Team 513 Operation Manual

Jared Carboy, Kaden Lane, Carlos Sanchez, Axcell Vargas

FAMU-FSU College of Engineering

Project Overview

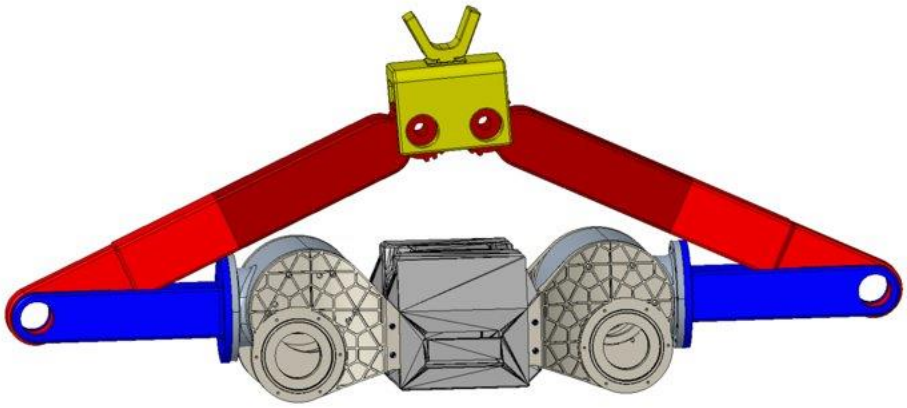


Figure 1: CAD Design

Project Description

The RE-RASSOR robot is the scaled down 3D printed version of the NASA RASSOR robot intended for research and educational purposes. The cost to transport payloads to the moon is astronomical, and the Florida Space Grant Consortium is looking to find new and innovative methods to lower the costs of moving robots to the lunar surface. By reutilizing the base of the RASSOR robot, the robot can be used for a wide range of tasks, including transportation. Team 513 has been tasked with modifying the robot base as well as any other components to design a collaborative transport system capable of lifting and moving 100 earth pounds.

Project Objective

Repurpose the RE-RASSOR mining robot into a transport system that can lift and move heavy payloads on the moon.

Component/Module description

Modules

The following modules are all major components of our design that all utilized in the transportation process. The electrical module includes wiring information, a connection diagram, and all electrical components being used. This module is responsible for the control aspect of the design. The lifting module is responsible for lifting the payload. This module consists of the 5-bar linkage and the shoulder joint. The final module is the connection module. This module is responsible for the connection to the sample payload and consists of a V-shaped hook and a gear connection.

Electric Module

The electric module is responsible for the control aspect of the transporter. Its parts include stepper motors, stepper motor drivers, an IMU, an Arduino Mega 2560, a 12V power supply, and wiring. A wiring diagram containing these components can be seen below.

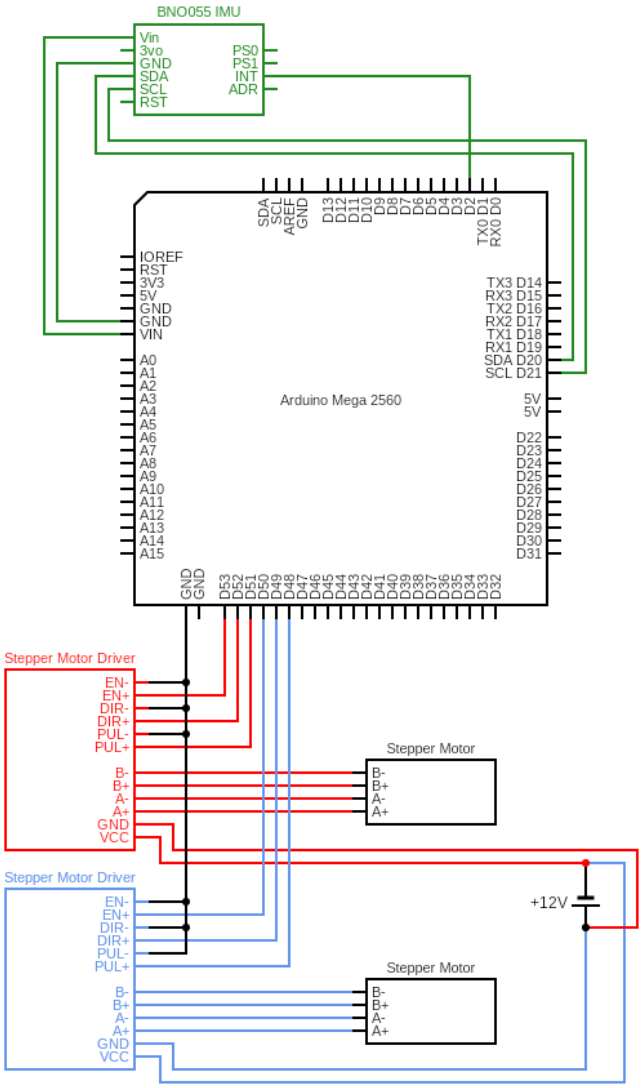


Figure 2: Wiring diagram for electrical components

This diagram displays how all electrical components in the design interact. The +12V represents the power supply which is plugged directly into an outlet. All these electrical components enable for the vertical lift and horizontal movement of our design.

Lifting Module

This module not only controls vertical lift but is also responsible for the stability of the robot as it controls motion in both the vertical and horizontal axis. The shoulder uses a gearbox to generate optimal torque from the stepper motor in effort to rotate the shoulder about the robot body. The 5-bar linkage design attaches to the shoulder to drive the end effector upwards to

generate a vertical lift. The bottom linkages are attached directly to the shoulder via a flange joint. The top linkages are divided into two components joined via a press fit in effort to fit the linkage on a 3D printer bed. The top linkages are connected to the bottom linkages with a pin that slides through the bearings at the connection points. To achieve the vertical lift, the shoulder generates torque from the stepper motor, driving the bottom linkages upward, ultimately forcing the end-effector at the end of the top linkages to rise.

Connection Module

This module controls the connection from the robot to the payload. It consists of a 1:1 gear connection at the end effector with an attached V-shaped hook. The V-shaped hook is connected via pins and bearings that slide through the middle of the gear connection. The 1:1 gear ratio ensures that the hook is constantly in line with the center of the robot body regardless of the orientation of the 5-bar linkage. With the hook and gear connection the robot can successfully attach to any payload within the appropriate peg connection.

CAD Design (Lifting & Connection Modules)

This section includes the CAD models of the overall design, including the lifting and connection modules.

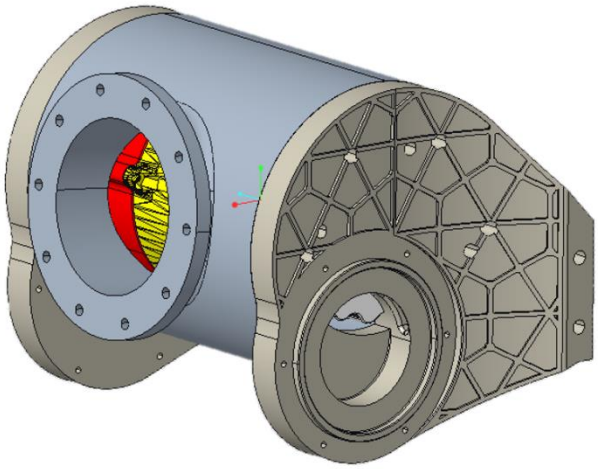


Figure 3: Assembled shoulder joint

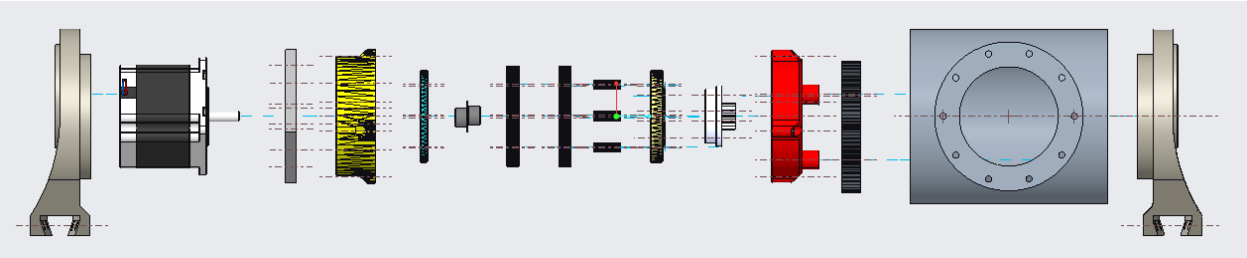


Figure 4: Exploded view of the shoulder joint

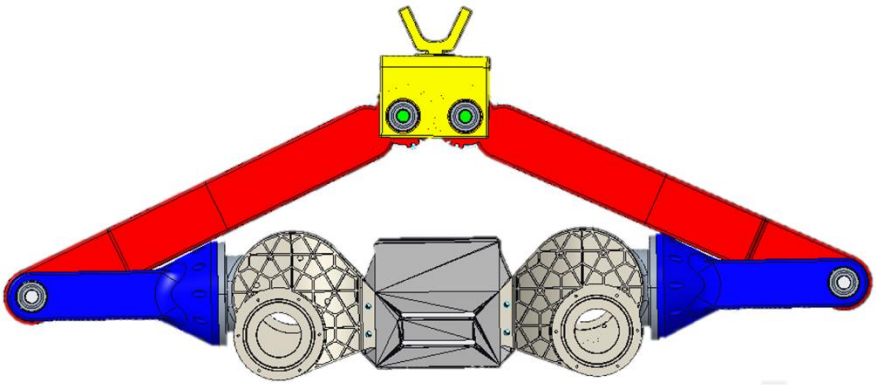


Figure 5: Assembled front view

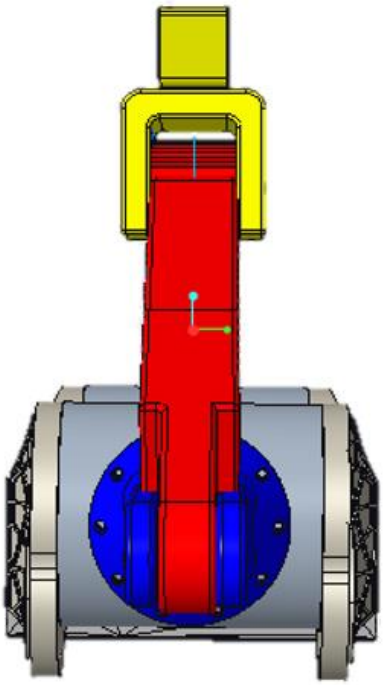


Figure 6: Assembled side view

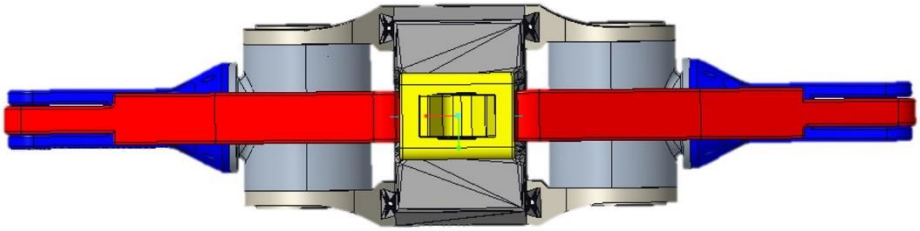


Figure 7: Assembled top view

Integration

Shoulder Assembly

The shoulder joints are reutilized from the Senior Design group at Florida Polytechnic. The complete assembly of the shoulder can be seen in Figure 4. Although the shoulder for our design was modeled using the Florida Polytechnic design group, the shoulder from a group here at FAMU-FSU may be used given the same flange joint is used. A different shoulder, however, may operate differently. Assembly instructions as well as the operation manual for the shoulder joint from the Florida Polytechnic team can be found linked below.

https://www.tinkercad.com/things/llDLYpZLnE6?sharecode=ltq_opnFMI9BRfd7aKXxHR_n65xQ-vriE18I-1rYTIc

Transport Assembly

The assembly for the transport system is relatively easy and can be seed in the CAD assembly below. The process will begin with the attachment of the bottom linkages (Item 1) to the flange joint seen in figure 3 using the M3 bolts and nuts. Once both bottom linkages are completely attached to the shoulder, the bearings (Item 5) can then be attached to the joint at the end of the bottom linkage with a simple press fit. If unable to press fit, it is recommended to align the center axis of the bearing to the center axis of the joint and use a mallet until the bearing is within the edges of the link. The same process can be used to attach the bearings to the top linkages (Item 2), the gear linkage (Item 3), and the hook (Item 4) connection points. Once the bearings are inserted into the top linkages, align the center axis of the bearings from the bottom linkages with those from the top linkages with the top linkage in the center. When aligned, slide the smaller pin (Item 6) through the center of the bearings until through the opposite end. Now, the gear linkage can be easily pressed onto the peg extruding from the top linkage. The final step is to align the bearings from the hook connection points with those in the gear linkages. When aligned, slide the longer pin (Item 7) through the bearings until through the opposite end. Now, the transport system should be entirely assembled onto the shoulder joints and ready for operation.

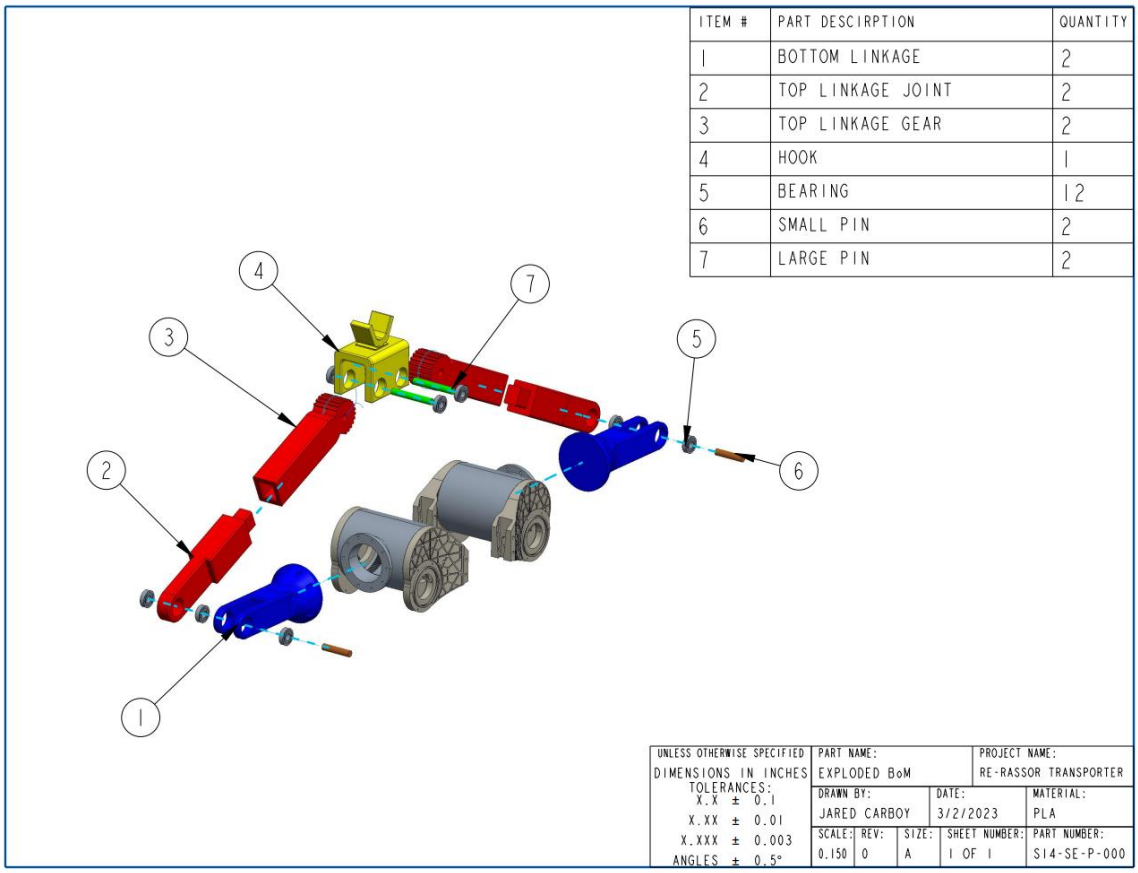


Figure 8: Bill of Materials

Operation

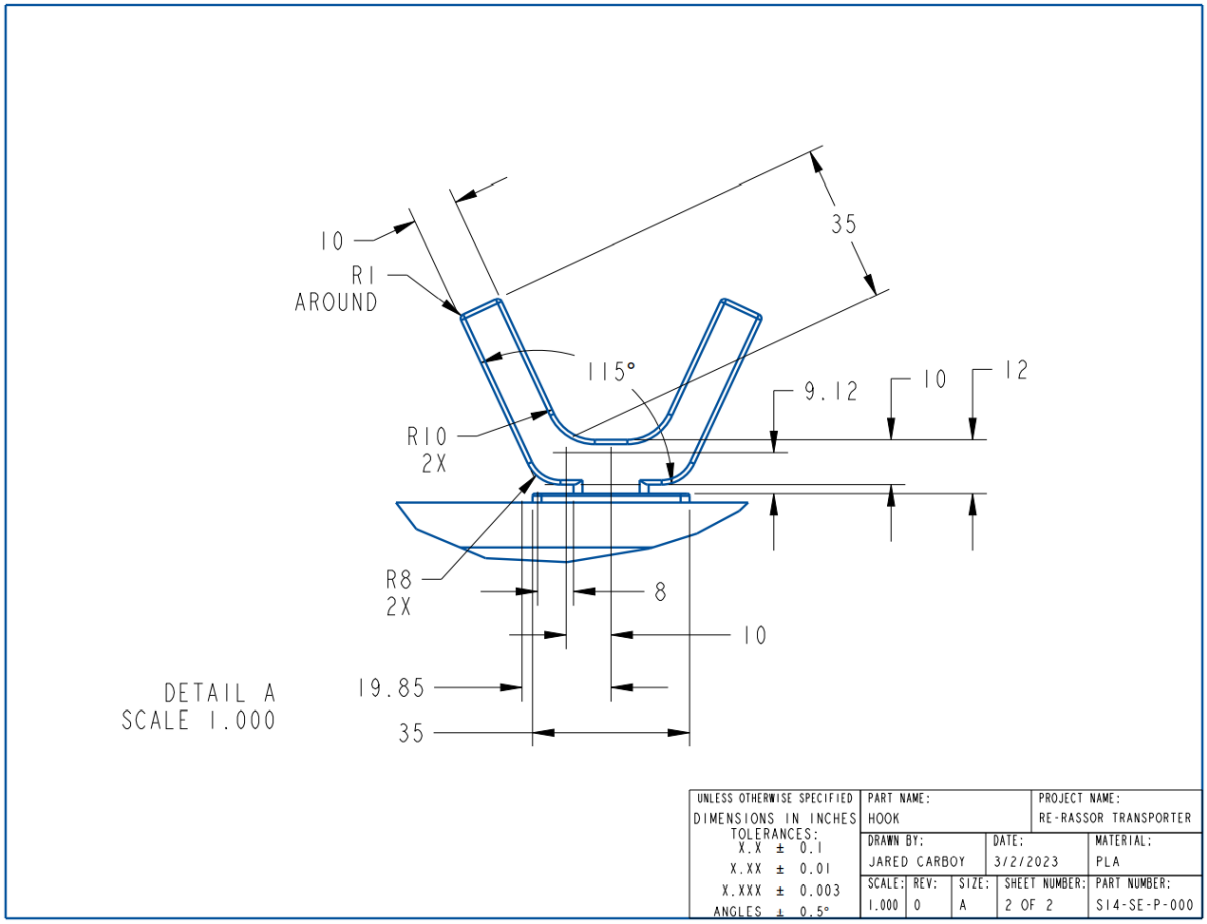
The target metrics for the sample payloads required to be transported on the moon is a 2x2x2m cube and a 5m long cylinder with a 0.5m diameter, both with an even weight distribution weighing 100 earth lbs. With four total connection points, each robot must be capable of lifting a minimum of 25 lbs. The designed connection pegs on the payload are placed 1.5m from each other and 0.25m above the ground. To test the functionality of the design, the two robots are placed 1.5 meters apart with a starting height of 0.25m measuring from the ground to the bottom of the hook. The weight is tested with a standard weightlifting barbell with free weights attached on either end to reach the desired weight of 50 lbs. The pegs are modeled to the same metrics as the end of a barbell. Once aligned and the weight applied, the user can run the Arduino code to inform the robots to lift the payload 150mm upwards in the z-direction.

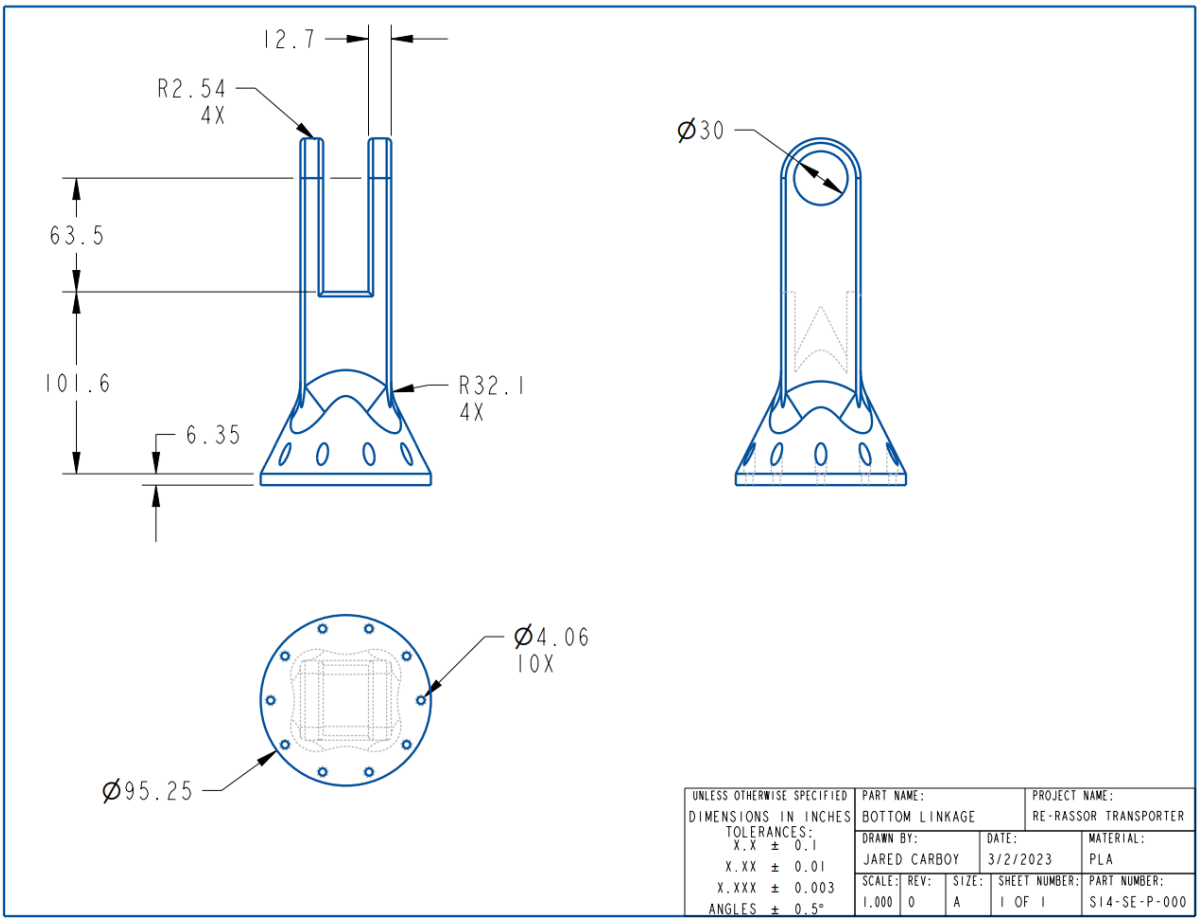
Once the robots prove capable of lifting the desired weight the total height, the robots are then tested on an incline of 5 degrees. Both robots are placed the same distance away on top of the designed slope. Once the weight is placed on the hooks of the robots, the orientation sensor automatically sends information to the controller to adjust the orientation of the shoulders to adjust the payload.

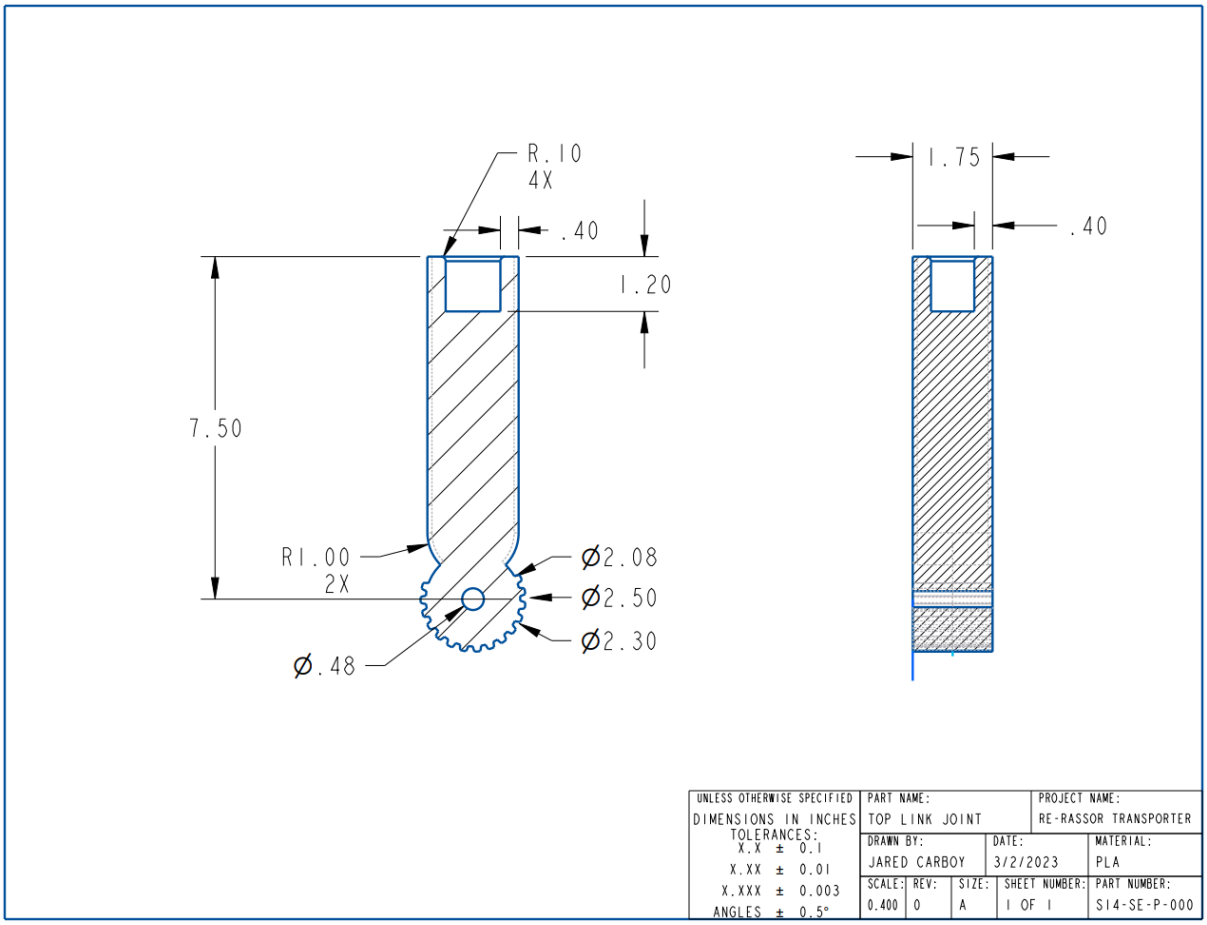
Troubleshooting

If there are any issues that arise when attempting to operate the design, there are few aspects that should be double checked that may be causing an issue. Primarily, as the wiring is quite complicated, it is important that all connections to the Arduino, the stepper motor, stepper motor drivers, and the power supply. It is possible they were arranged incorrectly, or wires came loose during operation. Another issue that could arise involves any debris within the gearbox in the shoulder hindering its ability to provide enough torque. An issue that arises when 3D printing is that material can come loose and fall within the gear connections, prohibiting proper rotation. Finally, issues may arise with the automation of the design. If there are any complications with the stability of the robot on inclines, check to ensure the code is running properly. If it is not, redownload the provided code and try again.

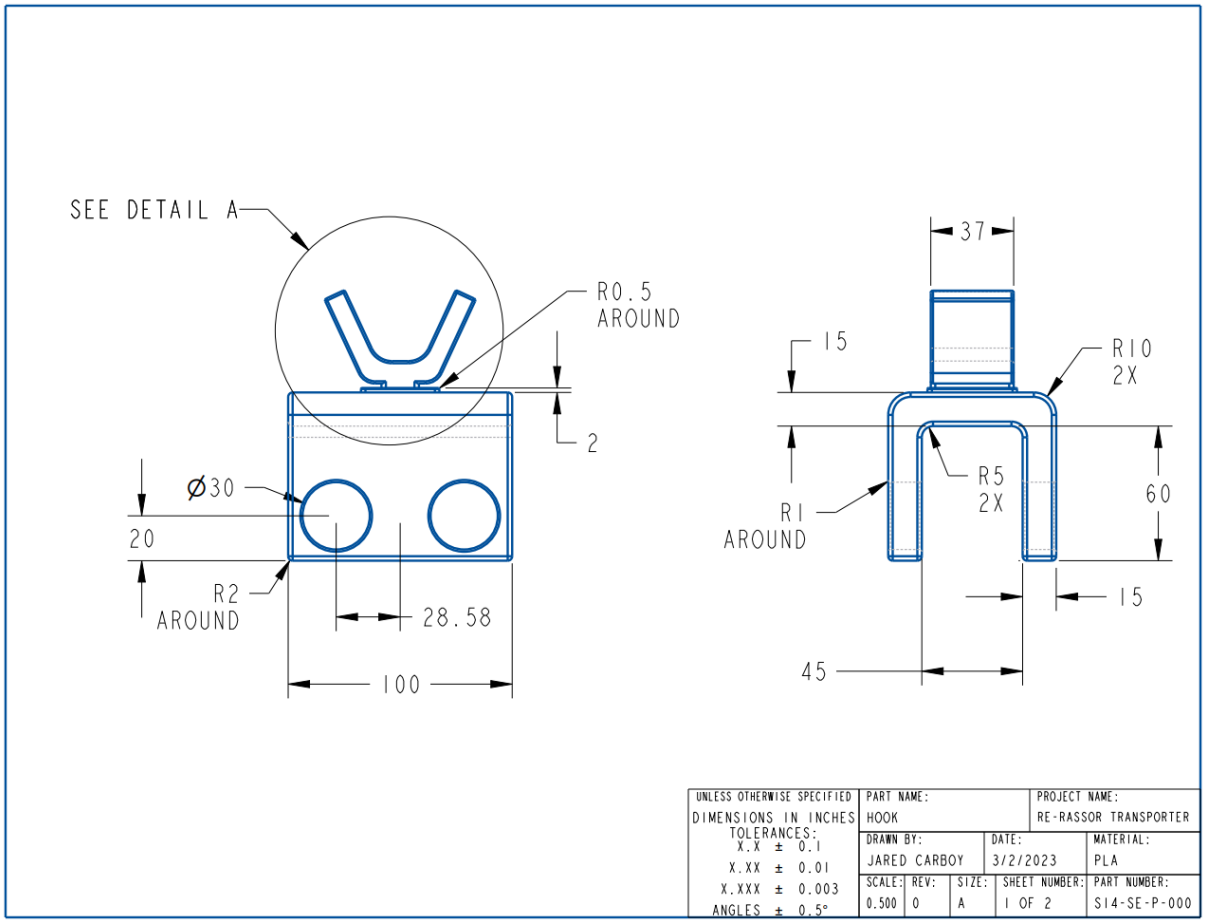
Appendix A – CAD Drawings

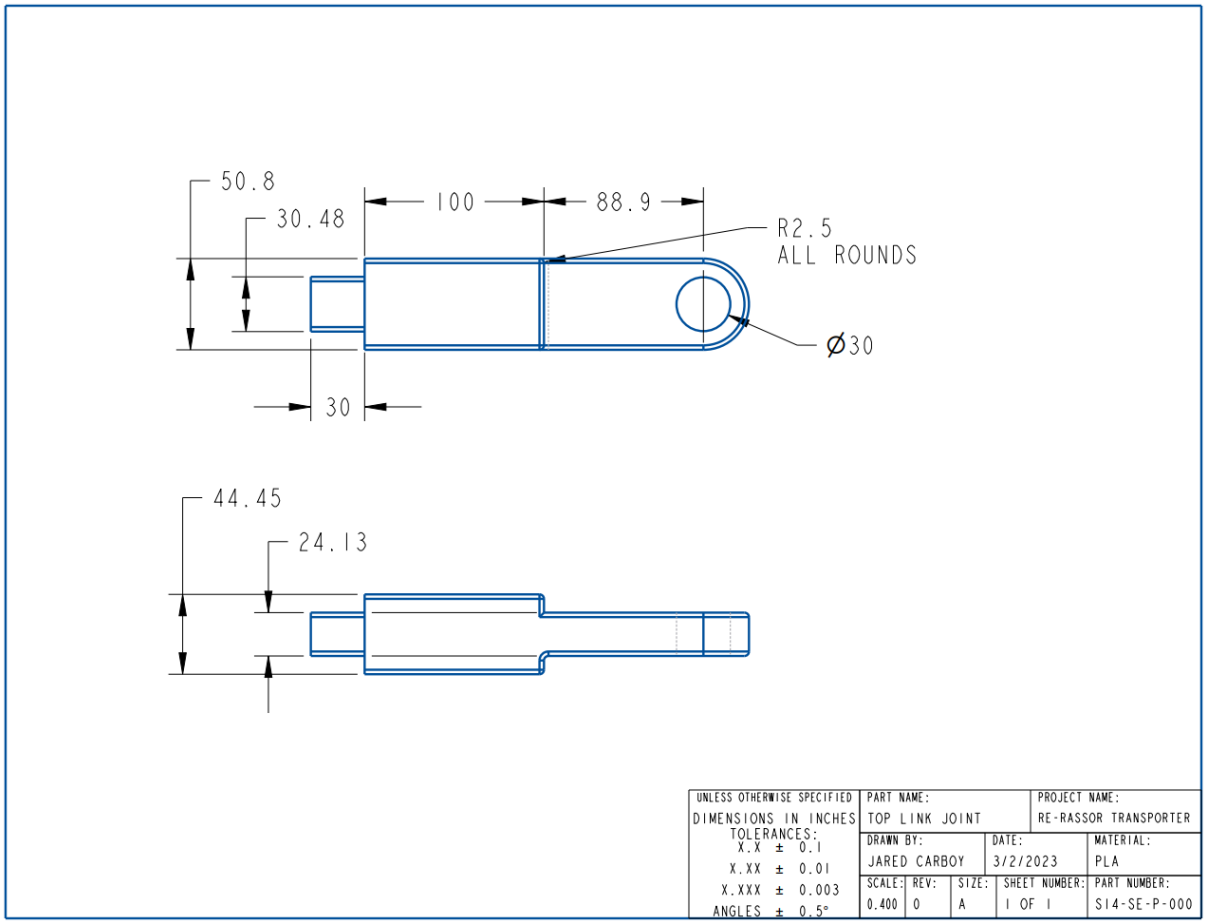



12.7
R2.54
4X
63.5
101.6
6.35
R32.1
4X
Ø30
Ø4.06
10X
Ø95.25
UNLESS OTHERWISE SPECIFIED
DIMENSIONS IN INCHES
TOLERANCES:
X.X ± 0.1
X.XX ± 0.01
X.XXX ± 0.003
ANGLES ± 0.5°
PART NAME:
BOTTOM LINKAGE
PROJECT NAME:
RE-RASSOR TRANSPORTER
DRAWN BY:
JARED CARBOY
DATE:
3/2/2023
MATERIAL:
PLA
SCALE:
1.000
REV:
0
SIZE:
A
SHEET NUMBER:
1 OF 1
PART NUMBER:
S14-SE-P-000



UNLESS OTHERWISE SPECIFIED DIMENSIONS IN INCHES TOLERANCES: X.X ± 0.1 X.XX ± 0.01 X.XXX ± 0.003 ANGLES ± 0.5°	PART NAME: TOP LINK JOINT			PROJECT NAME: RE-RASSOR TRANSPORTER
	DRAWN BY: JARED CARBOY		DATE: 3/2/2023	MATERIAL: PLA
	SCALE: 0.400	REV: 0	SIZE: A	SHEET NUMBER: 1 OF 1	PART NUMBER: S14-SE-P-000


SEE DETAIL A
R0.5
AROUND
2
Ø30
20
R2
AROUND
28.58
100
37
15
R10
2X
R5
2X
R1
AROUND
60
15
45
UNLESS OTHERWISE SPECIFIED
DIMENSIONS IN INCHES
TOLERANCES:
X.X ± 0.1
X.XX ± 0.01
X.XXX ± 0.003
ANGLES ± 0.5°
PART NAME:
HOOK
PROJECT NAME:
RE-RASSOR TRANSPORTER
DRAWN BY:
JARED CARBOY
DATE:
3/2/2023
MATERIAL:
PLA
SCALE:
0.500
REV:
0
SIZE:
A
SHEET NUMBER:
1 OF 2
PART NUMBER:
S14-SE-P-000


50.8
30.48
100
88.9
R2.5
ALL ROUNDS
Ø30
30
44.45
24.13
UNLESS OTHERWISE SPECIFIED
DIMENSIONS IN INCHES
TOLERANCES:
X.X ± 0.1
X.XX ± 0.01
X.XXX ± 0.003
ANGLES ± 0.5°
PART NAME:
TOP LINK JOINT
PROJECT NAME:
RE-RASSOR TRANSPORTER
DRAWN BY:
JARED CARBOY
DATE:
3/2/2023
MATERIAL:
PLA
SCALE:
0.400
REV:
0
SIZE:
A
SHEET NUMBER:
1 OF 1
PART NUMBER:
S14-SE-P-000

Appendix B – Arduino Code

```
#include <math.h>

// link lengths
double w = 0.34;// % gnd link (rerassor body)
double a1 = 0.175;// % link 1 (inner 5-bar link)
double a2 = a1;// % link 1 on the right side (the same bc yeah)
double b1 = 0.385;// % link 2 (outter 5 bar link)
double b2 = b1;// % link 2 read 2 comments up
double ground_th = 0;
double gnd_th = 0;

void stpCtrlR();
void stpCtrlL();

int steps[4] = {0b1,0b1000,0b010,0b100};
int stepsR[4] = {0b1,0b1000,0b010,0b100};
int stepsL[4] = {0b100,0b10,0b01000,0b1};
int i = 0;
//right motor
double desThR = 0;
int curPosR = 50;
int desPosR = 50; // desired pos
//left motor
double desThL = 0;
int curPosL = 50;
int desPosL = 50; // desired pos
//stepper delay
int dt = 50;

void setup() {
  // put your setup code here, to run once:
  DDRA = 0xFF;
  DDRB = 0xFF;
  Serial.begin(9600);
  delay(5000);
}

void loop() {
  // define height of lift
  double h = 0.3; // meters
  int ddt = 13;
  if(i<80){
    h = .3;
   gnd_th = 0;
  }else if(i<160){
    h = .5;
    gnd_th = 0;
  }else if(i<160+ddt){
    h = .5;
    gnd_th = gnd_th + .015;
    delay(10);
  }else if(i<160+ddt*3){
    h = .5;
    gnd_th = gnd_th - .015;
    delay(10);
```

```
  }else if(i<160+ddt*4){
    h = .5;
    gnd_th = gnd_th + .015;
    delay(10);
  }
  else if(i<160+ddt*6){
    Serial.print(i);
    Serial.print("\tEND\t\t\t");
    h = .5;
    delay(50);
  }else{
    h = .3;
  }

//  if(i==250)
//    i=161;

  Serial.print("thgnd = ");
  Serial.print(gnd_th*180/3.14159265359);

//  // read pot for ground angle
//  double sensorValue = analogRead(A0);
//  // map the pot value to a radian value
//  double ground_th = map(sensorValue, 0, 1023, -2.35, 2.35);
//   ground_th = sensorValue/1023*4.7-2.35;
//  if(ground_th > 0.0873) // upper limit
//    ground_th = 0.0873;
//  if(ground_th < -0.0873) // lower limit
//    ground_th = -0.0873;
  ground_th = 1.57 - gnd_th;
//
//  ground_th = 0;
//
//  Serial.print("g_th = ");
//  Serial.print(ground_th);
  // calcualte xp and yp that will balence the robot
  double yp = h*sin(ground_th);
  double xp = h*cos(ground_th);


///TEST/////////////////////////////////////////////////////////////////////////
/////////////////
//  if(i<100){
//  xp = 0;
//  yp = .3;
//  }else{
//    xp = 0;
//    yp = .52;
//  }
//
//  if(i==200)
//    i=0;

/////////////////////////////////////////////////////////////////////////////////
/////////////////
```

```
  // calculate the motor angles based on x and y
  angCalc(xp,yp); // returns value in desThR and desThL
//  Serial.print("xp = ");
//  Serial.print(xp);
//  Serial.print("\typ = ");
//  Serial.print(yp);
//
//  Serial.print("\tthR = ");
//  Serial.print(desPosR);
//  Serial.print("\tthL = ");
//  Serial.println(desPosL);

  desPosR = (desThR / 0.0314)+1; // convert to steps
  desPosL = desThL / 0.0314; // convert to steps



    Serial.print("\tdesThL = ");
    Serial.print(100-desPosL);
    Serial.print(" ");
    Serial.print(100-curPosL);

    Serial.print("\tdesThR = ");
    Serial.print(desPosR);
    Serial.print(" ");
    Serial.println(curPosR);


//
///TEST////////////////////////////////////////////////////////////////////
//////////////////
//  if(i<100){
//  desPosR = 20;
//  desPosL = 20;
//  }else{
//    desPosR = 50;
//    desPosL = 50;
//  }
//
////////////////////////////////////////////////////////////////////////////
//////////////////

  stpCtrlR();
  stpCtrlL();
  delay(dt);
  i++;
}

// caluclates motor angles based on an x and y coordinate centered at the
midpoint
void angCalc(double xp,double yp){
    xp = xp+0.34/2;

    double c1 = sqrt( pow(xp,2) + pow(yp,2) );
    double c2 = sqrt( pow(xp-w,2) + pow(yp,2) );

    double alpha1 = acos(xp/c1);
```

```
    double alpha2 = acos( (-xp+w)/c2 );

    double beta1 = acos( (pow(b1,2) - pow(a1,2) - pow(c1,2))/(-2*a1*c1) );
    double beta2 = acos( (pow(b2,2) - pow(a2,2) - pow(c2,2))/(-2*a2*c2) );

    desThL = beta1+alpha1;
    desThR = 3.14159265359-(beta2+alpha2);

//    Serial.print("desThL = ");
//    Serial.print(180-desThL*180/3.14159265359);
//
//    Serial.print("\tdesThR = ");
//    Serial.println(desThR*180/3.14159265359);
}

// move motor on the right based on desPosR
void stpCtrlR(){

  // move CW
  if( curPosR < desPosR)
    curPosR++;
  // move CCW
  if( curPosR > desPosR)
    curPosR--;

  //move the stepper motor or hold it if curPos = desPos
  PORTA = steps[curPosR%4];
}


// move motor on the left based on desPosL
// the direction of the angle is reversed from moving the right motor
void stpCtrlL(){

  // move CW
  if( curPosL < desPosL)
    curPosL++;
  // move CCW
  if( curPosL > desPosL)
    curPosL--;

  //move the stepper motor or hold it if curPos = desPos
  PORTB = steps[curPosL%4];

}
```