My Chemical Robot

Milestone 9 – Final Design Report

Instructor: Dr. Ehrman

ENES100 (0101): Introduction to Engineering Design

Elizabeth Childs

Ben Douty

Marc Harron

Brian Inzerillo

Bailey Murtha

Christoph Neisess

Yair Pincever

Eric Ramming

May 15, 2017

Table of Contents

Executive Summary

The purpose of this project is to make an Over Sand Vehicle (OSV) that is at most 350 mm by 350 mm that can navigate over sand. Additionally, as the chemical neutralization team, we must have our OSV do several additional tasks. At the beginning of the competition, the OSV must navigate from a random spot with a random orientation behind a wall. After clearing the wall, it must first maneuver itself within 250 mm of our target, the acidic chemical pool. Upon reaching the pool, it will take a 10-15 mL sample from the target using a pump connected to a storage container. From the pool, the OSV has to read the pH of the pool to 0.3 units of accuracy. Based on the reading, the OSV will then neutralize the acidic target with a baking soda, leaving the pH of the pool in between 6 and 8.We are using a 3D printed arm with a pH meter attached at the end to measure the final pH. To accomplish these tasks, the team decided to utilize a two-level base structure. The top floor is reserved for items that will be used during the process of the mission such as the: pH meter, 3D printed arm, baking soda, and collection tube, while on the bottom we will have the hardware necessary to complete the mission such as the Arduino, battery, motors, motor shield, etc. Additionally, the motor mounts are larger than required for the motors so that they will cover the motors and keep out any sand that the wheels may kick up; otherwise the sand could potentially damage the hardware needed to complete the mission. Overall, our OSV proved moderately successful at completing its mission. Individually, all of the systems worked very well, but we ran into some issues once they were all combined and mounted to the OSV. The three biggest issues we encountered with the OSV during operation were getting it to turn smoothly, get close enough to the chemical pool to actually take a pH reading, and deliver the neutralizing agent without missing the pool.

Introduction

Over the course of the semester, our team designed an Over Sand Vehicle (OSV) to complete a specific mission. The mission we were assigned was chemical neutralization, which consisted of the following tasks: get within 250 mm of the target site (the target site being an acidic chemical pool); take a 15 mL sample from the target; take the pH reading of this target within 0.3 units; and neutralize the acidic target with a base, either washing soda or baking soda. In addition, our OSV had to have a footprint smaller than 350x350mm, not weigh more than 3kg, and not have a replacement cost higher than 350 dollars.

In order to navigate to the chemical pool, the OSV had to first determine its starting location and orientation, then navigate around an obstacle wall. After moving past the wall, it was programmed to drive directly to the chemical pool to complete the remaining objectives.

To complete the mission we designed a moveable arm to lower the mission equipment into the chemical pool. There are two possible positions for the arm to be in, either vertical (sticking straight up, perpendicular to the OSV body) or horizontal (sticking out from the vehicle, parallel to the OSV body). A pH meter was mounted to the arm that dips into the target to read and send the pH back to command. We also have a tube mounted to the arm that is connected to a pump. This is for collecting the 15 mL sample out of the chemical pool. For neutralization, we designed a small servo-mounted container that held baking soda. The OSV was programmed to turn around and then dump the baking soda into the pool after taking the initial pH reading.

To hold the components of our OSV, we designed a structure that has two levels. The lower level houses the components needed to power and drive the OSV. This includes the 4 motors, the battery, and the motor shield. On the top level, we have our mission arm, pump, and the servo for operating the dump container. The container itself and the container used to hold the sample are mounted to the back of the OSV behind the pump. Finally, our distance sensor is mounted to the front of the OSV between the two front motor mounts. Figure 1 shows a rendering of the final OSV.

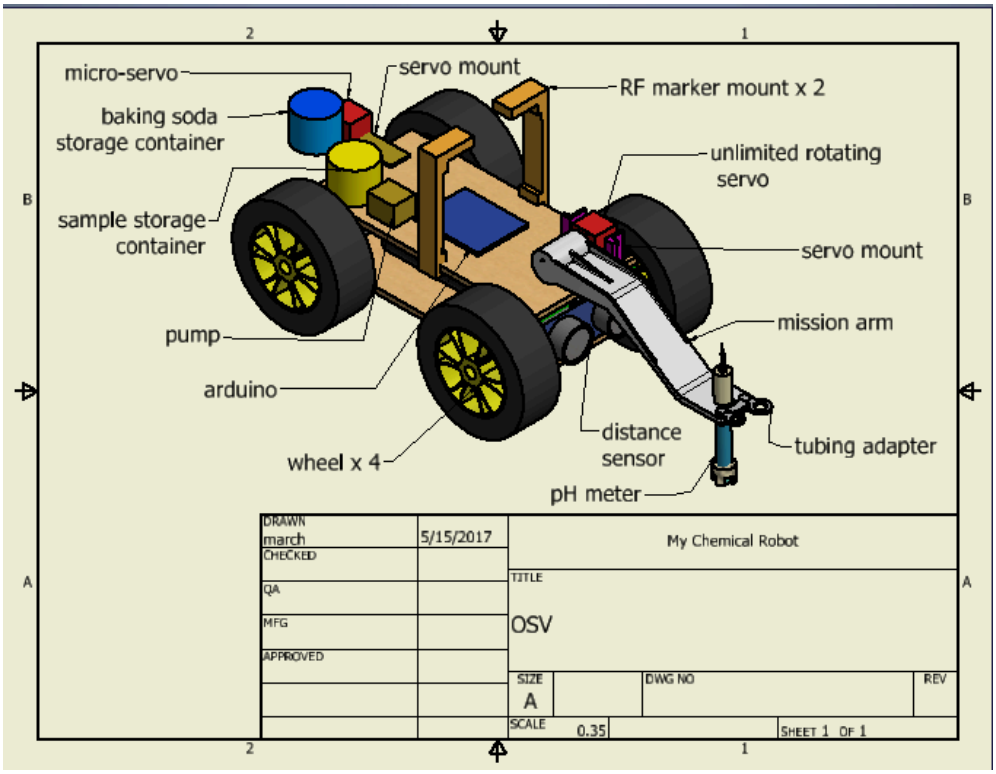


Figure 1: OSV with labeled components

Preliminary Design Shortcomings

The arm originally was supposed to hold the tubing and the pH meter. However, after the arm was printed and the tubing was bought, it was discovered that the tubing was too big and would not fit in the arm. As a result, we had to CAD an insert for the arm that allows the larger tubing to be held. Additionally, since our initial method for neutralization changed from pumping a solution of baking soda to dumping solid baking soda, our CAD (which initial had two tubing holes for neutralization and sample collecting) was only utilized for one hole.

We had planned to use a mini servo with a stall torque of 1.8 kg/cm in order to raise and lower the arm with the pH meter into the chemical pool. However, when calculating the amount of torque needed to raise the arm, we miscalculated the weight of the arm and pH meter combined, and the mini servo did not have enough torque to raise the pH meter. As a result, we ended up buying a new servo with a torque of 5.5kg/cm. However, the new servo was a 360°

unlimited servo, therefore we had to code the exact timing the servo should run instead of coding in a degree.

_____ There were two main issues with the initial design of the power system. First, the motor shield was occasionally not able to provide enough current to the motors, which became most obvious when the OSV attempted to turn via differential steering This is discussed in more detail in the performance evaluation section. In short, this was fixed by writing a special piece of code to control turning the OSV so that it wouldn't dig the wheels in too far. Second, the initial idea for a small kill switch on the breadboard of the OSV was thrown out because the switch couldn't handle the current coming through the motor shield. We eventually chose to use the switch from a power strip because it was actually able to handle this current.

Our preliminary plans for the structure of the OSV were relatively unchanged. For instance, the bi-level plywood concept was kept. However, there were several minor alterations to our original plans. First, we expanded the width of each plywood layers by 2 cm. This change allowed us to have more room for our components and prevented the leads of the motors on one side of the OSV from colliding with those on the other side. Secondly, we decreased the separation height between the two levels of the chassis by 0.72 cm, compared with our preliminary design. We altered it so that the arm would be completely horizontal, rather than angled downward, when the pH meter (located on the end of the arm) reached the water in the pool. Additionally, we modified our motor mounts to have a cylindrical casing that would cover the motor and ensure it didn't move around.

The majority of propulsion decisions made during the preliminary report were used throughout the entirety of the course. For example, we used differential steering, the same motors, and the same wheels that we detailed in the preliminary report. However, we did make two modifications to these components and systems. The first change came about when we first tested our motors on the OSV at 6 volts. It seemed to be struggling to power through the sand dunes that the wheels would create. Thus, we altered our power configuration so that the motors would be run directly off of the 8.4 volts supplied by our battery. The final change to our propulsion system dealt with our steering method. Because our wheels were digging into the sand when pivoting, we altered our differential steering code. The new system stopped the

pivoting after a second, and then had the OSV drive out of the hole it dug. Then it would pivot again, and reverse out of the new hole. This system is discussed in more detail in the propulsion section.

One of the minor changes to our OSV was how we secured the pH meter. We had initially planned to have a secondary mechanism for holding the pH meter in place, but this problem resolved itself thanks to the tolerances of the 3-D printing; the pH meter fits very snug in the arm, and does not move at all when the OSV performs its navigation or mission. Aside from the lack of need for a way to restrain the pH meter, we did not have to make any changes at all to the pH meter configuration.

A slightly bigger issue we had was determining what container to use to hold the sample. Up until the day before competition, we had planned on using a syringe to hold the sample because we already had it in our collection of parts, and it was easy to integrate into the rest of the system. We changed to the urine sample container the day before the competition because it became evident that it was difficult to empty the syringe once it had a sample in it. The fix was no additional cost, and it made measuring the final sample much simpler.

The one major setback in our mission was the breaking of our system to neutralize the acidic solution. Our initial design consisted of a pump that would deliver the baking soda as a solution. The first unexpected issue we came across was that despite being over a kilogram underweight, we were limited to a volume of about 300mL of solution (roughly 0.3 kg) because of the strain on the motors while turning. If we were to exceed this volume, our OSV would be far more likely to fail in its navigation due to unexpected slipping or inability to move. With this reduced volume allotted, a saturated solution of baking soda would not be enough to neutralize. In the largest and most acidic situation, we would only reach a pH of around 5 with 300 Ml of saturated solution. We tried to use a solution with baking soda suspended in it, but this ended up breaking our pump. We decided to try a gravity powered system, but we could not get a system that rigid or air-tight enough to prevent solution from flowing when it should not. In conclusion, we put a lot of effort into many different methods of neutralization, but in trying so many ideas,

we ran out of time with perfecting any single one, and ended up with a method that is less precise.

The final setback worth noting occurred when we changed our means of neutralization from using a pump to using a servo-powered dispenser. The use of the servo became an issue because all of our digital pins were used up, and the servo is intended to operate with a digital signal. To fix this, we ran the servo on a serial pin with a code that gave power in a way that simulates a digital signal (rapid on and off). By doing this we were eventually able to get the neutralization portion of the mission working.

Final OSV Design

Structure

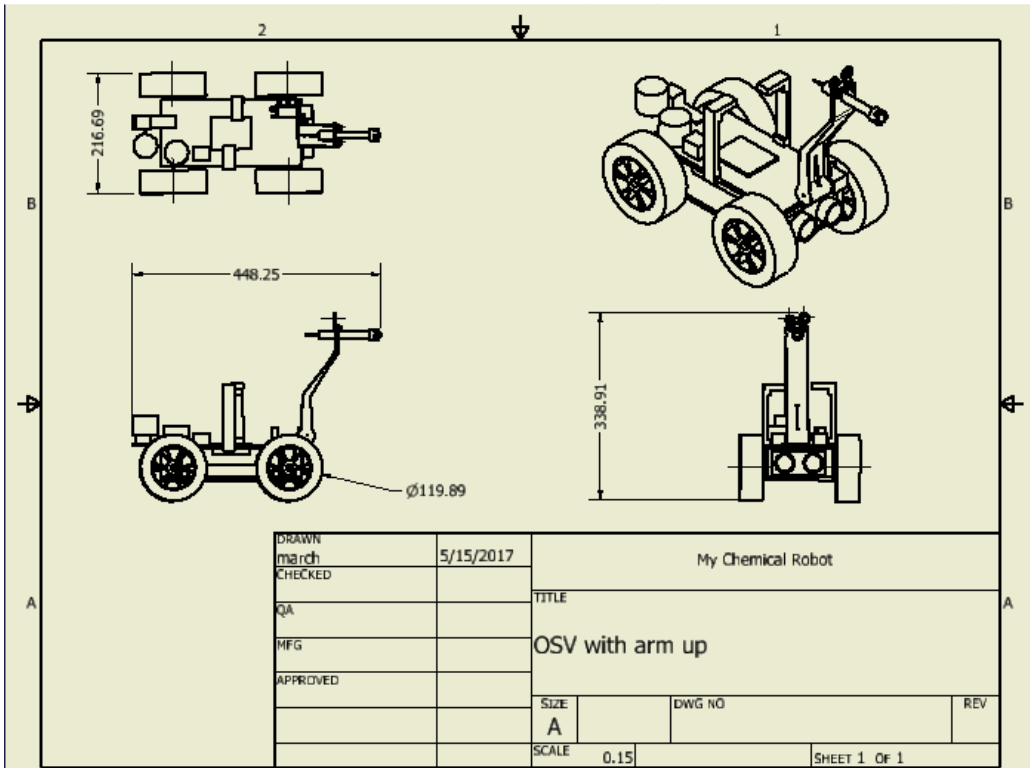


Figure 2: OSV Structure

After brainstorming many design concepts, we decided to go with a bi-level frame. Figure 3 shows the overall layout of the OSV and its major dimensions. We chose this design because it gave us enough room to house all of our components. In particular, it allowed the

mission-specific components to have access to the entire perimeter of the upper level. Furthermore, the center of gravity would remain relatively low since the components with the greatest mass (the motors and the battery) would be housed on the lower level. This bi-level frame is made from quarter inch plywood, and each level has dimensions of 14 cm in width and 25.4 cm in length. We used a table saw to cut a larger sheet of plywood to our desired dimensions. The separation between the two levels is 4.36 cm, and is maintained by four 3D printed motor mounts which double as supports for the bi-level frame. Each 0.64 cm thick motor mount is located in a corner of the frame, and is secured to both the top and bottom level via screws. As displayed in Figure 3 below, the mounts mirror the cylindrical shape of the motors to ensure they don't slide around. Additionally, this motor mount design protects the motors from any possible sand kicked up from the wheels.

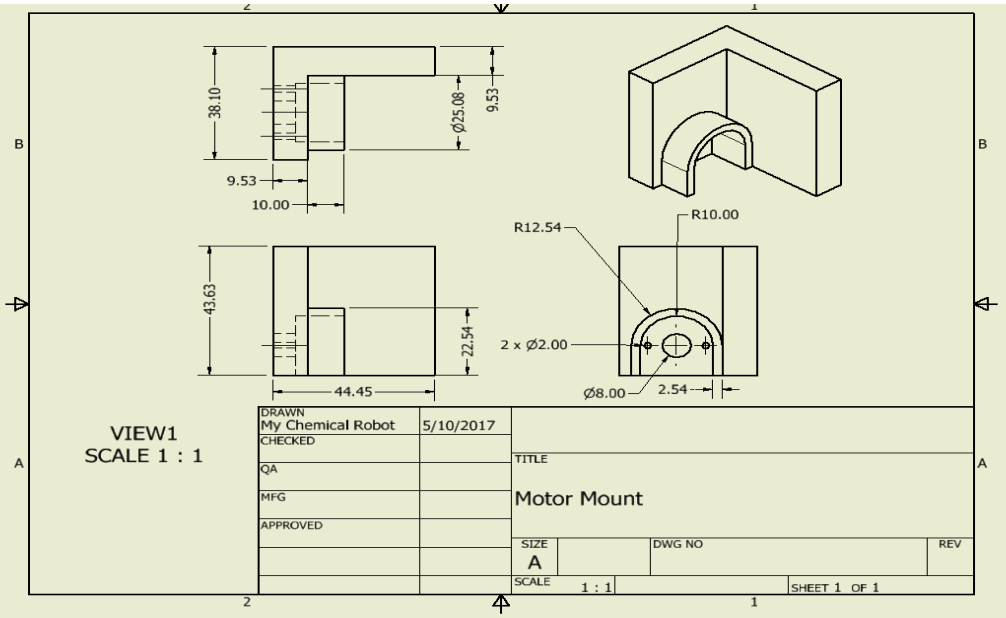


Figure 3: Motor Mount

The bi-level frame was then modified to better accommodate the placement of mission critical components. We also created a rig to hold the RF sensor. We used two 3D printed supports, whose specifications can be seen in Figure 4.

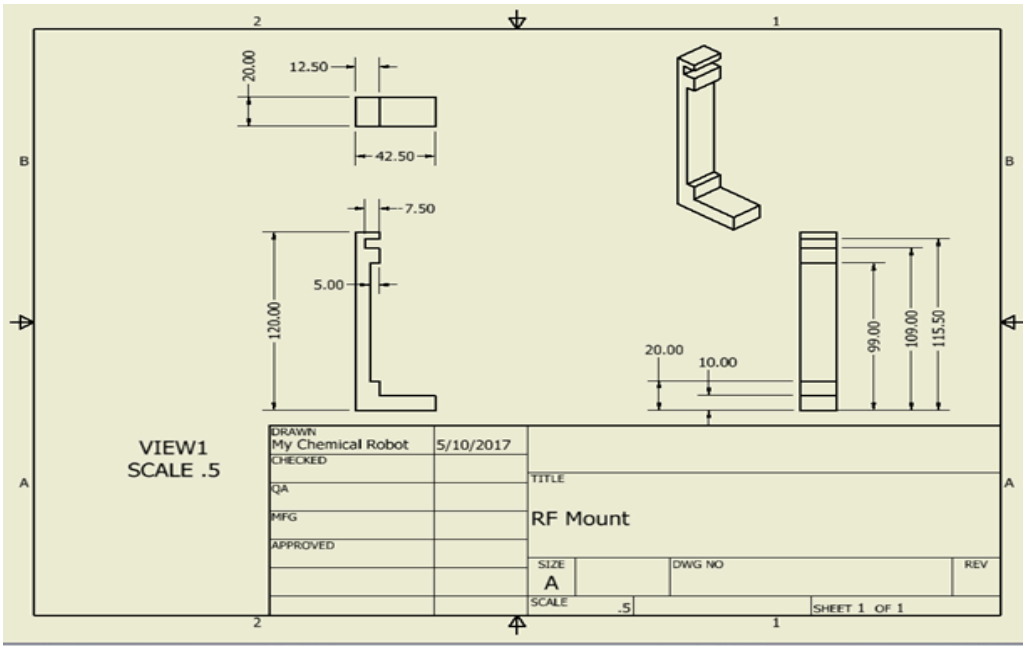


Figure 4: RF Sensor Holder

The following two figures demonstrate where each component is located on the OSV.

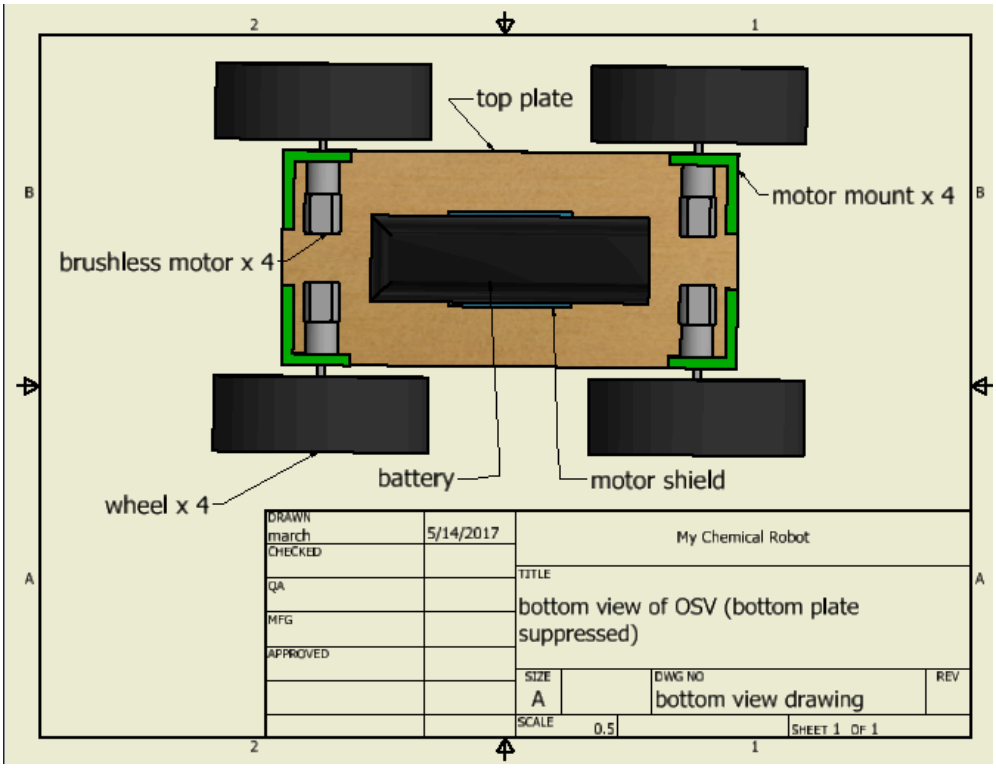


Figure 5: Bottom Level Components on OSV

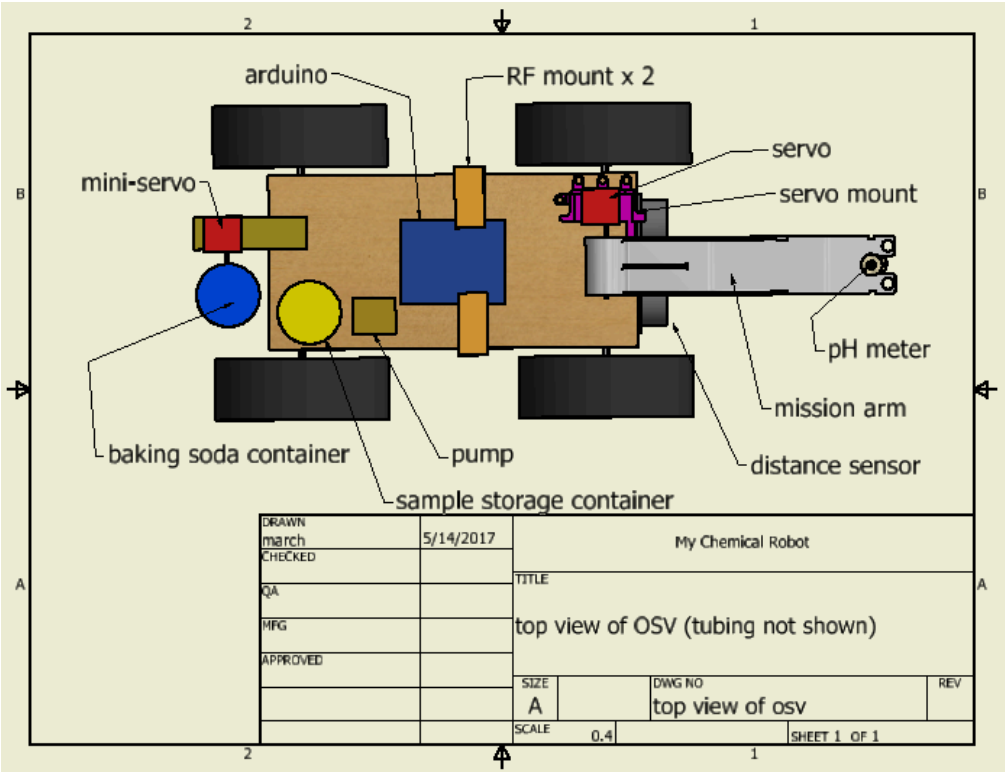


Figure 6: Top Level Components on OSV

After researching several wheel types, we narrowed down our choices to either chevron tires or sand tires. Ultimately, we decided to go with sand tires because their design is specifically tailored for the environment in which we would be operating. Once we decided on the type of wheel, we determined our preferred dimensions. To prevent the OSV from becoming stuck on sand dunes, and to raise our valuable components above the damaging sand, we decided to purchase wheels with relatively large diameters - 12 cm. Moreover, we decided to get tires with relatively large widths- 4.2cm- to ensure the OSV didn't sink into the sand. We debated a dual wheeled configuration to further increase the effective width, but discovered through testing that the single wheeled configuration was sufficient. To connect the wheels to the motors, we decided on a two part axle adapter system, in which an aluminum axle adapter would attach to the 4 mm motor shaft and then convert it to a 12 mm hexagonal shaft. We then 3D printed an axle adapter that went from a 12 mm hexagonal shaft to a 17 mm one to fit the wheels. We also tightened both the 3D axle adapter to the metal axle adapter and the metal axle adapter to the motor shaft with set screws to ensure strong connections.

Propulsion

The following is an in depth analysis of our propulsion systems and the reasoning that controlled our design decisions. After researching tire types and their strengths and weaknesses, we became convinced that sand tires would be optimal for navigating our terrain. Sand tires perform excellently in loose sand environments; however, they can also dig our vehicle into a pit. To get around this issue, we employed two concurrent solutions. First, we used pulse width modulation to slow down tire rotation, which allowed the vehicle to move forward more and dig less. Second, we developed a modified steering code (detailed in Navigation) that keeps the OSV from pivoting in one location, thus preventing a large buildup of sand.

Furthermore, because of their paddle-like design, the wheels are capable of throwing up sand as the vehicle moves. Fully aware of this weakness, we designed our motor mounts, which are located right next to each wheel, to prevent any sand from being thrown into the lower level of the chassis. After deciding on wheel type, we had to determine our desired diameter and width. In addition to structural motivation of wanting large diameters to raise the OSV above the sand, we also wanted large diameters from a propulsion perspective in order to reduce the coefficient of rolling resistance. Each tire will be driven by its own motor, which is located in a corner of the lower level of the chassis. To figure out the necessary motor characteristics required, we analyzed the forces acting on our OSV and determined that the torque needed at each wheel was 21.46 oz*in. Regarding the linear speed of our OSV, we aimed to make it move at 0.25 m/s. Therefore, each motor must spin at 39.79 rpm. All relevant calculations are shown on the next page.

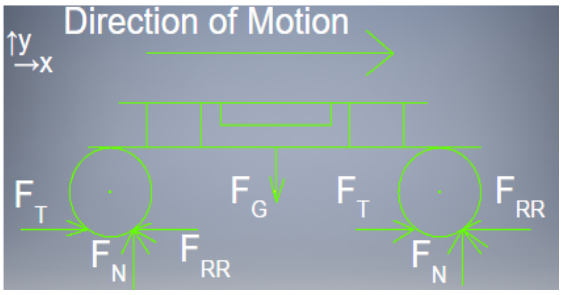


Figure 7: Free Body Diagram, Full Vehicle

$\Sigma F_y = 0$

$4F_N - F_G = 0$

$F_N = \frac{m \bullet g}{4}$

$F_N = 7.35\,N$

$C_{RR} = \left(\frac{3.33 \bullet F_N}{w \bullet d^2}\right)^{1/3}$

$C_{RR} = 0.343$

$\Sigma F_x = 0$

$4F_T - 4F_{RR} = 0$

$F_T = C_{RR} \bullet F_N$

$F_T = 2.52\,N$

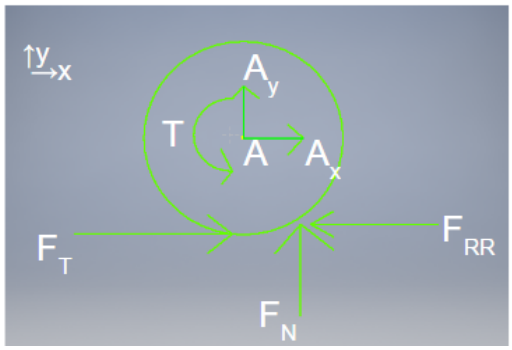


Figure 8: Free Body Diagram, Wheel

$\circlearrowleft \Sigma M_A = 0$

$r \bullet F_T + \tau = 0$

$\tau = - r \bullet F_T$

$\tau = - 15.15\,N \bullet cm \;\; or \;\; - 21.46\,oz \bullet in$

$v = \omega \bullet r$

$0.25 = 0.06 \bullet \omega$

$\omega = 4.17\,rad/sec \;\; or \;\; 39.79\,rpm$

Thus, we set out to find a motor that could satisfy the necessary 21.46 oz*in of torque needed to turn the wheels. Using excel, we tested multiple motors to see which would provide the desired rpm at 21.46 oz*in of torque. Our chosen motor had the following characteristics at 6 volts: 83.5 oz*in stall torque , 51 rpm no load speed, and 1.45 A stall current. To determine our operating point, based on our predetermined torque of 21.46 oz*in, we graphed the specifications and applied the linear formula y=mx+b. Figure 9 shows Torque vs Angular Velocity and Figure 10 shows Current vs Angular Velocity. The operating angular velocity found from Figure 9 was used to determine the operating current in Figure 10.

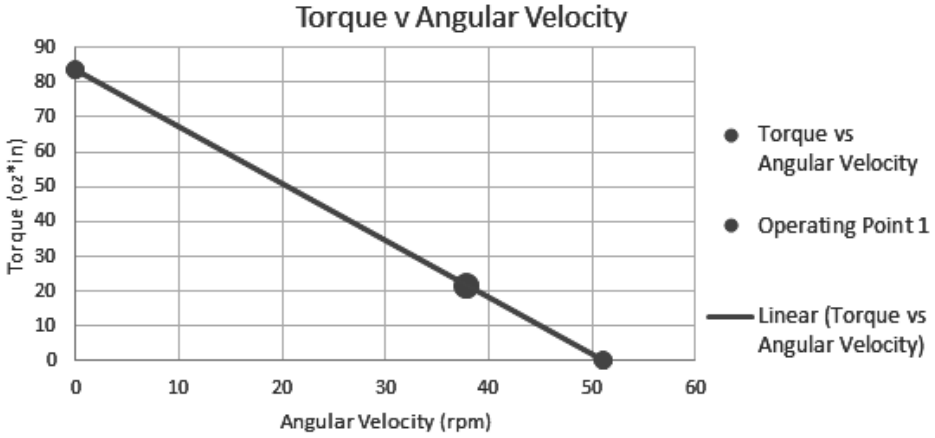


Figure 9: Torque vs. Angular Velocity

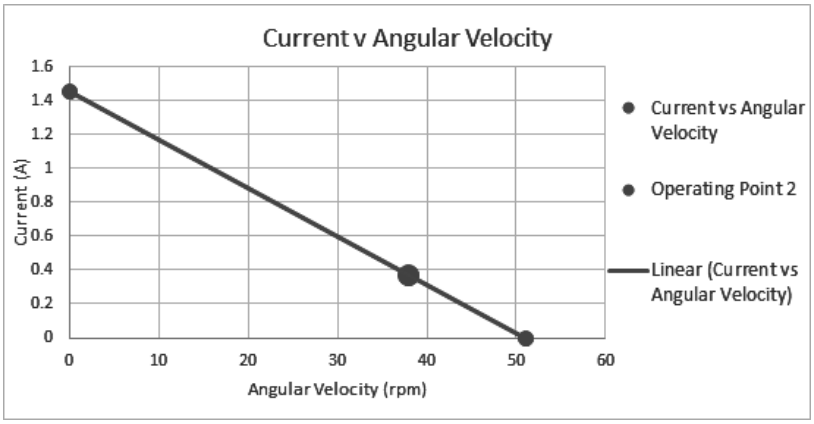


Figure 10: Current vs. Angular Velocity

Using the linear y=mx+b formula and a torque of 21.46 oz*in, operating point 1 and 2 were calculated to be (37.89 rpm, 21.46 oz*in) and (37.89 rpm, 0.37 A) respectively. Because of our careful motor selection, a geared transmission was not necessary to operate our OSV at the desired angular velocity of 39.79 rpm. Moreover, this motor had the ability to be scaled up; it was capable of handling anywhere between 6 and 18 volts. Therefore, we tested the motors on the OSV at 6 volts and, based on its performance, decided to run the motors directly off battery power at 8.4 volts.

We decided on differential steering so that we could make sharper turns. Since differential steering requires wheels to turn both forwards and backwards, we bought brushed motors. As noted earlier, the pivoting motion used in traditional differential steering would cause

our OSV to dig itself into a hole. Thus, after a second or two of traditional pivoting, our OSV would move forward, and then pivot again from that new position. After a few seconds of pivoting in this position, the OSV would back up to the original position and repeat the process until the full turn was achieved.

	Plywood Frame	Motor Mounts	Battery	Wheels	Motors	Water volume	Pumps/ servos	Arm/pH meter
Mass	250g	100g	420g	540g	180g	120g	130g	150g
Total	1.9 kg							

Figure 11: Estimated OSV Mass

Power

In its final configuration, the OSV ended up drawing approximately 3285mA of current. The table below in Figure 12 shows how much current each component drew during operation. This is higher than we initially estimated because we didn't actually have all of our components when we did our initial calculation. The biggest change that occurred between our initial current estimate and our final OSV was enlarging the servo that supports the arm. The unit we ended up selecting drew much more current than what we had originally budgeted for. The battery we chose to use was a seven cell, 8.4 volt nickel metal hydride (NiMH) battery with a 3000mAh capacity. We planned on having approximately one hour of continuous run time with all systems at full power using this battery. During testing, we found that this battery supplied ample power to all of the OSV's systems, and rarely required charging. This was as expected, because the OSV is not actually running all of its systems at full power simultaneously, meaning that the actual battery life is a lot longer than what was estimated.

Item	Current Draw (mA)
Pump	300
Small Servo	200
Big Servo	900
Drive Motors (4)	1600 (400 per motor)
pH Meter	250
Distance Sensor	35

Figure 12: Power Consumption Chart

The OSV's power system delivered the battery's power to the motor shield, which then provides power to a breadboard on top of the OSV. The breadboard sent the power first through a 5 volt regulator and then to the arduino. The breadboard also sent power separately to the sensors, servos, and pump motor. The latter was controlled through use of an analog pin on the arduino and a transistor to make sure it was not running constantly. The motor shield also provides power to the four drive motors via four H-bridges integrated into a pair of L293D motor driver chips. Figure 13 shows the wiring schematic for the OSV when it was finished.

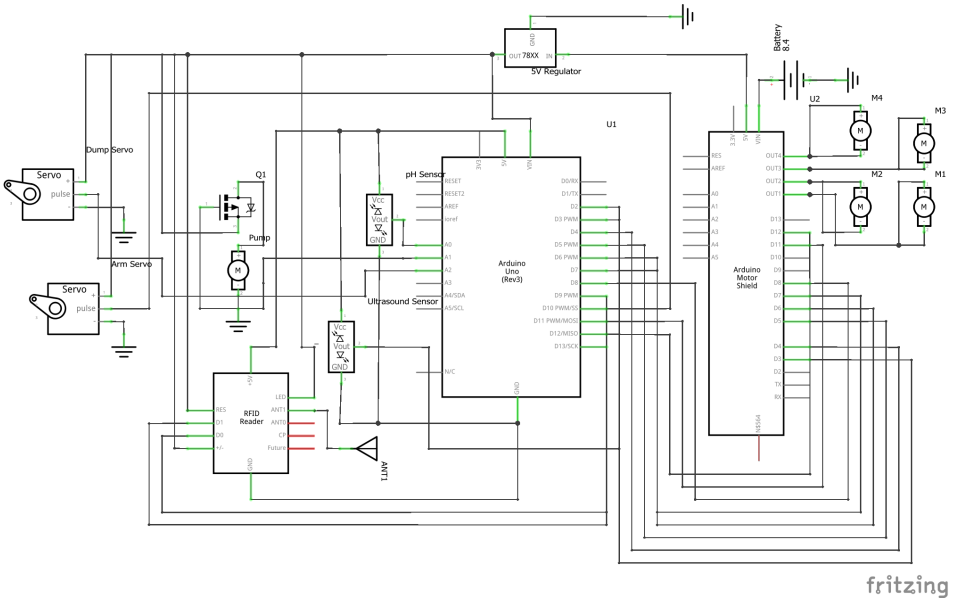


Figure 13: Wiring Schematic

OSV Mission

Our primary mission objectives were measuring/transmitting the pH of the acidic pool and collecting a sample. Once our OSV reached the acidic pool, the arm lowered down from its upright position into the descended position. In this position, the pH meter, which was on the end of the arm (see above), was submerged in the acidic solution. It then read the pH of the pool, and transmitted the pH to command. The pH meter sat in the recessed circular cutout on the end of the arm (see figure 2) . The cutout was sized such that the top portion of the pH meter rests on top of the arm, preventing the pH meter from falling down. The pH meter fits tight enough that no further harnessing was required.

The second part of the mission was collecting a sample. We accomplished this using a pump system. The pump and storage container were located on the rear of the OSV. The pump was self-priming, meaning it is able to pull air, in addition to water. The pump was connected to the storage container by plastic tubing. Also connected to the pump was a length of tubing that goes to the arm, and through a guide on the mission arm, with the end of the tubing lining up with the pH meter. This setup allowed the tube to be submerged in the pool when the

arm was descended, and remain out of the sand when the OSV was moving. The pump ran for about 5 seconds, during which 10-15 mL of solution moved into the storage container.

Up until the day before competition, we had planned on using a syringe to hold the sample because we already had it in our collection of parts, and it was easy to integrate into the rest of the system. However, we changed to the urine sample container the day before the competition because it was difficult to empty the syringe once it was filled with solution. Additionally, using a urine sample container did not significantly change the build cost of the OSV.

Our final design for neutralizing the solution consisted of a urine sample container filled with solid baking soda located at the back of the OSV. This container was attached to a servo, which was then attached to the OSV. After collecting the sample, the OSV turned 180 degrees, backed up, and dumped the baking soda into the acidic pool. While the solution neutralizes, the OSV turned around again to allow the pH meter to descend back in and transmit the final pH. Since all of the baking soda was poured out at once, we did not have a control over how much baking soda comes out. Optimally, we wanted enough baking soda in that it would have be able to neutralize the largest and most acidic solution, but not enough to overshoot titration.When we took the measurements of the pH of the pure baking soda solution, we got a reading of about 8.03. With the upper allowed limit of neutralization being a pH of 8, we were fairly confident that we could still fall within the criteria with our imperfect design even if the pH of the acidic solution is not very low because it would take a massive amount of baking soda to get above a pH of 8. It is worth noting that the expected pH of baking soda is closer to 8.4 . The difference in pH is most likely due to the reaction of sodium bicarbonate with water, forming carbonic acid and carbon dioxide. This reaction naturally occurs when baking soda is exposed to humid air over a long enough period of time.

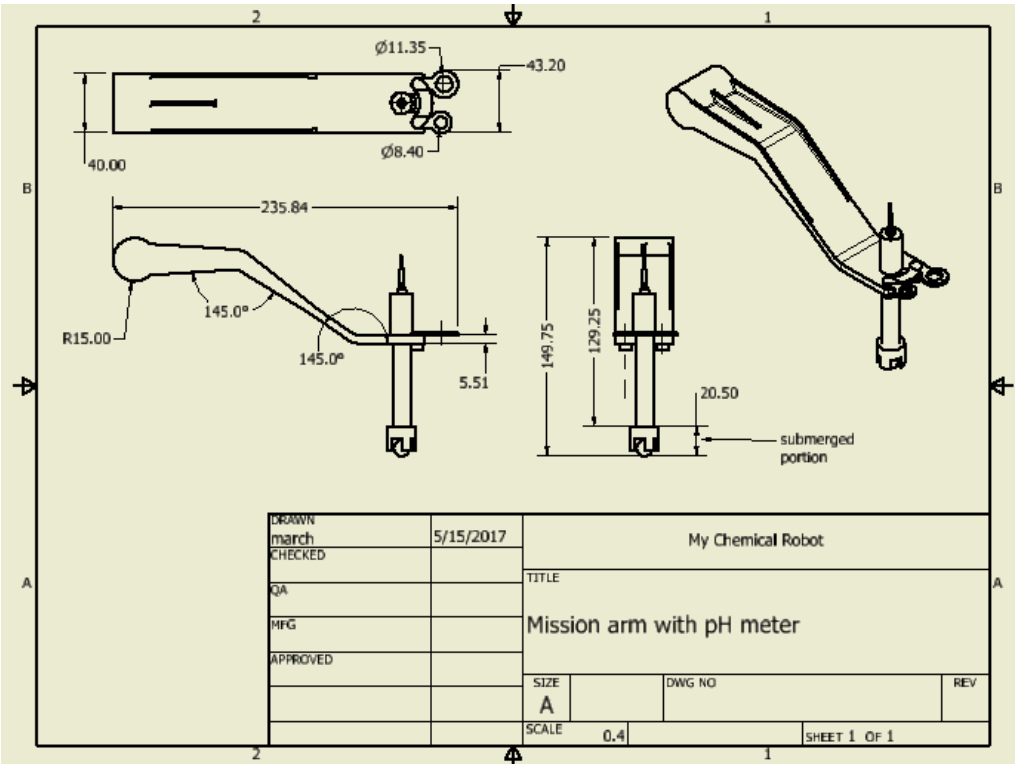


Figure 14: Arm with pH meter attached

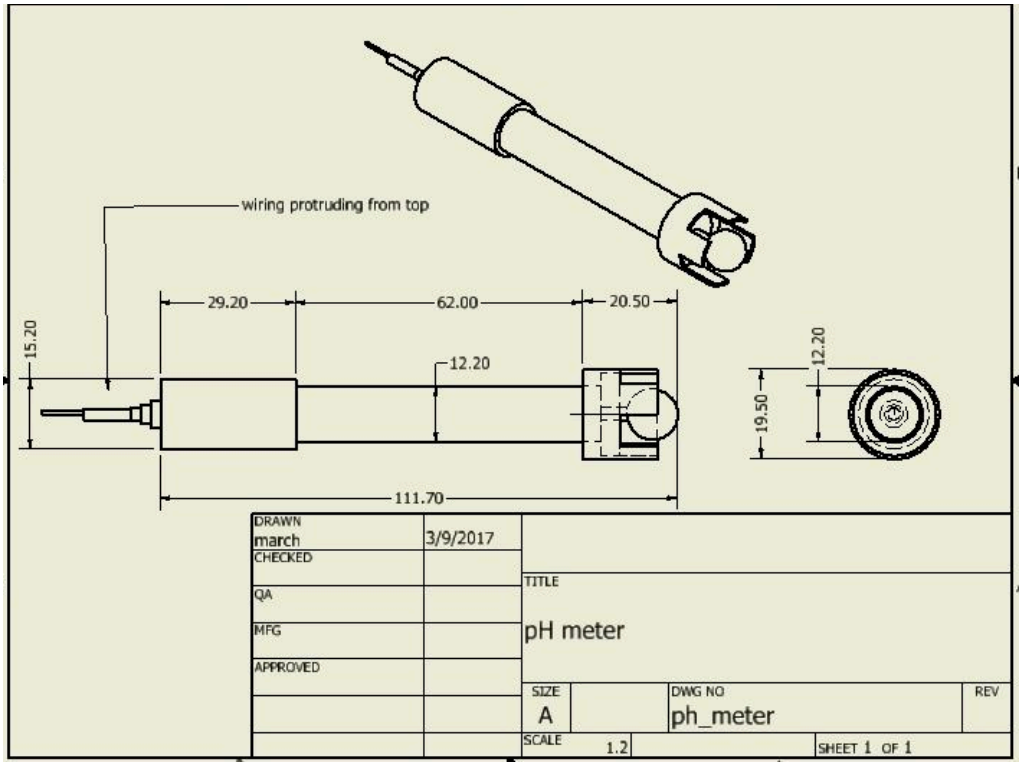


Figure 15: pH meter

Sensors and Actuators

The sensors on the OSV will include a distance sensor and a pH meter . The distance sensor is setup to output different voltages when it is between 2 and 300 cm. It will be located in the front of the OSV, so that the OSV will know when it nears a wall. The arduino can interpret the voltage output to detect the distance of the nearest obstacle.

The pH meter is also on the front of the OSV (on the arm), and when activated by the arduino, continuously senses the pH with an accuracy of .1 pH. However, the arduino will will only transmit the pH after it the continuous recordings are within .01 pH of each other to ensure the pH recording has stabilized.

In terms of actuators, we have a continuous servo that controls the arm and a small servo which delivers the baking soda. The continuous servo is on the front of the OSV to rotates 90° to allow the pH meter and sample collecting tube (located on the arm) to reach the chemical pool. The arduino tells the servo to run for .65 seconds in order to make the 90° rotation. The small servo is on the back of the OSV and rotates 115° in order to drop the baking soda into the chemical pool. The mini servo is controlled through PWM, so the arduino simply tells the servo to rotate 115°. This servo was located on the back because although it requires the OSV to turn around before neutralizing the solution, there was very little space to hold the baking soda in the front of the OSV.

In order to collect the sample we used a self priming pump. It was placed on the back of the OSV as a counterweight to the arm and has a tube running up the arm to enter the pool. In order to collect the 10-15 mL sample, the pump runs for 6.5 seconds.

Control Algorithm

The OSV must be able to autonomously complete its objectives, using feedback only from the sensors it carries onboard and the arena vision system. There are four important values for navigation: The X and Y coordinates, the heading, and the distance to the nearest obstacle in

front of the OSV. To update these values, a function will be written that uses the RF transmitter to update the X, Y and heading. This function will be run at the start of every motion the OSV makes. Another function will exist to return the distance to the nearest obstacle.

For movement, the OSV will need to be able to move forward, backwards and turn in both directions. Moving forward and backwards only required sending the FORWARD or BACKWARD command to the motor shield. Turning, however, was not quite as simple. The motor shield was unable to provide enough current to the motor to allow for differential steering. It would instead move slightly in the desired direction, then dig itself into the sand. To turn, the OSV did this motion for .3 seconds, then moved backwards for .3 seconds, repeated the turn motion for .3 seconds, then moved forward for .3 seconds. This yielded a turn of approximately .05-.1 radians per turn cycle. Using a closed loop error system, the turn function turns right or left based on which direction is the smallest, and zeroes in on the desired angle.

The OSV must first be able to navigate past the Obstacle Wall. Three randomization factors impact this objective: location of the Obstacle Wall(Top or Bottom), initial orientation and initial position. To start, the OSV turn to face the top, and move to the top edge of the arena. After facing the Obstacle Wall, it will check for the Obstacle Wall and if it is present, it will repeat this with the other edge of the arena. If it is not present, it will move forward until it reaches an X position of 1.2 and start navigation to the pool.

To reach the pool, the OSV will use this algorithm to calculate the angle to move to the Chemical Pool.

Turning to this angle, the OSV will move forward until the X or Y location of the pool is crossed by the OSV. At this point, the OSV will descend the arm, and begin to measure the pH. After it is transmitted, it will activate the sampling pump. Once the pump has run, it will ascend the arm, and do a half turn. Then it will back up and dump the baking soda, then return to the pool, descend the arm, and wait for the pH to equalize. It will then transmit this pH, and then end the mission.

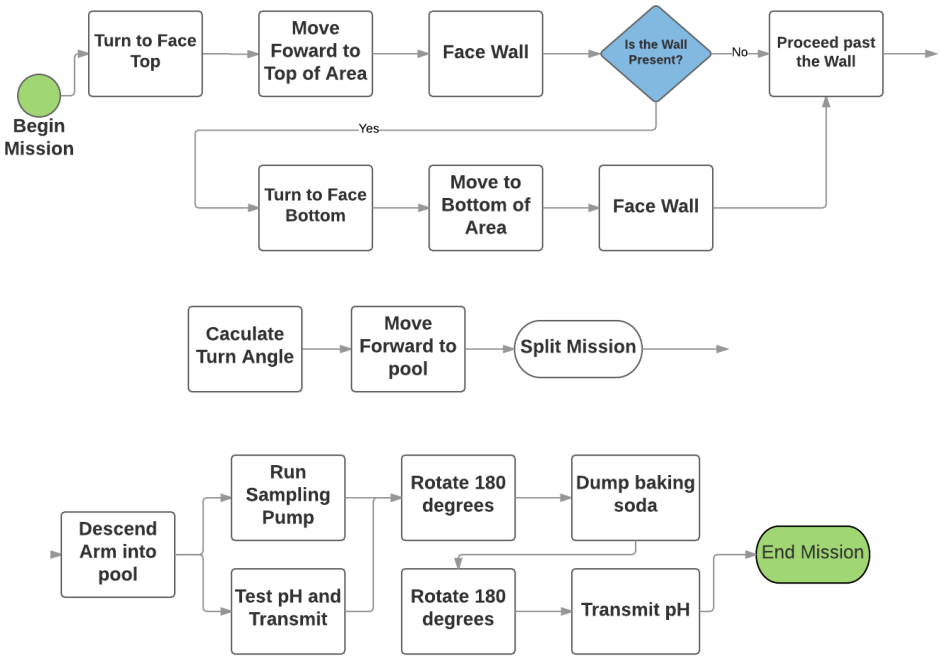


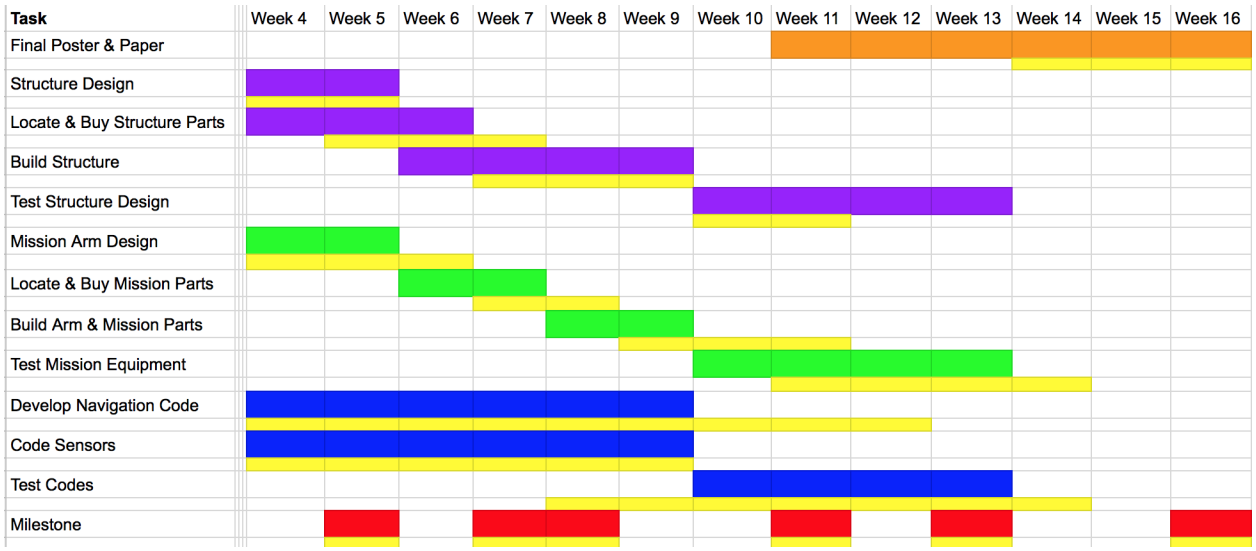
Figure 16: Control Flowchart

Bill of Materials

Item	Cost ($)
Battery	30.04
Tires	31.99
Servos	12.74
Pumps and Tubes	21.11
pH Meter	39.00
Ultrasonic Sensor	10.00
Baking Soda	1.42

Plywood Structure	1.50
Hex Adaptors	20.25
3D Printing	9.72
Arduino and Wires	20.00
Motors	47.00
Total	*253.70*

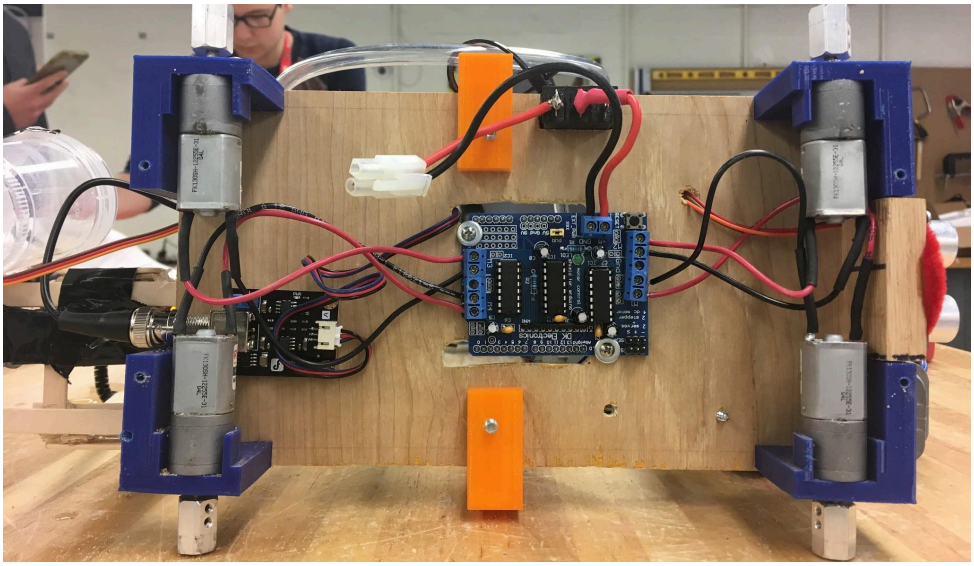
Gantt Chart



Construction Details

Since most of the components on the OSV ended up being complex shapes, much of the OSV was 3D-printed to make it easier to manufacture these parts. To prepare the base, we drilled multiple holes in series to make a rough cut in the top level. We then sanded out the cut to make

it more uniform and precise. After doing the same process to make another parallel cut, we aligned our motor shield so that its pin connections were just below the cuts. Then, we attached our motor shield to the underside of the top level with screws. The cuts we made allowed us to connect wires to the motor shield from the arduino and breadboard that are housed on the top level. Once we had completed installing the motor shield, we added a power switch to the OSV. To do this, we cut the battery's wire and then soldered on our switch. After applying heat shrink to the exposed wire-to-switch connection, we carved a rectangle out of the side of the top level in which the switch could sit. Next, we secured our battery to the middle of the lower level with velcro. We did this so that it could be easily removed for charging, but would remain secure when the OSV was in motion. Following the battery's installation, we secured our distance sensor to the front of the OSV using a piece of wood. The image on the next page shows the structure of the OSV with the bottom removed. This shows how the motors were attached to their mounts as well as how the motor shield was mounted in order to save space.



Bottom view of the OSV with the lower baseplate removed. Note the pass-throughs for the motor shield and ultrasound sensor wires.

Most of the mission components were mounted to the top of the OSV's upper deck. The arduino and breadboard used to distribute power were mounted there as well. The pump as well as the container for the sample were mounted in a small housing that hung off of the back of the top deck. We chose this layout because if we needed to uninstall any of these components later

we could get to them fairly easily. The only exception to this was the controller board for the pH meter, which was mounted underneath the top plate near the motor shield. This was done so we could keep the top deck as free as possible, and so we could declutter some of the wiring on the top of the OSV.

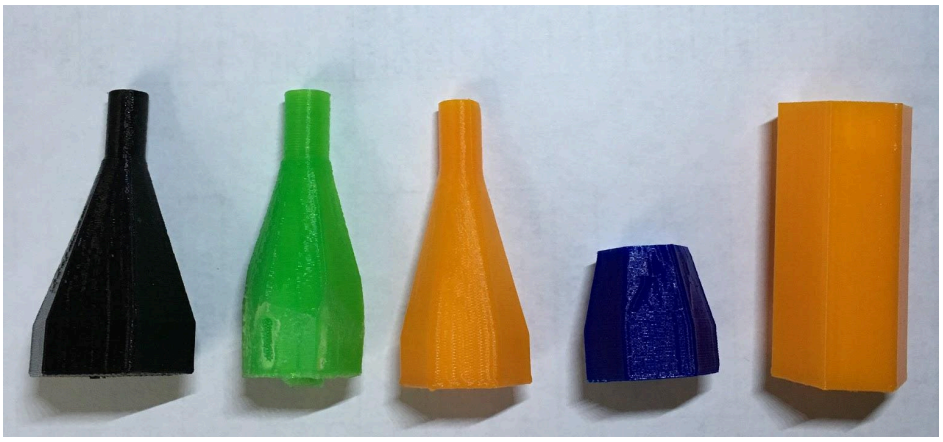
Performance and Evaluation

The arm, which was designed to hold the majority of the mission components, worked well in the end. However, the unlimited servo used to control the arm should have been a PWM servo. This would have been much easier to code, and our team would not have spent the time to discover the exact number of milliseconds needed to rotate the arm 90°.

We originally intended to use a pump to deliver our neutralization solution. This pump did not work for us because it was unreliable. When we were testing the pump to deliver our saturated neutralization solution, which was more of a diluted slurry, the pump broke because the fine sodium bicarbonate particles interfered with the movement of the motor inside. We had a very limited amount of solution we could put on the OSV so we could not reduce the amount of undissolved baking soda, and therefore the pump would not work for us. If given the opportunity to redo the project, we would try to find a way to either make our solution less viscous or we would try to find a pump that would suite our needs better.

There are a few changes that we would make to the structure and propulsion system if we had the opportunity to build the OSV again. First, we likely would spend more time considering how the wheels of the OSV would attach to the motors. This seems like a small detail, but it actually proved to be one of the biggest obstacles to getting our OSV functioning properly, and in the process of development we went through several iterations before arriving at a useable solution. We initially used a single wheel-to-shaft axle adapter, but eventually transitioned to a two part axle adapter system. The single axle adapter system relied on a 3D printed part that would connect the shaft to the 17 mm hexagonal slot in the wheel. However, we couldn't get the 3D printed part accurate enough to grip the motor shaft, so it would get stripped and become ineffective whenever the motor shaft would spin. Had we spent more time considering this

initially, we could have saved valuable development time later, which could have been put into other components on the OSV. Shown below is a selection of the adapters we produced to give an idea of how long this development process took.



Another issue encountered with the structure and propulsion of the OSV came from the motor shield. If we were to build another OSV, we would likely select a motor shield that can send more current to each motor. The setup on our finished OSV provided plenty of power to the motors for straight-line motion, but when it came to turning the large tires and longer wheelbase compared to other designs required much more power than the shield could provide. The solution we came up with involved programming in an elaborate turn sequence to avoid getting the OSV stuck, which again took up valuable time that could have been used for other building and development work. Additionally, while this technically did work (after a fair bit of adjustment) it was far from a perfect fix. A motor shield that could support more current draw would have been a better fix for the issue, and if we had more time we would have switched ours out.

Lessons Learned

Double check all calculations: If we had known that the calculated torque for the arm was incorrect, we would not have bought the wrong servo and needed to buy a new one.
Confer with the entire team before buying a part that affects the team: In the rush to buy a new servo, we did not realize we were ordering an unlimited rotation servo. This made the coding for

the servo harder than expected, and could have been avoided if we had conferred with the entire team first.

Taking time to evaluate risks is important: The decision to put a lot of baking soda through the pump directly led to it breaking. If we had looked at the risk of doing this, we may have avoided breaking the pump, and we may have not had to change our delivery mechanism.

Better Communication between subgroups: Although we worked as a whole team on a lot of the preliminary designing, we split up more as the building of the OSV became more involved. As designs changed for things related to each subgroup, the preliminary design where everything functioned well together became less relevant. One of our biggest issues was the limit on the amount of baking soda solution we could hold due to complications with turning under load. This unanticipated constraint contributed to the issues involving the method of neutralization. Given another chance, we would periodically meet as a whole group and discuss changes that were made to ensure all parts will work well together.

Meeting Minutes

Our team had official team meetings every tuesday. In our first few meetings we split our team into subteams to delegate certain tasks and responsibilities to certain people. From there we all still came to the same meetings, but the subteams would often split up to discuss certain topics that were relevant to their team (ex. While the mission subteam talked about different neutralization methods, the navigation team would talk about their code). We were all in the same place, so we could move between groups and make team decisions. Toward the end of the semester as system integration began, each sub-team met less and much of the work required all hands.

Code Uploaded To OSV

```
#include <enes100.h>
#include <marker.h>
#include <rf_comm.h>
#include <Servo.h>
#include <AFMotor.h>

//This is the master OSV code for the chemical neutralization team
//My Chemical Robot, Section 0101, and is authored by
//Brian Inzerillo and Christoph Neisess and Elizabeth Childs

//pinouts
//All other digital pins are taken by motor shield
#define SensorPin A0    // pin for pH meter
const int servoPin = 10;
const int pingPin = 2;
SoftwareSerial mySerial(9, 13);

//prototypes:
//all functions used within main must be prototyped here
void descendarm();
void ascendarm();
void stopping();
void starting();
void moveForward();
void moveForwardSlow();
void moveBackward();
float readpHMeter();
float abs(float);
void navigation();
float equilibrium();
void nav1();
void dumpSolution();
void turn(float);
void turn1(float,float);
void turn2(float,float,float);
void runSamplePump();
void moveBackward1();
void moveForward1();

// Definitions:
//Don't touch these, except for Offset, if recalibrating pH meter
const float pi = 3.14159;
```

```
#define Offset 1.00            //deviation compensate
#define samplingInterval 20
#define printInterval 800
#define ArrayLenth  40    //times of collection
int pHArray[ArrayLenth];  //Store the average value of the sensor feedback
int pHArrayIndex=0;

//Object Declarations
AF_DCMotor FL(1); // FL short for Front Left
AF_DCMotor FR(2); // FR short for Front Right
AF_DCMotor RR(3); // RL short for Rear Left
AF_DCMotor RL(4); // RR short for Rear Right

Servo armServo;   // create servo object to control a servo

Marker marker(4); //change this to whatever the current marker number is
RF_Comm rf(&mySerial, &marker);

void setup() {
  //everything is either all set or setup within the functions
}

int z = 0;
void loop() {
  while(z == 0) {
    //this while loop prevents the OSV
    //from completing its mission more than once
    //all main code goes here:
    float pH;
    nav1();
    rf.transmitData(CHEMICAL, NAV);
    descendarm();
    pH = equilibrium();
    rf.transmitData(BASE, pH);
    runSamplePump();
    delay(2000);
    ascendarm();

    //rotates the OSV for chemical neutralization
    if(marker.theta + pi > pi) {
      rf.updateLocation();
      turn2(marker.theta + pi, .02, 1);
      rf.updateLocation();
      moveBackward1();
      dumpSolution();
```

```
      delay(100);
      turn2(marker.theta - pi, .02, 1);
   }
   else {
      rf.updateLocation();
      turn2(marker.theta - pi, .02, 1);
      rf.updateLocation();
      moveBackward1();
      dumpSolution();
      delay(100);
      moveForward1();
      turn2(marker.theta + pi, .02, 1);
   }
   while(marker.x < 1.53) {
      moveForward();
    }
    stopping();
    delay(20000);
    descendarm();
    pH = equilibrium();
    rf.transmitData(BONUS, pH);
    rf.transmitData(END_MISSION, NO_DATA);
    z++;
  }
}

//This function returns the distance of the nearest obstacle
//in front of the OSV. Use within a conditional to check for wall
// Uses the suggested PING method for a clean reading
float checkObsDist() {
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(20);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(50);
  digitalWrite(pingPin, LOW);
  pinMode(pingPin, INPUT);
  long duration = pulseIn(pingPin, HIGH);
  return duration / 74 / 2;
}

//will bring the arm from 0 to 90 degrees
void ascendarm() {
  armServo.attach(servoPin);
  armServo.write(105);
```

```
  delay(500);
  armServo.detach();
}
//will bring the arm from 90 to 0 degrees
void descendarm() {
  armServo.attach(servoPin);
  armServo.write(85);
  delay(900);
  armServo.detach();
}

//will set the OSV moving forward and update postition every .01 seconds
void moveForward() {
  starting();
  FL.run(FORWARD);
  FR.run(FORWARD);
  RL.run(FORWARD);
  RR.run(FORWARD);
  delay(10);
  rf.updateLocation();
}

void moveForwardSlow() {
  FL.setSpeed(180);
  FR.setSpeed(180);
  RL.setSpeed(180);
  RR.setSpeed(180);
  FL.run(FORWARD);
  FR.run(FORWARD);
  RL.run(FORWARD);
  RR.run(FORWARD);
  delay(50);
  rf.updateLocation();
}

//will stop the osv from moving. This is the only command
//that will ever stop it. So always use it.
void stopping() {
  FL.run(RELEASE);
  FR.run(RELEASE);
  RL.run(RELEASE);
  RR.run(RELEASE);
  rf.updateLocation();
  delay(20);
}
```

```
//this will set the motor speed to 255.
void starting() {
  FL.setSpeed(255);
  FR.setSpeed(255);
  RL.setSpeed(255);
  RR.setSpeed(255);
}

void moveBackward() {
  starting();
  FL.run(BACKWARD);
  FR.run(BACKWARD);
  RL.run(BACKWARD);
  RR.run(BACKWARD);
  delay(10);
  rf.updateLocation();
}

void moveBackward1() {
  starting();
  FL.run(BACKWARD);
  FR.run(BACKWARD);
  RL.run(BACKWARD);
  RR.run(BACKWARD);
  delay(250);
  stopping();
  rf.updateLocation();
}

void moveForward1() {
  starting();
  FL.run(FORWARD);
  FR.run(FORWARD);
  RL.run(FORWARD);
  RR.run(FORWARD);
  delay(100);
  stopping();
  rf.updateLocation();
}

//returns the absolute value. I didn't want to import cmath.
 float abs(float a) {
   if(a < 0) return -a;
   if(a >= 0) return a;
```

```
 }

 //returns the pH once it has come to an equilibrium
 float equilibrium() {
  int j = 0;
  float ph1, ph2;
  while(j == 0) {
    ph1 = readpHMeter();
    rf.println(ph1);
    delay(2000);
    ph2 = readpHMeter();
    rf.println(ph2);
    delay(200);
    if(abs(ph1 - ph2) < .05) {
      j++;
    }
  }
  return ph1;
}

 // This is code for the pH meter. Do not touch. Ever.
 // It has been adapted from the provided sample code on
 // dfrobot.com for use in our code
 float readpHMeter() {
  static unsigned long samplingTime = millis();
    static unsigned long printTime = millis();
    static float pHValue,voltage;
    if(millis()-samplingTime > samplingInterval)
    {
      pHArray[pHArrayIndex++]=analogRead(SensorPin);
      if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
      voltage = avergearray(pHArray, ArrayLenth)*5.0/1024;
      pHValue = 3.5*voltage+Offset;
      samplingTime=millis();
    }
     //Every 800 milliseconds, print a numerical, convert the state of the LED indicator
    if(millis() - printTime > printInterval)
    {
      return pHValue;
      printTime=millis();
    }
}
//don't touch this either.
// This was also provided with the above function by
// dfrobot.com
```

```
double avergearray(int* arr, int number){
  int i;
  int max,min;
  double avg;
  long amount=0;
  if(number<=0){
    return 0;
  }
  if(number<5){   //less than 5, calculated directly statistics
    for(i=0;i<number;i++){
      amount+=arr[i];
    }
    avg = amount/number;
    return avg;
  }else{
    if(arr[0]<arr[1]){
      min = arr[0];max=arr[1];
    }
    else{
      min=arr[1];max=arr[0];
    }
    for(i=2;i<number;i++){
      if(arr[i]<min){
        amount+=min;        //arr<min
        min=arr[i];
      }else {
        if(arr[i]>max){
          amount+=max;    //arr>max
          max=arr[i];
        }else{
          amount+=arr[i]; //min<=arr<=max
        }
      }//if
    }//for
    avg = (double)amount/(number-2);
  }//if
  return avg;
}

//This is navigation code
void nav1() {
  turn2(pi/2, .05, 1);
  while(marker.y < 1.55) {
    moveForward();
  }
```

```
  turn2(0, .05, 1);
  if(checkObsDist() < 25) {
    turn2(pi/2, .05, 1);
    while(marker.y > .4) {
      moveBackward();
    }
   turn2(0, .05, 1);
    while(marker.x < 1.2) {
      moveForward();
    }
    stopping();
    turn2(atan((1.5-marker.y)/(2.0-marker.x)), .02 ,1.0);
    while(marker.y < 1.35) {
      moveForward1();
    }
    stopping();
    rf.updateLocation();
    turn2(atan((1.5-marker.y)/(2.0-marker.x)), .02 ,1.0);
    stopping();
  }
  else {
    turn2(.05, .05, 1);
    while(marker.x < 1.2) {
    moveForward();
    }
    stopping();
    rf.updateLocation();
    delay(2000);
    turn2(-1.0*atan((marker.y-1.5)/(2-marker.x)), .02, 1.0);
    while(marker.x < 1.55) {
      moveForward();
    }
    stopping();
    rf.updateLocation();
    turn2(-1.0*atan((marker.y-1.5)/(2-marker.x)), .02, 1.0);
    stopping();
  }
}
//this tells the servo to dump the baking soda
void dumpSolution() {
  pinMode(16, OUTPUT);
for(int i =0;i<35;i++)
{
digitalWrite(16, HIGH);
  delay(10);
```

```
  digitalWrite(16, LOW);
  delay(10);
}
  digitalWrite(16, HIGH);
  delay(4000);
  digitalWrite(16, LOW);
  delay(4000);
}
// this is the final iteration of our modified
// differential steering function. It's a mix of
// open loop and closed loop control as it
// is not one continuous movement
void turn2(float desiredAngle, float x, float c) {
  float error = marker.theta - desiredAngle;
  while(abs(error) > x ) {
      if(error < 0) {
        starting();
        FL.run(BACKWARD);
        FR.run(FORWARD);
        RL.run(BACKWARD);
        RR.run(FORWARD);
        delay(150);    //****
        stopping();
        rf.updateLocation();
        delay(20);
        error = marker.theta - desiredAngle;
        rf.println(error);
        if(error < x && error > -1.0*x) {
          break;
        }
        starting();
        FL.run(BACKWARD);
        FR.run(BACKWARD);
        RL.run(BACKWARD);
        RR.run(BACKWARD);
        delay(300); //******
        stopping();
        rf.updateLocation();
        delay(20);
        error = marker.theta - desiredAngle;
        rf.println(error);
        if(error < x && error > -1.0*x) {
          break;
        }
        starting();
```

```
  FL.run(BACKWARD);
  FR.run(FORWARD);
  RL.run(BACKWARD);
  RR.run(FORWARD);
  delay(150);  //******
  stopping();
  rf.updateLocation();
  delay(20);
  error = marker.theta - desiredAngle;
  rf.println(error);
  if(error < x && error > -1.0*x) {
     break;
  }
  starting();
  FL.run(FORWARD);
  FR.run(FORWARD);
  RL.run(FORWARD);
  RR.run(FORWARD);
  delay(300); //****
  stopping();
  rf.updateLocation();
  delay(20);
  error = marker.theta - desiredAngle;
  rf.println(error);
}
else {
  starting();
  FL.run(FORWARD);
  FR.run(BACKWARD);
  RL.run(FORWARD);
  RR.run(BACKWARD);
  delay(150);    //***
  stopping();
  rf.updateLocation();
  delay(20);
  error = marker.theta - desiredAngle;
  rf.println(error);
  if(error < x && error > -1.0*x) {
    break;
  }
  starting();
  FL.run(BACKWARD);
  FR.run(BACKWARD);
  RL.run(BACKWARD);
  RR.run(BACKWARD);
```

```
        delay(300); //***
        stopping();
        rf.updateLocation();
        delay(20);
        error = marker.theta - desiredAngle;
        rf.println(error);
        if(error < x && error > -1.0*x) {
          break;
        }
        starting();
        FL.run(FORWARD);
        FR.run(BACKWARD);
        RL.run(FORWARD);
        RR.run(BACKWARD);
        delay(150);  //***
        stopping();
        rf.updateLocation();
        delay(20);
        error = marker.theta - desiredAngle;
        rf.println(error);
        if(error < x && error > -1.0*x) {
           break;
        }
        starting();
        FL.run(FORWARD);
        FR.run(FORWARD);
        RL.run(FORWARD);
        RR.run(FORWARD);
        delay(300); //*****
        stopping();
        rf.updateLocation();
        delay(20);
        error = marker.theta - desiredAngle;
        rf.println(error);
      }
  }
  stopping();
}
void runSamplePump() {
  pinMode(15, OUTPUT);
  digitalWrite(15, HIGH);
  delay(6500);
  digitalWrite(15,LOW);
}
```