

Run ultrasonics

}

Turn to $y = 1$ (currently at $x = 2.25$)

Scan IR servo, get angle at which IR signal received

Turn rear side of OSV to input angle

Move in reverse until limit switch triggered

Move forward a little, lower the collection mechanism

Move in reverse for a slightly longer distance than was moved in forward direction

Raise collection mechanism

Turn to $\theta = 0$

Navigate back to $x < 1.25$, with object avoidance from above

Construction Details

Our final design as constructed is illustrated in Figures 1 and 2. The plywood chassis of our OSV was CNC cut with the dimensions of 160x140mm. The motor mounts were 3D printed

and fastened to the underside of the OSV using screws that threaded through the small pre-cut holes on the CNC piece. Using 3D printed wheel mounts, the paddle wheels were attached to the 6V motors in the motor mounts. In the front-underside of the OSV, a raft from a 3D printed Arduino Uno holder is hot glued and space for an Arduino Servo motor is cut out. This servo is fastened to the raft with hot glue. Two ultrasonic sensors with their corresponding mounts are hot glued to the extremes of the plywood base on the front of the OSV. 5mm behind the center of each of the ultrasonic sensor mounts, a 1cm wide hole is drilled to make room for the wires of the sensors. Immediately behind the hole on the left of the OSV is a mini breadboard dedicated to sensor circuitry, fastened to a 3D printed raft with electrical tape. Behind the mini breadboard on the face of the plywood base is an Arduino Sidekick breadboard dedicated to motor and RF circuitry, fastened to the plywood by duct tape. Our circuitry follows the schematic in Figure 7. Behind this breadboard on the left side of the OSV is a folded cardboard rectangular prism that is hot glued to the chassis. Inside it the OSV's kill switch hot glued in to fix in a stable position to the base.

The Arduino Uno that is used to run all OSV code is positioned behind the right ultrasonic sensor and is encased in a 3D printed Arduino Uno case. In the space on the plywood surface between the two right side tires, a prior iteration of motor housing is hot glued to the chassis of the OSV, with a 2cm wide hole drilled through the base to channel drive motor wires to the top of the OSV base. On top of this motor housing is a 20kg servo that is used to lift the advanced objective mechanism. This servo is hot glued to the housing and is positioned toward the left side of the OSV. Around this set up is the cardboard cut RF card holder that is also hot glued to the OSV base. In the back of the OSV, centered on the plywood base, is the advanced objective arm. This device is screwed to the OSV base with a hinge. The arm is connected to the 20kg servo using yarn. On the underside of the arm, a limit switch is hot glued such that it is closed when an object is flush against the underside of arm. The wiring for the limit switch is channeled under the motor wires on the underside of the OSV using a pink plastic straw to ensure its flexibility when the arm moves on the hinge.

Final Bill of Materials

Item	Price
4 x Wheels	\$35.99
Chassis	\$18.46
High torque servo	\$17.89
2 x ultrasonics	\$7.90
Battery pack	\$30.00
L298N	\$6.89
Uno	\$19.15
2 x 9v batteries	\$7.49
limit switch	\$1.23
4 x motors	\$80.00
3D printed parts	\$21.84
Total	\$246.84

Table 2. Price breakdown of OSV, as built.**Project Management**

Our team's project management plan was an adaptation of the Scrum agile project management. We utilized week-long sprints, which began every Monday and ended each Sunday. Every Friday, we held a group meeting to review the progress we had, plan the next week's work schedule, and reflect on which components of our project needed the most development. Our group kept track of the work that each individual was responsible for in an online taskboard using the website trello.com. In addition to our weekly meetings and class time, our group kept in communication using the GroupMe app and occasionally held group calls with Google Hangouts. While this system worked well for our group throughout the planning stage and during the beginning of the build stage, it broke down in late October, leading to increased disorder towards the end of the class.

A large part of what kept our project management plan on track throughout the beginning of the semester was the accountability that our weekly 10-3-1 reports provided. When the 10-3-1

reports were no longer required for a grade, our group stopped filling out the reports. This then lead to individuals not keeping track of the work they were doing, not sticking to the schedule we had set for ourselves, or how recognizing long they had been working on the same part. Our lack of commitment to time-management eventually caught up to us when we came up on Milestone 7 completely unprepared, behind schedule, and disorganized. Furthermore, our failures in Milestone 7 could have been a wake-up call, telling us to get organized, but instead, we continued on as we had been and did not ever return to our project management plan.

If we were to continue on in this project, the most important change we would make to our team management is reinstate weekly check-in meetings. These meetings do take up a lot of time on a Friday, when most people don't want to think about any more work and have checked out for the week, but they are crucial to our group staying organized and communicating effectively. Group meeting are where we all have a chance to get on the same page, voice our concerns, reflect on the work we've done, and plan out what we need to get done in order to stay on schedule. By all of us communicating in person about these things, we would be more on the same page as a group and we would all better understand the time constraints we would be working under. One of our biggest failures in the OSV project was our lack of regard for the deadlines we were supposed to be keeping, so by instituting a project management system that forces the members to be constantly aware of time constraints, our group would likely be more successful.

Disposal Plan

We plan on selling most of our electronic components to next semester's ENES100 groups. We bought some of our materials from past OSV teams, so this was the most logical disposal method we were drawn to. As many of us have acquaintances that will be taking the class next semester, this should not be difficult to accomplish. We will also be returning our borrowed electronics (ultrasonic and IR sensors, numerous arduino-compatible components, and many of the wires we used) to Advaith's friend that lent them to us at the beginning of the semester. Lastly, if we cannot sell, repurpose, or give away any of our materials (motor mounts, chassis, etc.), we will dispose of 3D printed parts in the scrap bin, and other materials in their designated disposal method (non-recyclable trash in the trash, recyclable materials in recycling

bins, batteries separately and safely disposed of at a battery collection location). We aim to dispose of our OSV in a responsible and sustainable manner, to lessen our project's impact on the environment.

Product Performance and Evaluation

Aside from our OSV not meeting our project performance expectations, we met almost all of the product specifications. Our OSV fit within all of the structure requirements (size and weight, no multi-purpose pre-built assemblies), and all of the power requirements (we used alkaline batteries that could power our OSV for more than 10 minutes). However, we failed to meet the performance requirements and some of the communication requirements. Our OSV was unable to complete all but one of the mission objectives (navigating over the rocky terrain), and was unable to successfully navigate to a waypoint. However, no part of our OSV detached in the arena and it was able to receive transmissions via the RF module. We also met the safety and cost requirements, in that we had a kill switch fastened to our battery at all times and that our total cost of construction was under \$350 (Table 2). We did not meet any of the mission requirements.

Several design changes that would have improved the functioning and performance of our OSV would be repairing the left side motors that no longer work, replacing the broken wiring, and reducing pin interference with the RF module. A large part of the reason our OSV failed to be successful was due to the fact that the motors on the left side of our OSV failed on the day of MS7 testing. Replacing them would allow our OSV to reliably move independently again. In addition, we found that many of our wires were no longer working for individual sensors and other various components. Identifying which wires were faulty and fixing this fundamental flaw in our circuitry would also improve our OSV's performance, because the Arduino would then be able to successfully control all aspects of our OSV. Lastly, the pin interference on our breadboard caused our RF communicator to fail to connect multiple times. It was overall unreliable, and a solution to this would be to use a soldered breadboard or a PCB to create more stable connections between our electrical components. If this did not solve the problem, we could construct a second breadboard just for the RF communicator, therefore isolating that specific circuit and removing the possibility of pin interference.

Lessons Learned

Modeling

As a group, we spent a lot of time modeling our OSV. Even through the planning stage of this project, we ran through the iterative process multiple times in an attempt to deviate our OSV from the tried and true yet non-innovative approaches to solve the mission aspect of the project, and also give ourselves a design challenge through the construction stage of the class. While modeling, we used a lot of physics based logic to arrange and rank our ideas, however reflecting on our actions, they lacked real comparable quantitative values and could have used supplemental research to compliment our imagination of the ideal OSV.

One of our earliest ideas were those of using the tennis-ball collecting type system to pick up the black box. Unfortunately, we were not able to carry through with our plan to use two of our four 12V motors to drive the collection system. Since we had to resort to a 4 wheel direct drive system, we were limited to only being able to use servos to run our collection system if any at all. This hurdle in our plan resulted in a multi-week long setback as we did not have a plan to achieve the advanced objectives. One lesson we learned through this experience was to not discount any idea from the beginning, and always keep back-ups handy. Had we anticipated some of the hurdles we may have run into, we may not have gotten so delayed on creating this crucial aspect of our OSV design.

Also regarding modeling, many members of our team gained a lot of essential CAD knowledge. Throughout the semester, 5 out of 7 members of the team designed and printed OSV parts that ended up on the final product of the OSV. None of these printed components were perfect the first time around, and regardless of whether the software used was Autodesk Inventor or Autodesk Fusion 360, the members of TGTWAPTT got a lot of practice designing and printing different components for our OSV.

Prototyping

One of the most important aspects of our OSV's development was how many iterations its parts went through before we made our final device. While all of the parts went through this

process in some capacity, some of the parts where iteration is most apparent include the motor mounts, chassis, and the pick-up mechanism.

The motor mounts we used on the final chassis were 3D printed casings that fit perfectly to the form of the motors and were secured to the chassis with four screws. The original design, however, was not so elegant. In addition to not actually fitting the motors, causing extra stress on the part, there was not an opening in the back of the part for the wires of the motor to come out. After another attempt at a similar design with updated dimensions and a hole in the back, we decided to use sets of smaller bands that screwed in at two points in order to cut down on the time they would take to print. While these bands worked for preliminary tests, they were flimsy and less precise than we needed them to be in order for our OSV to drive straight. To remedy this, we made our final design for the motor mounts which appear on our device and worked without fail. By rapidly developing new designs and understanding the difference between a temporary fix and a well-designed part, we were able to make a part on our OSV that worked perfectly.

The chassis of our OSV had a fairly consistent design throughout the build phase of the product, but we still updated it several times, increasing the precision we made it with each time. Our first chassis was cut by eye on the scroll saw and the holes were drilled by hand. While we did measure before cutting and drilling, we were much more concerned with making it fast rather than making it accurately. This meant we could make our OSV move by the end of the first week, but it also meant the design could not be permanent. After making our final design for the motor mounts, we drew out, cut and drilled a much more precise chassis. However this process was still done with the shop tools and was imperfect. The device could still not move in a perfectly straight line. This led our group to decide to have our final chassis CNC milled out of high density particleboard. The final chassis fit perfectly with the mounts and made it so our OSV could travel perfectly straight. This is a case where it may have been better for our group to commit to making a final design earlier in the process. By dedicating significantly more time to making a more precise chassis by hand, we wasted a significant amount of time considering we would choose to have the same exact design be milled anyways.

One of the biggest changes we made to the original design of our OSV was the method of collection we would use for the black box. Initially, we planned on using spinning wheels to pull

the box up onto rollers which would lift it off the ground. We quickly discovered after making a proof of concept model out of cardboard that this mechanism would only work if the black box was directly flush with the device. We quickly re-designed the mechanism to use two prongs at a 90 degree angle to take advantage of the small lip at the bottom of the black box. This fork slides under the lip then rotates back. Because the lip goes slightly past the center of gravity of the box, it the box will go with the fork. In testing, this mechanism, when implemented, not only consistently picked up the black box, it did so regardless of the initial angle of the box, making it more effective at completing the mission. By using a mock-up before building the real thing, we were able to quickly discover a flaw in our OSV's design and make changes without wasting much time attempting to make a part that wouldn't work.

Troubleshooting

We had to work on troubleshooting our OSV numerous times throughout the semester, up until the time of the actual performance test, and numerous times after the test as well.

Troubleshooting our OSV included using multimeters to check where in the circuit current was and wasn't actually flowing, retracing our wiring steps to find the sources of the problems,

One of the more common mistakes we encountered was wiring. One particular instance that happened extremely often was forgetting to common ground all of our components together (i.e., Uno, L298N, and batteries). Without doing so, our components would not receive the signals and power that they required, and this resulted in a lot of frustration because we would forget how necessary this simple step is.

Another common mistake was having faulty conditions within the code. Some sections of our code account for multiple specific scenarios. If there were any inconsistencies in these scenarios, e.g. by having conditions that did not match the intended scenario, the resulting action would not match the expected movement. A related mistake occurred at the beginning, where we did not recognize the need to include a range of possible values within the conditions. For example, instead of the condition for an action being when theta equals 0, the implemented condition needed to be something like when theta is between -0.2 and 0.2.

An important lesson we, as a team, learned from this experience was that as much as

we'd like to believe that we would be the group to create the perfect OSV without any issues, such a scenario is highly unlikely. It took many hours to undo the the mistakes that had taken us hours to make initially. A significant percentage of the time spent on modeling, building, and programming the OSV was troubleshooting to find the new issue that arrived that day. But, it was in this portion of the process that we learned the most. Having to understand why certain aspects of our OSV didn't worked forced us to better understand the individual components of the OSV, as well as how these components blended together as a system.

When working on the programming, we learned that writing small portions of code at a time to test on the hardware was very helpful, as the lack of debugging capabilities in the Arduino IDE made it difficult to determine which specific parts of a large chunk of code were failing. In addition, saving old versions of code allowed us to more easily determine whether any problems were due to hardware or software. These old versions also provided us with base code that we could use for multiple different navigation algorithms, making the process of redesigning after a failed iteration slightly simpler.

Teamwork and Communication

Communication between members of the coding team was one aspect that was necessary to the success of the OSV. Being able to understand each other's code was critical in allowing multiple people to contribute to the OSV's software in a productive manner. It is also important to communicate with the other programmers about which aspects of the code each person will be writing and how they plan on the program to logically run. If this step is not taken seriously, the team members may find themselves writing a program that was already written by another member, or writing a program that is incompatible with the other aspects of the program. We tried our best to stay on the same page by copying the code into a folder in Google Drive, then changed to GitHub in the hopes of making the process of updating and sharing code a little bit easier. However, it was not entirely successful, as the process was still time consuming and inconvenient due to the Arduino IDE lacking the capability of connecting directly to GitHub. If this linking capability existed, updates to the code would be automatically added to the GitHub repository. Instead, we had to manually upload files to share them, and copy and paste every time

we wanted to add a new version to our personal Arduino libraries. We tended to forget to update the code after every single version, so we had lapses in synchronization between programming team members.

Communication between those building the OSV and those programming is critical in creating an OSV of a specific design. Without knowledge of the OSV's design and the way in which it will carry out the specific mission at hand, it is virtually impossible for the programmers to write a program that will fit the OSV's desired function.

Communication between all team members about each individual's role in creating the OSV is also very important. We learned from experience that it is very difficult to be productive in building the OSV without a structured division of specific and general tasks for each teammate. It was also very difficult to be productive at times when any future tasks needed to be done were contingent on the completion of one very long and difficult task that could only be done by a certain subgroup of the team. At these times, we would try to prepare ourselves for the tasks ahead, but we would often find ourselves without much to do, waiting for a single task to be finished. Identifying these long tasks early on and communicating this information with the team would be helpful in maintaining a constantly productive and efficient team. Identifying these tasks could allow different individuals to take upon themselves a greater amount of easier job, allowing the members of the specific subgroup to spend more time on these longer tasks earlier.

Project Management

Our original plan involved daily check-ins and updates to the team Google Drive folder and the Trello backlog, so that everyone on the team could know what everyone else was doing at any time, if they desired to do so. However, after a while, this started to fail as we slowly stopped updating both. In retrospect, this brought about a lot of confusion and time wasted that was spent updating each other on our various activities and verbally planning out what still needed to be done each class. Organization is essential to having an even and efficient distribution of roles and labor, so without it, our team struggled to work divided in a way that would allow us to complete our tasks and milestones as quickly and effectively as possible.

Appendix A

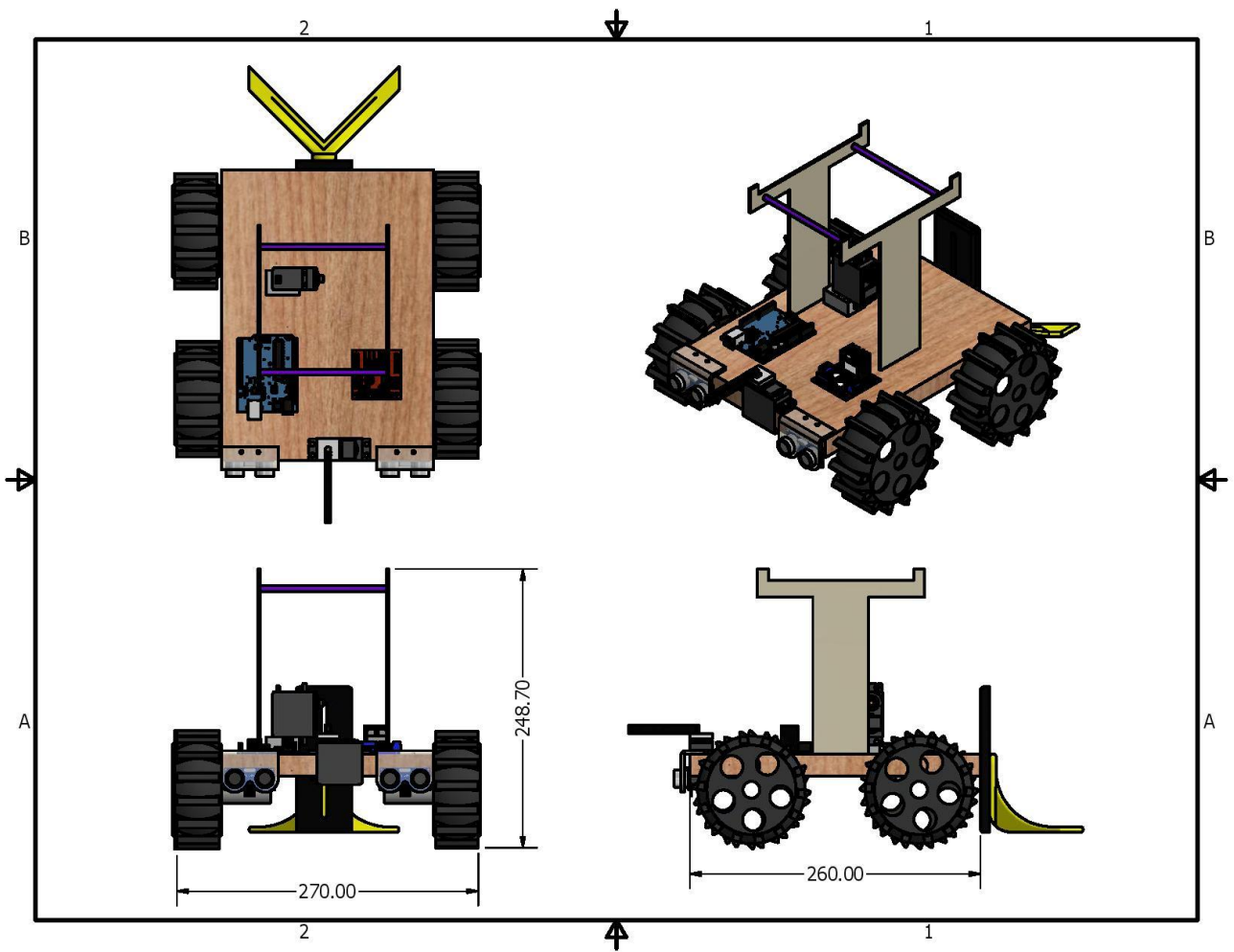


Figure 1: Dimensioned CAD of Final Design

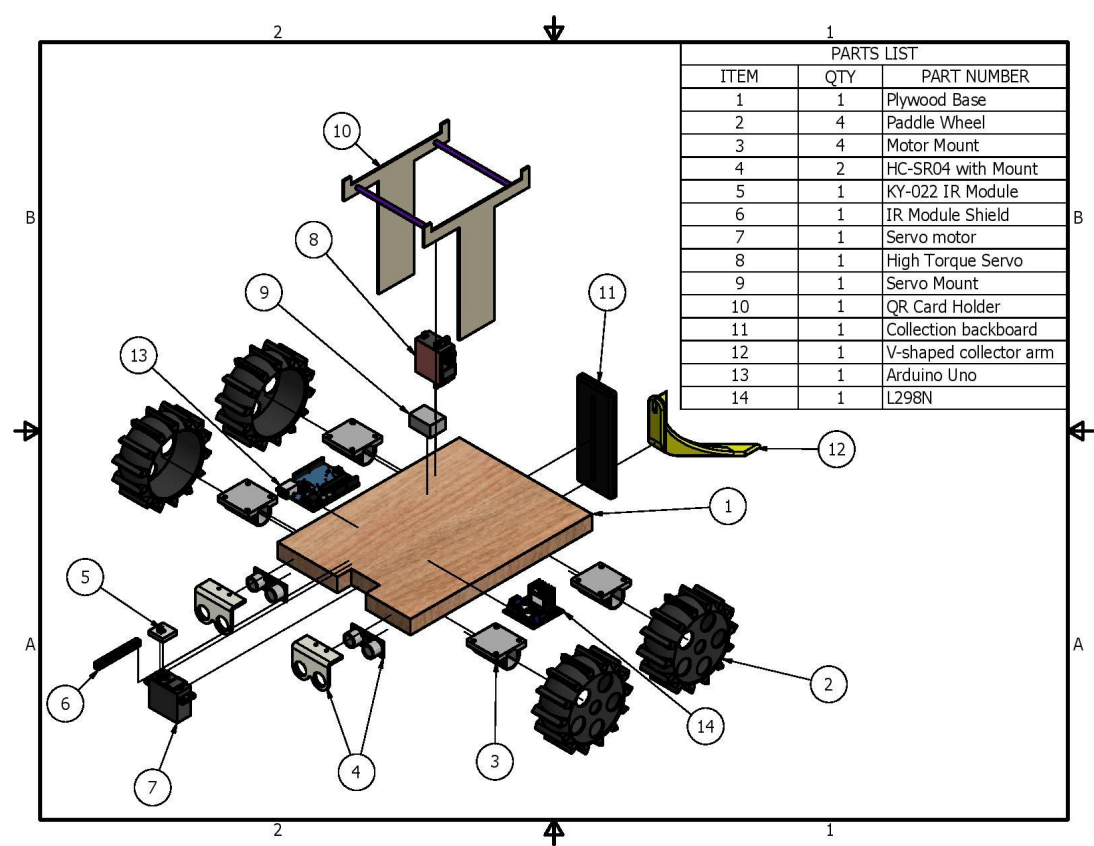


Figure 2: Exploded View CAD Drawing of Final Design

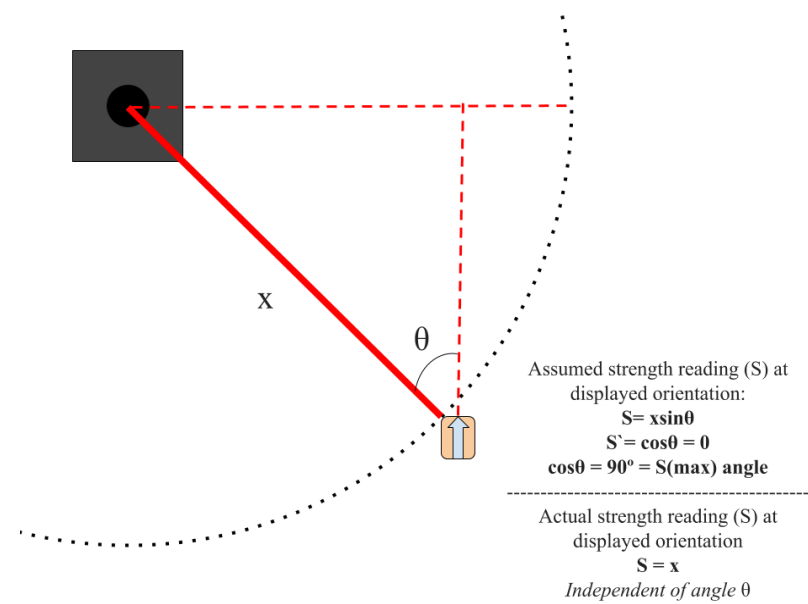


Figure 4: IR Sensing Schematic

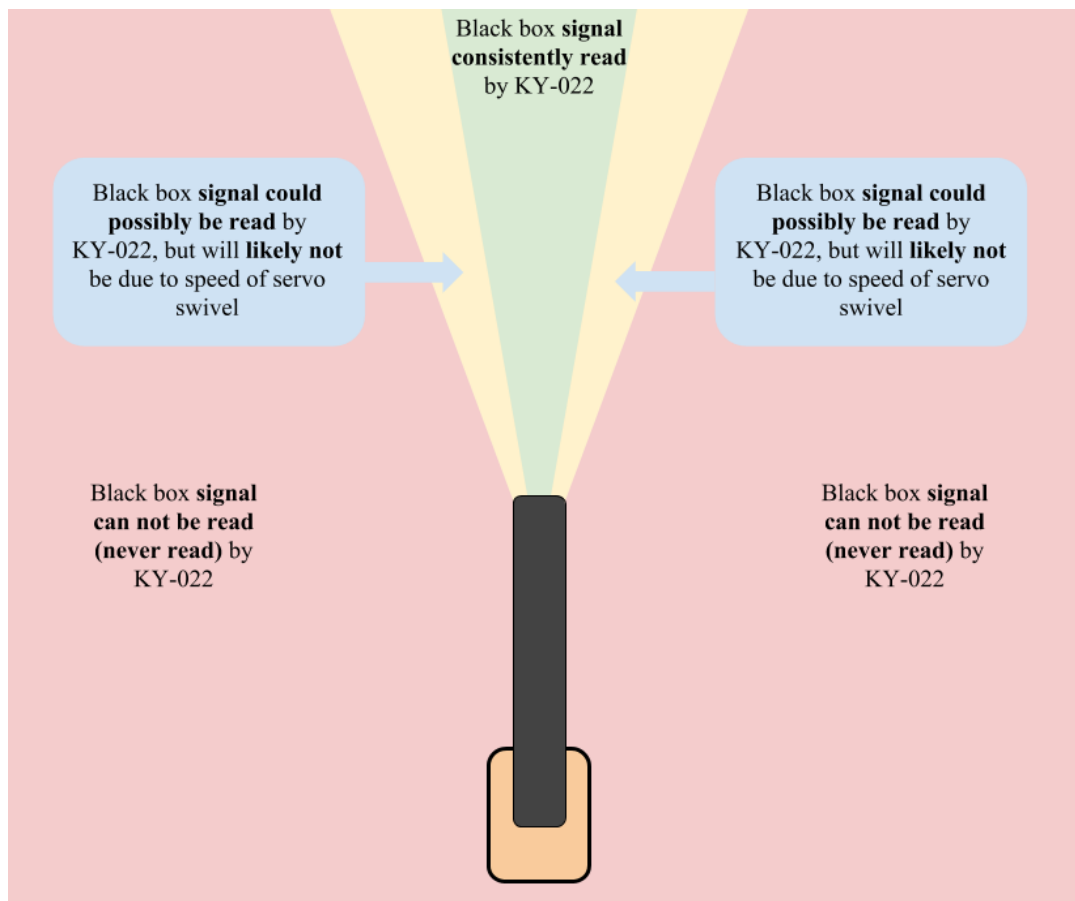


Figure 6: IR Sleeve Sensing Schematic

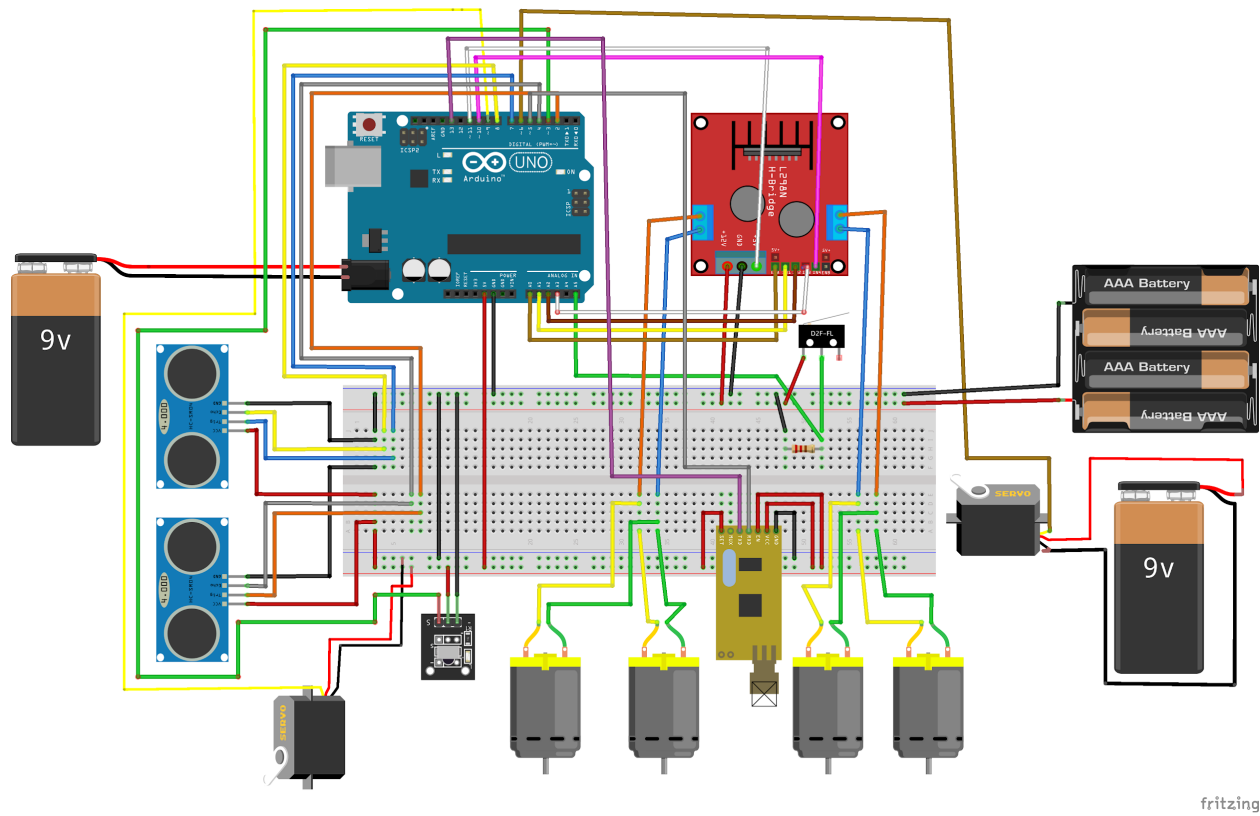


Figure 7: Final Circuitry Schematic

Appendix B: Meeting Minutes

Black Box Group
MEETING 1 09/09/18

09 SEPTEMBER 2018 / 1:00 PM / ROOM 2100B McKeldin Library

ATTENDEES

Carson Hoover, Caitlin Lee, Advait Bantval, Jack Irvin, Katie Crawford

ABSENT

Sahar Farajun (excused), Gianna Moore (unexcused)

AGENDA

Organize Minutes (1-1:15)

Create a Backlog on Trello (1:15-2:15)

10-3-1 (2:15-

Determine Nav/Mission Roles -> Decided to move until next

Meeting Schedule

Group Norms

Brainstorm for the Project

NOTES

- **Caitlin is good at programming but doesn't like it**
- **We need to schedule a research timeline**
- **Someone needs to research programs for autonomous navigation → Advait and Katie**
- **Research methods of finding the Black Box → Advait and Katie**
- **Someone needs to become an expert in Arduino coding → Sahar, Caitlin, Jack**
 - **Some kind of way of recording and transmitting location → Gianna**
- **Research how to locate device with camera → Carson**
 - **How device will communicate with the camera to orient itself**

- **We have decided to go for the extra credit in the project (in case we can't get an accurate enough location for the black box)**
- **DEFINITION OF DONE-** By the end of the week we will each have a strong understanding of topic assigned and all possible options that apply to OSV
 - Is able to give recommendation on course of action based on understanding
- **Group Norms**
 - In order to keep accountability, everyone will record how much time they spend daily on the project in the Trello on their action card

ACTION ITEMS

- 1. Each member will complete the assigned task on the Trello**
- 2. We will all keep up to date with the assigned flipped content videos for the class**

NEXT WEEK'S AGENDA

1. Start dissecting past OSV projects
 - a. Compare to our research
 - b. Decide which will be most applicable
2. Discuss each group member's strengths and weaknesses to determine group roles and how the mission will be split up

09 SEPTEMBER 2018 / 1:00 PM / ROOM 2100B McKeldin Library

ATTENDEES

Carson Hoover, Caitlin Lee, Advait Bantval, Jack Irvin, Katie Crawford

ABSENT

Sahar Farajun (excused), Gianna Moore (unexcused)

AGENDA

Organize Minutes (1-1:15)

Create a Backlog on Trello (1:15-2:15)

10-3-1 (2:15-

Determine Nav/Mission Roles -> Decided to move until next

Meeting Schedule

Group Norms

Brainstorm for the Project

NOTES

- **Caitlin is good at programming but doesn't like it**
- **We need to schedule a research timeline**
- **Someone needs to research programs for autonomous navigation → Advait and Katie**
- **Research methods of finding the Black Box → Advait and Katie**
- **Someone needs to become an expert in Arduino coding → Sahar, Caitlin, Jack**
 - **Some kind of way of recording and transmitting location → Gianna**
- **Research how to locate device with camera → Carson**
 - **How device will communicate with the camera to orient itself**
- **We have decided to go for the extra credit in the project (in case we can't get an accurate enough location for the black box)**

- **DEFINITION OF DONE-** By the end of the week we will each have a strong understanding of topic assigned and all possible options that apply to OSV
 - Is able to give recommendation on course of action based on understanding
- **Group Norms**
 - In order to keep accountability, everyone will record how much time they spend daily on the project in the Trello on their action card

ACTION ITEMS

- 1. Each member will complete the assigned task on the Trello**
- 2. We will all keep up to date with the assigned flipped content videos for the class**

NEXT WEEK'S AGENDA

1. Start dissecting past OSV projects
 - a. Compare to our research
 - b. Decide which will be most applicable
2. Discuss each group member's strengths and weaknesses to determine group roles and how the mission will be split up

MEETING 09/04

20 SEPTEMBER 2018 / 6:00 PM / 2100G

ATTENDEES

Carson Hoover, Caitlin Lee, Jack Irvin, Advait Bantval, Gianna Moore, Sahar Farajun

AGENDA

- Plan OSV (1.0) build
- Plan MS1
 - Responsibilities/roles

NOTES

- Picked motors, motor drivers, piece of wood for platform, possibly 8 batteries, 4 sand/paddle wheels
- Discussed team organization
- Changing to physical backlog

ACTION ITEMS

1. Order parts by next friday (9/28)
2. Individuals make as many slides as possible, as much material as possible
3. Meet again to organize and sort into presentation slides and hidden slides

NEXT WEEK'S AGENDA

- Meet again to finalize MS1

TGTWAPTT

MEETING NAME 09/28

28 SEPTEMBER 2018 / 3:00 PM / ROOM 2100B

ATTENDEES

Caitlin Lee, Carson Hoover, Gianna Moore, Sahar Farajun, Katie Crawford, Jack Irvin

AGENDA

- Organization
 - Trello
 - Gantt chart
 - Scheduling
 - Check subteams/roles
- Suspension
- Parts list/plan to order
- Plan build

NOTES

- No Gantt chart
- Use when2meet.com
- Jack moved to build team for Level 1
- Suspension: currently no suspension, may change if it becomes a problem later
- Parts list:

Part	Quantity	Source	Price
Wood (chassis)		Katie	\$0
Wheels	4	Unsure	\$35
Motor Mounts	4	3D printing	
Batteries			
IR Receiver	2	Adafruit	2 x \$2 = \$4
Stepper Motor	2	Unsure	Unsure

●

ACTION ITEMS

1. Gianna, Katie, Sahar, Advait, Jack: watch weekly videos
2. Sahar: research “mapping” for navigation, start pseudocode for movement
3. Tentative assignment of roles for MS2/3:
 - a. Plans (project management, construction/testing), mission: Carson
 - b. Structure: Katie, (advait)
 - c. Control algorithm, sensors/actuators: Caitlin, Sahar
 - d. Power: Jack
 - e. Propulsion: Advait
 - f. Anticipated difficulties: Gianna
4. Caitlin: research stepper motors, start pseudocode for movement

NEXT WEEK’S AGENDA

- Work on MS2 & 3

Appendix C - Final Code

```
#include <IRremote.h>

#include <IRremoteInt.h>

///  
//include <IRremoteTools.h>

#include "Enes100.h"

#include <Servo.h>

Enes100 enes("What's Wrong With U??!!?!11!?!ONE", BLACK_BOX, 11, 5, 13); //enes("name",  
MISSION, image_num, connect to pin 5, connect to pin 13)

int trigL = 2; //left ultrasonic

int echoL = 4;

int trigR = 7; //right ultrasonic

int echoR = 8;

int distL;

int distR;

Servo IR_servo; //IR sensor system

int IR_servo_pin = 9;

int RECV_PIN = 3;

IRrecv irrecv(RECV_PIN);

decode_results results; // decode_results class is defined in IRremote.h

Servo BB_servo; //BB collection servo

int BB_servo_pin = 6;
```

```
//limit switch = A5
```

```
int enA = 11; /* PMW control speed motor 1*/      // motor one
```

```
//in1 = A0; /*Output HIGH or LOW for direction */
```

```
//in2 = A1; /*Output LOW or HIGH for direction */
```

```
int enB = 10; /* PMW control speed motor 1*/      // motor two
```

```
//in3 = A2; /*Output HIGH or LOW for direction */
```

```
//in4 = A3; /*Output LOW or HIGH for direction */
```

```
int pwm_turn = 250;
```

```
int pwm_move = 200;
```

```
void forward(int sp){                          //move forwards
```

```
    // motor 1
```

```
    digitalWrite(A0, HIGH);
```

```
    digitalWrite(A1, LOW);
```

```
    digitalWrite(enA, 180);
```

```
    // motor 2
```

```
    digitalWrite(A2, LOW);
```

```
    digitalWrite(A3, HIGH);
```

```
    digitalWrite(enB, 180);
```

```
    delay(200);
```

```
    off();  
}  
  
void turnCCW(int sp){                                //turn counter clockwise  
    // motor 1; turn right side forwards  
    digitalWrite(A0, HIGH);  
    digitalWrite(A1, LOW);  
    digitalWrite(enA, sp);  
  
    // motor 2; turn left side backwards  
    digitalWrite(A2, HIGH);  
    digitalWrite(A3, LOW);  
    digitalWrite(enB, 255);  
  
    delay(200);  
    off();  
}  
  
void turnCW(int sp){                                  //turn clockwise  
    // motor 1; turn right side backwards  
    digitalWrite(A0, LOW);  
    digitalWrite(A1, HIGH);  
    digitalWrite(enA, sp);  
  
    // motor 2; turn left side forwards  
    // motor 2  
    digitalWrite(A2, LOW);  
    digitalWrite(A3, HIGH);
```

```
digitalWrite(enB, 255);

delay(200);
off();
}

void reverse(int sp){                                //move in reverse
// motor 1
digitalWrite(A0, LOW);
digitalWrite(A1, HIGH);
digitalWrite(enA, 180);

// motor 2
digitalWrite(A2, HIGH);
digitalWrite(A3, LOW);
digitalWrite(enB, 180);

delay(1000);
off();
}

void off(){                                          //motors off
digitalWrite(A0, LOW);
digitalWrite(A1, LOW);

digitalWrite(A2, LOW);
digitalWrite(A3, LOW);
```

```
}
```

```
bool is_up(float theta){                                     //check if OSV facing up (theta >= 0)
    if(theta >= 0){
        return true;
    }
    return false;
}
```

```
bool is_down(float theta){                                  //check if OSV facing down (theta < 0)
    if(theta < 0){
        return true;
    }
    return false;
}
```

```
bool is_right(float theta){                                 //check if OSV is facing right (-PI/2 <= theta <= PI/2)
    if(abs(theta) <= PI/2){
        return true;
    }
    return false;
}
```

```
bool is_left(float theta){                                  //check if OSV is facing left (PI/2 < theta <= PI OR -PI/2 >
theta >= -PI)
    if(abs(theta) > PI/2){
        return true;
    }
}
```

```
}  
return false;  
}  
  
void turn_y_1(int sp){           //turn to face y = 1  
    if(enes.location.y > 1){     //above y = 1, turn to -PI/2  
        enes.println("Above y = 1");  
        if(is_left(enes.location.theta)){  
            enes.println("\tFacing left. Turning CCW");  
            while(enes.location.theta > 0 || enes.location.theta < (-1*PI/2)-0.4){  
                turnCCW(sp);  
                enes.updateLocation();  
            }  
            off();  
        }  
        else{  
            enes.println("\tFacing right. Turning CW");  
            while(enes.location.theta > 0 || enes.location.theta > (-1*PI/2)+0.4){  
                turnCW(sp);  
                enes.updateLocation();  
            }  
            off();  
        }  
    }  
}  
  
else{           //below y = 1, turn to PI/2  
    enes.println("Below y = 1");
```

```
if(is_left(enes.location.theta)){
    enes.println("\tFacing left. Turning CW");
    while(enes.location.theta < 0 || enes.location.theta > (PI/2)+0.4){
        turnCW(sp);
        enes.updateLocation();
    }
    off();
}
else{
    enes.println("\tFacing right. Turning CCW");
    while(enes.location.theta < (PI/2)-0.4){
        turnCCW(sp);
        enes.updateLocation();
    }
    off();
}
}

void move_y_1(int sp){                                //move to y = 1
    while(fabs(enes.location.y-1)>0.1){                //forwards until within 5 cm of y = 1
        forward(sp);
        enes.updateLocation();
    }
    off();
}
```



```
while(enes.location.theta > 0 || enes.location.theta > (-1*PI/2)+0.4){  
    turnCW(sp);  
    enes.updateLocation();  
}  
off();  
}
```

```
void turn_up(int sp){                                //turn up (theta = PI/2)  
    while(enes.location.theta > 0 || enes.location.theta > (-1*PI/2)+0.4){  
        turnCW(sp);  
        enes.updateLocation();  
    }  
    off();  
}
```

```
bool ultrasonic(){                                //run ultrasonics  
    int rDistTot = 0;  
    int lDistTot = 0;  
    for(int i = 0; i < 3; i++){  
        //left ultrasonic  
        digitalWrite(trigL, LOW);    //clear trig  
        delayMicroseconds(2);  
        digitalWrite(trigL, HIGH);    //left US on  
        delayMicroseconds(10);  
        digitalWrite(trigL, LOW);    //left US off  
        long tL = pulseIn(echoL, HIGH); //time to receive signal  
        int d = tL * 0.034 / 2;        //dist to object
```

```
lDistTot += d;

//right ultrasonic
digitalWrite(trigR, LOW);    //clear trig
delayMicroseconds(2);
digitalWrite(trigR, HIGH);   //right US on
delayMicroseconds(10);
digitalWrite(trigR, LOW);    //right US off
long tR = pulseIn(echoR, HIGH); //time to receive signal
d = tR * 0.034 / 2;          //dist to object
rDistTot += d;
}

distL = lDistTot/3;
distR = rDistTot/3;
if(distL > 0 || distR > 0){
    return true;
}
return false;
}

void ultrasonic_movement(){
    if(distL < 30 && distR < 30){    //if both sense, depends on y position
        off();
        enes.println("left and right detected");
        delay(1000);
        if(enes.location.y < 1){      //if below y = 1, turn up, move 30 cm
            enes.println("below y = 1, turning up");
```

```
    turn_up(pwm_turn);          //turn up (PI/2)
    delay(500);
    float safe_y = enes.location.y + 0.3;
    enes.println("moving up");
    move_up_y(pwm_move, safe_y);    //move 30 cm up
    off();
}

else{          //else, turn down, move 30 cm
    enes.println("above y = 1, turning down");
    turn_down(pwm_turn);          //turn down (-PI/2)
    delay(500);
    float safe_y = enes.location.y - 0.3;
    enes.println("moving down");
    move_down_y(pwm_move, safe_y);    //move 30 cm down
    off();
}

delay(500);
enes.println("turning to theta = 0");
turn_0(pwm_turn);          //turn to theta = 0
delay(500);
off();
}

else if(distL < 30){          //if left senses, turn down, move 15 cm
    off();
    enes.println("left detected");
    delay(500);
    enes.println("turning down");
```

```
turn_down(pwm_turn);          //turn down (-PI/2)
delay(500);
float safe_y = enes.location.y - 0.15;
enes.println("moving down");
move_down_y(pwm_move, safe_y);    //move 15 cm down
off();
delay(500);
enes.println("turning to theta = 0");
turn_0(pwm_turn);              //turn to theta = 0
delay(500);
off();
}
else if(distR < 30){           //if right senses, turn up, move 15 cm
  off();
  enes.println("right detected");
  delay(500);
  enes.println("turning up");
  turn_up(pwm_turn);           //turn up (PI/2)
  delay(500);
  float safe_y = enes.location.y + 0.15;
  enes.println("moving up");
  move_up_y(pwm_move, safe_y);  //move 15 cm up
  off();
  delay(500);
  enes.println("turning to theta = 0");
  turn_0(pwm_turn);           //turn to theta = 0
  delay(500);
```

```
    off();
}
}

void move_up_y(int sp, float safe_y){           //move in positive y direction, until y = safe_y
    while(enes.location.y < safe_y - 0.05){
        forward(pwm_move);
        enes.updateLocation();
    }
    off();
}

void move_down_y(int sp, float safe_y){         //move in negative y direction, until y = safe_y
    while(enes.location.y > safe_y + 0.05){
        forward(pwm_move);
        enes.updateLocation();
    }
    off();
}

void turn_theta(int sp, float theta){          //turn to input theta
    while(fabs(enes.location.theta - theta) < 0.4){
        turnCCW(sp);
        enes.updateLocation();
    }
    off();
}
```

```
float IR(){
    float deg;
    for(int c = 0; c < 3; c++){
        for(int i = 0; i < 180; i++){ //writes servo from 1 to 180 deg
            IR_servo.write(i);
            delay(60);
            if (irrecv.decode(&results)) { //if IR signal received, stop servo, break loop
                deg = i;
                return (i * PI/180)-(PI/2);
            }
        }
    }
    return -1; //return radians of IR signal
}
```

```
bool destination() { //checks if limit switch triggered
    if(digitalRead(A5) == LOW){
        return true; //returns true if triggered
    }
    return false; //returns false if not triggered
}
```

```
void dest_cheat() { //cheatsy BB destination code
    enes.navigated();
    enes.println("Destination location: ");
    float x, y;
```

```
for(int i = 0; i <= 12; i++){
    x = 2.25 + (i*0.1);
    for(int j = 0; j <= 6; j++){
        y = 0.5 + (j*0.1);
    }
    enes.print("(");
    enes.print(x);
    enes.print(", ");
    enes.print(y);
    enes.print(") OR ");
}
}

//IRrecv irrecv(RECV_PIN);      //set IR module pin

//decode_results results;

void setup() {
    pinMode(enA, OUTPUT);
    pinMode(enB, OUTPUT);
    pinMode(A0, OUTPUT);
    pinMode(A1, OUTPUT);
    pinMode(A2, OUTPUT);
    pinMode(A3, OUTPUT);
    pinMode(A5, INPUT);          //Limit switch
    pinMode(trigL, OUTPUT);
    pinMode(trigR, OUTPUT);
```

```
pinMode(echoL, INPUT);
pinMode(echoR, INPUT);
irrecv.enableIRIn();

IR_servo.attach(IR_servo_pin); //IR servo
IR_servo.write(0);

BB_servo.attach(BB_servo_pin); //BB servo
BB_servo.write(0);
enes.println("setup: connected");
delay(2000);
}

void loop() {
  while(enes.updateLocation()){
    enes.println("loop");
    enes.print("");
    enes.print(enes.location.x);
    enes.print(", ");
    enes.print(enes.location.y);
    enes.println("");
    if(enes.location.x < 1.2){ //before x = 1.25
      enes.println("Before x = 1.2. Turn to y = 1.");
      turn_y_1(pwm_turn); //turn to face y = 1
      delay(1000);
      enes.println("Move to y = 1");
      move_y_1(pwm_move); //move to y = 1
      delay(1000);
      enes.println("turn to theta = 0");
```



```
turn_0(pwm_turn);           //turn to theta = 0
delay(1000);
enes.println("move over rocks");
move_rocks(pwm_move);        //move over rocks (to x = 1.25)
delay(1000);
turn_0(pwm_turn);           //readjust: turn to theta = 0
enes.updateLocation();
}

if(enes.location.x < 2.2){    //before x = 2.25 --> obstacle avoidance
    enes.println("before x = 2.2");
    if(ultrasonic()){        //if obstacle, object avoidance
        enes.println("object detected");
        ultrasonic_movement();
        off();
    }else{
        enes.println("no object detected");
        forward(pwm_move);    //no obstacle, move forward
    }
}

else{
    off();
    delay(1000);
    float BB_rad = IR();
    if(BB_rad == -1){
        dest_cheat();
        delay(10000);
    }else{
```

```
while(enes.location.x < 3.5 && !destination()){ //within BB area

    forward(pwm_move);

    if(ultrasonic()){                //if obstacle, object avoidance

        ultrasonic_movement();

        off();

    }else{

        forward(pwm_move);           //no obstacle, move forward

    }

}

off();

while(!destination()){              //while limit switch not triggered, move up a little

    turn_up(pwm_turn);               //turn up (PI/2)

    off();

    delay(500);

    float y = enes.location.y;

    move_up_y(pwm_move, y + 0.25);    //move up 25 cm

    turn_theta(pwm_turn, PI);         //turn left (PI)

}

while(enes.location.x > 2.5 && !destination()){ //while limit switch not triggered, moving back
towards x = 2.5

    if(ultrasonic()){                //obstacle avoidance

        ultrasonic_movement();

        off();

    }else{

        forward(pwm_move);           //no obstacle --> continue moving forwards

    }

}
```

```
}  
off();  
while(!destination()){           //while limit switch not triggered, move halfway down BB area  
    turn_down(pwm_turn);         //turn down (-PI/2)  
    off();  
    delay(500);  
    float y = enes.location.y;  
    move_down_y(pwm_move, y - 0.5); //move down 50 cm  
    turn_theta(pwm_turn, PI);     //turn left (PI);  
}  
while(enes.location.x < 3.25 && !destination()){ //while limit switch not triggered, moving towards x  
= 3.25  
    if(ultrasonic()){             //obstacle avoidance  
        ultrasonic_movement();  
        off();  
    }else{  
        forward(pwm_move);        //no obstacle --> continue moving forwards  
    }  
}  
off();  
if(destination()){               //limit switch triggered, print message  
    enes.navigated();  
    enes.print("Black Box found at: (");  
    enes.print(enes.location.x);  
    enes.print(", ");  
    enes.print(enes.location.y);  
    enes.println(")");
```

```
delay(1000);

float rev = enes.location.theta - PI; //determine theta for rear to face BB

if(rev < -PI){
    rev = rev + (2*PI);
}else if(rev > PI){
    rev = rev - (2*PI);
}

forward(pwm_move-30);

delay(500);

while(fabs(enes.location.theta - rev) < 0.2){ //turn to reverse angle

    turnCCW(pwm_turn);

    delay(400);

    enes.updateLocation();

}

off();

delay(500);

reverse(pwm_move-20); //reverse, slower than normal

delay(2000);

off();

delay(500);

for(int i = 270; i >= 0; i--){ //run BB collection servo

    BB_servo.write(i);

    delay(40);

}

while(fabs(enes.location.theta - PI) < 0.2){ //turn to face theta = PI

    turnCCW(pwm_turn);
```

```
    enes.updateLocation();
}
for(int i = 15; i <= pwm_move; i++){          //accelerate to pwm_move from 0
    forward(i);
}
while(enes.location.x > 0.65){                //navigate back to landing zone
    if(ultrasonic()){                          //obstacle avoidance
        ultrasonic_movement();
        off();
        for(int i = 15; i <= pwm_move; i++){    //accelerate
            forward(i);
        }
    }else{
        forward(pwm_move);                      //no obstacle --> continue moving forwards
    }
}
off();                                         //DONE
delay(10000);
}
else{                                         //limit switch not triggered
    off();
    dest_cheat();                             //cheatsy code
    delay(10000);                             //DONE
}
}
}
```