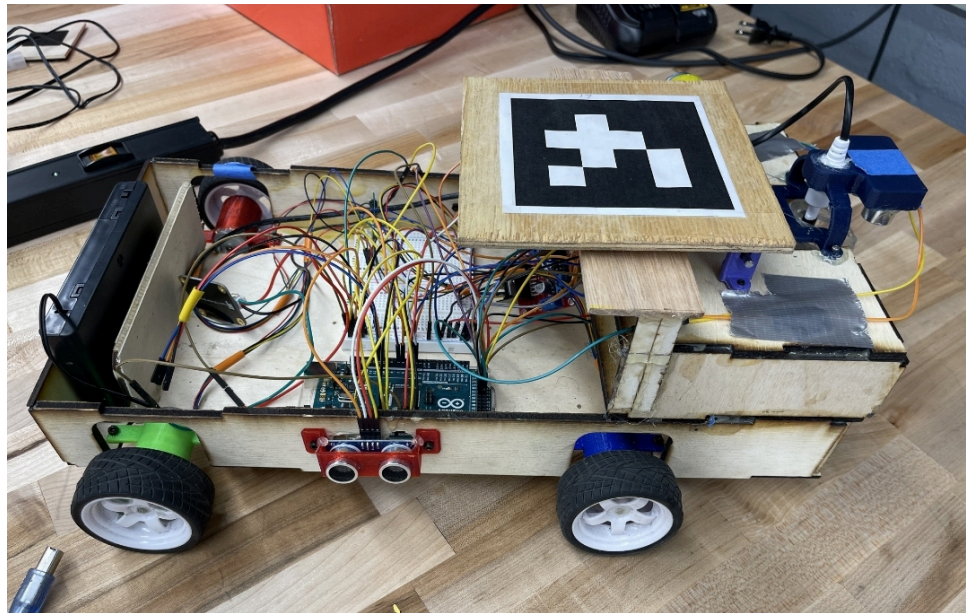
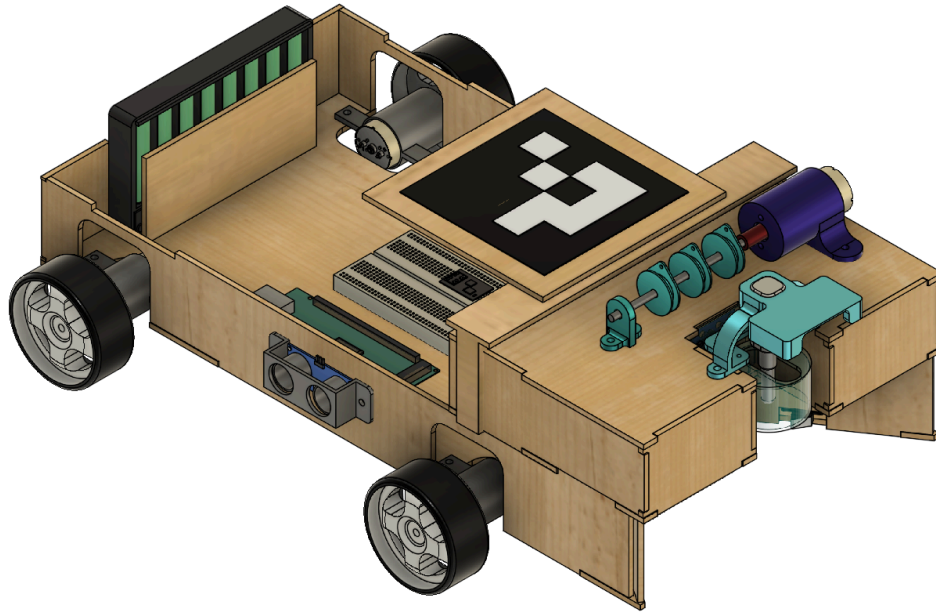


MS9: Water Sampling

The Water Tribe

Avatars: Ari Burman, Taraneh Moline, Aldrin Martin, Aditya Tejawani, Jack Noyes, Mahi Sanghavi, Kavyesh Pasham



Meet the Painted Lady!

Constructing a Prototype

The first step in creating *The Painted Lady* was laser cutting out her body. This included the chassis, side walls, back wall, two cut walls that helped her reach the water pool in level 1 (L1), the wall that held the battery pack in place, the floor/roof of level 2 (L2), L2 side walls, and two types of cut walls to account for the pulley system in L2.

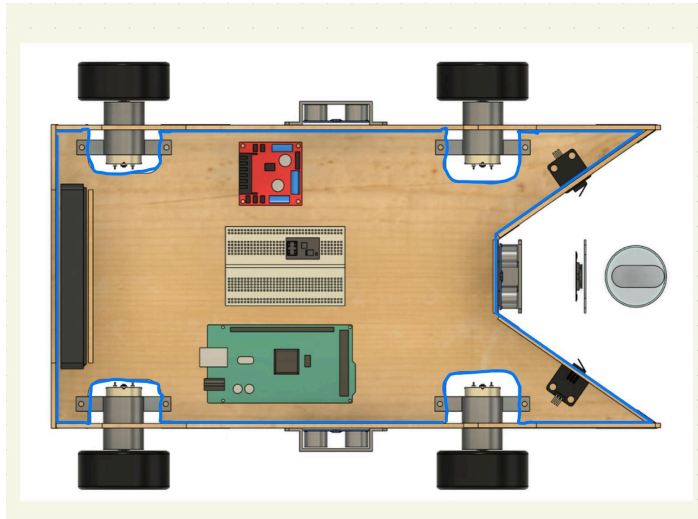


Figure 1. Bird's eye view of the OTV. The laser cut outline of the chassis is shown.

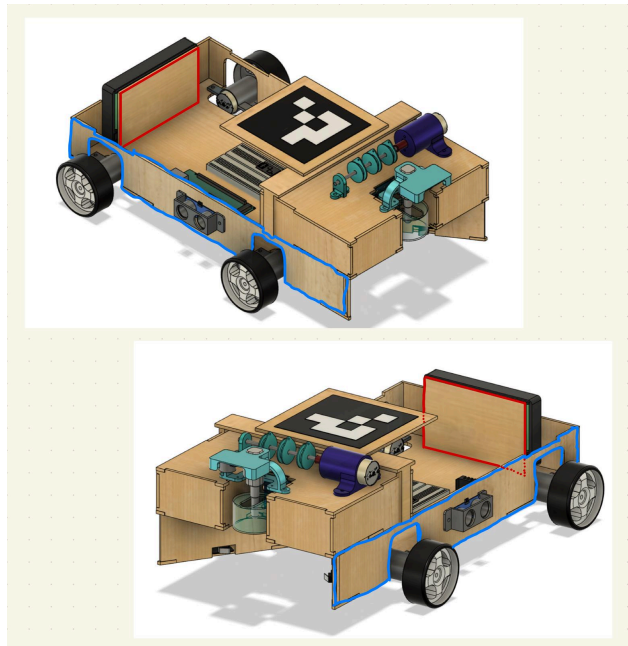


Figure 2. Outline of the laser cut of the L1 side walls (in blue) and wall to hold battery pack in place (in red).

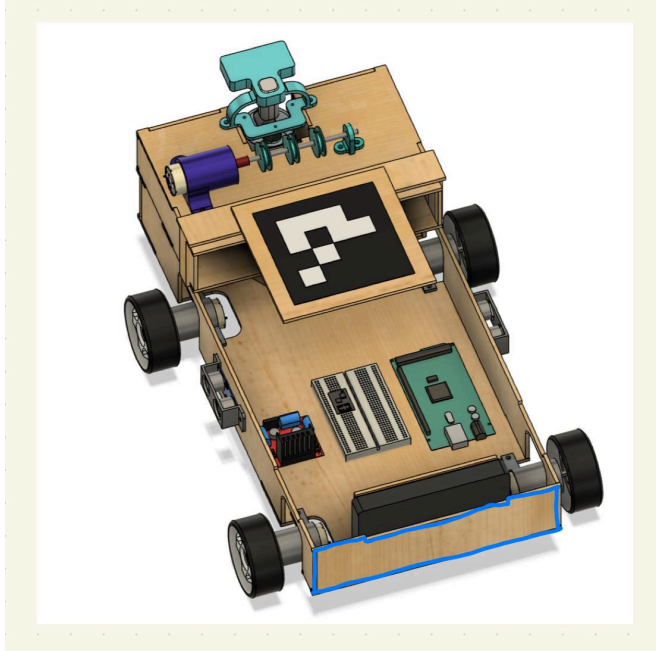


Figure 3. Outline of the laser cut of the OTV's back wall.

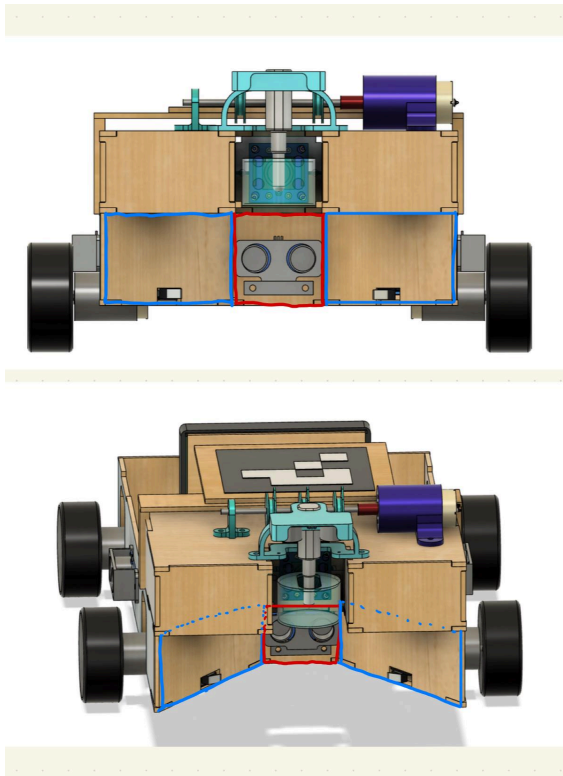


Figure 4. Outline of the laser cut of L1 walls 1 (in blue) and 2 (in red) to support the system that reaches the water pool.

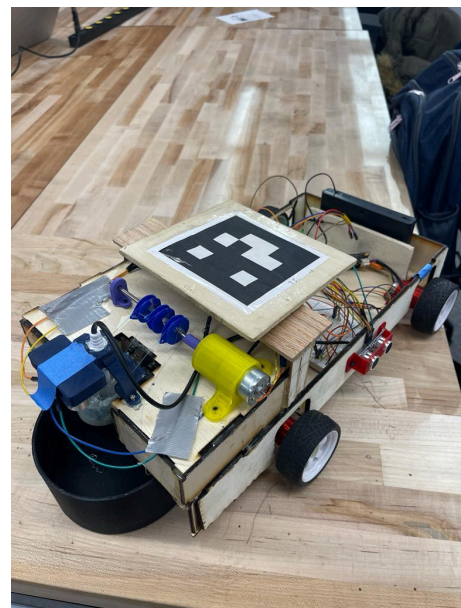


Figure 14: Corner Top View of Real OTV

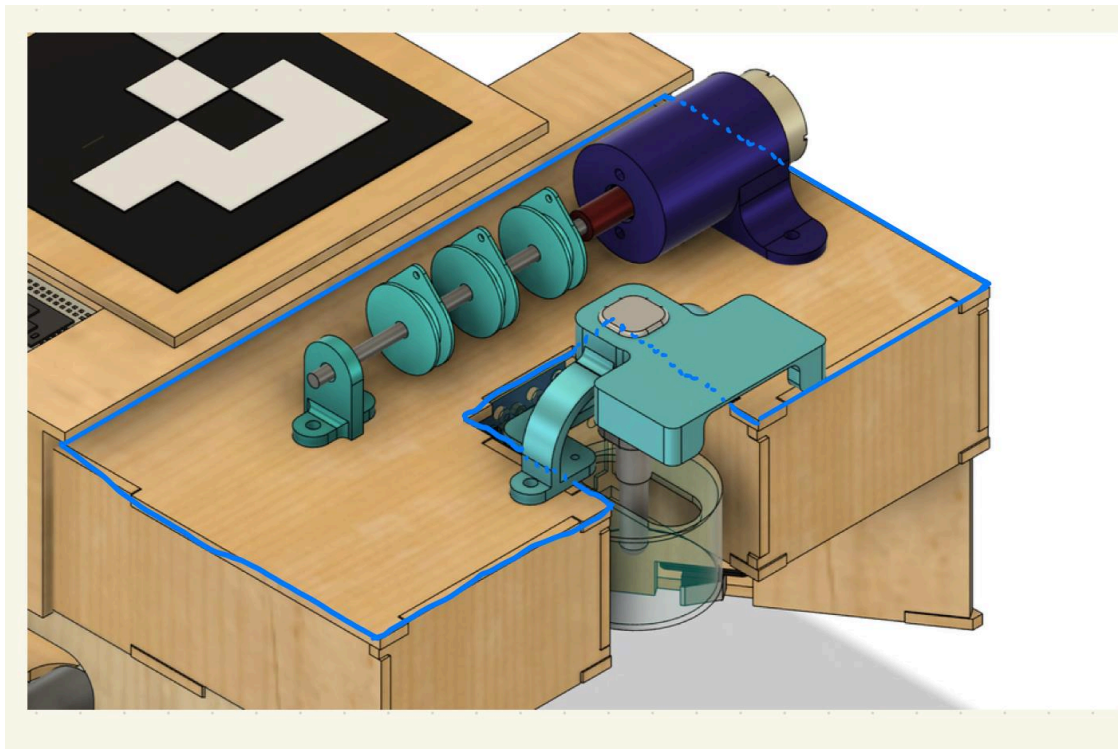


Figure 5. Outline of the laser cut of L2's roof.

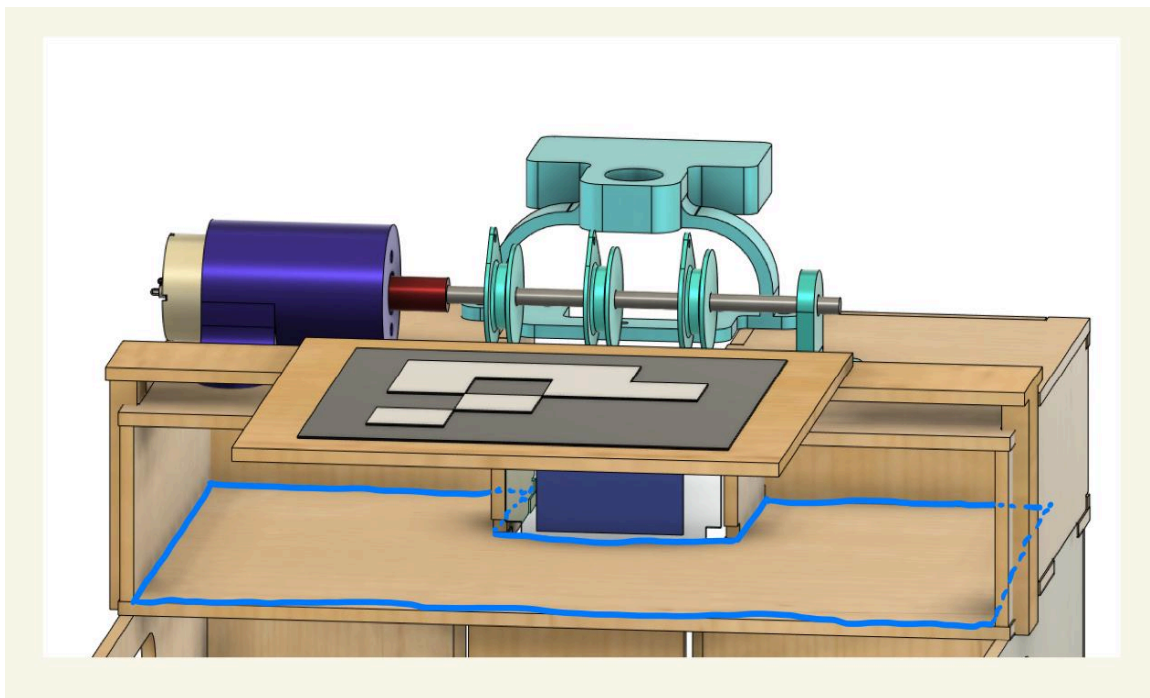


Figure 6. Outline of the laser cut of L2's floor.

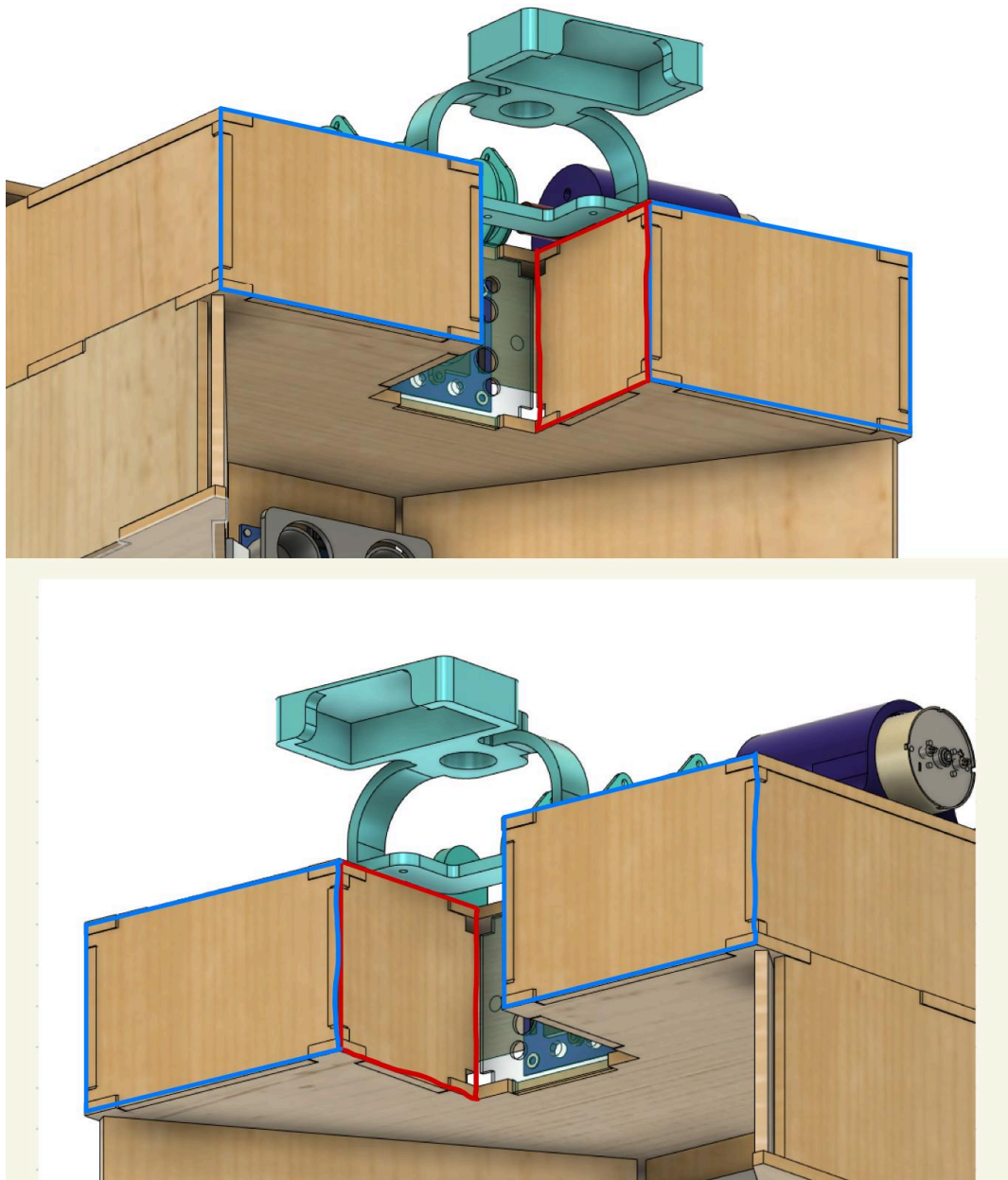


Figure 7. Outline of the laser cuts of L2 walls 1 (in blue) and 2 (in red) to support the pulley system.

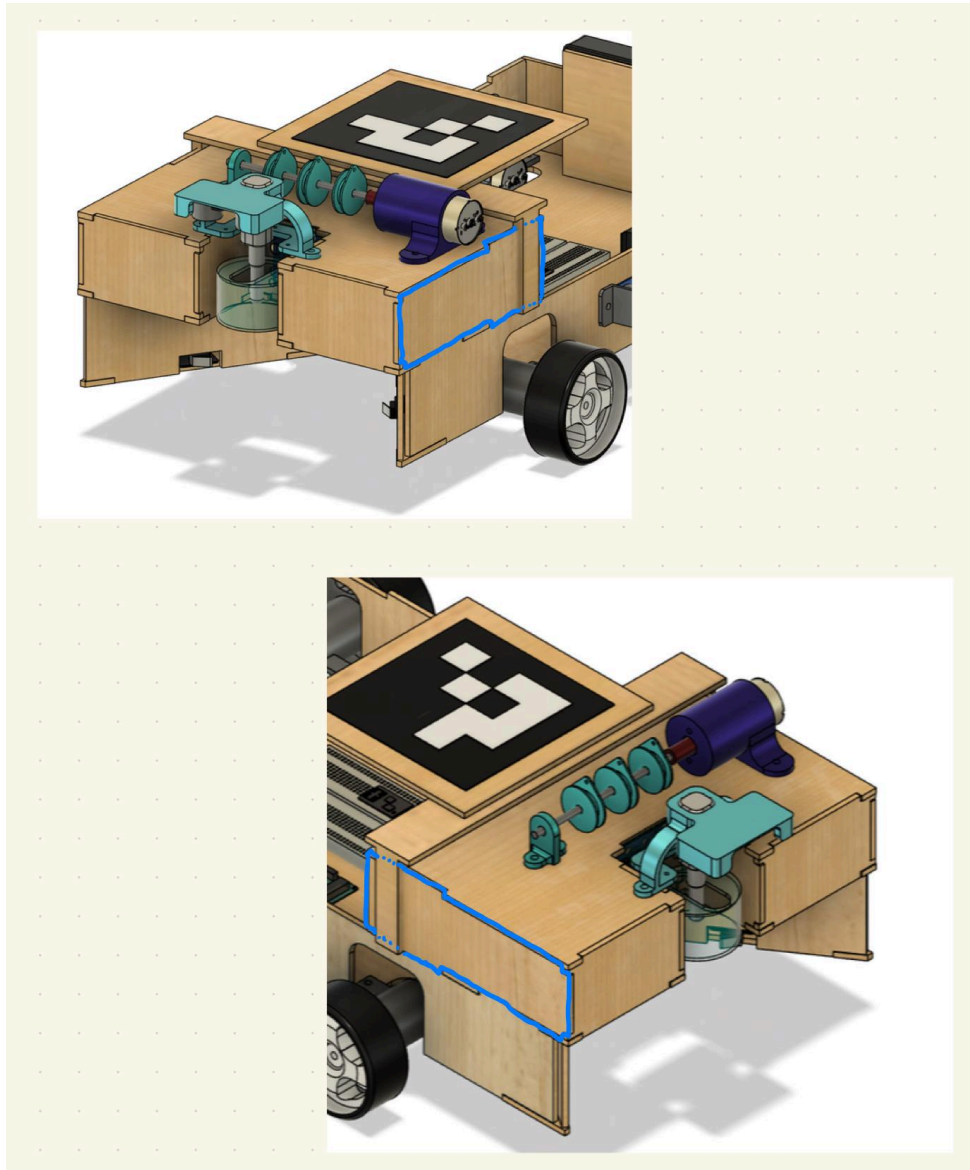


Figure 8. Outline of the laser cuts of L2 sidewalls.

The chassis, side walls, back wall, battery pack holder, and walls to support reaching the water pool together made up the first level, the base of *The Painted Lady*. These pieces were assembled by means of wood glue, and extra care was taken to ensure that these pieces were snug and that the grooves fit in with each other.

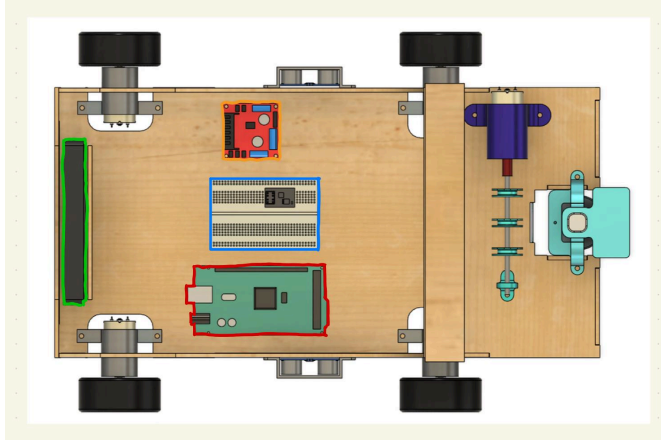


Figure 9. Outline of the major, non-3D-printed components for the chassis. They include the breadboard (in blue), Arduino (in red), battery pack (in green), and the H-bridge (in orange).

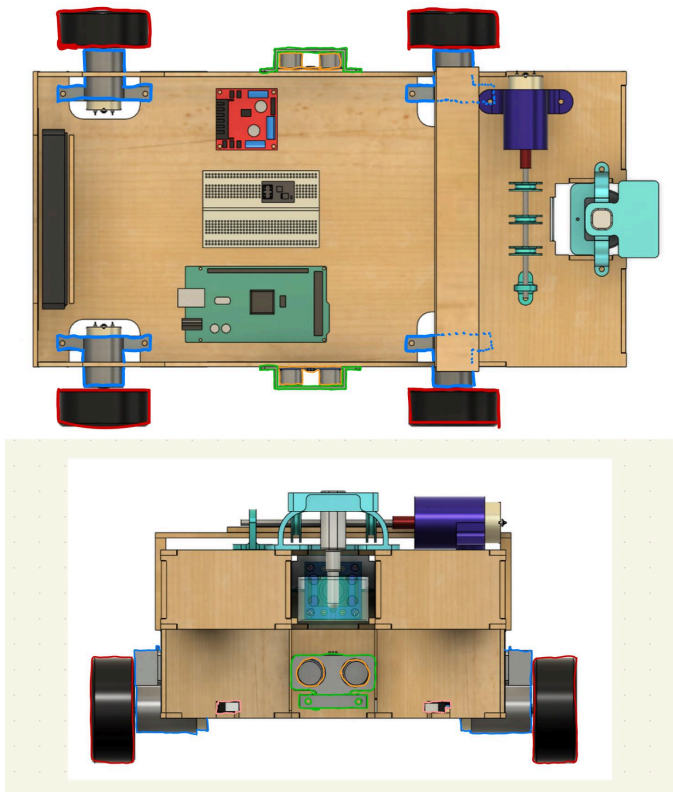


Figure 10. Outline of the major components with their respective 3D-printed mounts. They include the wheels (in red), wheel motor and mounts (in blue), ultrasonic sensor mounts (in green), bumper switches (in pink), and ultrasonic sensors themselves (in orange).

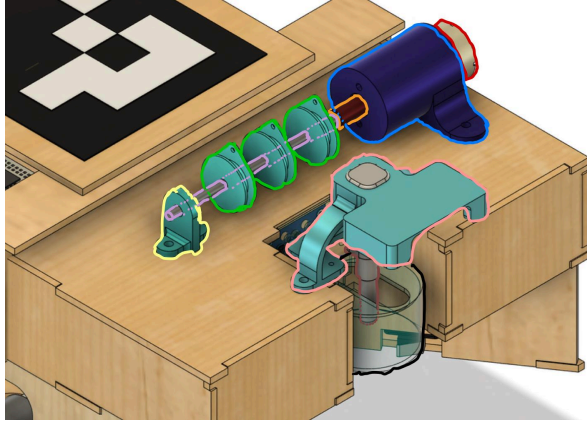


Figure 11. Outline of the motor (in red), motor mount (in blue), pulley wheels (in green), axle (in purple), axle adapter (in orange), pulley mount (in yellow), TDS/Ultrasonic sensor mount (in pink), TDS sensor (in brown), and water cup (in black).

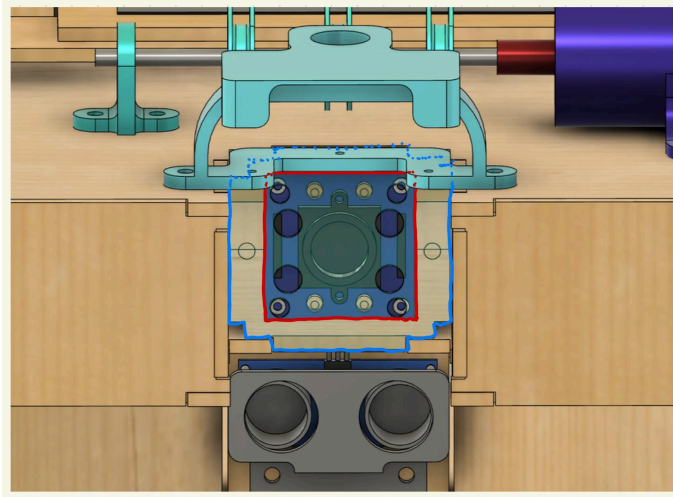


Figure 12. An outline of the acrylic mount for the color sensor (in blue), as well as the color sensor itself (in red).

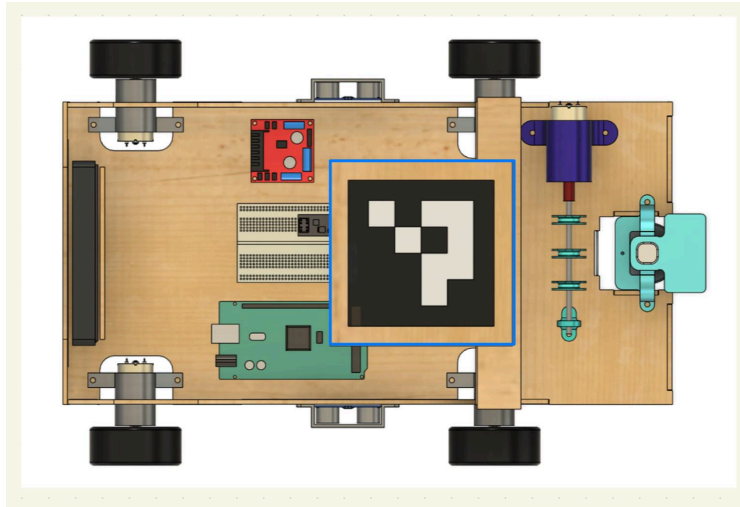


Figure 13. The complete OTV with the addition of the Aruco marker, outlined, on the top.

Next, the major components of L1 that did not need to be 3D-printed were affixed to the chassis. These components consisted of the breadboard, Arduino, and battery pack (Figure 9). Next, the 3D-printed components and their respective electronic components, such as the wheel motor mounts and ultrasonic sensor mounts, wheels, axles, bumper switches, and ultrasonic sensors (Figure 10), were affixed to the chassis and L1 side walls. This was done by drilling holes into L1 side walls and chassis, and then using screws to secure the mount and the electronic component together in place. After the components were secured, they were connected to the arduino and breadboard with wires. Hot glue and duct tape was used around wood joints and screws in order to provide extra support and stability wherever needed.

After the first level was completed, the second level of the OTV was constructed. This was made by joining together the L2 floor, roof, side walls, and L2 walls to support the pulley system (Figures 5-8). Following that, the necessary 3D-printed parts for L2 (Pulley wheels, pulley motor mount, pulley axle adapter, pulley axle mount, and Total Dissolved Solids (TDS) / ultrasonic sensor mount) were affixed to the second level of the OTV, and their respective electronic components were connected to the OTV with the necessary wiring for them to be online (Figures 11). Next, the water retention cup and mount for the color sensor were laser cut from an acrylic board, and the color sensor mount was bound in between the L2 walls with the color sensor behind it, with holes in the front of the sensor to allow for the LEDs on the color sensor to go through (Figure 12). The water cup was attached to the pulley system by means of

fishing line, with one strand of the line being tied to one pulley wheel, for a total of three strands being used to secure the water cup (Figure 11).

Finally, in order for the OTV to work with the wifi navigation module, a print of the Aruco marker to a wooden board on the top of the OTV was attached (Figure 13). Support beams for the Aruco marker were cut from a wood board, and the two pieces were connected with Velcro.

Mission

For our mission, we had multiple requirements and challenges. The requirements were to navigate to within 150 mm of the edge of the water pool, measure and correctly transmit whether the water in the pool is fresh unpolluted, fresh polluted, salt unpolluted, or salt polluted, measure and transmit the depth of the water in the pool to within 4 mm. We also needed to collect a sample of at least 20 ml of water from the pool and maintain the water sample on the OTV until the end of the run, navigate completely past the three obstacles, and navigate completely into the destination zone.

To complete these requirements, there were quite a few challenges. Some of these included properly detecting the mission site, ensuring the OTV was able to collect the right amount of water, and ensuring proper lighting conditions for the color sensor to detect dye that signifies pollution of water. Also we needed to ensure that no water leaked out of the cup during navigation, and that the OTV had a fully autonomous navigation system that could detect different types of obstacles, and move around them.

The mission had numerous parts each with its own specific components and process. To start off, the OTV needed to reach the mission site from its start position. This was done through utilizing the Aruco marker (figure 1 and figure 13) to turn and make sure that the OTV was facing the direction of the mission site. The OTV then proceeded forward, being guided by coordinates from the vision system.

Once the OTV got closer to the mission site, it slowed down. At that point, as the OTV approached the water basin, it used the Ultrasonic sensor located on the top of the OTV, which was facing down, to measure the distance to the ground. Then, once the OTV reached the basin, as confirmed by the bumper switches, it measured the distance to the water surface. Using

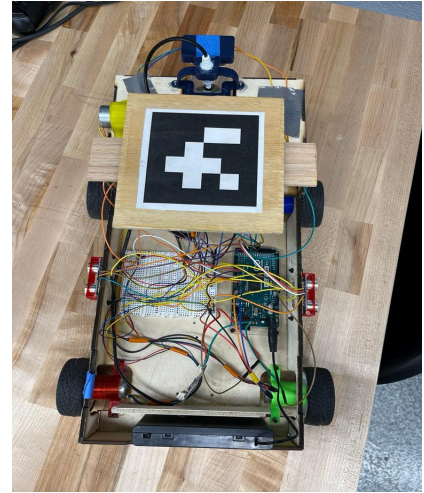


Figure 13: Real OTV - Top View with Aruco Marker

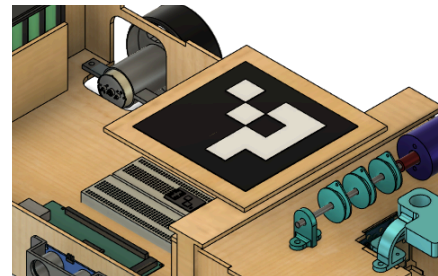


Figure 1: Aruco Marker on OTV

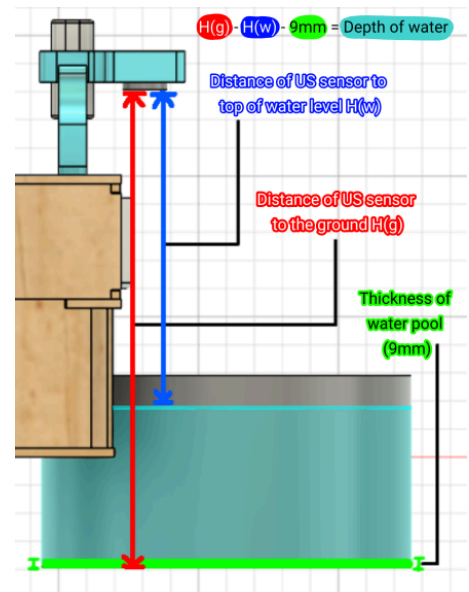


Figure 2: Ultrasonic Sensor Measuring Distances to the ground and then to the water surface

these values as well as the given width of the basin base, we could calculate the depth of the water and transmit it back using the ESP8266 module. This process is displayed in figure 2.

To confirm we were at the mission site, we used bumper switches so that once they both confirmed contact, we were certain that the OTV was in the optimal position to begin water extraction and testing. These switches are seen in figure 3.

First, the pulleys lowered the cup down into the water basin. The cup (figure 12) remained submerged for a predetermined amount of time to ensure the cup was full (figure 4). Then, the pulley system reeled the cup back up into its rest position. (The calculations to justify this are at the end of the section).

In this position, the cup was perfectly positioned under the TDS, so when the cup was reeled up, the TDS sensor entered the hole in the top of the cup and was already in the water, immediately able to test for salt content (figure 5). The result of whether the water had salt content was then transmitted back through the ESP8266 module.

At this point, the cup was perfectly aligned with the color sensor (figure 5 & 6), which detected the presence of pollution in the form of green dye. The cup was entirely made out of transparent material to allow for the color sensor to perform its test while the water remained in the cup. Also, the color sensor itself had an acrylic faceplate to ensure that it didn't get damaged in the event water leaked out of the cup (figure 6). The result of the pollution test was also transmitted back using the ESP8266 module.

With the mission-specific tasks completed, the OTV proceeded towards the next part of the course: the obstacles. To navigate around these, the OTV used the Ultrasonic sensors. The OTV traveled according to an algorithm (which will be

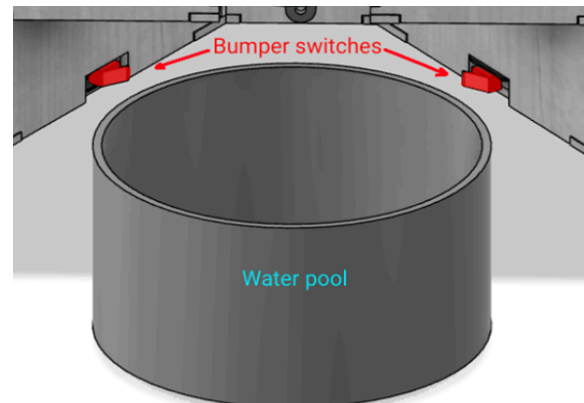


Figure 3: Bumper Switches interacting with Water Basin



Figure 12: Real OTV - Cup/Pulley System

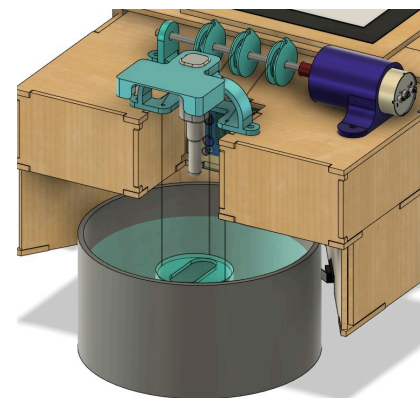


Figure 4: Cup submerged in Water Basin

explained later) in which the OTV stopped at designated points and assessed whether there was an obstacle in that position. This detection was done using the three Ultrasonic Sensors (right, left, and front). There could have been a potential difficulty in detecting the rumbles, given that they were very low to the ground. However, given the sensors' effective angle of 15 degrees, this ensured that they could detect even obstacles that were only 1.5 cm off the ground.

Finally, the OTV navigated towards the limbo bar and proceeded under it, given that the OTV had a height of 134.4mm while the bar had a height of 184 mm.

Regarding the technical specifications of the sensors, the Ultrasonic sensors had an accuracy of $\pm 0.3\text{cm}$ with a range of 2 cm - 450 cm, drawing a combined total of 60mA between the 4 of them. And, the TDS sensor had $\pm 10\%$ full-scale accuracy with a range of 0 to 1000 ppm. The TDS sensor drew 6mA. Finally, the color sensor had a nonlinearity error of $- 0.2\%$ at 50kHz with a color range of 300 nm to 1100 nm wavelength, drawing 10mA of power.

The parts for the mission were attached to the chassis largely by 3D printing a mount for them and then attaching the part to its mount. The general structure of the chassis was made from laser-cut plywood that was fitted together and then glued with wood glue. The Ultrasonic sensors on the sides and front had 3D printed mounts that had 3.6mm diameter holes for drilling into plywood. The mount for the TDS sensor also included a mount for the Ultrasonic Sensor that faced down for measuring water depth. This mount was 3D printed and had 3.6mm diameter holes for drilling into the top front of the OTV. The bumper switches had holes and could be drilled directly into plywood on the bottom front of the chassis. The color sensor was in the very front of the OTV, attached to a 3D printed mount that had 3.6mm diameter holes for drilling into the plywood. The pulley system was attached to the top front of the OTV and included a 3D printed mount for the motor and the axis. The mount had 3.6mm holes for drilling into the plywood. The details and more precise dimensions of these mounts can be found in the engineering drawing section.

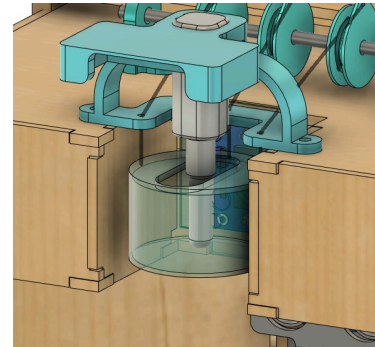


Figure 5: TDS sensor in cup

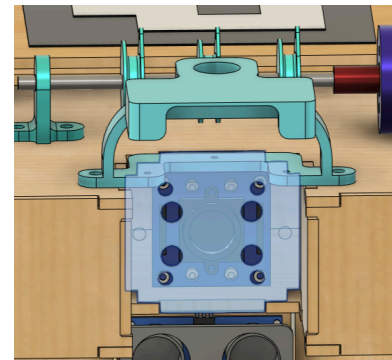


Figure 6: Color Sensor with acrylic protective cover

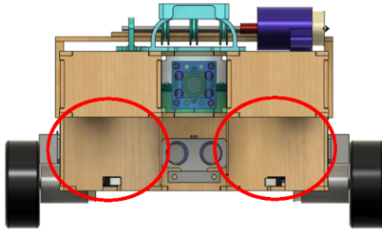


Figure 7: Bumper Switch Location

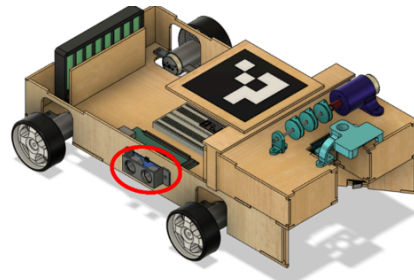


Figure 8: Side Ultrasonic Sensor

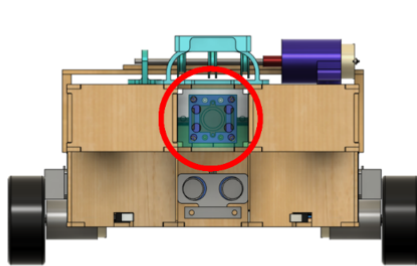


Figure 9: Color Sensor

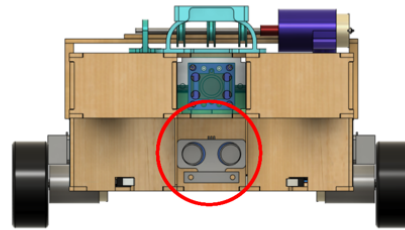


Figure 10: Front Ultrasonic Sensor

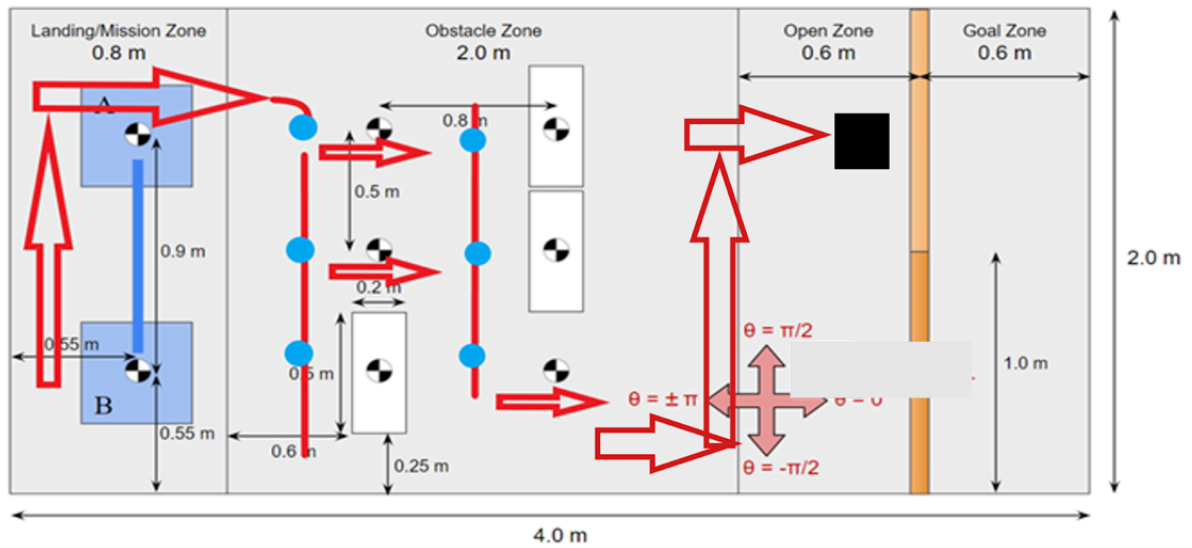


Figure 11: OTV Navigation Plan

This (figure 11) outlines the complete process that the OTV went through to complete the missions and reach the mission site.

First, the OTV used the Aruco Marker to orient itself using the vision system to face the mission site. The vision system detected the angle the OTV was originally positioned at and

reported that value. This value was then used in the propulsion algorithm, which instructed the wheel to move in opposite directions to turn the OTV. The algorithm was calibrated to know how long to turn the wheels for any given amount of turning necessary. Once the vision system used the Aruco Marker to confirm that the OTV was facing the direction of the mission site, the Arduino then sent power to the motors. The OTV moved in a straight line until the bumper switches were pressed (since the OTV was up against the water bowl for the mission) which served as a trigger for the algorithm to stop the propulsion system.

At this point, the OTV was in the correct position to begin the mission. First, the OTV used the downward-facing Ultrasonic sensor to obtain the height from the sensor to the face of the water. Then, using this value as well as the known height of the sensor above the ground and the thickness of the bowl, the algorithm calculated the depth of the water and transmitted this data back through the WIFI module.

Having finished those calculations, the Arduino sent power to the pulley motor to lower the collection cup by the correct length that had been pre-obtained to ensure the cup was fully submerged. The cup remained underwater for a predetermined amount of time until it was completely filled with water. The motor then began rotating in the opposite direction to lift the cup—this amount of time it took to lift the filled cup had already been calculated beforehand to ensure it returned exactly back to its original position. As the cup was fully reeled up, the dissolved solids sensor aligned with the hole in the cup. The water was then measured by the dissolved solids sensor as well as the color sensor, which was positioned in a fixed position aligned with the water cup when it was fully lifted. The values from the dissolved solids sensor and the color sensor were sent to the Arduino, which were used by the algorithm to determine whether the water was toxic and whether it was polluted. The results were displayed on the computer. This processing step took about 4 seconds to complete.

The OTV then continued to the obstacles section. First, the motors for the wheels turned in reverse so the OTV could back away from the mission site. When the OTV was about 15 cm from the mission site (as confirmed by the vision system), the OTV turned 90 degrees (either to the right or the left, depending on the initial start position). From this point forward, for the sake of simplicity, we are assuming that the OTV started in the bottom left corner as seen in figure 12; however, it could have just as easily started in the top left corner, and the whole process would have been reversed. It then proceeded about 55 cm towards the first column of obstacles.

Given that this position was roughly in the middle of the column, the OTV was instructed to turn left. Then, the OTV moved straight for roughly 25 cm until it lined up with the top blue dot seen in figure 12. This was the center of where a potential obstacle could have been in that position. The Ultrasonic sensor on the right side was used to detect the distance in front of it. That value was sent to the Arduino and compared to predetermined values. If that value represented a value consistent with an object being in that position, the OTV moved backward about 50 cm so it was now in front of the 2nd position in the column (the middle blue dot in the 1st column of figure 12). This position was confirmed by the vision system.

The OTV repeated the process of using the Ultrasonic sensor to assess whether there was an obstacle there. If the Arduino decided there was, then the 3rd position must be clear given that there could only be a maximum of 2 objects per column. However, if during its assessment of the 1st column, the Arduino detected that there was no object in that position, the OTV went through that opening onto the 2nd column. Either way, once the OTV reached the 2nd column, it again tried to navigate to either the top or bottom so that it could then move in a straight line and use the side sensors to check (at each blue dot in figure 12) whether there was, in fact, an obstacle adjacent to it, and proceeded accordingly.

After clearing the 2nd column of obstacles, the vision system was used to navigate the OTV to be directly in front of the limbo bar (as denoted by the black box on figure 12). Then, the OTV proceeded forward into the goal zone as it went underneath the bar.

Calculations to justify that pulley system can lift the collection cup:

Total cup weight = .0695kg

Safety factor = 1.15

Distance cup is pulled = 0.16m

$$0.0695kg * 9.8m/s = 0.6811N$$

$$(0.6811N * 0.16m) * 1.15 = .108976 Nm$$

Motor used to lift cup has roughly 0.245Nm stall torque, and thus can lift cup

Specifics about fishing line and its capability regarding lifting the cup:

0.45mm diameter fishing line capable of holding up to 43 lbs of weight per line

Propulsion Modeling

The sole requirement for our OTV is to overcome the force of friction, as our OTV has been estimated to be able to go under the hump obstacle at the end of the course. We decided upon using four wheel drive in order to facilitate easier, tighter turns and to increase the speed. It also ensures less risk of failure during a run, as if one motor malfunctions during operation we still have 3 remaining motors that can drive the OTV, as opposed to if we were using 2 wheel drive we would only have 1 remaining motor and we would likely lose the ability to turn.

Minimum torque our motors need to provide to overcome the force of friction is 0.00547 Nm
Calculations used:

$$\text{Weight} = 3\text{kg}$$

$$\text{Radius} = 0.06477\text{m}$$

$$\text{Safety Factor} = 1.15$$

$$\text{Rolling Resistance Coefficient (Crr)} = 0.01 \text{ (Same friction coefficient as car tires on asphalt)}$$

$$3\text{kg} / 4 = .75\text{kg} \quad .75\text{kg} * 9.8\text{m/s} = 7.35\text{N} \text{ (Force applied from gravity on each wheel)}$$

$$7.35\text{N} * 0.01 = 0.0735\text{N} \text{ (This is the force of rolling resistance [Frr])}$$

$$4F_t - 4F_{rr} = 0$$

$$F_t = F_{rr}$$

Force of torque must at least be equal to force of rolling resistance, which is 0.0735N

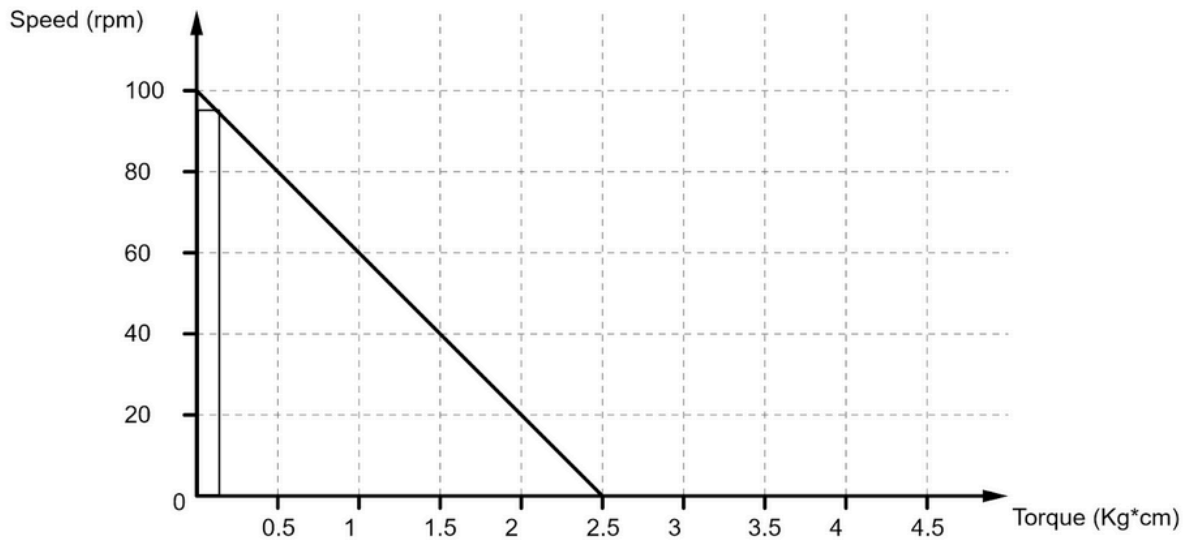
$$F_t * R = T \quad 0.0735\text{N} * 0.06477\text{m} = 0.00476\text{Nm}$$

$$T * SF \quad 0.00476\text{Nm} * 1.15 = 0.00547\text{Nm}$$

Minimum torque our motor requires is shown to be 0.00547 Newton meters.

Our current goal for our OTV is a minimum of 3 centimeters per second, but ideally we'd like it to go as fast as the motors can take it.

Our OTV motor is measured to have a 2.5 kilogram centimeter stall torque, and a no-load speed of 100 RPM. It also has a rated torque & speed of 0.75 kilogram centimeters of torque and 70 RPM. With this information, we are able to create our motor characteristic graph and locate what our operating RPM would be.



Torque graph equation: $RPM = 100 - 40T$

Operating torque = 0.0558kgcm (0.00547Nm)

Operating angular speed = 97.768RPM

Operating linear speed = 0.66m/s

Operating current = 0.057A

We run the motor at max possible speed, and our required torque is 0.00547 Newton meters which is equivalent to 0.0558 kilogram centimeters, which on this graph is equivalent to 97.768 RPM, which will be our operating angular speed. When we convert that angular speed into linear speed, we get roughly 0.66 meters per second. To acquire our operating current, we used the measured amperage of when the motor is operating with no load and when the motor is stalling, which is 0.05 amps and 0.35 amps respectively, and then we created a linear equation from it ($RPM = -333.333A + 116.667$) and plugged in our expected operating speed. From this we got roughly 0.057 amps, which is likely to be our operating current.

Actual operating speed = 0.21m/s Actual operating RPM = 30.96RPM

Actual operating torque = 1.73kgcm

We took a video of our OTV crossing a certain distance and used it to determine that our OTV actually operates at roughly 0.21 meters per second, which is equivalent to roughly 30.96 RPM

and 1.73kgcm of torque. The operating torque and actual torque are significantly different, with the difference between the two being roughly 1.67kgcm. It is possible the reasons behind these differences could be the fact that the torque calculated is only the bare minimum torque to overcome friction thus the OTV would need more than that to actually get moving, or because the friction between the arena and wheels is different than the friction coefficient used for calculations, or perhaps other reasons.

Center of Gravity Calculation:

Weight (kg)

$(0.25 \text{ kg} + 0.015 \text{ kg} + 0.015 \text{ kg} + 0.038 \text{ kg} + 0.030 \text{ kg} + 0.037 \text{ kg} + 0.023 \text{ kg} + 0.023 \text{ kg} + 0.082 \text{ kg} + 0.0031 \text{ kg} + 0.0031(3) \text{ kg} + 0.0090 \text{ kg} + 0.0091(3) \text{ kg} + 0.050 \text{ kg} + 0.018 \text{ kg} + 0.0079 \text{ kg} + 0.0042(4) \text{ kg} + 0.011 \text{ kg} + 0.0023 \text{ kg} + 0.01 \text{ kg}) = \mathbf{0.59353 \text{ kg}}$

$\bar{X}$ (mm)

25.46838551 mm (Calculated from [Center of Mass Calculations](#))

$\bar{Y}$ (mm)

-3.354665439 mm (Calculated from [Center of Mass Calculations](#))

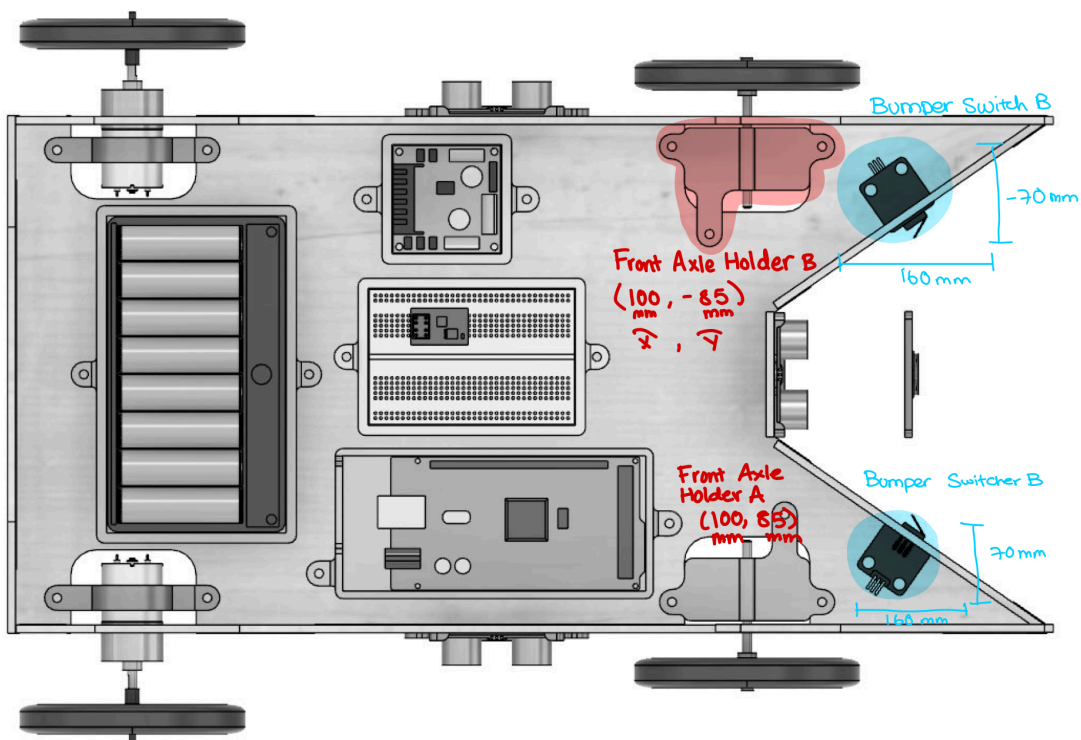
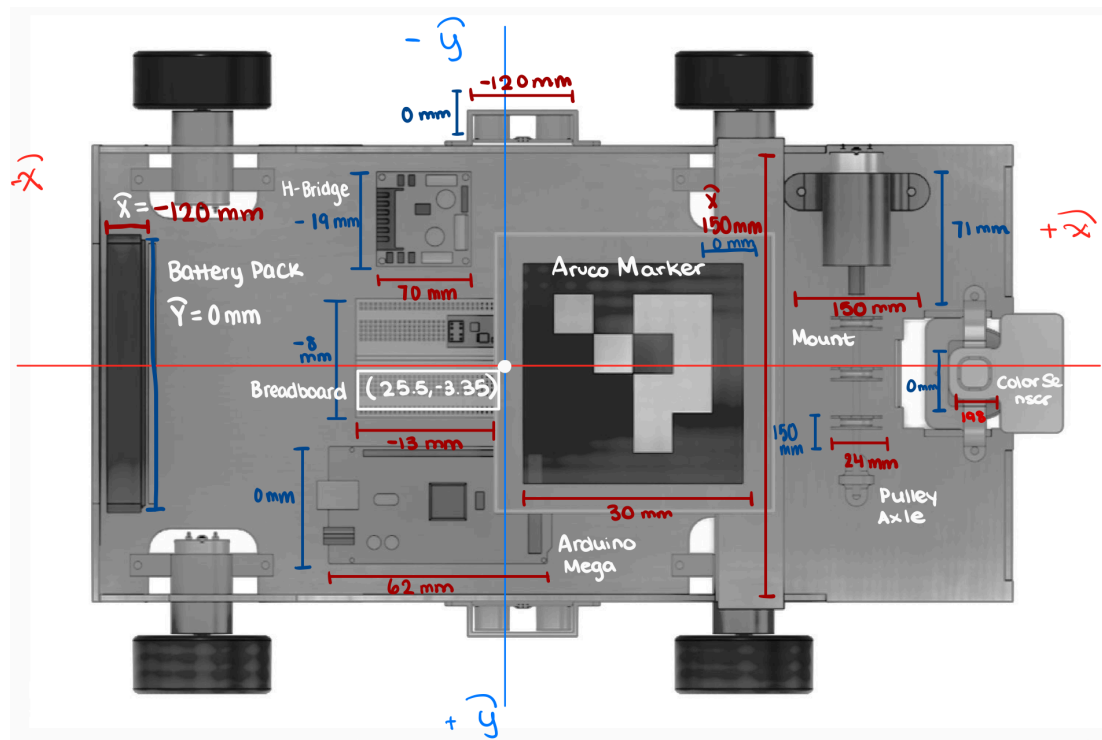
Center of Mass Coordinate Point ($\bar{X}, \bar{Y}$): **(25.46838551 mm, --3.354665439 mm)**

Interpretation of Center of Mass: This coordinate point passes the center of mass test because it is close to the theoretical/ideal center of mass which would be (0,0). Due to the front load and bottom load of our OTV (due to the pulley system collecting the water sample), it is understandable that the calculation yielded an X bar value in the positive direction, as that is where the load of our OTV resides. However, it is important to understand that the coordinate point is still very close to the theoretical center, and thus allows it to pass and is an acceptable center of mass for the OTV overall.

Center of Mass Table (with respect to individual coordinates of the components on the chassis)

A	B	C	D	E	F	G
Item	Weight (kg)	$\bar{X}$ (mm)	$\bar{Y}$ (mm)	Total $\bar{X}$ (mm)	Total $\bar{Y}$ (mm)	Coordinate
Battery pack	0.25	-120	0	-30	0	
Bumper switch A	0.015	160	70	2.4	1.05	
Bumper switch B	0.015	160	-70	2.4	-1.05	
Breadboard	0.038	-13	-8	-0.494	-0.304	
H-bridge	0.03	70	-19	2.1	-0.57	
Arduino mega	0.037	62	0	2.294	0	
2nd Level box wood	0.023	150	0	3.45	0	
Pulley axle	0.082	150	0	12.3	0	
Pulley wheel J	0.0031	150	0	0.465	0	
Pulley wheel K	0.0031	150	-24	0.465	-0.0744	
Pulley wheel L	0.0031	150	24	0.465	0.0744	
Pulley motor	0.009	150	-71	1.35	-0.639	
Ultrasonic sensor A	0.0091	-120	0	-1.092	0	
Ultrasonic sensor B	0.0091	120	0	1.092	0	
Ultrasonic sensor C	0.0091	130	0	1.183	0	
Salinity sensor	0.05	198	0	9.9	0	
Color sensor	0.018	173	0	3.114	0	
Salinity sensor mount	0.0079	198	0	1.5642	0	
US sensor A mount	0.0042	-120	0	-0.504	0	
US sensor B mount	0.0042	120	0	0.504	0	
US sensor C mount	0.0042	-210	0	-0.882	0	
US sensor D mount	0.0042	130	0	0.546	0	
Pulley motor mount	0.011	150	-71	1.65	-0.781	
Pulley axle holder	0.0023	150	47	0.345	0.1081	
Cup	0.01	198	0	1.98	0	
				Coordinate	Coordinate	
	0.6516			25.46838551	-3.354665439	

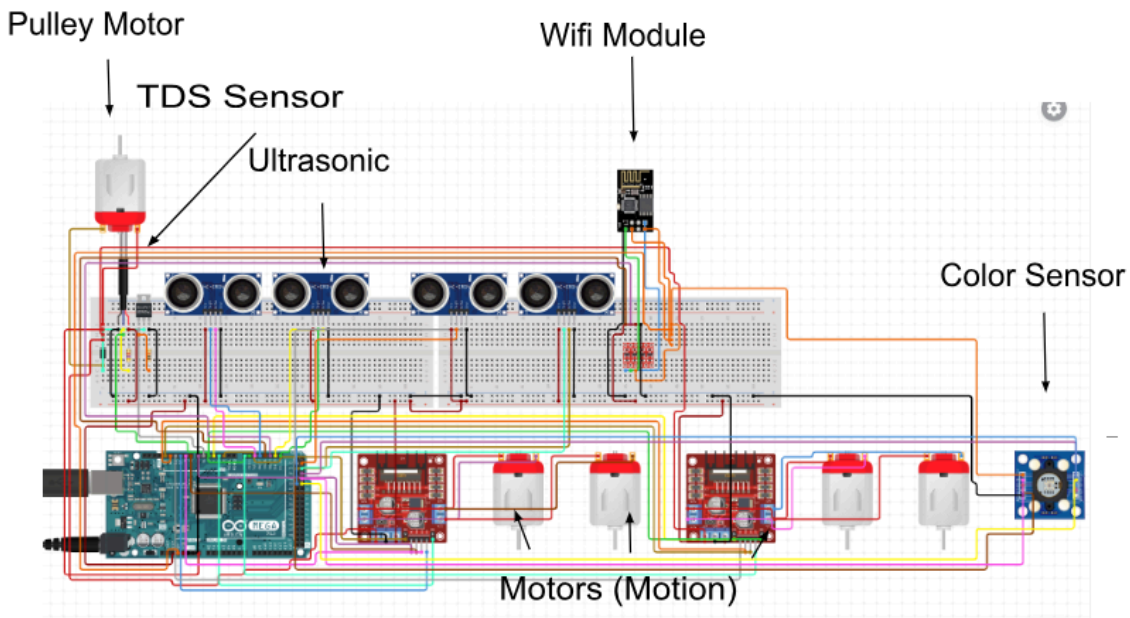
Center of Mass Diagram with Top View Engineering CAD Design



Electronics

The battery consists of 8 1.5V Zinc AA batteries. In total the power output is 1.5V. The max recommended discharge rate for AA batteries is .5 to .75A.

Complete Circuit Diagram



*Note: In this diagram, the movement motors are connected to an hbridge. Due to our motors having hbridge logic built into them, this is the only way to represent the connections. Additionally, the singular motor connected at the top is connected alone to an hbridge. The circuit drawing software did not allow connecting a singular motor to an hbridge. For more specific circuit drawings, look below.

Circuit Diagrams

Ultrasonic



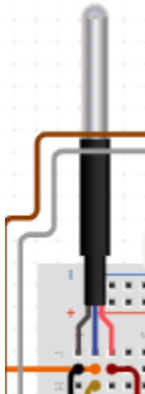
1st pin: Power

2nd pin: Arduino control pin (See Pin Connections Below)

3rd pin: Arduino control pin (See Pin Connections Below)

4th pin: Ground

TDS Sensor

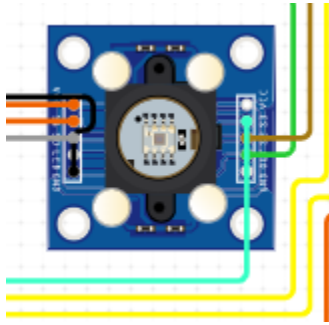


1st pin: Ground

2nd pin: Arduino control pin (See Pin Connections Below)

3rd pin: Power

Color Sensor



S0: Arduino control pin (See Pin Connections Below)

S1: Arduino control pin (See Pin Connections Below)

S2: Arduino control pin (See Pin Connections Below)

S3: Arduino control pin (See Pin Connections Below)

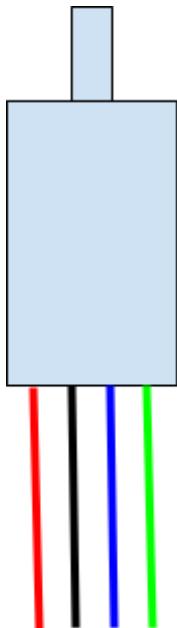
VCC: power

GND: ground

OUT: Arduino control pin (See Pin Connections Below)

Motor Circuit Diagrams

*Note: The Left Motors share same pin connections and the Right Motors share the same pin connections. Also, The motors the OTV uses only utilizes 4 wires to turn and does not require an H-Bridge to be controlled.



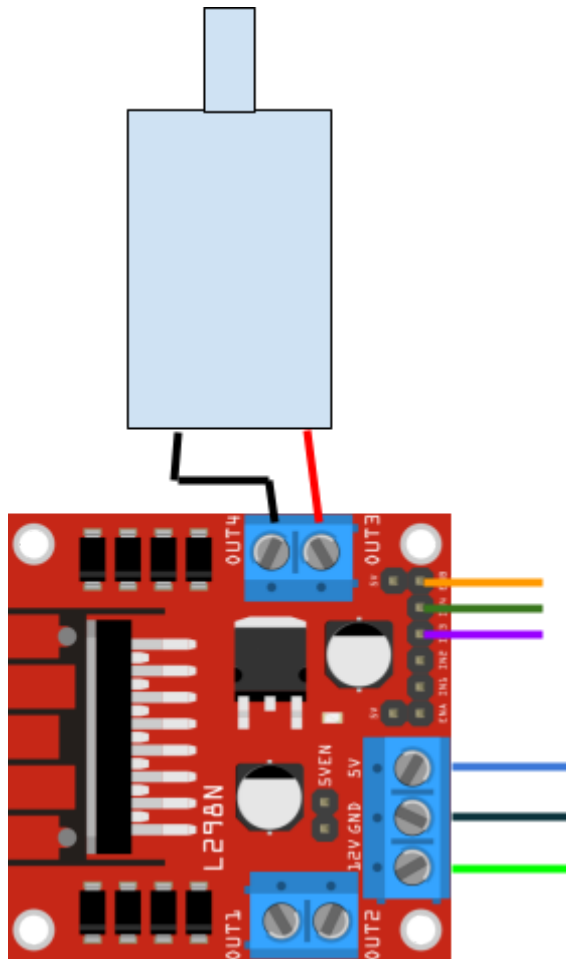
Red: power (12V)

Black: Ground

Blue: Arduino control pin (See Pin Connections Below)

Green: Arduino control pin (See Pin Connections Below)

Pulley System



Pulley Motor

+: H-Bridge Out 4

-: H-Bridge Out 3

H-Bridge

Out4: Pulley Motor +

Out3: Pulley Motor -

12V: Vin

GND: -breadboard

5V: +breadboard

ENB: Arduino control pin (See Pin Connections Below)

IN3: Arduino control pin (See Pin Connections Below)

IN4: Arduino control pin (See Pin Connections Below)

Total Runtime Current Draw

- Ultrasonic sensors: $15\text{mA} * 4 = 60\text{mA}$
- TDS Sensor: 6mA
- Color Sensor: 10mA
- DC Motors: $130\text{mA} * 3 = 390\text{mA}$
- Wifi Module: 120 mA
- Arduino Mega: 100mA

Total: 686 mA

The 12V battery pack supplies current at a rate of 2800 mAh. The battery supplies voltage for approximately 4 hrs.

The sensors like the ultrasonic sensors and the tds sensor will be connected to the 5V power supply on the Arduino Mega board. The motors on the other hand are powered by the 12V power from the Vin port on the Arduino Mega which takes the 12V power from the battery pack.

Arduino Pin Assignment Chart

Left Ultrasonic Sensor

- GND: -breadboard
- +: +breadboard
- Trig: 27
- Echo: 28

Right Ultrasonic Sensor

- GND: -breadboard
- +: +breadboard
- Trig: 29
- Echo: 30

Depth Ultrasonic Sensor

- GND: -breadboard
- +: +breadboard
- Trig: 31
- Echo: 32

Front Ultrasonic Sensor

- GND: -breadboard
- +: +breadboard
- Trig: 33
- Echo: 34

Color Sensor

- S0: 35
- S1: 36
- S2: 37
- S3: 38
- VCC: +breadboard
- GND: -breadboard
- OUT: 39

WiFi Module

- GND: -breadboard
- VCC: +breadboard
- TX: 10
- RX: 11

TDS Sensor

- VCC: +breadboard
- GND: -breadboard
- A: A0(54)

Left Motors (Rear Left Motor and Front Left Motor receive same controls)

- +: Vin
- -: GND
- PWM: A1(55)
- CW/CCW: 3

Right Motors (Rear Right Motor and Front Right Motor receive same controls)

- +: Vin
- -: GND
- PWM: A2(56)
- CW/CCW: 4

Pulley Motor

- +: H-Bridge Out 4
- -: H-Bridge Out 3

H-Bridge

- Out4: Pulley Motor +
- Out3: Pulley Motor -
- 12V: Vin
- GND: -breadboard
- 5V: +breadboard
- ENB: A3(57)
- IN3: 6
- IN4: 7

Full Code

```
#include "Enes100.h"
```

```
#define leftPWM 55
```

```
#define rightPWM 56
```

```
#define leftDirection 3
```

```
#define rightDirection 4
```

```
const int ENB = 57; // PWM speed control for motor
```

```
const int IN3 = 6; // Input 3
```

```
const int IN4 = 7; // Input 4
```

```
#define TX 10
```

```
#define RX 11
```

```
float target;
```

```
// defines pins numbers
```

```
const int depthtrigPin = 31;
```

```
const int depthechoPin = 32;
```

```
// defines variables
```

```
long depthduration;
```

```
double depth;
```

```
//Left Ultrasonic declarations
```

```
const int lefttrigPin = 27;
```

```
const int leftechoPin = 28;
```

```
long leftduration;
```

```
float leftObstacle;
```

```
//Right Ultrasonic declarations
```

```
const int righttrigPin = 29;
```

```
const int rightechoPin = 30;
```

```
long righduration;
```

```
float rightObstacle;
```

```
//Front Ultrasonic declarations
```

```
const int fronttrigPin = 33;
const int frontechoPin = 34;
long frontduration;
float frontObstacle;

float final_x = 3.81;
float final_y = 1.56;
float distance;
float leftdistance;
float rightdistance;

#define TdsSensorPin A0
#define VREF 5.0    // analog reference voltage(Volt) of the ADC
#define SCOUNT 30    // sum of sample point
int analogBuffer[SCOUNT]; // store the analog value in the array, read from ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0, copyIndex = 0;
float averageVoltage = 0, tdsValue = 0, temperature = 25;
String saltornot = "";

#define S0 35
#define S1 36
#define S2 37
#define S3 38
#define sensorOut 39

int redfrequency = 0;
int bluefrequency = 0;
int greenfrequency = 0;
boolean greencolor = false;
```

```
String tdsVal;  
String depthVal;  
String colorVal;
```

```
void setup() {  
  pinMode(leftPWM, OUTPUT);  
  pinMode(rightPWM, OUTPUT);  
  pinMode(leftDirection, OUTPUT);  
  pinMode(rightDirection, OUTPUT);  
  
  pinMode(TdsSensorPin, INPUT);  
  
  pinMode(S0, OUTPUT);  
  pinMode(S1, OUTPUT);  
  pinMode(S2, OUTPUT);  
  pinMode(S3, OUTPUT);  
  pinMode(sensorOut, INPUT);  
  
  // Setting frequency-scaling to 20%  
  digitalWrite(S0,HIGH);  
  digitalWrite(S1,LOW);  
  
  pinMode(depthtrigPin, OUTPUT); // Sets the trigPin as an Output  
  pinMode(deptechoPin, INPUT); // Sets the echoPin as an Input  
  
  pinMode(lefttrigPin, OUTPUT); // Sets the trigPin as an Output  
  pinMode(leftechoPin, INPUT); // Sets the echoPin as an Input  
  pinMode(righttrigPin, OUTPUT); // Sets the trigPin as an Output  
  pinMode(rightechoPin, INPUT); // Sets the echoPin as an Input  
  pinMode(fronttrigPin, OUTPUT); // Sets the trigPin as an Output  
  pinMode(frontechoPin, INPUT); // Sets the echoPin as an Input
```

```

// WiFi pin modes
pinMode(TX, OUTPUT);
pinMode(RX, INPUT);

// Set H-bridge pins as OUTPUT
pinMode(ENB, OUTPUT);
pinMode(IN3, OUTPUT);
pinMode(IN4, OUTPUT);

// Initialize motor in the stop state
digitalWrite(ENB, LOW);
digitalWrite(IN3, LOW);
digitalWrite(IN4, LOW);
Enes100.begin("Water Tribe", WATER, 217, 10, 11); // Team Name: Water Tribe, Mission:
WATER, TX: 10, RX: 11
delay(10300);
Serial.begin(9600);
}

void loop() {

faceMissionSite();
//Movement Algorithm 1
//turnLeft();
//delay(500);
rightdistance = rightultrameasure();
leftdistance = leftultrameasure();
if(leftdistance < rightdistance){
    deliverymissionalgorithm1();
    if(checkForObstacles()){//Block 3

```



```

shiftLeft();//Block 2
if(checkForObstacles()){//Block 2
  shiftLeft();//Block 1
  moveForward();
  delay(3800);//Block 4
  if(checkForObstacles()){
    shiftRight();
    if(checkForObstacles()){//Block 5
      shiftRight();//Block 6
      moveForward();
      delay(3800);
      shiftLeft();
      shiftLeft();
      moveForward();
      delay(3800);//Finish
    }
  }
  else{
    moveForward();
    delay(3800);
    shiftLeft();
    moveForward();
    delay(3800);//Finish
  }
}
else{
  moveForward();
  delay(5000);
}
}
else{
  moveForward();

```

```
    delay(3800);
    shiftLeft();//Block 4
    if(checkForObstacles()){
        shiftRight();
        if(checkForObstacles()){//Block 5
            shiftRight();//Block 6
            moveForward();
            delay(3800);
            shiftLeft();
            shiftLeft();
            moveForward();
            delay(3500);//Finish
        }
    }
    else{
        moveForward();
        delay(3800);
        shiftLeft();
        moveForward();
        delay(3800);//Finish
    }
}
else{
    moveForward();
    delay(5000);
}
}
}
else{//Block 4
    moveForward();
    delay(2500);
    shiftLeft();
```

```

    shiftLeft();
    moveForward();
    delay(2500);
}
}
else{
    deliverymissionalgorithm2();
    shiftRight();
    shiftRight();
    if(checkForObstacles()){//Block 3
        shiftLeft();//Block 2
        if(checkForObstacles()){//Block 2
            shiftLeft();//Block 1
            moveForward();
            delay(3800);//Block 4
            if(checkForObstacles()){
                shiftRight();
                if(checkForObstacles()){//Block 5
                    shiftRight();//Block 6
                    moveForward();
                    delay(3800);
                    shiftLeft();
                    shiftLeft();
                    moveForward();
                    delay(3800);//Finish
                }
            }
        }
    }
    else{
        moveForward();
        delay(3800);
        shiftLeft();
        moveForward();
    }
}

```

```

        delay(3800);//Finish
    }
}
else{
    moveForward();
    delay(5000);
}
}
else{
    moveForward();
    delay(3800);
    shiftLeft();//Block 4
    if(checkForObstacles()){
        shiftRight();
        if(checkForObstacles()){//Block 5
            shiftRight();//Block 6
            moveForward();
            delay(3800);
            shiftLeft();
            shiftLeft();
            moveForward();
            delay(3800);//Finish
        }
    }
    else{
        moveForward();
        delay(3800);
        shiftLeft();
        moveForward();
        delay(3800);//Finish
    }
}
}

```

```

    else{
        moveForward();
        delay(5000);
    }
}
}
else{//Block 4
    moveForward();
    delay(2500);
    shiftLeft();
    shiftLeft();
    moveForward();
    delay(2500);
}
}
}

void faceMissionSite() {
    float locForFace = Enes100.getTheta();
    if(Enes100.getX() < 1){
        while(!(Enes100.getTheta() >= 1.49) || !(Enes100.getTheta() < 1.65)){
            if(locForFace > 0){
                turnLeft();
                delay(500);
                brake();
                delay(2000);
                Enes100.println(Enes100.getTheta());
            }
            else{
                turnRight();
                delay(500);
            }
        }
    }
}

```

```

        brake();
        delay(2000);
        Enes100.println(Enes100.getTheta());
    }

}

}

else{
    while(!(Enes100.getTheta() <= -1.48) || !(Enes100.getTheta() > -1.63)){
        if(locForFace > 0){
            turnRight();
            delay(300);
            brake();
            delay(2000);
            Enes100.println(Enes100.getTheta());
        }
        else{

            turnLeft();
            delay(300);
            brake();
            delay(2000);
            Enes100.println(Enes100.getTheta());
        }

    }

}

}

void deliverymissionalgorithm1(){

```

```
faceMissionSite();
moveForward();
delay(4500);
brake();
delay(2000);
//Pulley stuff
rotateCounterclockwise(150); // 255 is the PWM value for maximum speed
delay(1000);

// Stop the motor
stopMotor();
delay(5000);
depthVal = measureDepth();
Enes100.println("The depth is: " + depthVal);
rotateClockwise(150);
delay(800);
colorVal = measureColor();
Enes100.println("Color:" + colorVal);
stopMotor();
delay(1000);
rotateClockwise(150);
delay(400);
stopMotor();
delay(1000);
tdsVal = measureTDS();
Enes100.println("Water Type:" + measureTDS());
delay(1000);

reverse();
delay(4300);
```

```
turnRight();  
delay(3500);  
brake();  
moveForward();  
delay(2500);  
brake();  
delay(1000);  
}
```

```
void deliverymissionalgorithm2(){  
    moveForward();  
    delay(4300);  
    brake();  
    delay(2000);
```

```
//Pulley stuff
```

```
rotateCounterclockwise(150); // 255 is the PWM value for maximum speed  
delay(1000);
```

```
// Stop the motor
```

```
stopMotor();  
delay(5000);  
depthVal = measureDepth();  
Enes100.println("The depth is: " + depthVal);  
rotateClockwise(150);  
delay(800);  
colorVal = measureColor();  
Enes100.println("Color:" + colorVal);  
stopMotor();  
delay(1000);  
rotateClockwise(150);
```



```
delay(400);
stopMotor();
delay(1000);
tdsVal = measureTDS();
Enes100.println("Water Type:" + measureTDS());
delay(1000);
```

```
reverse();
delay(4300);
turnLeft();
delay(3000);
brake();
moveForward();
delay(2500);
brake();
delay(1000);
}
```

```
bool checkForObstacles(){
    int distance = 0;
    distance = frontultrameasure();
    if(distance < 75){
        return true;
    }
    return false;
}
```

```
void shiftRight(){
    turnRight();
    delay(3200);
    moveForward();
}
```

```

    delay(3000);
    turnLeft();
    delay(3200);
}

void shiftLeft() {
    turnLeft();
    delay(3900);
    moveForward();
    delay(3000);
    turnRight();
    delay(3300);
}

void rotateCounterclockwise(int speed) {
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    analogWrite(ENB, speed); // Set the speed using PWM
}

void rotateClockwise(int speed) {
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, speed); // Set the speed using PWM
}

// Function to stop the motor
void stopMotor() {
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, 0); // Set the speed to 0 to stop the motor
}

```

```

float leftultrameasure(){
    digitalWrite(lefttrigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(lefttrigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(lefttrigPin, LOW);
    leftduration = pulseIn(leftechoPin, HIGH);
    leftObstacle = leftduration * 0.034 / 2;
    return leftObstacle;
}

float rightultrameasure(){
    digitalWrite(righttrigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(righttrigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(righttrigPin, LOW);
    rightduration = pulseIn(rightechoPin, HIGH);
    rightObstacle = rightduration * 0.034 / 2;
    return rightObstacle;
}

float frontultrameasure(){
    //Front ultra measure
    digitalWrite(fronttrigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(fronttrigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(fronttrigPin, LOW);
    frontduration = pulseIn(frontechoPin, HIGH);
    frontObstacle = frontduration * 0.034 / 2;
    return frontObstacle;
}

```

```
void turnLeft(){
    //Turn Left
    analogWrite(leftPWM, 100);//changed
    digitalWrite(leftDirection, HIGH);
    analogWrite(rightPWM, 100);//changed
    digitalWrite(rightDirection, HIGH);
}
void turnRight(){
    //Turn Right
    analogWrite(leftPWM, 100);
    digitalWrite(leftDirection, LOW);
    analogWrite(rightPWM, 100);
    digitalWrite(rightDirection, LOW);
}
void moveForward(){
    //Move Forward
    analogWrite(leftPWM, 100);
    digitalWrite(leftDirection, LOW);
    analogWrite(rightPWM, 100);
    digitalWrite(rightDirection, HIGH);
}
void reverse(){
    analogWrite(leftPWM, 100);
    digitalWrite(leftDirection, HIGH);
    analogWrite(rightPWM, 100);
    digitalWrite(rightDirection, LOW);
}
void brake(){
    //Stop
```

```

analogWrite(leftPWM, 200);
digitalWrite(leftDirection, LOW);
analogWrite(rightPWM, 200);
digitalWrite(rightDirection, LOW);
}

```

```

String measureTDS() {
    static unsigned long analogSampleTimepoint = millis();
    if (millis() - analogSampleTimepoint > 4000) //every 40 milliseconds, read the analog value
    from the ADC
    {
        analogSampleTimepoint = millis();
        analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin); //read the analog value and
    store into the buffer
        analogBufferIndex++;
        if (analogBufferIndex == SCOUNT)
            analogBufferIndex = 0;
    }
    static unsigned long printTimepoint = millis();
    if (millis() - printTimepoint > 8000)
    {
        printTimepoint = millis();
        for (copyIndex = 0; copyIndex < SCOUNT; copyIndex++)
            analogBufferTemp[copyIndex] = analogBuffer[copyIndex];
        averageVoltage = getMedianNum(analogBufferTemp, SCOUNT) * (float)VREF / 1024.0; //
    read the analog value more stable by the median filtering algorithm, and convert to voltage value
        float compensationCoefficient = 1.0 + 0.02 * (temperature - 25.0); //temperature compensation
    formula: fFinalResult(25°C) = fFinalResult(current)/(1.0+0.02*(fTP-25.0));
        float compensationVoltage = averageVoltage / compensationCoefficient; //temperature
    compensation
}

```

```
    tdsValue = (133.42 * compensationVolatge * compensationVolatge * compensationVolatge -  
255.86 * compensationVolatge * compensationVolatge + 857.39 * compensationVolatge) * 0.5;
```

```
//convert voltage value to tds value
```

```
    if(tdsValue > 120){  
        return "Salt water";  
    }  
    else{  
        return "Freshwater";  
    }
```

```
  
    //Serial.print("voltage:");  
    //Serial.print(averageVoltage,2);  
    //Serial.print("V  ");  
    //Serial.print("TDS----Value:");  
    //Serial.print(saltornot);  
    //Serial.print(tdsValue, 0);  
    //Serial.println("ppm");  
}
```

```
}
```

```
int getMedianNum(int bArray[], int iFilterLen)
```

```
{  
    int bTab[iFilterLen];  
    for (byte i = 0; i < iFilterLen; i++)  
        bTab[i] = bArray[i];  
    int i, j, bTemp;  
    for (j = 0; j < iFilterLen - 1; j++)  
    {  
        for (i = 0; i < iFilterLen - j - 1; i++)  
        {
```

```

    if (bTab[i] > bTab[i + 1])
    {
        bTemp = bTab[i];
        bTab[i] = bTab[i + 1];
        bTab[i + 1] = bTemp;
    }
}
}

if ((iFilterLen & 1) > 0)
    bTemp = bTab[(iFilterLen - 1) / 2];
else
    bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
return bTemp;
}

String measureColor(){
    // Setting red filtered photodiodes to be read
    digitalWrite(S2,LOW);
    digitalWrite(S3,LOW);
    // Reading the output frequency
    redfrequency = pulseIn(sensorOut, LOW);
    // Printing the value on the serial monitor
    Serial.print("R= "); //printing name
    Serial.print(redfrequency); //printing RED color frequency
    Serial.print(" ");
    delay(100);

    // Setting Green filtered photodiodes to be read
    digitalWrite(S2,HIGH);
    digitalWrite(S3,HIGH);
    // Reading the output frequency
    greenfrequency = pulseIn(sensorOut, LOW);

```

```
// Printing the value on the serial monitor
Serial.print("G= "); //printing name
Serial.print(greenfrequency); //printing RED color frequency
Serial.print(" ");
delay(100);
```

```
// Setting Blue filtered photodiodes to be read
digitalWrite(S2, LOW);
digitalWrite(S3, HIGH);
// Reading the output frequency
bluefrequency = pulseIn(sensorOut, LOW);
// Printing the value on the serial monitor
Serial.print("B= "); //printing name
Serial.print(bluefrequency); //printing RED color frequency
Serial.println(" ");
delay(100);
```

```
if(redfrequency < 260 && redfrequency > 240 && greenfrequency > 320 && greenfrequency
< 340 && bluefrequency > 360 && bluefrequency < 373){
    return "green";
}
return "clear water";
}
```

```
float measureDepth(){
    // Clears the trigPin
    digitalWrite(depthtrigPin, LOW);
    delayMicroseconds(2);

    // Sets the trigPin on HIGH state for 10 microseconds
    digitalWrite(depthtrigPin, HIGH);
    delayMicroseconds(10);
```



```
digitalWrite(depthtrigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
depthduration = pulseIn(depthechoPin, HIGH);

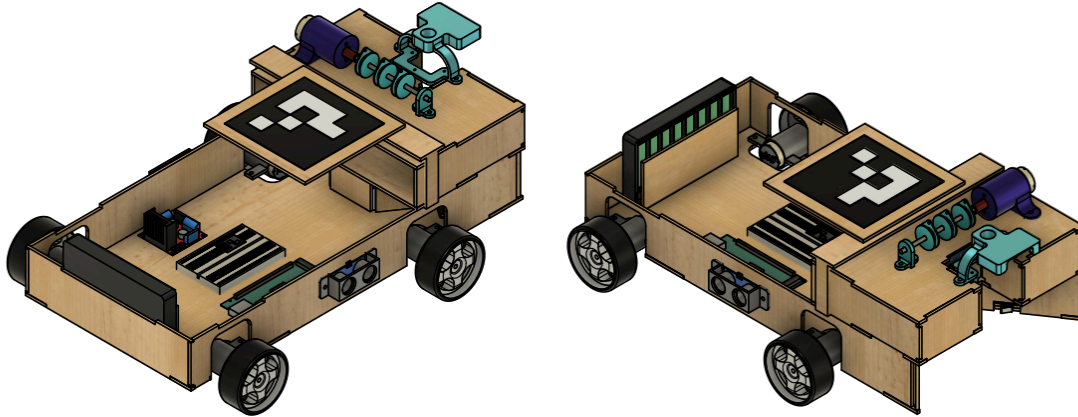
// Calculating the distance
depth = (((13.5 - (depthduration * 0.034 / 2) - 0.9 + 0.3) * 10)-10.0); //in mm

// Prints the distance on the Serial Monitor
//Serial.print("Distance: ");
//Serial.println(depth);

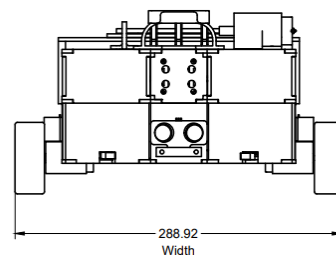
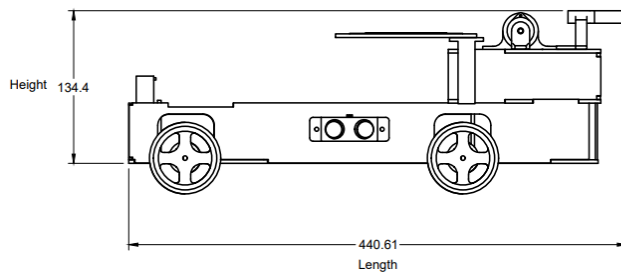
// Wait for 2 seconds before the next measurement
delay(2000);
return depth;
}
```

Compiled Set of Engineering Drawings

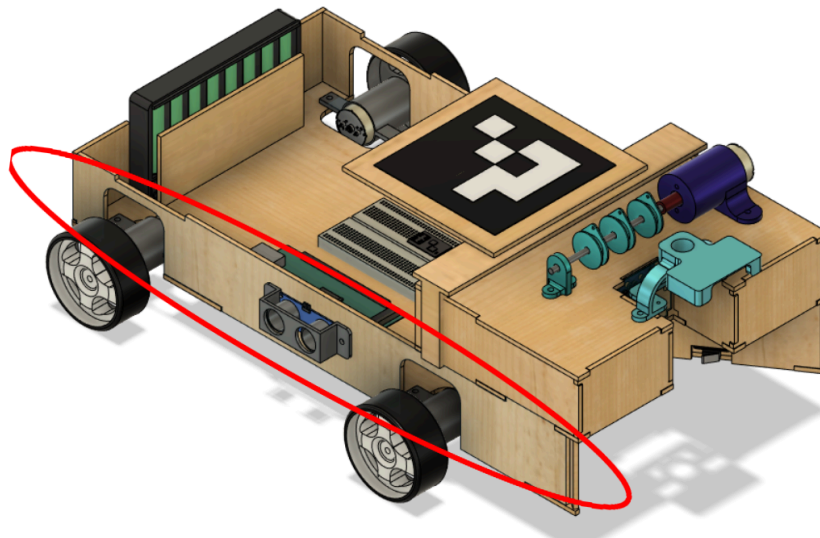
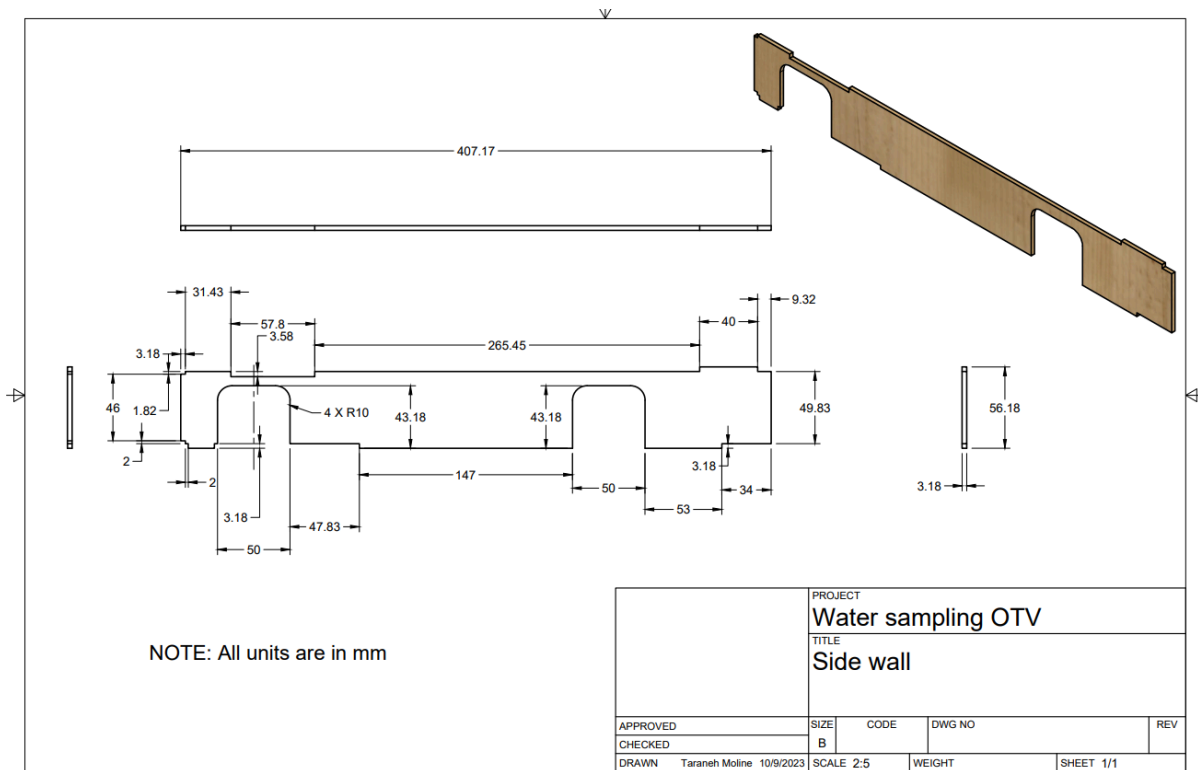
Overall OTV dimensions (all dimensions in mm)



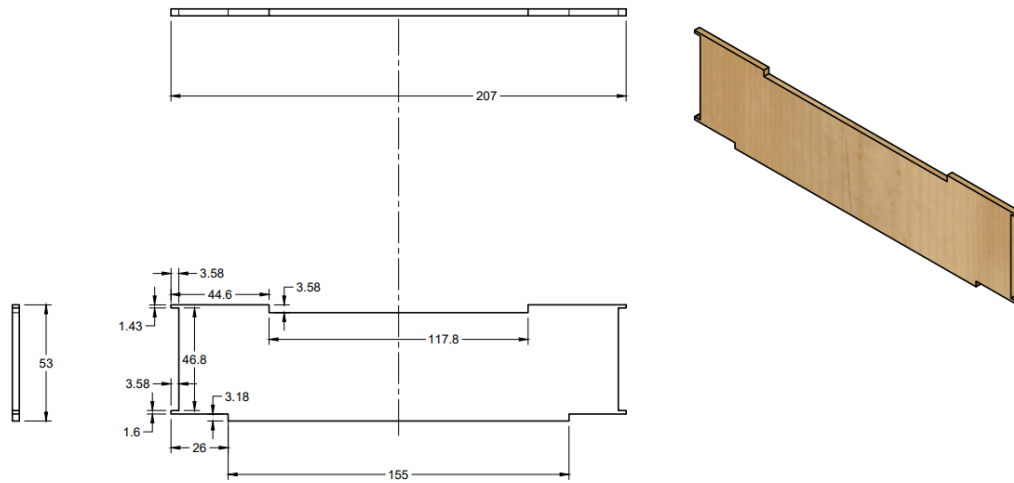
NOTE: All units are in mm



Laser-cut plywood: This is the sidewall of the OTV. The large gaps allow room for the wheel motors in the back, and the axle setup on the front. The grooves allow it to connect to the chassis, back wall, and the second level of the wooden structure. Quantity - 2

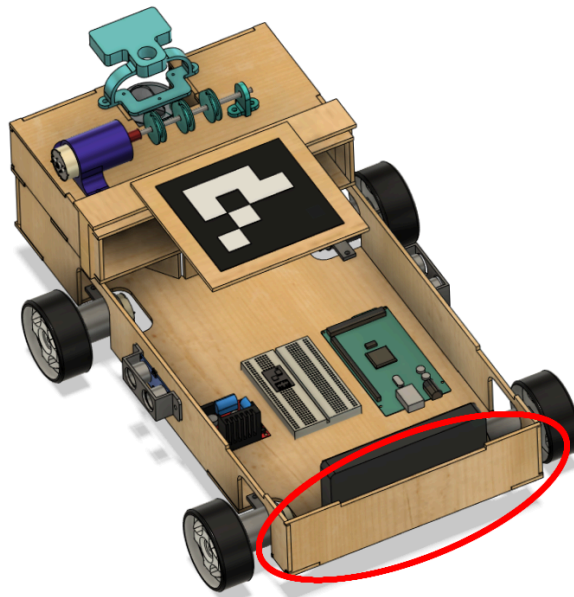


Laser-cut plywood: This is the back wall of the OTV. The grooves allow it to connect to the chassis and the side walls. Quantity - 1

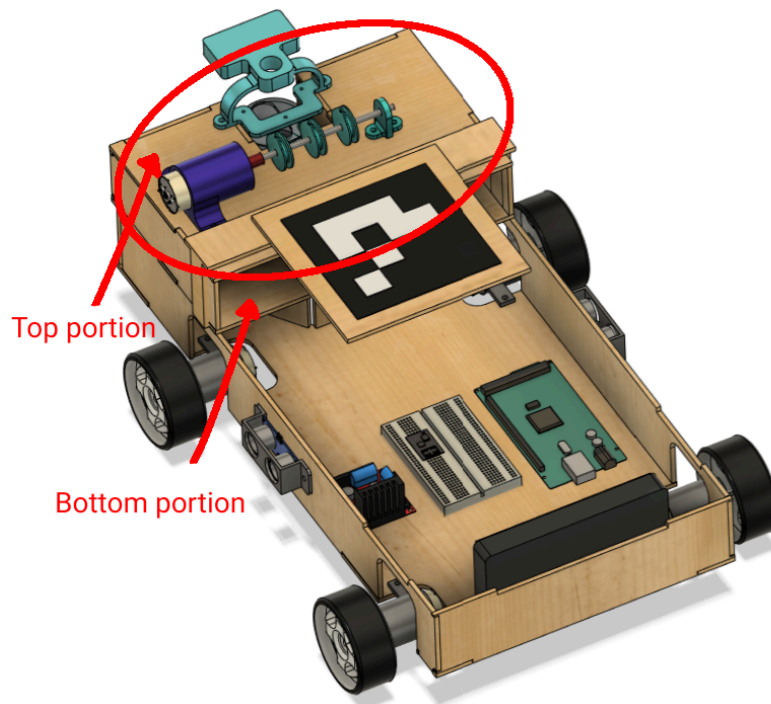
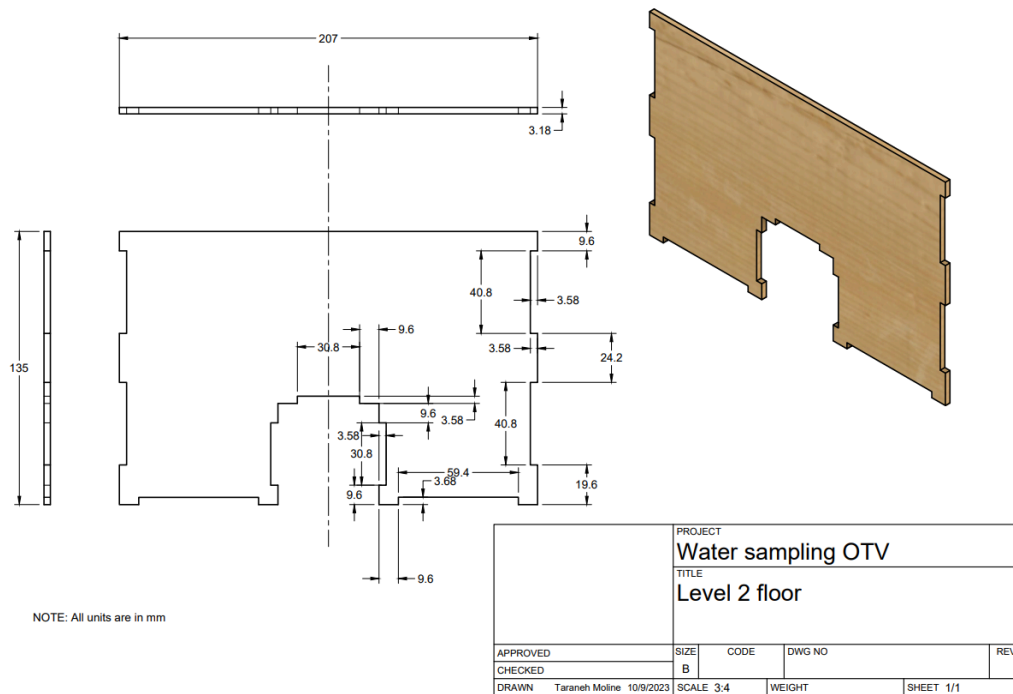


NOTE: All units are in mm

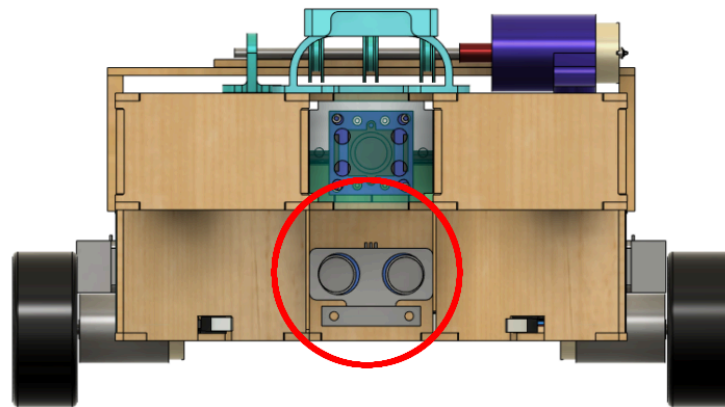
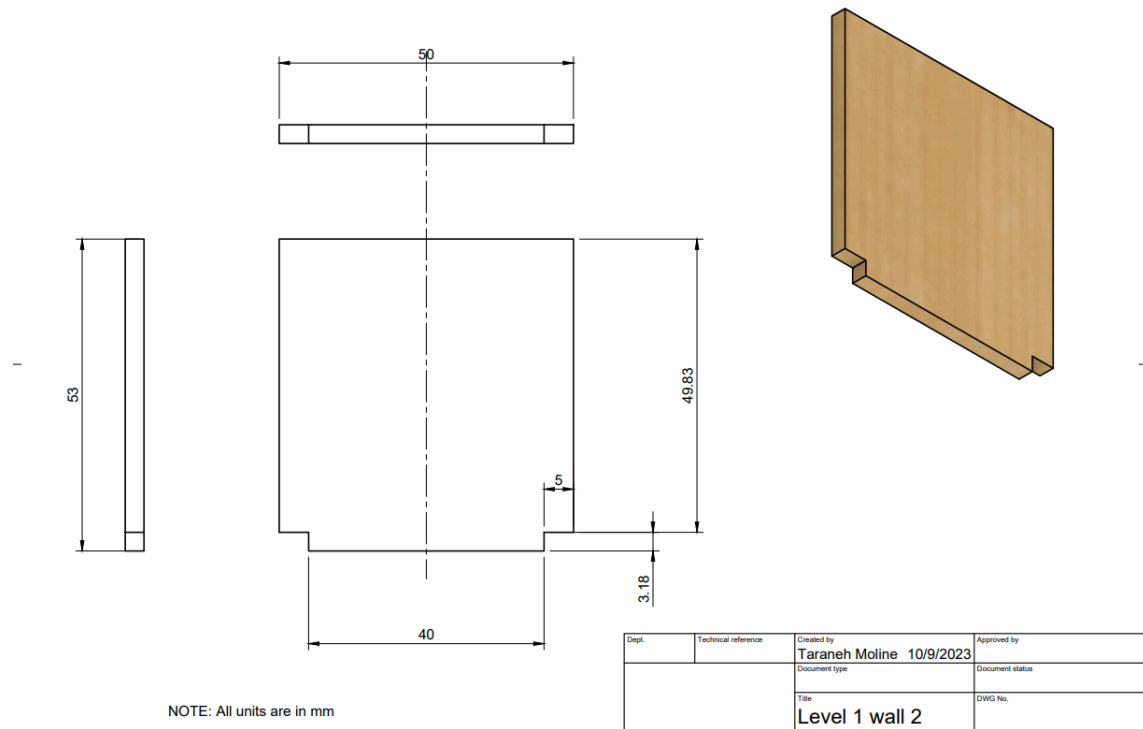
PROJECT				
Water sampling OTV				
TITLE				
Back wall Level 1				
APPROVED	SIZE	CODE	DWG NO	REV
CHECKED	B			
DRAWN	Taraneh Moline 10/10/2023	SCALE 3:4	WEIGHT	SHEET 1/1



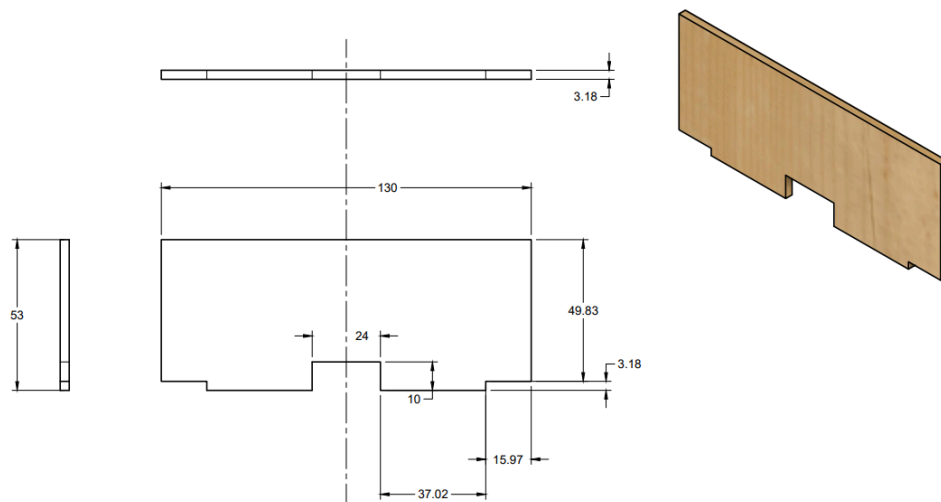
Laser-cut plywood: This is the floor that begins and ends the second level of the wooden structure, which contains the pulley system. The grooves allow it to connect to the level 1 and 2 sidewalls. Quantity - 2



Laser-cut plywood: This is one of the sidewalls for the second level of the wooden structure. The grooves connect it to the top and bottom of the second level floors. Quantity - 2

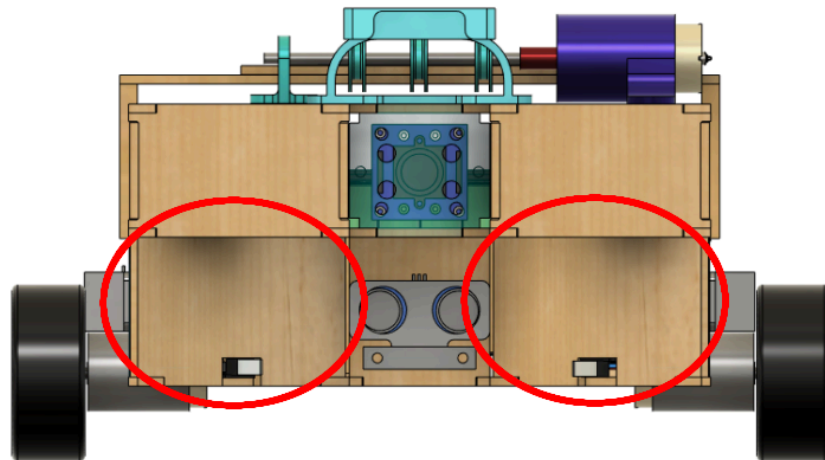


Laser-cut plywood: This is one of the slanted side walls that creates the triangle-like structure on the front of the OTV. The gap on the bottom creates room for the bumper switches, and the grooves connect it to the chassis. Quantity - 2

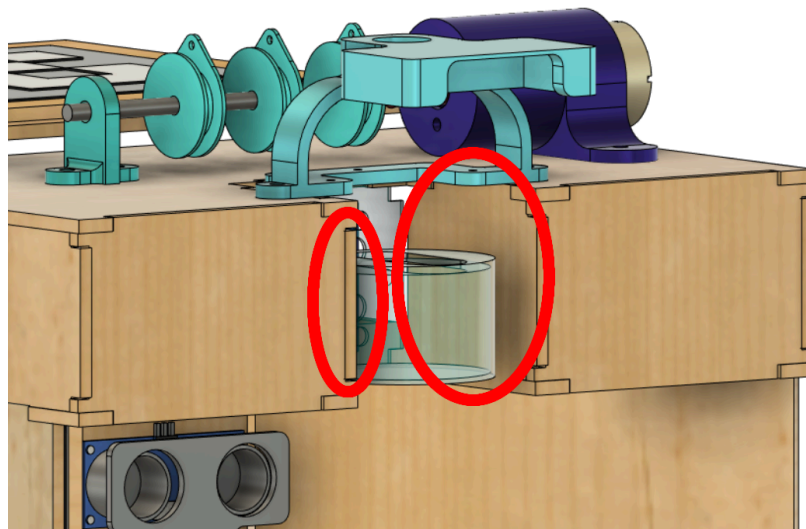
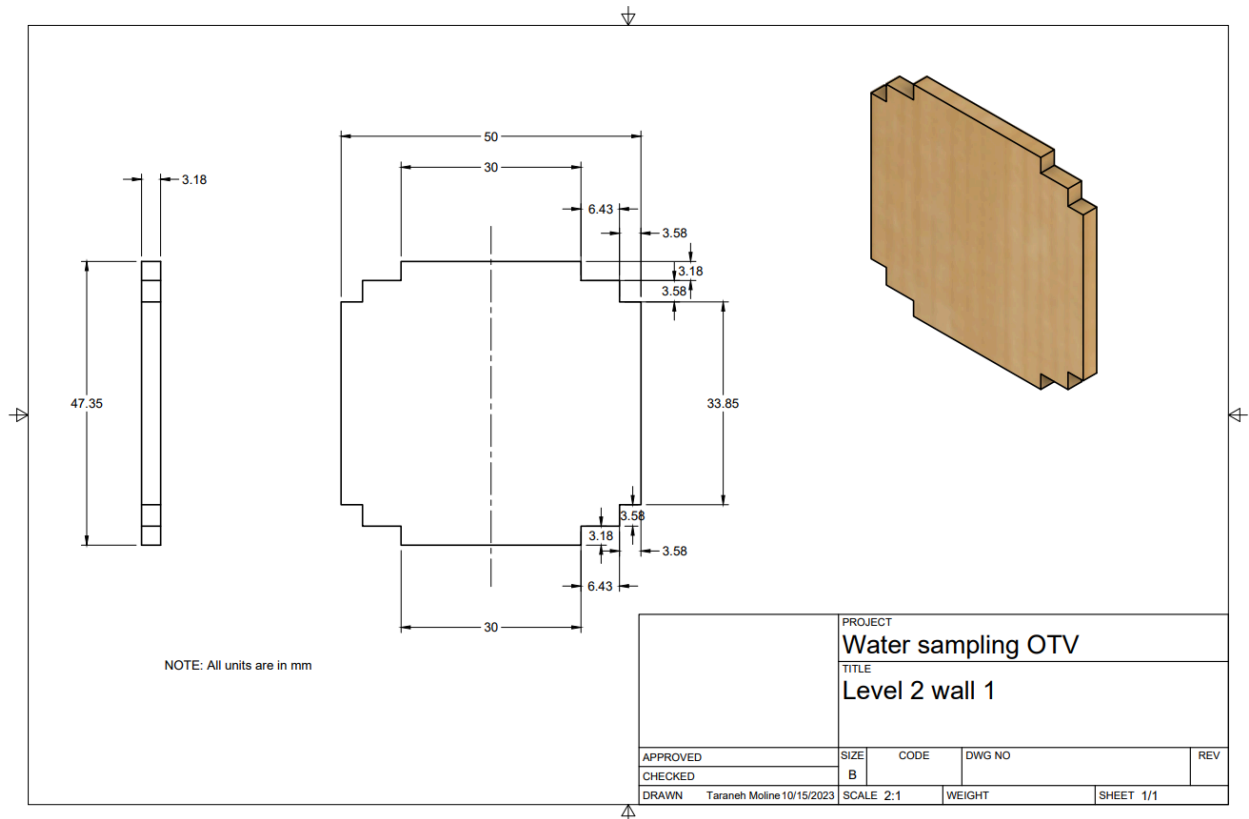


NOTE: All units are in mm

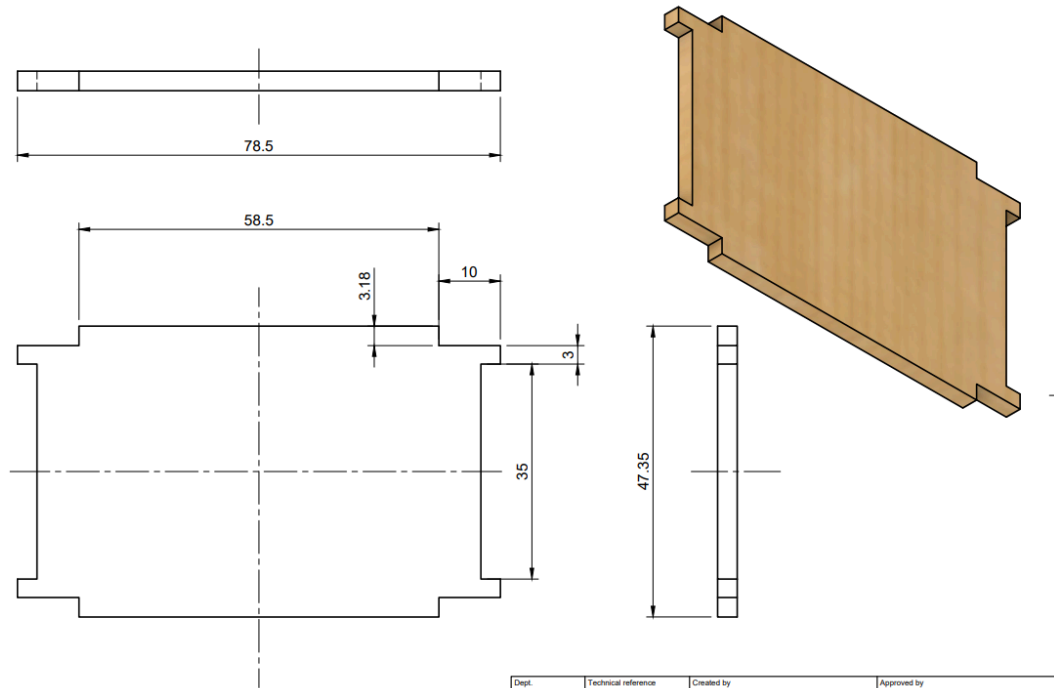
PROJECT				
Water sampling OTV				
TITLE				
Level 1 wall 3				
APPROVED	SIZE	CODE	DWG NO	REV
CHECKED	B			
DESIGNED	TECHNICAL	SCALE	DATE	SHEET



Laser-cut plywood: This is one of the sidewalls on the second level of the wooden structure. The grooves connect it to the second level floor, and the second level front walls. Quantity - 2

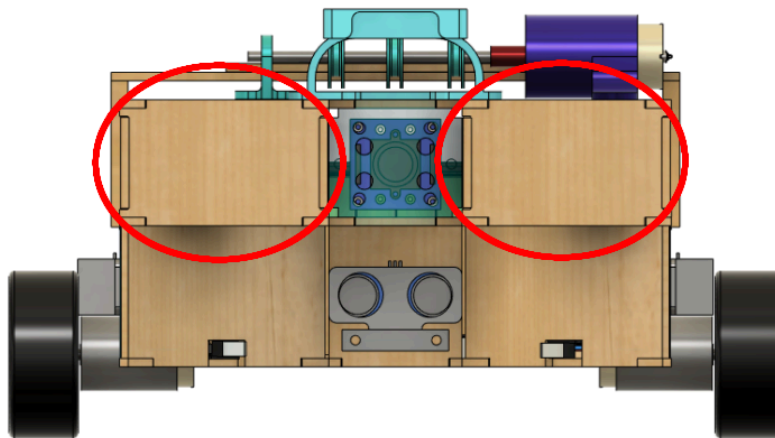


Laser-cut plywood: This is the front wall on the second level of the wooden structure. The grooves connect it to the second level sidewalls, and the second level floors. Quantity - 2

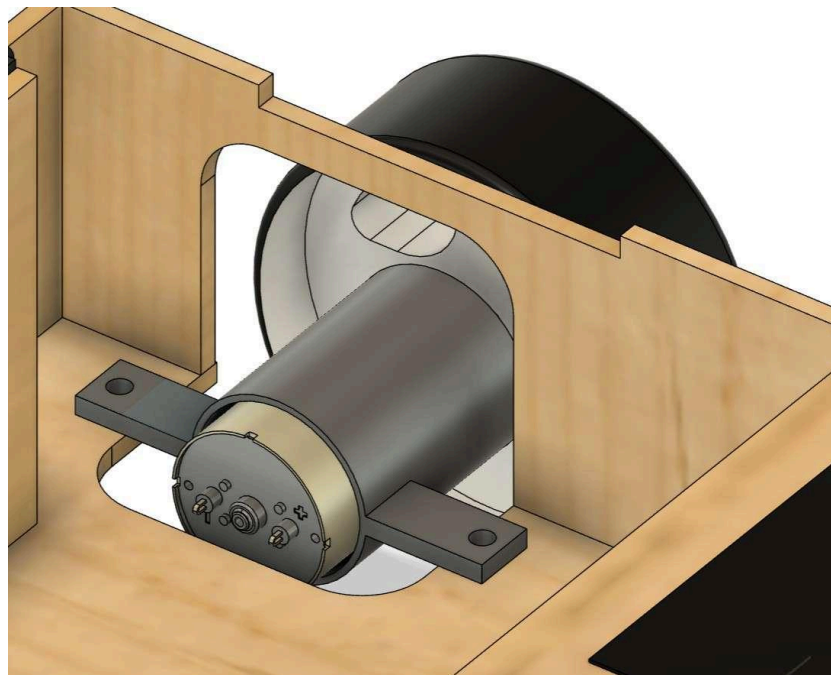
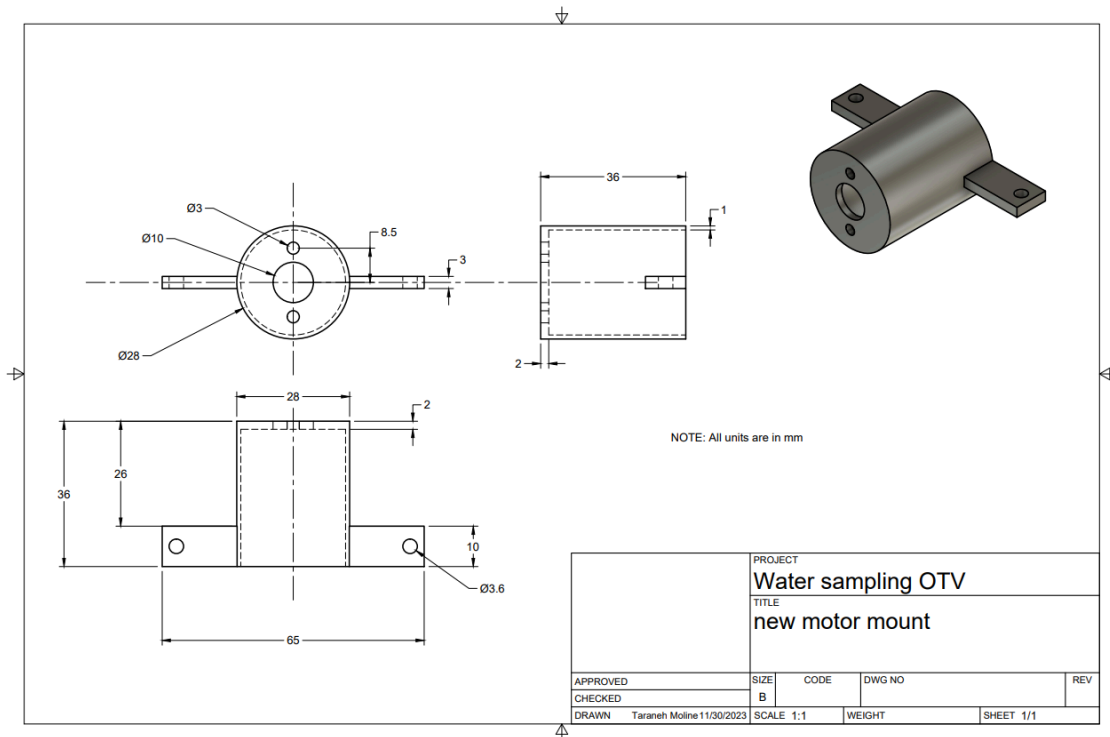


NOTE: All units are in mm

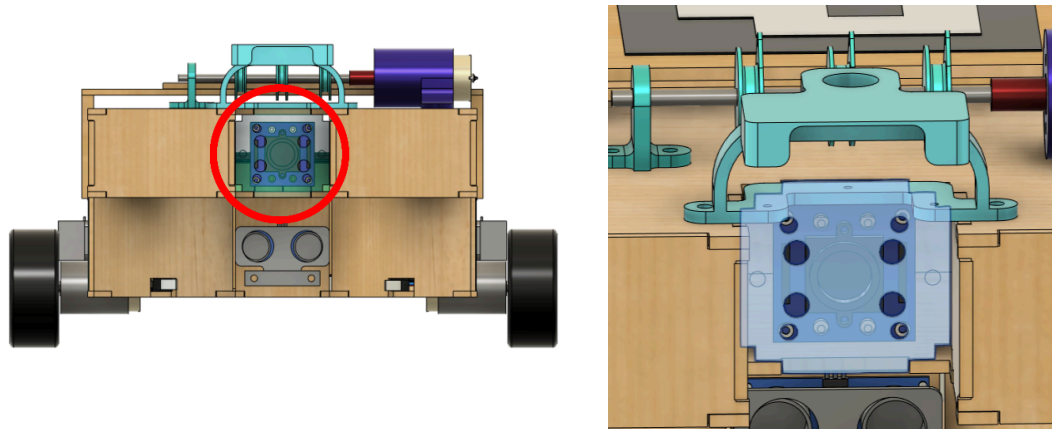
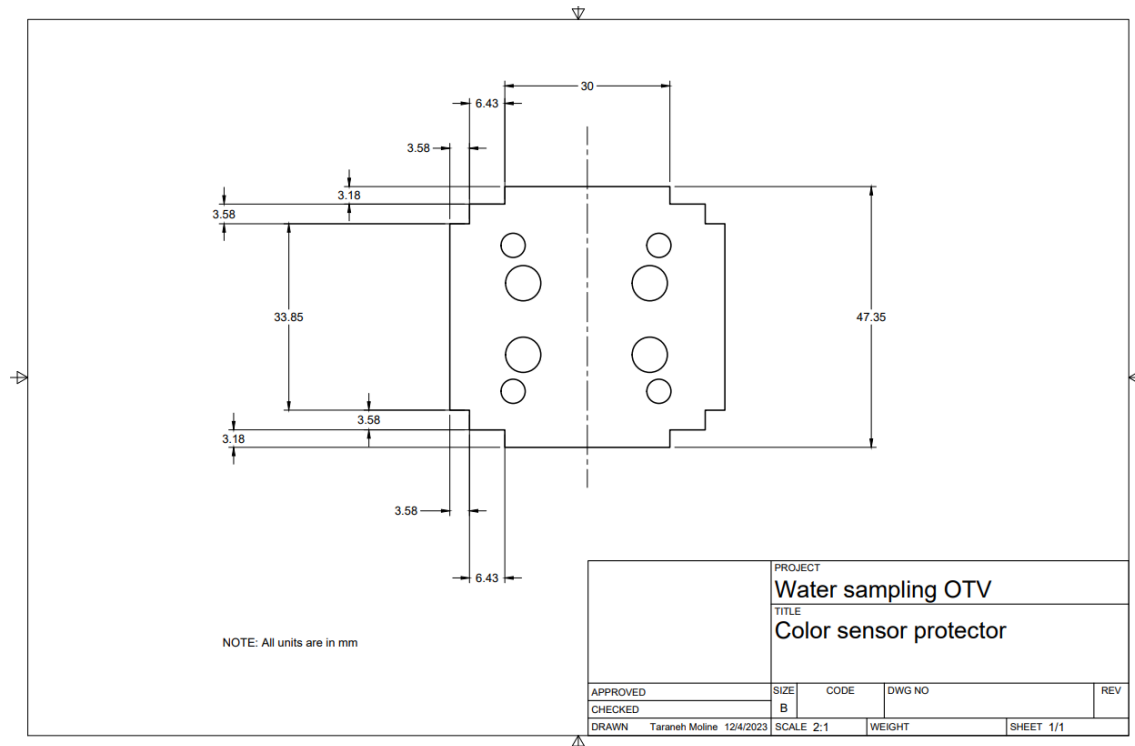
Dept.	Technical reference	Created by Taraneh Moline 10/9/2023	Approved by
		Document type	Document status
		Title Level 2 wall 2	DWG No.



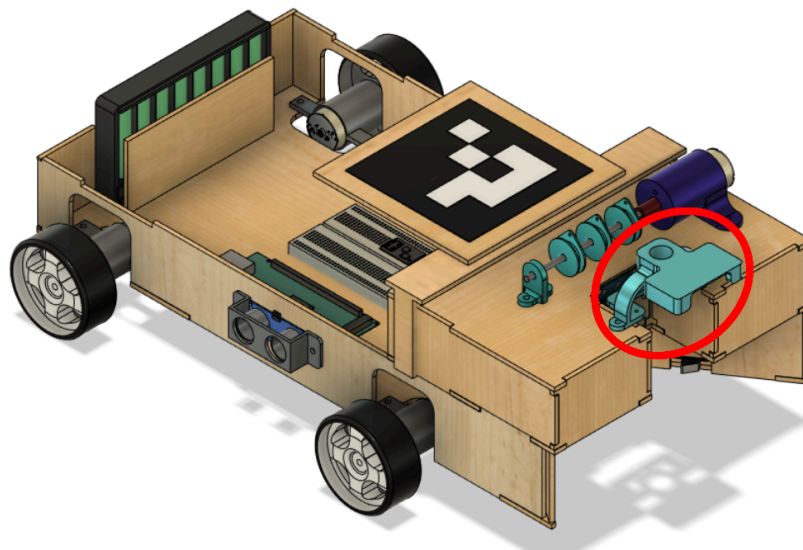
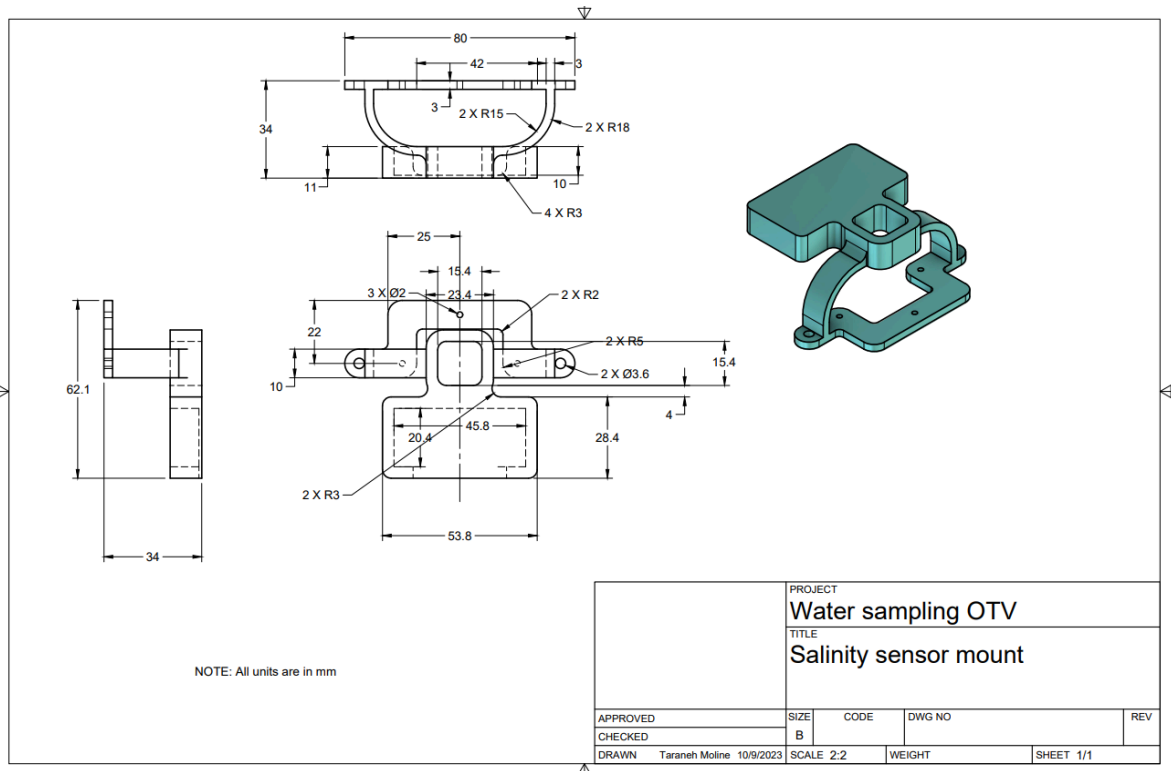
3D Print: This is the mount for all four motors. It was drilled into the chassis. Quantity - 4



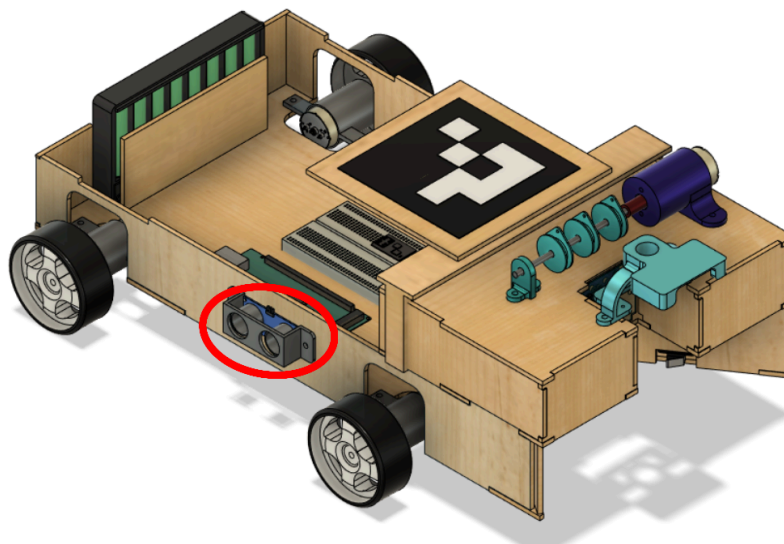
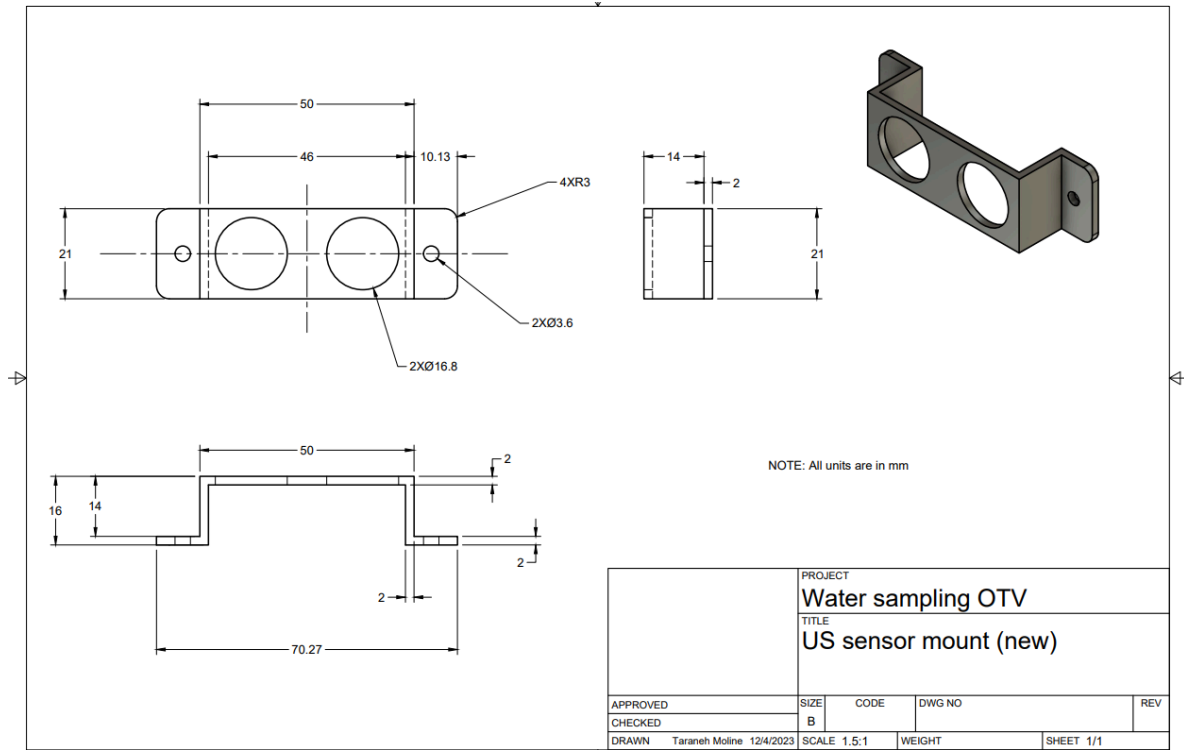
Laser-cut acrylic: This is the structure that protects the color sensor from water damage. The grooves connect it to the floor and sidewalls of the second level. Quantity - 1



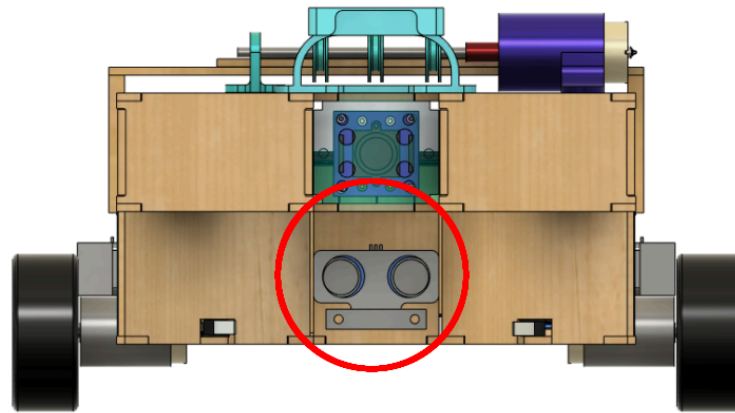
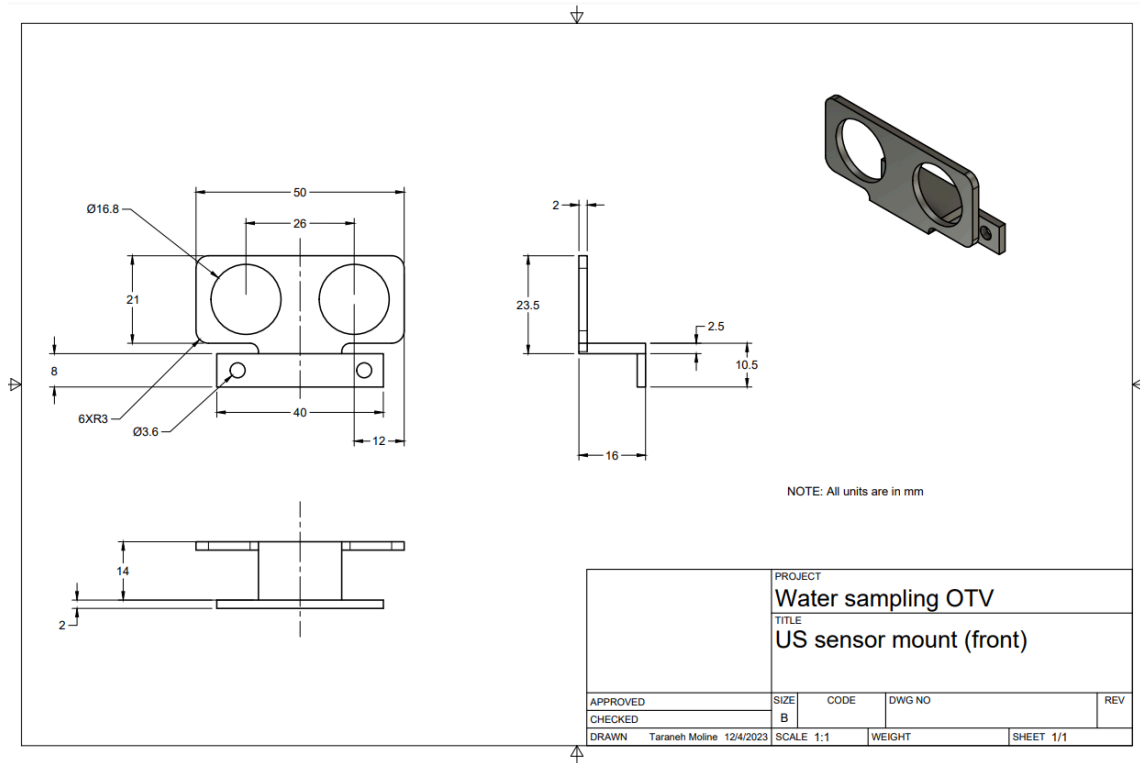
3D Print: This is the mount for the salinity sensor and the ultrasonic sensor used to detect water depth. The small holes held the pulley string, and they were aligned with the water cup to ensure stability. It was drilled into the top floor of the second level. Quantity - 1



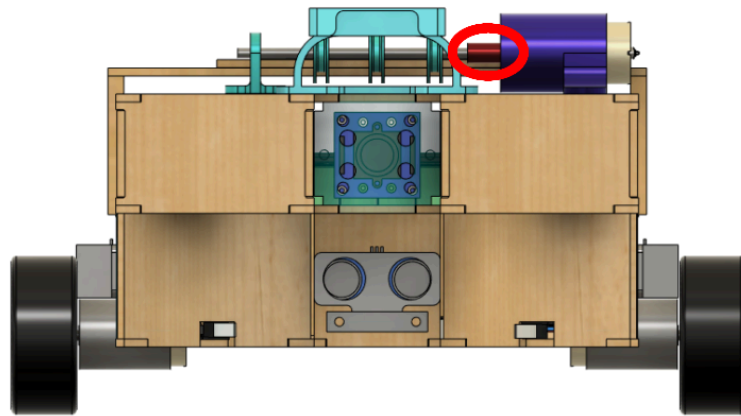
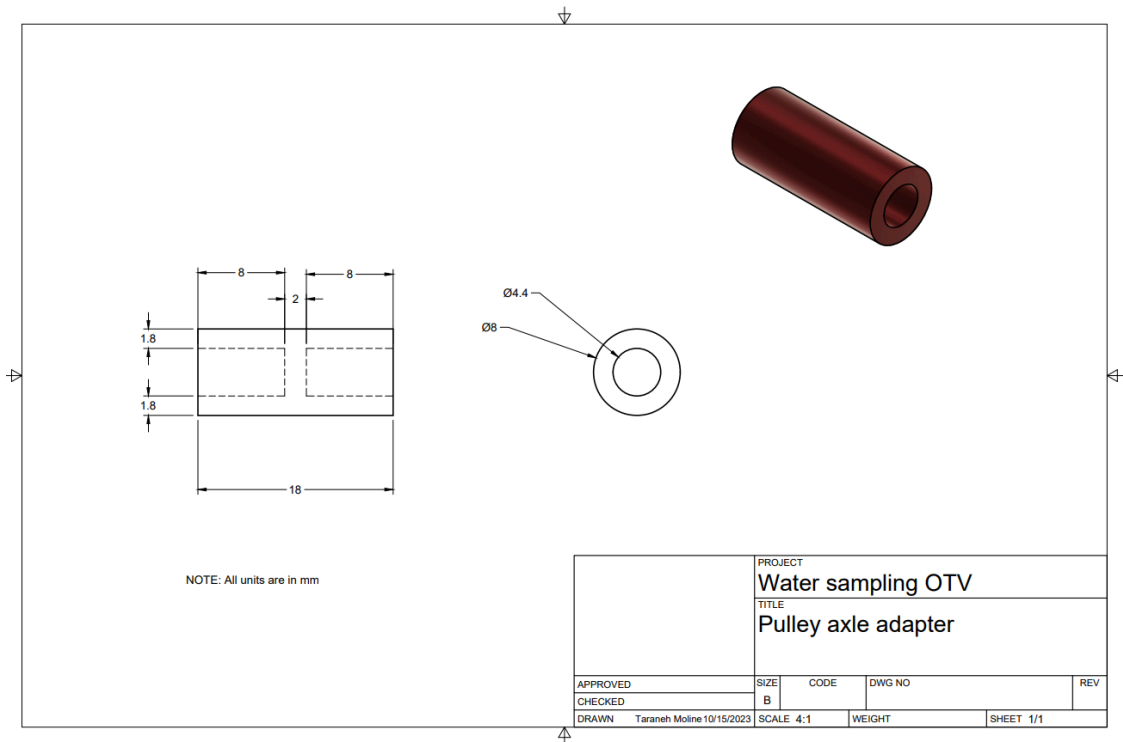
3D Print: This is the mount that held the ultrasonic sensors on the side of the OTV. It was drilled into the sidewalls. Quantity - 2



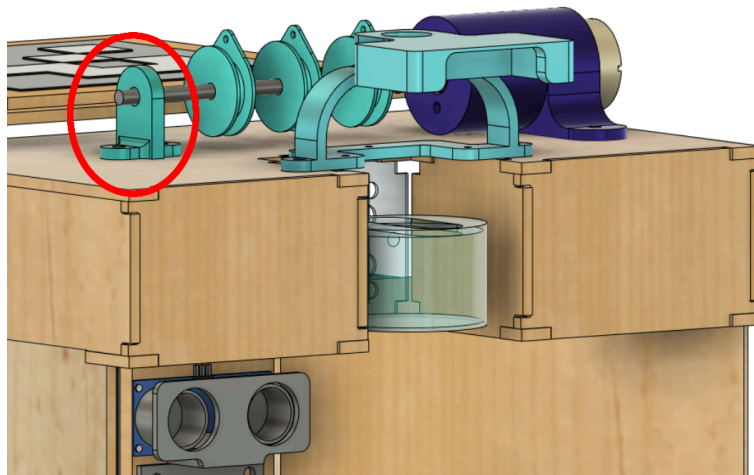
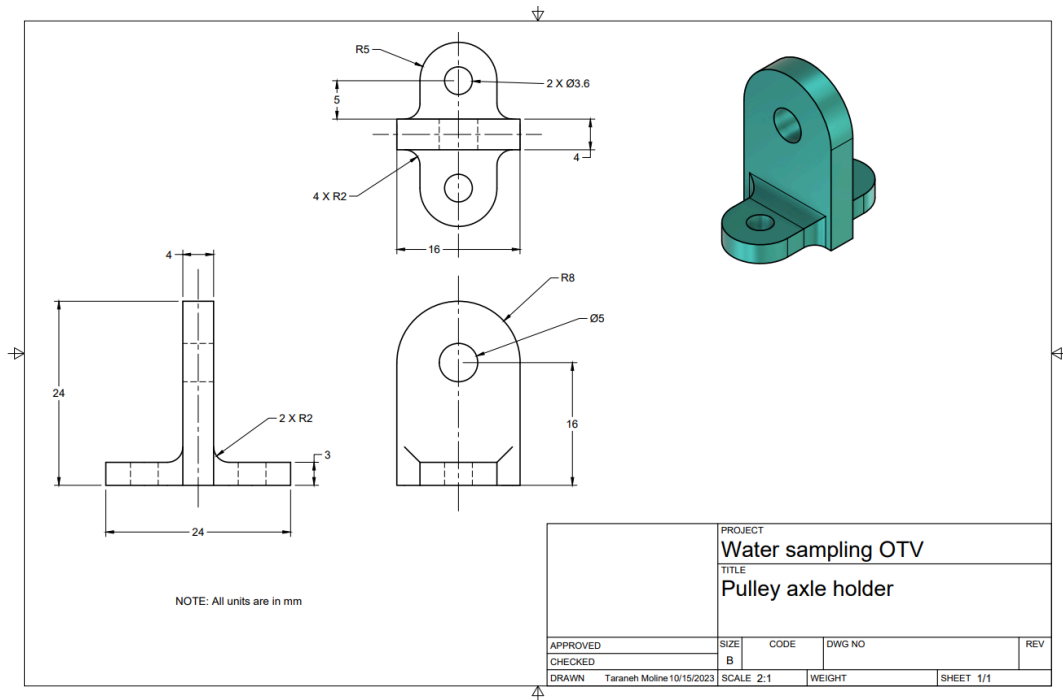
3D Print: This is the mount that holds the ultrasonic sensor on the front of the OTV. It was drilled into the front wall of the first level. Quantity - 1



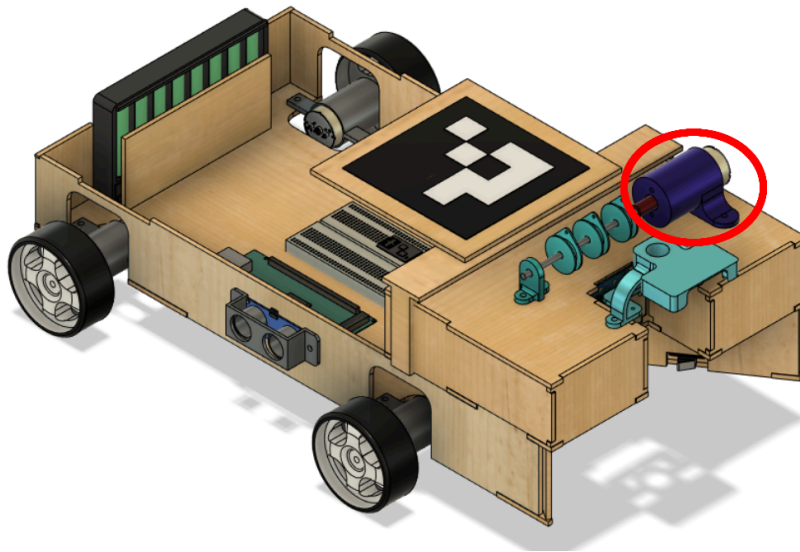
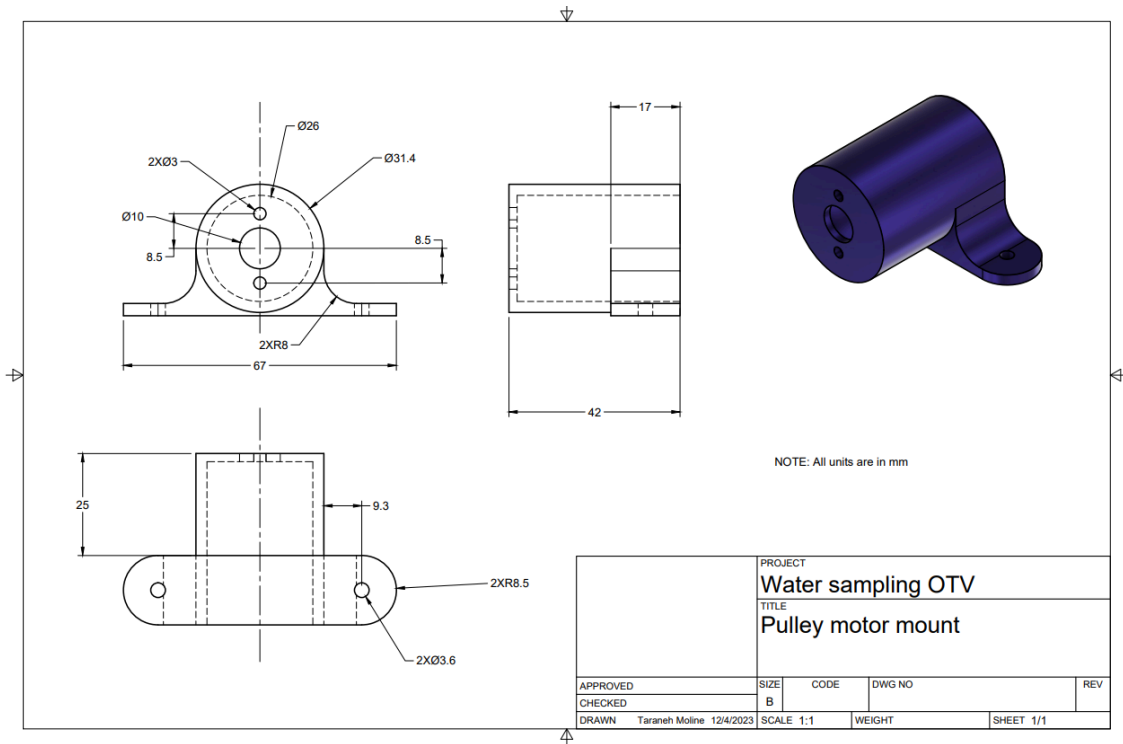
3D Print: This is the adapter that connected the axle from the pulley system motor to the longer axle. Quantity - 1



3D Print: This is the mount that held the end of the axle on the pulley system to ensure stability. The hole has a 0.7mm margin between itself and the axle to allow free rotation while still maintaining stability. It was drilled into the top floor of the second level. Quantity - 1

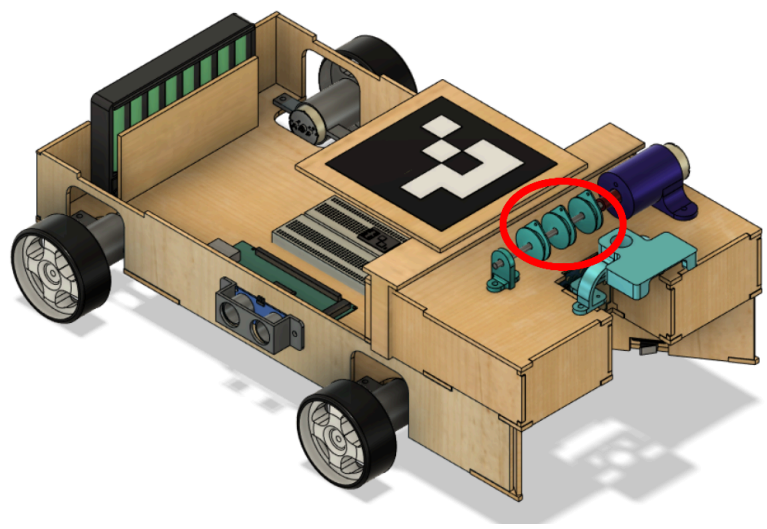
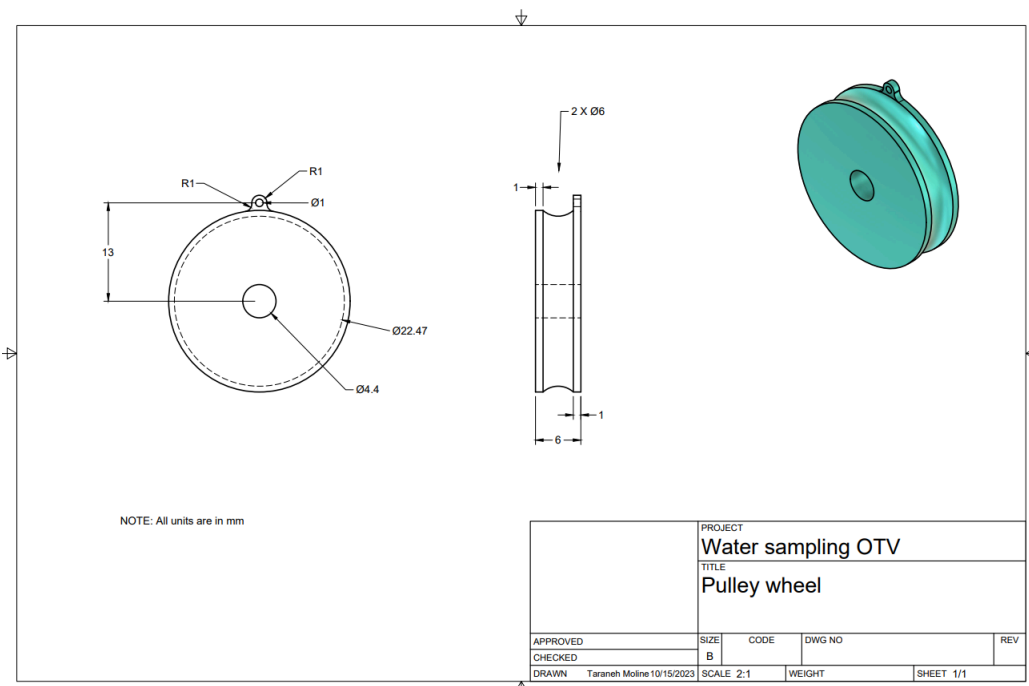


3D print: This is the mount for the motor being used in the pulley system. It was drilled into the top floor of the second level. Quantity - 1

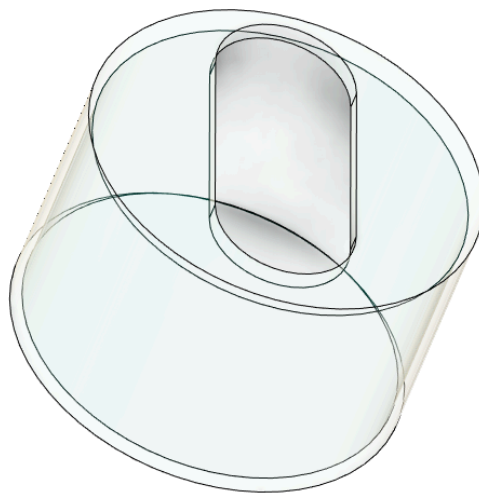
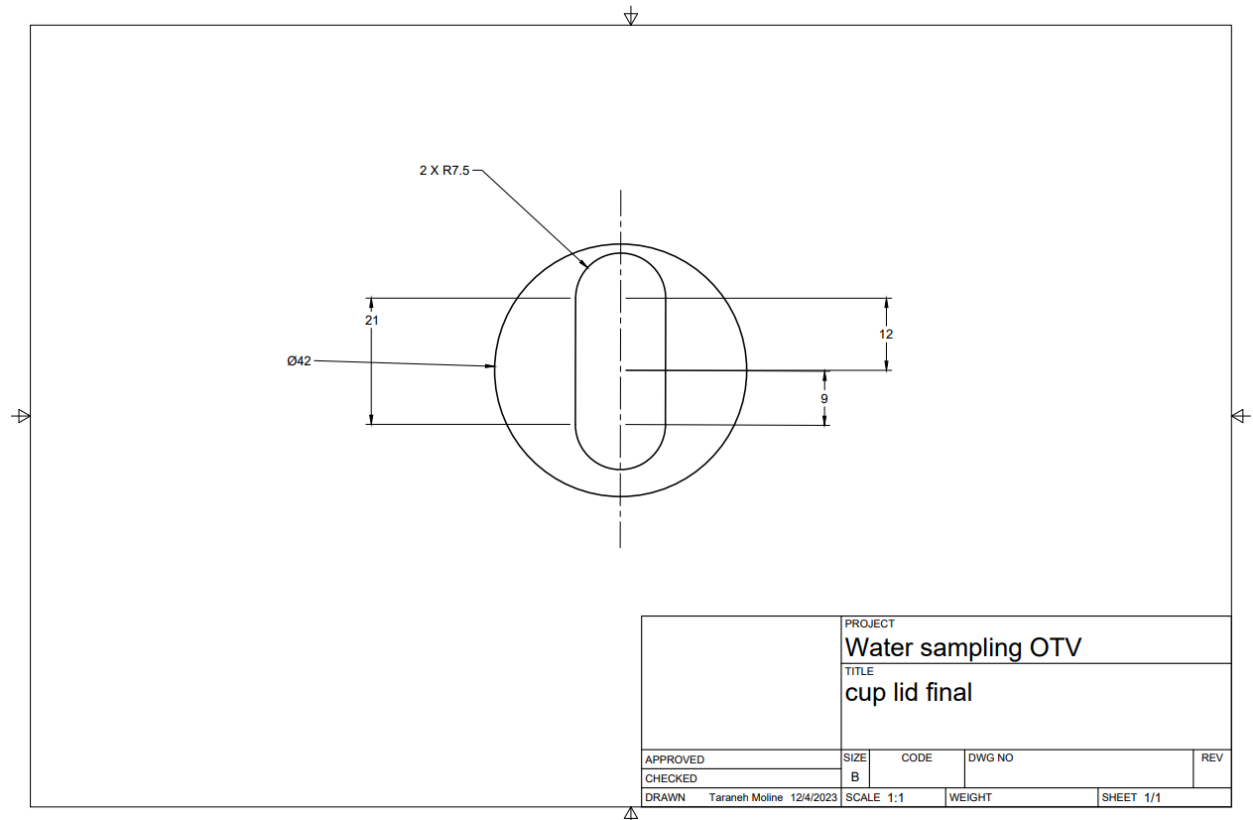


3D Print: This is the pulley wheel that held the string for the pulley system. The three of them were connected to the long axle attached to the motor, and they were aligned with the small holes on the salinity sensor mount. The small hole on top was where one end of the string was tied.

Quantity - 3



Laser-cut acrylic: This is the lid that was placed on top of the water collection cup. Its design allowed it to be angled down in order to collect a water sample from every possible water depth, including 20mm. Quantity - 1



Troubleshooting and Iteration

Tell the story of one part of your OTV design that was iterated throughout the life cycle of the project. What was the driving factor for iteration?

One part of our OTV that had to be iterated was the motor powered pulley cup system. Our main tasks as the water sampling group was to collect a 20 milliliter sample of water, perform a test with different sensors, and then safeguard the sample back to the site. Our team had a variety of ideas to accomplish this. The first idea was to use a suction type system where a tube would be inserted into the cup and then filled into a cup/container within the OTV. Another idea we had was to use a pulley system that would drop a cup down and hold the sample. Ultimately this was the method that we went with, since we found it easier to assemble with our team's skills.

For the pulley system, we decided whether to use an actual pulley system with gears that we can buy or to custom make gears. We realized that using gears would make things unnecessarily difficult, so we decided to use three rings that would hold and loop the line connected to the cup to raise it up and down. We would slide the rings onto an axle and connect it to the motor. Tara, who is really skilled with 3D designing, created the rings and fishing line was bought off of Amazon. We could've saved some money after discovering that there was some in the lab, but it did not affect us financially that much. One problem we encountered was that the axle we used to slide the rings on was not the same diameter as the axle in the motor. We planned on pulling the axle out from the motor and replacing it with a longer one. To fix this we created an adapter between the motor axle and axle with the rings. We hot glued the pieces together, but on our first attempt we put too much and it caused the axles to spin very slowly. The pieces were heated with a heat gun to be removed and a thinner layer was applied.

Now that we had the main pulley system finished, we had to create the cup. We first went with a 3D printed cup in order to save money from buying one. There were two problems with this cup, with the first being the size. It was a little too big in size and it would rub against the wood pieces when being brought up or down. Instead of creating a new cup since we were preoccupied with other pieces, we decided to sand the sides. This worked, but led to our second problem, which was the cup being unbalanced on the way up. We thought that perhaps the material of the cup made it heavier since air could slip through the thin space between the 3D printed material. We then created an acrylic cup, but it would still be slightly unbalanced. At first

we suggested having a new string system, but we ended up using a taller and thinner cup that we found, and putting a small coin in to balance the cup. We finally had the pulley cup system working effectively.

The driving factor for our iteration was obviously the problems we encountered, but also availability of resources and budget. Tara was the main member in our group who could create detailed 3D sketches, so she wasn't always available to design another piece as she was busy doing something else. We also wanted to save money as much as we can since we already spent a good portion on buying replacement parts like motors for the wheels.

Teamwork and Project Management

How did the team stay encouraged to persist through failure?

Like any engineering process, constructing our OTV was nothing if not challenging. From having to change the wheels, wiring the motors, getting the code just right, to repeatedly 3D printing parts, adjusting the cup dimensions, and increasing the density of the cup to let it sink, there were no shortages of failures where we had to improvise a solution.

Though, this is the beauty of engineering; the road to success is paved with failure. However, staying motivated to keep pushing through failure is key towards succeeding. For our team, there were three critical aspects that kept pushing us to push through: our goals/aspirations as a team, the team environment, and support provided from each individual team member.

We all shared goals which encouraged us to keep moving forward in the face of failure because we wanted to see our goals manifested. For example, we all wanted to learn valuable engineering skills/knowledge, whether it be circuits, mechanics, construction, CAD, or coding so that we could translate this skills/knowledge from the classroom into the workforce. We recognized the unique opportunity this class provided, as many other engineering programs do not construct full-fledged OTVs, so we wanted to take advantage of our resources and make the most of this chance. Despite knowing the OTV didn't need to be entirely successful to pass this class, we all wanted a functional OTV, and to utilize all available resources. This motivated us to keep working, keep brainstorming, and staying after class to continue working on our OTV instead of just giving up.

In order to brainstorm, team members must be willing to verbalize their ideas. In order to do this, feeling comfortable in the environment in which you're speaking can help people express themselves without leaving ideas out in fear of judgment/criticism. As a team, we have established a welcoming environment with each other that has been the most crucial encouragement to overcome failures. Since the beginning of the semester, the team has gotten to know each other better, shared many laughs and stories, as well as bonded through conversations and construction of the OTV itself. This has established a positive team environment, which makes it easier for us to ask questions about the project and express their ideas freely. It's much easier to talk to your friends than random strangers. Delegating work, communicating our ability in completing tasks, and overall having a cohesive flow in the team has allowed us to actually carry out the plans that we discuss in brainstorming sessions. We come into the lab knowing that

we have the support of our fellow team members, and this helps us persist in overcoming our failures.

Having each other's backs was incredibly motivating, because if you were stuck then all you had to do was ask the person next to you. Each member's words and actions encouraged the rest of the group to keep working and keep persisting. Ari made sure to check up on everyone's progress and help people improve their work. This motivated the rest of us to keep working, and he made sure not to leave anyone behind. Tara's empathy made sure everyone was included and treated with the most respect. Feeling valued as a team member encourages everyone to keep contributing to the team, instead of just giving up overall. Moreover, Tara answered any question, no matter how silly or serious, which made her approachable to work with. Her presence contributed to a comfortable environment to express new ideas and ask questions which is critical for a team to problem-solve and brainstorm together. Mahi connected with the team members, providing words of encouragement which added towards the positive team atmosphere and kept spirits up for everyone to keep working. Adi demonstrated dedication towards the project, being an extremely reliable and efficient team member and having technical experience in coding and wiring circuits. This was motivational because in order to come up with ideas, having technical knowledge is critical to figure out what ideas are plausible and which ones might not necessarily have merit. Jack added technical value to the team as well in mechanics and communicated efficiently, which led towards the team being motivated to ask questions without hesitation. Aldrin added to a positive team environment, both through technical skills and conversation. He was easy to talk to which added to the cohesiveness of the team and a positive team atmosphere. Kavi notified the group of any changes in schedule and willingness to help out in whatever capacity he is able to. Communication is crucial towards figuring out how to delegate tasks and complete the project, which motivated us to persist in the project because we knew it could be accomplished when people took responsibility in completing their part of the project. Finally, our team also knew when to take breaks and come back to an issue with fresh perspectives. This prevented burnout and frustration, and ultimately led to more productivity in overcoming issues.

Most of all, what kept us encouraged to persist through our failures was each other. Having the support of your mentors and your team members, the comfort of a team environment, makes any failure conquerable.

Code Sources

- Tamojit, Santos, S., Almeida, B., Validio, Jjmn, Tabor, & Johnmerry. (2022, April 8). *Arduino with TDS Sensor (Water Quality Sensor)*. Random Nerd Tutorials.
<https://randomnerdtutorials.com/arduino-tds-water-quality-sensor/>
- Maris, Dejan, Tofiq, Colonel, Rajesh, Noah, Nedelkovski, D., Kamlesh, Uday, Petkus, D., Wanix, Sanath, Quddusi, A. M., Khang, N., Anh, L. T., Schurke, Jordan, D., FerNava, Kalhara, S., ... Shashi. (2022, February 18). *Arduino color sensing tutorial - TCS230 TCS3200 color sensor*. How To Mechatronics.
<https://howtomechatronics.com/tutorials/arduino/arduino-color-sensing-tutorial-tcs230-tcs3200-color-sensor/>