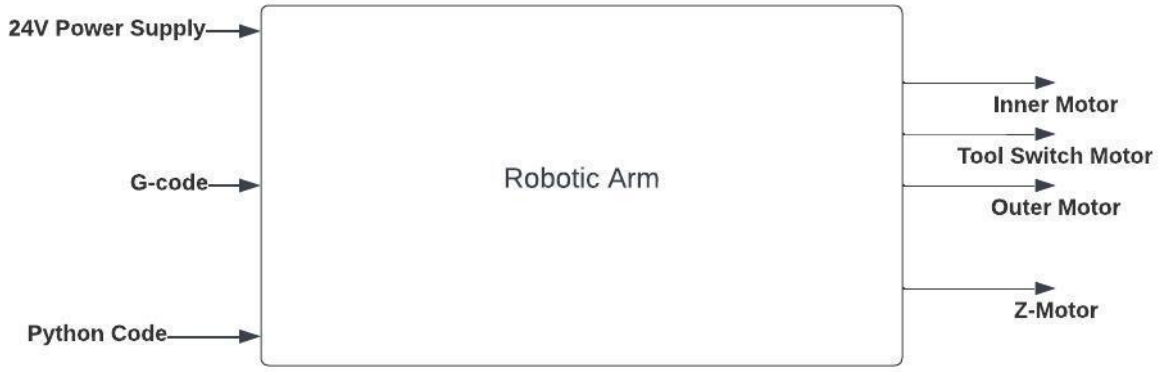
Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine
Robotic Arm 002-1

System Level Verification



Visual 1: Black Box Diagram

System function description The robotic arm has 7 main functionality components. The first being that the arm must be able to draw at a speed greater than 4 inches per second. The arm must draw a 10 inch straight line that varies by less than a quarter of an inch. The arm must use SCARA technology with two rotatable joints. The arm will use G-code as input commands and will be processed using python. The arm will be able to use different writing devices and be switched automatically using a 4th motor. The arm will also be able to move up and down to start and stop drawings.

Interface name {**must be a system level interface — no showing things "inside" the system**}	Interface properties {**each interface must have at least two properties; power interfaces must list all four required properties**}
24V Power Supply	(1)Vmax: 36 V (2)Vmin: 24 V I nominal: 300 mA I peak: 600 mA
Python Code	(3)Parsing G-code (4)Inverse Kinematics for finding angles for the angle
G-Code	(5)G command per line (6)Coordinate per line
Inner Motor	(7)Number Steps (8)Direction
Outer Motor	(9)Number Steps

	(10)Direction
Tool Change Motor	(11)Tool to use (12)Degrees to move
Z Motor	(13)Number Steps (14)Direction

Table 1: Interface definition table

System test plan:

Plug in the arduino to your pc, and turn on power from the power supply to 24V to 36V (recommend test with 24V)(1)(2). Run a python script which then asks for a G-code file. The python script will then parse through the G-code file.(3) After parsing the file(5)(6) the coordinates will be processed through inverse kinematics(4) to get the angles the steppers must move to reach the desired coordinates. The angles are then divided to get the number of steps needed to get the desired angle for each motor. This is then written to the arduino which sends the signals to the motors. The outer motor and inner motor are sent two sets of instructions, one to determine the direction they should move(7)(9), and the other to determine the angle they should move.(8)(10) If the G command “M6” is received, the arduino will send a tool number to

Team Members	Evan Shaw	Peter Hull	Sebastian Thorp Astrid Delestine
	Requirements		
	Customer Requirement: The system should be fast.		
	Engineering Requirement: The system must draw faster than 4 inches per second.		
	Customer Requirement: The system must be accurate.		
	Engineering Requirement: The system must draw a 10 inch straight line +/- .25 inch. This includes both the overall length of the line and ensuring the line does not vary more than .25 inches of being perfectly straight.		
	Customer Requirement: The system needs to be inexpensive and manageable to manufacture.		
	Engineering Requirement: The robotic arm will use a SCARA topology, with two rotating joints to control arm actuation.		
	Customer Requirement: The system must have a commonly known interface.		
	Engineering Requirement: Controlling commands will be input as G-code commands. These commands must be made available within the Python or MATLAB GUI.		
	G0, G1, G90, G91, G20, G21, M2, M6, M72.		
	Customer Requirement: The system must use different types of writing tools.		
	Engineering Requirement: Upon receiving an M6 command the machine operator must mount a crayon, pen, or pencil within 15 seconds.		
	Additional Requirements		
	Customer Requirement: Automatic tool changer		
	Engineering Requirement: Add the functionality to connect to a seperate computer to access control to the arm		
	Customer Requirement: Z axis motor control		
	Engineering Requirement: Add an additional motor on the base of the stand to control the upward motion of the arm		

the servo motor (11) which will then move a specified number of degrees to change the tool.(12) Before and after the tool is changed, the Z motor will move up before the tool is changed(13) in order to reduce extra friction, and down after the tool has been changed(14)

Link to drive with all demonstration videos:

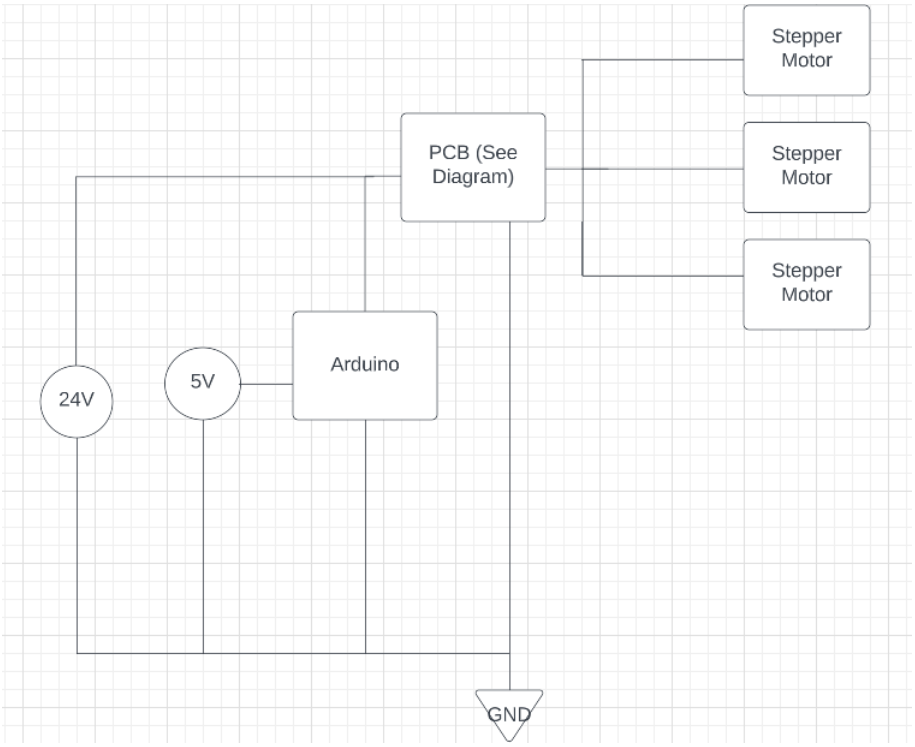
https://drive.google.com/drive/folders/1-rMVASdVwKBNt0ySMDoIL1zaB3FnI-sC?usp=sharing

Team Member	Time Spent
Sebastian Thorp	21.3
Peter Hull	21.6
Evan Shaw	20.5
Astrid Delestine	22

Table 2: Time Report (time in hours)

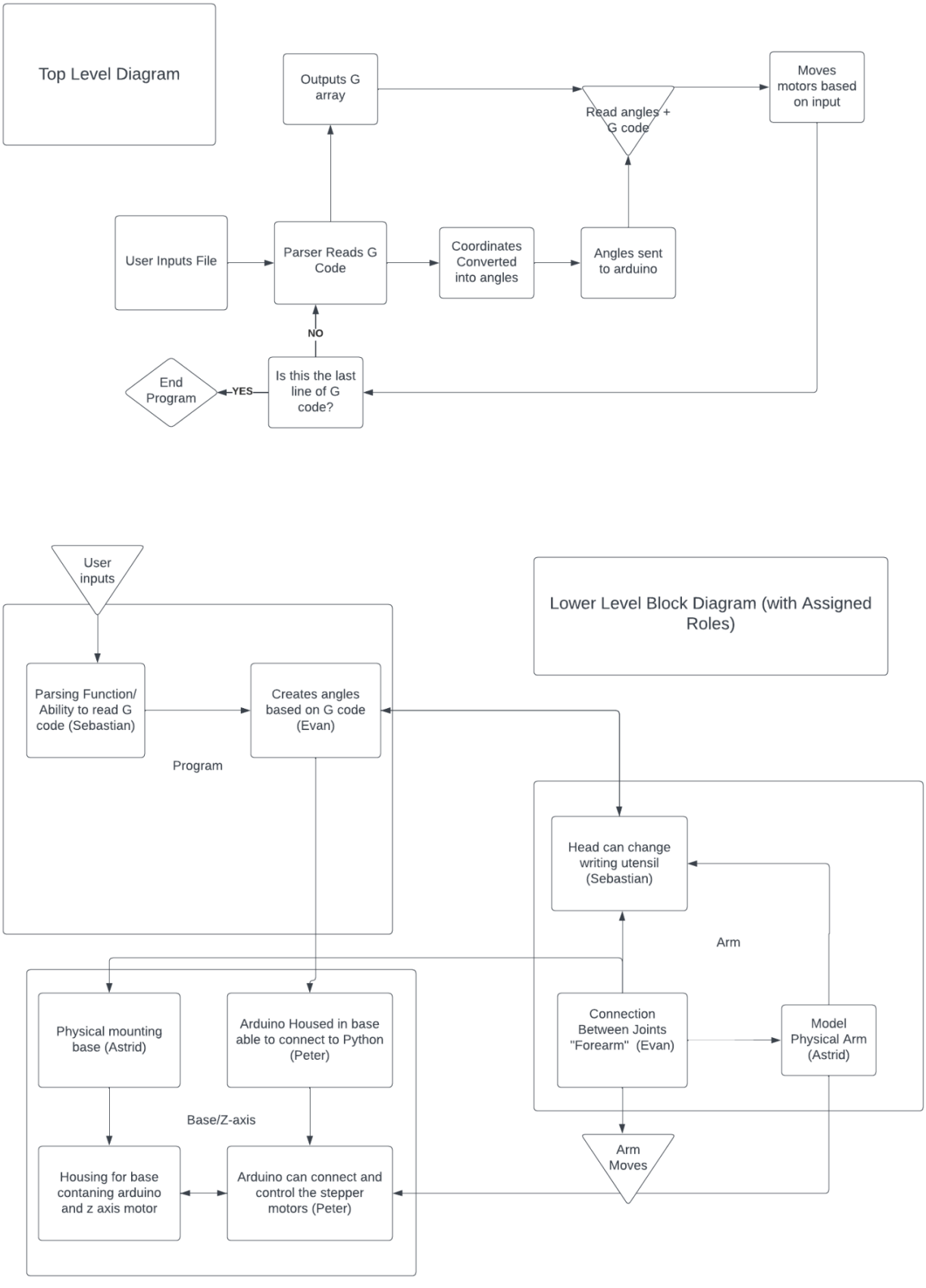
Object	Quantity	Price (individual)	Team Member	Total Cost
Arduino	2	$6	Peter Hull	$16
Stepper Motor	4	$8	Peter Hull	$24
10 Ft PVC	1	$5	Astrid Delestine	$5
PCB	10	$3	Evan Shaw	$30
Base Materials	1	Free (OSU provided)		
Stepper Drivers	6	$3	Peter Hull	$18
			Total	$93

Table 3: Bill of materials



Visual 2: Basic electrical schematic

Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine
Robotic Arm 002-1



Visual 3: Top Level Block Diagram

Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine
Robotic Arm 002-1

```python
import math
import pyfirmata
import time
from pyfirmata import Arduino, util
board = pyfirmata.Arduino('COM6')

def welcome():
    file = input('What file would you like to run?:    ')
    return file

#Used to parse and seperate the G code file into correst peicess
def parse_file():
    #filename = input("What file would you like to run?:  ")
    filename = "example1.txt"
    file_object = open(filename, "r")
    #file_object = open(file, "r")
    i = 0
    m72check = 0
    coord_dict['Z'].append(0)
    coord_dict['T'].append('T1')
    for curr_line in file_object:
        curr_line = curr_line.replace(" ","")
        curr_line = curr_line.replace("\n","")
        if "M2" in curr_line:
            break
        if "M72" in curr_line:
            m72check = 1
            continue
        if "M6" in curr_line:
            coord_dict['Y'].append(coord_dict['Y'][i-1])
            coord_dict['X'].append(coord_dict['X'][i-1])
            T_loc = curr_line.find("T")
            coord_dict['T'].append(curr_line[T_loc:])
            coord_dict['G'].append('M6')
            coord_dict['F'].append(coord_dict['F'][i-1])
            coord_dict['Z'].append(coord_dict['Z'][i-1])
            i = i+1
            continue
        if "Z" in curr_line:
            z_loc = curr_line.find("Z")
            coord_dict['G'].append('G100')
            coord_dict['T'].append(coord_dict['T'][i-1])
            coord_dict['Y'].append(coord_dict['Y'][i-1])
            coord_dict['X'].append(coord_dict['X'][i-1])
            coord_dict['F'].append(coord_dict['F'][i-1])
            coord_dict['Z'].append(curr_line[z_loc:])
```

Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine
Robotic Arm 002-1

```
            i = i + 1
            continue
        x_loc = curr_line.find("X")
        y_loc = curr_line.find("Y")
        g_loc = curr_line.find("G")
        if "G01" in curr_line:
            f_loc = curr_line.find("F")
            coord_dict['Y'].append(curr_line[y_loc+1:f_loc])
            coord_dict['T'].append(coord_dict['T'][i-1])
            coord_dict['X'].append(curr_line[x_loc+1:y_loc])
            coord_dict['G'].append(curr_line[:x_loc])
            coord_dict['F'].append(curr_line[f_loc+1:])
            coord_dict['Z'].append(coord_dict['Z'][i-1])
        else:
            coord_dict['Y'].append(curr_line[y_loc+1:])
            coord_dict['X'].append(curr_line[x_loc+1:y_loc])
            coord_dict['G'].append(curr_line[:x_loc])
            coord_dict['F'].append(coord_dict['F'][i-1])
            coord_dict['Z'].append(coord_dict['Z'][i-1])
            coord_dict['T'].append(coord_dict['T'][i-1])
        if m72check == 1:
            coord_dict['F'][i] = 1000
        i = i+1



    #print(coord_dict)

#Uses inverse kinematics to determine the desired angle based on coordinates
def angle(x, y, scale):
    L = 5.25
    if(scale == 'mm'):
        L = 5.25 * 25.4

    hypotnuse = math.sqrt(pow(x,2) + pow(y,2))
    s1 = hypotnuse/2
    s2 = math.sqrt(pow(L,2) - pow(s1,2))

    aB = math.atan(s2/s1)
    x2 = x/2
    y2 = y/2

    aA = math.atan(x2/y2)

    servo1angle = aA+aB
    servo1angle = (servo1angle/ (2* 3.141)) * 360
```

Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine
Robotic Arm 002-1

```
93
94      x3 = -L*math.sin(aA+aB)
95      y3 = -L*math.cos(aA+aB)
96      servo2angle= math.atan((x-x3)/(y-y3));
97
98      servo2angle= (servo2angle / (2 * 3.14159)) * 360
99
100     if ((y-y3)>0):
101         servo2angle=servo2angle-180
102     angles = [servo2angle, servo1angle]
103     return angles
104
105 #Runs motors like normal, changes based on g commands
106 def run_servos_G00(n1, n2, c1, c2, scale, line):
107     angles = angle(n1, n2, scale)
108     servo1angle = angles[0]
109     servo2angle = angles[1]
110     print(len(servo), len(servo2))
111     servo.append(servo1angle)
112     servo2.append(servo2angle)
113     #Determines how far to travel based on new desired angle and previous ar
114     movementservo1 = servo[len(servo)-1] - servo[len(servo)-2]
115     movementservo2 = servo[len(servo2)-1] - servo[len(servo2)-2]
116     #Moves servo1 backwards IE negative angle
117     num1steps = movementservo1/1.8
118     num2steps = movementservo2/1.8
119     if(movementservo1 < 0):
120         board.digital[5].write(0)
121         for i in range(0-int(num1steps)):
122             board.digital[2].write(1)
123             board.digital[2].write(0)
124             time.sleep(1/(int(coord_dict['F'][line])))
125             #time.sleep(0.01)
126     #Servo angle is positive
127     else:
128         board.digital[5].write(1)
129         for i in range(int(num1steps)):
130             board.digital[2].write(1)
131             board.digital[2].write(0)
132             time.sleep(1/(int(coord_dict['F'][line])))
133             #time.sleep(0.01)
134     #Moves servo2 backwards IE negative angle
135     #print(servo2)
136     if(num2steps < 0):
137         board.digital[6].write(1)
138         for i in range(10*(0-int(num2steps))):
```

Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine

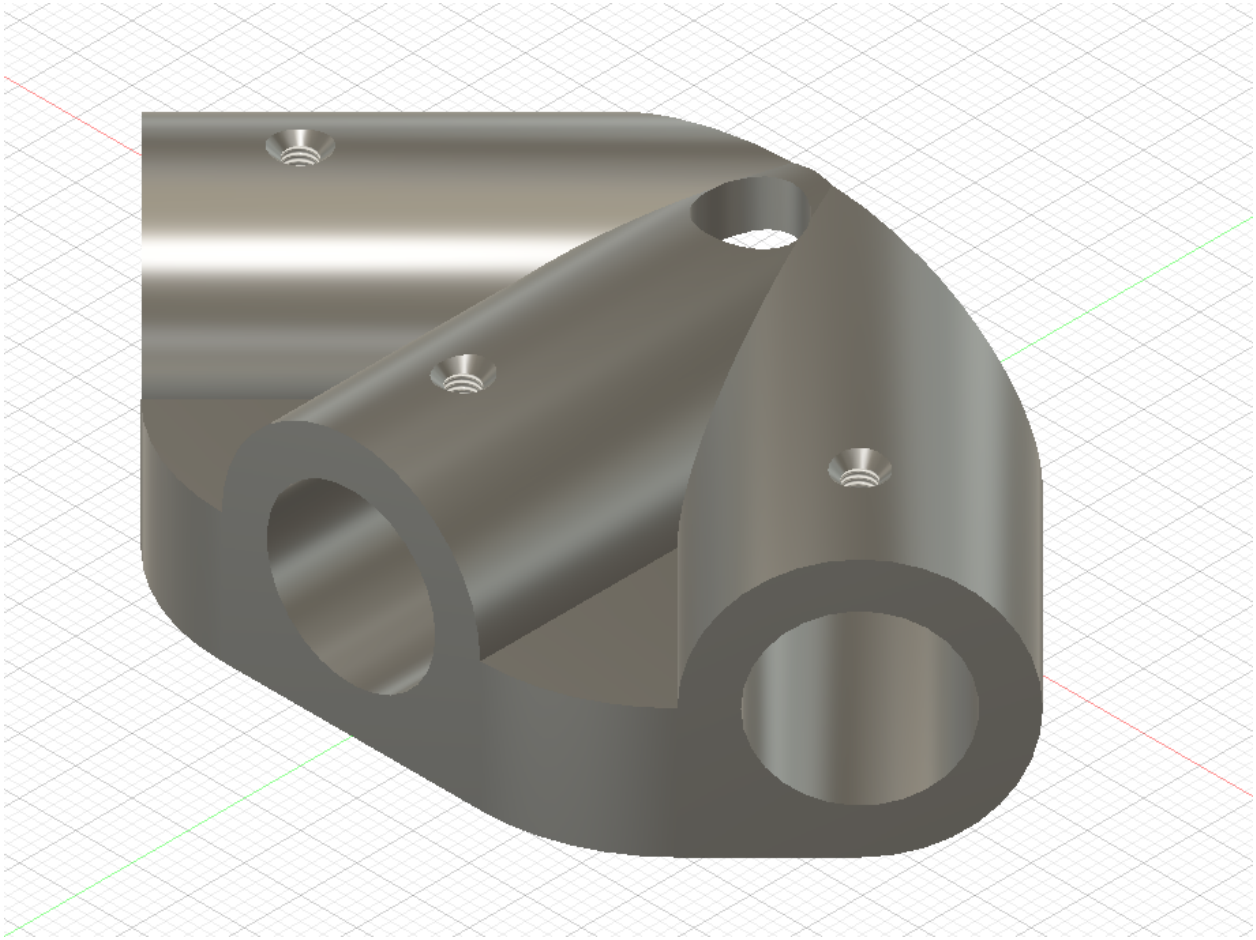
Robotic Arm 002-1

```
            board.digital[3].write(1)
            board.digital[3].write(0)
            time.sleep(1/(int(coord_dict['F'][line])))
            #time.sleep(0.01)
    #Servo2 angle is positive
    else:
        board.digital[6].write(0)
        for i in range(10 * int(num2steps)):
            board.digital[3].write(1)
            board.digital[3].write(0)
            time.sleep(1/(int(coord_dict['F'][line])))
            #time.sleep(0.01)

def lift_arm(height):
    if(height == 1):
        board.digital[7].write(1)
        for i in range(200):
            board.digital[4].write(1)
            board.digital[4].write(0)
            time.sleep(0.001)
    else:
        board.digital[7].write(0)
        for i in range(200):
            board.digital[4].write(1)
            board.digital[4].write(0)
            time.sleep(0.001)

#Once the G code file is parsed, we use this to run the entire program
def run_program():
    for i in range(len(coord_dict['X'])):
            time.sleep(2)
            print(int(coord_dict['X'][i]), int(coord_dict['Y'][i]))
            if coord_dict['G'][i] == 'G00':
                run_servos_G00(int(coord_dict['X'][i]), int(coord_dict['Y'][i]), int(coord_dict['X'][i-1]), int(coord_dict['Y'][i-1]), 'inches', i)
            elif coord_dict['G'][i] == 'G01':
                run_servos_G00(int(coord_dict['X'][i]), int(coord_dict['Y'][i]), int(coord_dict['X'][i-1]), int(coord_dict['Y'][i-1]), 'inches', i)
            elif coord_dict['G'][i] == 'G90':
                run_servos_G00(int(coord_dict['X'][i]), int(coord_dict['Y'][i]), int(coord_dict['X'][i-1]), int(coord_dict['Y'][i-1]), 'inches', i)
            elif coord_dict['G'][i] == 'G91':
                n1 = int(coord_dict['X'][i]) + int(coord_dict['X'][i-1])
                n2 = int(coord_dict['Y'][i]) + int(coord_dict['Y'][i-1])
                run_servos_G00(n1, n2, int(coord_dict['X'][i-1]), int(coord_dict['Y'][i-1]), 'inches' , i)
            elif coord_dict['G'][i] == 'G20':
                run_servos_G00(int(coord_dict['X'][i]), int(coord_dict['Y'][i]), int(coord_dict['X'][i-1]), int(coord_dict['Y'][i-1]), 'inches', i)
            elif coord_dict['G'][i] == 'G21':
                run_servos_G00(int(coord_dict['X'][i])/25.4, int(coord_dict['Y'][i])/25.4, int(coord_dict['X'][i-1]), int(coord_dict['Y'][i-1]), 'mm', i)
```

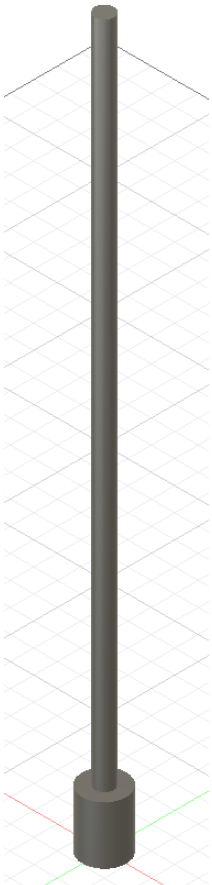
```
            elif coord_dict['G'][i] == 'M6':
                change_tool(coord_dict['T'][i])
            elif coord_dict['G'][i] == 'G100':
                lift_arm(coord_dict['Z'][i])

def moveServo(angle):
        board.digital[9].write(angle)

def change_tool(tool):
    lift_arm(1)
    if tool == "T1":
            moveServo(42)
    elif tool == "T2":
            moveServo(90)
    elif tool == "T3":
            moveServo(137)
    lift_arm(0)

servo = []
servo2 = []

coord_dict = {"X":[], "Y":[], "Z":[], "G":[], "F":[], "T":[]}
#file = welcome()
parse_file()
time.sleep(2)
print('go')
servo.append(0)
servo2.append(180)
run_program()
print(coord_dict)
print(servo)
print(servo2)
```

Visual 4: Code - Including Parsing function, welcome function, inverse kinematics function, movement function, lifting function, tool changing function, and main function

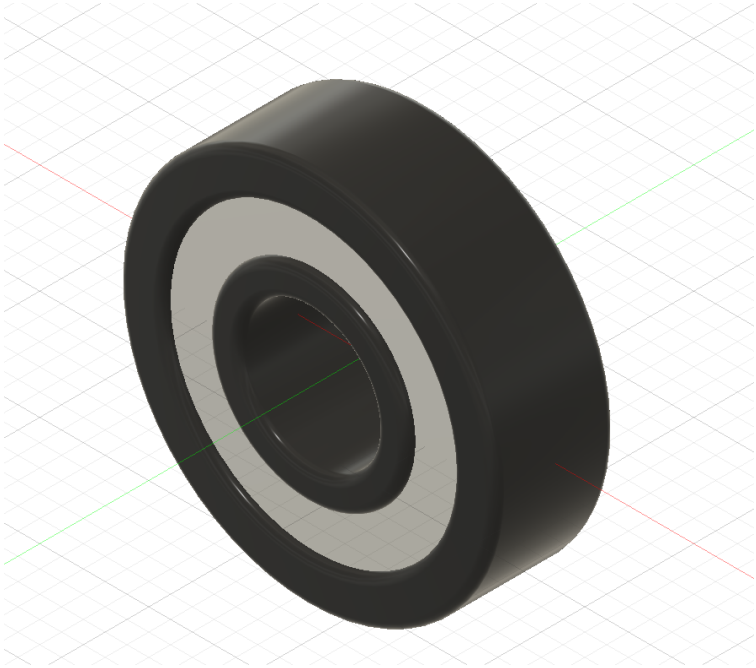
Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine
Robotic Arm 002-1



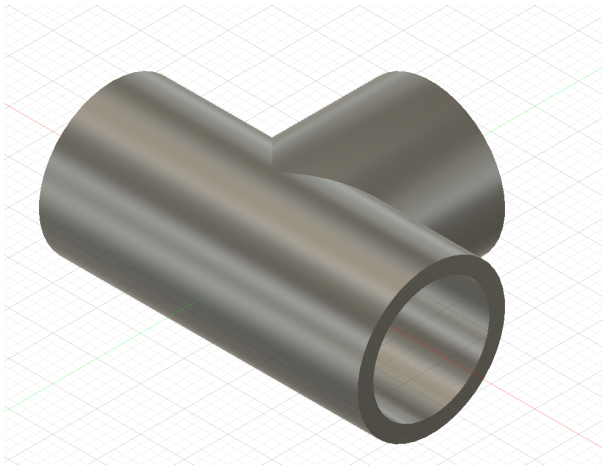
CAD Model 1: Tool holding model



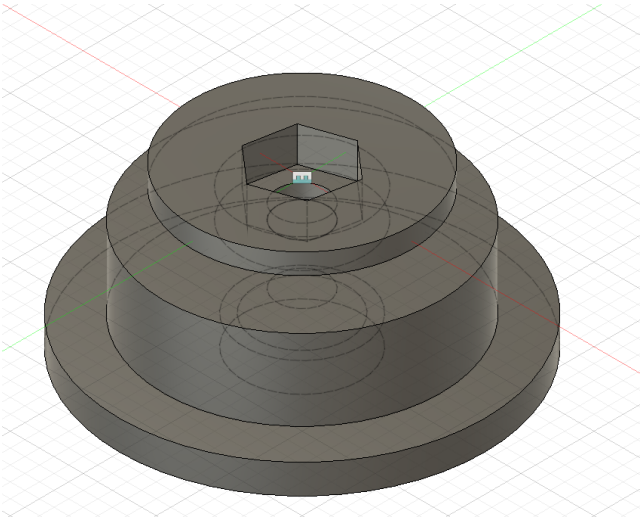
CAD Model 2: Threaded Rod (used for lifting arm)



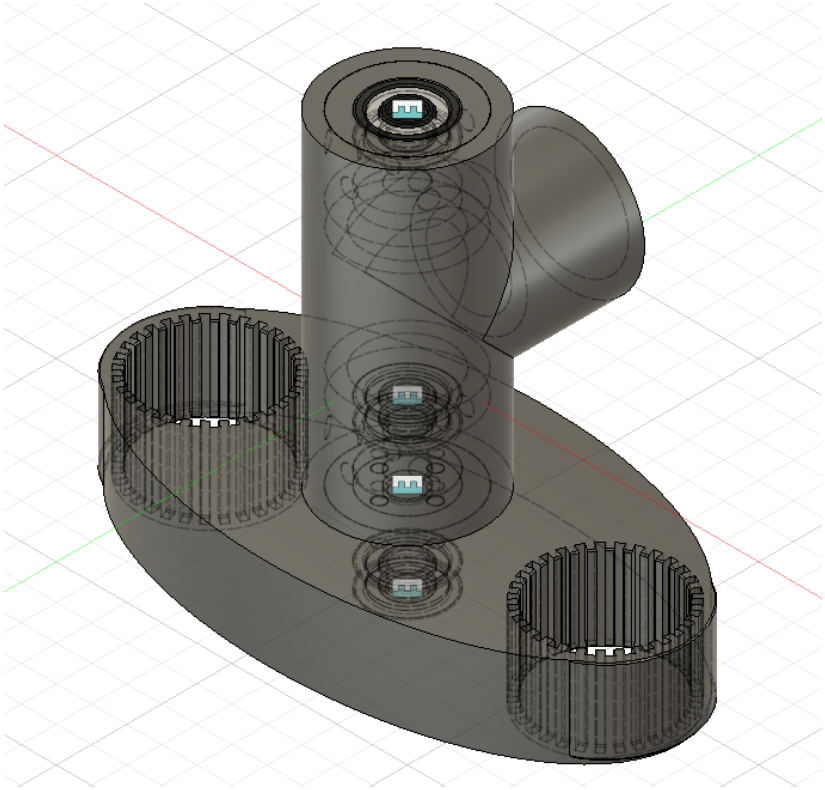
CAD Model 3: Bearing



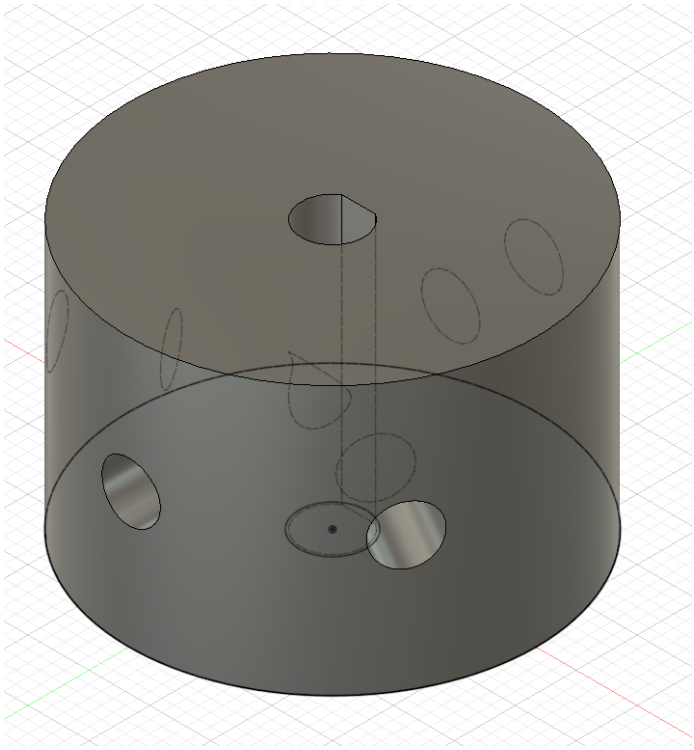
CAD Model 4: Brace holding PVC together



CAD Model 5: Stepper Holstering Mechanism

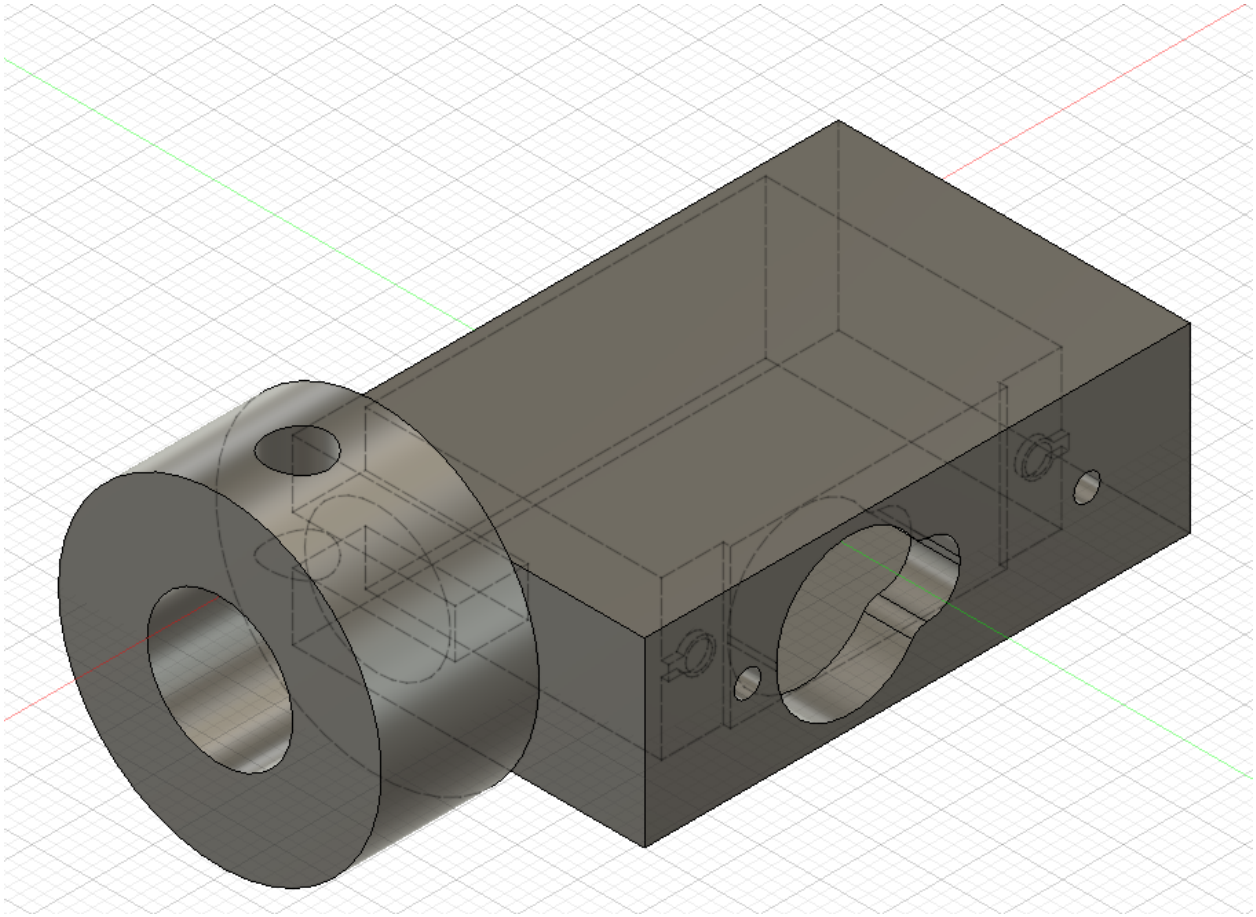


CAD Model 6: Brace for arm lifting mechanism

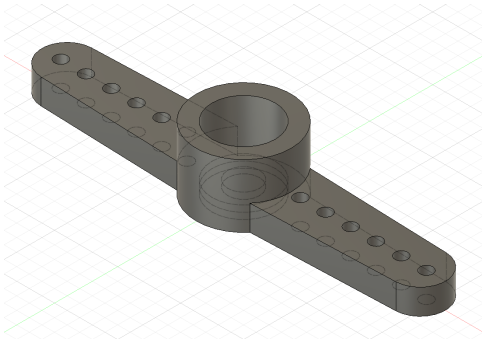


CAD Model 7: Stepper gripping mechanism

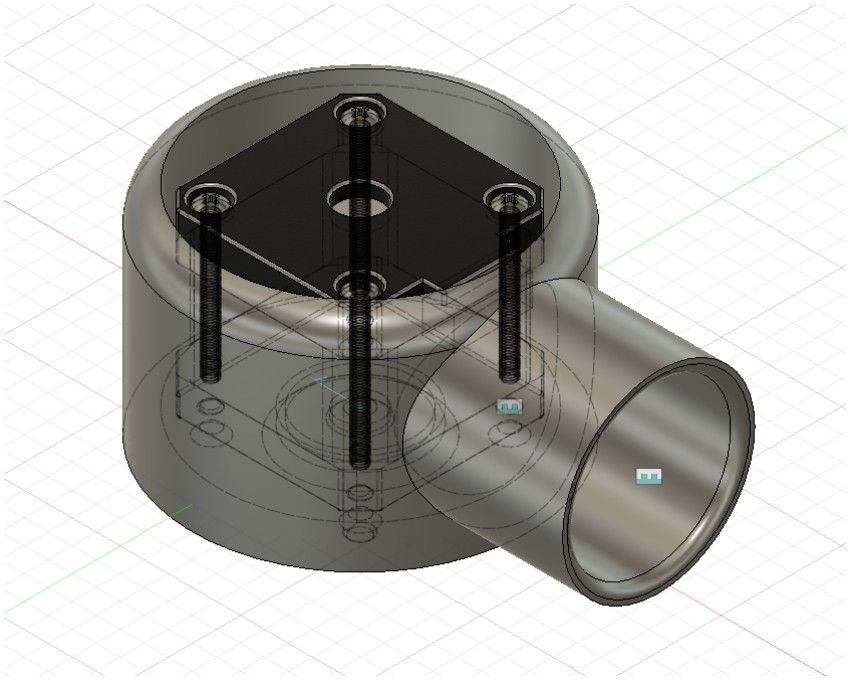
Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine
Robotic Arm 002-1



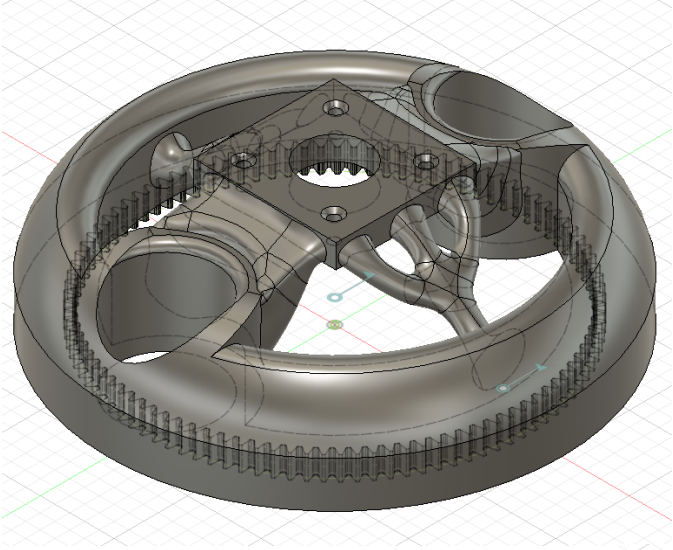
CAD Model 8: Servo motor Housing



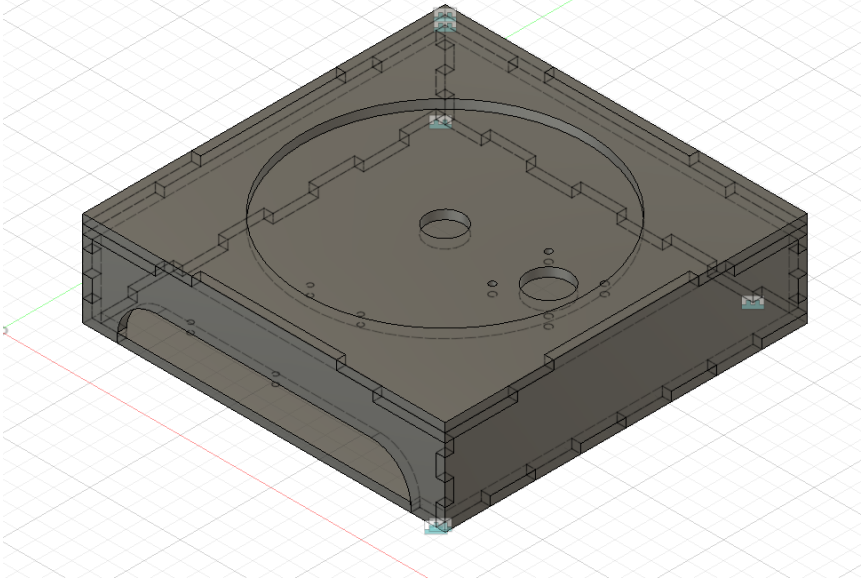
75CAD Model 9: Servo motor rotating propellor



CAD Model 10: Housing for stepper motor 2

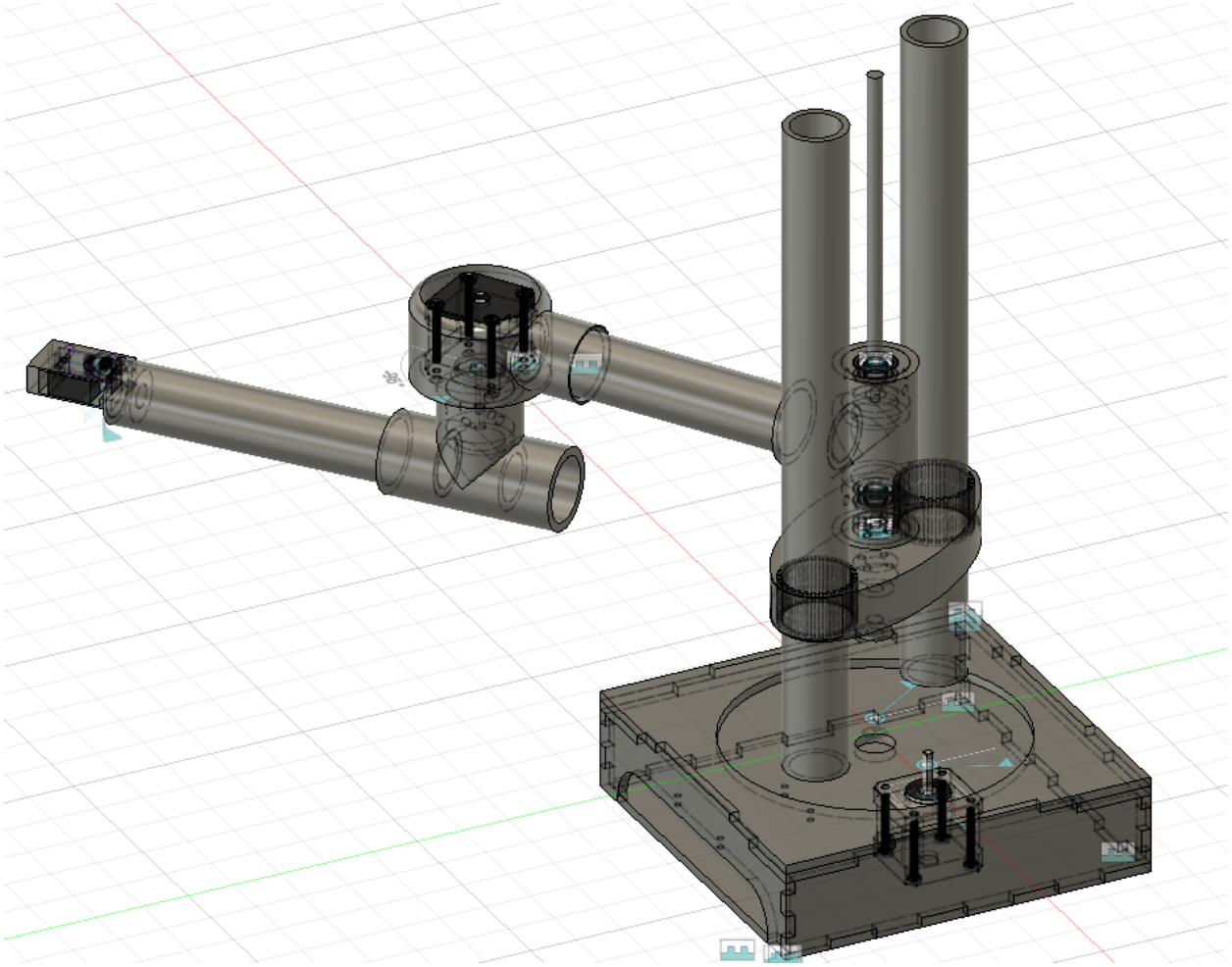


CAD Model 11: Rotating base of arm



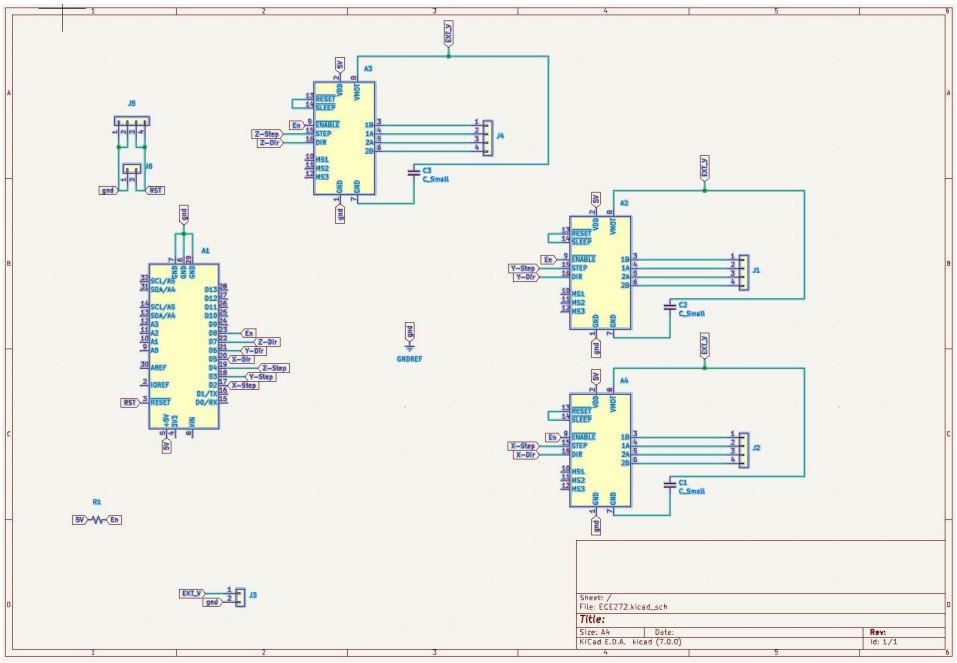
CAD Model 12: Housing for electrical components

Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine
Robotic Arm 002-1

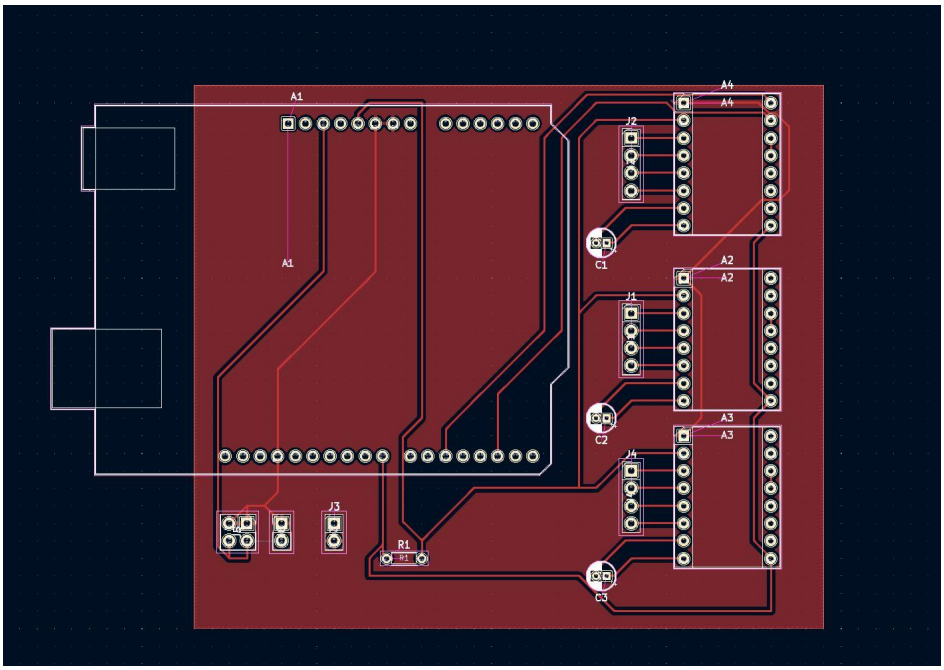


CAD Model 13: Complete CAD model of arm

Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine
Robotic Arm 002-1

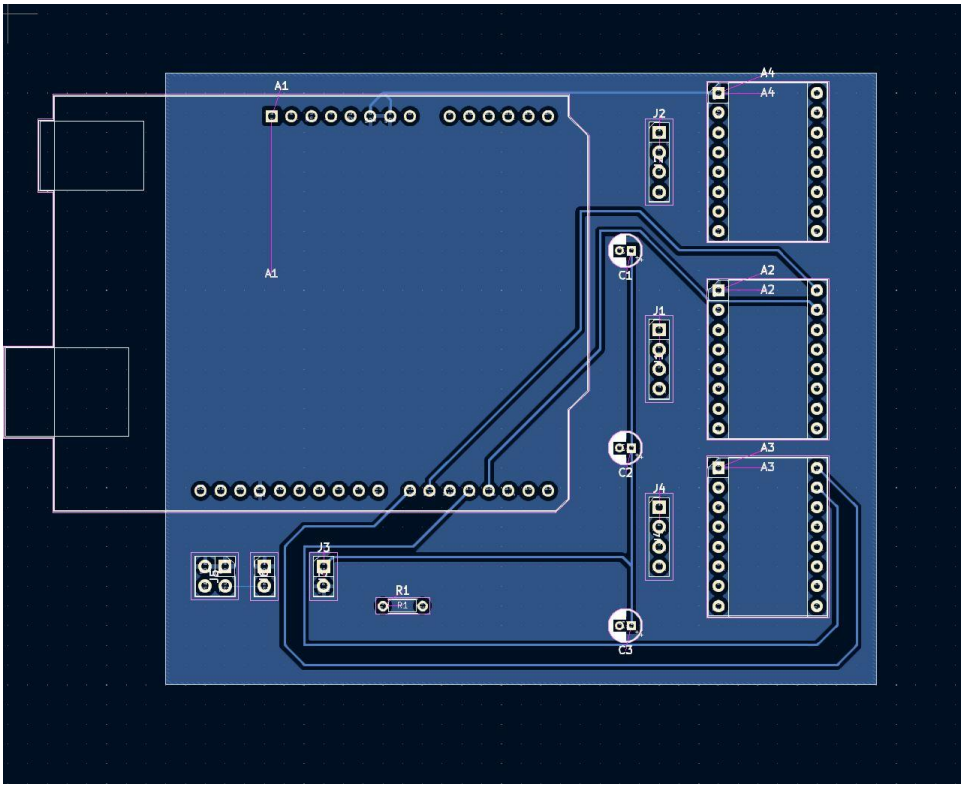


PCB Schematic



PCB Front Copper

Evan Shaw, Peter Hull, Sebastian Thorp, Astrid Delestine
Robotic Arm 002-1



PCB Back Copper