Pick and Place Robot Arm

Hongyi Chen (Dropped)
School of Engineering
Worcester Polytechnic Institute
Pittsburgh, USA
hchen8@wpi.edu

Kathia Coronado
School of Engineering
Worcester Polytechnic Institute
Alexandria, VA
kcoronado@wpi.com

Taqi Hussain
School of Engineering
Worcester Polytechnic Institute
New York, USA
thussain@wpi.edu

Josephine Leingang
School of Engineering
Worcester Polytechnic Institute
San Jose, CA
jtleingang@wpi.edu

***Abstract*— The ability to pick something up and transport it to a specific location has long been a human skill used primarily in manufacturing, but also other industries as well. In recent years this repetitive task has been automated using robots. Humans fatigue easily during repetitive tasks, are more prone to make mistakes, and slow down rate of production; therefore, we build robots to perform pick and place operations. The project deals with creating a simulation of a robot arm going through the motions of picking up an object and placing it at a target location. This simulation will be made using MATLAB. The robot has five degrees of freedom and will be built and tested in a real-world environment using an Arduino.**

Keywords—Robot Kinematics, Inverse Kinematics, Trajectory Planning

I. Introduction

History has shown that humans are not very efficient workers when it comes to completing highly repetitive tasks. Robots work non-stop with high efficiency which is why there is an enormous need for them in industry. Pick and place robot arms can be used in assembly, packaging, bin-picking, and inspection; this diversity allows almost all industries to benefit greatly from them.

A pick and place manipulator alone is not the most versatile robot, but because of its extreme efficiency at certain highly in demand tasks it is immensely important to society at large [1].

Simple pick and place robots with only one attribute are not too complicated, however future companies need flexible robots that can work in many situations and have multiple attributes. Many features can be added to this robot to make it more flexible and increase its productivity such as, obstacle detection, mobility, material detection, increased precision etc. [3].

Some of these features may increase the design complexity of the system; however, most of them could be handled through software engineering which is much cheaper.

Most of the current pick and place manipulators used in industry do not require high precision because they mostly just automate repetitive tasks. As technology increases, the need for human supervision decreases which is why there is a demand for manipulators with multiple features. This system will pick up an object from a specified location and place it in a target location.

II. Overview of Model

A. Symbols

The parameters of importance are:

- p_d: Robot arm end effector position
- *q:* Robot arm joint variables
- *J:* Jacobian matric in forward velocity kinematics
- *f:* Homogeneous transform function matrix in forward kinematics

B. Inverse Kinematics

For a general *n* degree-of-freedom open chain with forward kinematics *f(q)*, the inverse kinematics problem can be stated as follows: given a homogeneous transform function *f*, find solutions q that satisfy $p_d = f(q)$. Here, we adopt numerical inverse kinematics to solve this equation [2].

$$p_d = f(q)\ ) \qquad [1]$$

$$= f(q_0) + \frac{\partial f}{\partial q}(q - q_0)$$

$$= f(q_0) + J * (q - q_0)$$

$$\Delta q = J^{-1} * (p_d - f(q_0)) \qquad [2]$$

The above equation suggests using an iterative algorithm for finding *q*:

1) Initialization: Given p_d and an initial guess q_0, set *i = 0.*
2) Set $e = p_d - f(q^i)$, while ‖e‖ > ε for some small ε:
- Set $q^{i+1} = q^i + J^{-1} * e$.
- Increment i.

C. Trajectory Planning

During robot motion, the robot controller is provided with a steady stream of goal positions and velocities to track. This

specification of the robot position as a function of time is called a trajectory. The simplest type of motion is from rest at one configuration to rest at another which is called point-to-point motion [2]. Here we will do a time scaling *q(t)* of a path that ensures the motion is appropriately smooth and satisfies any constraints on the robot velocity and acceleration.

A convenient form for the time scaling *q(t)* is a cubic polynomial of time,

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

A point-to-point motion in time T imposes the initial constraints on $q(0)$ and $\dot{q}(0)$ and the terminal constraints $q(T)$ and $\dot{q}(T)$. Evaluating equation and its derivative, we can solve the four constraints for a_0, a_1, a_2, a_3. In this way, we generate a joint space path from start position to end position.

III. Simulation

A. Primary Simulation

Using MATLAB, we created a simulation displaying the trajectory/motion of a pick and place manipulator moving from the home position of our experimental robot to a specified target position.

We solved for the Forward Kinematics of the robot to get the position of the end-effector with respect to the base frame. To do this first created a drawing representation of the 5-DOF experimental robot arm in its home position, as shown in Figure 1. This allowed us to visualize the joints in relation to the base frame. Figure 1 shows the location of each joint as well as the distances between joints.

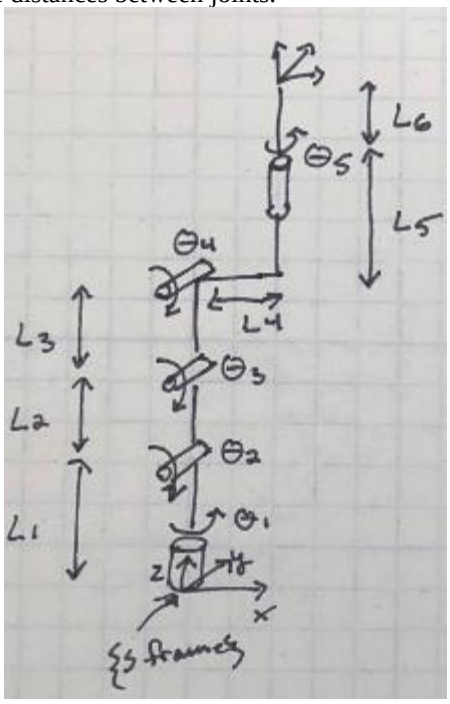

Figure 1: Robot Diagram

This then allowed the us to create our homogeneous transformation matrix "M" of the position and orientation of the end effector frame with respect to the base frame. Matrix "M" is shown in Eqn 1. Then the joint velocity vectors and rotation vectors were also created for each joint, shown in Table 1.

$$M = \begin{bmatrix} 1 & 0 & 0 & L4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L1 + L2 + L3 + L5 + L6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [4]$$

Table 1: Rotation and Velocity Vectors of Each Joint

i	ω_i	v_i
1	[0; 0; 1]	[0; 0; 0]
2	[0; 1; 0]	[-L1; 0; 0]
3	[0; 1; 0]	[-L1-L2; 0; 0]
4	[0; 1; 0]	[-L1-L2-L3; 0; 0]
5	[0; 0; 1]	[0; -L4; 0]

The rotation skew symmetric matrix and velocity vectors of each joint were then used to populate homogenous transformation matrices for joints 1-5. Equations 5-7 were created.

$$ES(n) = e^{[S]\theta} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \quad [5]$$

$$R = I + sin\theta[\hat{\omega}] + (1 - cos\theta)[\hat{\omega}]^2 \quad [6]$$

$$p = (I\theta + (1 - cos\theta)[\omega] + (\theta - sin\theta)[\omega]^2)v \quad [7]$$

The forward kinematics of the end effector with respect to the base frame was calculated using Eqn 8.

$$FK = ES1 * ES2 * ES3 * ES4 * ES5 * M \quad [8]$$

Then the Jacobian was found by taking combining the Jv and Jw components. To find the 3x5 Jv matrix the partial derivative of the forward kinematics position vector was taken with respect to each joint variable. The find the 3x5 Jw matrix, the 3x1 z column of each joint variable's rotation matrix, R shown in Equation 9. Jv and Jw were combined to create the 6x5 Jacobean matrix.

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \quad [9]$$

We then prompted the user in MATLAB for the pick and place locations as well as the time to complete the entire pick/place operation; however, we also provided our own example of the desired positions and hard-coded it into MATLAB for ease of use. Figure 1 shows the pick and place locations for the simulation example shown later.

```
%Define desired positions
%      [ X     Y    Z       Time ]
xd = [ 50      0   400       0;
       200     0    10       5;
       200     0   100     7.5;
       0    -200   100    12.5;
       50      0   400      15 ];
```

Figure 2: Desired Pick and Place Locations

We then create a for loop to go through each pick and place location and plot the movement of the experimental 5-DOF robotic arm.

This for loop goes through the desired positions matrix two rows at a time. The top row is assigned as the initial position and the second is the final position. For each position the x, y and z coordinates and time values are taken from the desired position matrix, Figure 2.

The joint variables for the initial and final locations are then calculated. A while loop creates an initial guess for the joint variable values and substitutes them into the Jacobian J. The while loop checks the error amount between the desired x, y and z coordinates and the actual. It then adds a delta value to the actual joint variables and iterates through the loop until the error amount between the desired and actual position is below 0.1. The joint variables for the initial and final desired positions are now calculated.

To plan the motion of the robot arm, trajectory planning was used. Trajectory describes the position of an object as a function of time. The goal of trajectory planning is to move an object from one point to another while avoiding collisions over a specific amount of time. Trajectory planning generates a path between the initial and final position using a set of equations. The position, velocity and acceleration of the joint variables are defined as functions of time. The cubic polynomial equation for the position of the joint variables as a function of time is shown in Equation 10. Solving for this equation and its derivative for both the final and initial positions and velocities, then defines values for a0, a1, a2, and a3.

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \qquad [10]$$

The position, velocity and acceleration of the joint variables is defined as a function of time and can be plotted. A matrix to hold the initial and end time values (in seconds) as well as the iteration value is then created. A for loop iterates through the time and calculates the joint variables for that instance. Then the joint variables are substituted into the forward kinematics to find the positions of each joint. Now for each time integration the joints are plotted.

This simulation shows 4 views of the robot's movement: one isometric view, two side views (to show the X-Z and Y-Z movement), and a top view. These views are visible in Figure 3. This allows the user to understand the movement of the joints in each perspective.

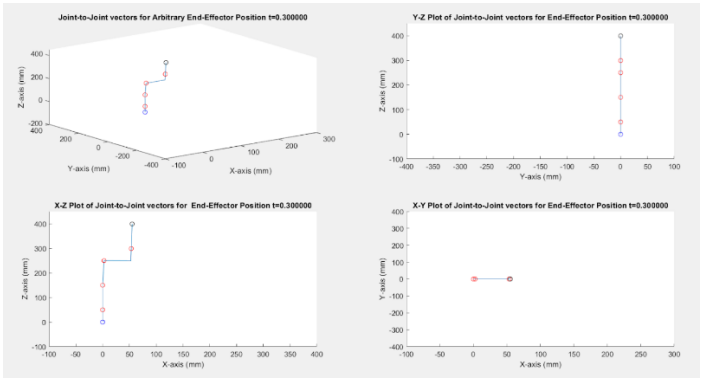

Figure 3: Simulation in the Home Position

The joints are represented as red dots and the links are shown as blue lines. For clarity, the end effector is shown as a black dot and the first joint is shown as a blue dot. The simulation also displays the current time in the plot headers. The robot begins in the home position (Figure 3), then moves to grab an item (Figure 4), lifts up (Figure 5), moves to the place location (Figures 6 & 7) and then returns to home.

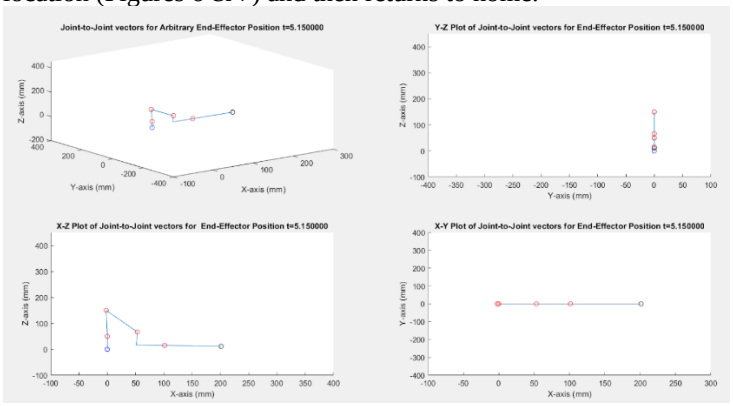

Figure 4: Simulation in the Pickup Location

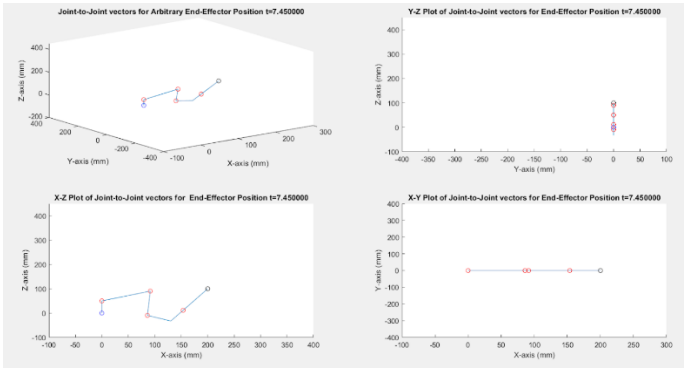

Figure 5: Simulation Lifting the Object

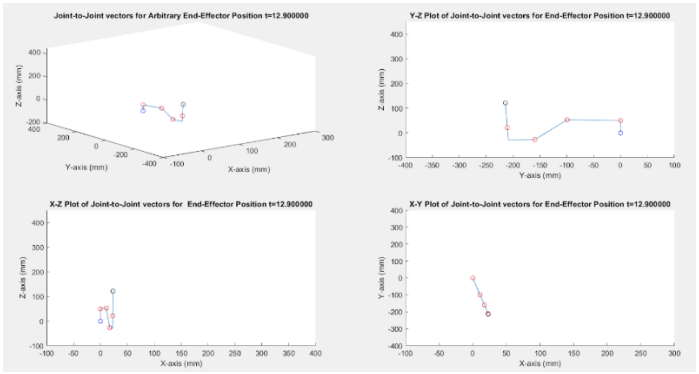

Figure 6: Simulation Moving Between Pick and Place Locations

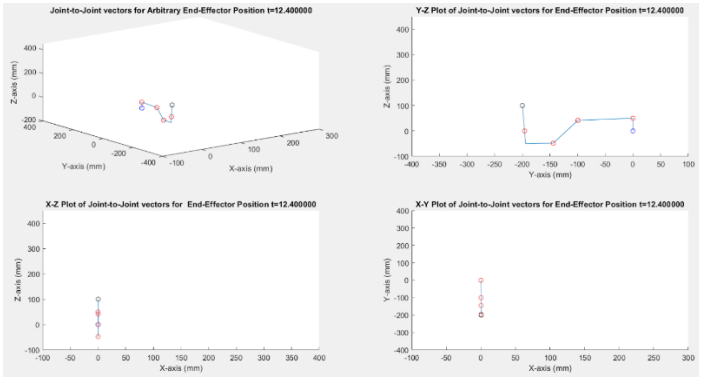

Figure 7: Simulation at Place Location

B. *Secondary Simulation*

This simulation is just an alternative representation of our physical robot. We did this just to show an example of a robots complete motion from the home position to the pick and place locations and then back to the home position. To complete this simulation, we used MATLAB as well as the MATLAB Robotics Toolbox. To create the links of the robot we set the values for the Denavit-Hartenberg parameters (theta (θ), d, r, alpha (α), and offset). Theta is the joint angle, d is the link offset, r is the link length, alpha is the link twist, and offset is the joint variable offset. We connected all the links together using the command "SerialLink" from the MATLAB Robotics Toolbox. Using trial and error we developed the joint angles needed to complete our desired trajectory and subsequently plotted the motion.

.

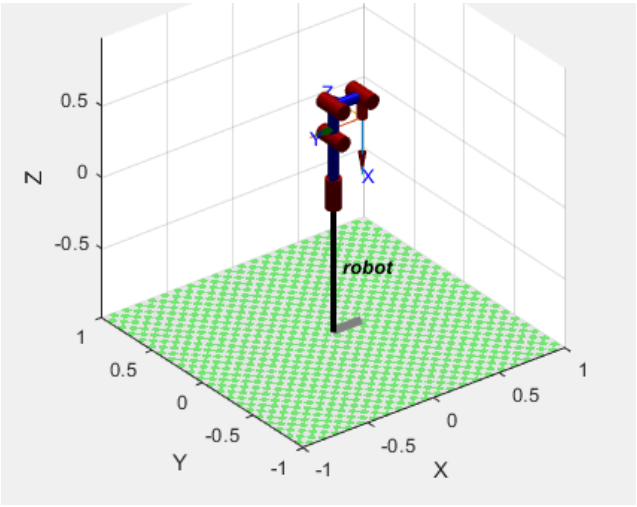

Figure 8: Robot at home position.

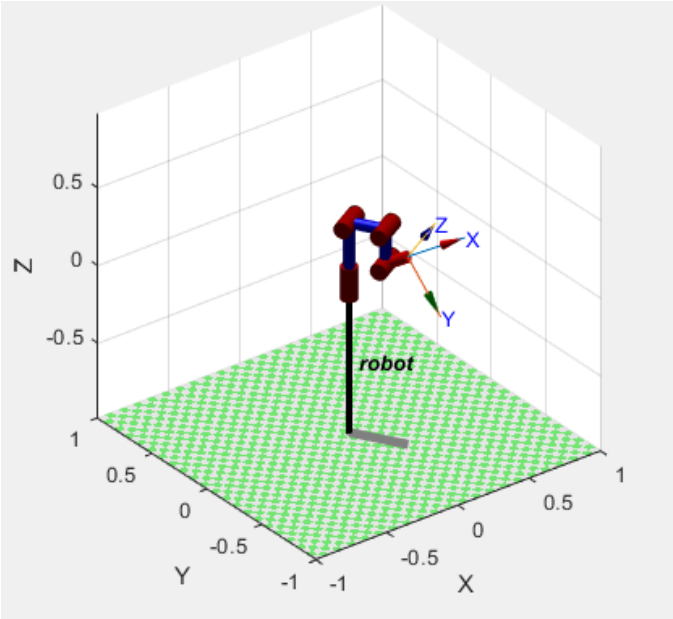

Figure 9: Robot at pick location.

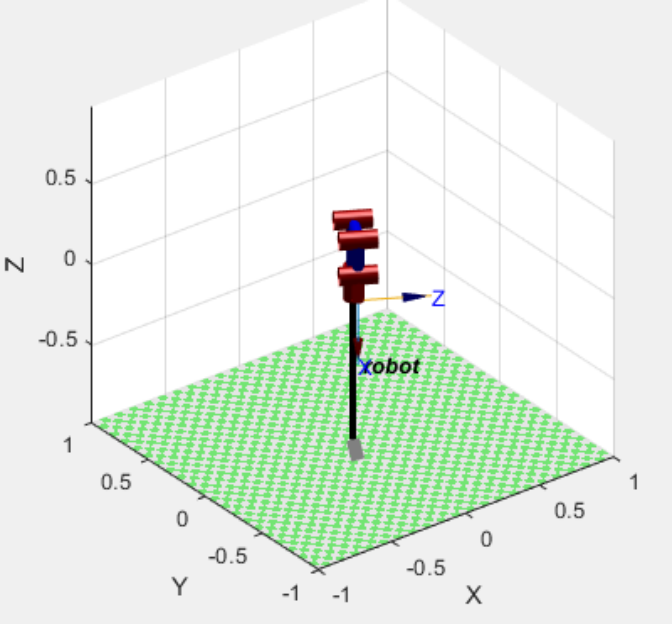

Figure 10: Robot at place location.

IV. Experiment

The purpose of the experimental system was to enhance our group's understanding of how robots, particularly arm robots, operate and are controlled in a real-world setting. This experiment also gave us the opportunity to experience firsthand some of the constraints and difficulties that robot operators and hobbyists face when operating a robot: such as faulty parts and power and control issues

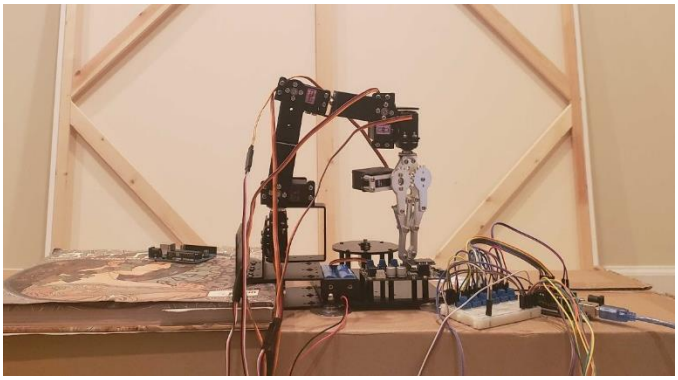

Figure 11: Five degree of freedom aluminum robot arm system used as the experimental system.

For the experimental system we constructed a five degree of freedom aluminum robotic arm utilizing six MG996R servo motors, see Figure 11. We programmed the system using an Arduino UNO microcontroller. The arm was to function as a "Pick and Place" robot. Its goal was to pick up a stack of Oreos from one location and transport them to another target location, see. Figure 12 displays the robotic arm picking up a stack of Oreos.

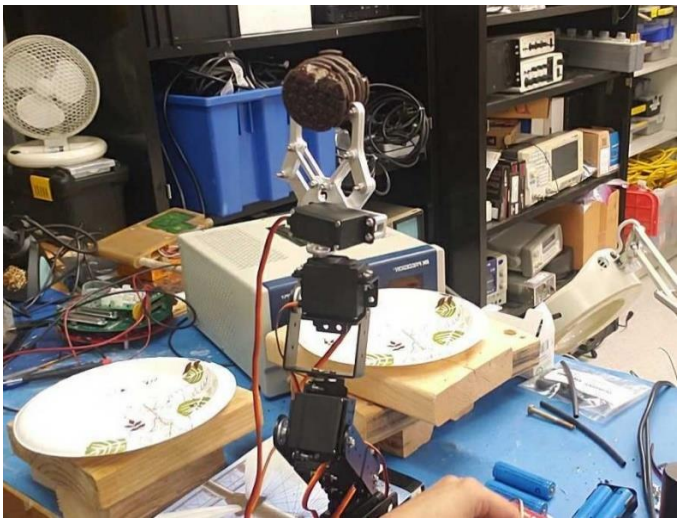

Figure 12: Robotic Arm lifting a stack of Oreos cookies.

The robotic arm circuit consists of six MG996R metal gear torque digital servo motors, connected in parallel. Five of the servo motors were used as joints for the links of the robot arm and one motor was used to open and close the gripper. The circuit also included six potentiometers. Each potentiometer was used to vary one of the joint angles of the robotic arm. A potentiometer is a three-terminal resistor with a radial dial that can be mechanically manipulated to vary the resistance in a circuit and acts as a voltage divider. An Arduino UNO microcontroller was chosen and programmed for this circuit. An Arduino microcontroller was chosen for this experiment because Arduino's are relatively cheap platform and readily available online and in stores. Two 18650 Lithium Ion batteries were used as an external power supply for this system.

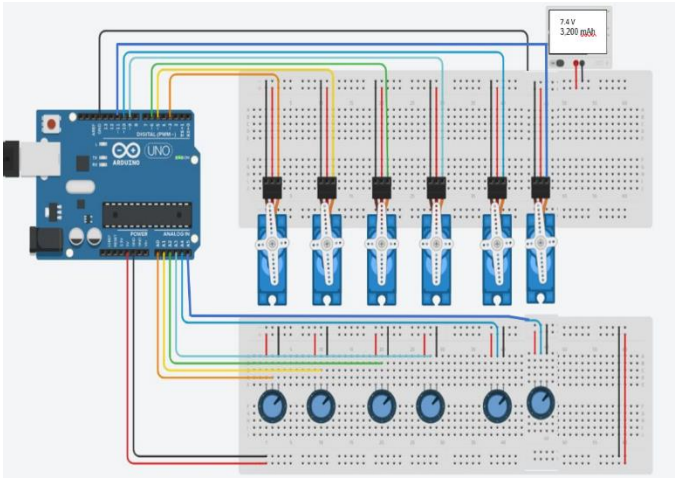

Figure 13: Initial circuit diagram for five DOF robotic arm.

Figure 13 shows the initial circuit diagram for the system and all its components This circuit required a lot of Arduino processing power, and so it caused the servo motors to jitter. It also drained the battery very fast. The batteries died after about 20 minutes of use. This made it difficult to move the arm to a from desired positions.

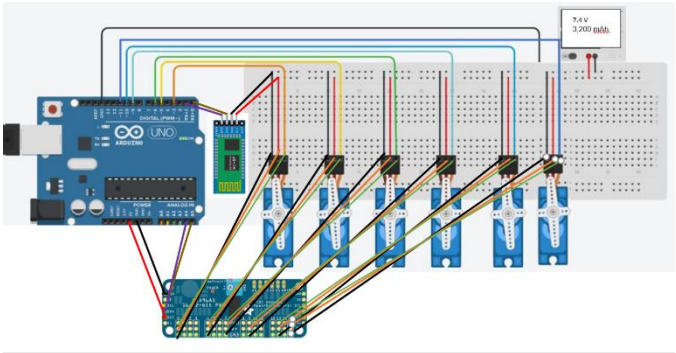

Figure 14: Final circuit diagram for five DOF robotic arm.

Figure 15: Mobile phone app controller for robotic arm.

To remedy this, a 16-channel 12-bit servo driver was introduced to the circuit. This driver is cable of allowing a board to control up sixteen servos on just two pins. The servo driver reduces the overhead Arduino processing power. An HC-06 Bluetooth module was also added to the circuit. it would allow a user to control the servo motors with their phone via Bluetooth. Figure 14 displays the improved circuit. We created mobile phone application, see Figure 15, using MIT inventor app that allowed the user to use sliders to input joint values. It also allowed the user to see the values of the joint values that they inputted.

This circuit was tested and compared with the previous circuit. Unfortunately, we were not able to test app controller. The Bluetooth module that we had ordered came damage and would not pair with any phones. Instead of controlling the servos in the second circuit with an app, we controlled them with potentiometers.

The second circuit performed much better than the first circuit. The second circuit was able to handle about an hour of use before the battery died. The servos still experienced some jitter occasionally, but it was much less than when they were connected using the first circuit. It also allowed for the robot it be more easily and accurately controlled and thus we were able to pick up an Oreo from one plate and place it on another. In the future in order to improve the performance, I would like to included a voltage regulator the circuit, purchase a new blue tooth module to control the arm with an app, and use a variable DC power supply instead of the batteries. The voltage regulator would allow the system to have fixed output voltage, I believe that this would allow the system to operate and maneuver better and more accurately. Using a power supply would allow me to use the arm for longer, as batteries tend to die.

REFERENCES

[1] Ove Larson and Charles Davidson. Flexible arm, particularly a robot arm, July 19 1983. US Patent 4,393,728.

[2] K.Lynch and F.Park. Modern Robotics: Mechanics, Planning, and Control. 2017

[3] S Saravana Perumaal and N Jawahar. Automated trajectory planner of industrial robot for pick-and-place task. *International Journal of Advanced Robotic Systems, 10(2):100, 2013.*