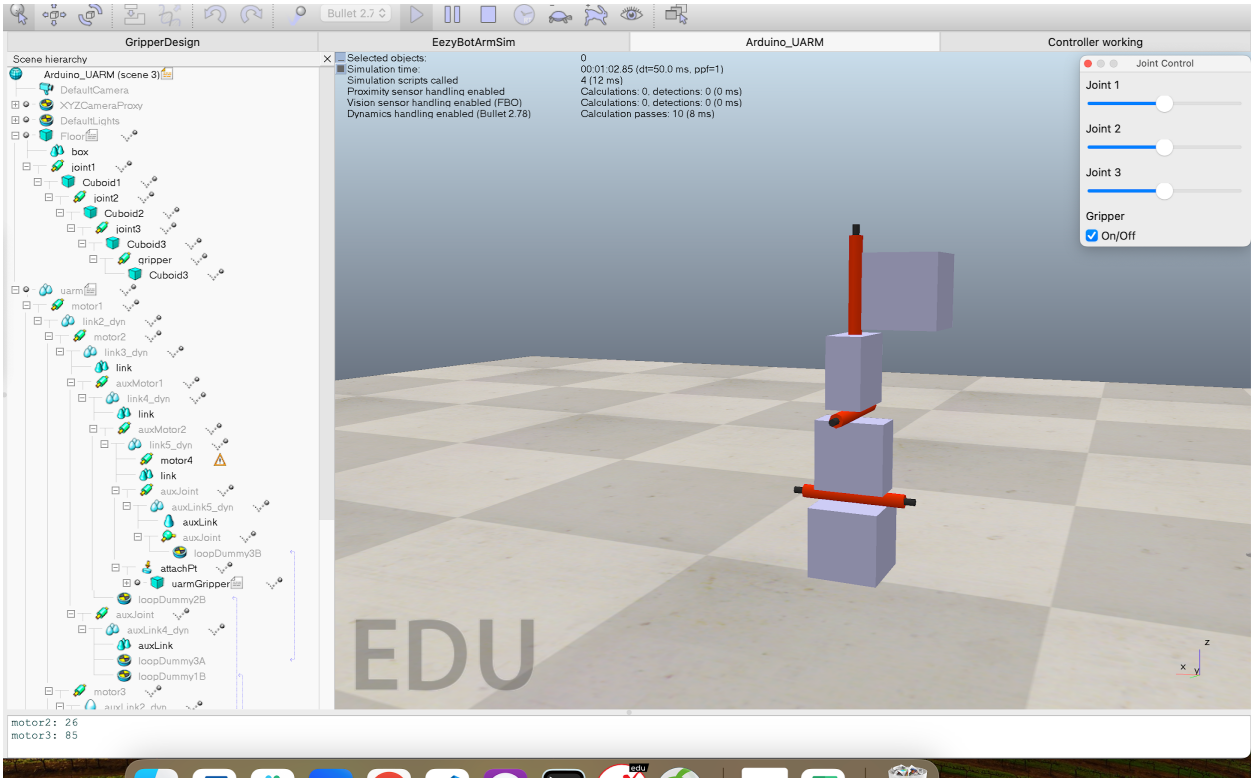
Data Collection

To collect data as SCRAP was developed, a Google spreadsheet was used to record various observations throughout each iteration. The spreadsheet helped to organize what successes and failures throughout each step of my design process. The final spreadsheet is included below, followed by significant iterations. For each of the seven significant iterations, a description of the achieved function at that point of the project is given, and an explanation of why that iteration was important is provided. Also, pictures of SCRAP at each iteration are included.

Iteration # 2: A model made of 4 blocks with 4 joints controlled by sliders prints the position of each joint continuously to the terminal in the Simulation when a joint is changed. This iteration was important as it was the first model that achieved motion control with the sliders successfully.

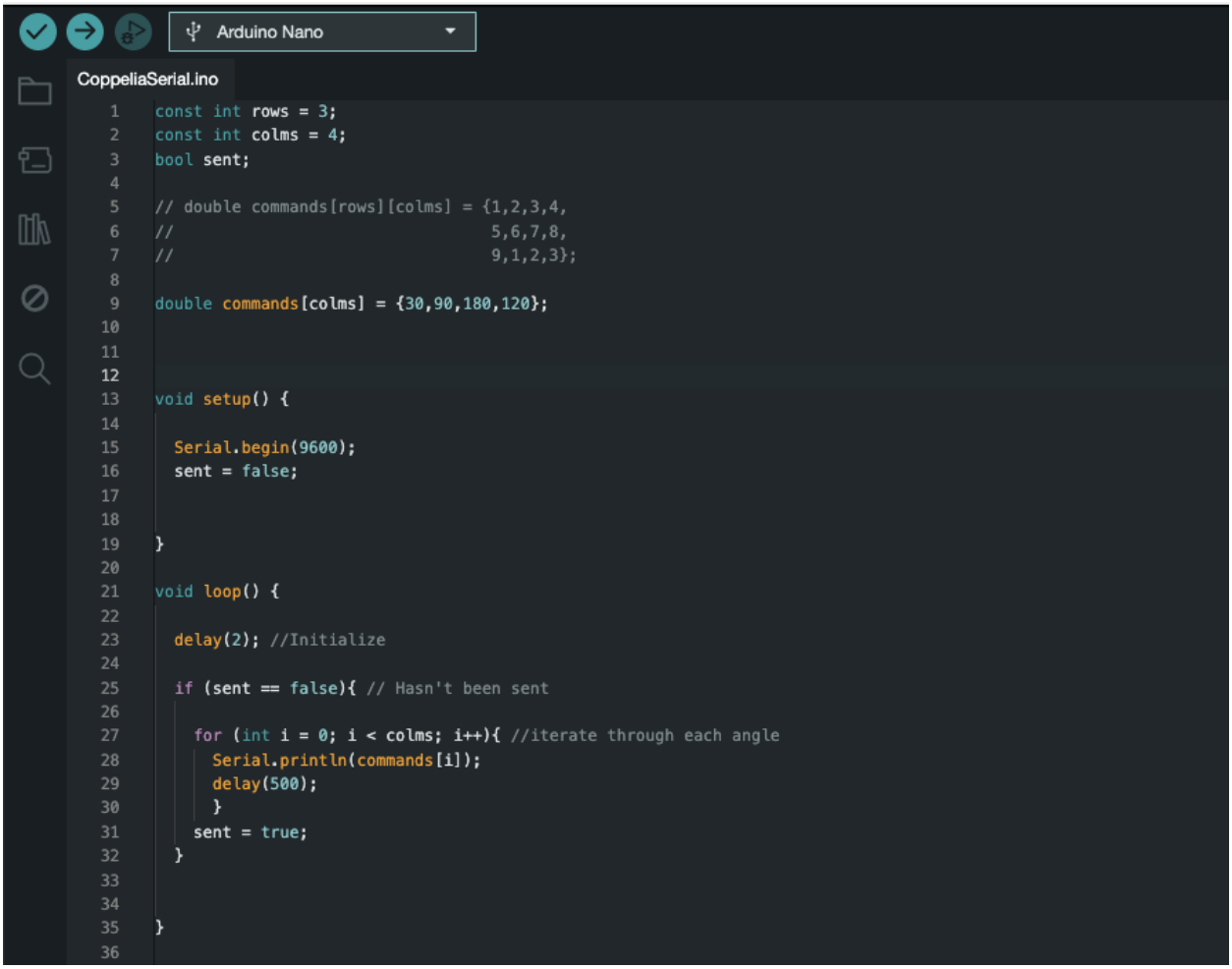



```python
#python

import serial.tools.list_ports
import serial
import time


def sysCall_init():

    findPort()

    global joint1
    joint1 = sim.getObject('./joint1')
    print("joint1: " + str(joint1))
    global joint2
    joint2 = sim.getObject('./joint2')
    print("joint2: " + str(joint2))
    global joint3
    joint3 = sim.getObject('./joint3')
    print("joint3: " + str(joint3))
    global gripper
    gripper = sim.getObject('./gripper')
    print("gripper: " + str(gripper))

    global q1ref
    q1ref = math.radians(90)
    global q2ref
    q2ref = math.radians(90)
    global q3ref
    q3ref = math.radians(90)
    global gripref
    gripref = math.radians(90)
    sim.setJointTargetPosition(joint1, q1ref)
    sim.setJointTargetPosition(joint2, q2ref)
    sim.setJointTargetPosition(joint3, q3ref)
    sim.setJointTargetPosition(gripper, gripref)


    ui = simUI.create('<ui enabled="true" modal="false" title="Joint Control" closeable="true" layout="vbox" placement="relative" position="20,20">'
    '<label enabled="true" text="Joint 1"></label>'
    '<hslider enabled="true" minimum="-90" maximum="90" on-change="onJ1Change"></hslider>'
    '<label enabled="true" text="Joint 2"></label>'
    '<hslider enabled="true" minimum="-90" maximum="90" on-change="onJ2Change"></hslider>'
    '<label enabled="true" text="Joint 3"></label>'
    '<hslider enabled="true" minimum="-90" maximum="90" on-change="onJ3Change"></hslider>'
    '<label enabled="true" text="Gripper"></label>'
    '<checkbox checked="true" text="On/Off" on-change="onGripperChange"/>'
    '</ui>')
```

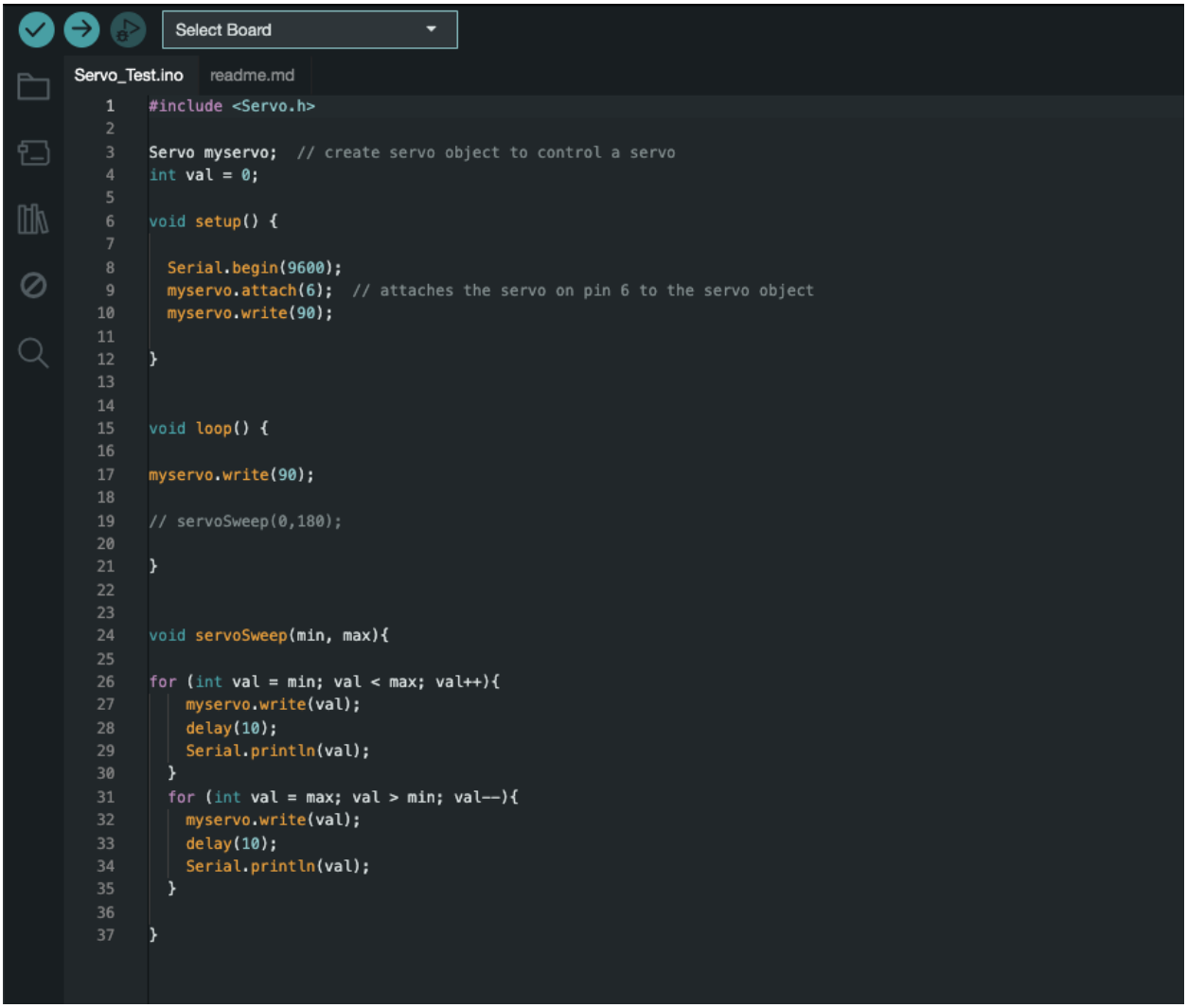
```
def findPort():
    ports = serial.tools.list_ports.comports()
    global finalPort
    finalPort = ''
    portList = []

    for onePort in ports:
        portList.append(str(onePort))
    for port in portList:
        usb = ""
        for i in range (8,17):
            usb += port[i]
        if usb == "usbserial":
            for i in range (0,22):
                finalPort += port[i]
            break
    print("final port: " + finalPort)

    global ser_com
    ser_com = serial.Serial(finalPort, 9600)
    print("initializing...")
    time.sleep(1)


def onQ1Change(uiHandle, id, newValue):

    global q1ref
    q1ref = math.radians((newValue + 90))
    sim.setJointTargetPosition(joint1, q1ref)
    sendNewCoords()

def onQ2Change(uiHandle, id, newValue):

    global q2ref
    q2ref = math.radians((newValue + 90))
    sim.setJointTargetPosition(joint2, q2ref)
    sendNewCoords()

def onQ3Change(uiHandle, id, newValue):

    global q3ref
    q3ref = math.radians((newValue + 90))
    sim.setJointTargetPosition(joint3, q3ref)
    sendNewCoords()
```

```
def onGripperChange(uiHandle, id, newValue):
    global gripref

    if (newValue == 0):
        gripref = math.radians(180)
        sim.setJointTargetPosition(gripper, gripref)
        #open the gripper by moving it's motor
    elif (newValue == 2):
        gripref = math.radians(0)
        sim.setJointTargetPosition(gripper, gripref)
        #close the gripper by moving it's motor
    sendNewCoords()



def sendNewCoords():

    global q1ref
    global q2ref
    global q3ref
    global gripref

    currentCoordinates = [math.degrees(q1ref), math.degrees(q2ref), math.degrees(q3ref), math.degrees(gripref)]
    toWrite = ','.join(str(coord) for coord in currentCoordinates)
    print(toWrite)
    toWrite = toWrite +'\n'

    global ser_com
    ser_com.write(toWrite.encode())


def sysCall_actuation():

    pass

```

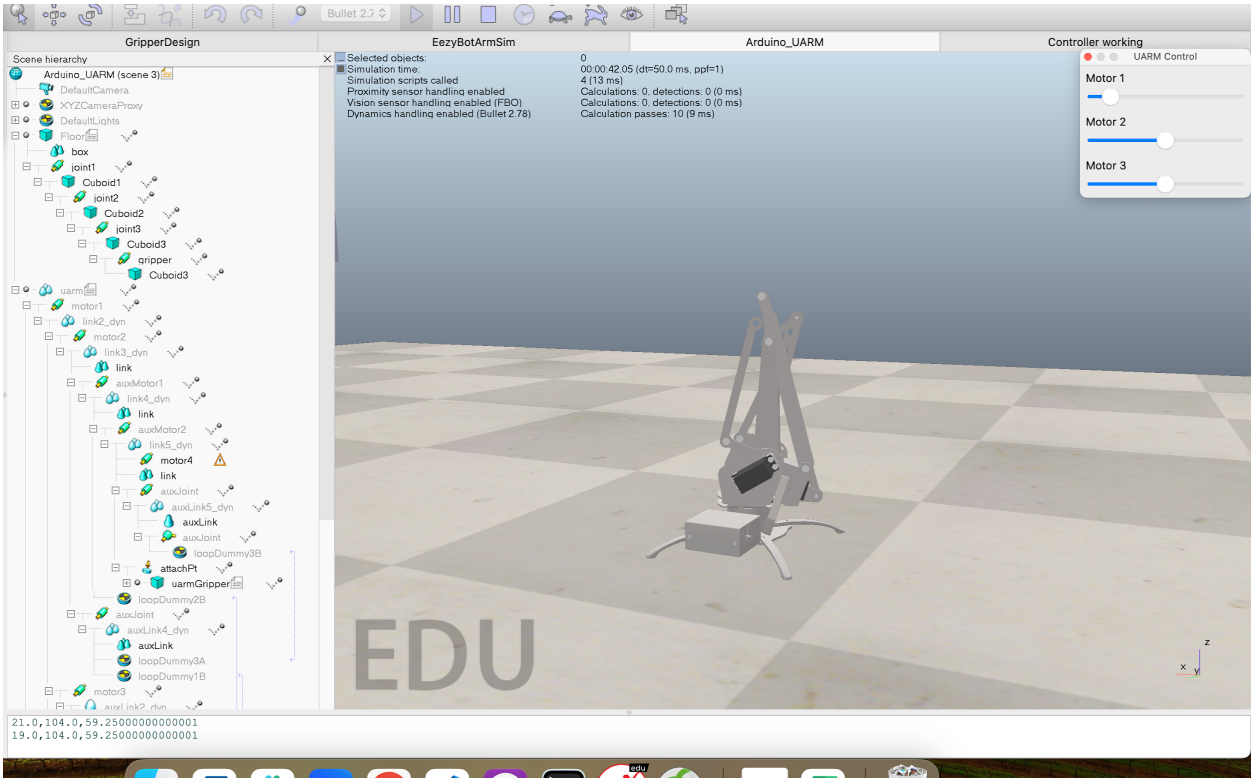
Iteration # 3: This iteration uses the CoppeliaSerial Sketch to communicate between the simulation and Arduino to send the lines of coordinates of each joint as the sliders are moved. This iteration was important as it was the first model that utilized serial communication between the simulation and Arduino to send joint coordinates.



Iteration # 5: A servo is connected to the Arduino and uses the simulation with 4 blocks with the Servo_Test sketch to move the motor in accordance with changes in the simulation. This iteration is important as it was the first test that successfully used the simulation to move a motor accordingly.



Iteration # 6: The UARM model is loaded into CoppeliaSim and sliders are functional but only the base rotation joint functions properly. This iteration is important as it uses a simulated robot similar to the physical robot, but the full functionality of the simulation was hindered.

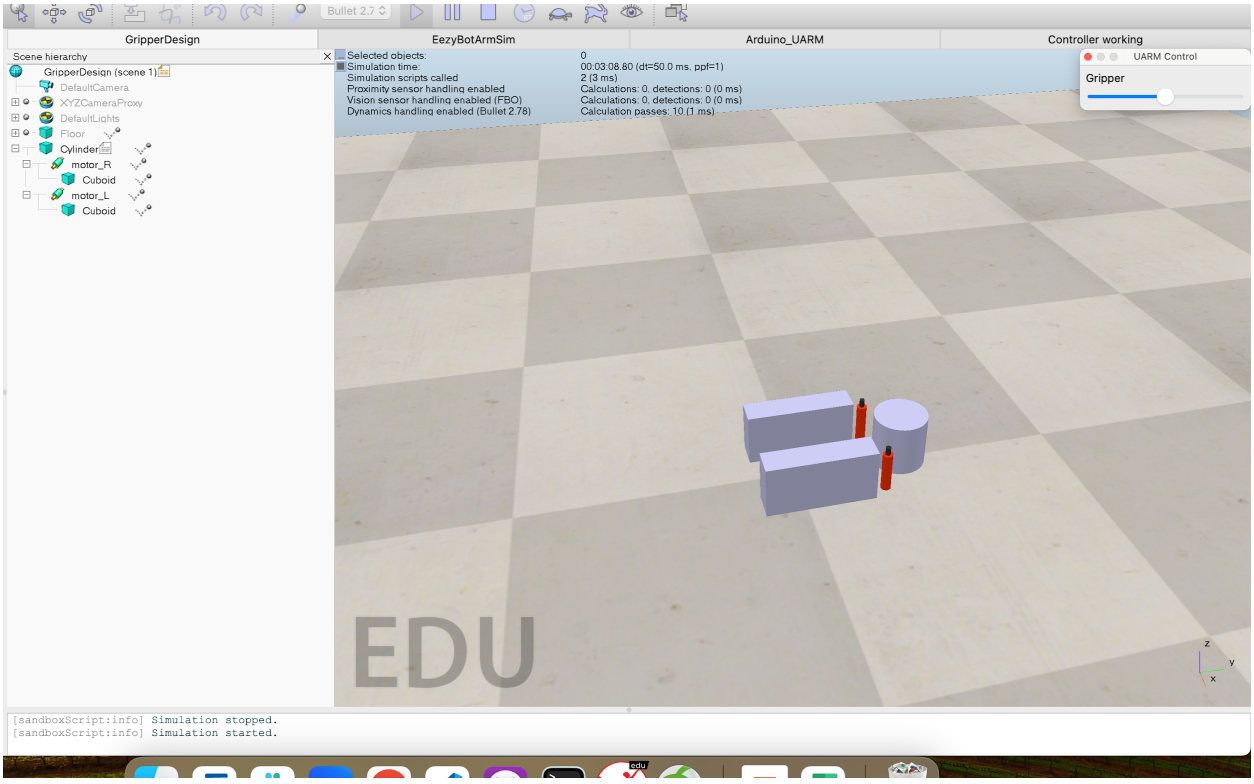



```python
#python

import serial.tools.list_ports
import serial
import time


def sysCall_init():

    findPort()

    global motor1
    motor1 = sim.getObject('/motor1')
    print("Motor1: " + str(motor1))
    global motor2
    motor2 = sim.getObject('/motor2')
    print("Motor2: " + str(motor2))
    global motor3
    motor3 = sim.getObject('/motor3')
    print("Motor3: " + str(motor3))
    #global gripper
    #gripper = sim.getObject(':/motor4')
    #print("gripper: " + str(gripper))

    global m1ref
    m1ref = math.radians(90)
    global m2ref
    m2ref = math.radians(104)
    global m3ref
    m3ref = math.radians(59.25)
    #global gripperRef
    #gripref = math.radians(90)
    sim.setJointTargetPosition(motor1, m1ref)
    sim.setJointTargetPosition(motor2, m2ref)
    sim.setJointTargetPosition(motor3, m3ref)
    #sim.setJointTargetPosition(gripper, gripperRef)


    ui = simUI.create('<ui enabled="true" modal="false" title="UARM Control" closeable="true" layout="vbox" placement="relative" position="20,20">'
    '<label enabled="true" text="Motor 1"></label>'
    '<hslider enabled="true" minimum="-90" maximum="90" on-change="onM1Change"></hslider>'
    '<label enabled="true" text="Motor 2"></label>'
    '<hslider enabled="true" minimum="-90" maximum="90" on-change="onM2Change"></hslider>'
    '<label enabled="true" text="Motor 3"></label>'
    '<hslider enabled="true" minimum="-90" maximum="90" on-change="onM3Change"></hslider>'
    #'<label enabled="true" text="Gripper"></label>'
    #'<checkbox checked="true" text="On/Off" on-change="onGripperChange"/>'
    '</ui>')
```

```
def findPort():
    ports = serial.tools.list_ports.comports()
    global finalPort
    finalPort = ''
    portList = []

    for onePort in ports:
        portList.append(str(onePort))
    for port in portList:
        usb = ""
        for i in range (8,17):
            usb += port[i]
        if usb == "usbserial":
            for i in range (0,22):
                finalPort += port[i]
            break
    print("final port: " + finalPort)

    global ser_com
    ser_com = serial.Serial(finalPort, 9600)
    print("Initializing...")
    time.sleep(1)



def onM1Change(uiHandle, id, newValue):

    global m1ref
    m1ref = math.radians((newValue + 90))
    sim.setJointTargetPosition(motor1, m1ref)
    sendNewCoords()

def onM2Change(uiHandle, id, newValue):

    global m2ref
    m2ref = math.radians((newValue + 90))
    sim.setJointTargetPosition(motor2, m2ref)
    sendNewCoords()

def onM3Change(uiHandle, id, newValue):

    global m3ref
    m3ref = math.radians((newValue + 90))
    sim.setJointTargetPosition(motor3, m3ref)
    sendNewCoords()

def onGripperChange(uiHandle, id, newValue):
    global gripref
```

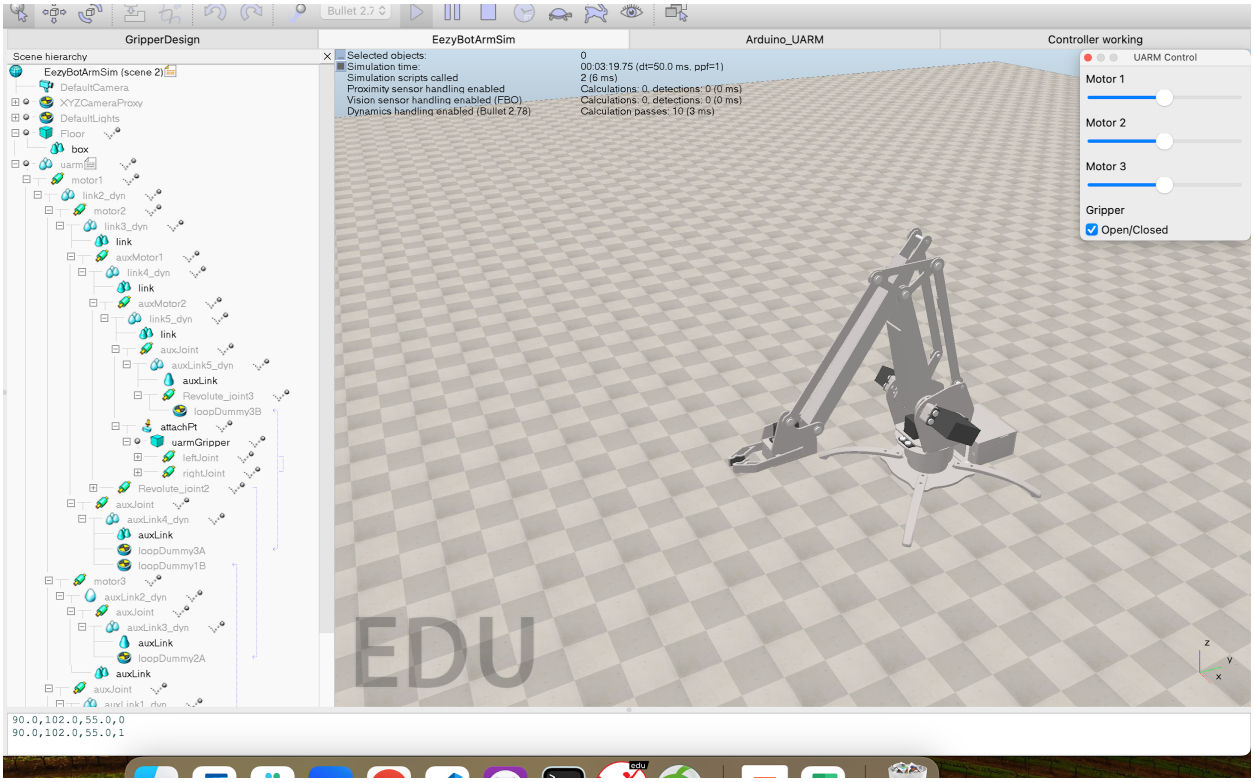
```
def onGripperChange(uiHandle, id, newValue):
    global gripref

    if (newValue == 0):
        #print("opened")
        gripref = math.radians(180)
        sim.setJointTargetPosition(gripper, gripref)
        #open the gripper by moving it's motor
    elif (newValue == 2):
        #print("closed")
        gripref = math.radians(0)
        sim.setJointTargetPosition(gripper, gripref)
        #close the gripper by moving it's motor
    sendNewCoords()



def sendNewCoords():

    global m1ref
    global m2ref
    global m3ref
    #global gripperRef

    currentCoordinates = [math.degrees(m1ref), math.degrees(m2ref), math.degrees(m3ref)]#, math.degrees(gripperRef)]
    toWrite = ','.join(str(coord) for coord in currentCoordinates)
    print(toWrite)
    toWrite = toWrite +'\n'

    global ser_com
    ser_com.write(toWrite.encode())



def sysCall_actuation():

    pass


def sysCall_sensing():
    # put your sensing code here
    pass

def sysCall_cleanup():
    # do some clean-up here
    pass
```

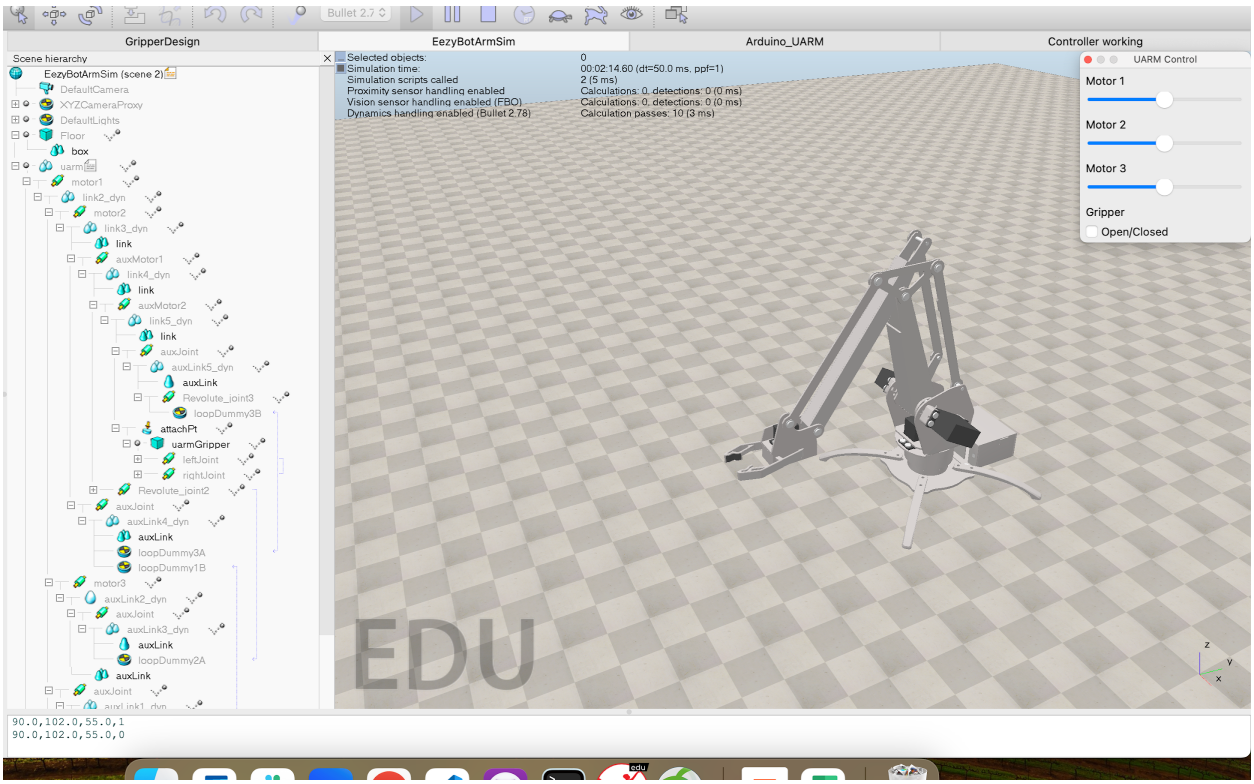
Iteration # 9: A simulation of only a gripper model is functional by controlling the left and right joints of the gripper simultaneously. To this point, the robot simulation was functional except for the gripper. The development of this targeted simulation to achieve gripper functionality allowed for easy implementation in the final stimulation.




```python
#python

def sysCall_init():

    global gripperRight
    gripperRight = sim.getObject('./motor_R')
    global gripperLeft
    gripperLeft = sim.getObject('./motor_L')

    global gripperRef
    gripperRef = math.radians(0)
    sim.setJointTargetPosition(gripperRight, gripperRef)
    sim.setJointTargetPosition(gripperLeft, gripperRef * -1)

    # Creates the UI box with Sliders to control the robot

    ui = simUI.create('<ui enabled="true" modal="false" title="UARM Control" closeable="true" layout="hbox" placement="relative" position="20,20">'
    '<label enabled="true" text="Gripper"></label>'
    '<hslider enabled="true" minimum="-30" maximum="30" on-change="onGripperChange"></hslider>'
    '</ui>')

def onGripperChange(uiHandle, id, newValue):

    global gripperRef
    gripperRef = math.radians(newValue)
    sim.setJointTargetPosition(gripperRight, gripperRef)
    sim.setJointTargetPosition(gripperLeft, gripperRef * -1)
```

Iteration # 12: The robot has smooth motion at all joints and is able to move with the servo motors. This iteration is important as it is the first fully functional iteration of the simulated robot. It works effectively using the slider UI.





```python
#python

import serial.tools.list_ports
import serial
import time


def sysCall_init():

    findPort()

    global motor1
    motor1 = sim.getObject('./motor1')
    global motor2
    motor2 = sim.getObject('./motor2')
    global motor3
    motor3 = sim.getObject('./motor3')
    global gripperRight
    gripperRight = sim.getObject('./rightJoint')
    global gripperLeft
    gripperLeft = sim.getObject('./leftJoint')

    global m1ref
    m1ref = math.radians(90)
    sim.setJointTargetPosition(motor1, m1ref)
    global m2ref
    m2ref = math.radians(90)
    sim.setJointTargetPosition(motor2, m2ref)
    global m3ref
    m3ref = math.radians(90)
    sim.setJointTargetPosition(motor3, m3ref)
    global gripperRef
    gripperRef = math.radians(0)
    sim.setJointTargetPosition(gripperRight, gripperRef)
    sim.setJointTargetPosition(gripperLeft, gripperRef)

    # Creates the UI box with Sliders to control the robot

    ui = simUI.create('<ui enabled="true" modal="false" title="ZMBK Control" closeable="true" layout="vbox" placement="relative" position="20,20">'
    '<label enabled="true" text="Motor 1"></label>'
    '<hslider enabled="true" minimum="-30" maximum="30" on-change="onM1Change"></hsliders>'
    '<label enabled="true" text="Motor 2"></label>'
    '<hslider enabled="true" minimum="-30" maximum="30" on-change="onM2Change"></hsliders>'
    '<label enabled="true" text="Motor 3"></label>'
    '<hslider enabled="true" minimum="-30" maximum="30" on-change="onM3Change"></hsliders>'
    '<label enabled="true" text="Gripper"></label>'
    '<combobox checked="true" text="Open/Closed" on-change="onGripperChange"/>'
    '</ui>')
```

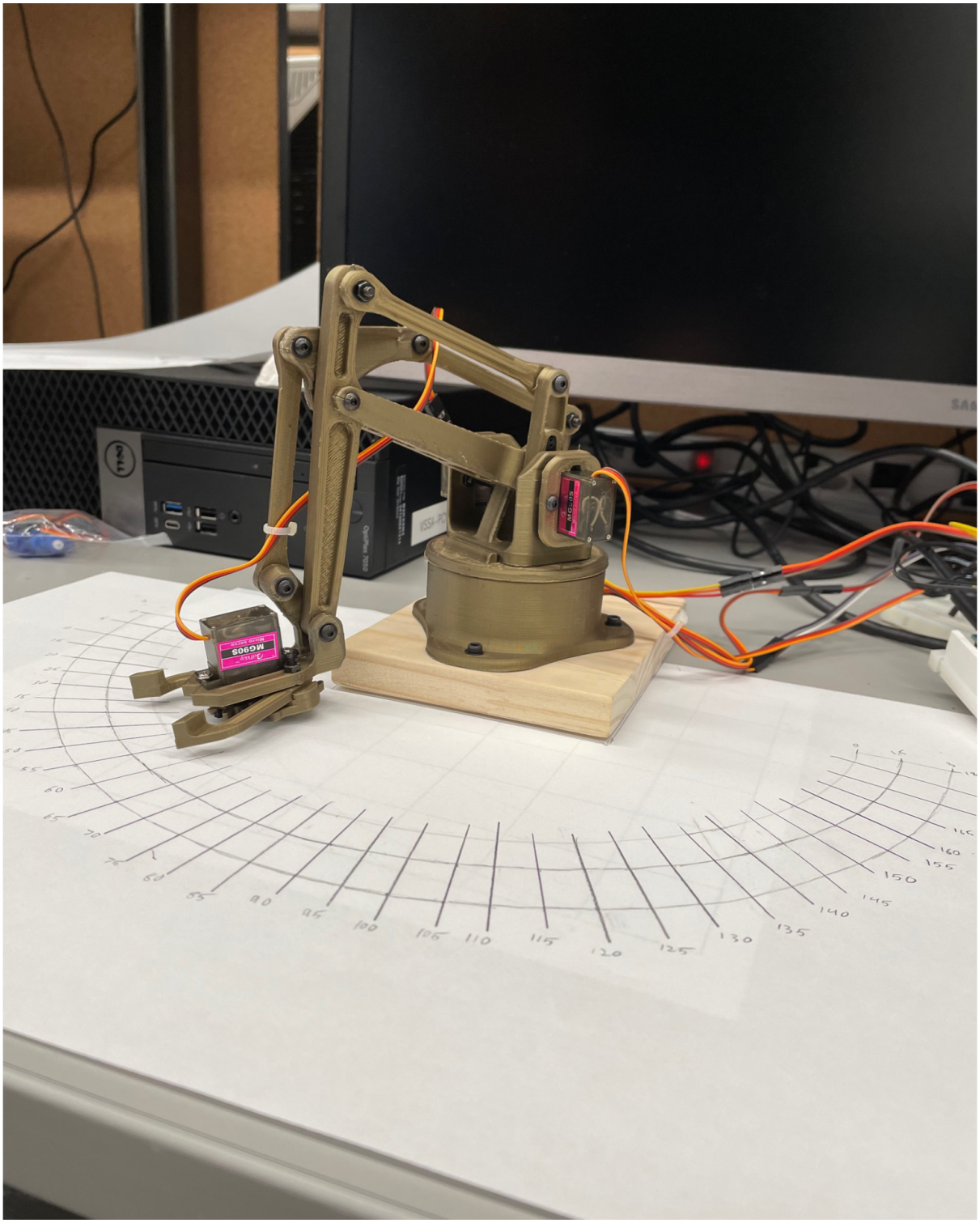
```python
def findPort():
    ports = serial.tools.list_ports.comports()
    global finalPort
    finalPort = ''
    portList = []

    # form a list of ll available ports
    for onePort in ports:
        portList.append(str(onePort))
    # Check to see if any of the available ports match the expected name of an Arduino nano
    for port in portList:
        usb = ""
        for i in range (8,17):
            usb += port[i]
        if usb == "USB-SERIAL":
            for i in range (0,22):
                finalPort += port[i]
            break
    print("com port: " + finalPort)

    # Open the com port with the Arduino
    global ser_com
    ser_com = serial.Serial(finalPort, 9600)




def onM1Change(uiHandle, id, newValue):

    global m1ref
    m1ref = math.radians(newValue + 90)
    sim.setJointTargetPosition(motor1, m1ref)
    sendNewCoords()

def onM2Change(uiHandle, id, newValue):

    global m2ref
    m2ref = math.radians(newValue + 90)
    sim.setJointTargetPosition(motor2, m2ref)
    sendNewCoords()

def onM3Change(uiHandle, id, newValue):

    global m3ref
    m3ref = math.radians(newValue + 90)
    sim.setJointTargetPosition(motor3, m3ref)
    sendNewCoords()
```

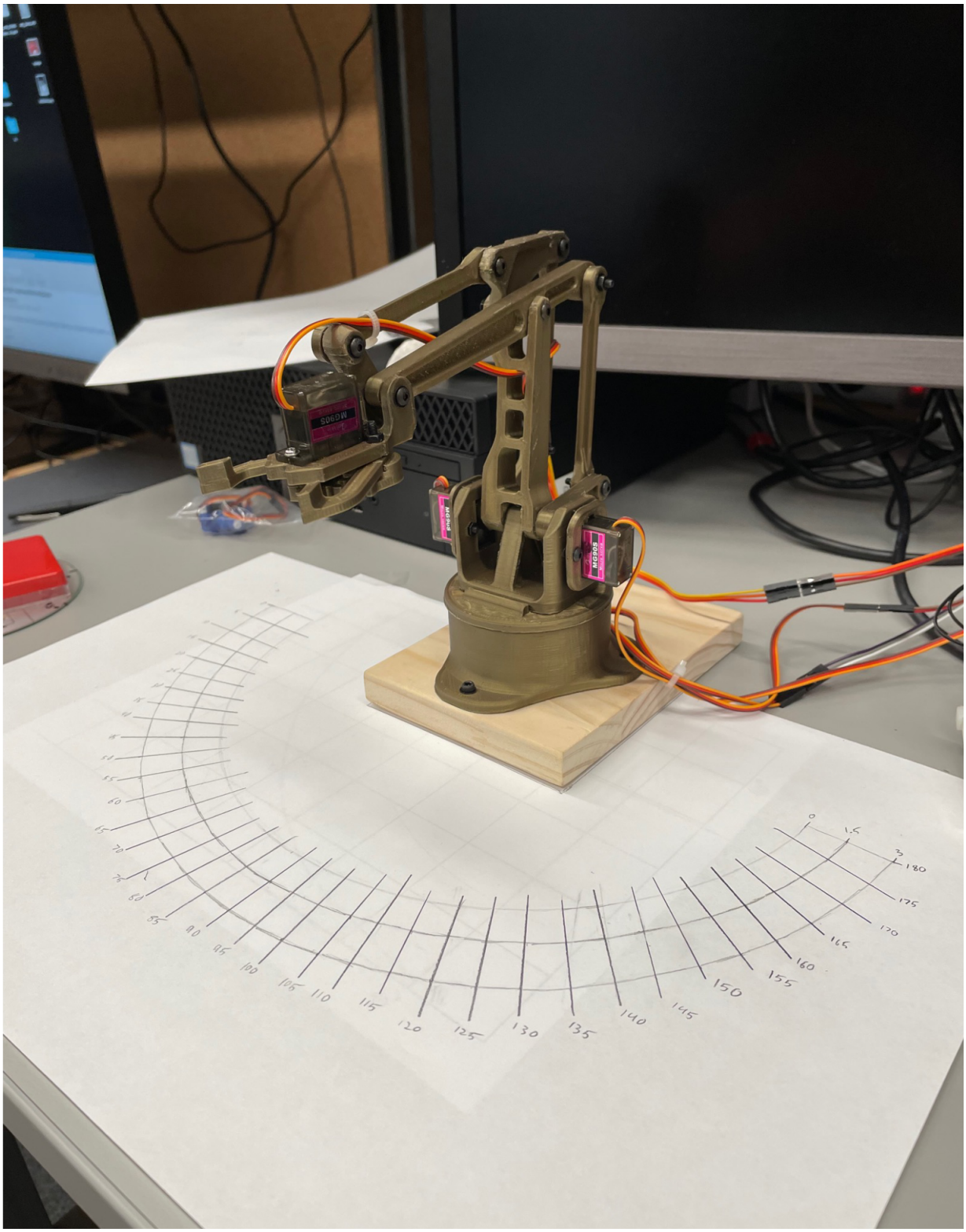
```
def onGripperChange(uiHandle, id, newValue):
    global gripperRef

    if (newValue == 0):
        gripperRef = 0
        sim.setJointTargetPosition(gripperRight, math.radians(0))
        sim.setJointTargetPosition(gripperLeft, math.radians(0))
    elif (newValue == 2):
        gripperRef = 1
        sim.setJointTargetPosition(gripperRight, math.radians(-9))
        sim.setJointTargetPosition(gripperLeft, math.radians(-9))

    sendNewCoords()


def final_map(value, in_min, in_max, out_min1, out_max1, out_min2, out_max2, threshold):
    if value < threshold:
        return map_value(value, in_min, threshold, out_min1, out_max1)
    else:
        return map_value(value, threshold, in_max, out_min2, out_max2)

def map_value(value, in_min, in_max, out_min, out_max):
    return (value - in_min) * (out_max - out_min) // (in_max - in_min) + out_min


def sendNewCoords():

    global m1ref
    global m2ref
    m2send = final_map(math.degrees(m2ref), 0, 180, 180, 90, 90, 0, 104)
    global m3ref
    m3send = final_map(math.degrees(m3ref), 0, 180, 180, 90, 90, 0, 35)
    global gripperRef

    currentCoordinates = [math.degrees(m1ref), m2send, m3send, gripperRef]
    toWrite = ','.join(str(coord) for coord in currentCoordinates)
    print(toWrite)
    toWrite = toWrite +'\n'

    global ser_com
    ser_com.write(toWrite.encode())


def sysCall_actuation():
    # put your repeated code here
    pass
```

Iteration # 15: Both the simulation and the robot function as expected when controlled by the simulation at this point. This iteration is important as it sees the development of a functional Python script in the simulation and a successful Arduino script. The two programs work together effectively to control the robot.

```
59
60
61    if (Serial.available() > 0) {
62      coords = Serial.readStringUntil('\n');
63
64      while (index < coords.length()) { // Iterate through the CSV line
65        int commaIndex = coords.indexOf(',', index);
66
67        if (commaIndex != -1) { // Check if a comma was found
68          // Extract the substring between start and commaIndex
69        String substr = coords.substring(start, commaIndex);
70        commands[j] = substr.toInt();
71        start = commaIndex + 1; // Move the start index to the character after the comma
72
73        } else {
74          // If no comma is found, extract the substring from start to the end of the line
75          String substr = coords.substring(start);
76          commands[j] = substr.toInt();
77          break;  // Exit the loop since we reached the end of the coordinate line
78        }
79        // Move the index to the character after the comma
80        index = commaIndex + 1;
81        j++;
82      }
83    }
84
85
86  }
87
88  void setServos(){
89
90
91    baseServo.write(commands[0]);
92    rightServo.write(commands[1]);
93    leftServo.write(commands[2]);
94
95
96    if (commands[3] == 1) {
97      //Closed
98      gripServo.write(35);
99    } else {
100     //Opened
101     gripServo.write(90);
102   }
103
104 }
105
106 void servoArc(){
107
108   for (int i = 0; i < 181; i += 5){
109     Serial.println(i);
110     baseServo.write(i);
111     delay(4500);
112   }
113
114 }
```

Iteration # 17: The robot can be given the coordinates of an object to which it will move and pick up the object. Then a new set of coordinates can be given to move the object to. This iteration showcases the final functionality of SCRAP as a pick-and-place system.




MG90S
MG90S
0
15
30
180
175
170
165
160
155
150
145
140
135
130
125
120
115
110
105
100
95
90
85
80
75
70
65
60