**Team 17 Robot Report**

**System Description**

**Design Goals:**

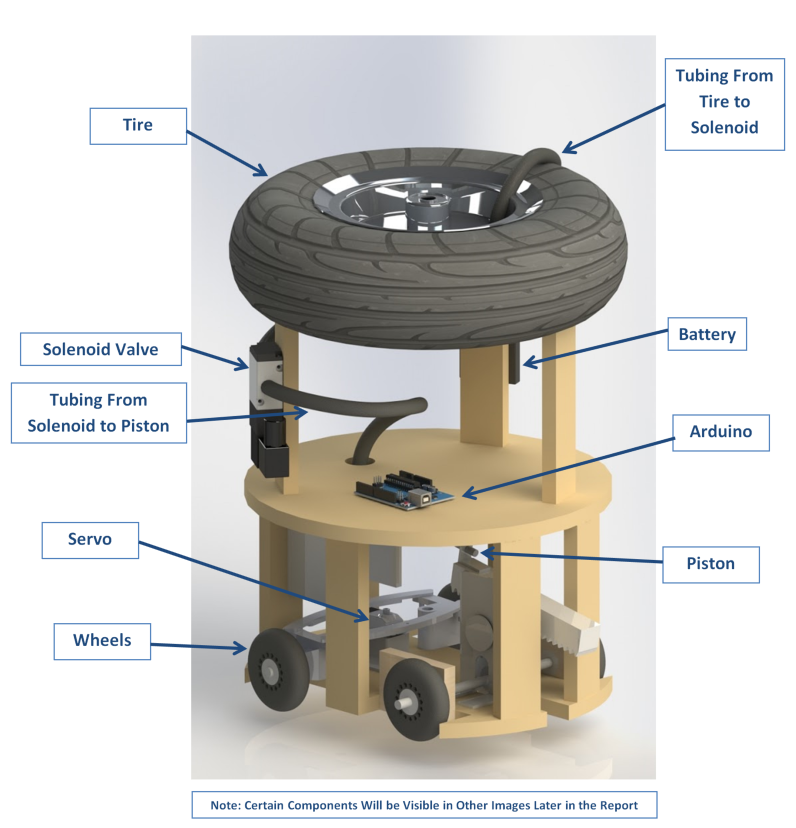
1. Mechanical
   1. Create a strong and stable frame for our robot.
   2. Use two forms of racks and gears to power and steer the robot.
2. Software
   1. Design so that the minimum amount of movements is necessary to reach each point
   2. Incorporate closed-loop feedback control mechanism by achieving and maintaining desired output condition.

**How our design was set apart from other robots:**

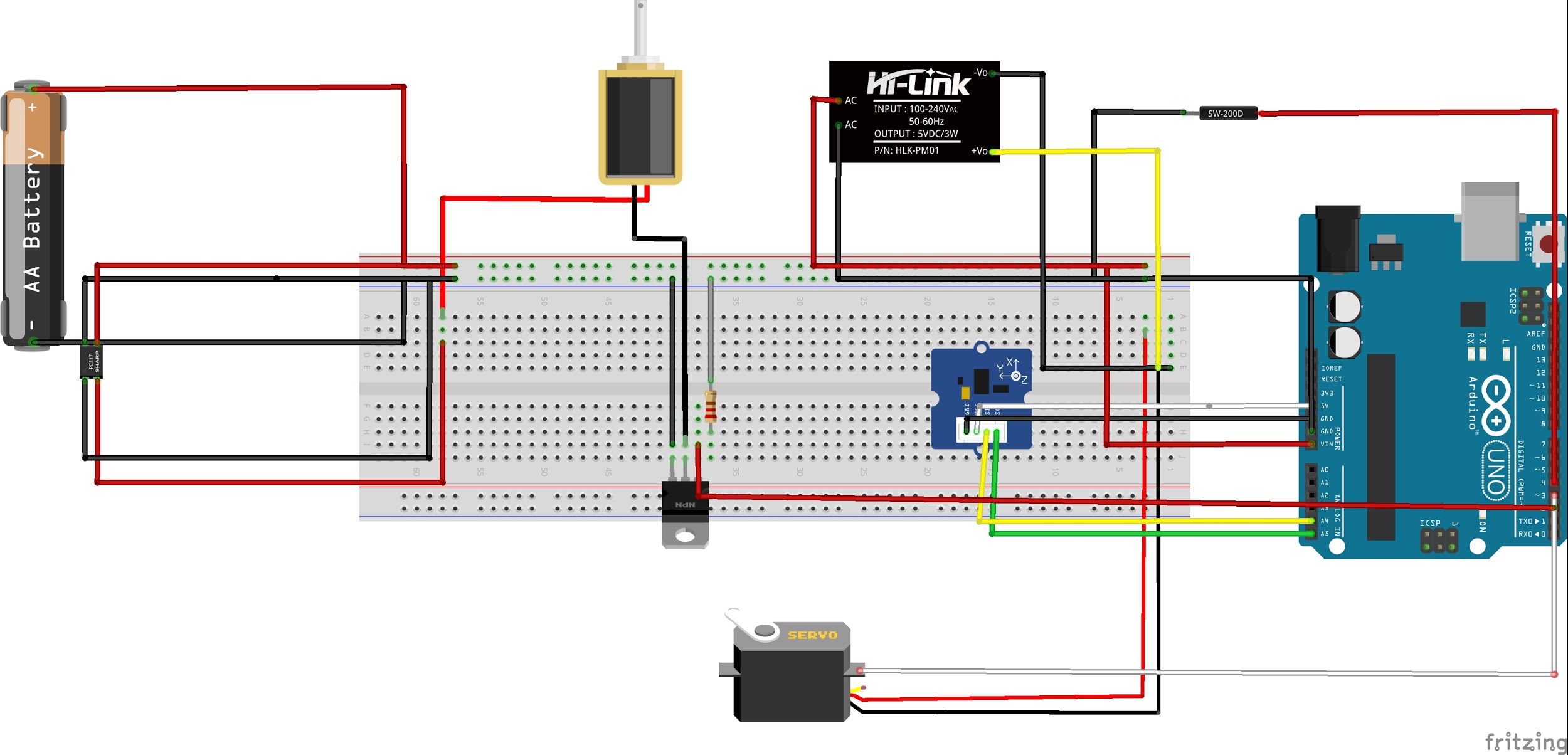
1. Mechanical
   1. Our robot design featured three levels. The first level held our steering and drive, the second level featured our electronics equipment, and the third level held our tire. The second level was supported by six vertical beams and the third level was supported by three vertical beams. These beams made the design very stable and meant that we did not need to worry about the robot having structural issues. However, this design did have its flaws. The strong structure was heavy and weighed down on our steering assembly, making it hard for the robot to turn.
   2. For our steering and drive systems, we wanted to use gears. Rack and pinion systems were a great choice for converting power from the motors directly to the wheels and their axles. However, these systems ended up becoming more complicated over time. Originally, the piston fired over the back wheels on a single gear with a one way bearing inside (Shown in “**Drive System Designs and Exploded View**”) . This meant that the power for the robot had to come from the piston retracting rather than firing and we quickly learned while testing that this would not be enough to move the robot. Once again, the weight of the robot became an issue. The solution was to add another gear above the axle mounted gear that the piston would fire on. This system worked far better than our original system. For our steering system, we originally wanted to use a similar rack and pinion system, but decided to connect the servo directly to the steering arms. The Steering system is explained in detail in “**Steering Mechanism Evolution and Exploded View.**”
2. Software
   1. Our software combined and made adjustments to different parts of the code that we utilized during our labs. We applied what we learned for closed feedback control in our labs to actuate and control the robot. Using the magnetometer, we compared the robot’s direction with the servo angle in order to steer the robot to the desired location. We chose the closed-loop feedback control system to eliminate any unnecessary additional steps that may be required to operate the robot. Instead, the control algorithm simply considers different cases depending on the parameters and processes through a simple proportional control for angle correction.

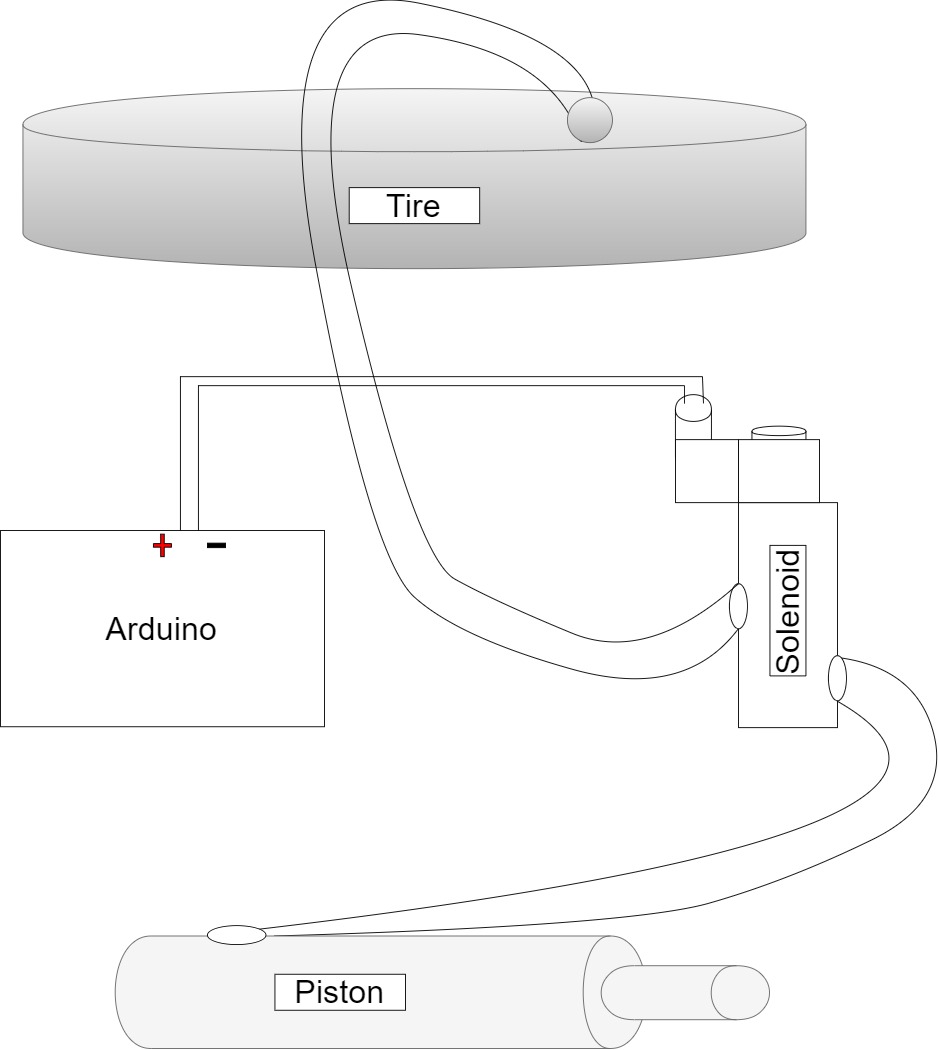
**Photograph** **and CAD model of our final robot design**

****

****

**Electrical** **and pneumatic circuit diagrams of your final robot design**

****

****

**Documented software code of your final robot design**

#include <Servo.h>

#include <Wire.h>

#include <LSM303.h>

LSM303 compass;

Servo myservo;

int servoPin = 3; // servomotor pin

int solenoidPin = 2; // mosfet pin

int switchPin = 4; // switch pim

int pos = 0; // variable to store the servo position

int switchState; // variable that stores the Reed switch state

int servoDir = 0; // variable that stores the direction the motor is turning in the demo program

int solenoidState = LOW; // variable that stores if solenoid is on or off

unsigned long previousMillis = 0; // store last time solenoid was updated

const long interval = 1000; // interval at which to turn solenoid on and off (milliseconds)

int headingdifference = 0;

int rolls = 0;

int lastState = 0;

int stage = 0;

int startingcase = 1;

int i = 0;

int t = 0;

int timeelaped = 0;

void setup() {

myservo.attach(servoPin); // attaches the servo on pin 3 to the servo object

pinMode(solenoidPin, OUTPUT); //Sets the pin as an output

pinMode(switchPin, INPUT\_PULLUP); //Sets the pin as an input\_pullup

Serial.begin(9600); // starts serial communication @ 9600 bps

Wire.begin();

compass.init();

compass.enableDefault();

compass.writeReg(0x24, 0x74); // updates at a higher rate on the magentometer

/\*

Calibration values; the default values of +/-32767 for each axis

lead to an assumed magnetometer bias of 0. Use the Calibrate example

program to determine appropriate values for your particular unit.

\*/

compass.m\_min = (LSM303::vector<int16\_t>){-1200, -1273, +1359};

compass.m\_max = (LSM303::vector<int16\_t>){+1204, +718, +1829};

}

void loop() {

if (t == 0)

{

delay(15000);

t = 1;

timeelaped = millis();

}

////////////// SOLENOID VALVE ///////////////////////////////////////////////////

unsigned long currentMillis = millis(); //piston on

if (currentMillis - previousMillis >= interval) {

previousMillis = currentMillis;

if (solenoidState == LOW) {

solenoidState = HIGH;

} else {

solenoidState = LOW;

}

digitalWrite(solenoidPin, solenoidState); //Switch Solenoid ON/oFF

}

////////////// REED SWITCH ///////////////////////////////////////////////////

switchState = digitalRead(switchPin);

if (switchState == 1 && lastState != 1) //read switch

{

rolls = rolls + 1;

}

lastState = switchState;

Serial.println("Num of rolls is: " + String(rolls));

Serial.println("Distance traveled: " + String(rolls\*3.14\*3.937) + "inches" );

if (rolls >= 10)

{

Serial.println("Start of Stage 2");

if (startingcase == 1)

{

myservo.write(130);

delay(5000);

}

else{

myservo.write(50);

delay(5000);

}

i=0;

while (i < 10)

{

if (currentMillis - previousMillis >= interval)

{

previousMillis = currentMillis;

if (solenoidState == LOW) {

solenoidState = HIGH;

} else {

solenoidState = LOW;

}

digitalWrite(solenoidPin, solenoidState); //Switch Solenoid ON/oFF

}

i = i + 1;

}

stage = 2;

Serial.println("Stage 3");

while (millis >= 70\*1000)

{

}

}

while ((rolls >= 10) && (stage == 2)) //STAGE 3 CLSOED FEEDBACK LOOP

{

compass.read();

float heading = compass.heading(); // reads compass

headingdifference = 246 - heading; // gets angle difference of direction with 246 as north

Serial.println(heading);

if(headingdifference <= pos) { //change servo position

pos--;

Serial.println("turn left");

}

else

{

pos++;

Serial.println("turn right");

}

myservo.write(pos); // tell servo to go to position in variable 'pos'

Serial.println(heading);

Serial.println((headingdifference)); // print to serial of diff

delay(10);

if (currentMillis - previousMillis >= interval)

{

previousMillis = currentMillis;

if (solenoidState == LOW) {

solenoidState = HIGH;

} else {

solenoidState = LOW;

}

digitalWrite(solenoidPin, solenoidState); //Switch Solenoid ON/oFF

}

Serial.print(" ");

Serial.println(heading);

while (millis >= 70\*1000)

{

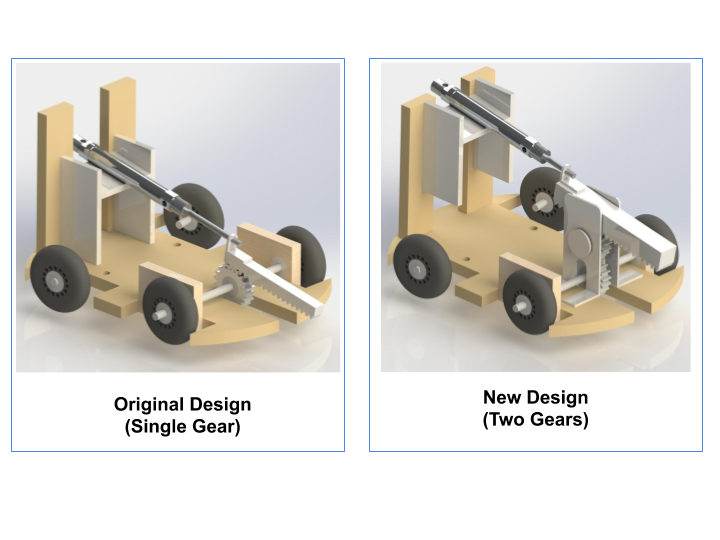
}

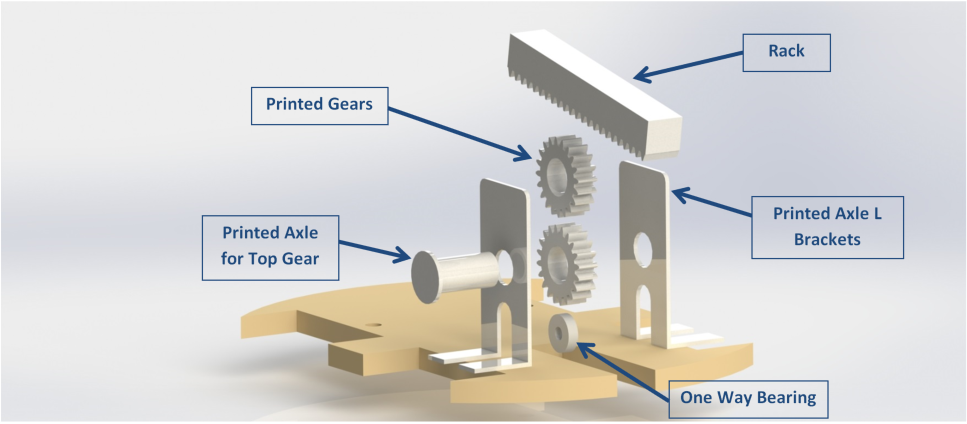
delay(10);

}

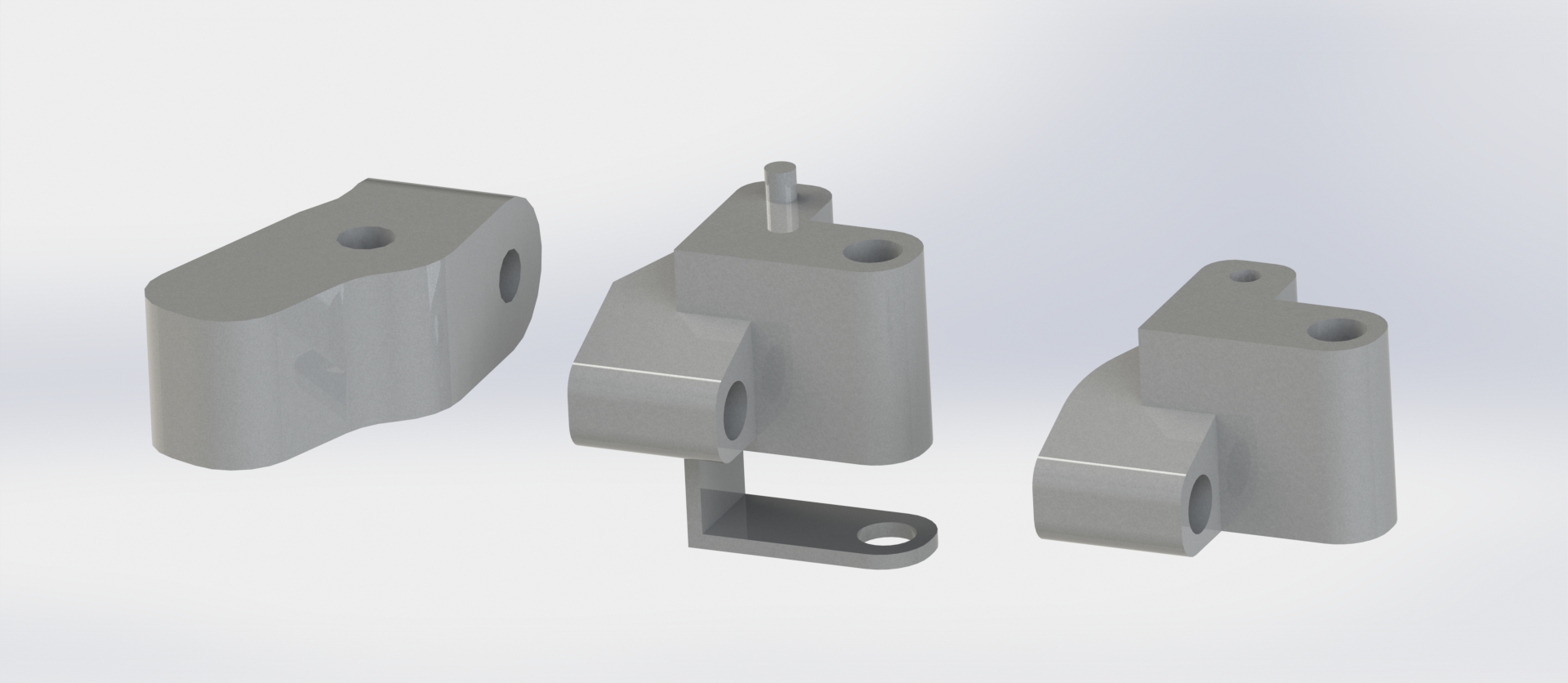
}

**Drive System Designs and Exploded View**

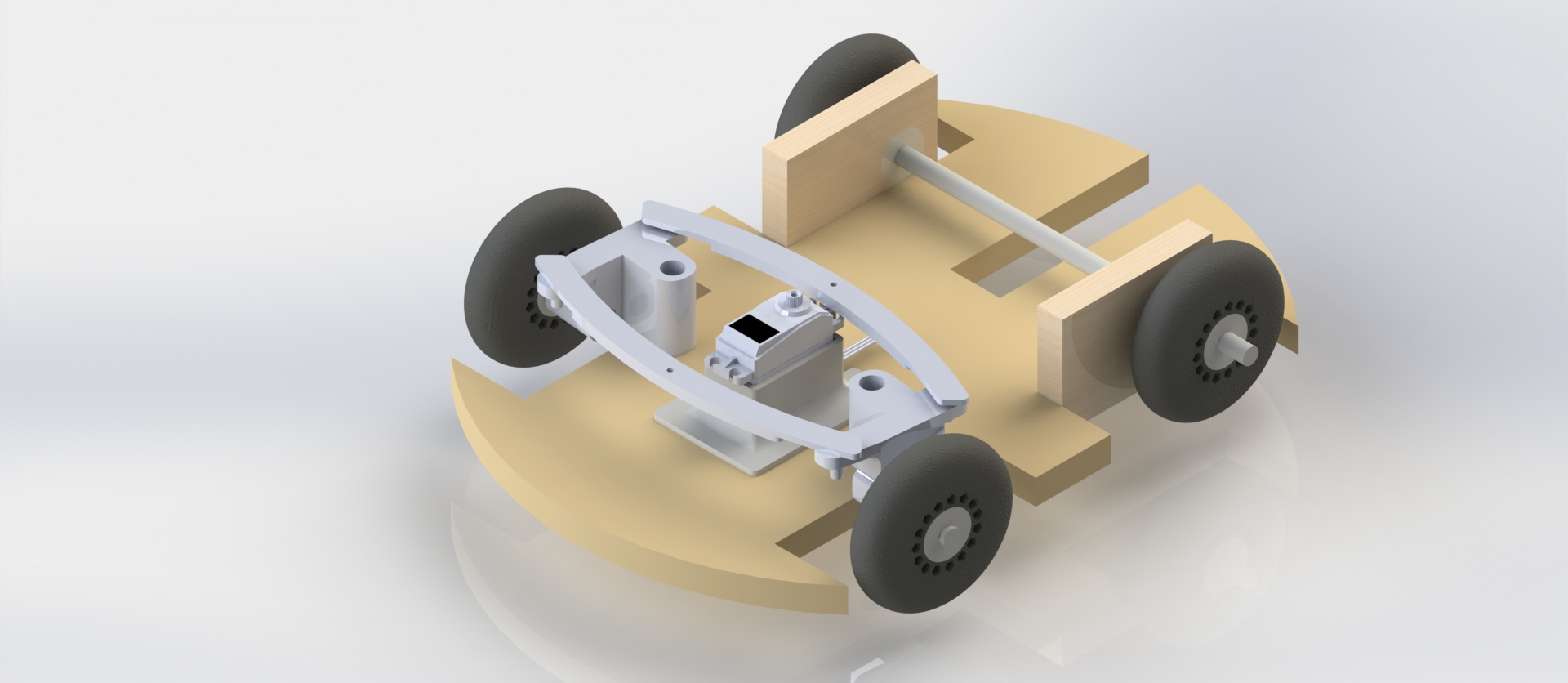


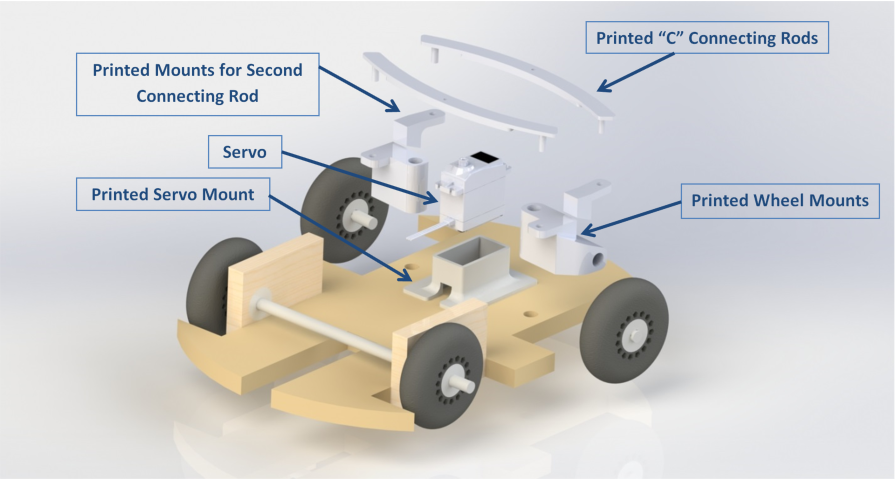


**Steering Mechanism Evolution and Exploded View:**



I put together a detailed history of the steering mechanism and how it evolved into the parts that we ended up using in our final design. Coming up with a working model was difficult at first, but with time and some test prints our team was able to find a solution that worked. The first problem that we had was where to put the axle for the wheels. Originally, we wanted the axle to be in line with the vertical connection (or king pin as it may be referred to), but the bolts going through the axle were long and that meant the wheels had to be very far from the robot. My future designs (pictured above) all have the axle mounted to the side of the vertical connection.





The next design challenge was designing a connection for both wheels that would allow the servo to turn it but also keep the wheels pointed in the correct direction. The original design featured one small connecting rod (C shaped piece running from wheel to wheel). This design worked in theory and when it was assembled with just the base plate. However, the weight of the robot once again created issues with this design. The steering was simply too heavy for a small arm. The final design we have on the robot features a second arm in the front and a thicker arm in the back. The added thickness keeps the arm from bending and the front arm keeps the wheels from turning in towards each other.

**Summary of Contributions**

**Team Members:**

1. Bryan (Electrical)
   1. Ensured proper wiring and soldering of components and troubleshooted problematic components.
   2. Mounted electrical components of the robot to the final design and ensured proper clearance and no interference between the components.
   3. Care and storage of the robot and its hardware. Ensured proper transportation to each team member.
   4. Helped ensure that each electrical component outputted correct data in the Serial Monitor and helped the Control engineer troubleshoot in Arduino
2. Michael (Mechanical)
   1. Helped get extra wiring and solder materials and tools to help electrical wiring when included parts were insufficient.
   2. Cut frame pieces using the machines in engineering tower
   3. 3D printed gears and other components necessary for fabricating the mechanisms of the robot.
   4. Transported the robot to teammates and testing locations when necessary.
3. Edward (Control)
   1. Assisted in providing alternatives for technical difficulties with mechanical design
   2. Obtained missing parts for the robot from multiple sources
   3. Responsible for optimizing and debugging the code after running through trials
   4. Took part in transporting components of the robot to multiple locations for assembling and testing

**Strategies:**

One strategy that would ensure productivity and good teamwork would be to acknowledge the schedules of each team member and being able to cover their roles effectively when schedules do not align to proceed with the development of a project.

Another strategy is to understand that although roles are assigned, each member can and should be flexible with their abilities to contribute to the final project outside of their own roles.