Build Log of a 3DoF Robotic Arm

Revision 4

Jim L

23/07/2023

*For educational purposes only*

Introduction and Parts List

A robot arm with wires connected to a circuit board

Description automatically generated

In early July of 2023, I set out to build a simple robotic arm using 3d printed parts as a proof of concept and a technical demo. I wanted to keep everything as simple as possible and use easily accessible components and software. This report will serve as a technical document and a catalog of the build process in addition to a guide for any potential replication. If you have any questions or comments, you can reach me at [liujim55@gmail.com](mailto:liujim55@gmail.com).

**Design and Parts List**

The design was an adaptation of the EEZYbotARM by *theGHIZmo* which can be found [here](https://www.instructables.com/EEZYbotARM/). Although an additional O ring was needed to provide a 5mm offset to adapt to the SG90 servos I had on hand.

A set of parts on a table

Description automatically generated

The STL for the 3d printed components can be found [here](https://www.thingiverse.com/thing:1015238). Additionally, you will need some screws and nuts as specified below:

7 M4 Nuts 4 M3x12 Hex Screw

15 M4 Washers 2 M3x12 TCEI screw

7 M3 Nuts 2 M3x20 TCEI screw

1 M3x30 Screw 5 M4x20 round hex recess screw

2 M3 Washers Brass pipe 4x3x22, x3x26

For the electronic components I used:

* 5 SG90 Servos
* 5 2 pin buttons
* Arduino Uno
* A breadboard
* A potentiometer
* Wires

**Assembly of the Robotic Arm**

The first step is to build the arm structure itself. I started by assembling the main joints with M4 screws before securing the other end with a M4 washer. Ideally these joints should be snug, but not so tight that they are immovable as the servos are relatively underpowered.

A white plastic parts on a black surface

Description automatically generated

A hand holding a white plastic object

Description automatically generated

Next you want to build the other forearm section using 2 M4 screws and nuts. Note the orientation of the bottom segment as its crucial for servo placement.

A white plastic arm on a table

Description automatically generated

A white plastic arm on a black surface

Description automatically generatedA white plastic parts on a table

Description automatically generated

The next step is to put the entire arm assembly together with the main ladder frame. At this point, I found it best to test fit the brass tube and cut it to length as you have the arm as a guide. However, remove the tube from the arm when cutting or else you risk snapping the 3d prints if you put too much weight on them.

A white object on a table

Description automatically generated

*Assembly of the main arm section, note the orientation of each piece and the location of the brass tube.*

The brass tube is used to provide a buffer between the M3x30 screw and the 3d print as that screw will act as the pivot axis for the arm and needs a smooth barrel in which to rotate in. Once again, ensure all limbs can easily be moved.

***Problem:*** *One of the 3d printed pieces snapped while cutting the brass tube.*

***Solution:*** *Print a replacement piece and cut the brass tube outside of the print itself.*

A white plastic arm on a table

Description automatically generated

With the main arm assembled, I started work on the base of the robotic arm. You will need a servo and M3 TCEI screws for this part as the M4 screw heads don’t have enough clearance for the walls.

A white plastic object with wires and screws

Description automatically generated

A hand holding a white tape

Description automatically generated

***Problem:*** *The 3d printed hole for the servo wires was too small for the wire to pass through.*

***Solution:*** *Manually enlarged the notch with a knife and a file.*

A white plastic object with orange wire

Description automatically generated

***Problem:*** *There was a gap between the servo and the top portion (the bit that rotates) of the assembly leading to significant wobble.*

***Solution:*** *Printed an O ring with a 5mm thickness and the same diameter as the base to act like a giant washer so that the top half was firmly sitting on the bottom half and not the servo itself.*

The last component to assemble is the claw gripper itself. This component was a bit tricky and took be a couple of tries to get right. Essentially you want to assemble it with the servo being in the 0-degree position so that it maximizes the amount the claw can open and close.

A close-up of a small white object

Description automatically generated

You want to use the slightly smaller but longer M3 by 20 TCEI screws for this bit.

Note the claw design is suck that the gears sit on each other in 2 layers. The top layer is the servo and gear with a hole in the center and the bottom layer is the grippers and the other gear.

A small white robot with orange wire

Description automatically generated

Lastly for final assembly, you want to fix everything together with M4 screws. A white robot arm with orange wire

Description automatically generated

A white robotic arm with orange wires

Description automatically generated

Notice how the indents in the limbs are so that the servo heads can be slotted into them to move the arm. Additionally, another piece of the brass tube is cut and used as a barrel to join the arm to the base.

**Programming of the Robotic Arm**

In order to control the servo motors to move the robotic arm, I used the Arduino Uno to act as the intermediary between my computer and the servos. I had 3 coding milestones in mind when I set out and they are as follows. Milestone 1: Manual control of each limb of the arm using potentiometers. Milestone 2: Computer control of each limb by entering the angle each servo should pivot to. Milestone 3: Computer control of the arm by entering Cartesian coordinates of a desired location and use inverse kinematics to determine the degree of rotation for each servo. Each milestone will be explored in the following sections and the code will be publicly available.

**Milestone 1**

The concept behind milestone 1 was to have 5 different potentiometers to independently control each limb of the arm. However, since I lacked 5 potentiometers, a workaround solution had to be found. Instead, I used 5 different buttons to select between each limb. The buttons were connected in parallel with each other, then in series with the potentiometer so that as long as a single button was pressed, only 1 limb would have the completed circuit running from the Arduino to the servo. The principle behind the code for this section is built upon the factor that SG90 servos turn to a specific degree based on the voltage provided to the servo. Since each servo can rotate 180 degrees and the potentiometer provides a number between 0 and 1023, I used the map function to directly relate each value received from the potentiometer straight into an angle for the servo to rotate to.

A robot arm with wires and a circuit board

Description automatically generated

Note: The Arduino takes both an input from the working servo and from the analog potentiometer.

The code for this milestone can be found in *robotic\_arm\_manual.ino*

**Milestone 2**

The goal for this milestone was to be able to input an angle for each servo and have the arm move into the corresponding position. For this milestone, the circuit had to be rewired as the buttons and potentiometer was no longer needed. Instead, each of the 5 data wires for the servos were plugged straight into the Arduino board. This way we can bypass the map function and use the Servo.h library to specify an exact angle for each individual servo to turn to. This greatly reduced the amount of code needed and improved the accuracy of the program. The challenge for this program was that the servo’s movement wasn’t linear and was in fact quite jarring. To resolve this, I applied the ramp function from the Ramp.h library. So instead of specifying “move to 150 degrees” and letting the servo rotate on its own, the ramp function would continuously input numbers leading up to 150 degrees such that by a certain amount of time later, the angle would reach 150 degrees. This broke the rotation up from 1 direct step to a lot of smaller steps which essentially allowed me to specify the speed of the rotation. By reducing the rotation speed to 5s, the arm’s movement was much smoother and less erratic.

The code for this milestone can be found in *robotic\_arm\_angle.ino*

**Milestone 3**

This milestone used the same circuit as the previous milestone, so no hardware changes were necessary. However, the code was basically rewrote to incorporate everything within 2 functions: move and calculate. The move function was basically milestone 2, where certain “destination” angles were passed in and function would move the arm into that position in a smooth manner. The calculate function was a result of the goal for this section, which was to be able to input cartesian coordinates instead of angles and have the computer calculate the required angles. Using inverse kinematics and basic trigonometry I was able to roughly approximate the angles each servo needed to rotate to for the final grabber to be in the specific cartesian coordinate position. I found [this](https://www.youtube.com/watch?v=Q-UeYEpwXXU&ab_channel=RoTechnic) video really helpful in my overall understanding of the topic. Finally, the last problem to overcome was the fact that these servos don’t remember their starting position so upon running the program all the servos would default to (0,0,0) even if the last instance left them at a different position. What resulted was the robot shaking itself back to the origin upon every execution before moving into the desired position. Without a complete hardware overhaul there was no way to fix this problem so as a temporary solution, I just told the robot to move back to the origin as the final command in every execution.

The code for this milestone can be found in *robotic\_arm\_angle\_rev3.ino* and the calculations can be found in *Cartesian Calculations.xlsx*

**Final resources**

Most of the content recorded for milestones 2 and 3 were incorporated into short demo videos which provide both a visual and auditory summary of the content as detailed above. I would encourage anyone interested to have a look. Finally, a summary video detailing each coding milestone was produced and can be found in the project folder titled as Robotic Arm Mk1.