Isaac Rodriguez-Jimenez

Eco-Robot Project

A robot created entirely from its own packaging

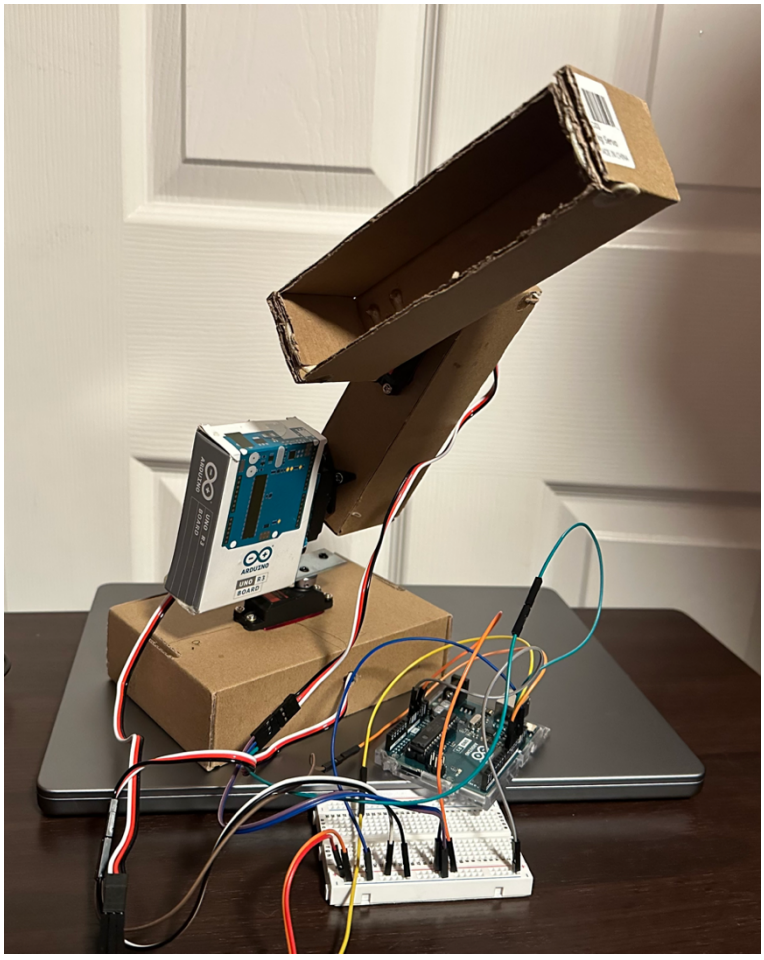
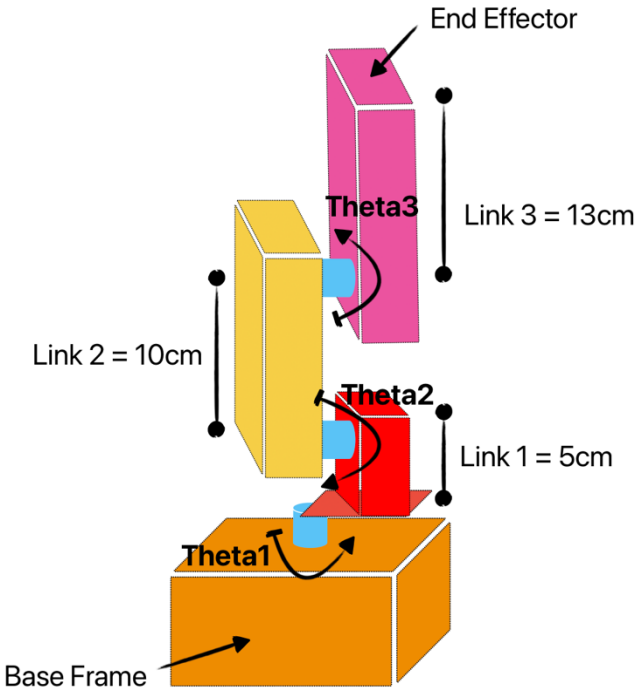


Table of Contents

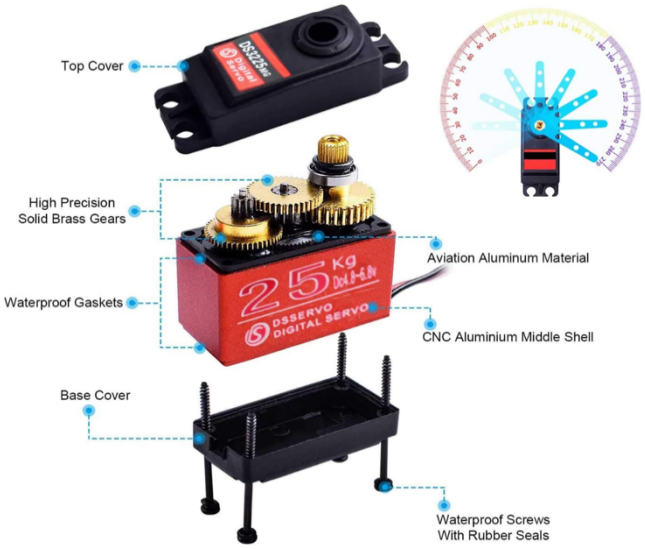
Design Specifications



- In this figure the lengths are described for each link.

- The revolute joints are show as blue cylinders in this figure with their respective direction of rotation.

Universal Digital Servos



These motors are advertised to rotate up to 270 degrees. Under my testing, I found them to work reliably up to 200 degrees. I decided to make their maximum rotation 180 degrees.

This high torque servo is meant to be powered by 4.8 to 6.8 volts. They are pretty strong and capable of easily stripping plastic attachments.

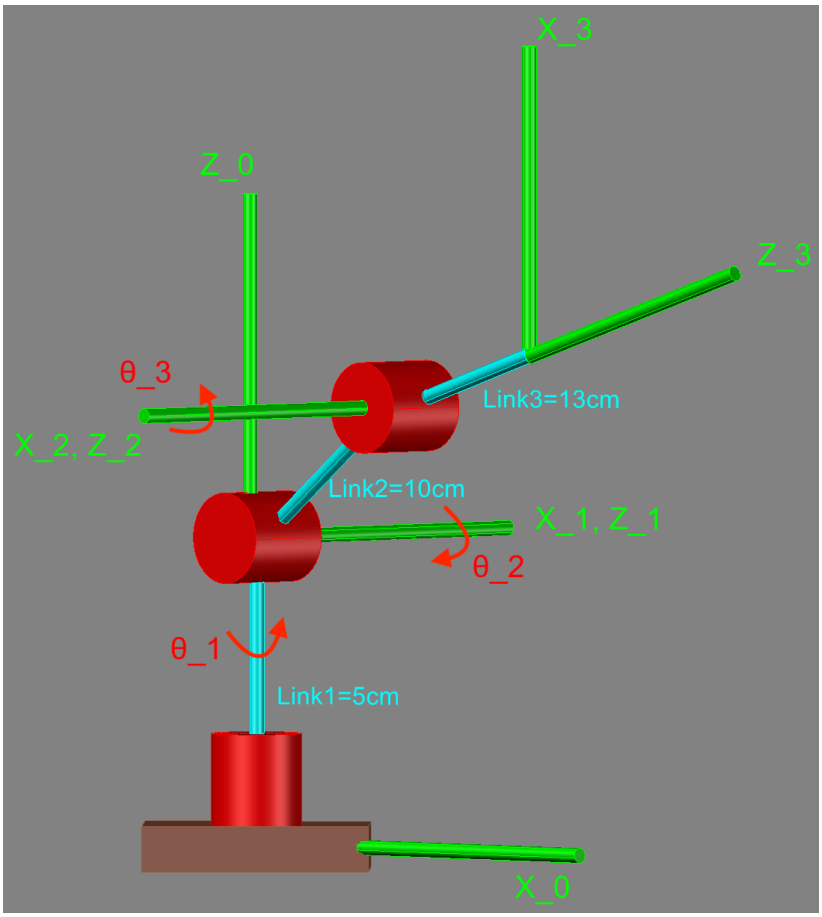
These servos weigh 60 grams and have wet/dry applications due its waterproof gaskets.

Arduino UNO R3

The microcontroller I used was an Arduino UNO R3. This is a very versatile machine, but it does have its draw backs when used in robotics. It simply does not provide enough power to operate multiple motors consistently. If you can work around the power issue, this Arduino would be a good choice. The 6 PWM outputs give the user the capability to control a 6 DOF robot with low power requirements.

Mathematical Models

Forward Kinematic:



This is a simplified model used to calculate the DH parameters. The 4 parameters are used to establish a connection between the reference frames and the robot.

The purpose of the forward kinematics is to determine a set of equations that will determine where the end effector will land after rotating the motors in certain directions.

DH Parameters

Link	θ	d	a	α
1	θ_1*	L_1	0	-90
2	θ_2* - 90	0	L_2	180
3	θ_3* + 180	0	L_3	90

General Forward Kinematic Model:

$$A_i = \begin{matrix} C_{\theta_i} & -S_{\theta_i}C_{\alpha_i} & S_{\theta_i}S_{\alpha_i} & a_iC_{\theta_i} \\ S_{\theta_i} & C_{\theta_i}C_{\alpha_i} & -C_{\theta_i}S_{\alpha_i} & a_iS_{\theta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{matrix}$$

Using the matrix above we find our own values using the DH parameters.
Frame 0 – 1:

$$A_1 = \begin{matrix} Cos_{\theta_1} & 0 & Sin_{\theta_1} & 0 \\ Sin_{\theta_1} & 0 & Cos_{\theta_1} & 0 \\ 0 & -1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Frame 1 – 2:

$$A_2 = \begin{matrix} Cos(\theta_2 - 90) & Sin(\theta_2 - 90) & 0 & 10Cos(\theta_2 - 90) \\ Sin(\theta_2 - 90) & -Cos(\theta_2 - 90) & 0 & 10Sin(\theta_2 - 90) \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Frame 2 – 3:

$$A_3 = \begin{matrix} Cos(\theta_3 + 180) & Sin(\theta_3 + 180) & 0 & 13Cos(\theta_3 + 180) \\ Sin(\theta_3 + 180) & -Cos(\theta_3 + 180) & 0 & 13Sin(\theta_3 + 180) \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

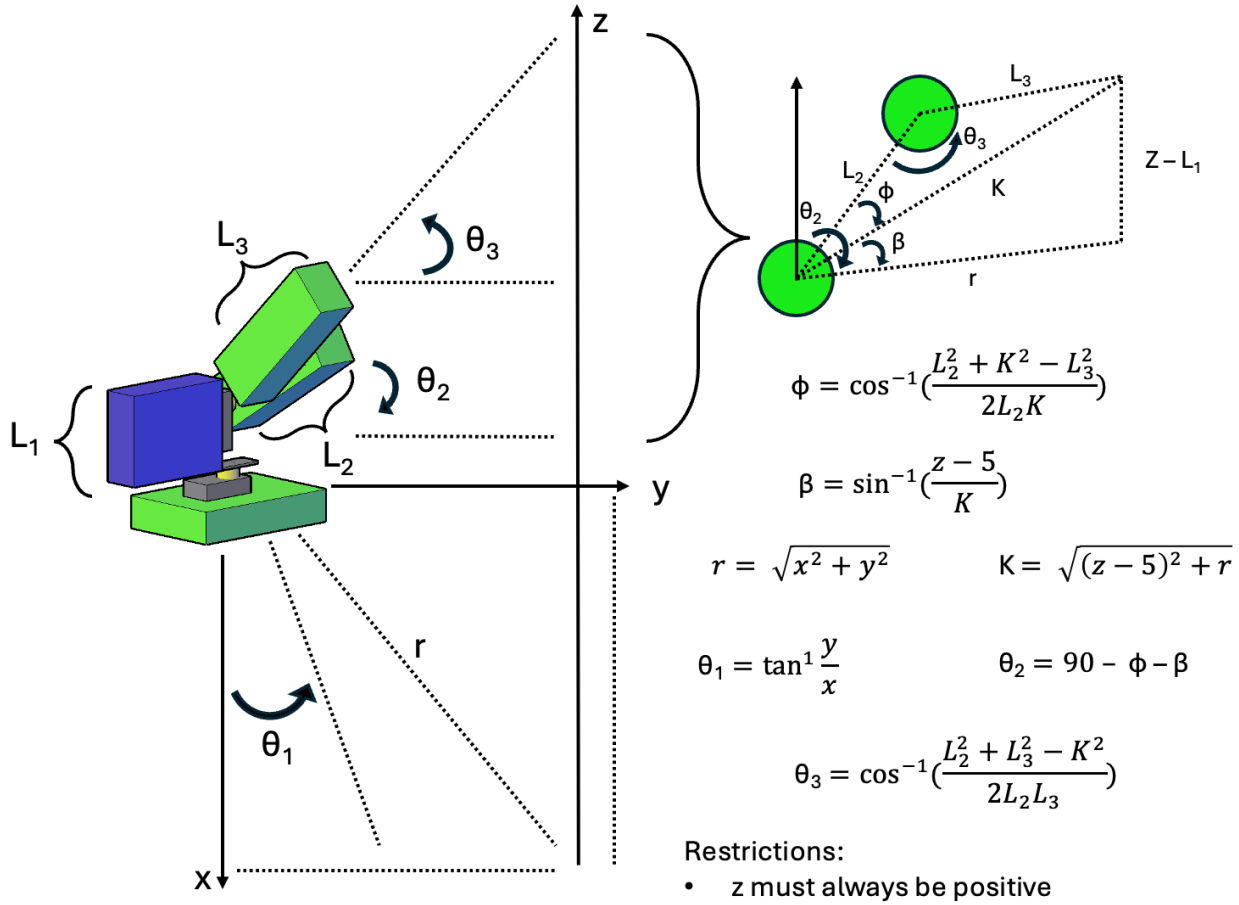
Homogeneous Transformation using MATLAB

$H = A_1A_2A_3$ Frame 0 to 3.
dx, dy, and dz are directly calculated using MATLAB in order to determine X, Y, and Z position coordinates of the end effector using only the rotation of the servo motors.

```
dx =

10*cos((pi*theta1)/180)*cos((pi*(theta2 - 90))/180) +
 13*cos((pi*theta1)/180)*cos((pi*(theta2 - 90))/180)*cos((pi*(theta3
 + 180))/180) + 13*cos((pi*theta1)/180)*sin((pi*(theta2 -
 90))/180)*sin((pi*(theta3 + 180))/180)


dy =

10*sin((pi*theta1)/180)*cos((pi*(theta2 - 90))/180) +
 13*sin((pi*theta1)/180)*cos((pi*(theta2 - 90))/180)*cos((pi*(theta3
 + 180))/180) + 13*sin((pi*theta1)/180)*sin((pi*(theta2 -
 90))/180)*sin((pi*(theta3 + 180))/180)


dz =

13*cos((pi*(theta2 - 90))/180)*sin((pi*(theta3 + 180))/180)
 - 10*sin((pi*(theta2 - 90))/180) - 13*cos((pi*(theta3 +
 180))/180)*sin((pi*(theta2 - 90))/180) + 5
```

Inverse Kinematics:

These formulas describe how to extract the 3 rotational angles needed to reach a point in the coordinate plane.



$$\phi = \cos^{-1}\left(\frac{L_2^2 + K^2 - L_3^2}{2L_2K}\right)$$

$$\beta = \sin^{-1}\left(\frac{z-5}{K}\right)$$

$$r = \sqrt{x^2 + y^2} \qquad K = \sqrt{(z-5)^2 + r}$$

$$\theta_1 = \tan^1\frac{y}{x} \qquad \theta_2 = 90 - \phi - \beta$$

$$\theta_3 = \cos^{-1}\left(\frac{L_2^2 + L_3^2 - K^2}{2L_2L_3}\right)$$

Restrictions:

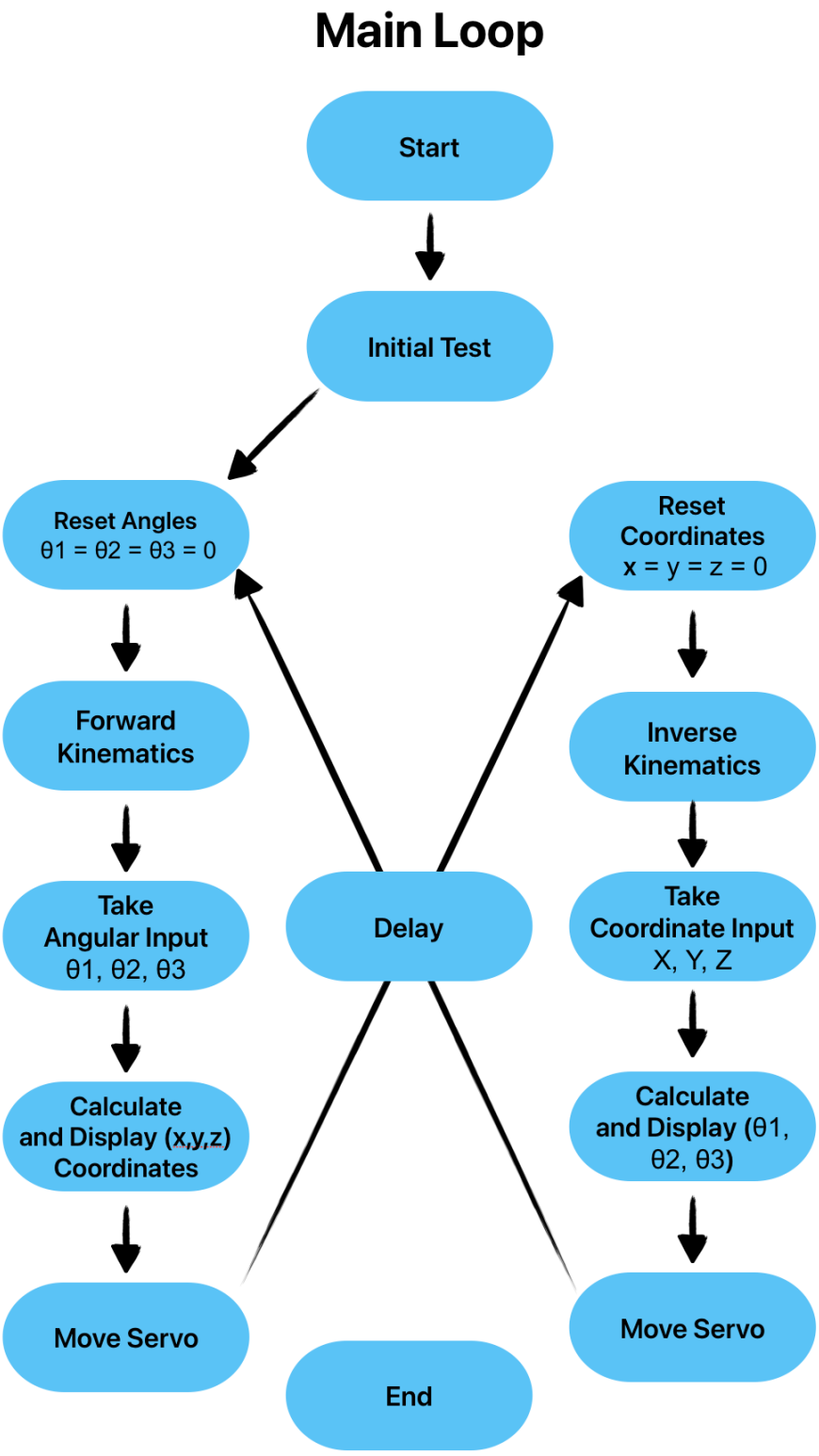
- z must always be positive
- θ_3 must be between 0° & 180°

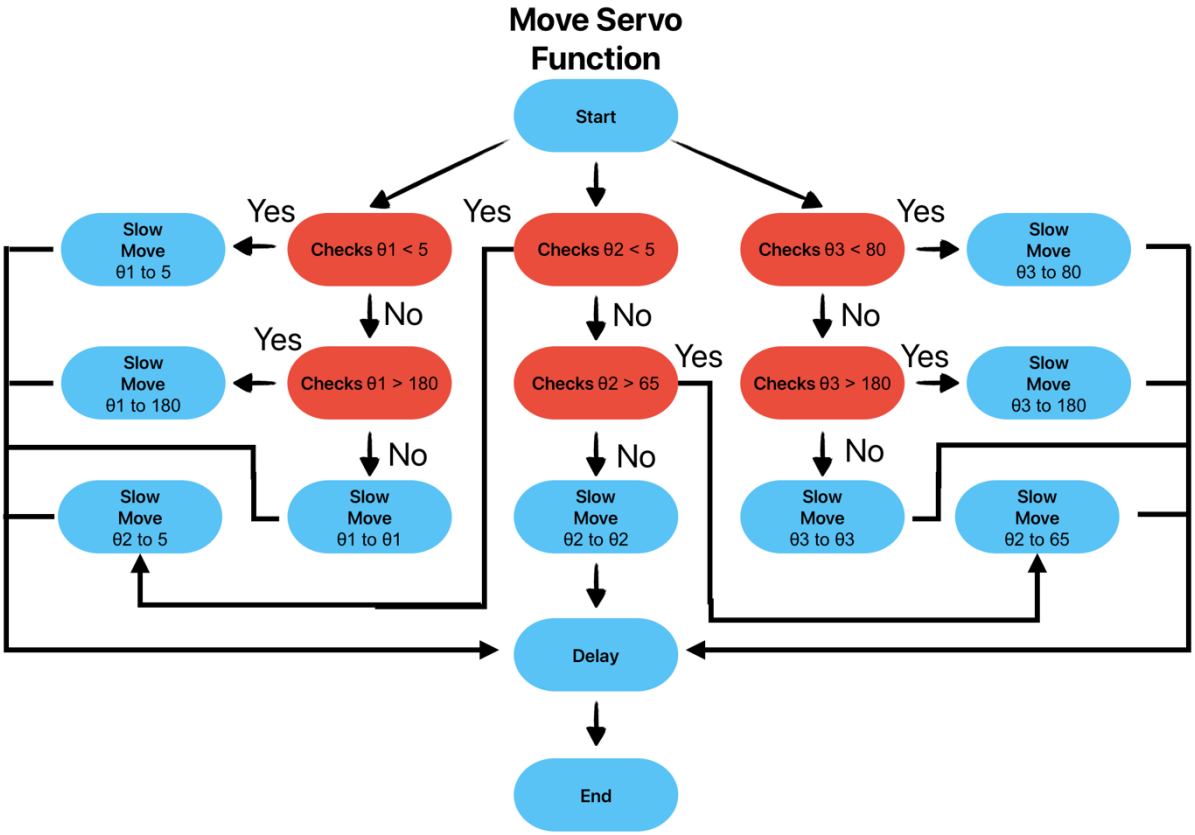
L_1 = 5cm

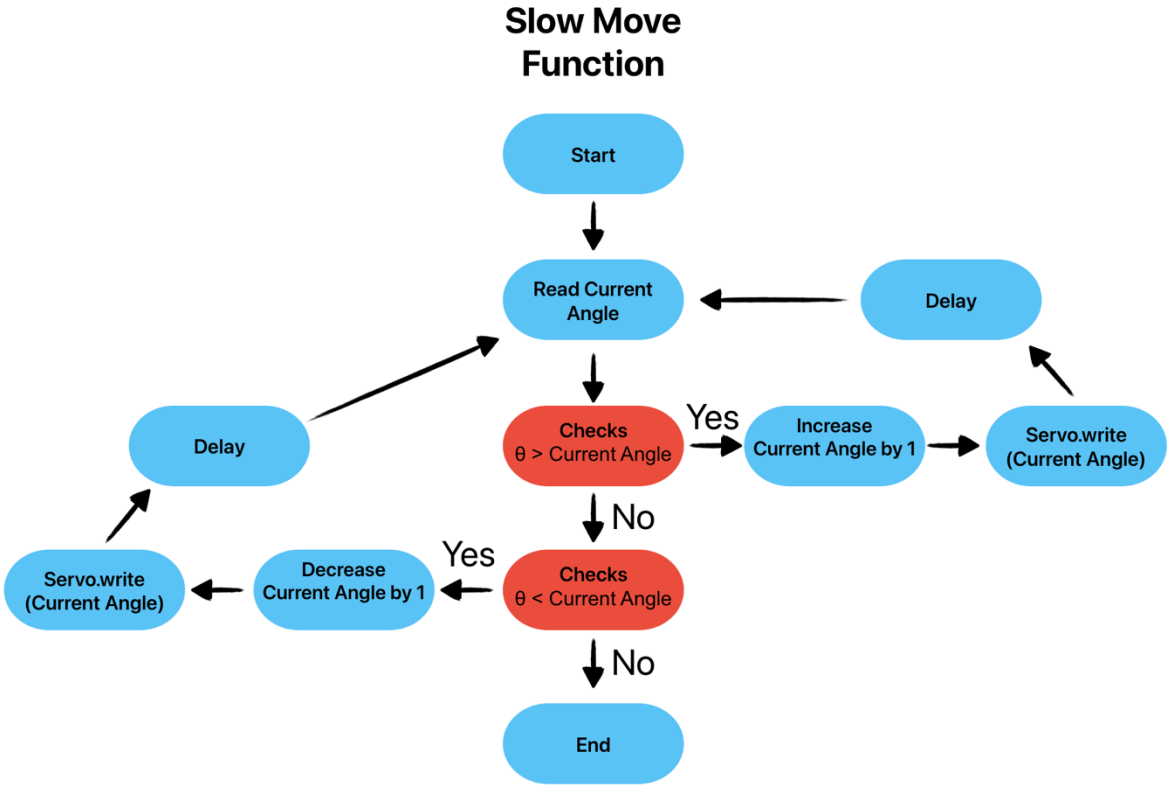
L_2 = 10cm

L_3 = 13cm

Program Flow Chart




Move Servo Function
Start
Checks θ1 < 5
Yes
Slow Move θ1 to 5
No
Checks θ1 > 180
Yes
Slow Move θ1 to 180
No
Slow Move θ2 to 5
Slow Move θ1 to θ1
Checks θ2 < 5
Yes
No
Checks θ2 > 65
Yes
No
Slow Move θ2 to θ2
Checks θ3 < 80
Yes
Slow Move θ3 to 80
No
Checks θ3 > 180
Yes
Slow Move θ3 to 180
No
Slow Move θ3 to θ3
Slow Move θ2 to 65
Delay
End


Slow Move Function
Start
Read Current Angle
Delay
Checks θ > Current Angle
Yes
Increase Current Angle by 1
Servo.write (Current Angle)
Delay
No
Checks θ < Current Angle
Yes
Decrease Current Angle by 1
Servo.write (Current Angle)
No
End

Results and Discussion

The following are two tables are tests that show the output of the software with the given input.

Forward Kinematics

INPUT			OUTPUT		
θ_1	θ_2	θ_3	X	Y	Z
5	5	180	2.06	0.18	27.91
90	60	100	0.01	16.99	0.02
5	65	150	21.93	1.92	8.05
150	45	100	-15.3	8.87	4.58
100	30	150	-2.81	16.04	20.12

Inverse Kinematics

INPUT			OUTPUT		
X	Y	Z	θ_1	θ_2	θ_3
10	10	1	45.01	45.78	78.28
10	10	10	45.02	11.82	80.30
15	1	15	3.82	11.70	102.72
0.5	15	10	88.14	16.49	85.91
-13	5	10	159.04	10.66	78.95

I have found that the forward kinematics to be very accurate. The inverse kinematics worked but my θ_2 had some error. When I would get close to the extremities of its operational limits, the error would increase. Usually, the error was pretty low, but I would consider it acceptable.

Conclusion

This project was a great exercise in order to realize how your code has real implications. If not careful you can accidentally destroy your robot. It is important to carefully design the robot parameters and derive the forward and inverse kinematics in order to avoid collisions with its self or the ground.

I used the left-over boxes where the motors and Arduino came in. From here I assembled my robot by cutting the boxes and gluing certain parts together. It was difficult to come up with the kinematics equations since my 2nd motor and 3rd motors moved in different directions. Looking at my flowchart you can see that my software has three different levels. The main loop that

never ends and always prompts you for an angle or coordinate component. The move servo function that makes sure the robot doesn't crash into anything and the slow move function that actually takes care of moving the motors in a slower fashion. Each of these functions not only considers the software side but also considers the servos movement speed and angular accuracy.

Appendix

Arduino Code

```
#include <Servo.h>

Servo base_servo;
Servo shoulder_servo;
Servo elbow_servo;

int base_pin = 9;
int shoulder_pin = 10;
int elbow_pin = 11;

int low = -30;
int high = 30;

int pos = 0;
float base_angle = 60, shoulder_angle = 60, elbow_angle = 200;
float x_coor = 0, y_coor = 0, z_coor = 0, new_z_coor;
double theta1 = 0, theta2 = 0, theta3 = 0;
float r = 0, k = 0, phii = 0, gamm = 0;
//arm lengths in cm
float l1 = 5, l2 = 10, l3 = 13;

float x = 0, y = 0, z = 0;
float t1 = 0, t2 = 0, t3 = 0;

float pii = 3.14;
```

```
void setup() {
  base_servo.attach(base_pin);
  shoulder_servo.attach(shoulder_pin);
  elbow_servo.attach(elbow_pin);
  Serial.begin(9600);

  Serial.println("Performing Initial Tests. Moving to Home
Position!");
  moveservos(base_angle, shoulder_angle, elbow_angle);
  Serial.println("Ready to Take Commands!");
  delay(3000);
}



void loop() {

  //Angle limits
  //base_angle_min = 30, base_angle_max = 120;
  //shoulder_angle_min = 30, base_angle_max = 90;
  //shoulder_angle_min = -60, base_angle_max = 0;


  base_angle = 0;
  shoulder_angle = 0;
  elbow_angle = 0;

  // Take angle input:
  Serial.print("Input Base Angle (5 to 180):");
  while (base_angle == 0) {
    base_angle = Serial.parseFloat();
  }
  Serial.println(base_angle);

  Serial.print("Input Shoulder Angle (5 to 65):");
  while (shoulder_angle == 0) {
```

```
    shoulder_angle = Serial.parseFloat();
  }
  Serial.println(shoulder_angle);

  Serial.print("Input Elbow Angle (80 to 180):");
  while (elbow_angle == 0) {
    elbow_angle = Serial.parseFloat();
  }
  Serial.println(elbow_angle);
  t1 = base_angle * (pii / 180);
  t2 = shoulder_angle * (pii / 180);
  t3 = elbow_angle * (pii / 180);

  //Coordinates - came from matlab (dx, dy, and dz)

  x = 10 * cos(t1) * cos(t2 - (pii / 2)) + 13 * cos(t1) * cos(t2
- (pii / 2)) * cos(t3 + pii) + 13 * cos(t1) * sin(t2 - (pii /
2)) * sin(t3 + pii);

  y = 10 * sin(t1) * cos(t2 - (pii / 2)) + 13 * sin(t1) * cos(t2
- (pii / 2)) * cos(t3 + pii) + 13 * sin(t1) * sin(t2 - (pii /
2)) * sin(t3 + pii);

  z = 13 * cos(t2 - (pii / 2)) * sin(t3 + pii) - 10 * sin(t2 -
(pii / 2)) - 13 * cos(t3 + pii) * sin(t2 - (pii / 2)) + 5;


  Serial.println("Angles (degrees):");
  Serial.print("(");
  Serial.print(base_angle);
  Serial.print(",");
  Serial.print(shoulder_angle);
  Serial.print(",");
  Serial.print(elbow_angle);
  Serial.println(")");
```

```
  Serial.println("Coordinates (cm):");
  Serial.print("(");
  Serial.print(x);
  Serial.print(",");
  Serial.print(y);
  Serial.print(",");
  Serial.print(z);
  Serial.println(")");

  moveservos(base_angle, shoulder_angle, elbow_angle);

  delay(3000);

  // write code for inverse kinematics
  x_coor = 0;
  y_coor = 0;
  z_coor = 0;
  // enter your coordinates:
  Serial.print("Input X coordinate (0 to ?):");
  while (x_coor == 0) {
    x_coor = Serial.parseFloat();
  }
  Serial.println(x_coor);

  Serial.print("Input Y coordinate (0 to ?):");
  while (y_coor == 0) {
    y_coor = Serial.parseFloat();
  }
  Serial.println(y_coor);

  Serial.print("Input Z coordinate (0 to ?):");
  while (z_coor == 0) {
    z_coor = Serial.parseFloat();
  }
  Serial.println(z_coor);
```

```
// Inverse Kinematics from Matlab
  new_z_coor = z_coor - 5;
  r = sqrt((x_coor * x_coor) + (y_coor * y_coor));
  k = sqrt((new_z_coor * new_z_coor) + (r*r));

  theta1 = atan2(y_coor,x_coor) * 180 / pii;

  phii = acos(((l2 * l2) + (k * k) - (l3 * l3)) / (2 * l2 * k))
* 180 / pii;
  gamm = asin(new_z_coor / k) * 180 / pii;

  theta2 = 90 - phii - gamm;

  theta3 = acos(((l2 * l2) + (l3 * l3) - (k * k)) / (2 * l2 *
l3)) * 180 / pii;


  Serial.println("Coordinates (cm):");
  Serial.print("(");
  Serial.print(x_coor);
  Serial.print(",");
  Serial.print(y_coor);
  Serial.print(",");
  Serial.print(z_coor);
  Serial.println(")");


  Serial.println("Angles (degrees):");
  Serial.print("(");
  Serial.print(theta1);
  Serial.print(",");
  Serial.print(theta2);
  Serial.print(",");
  Serial.print(theta3);
  Serial.println(")");
```

```
  moveservos(theta1, theta2, theta3);

  delay(3000);
}


// function to move servos

void moveservos(float angle1, float angle2, float angle3) {

  if (angle1 < 5) {
    //base_servo.write(20);
    slowmove(base_servo, 5 * 0.72);
    Serial.println("Base angle too small moving to 5!");
  } else if (angle1 > 180) {
    //base_servo.write(110);
    slowmove(base_servo, 180 * 0.72);
    Serial.println("Base angel too large moving to 180!");
  } else {
    //base_servo.write(angle1);
    slowmove(base_servo, angle1 * 0.72);
    Serial.println("Base Servo moved to assigned angle");
  }
  delay(500);

  if (angle2 < 5) {
    //shoulder_servo.write(25);
    slowmove(shoulder_servo, 5 * 0.72);
    Serial.println("Shoulder angle too small moving to 5!");
  } else if (angle2 > 65) {
    //shoulder_servo.write(85);
    slowmove(shoulder_servo, 65 * 0.72);
    Serial.println("Shoulder angel too large moving to 65!");
  } else {
    //shoulder_servo.write(angle2);
    slowmove(shoulder_servo, angle2 * 0.72);
```

```
    Serial.println("Shoulder Servo moved to assigned angle");
  }
  delay(500);
  if (angle3 < 80) {
    //-elbow_servo.write(20);
    slowmove(elbow_servo, 80 * 0.72);
    Serial.println("Elbow angle too small moving to 80!");
  } else if (angle3 > 180) {
    //elbow_servo.write(80);
    slowmove(elbow_servo, 180 * 0.72);
    Serial.println("Elbow angle too large moving to 180!");
  } else {
    //elbow_servo.write(angle3);
    slowmove(elbow_servo, angle3 * 0.82);
    Serial.println("Elbow Servo moved to assigned angle");
  }
  delay(500);
}

void slowmove(Servo servo, float angle) {
  float curangle = servo.read();

  if (angle > curangle) {
    while (angle > curangle) {
      curangle++;
      servo.write(curangle);
      curangle = servo.read();
      delay(25);
    }
  } else if (angle < curangle) {

    while (angle < curangle) {
      curangle--;
      servo.write(curangle);
      curangle = servo.read();
      delay(25);
```

```
    }
  }
}
```