Vision E600

Introduction to Mechanical Engineering

December 2nd, 2020

Overall report manager - Elena Moore

Design manager - Zachariah Bath

Construction manager - Megan Beyer

Preliminary testing manager - Zachariah Bath

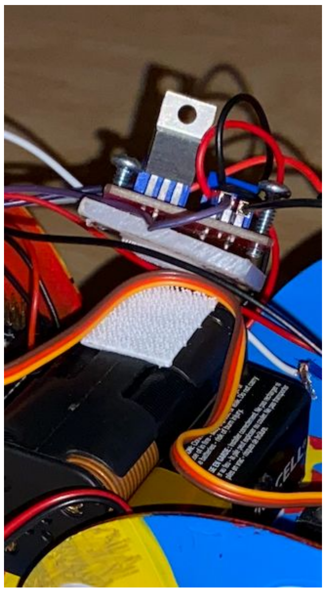
Final performance manager - Elena Moore

1. Introduction (Zachariah):

Students were given the task of producing trolley-like cars that could speed up, slow down, and stop based on how far something in front of it was. The goal of this project was to introduce the concept of self-driving cars to students. In order to do this, students were required to build a "trolley" constructed out of the given materials: a particle board chassis, four CD wheels, up to two 6V toy motors, a 6V battery pack, a MOSFET (metal–oxide–semiconductor field-effect transistor), and an ultrasonic sensor. Students were also allowed to design their own parts using SolidWorks and could also scavenge through a mess of materials during lab time. We were given roughly four-five weeks to fulfill the requirements of the project.

2. Design (Megan):

The main decisions we had to make in the design process were what parts needed to be 3D printed, what material we were going to use to attach everything together, how to mount all of our components to the top of the chassis, and what gear ratio our trolley needed to run on its own. Some of these decisions needed to be made very quickly, such as which materials to 3D print. We knew which parts needed to be printed, but we didn't know the exact measurements that early in the process. Designing the wheel hub and the axle mount (pictures left) as well as the motor mount seemed simple at first, but when the pieces arrived we realized some of them were too big. Luckily, this didn't prove to be a huge problem because we sanded a few parts down or added material to them so that they would fit more snug to the part they came in contact with. To attach all of our components together, we decided to use a combination of super glue, hot glue, and Command velcro strips. For the more delicate pieces, like the MOSFET, Arduino, and battery pack, we used command strips to mount them to the chassis (pictured right). For parts that needed to be really sturdy, like the bond between the CD and 3D printed wheel hub, we used hot glue and superglue. A lot of the hot glue and super glue was trial and error, and we made it up as we went to see which would perform better. Mounting everything onto the chassis was a bit more straightforward; we just needed to figure out which arrangement would make the most sense. Mounting the MOSFET on top of the batteries (pictured right) which were in the middle of the chassis next to the Arduino was a decision we made so that all the wires could meet in the middle rather than travel all around the trolley. We also



made the decision to increase our gear ratio much later in the process which set us back a day in time. With our initial gear ratio of around 1:2, our trolley had barely enough torque to start moving on its own and it had to be pushed for it to get going. We had the option to adjust the code to make the motors work harder to get going or to switch the gears out all together. Adjusting the code didn't work no matter how hard the motor was running, so we decided to increase our gear ratio. At least a 1:5 gear ratio was needed to get our trolley running, so we replaced the gear on the axle for a much larger one. Come competition day, that decision was crucial to the speed and success of our trolley.

3. Construction (Elena):

There were three main parts that had to be designed and printed for the construction of the trolley, a part to mount the axle to the bottom of the chassis, a part to mount the motor to the top of the chassis, and wheel hubs. The attachment of these three components needed to be slightly more complex than that of others and therefore required a more specifically designed part. The axle needed to be able to rotate freely underneath the car, yet still needed to be held tightly enough so as not to change position too drastically, thus requiring a mount with a casing of diameter just slightly larger than that of the axel. Our axle was significantly smaller than the inner diameter of our CD wheels, so a part needed to be made to decrease the inner diameter of the wheels so that they would fit snugly onto the axle. This part included two plates that fit onto the center of the wheel with small inner diameters specific to the axle. For the third part, while we were provided motor mounts, those provided were fairly bulky and didn't seem to hold the motor in the most effective way that would fit into the rest of our trolley's design. Therefore we printed a simple piece to fit tightly over the motor and be drilled directly into the chassis. By making sure that none of these parts were overly complicated, we were able to effectively utilize the 3D printers for all parts with no complications. One major advantage was that upon realizing that our axle would be too far from the chassis to reach the gear on the motor, we were able to sand down the printed axle mounts to get the axle closer to the motor. Another advantage to the printed plastic material was that by applying acetone to the wheel hubs, we could soften the plastic in order to fit the hub onto grooves we had drilled into the axle to achieve an even tighter fit. The only disadvantage we faced was that the axle casing on the mount was too thin and broke upon too much force. However this wasn't necessarily a fault of the material, but could have been fixed with a different dimension. Beside the 3D printed parts, we used velcro to stick the Arduino board, batteries, and MOSFET onto the chassis, which was sturdy yet made removal for adjustments and disassembly very easy.

4. Preliminary Testing (Elena):

The first time the trolley was tested, while it drove, it was evident that changes needed to be made. Primarily it became obvious that our gear ratio was insufficient. Previously it was practically 1:2, which made our trolley far too slow and difficult to even make the necessary measurements. We increased the ratio to 1:5 which made our car significantly faster, in fact it became so fast that upon second testing we had to alter our code to increase the stopping distance so as not to collide with the car or wall picked up by our trolley's sensor. In order to increase deceleration capabilities, we added a servo motor as a brake directly onto the back wheel. The code was altered so that the brake would activate at a closer distance from the trolley in front after our trolley had begun to decelerate. Another significant change we had to make dealt with fitting the wheels onto the axle. We had some difficulty getting the wheel to not slide off the axle or fall out of place. Our solution was to drill grooves into the axle and soften the plastic of the wheel hubs before sliding the wheel on so that when the hubs hardened, they would be fixed into the axle grooves and not move position. This worked considerably well and fixed both the wheels and axle sturdily into place. Besides those two major adjustments, the rest of the trolley worked well and all that had to be altered after that was the code to catch up with the new speed of the trolley.

5. Final Competition and Testing (Elena):

Our trolley performed admirably in competition with the other groups'. Its speed was similar, if not slightly faster than that of the other trolleys. We were able to achieve more torque to all four wheels with our high gear ratio and by using two motors that divided the torque between the front and back wheels. When following directly behind the other cars, we were able to make it through the course twice with only one small collision. Even upon cornering our trolley required little to no help. The only area that required some alteration before success was the 3-meter dash. Our trolley was barely able to stop in time before colliding with the paper bag after reaching full speed. However after adjusting the stopping distance in the code, our trolley was able to make a complete stop just a few centimeters before the bag. Overall our trolley was consistently reliable throughout the whole competition. The design was sturdy enough to even hold up during collisions and our "off-road" tests of jumping over the track, and the wiring and code constantly performed as expected.

6. Future Work (Megan):

If our group started over from scratch, we would have come up with a more solid design for the gears and the axle mount so that we could have used a larger gear ratio from the beginning and we wouldn't have had to sand down our axle mount so much because it was too thick. We would have also planned to have a wheel brake earlier in the design so that we wouldn't worry too much about our code or our motor being the reason it wouldn't stop at a short distance. The advice we would give future ME students is that you really do need to add a brake, whether it's a gear brake or a wheel brake The groups that didn't have a wheel brake really struggled to get it to stop with minimal crashes. We would also tell them to use materials already given to you instead of trying to 3D print everything. Our group avoided that issue by using the gears and axle that were given to us and only 3D printing the bare minimum. For next semester, a qualification should be added to make sure that the back of the trolley can be sensed by the one running behind it. Our trolley was difficult to sense from behind because it didn't have a flat piece of construction paper, so ours had to go last. We think another cool performance measure that would be fun is measure the trolley's top speed or race them on competition day.

Appendix:

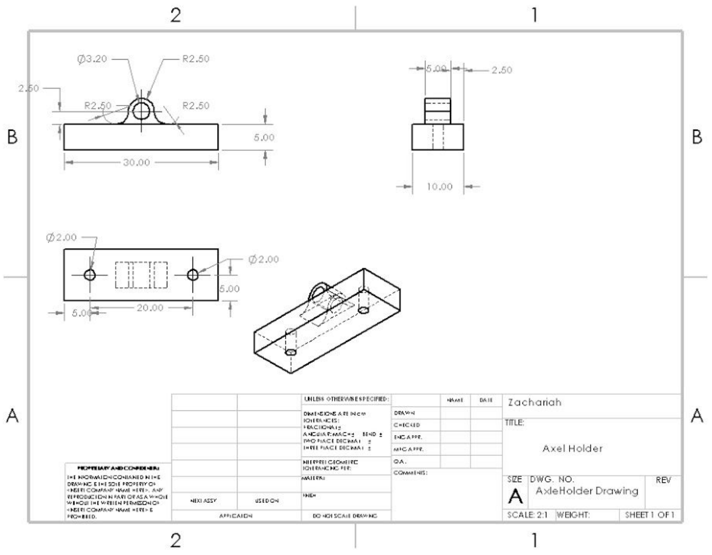
I. Code, Lead Designer - Zachariah:

```
#define triggerPin 6                          // Use pin 6 to send the chirp
#define echoPin 5                             // Use pin 5 to get the echo
#define StopPin 8                            // Use pin 8 for 'Stop' Lights
#include <Servo.h>                           //used for servo motor variable
#define motorPin 2                           // Minutes for the whole pumping cycle
long echoTime;                               // Time between chirp and echo
long dist;                                   //sets dist to long type
float PulsePeriod;                           // Period [ms] of the square wave to the motor, in milliseconds
float DutyCycle;                             // 0-100 [%], percent of the time the motor runs
float onTime;                                 // Time [s] that the motor is going to be on
Servo myservo;                               //creates variable name for our servo motor
void setup() {
        Serial.begin (9600);
        pinMode(motorPin, OUTPUT);            //pin for motor
        pinMode(triggerPin, OUTPUT);         //pin for trig on Ultrasonic Sensor
        pinMode(echoPin, INPUT);             //pin for echo
        pinMode(StopPin,OUTPUT);              //pin for 'Stop' lights
        myservo.attach(4);                   //attaches servo motor to output pin 4
        myservo.write(150);                  //sets motor to this degree
}
void loop() {
        digitalWrite(triggerPin, LOW);       // Clear the trigger output
        delayMicroseconds(2);                 // Gives some of the electronics time to catch up
        digitalWrite(triggerPin, HIGH);      // This starts the chirp
        delayMicroseconds(50);                // This is how long the chirp lasts
        digitalWrite(triggerPin, LOW);       // This ends the chirp
        echoTime = pulseIn(echoPin, HIGH);   // PulseIn is the timer function of the Arduino
        dist = (echoTime/2) / 29.1;          // Converts speed of sound into cm (but only the one way trip)
        Serial.print(dist);                   //prints dist in serial monitor
```

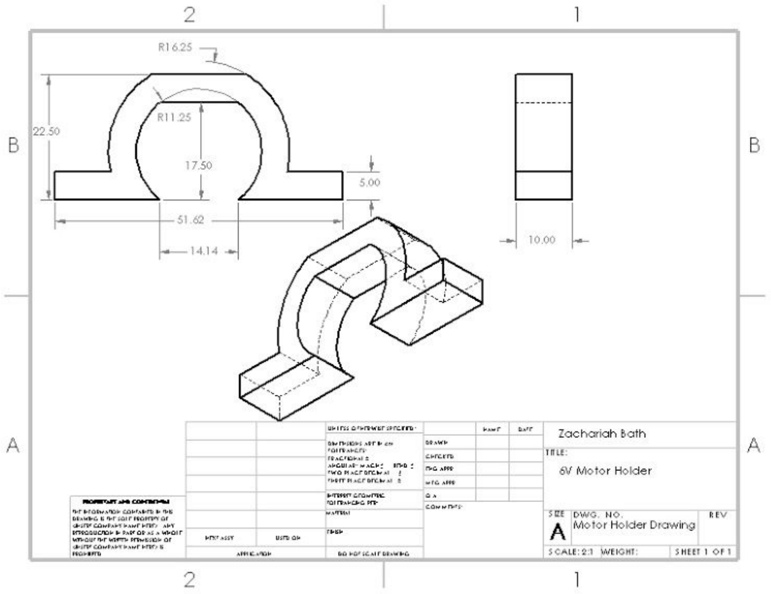
```
            Serial.println(" cm");                       //adds units to dist
if(dist <= 15){                                         //does this if dist less than 15
            digitalWrite(StopPin, HIGH);                 //turns on 'stop' light
            DutyCycle = 0;                              // motor runs at this %
            myservo.write(100);                         //sets motor to this degree
            PulsePeriod = 50;                           //Arduino works on milliseconds
            onTime = (DutyCycle/100)*PulsePeriod;        // Actual time the motor is actually running
            digitalWrite(motorPin, HIGH);               // Turn the motor on
            delay(onTime);                               //delays for amount of time before turning motor off
            digitalWrite(motorPin, LOW);                 // Turn the motor off
            delay(PulsePeriod - onTime);                 // motor stays off for the remainder of the period
}
if(dist < 20){                                          //does this if dist less than 20
            digitalWrite(StopPin, LOW);
            DutyCycle =0;                               // motor runs at this %
            myservo.write(100);                           //sets motor to this degree
            PulsePeriod = 50;                           //Arduino works on milliseconds
            onTime = (DutyCycle/100)*PulsePeriod;        // Actual time the motor is actually running
            digitalWrite(motorPin, HIGH);                // Turn the motor on
            delay(onTime);
            digitalWrite(motorPin, LOW);                 // Turn the motor off
            delay(PulsePeriod - onTime);                // motor stays off for the remainder of the period
 }
if(dist > 20 and dist <=30){                            //does this if dist greater than 20 and less than or equal to 30
            digitalWrite(StopPin, LOW);
            DutyCycle = (1)*dist;                        // motor runs at this %
            myservo.write(150);                          //sets motor to this degree
            PulsePeriod = 50;                           //Arduino works on milliseconds
            onTime = (DutyCycle/100)*PulsePeriod;        // Actual time the motor is actually running
            digitalWrite(motorPin, HIGH);               // Turn the motor on
            delay(onTime);
            digitalWrite(motorPin, LOW);                // Turn the motor off
            delay(PulsePeriod - onTime);                // motor stays off for the remainder of the period
}
if(dist > 30 and dist <= 40){                           //does this if dist greater than 30 and less than or equal to 40
            digitalWrite(StopPin, LOW);
            DutyCycle = (1.5)*dist;                     // motor runs at this %
            myservo.write(150);                         //sets motor to this degree
            PulsePeriod = 50;                            //Arduino works on milliseconds
            onTime = (DutyCycle/100)*PulsePeriod;        // Actual time the motor is actually running
            digitalWrite(motorPin, HIGH);                 // Turn the motor on
            delay(onTime);                               //delays for amount of time before turning motor off
            digitalWrite(motorPin, LOW);                 // Turn the motor off
            delay(PulsePeriod - onTime);                // motor stays off for the remainder of the period
}
 if(dist > 40 and dist <= 50){                          //does this if dist greater than 40 and less than or equal to 50
            digitalWrite(StopPin, LOW);
            DutyCycle = (2)*dist;                        // motor runs at this %
            myservo.write(150);                          //sets motor to this degree
            PulsePeriod = 50;                      //Arduino works on milliseconds
            onTime = (DutyCycle/100)*PulsePeriod;   // Actual time the motor is actually running
            digitalWrite(motorPin, HIGH);           // Turn the motor on
            delay(onTime);                         //delays for amount of time before turning motor off
            digitalWrite(motorPin, LOW);            // Turn the motor off
            delay(PulsePeriod - onTime);            // motor stays off for the remainder of the period
 }
 if(dist > 50 and dist <= 2000){                   //does this if dist greater than 50 and less than or equal to 2000
            digitalWrite(StopPin, LOW);
            DutyCycle = 100;                        // motor runs at this %
            myservo.write(150);                     //sets motor to this degree
            PulsePeriod = 50;                      //Arduino works on milliseconds
            onTime = (DutyCycle/100)*PulsePeriod;   // Actual time the motor is actually running
            digitalWrite(motorPin, HIGH);          // Turn the motor on
            delay(onTime);                         //delays for amount of time before turning motor off
            digitalWrite(motorPin, LOW);            // Turn the motor off
            delay(PulsePeriod - onTime);            // motor stays off for the remainder of the period
 }
 if(dist > 50 and dist <= 2000){                   //does this if dist greater than 50
            digitalWrite(StopPin, LOW);
            DutyCycle = 0;                         // motor runs at this %
            myservo.write(150);                    //sets motor to this degree
            PulsePeriod = 50;                      //Arduino works on milliseconds
            onTime = (DutyCycle/100)*PulsePeriod;   // Actual time the motor is actually running
            digitalWrite(motorPin, HIGH);           // Turn the motor on
            delay(onTime);                         //delays for amount of time before turning motor off
            digitalWrite(motorPin, LOW);            // Turn the motor off
            delay(PulsePeriod - onTime);            // motor stays off for the remainder of the period
 }
}
```

II. SolidWorks Drawings, Lead Part Designer - Zachariah:

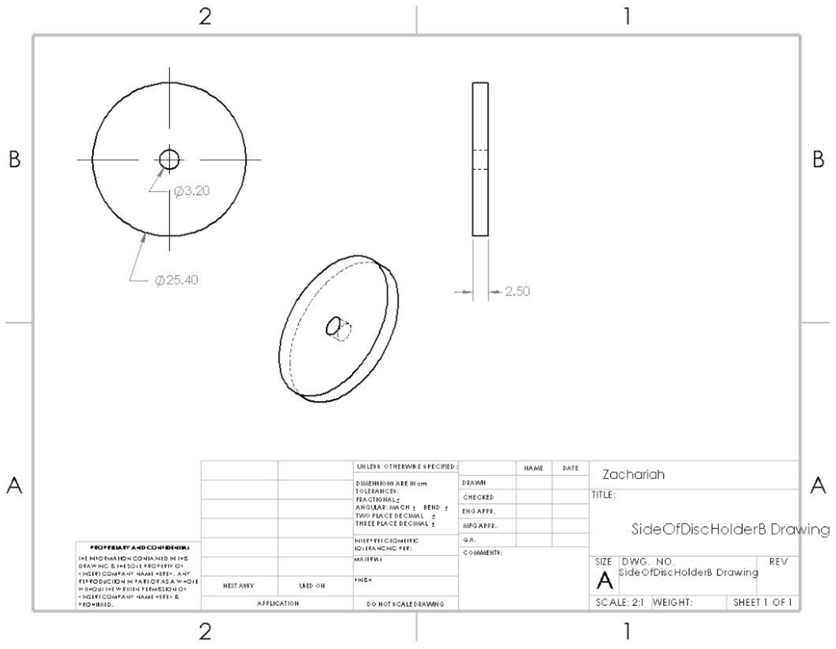
A. Axle Holder (printed 4, 2 for each axle):



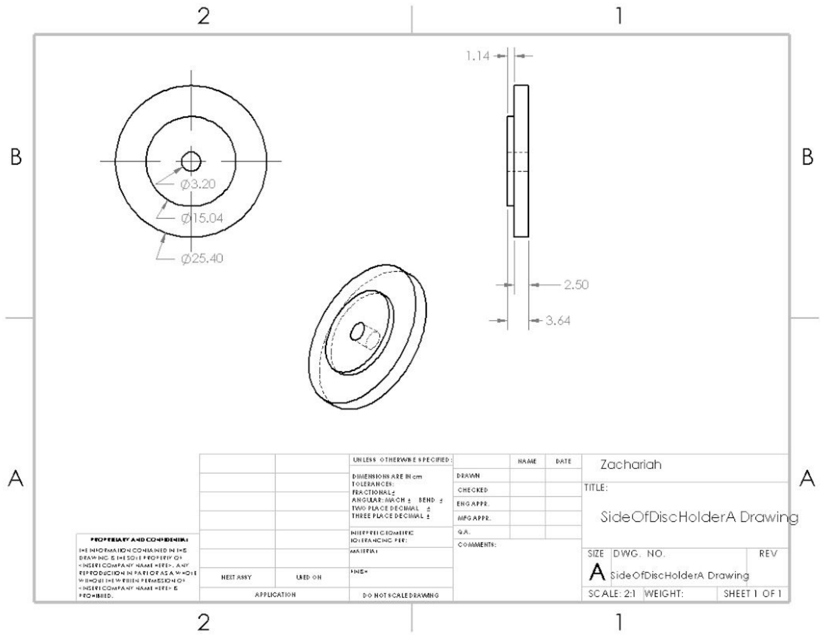
B. 6V Motor Holder (printed 2, one for each motor):



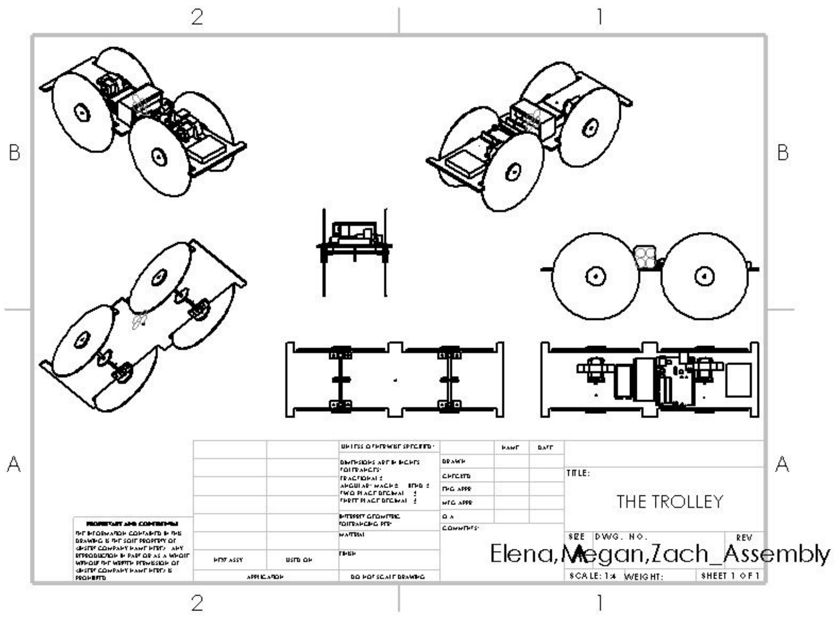
C. Wheel Hub Part A (printed 4, 1 for each wheel):



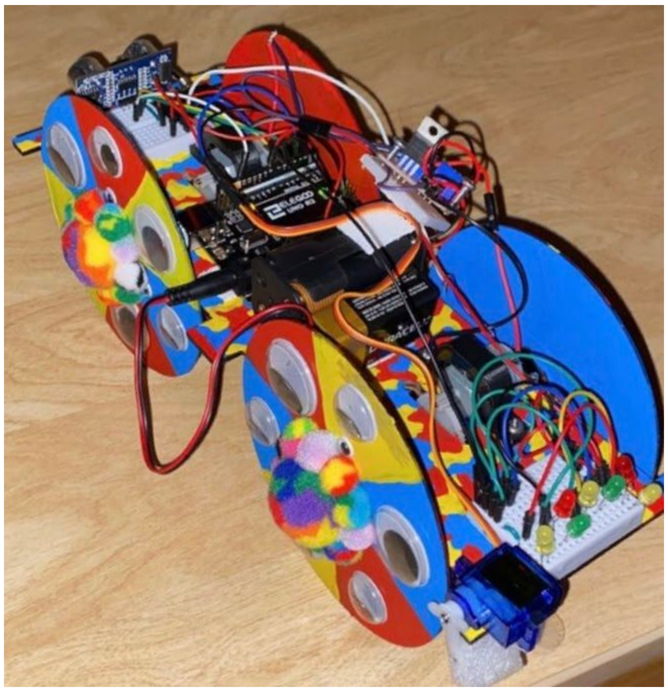
D. Wheel Hub Part B (printed 4, 1 for each wheel):



III. SolidWorks Assembly Drawing, Lead SolidWorks Assembler - Zachariah:



one caveat is we do not have the servo motor, as it was a last minute design (its shown in the following picture)



*the servo motor is about here ^^^^^^^^*