

## **Use Authorization**

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission to download and/or print my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature

Date

SCARA Robot controlled by ROS

BY

Cenjiong GAO

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Mechanical Engineering

College of Engineering

Idaho State University

August 2019

## **COMMITTEE APPROVAL**

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of Cenjiong Gao find it satisfactory and recommend that it be accepted.

\_\_\_\_\_**Major Advisor**

Dr. Alba Perez Gracia

\_\_\_\_\_**Member**

Dr. Anish Sebastian

\_\_\_\_\_**Graduate Faculty Representative**

Dr. Joel Bocanegra

# Acknowledgements

Firstly, I would like to express my warmest gratitude to Dr. Alba Perez-Gracia, for her instructive suggestions and valuable comments on the writing of this thesis and my project. Without her helps and supports, the present thesis would not have been accomplished.

Special thanks goes to Dr. Anish Sebastian and Dr. Joel Bocanegra who are supporting me and willing to participate in my final defense committee.

I would also like to thanks to my friends and classmates helping me search for reference and giving me advices.

Finally, I would like to thanks to my parents and girlfriend who are encouraging me on Pursuing high education.

# Contents

Acknowledgements.....	iii
List of Figures.....	vi
List of Tables.....	viii
1 Introduction .....	1
1.1 Thesis Goals.....	1
1.2 Literature review .....	1
1.3 Organization of the Thesis .....	2
2 SOFTWARE.....	3
2.1 ROS.....	3
2.2 Why ROS? .....	3
2.3 What is ROS .....	3
2.4 Connecting the Arduino Uno and ROS .....	4
3 HARDWARE .....	7
3.1 Arduino Uno .....	7
3.2 NEMA 17 stepper motor.....	8
3.3 28 BYJ-48 stepper motor .....	10
3.4 A4988 Stepper Motor Driver.....	11
3.5 SG 90 servo motor .....	12
4 The SCARA ROBOT .....	14
4.1 Type of joints .....	14
4.2 Forward Kinematics.....	15
4.3 Inverse Kinematics.....	17
4.4 Path Planning .....	18
4.4.1 Workspace of SCARA Robot.....	18

4.4.2	Path Planning of SCARA Robot .....	19
5	Mechanical Design .....	22
5.1	Dimension of the SCARA Robot.....	22
5.2	Power for SCARA Robot.....	22
5.3	The Components Design.....	23
5.3.1	The Bases.....	23
5.3.2	The links .....	24
5.3.3	The Gripper.....	24
6	Coding .....	28
6.1	Pseudocode for stepper motor.....	28
6.2	Pseudocode for servo motor.....	28
7	Conclusion .....	30
8	Reference .....	31
9	Appendix A Technical Drawings of SCARA Robot.....	33
9.1	Code for stepper motor .....	49
9.2	Code for Servo Motor .....	51
10	Appendix C Shape of gripper .....	53
11	Appendix D Overview of SCARA Robot in 3D .....	55

# List of Figures

Figure 1 The process of communication between two nodes.....	3
Figure 2 Screenshot of ROS Master in terminal.....	5
Figure 3 Screenshot of connection between ROS and Arduino IDE in terminal .....	6
Figure 4 Arduino Uno[10] .....	7
Figure 5 Arduino IDE.....	8
Figure 6 NEMA 17 stepper motor .....	9
Figure 7 Specification of NEMA 17 stepper motor.....	9
Figure 8 Dimension of NEMA 17 stepper motor[8].....	10
Figure 9 A4988 Driving Board[7] .....	10
Figure 10 28 BYJ-48 stepper motor[9].....	11
Figure 11 The dimension of SG 90 servo motor[6].....	13
Figure 12 the illustrates the typical robot joints5[1].....	14
Figure 13 SCARA Robot frame placement .....	16
Figure 14 SCARA top view scheme.....	17
Figure 15 Workspace of SCARA Robot.....	18
Figure 16 Top view of SCARA Robot .....	19
Figure 17 Path planning of Way A1and A2 .....	20
Figure 18 Path planning of Way B1 and B2.....	20
Figure 19 Scheme of SCARA Robot.....	22
Figure 20 The base of SCARA Robot .....	23
Figure 21 The connection between white base2 and NEMA stepper motor .....	23
Figure 22 The Second link.....	24
Figure 23The front view of SCARA Robot.....	25
Figure 24 The front view of SCARA Robot.....	25
Figure 25 The front view of SCARA Robot.....	26
Figure 26 The front view of SCARA Robot.....	26
Figure 27 The gripper with rubbers .....	27
Figure 28 Technical drawing of Base1 .....	33
Figure 29 3D view of Base1 .....	34

Figure 30 Technical drawing of Base2 .....	35
Figure 31 3D view of Base2 .....	36
Figure 32 Technical drawing of Base3 .....	37
Figure 33 3D view of Base3 .....	38
Figure 34 Technical drawing of Link1 .....	39
Figure 35 3D view of Link2 .....	40
Figure 36 Technical drawing of connection component .....	41
Figure 37 3D view of connection component.....	42
Figure 38 Technical drawing of Link2 .....	43
Figure 39 3D view of Link2 .....	44
Figure 40 Technical drawing of Link3 .....	45
Figure 41 3D view of Link3 .....	46
Figure 42 Technical view of Link4.....	47
Figure 43 3D view of Link4 .....	48
Figure 44 Running motor with 30 degrees .....	53
Figure 45 Running motor with 60 degree.....	53
Figure 46 Running motor with 90 degree.....	53
Figure 47 Running motor with 120 degree.....	54
Figure 48 Running motor with 150 degree.....	54
Figure 49 3D view of SCARA Robot.....	55



# List of Tables

Table 1 The specification of 28 BYJ-48 stepper motor.....	11
Table 2 Feature of A4988 stepper motor driver[7].....	12
Table 3 Connection between NEMA17 stepper motor and stepper motor driver .....	12
Table 4 connection between 28 BYJ-48 stepper motor and stepper motor driver .....	12
Table 5 The feature of SG90 servo motor[6].....	13
Table 6 The number of degrees of freedom by basic joints[1].....	15
Table 7 The D-H Algorithm of SCARA Robot.....	15
Table 8 Coordinate of Point A, Point B and Length of Link1 and Link2 .....	20
Table 9 Position of Point A and Point B.....	20
Table 10 Steps for stepper motors rotate .....	21

# SCARA Robot controlled by ROS

## Thesis Abstract

This SCARA Robot is able to do a pick and place, which grips wood cube (20mm x 20mm x 20mm) from one place to another required position by Kinematics method. Robot consists of one base, two links, two stepper motors, two servo motors and one end-effector. And thesis introduces the way of controlling SCARA Robot by using ROS.

Key Words: SCARA, ROS, Robot

# 1 Introduction

## 1.1 Thesis Goals

The goal of this thesis is to design economical SCARA Robot to finish pick and place task accurately for manufacture in order to reduce the costing of producing products. Also, this robot is controlled by Arduino Uno through using Robot Operating System (ROS) directly with Arduino Integrated Development Environment (IDE).

To achieve this goal, I choose Arduino Uno and ROS control the whole system, two stepper motors and two servo motors as actuators, 3D printer to print the links to connect with the motors and gripper.

## 1.2 Literature review

SCARA Robot was established in 1978 by Professor Hiroshi Makino who is from Yamanashi University in Japan.

The use of industrial robots for more than 60 years has made it possible to increase productivity and improve the quality of manufactured products.[13] Especially, SCARA Robot are widely used in manufacturing industry to replace human do repetitive tasks for reducing cost and ratio of accidents.

The robot has mechanical, electronic, control system, and the intuitive graphic interface designed and implemented for it allows the user to easily command this robot and to generate trajectories for it.[5] So the first step is to decide what controller is used for the robot. Matlab/Simulink programming was used for controlling the SCARA Robot by Claudio Urrea, Juan Cortes, Jose Pascal[5]. However, it is not economic way for robot amateur to subscribe Matlab. So, ROS as an open source system is a good option. The unique about ROS is the level of widespread support for ROS across the robotics community. This “critical mass” of support makes it reasonable to predict that ROS will continue to evolve, expand, and improve in the future.[1]

Forward Kinematics is important in designing moving robot as the aim is to get accurate position and orientation of end effector. Compared with Forward Kinematics, the aim of Inverse Kinematics is to get the value of parameter that allow a specific position to be reached. The Figure 2.4 shows the SCARA frames placement. From the Figure 2.4, there are only four variables:  $\theta_1$ ,  $\theta_2$ ,  $\theta_3$  d4. Table 2.1 shows the link and joint parameters,[12] which can be used to get the transformation matrix.

### **1.3 Organization of the Thesis**

The thesis is organized as follows. Chapter 1 discuss the Software used in this project. Chapter 2 introduces the hardware used in this project. Chapter 3 shows the process of designing SCARA Robot.

## 2 SOFTWARE

### 2.1 ROS

### 2.2 Why ROS?

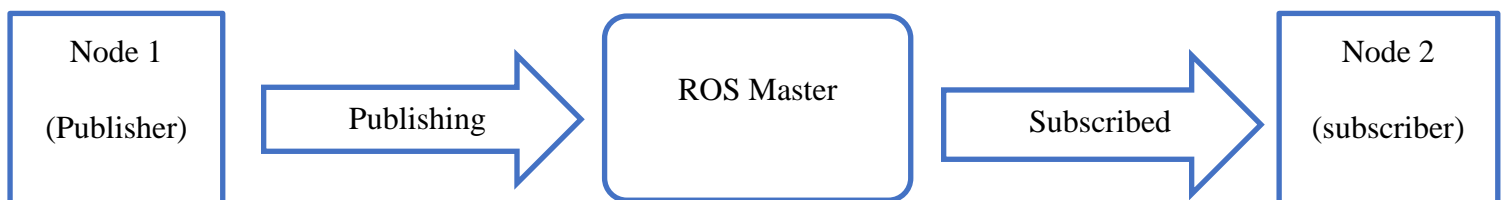
Robotics is widely used in variety area and robotics communities play a significant role in recent year. However, it is difficult for software developers to build robots they need. The Robot Operating System (ROS) can effectively solve this problem.

### 2.3 What is ROS

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management.

ROS consists of ROS master, ROS packages, ROS messages, ROS services, ROS node, ROS topics and ROS bags.

ROS Master: the ROS master help the nodes to find out each other to exchange messages and invoke services. Different kinds of nodes are connected by ROS Master. The following show the process of communication between two nodes:



*Figure 1 The process of communication between two nodes*

**ROS Package:** A ROS package is a coherent collection of files, generally including both executables and supporting files, that serves a specific purpose.[1]

**ROS Nodes:** Nodes are able to perform computation to figure out different kinds of tasks in a robot by using a simple structure.

**Parameter services:** The parameter services, are one part of ROS Master, are able to save the data in central location for nodes to access and modify.

**Messages:** Messages are used to communicate between nodes.

**Topics:** Topics are the way of transporting messages between nodes. If nodes send messages through topics then I say nodes publish a topic. If nodes receive messages through topics then we say nodes subscribe a topic.

**Bags:** Bags are a format for saving and playing back ROS message data. Bags are an important mechanism for storing data, such as sensor data, which can be difficult to collect but is necessary for developing and testing robot algorithms. Bags are very useful features when I work with complex robot mechanisms. [3]

## **2.4 Connecting the Arduino Uno and ROS**

One package, named `roserial_arduino`, is used for communication directly between ROS and Arduino IDE. Rosserial provides a ROS communication protocol that works over your Arduino's UART. It allows your Arduino to be a full-fledged ROS node which can directly publish and subscribe to ROS messages, publish TF transforms, and get the ROS system time. [11]

The first step is to install the Rosserial package by following code in terminal. Change the name of ROS version you are using from indigo: e.g. kinetic.

```
sudo apt-get install ros-indigo-roserial-arduino
```

```
sudo apt-get install ros-indigo-roserial
```

Then build code in workspace.

```
cd <ws>/src
```

```
git clone https://github.com/ros-drivers/roserial.git
```

```
cd <ws>
```

`catkin_make`


<ws> means workspace. “catkin\_make” is designed to build all of the packages in your workspace directory. It will perform several configuration steps(especially the first time you run it) and create subdirectories called devel and build within your workspace.[1]

The next step is to copy the ros\_lib that has been create by previous step to Arduino library in order to enable Arduino communicate with ROS.

Finally restarting Arduino IDE, the ros\_lib appears in file>example.

After correctly connecting Arduino and computer, the first step is to start the ROS Master with the roscore;

The following shows the terminals of ROS Master in Ubuntu:

A terminal window with a dark purple background and white text. The text shows the process of starting the ROS Master. It begins with a prompt to press Ctrl-C to interrupt, followed by a message about log file disk usage. Then, it shows the start of the roslaunch server with a specific URI and the ROS version 1.12.14. A summary section follows, listing parameters like /rostdistro: kinetic and /rosversion: 1.12.14. The nodes section shows the auto-starting of a new master, the master process starting with PID 5582, the ROS\_MASTER\_URI being set to the same URI, the /run\_id being set to a long hexadecimal string, the roscore process starting with PID 5595, and finally the core service [/roscout] being started.

```
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://perealba093981:35333/
ros_comm version 1.12.14

SUMMARY
=====
PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

NODES

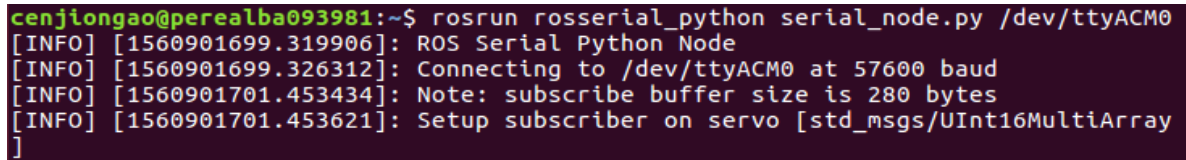
auto-starting new master
process[master]: started with pid [5582]
ROS_MASTER_URI=http://perealba093981:11311/

setting /run_id to 1fc3a660-921b-11e9-8193-1866da0dd4f6
process[roscout-1]: started with pid [5595]
started core service [/roscout]
```

Figure 2 Screenshot of ROS Master in terminal

Then uploading stepper motor.ino in Arduino IDE, run *roslaunch roserial\_python serial\_node.py /dev/ttyACM0* in a new terminal, make sure set the correct port used in Arduino IDE in advance. In this example, the serial port is *ttyACM0*.

The following shows the terminal of connection setting between ROS and Arduino IDE:



```
cenjiongao@perealba093981:~$ roslaunch roserial_python serial_node.py /dev/ttyACM0
[INFO] [1560901699.319906]: ROS Serial Python Node
[INFO] [1560901699.326312]: Connecting to /dev/ttyACM0 at 57600 baud
[INFO] [1560901701.453434]: Note: subscribe buffer size is 280 bytes
[INFO] [1560901701.453621]: Setup subscriber on servo [std_msgs/UInt16MultiArray
]
```

Figure 3 Screenshot of connection between ROS and Arduino IDE in terminal

Execute the code once by run *rostopic pub motor2/start std\_msgs/Empty --once* in a new terminal. The stepper motors will rotate in 15 rpm.



## 3 HARDWARE

### 3.1 Arduino Uno

Arduino is an open-source platform, which is based on hardware and software. It also is a tool for fast prototyping, which aims at people who have no background in electronics and programming, for example high school students, hobbyists and artists. The functionality of the Arduino is variety: light on a sensor, finger on a button or twitter message.

Arduino Uno as a micro controller, there are two power outputs: 5 volts and 3.3 volts for output and 13 digital pins for communicating with computer via USB cable.

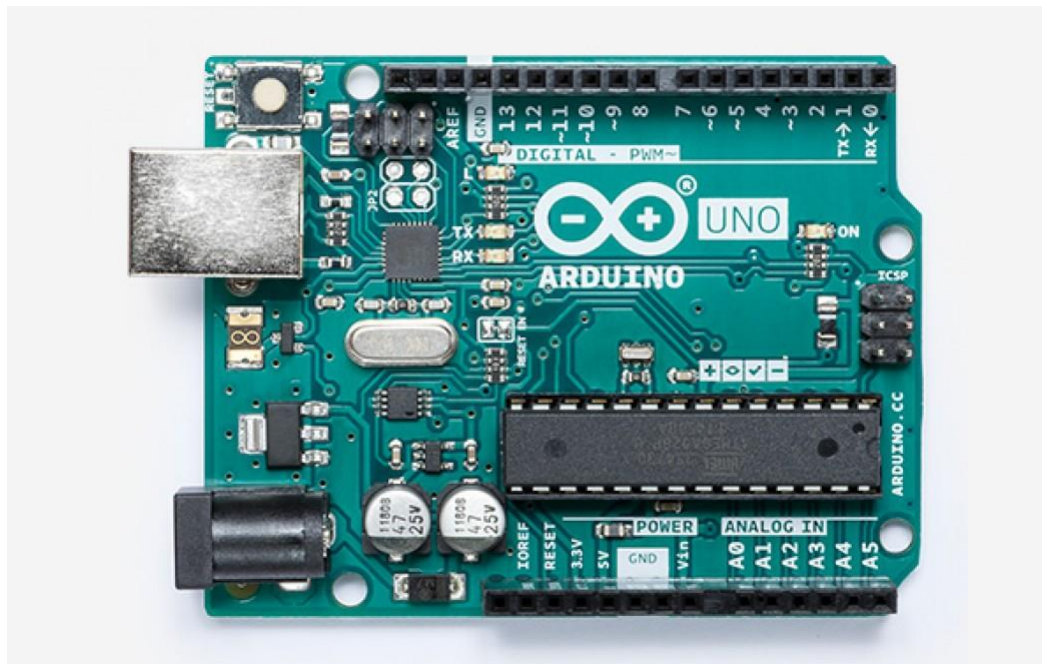


Figure 4 Arduino Uno[10]

The Arduino IDE that provide lots of libraries and examples for users to use.



*Figure 5 Arduino IDE*

### 3.2 NEMA 17 stepper motor

The step means fixed angle increments. It can move in accurate, which likes a servo motor. Compared with servo motor, stepper motor can move to a pre-defined position via sending plus from stepper driver and rotate continuously.

The first link connects with the NEMA 17 stepper motor (42.3mm x 42.3mm x 39mm), which is a four phase and permanent magnet stepper motor. There are 200 steps in one revolution while the shaft only has 1.8 degree for full step and 0.9 degree for half-stepping mode.



Figure 6 NEMA 17 stepper motor

Although NEMA 17 stepper motor is cheap, it has various of functionalities in industrial application. The following is the specification of NEMA 17 stepper motor:

Stepper Motor Specifications	
<b>Mosaic Part No.:</b>	STEPMOT-1
<b>Manufacturer Part No.:</b>	42BYG228
<b>Size:</b>	NEMA 17
<b>Drive system:</b>	Unipolar
<b>Step angle:</b>	1.8° full step 0.9° half-step
<b>Phase/Windings:</b>	4/2
<b>Voltage &amp; Current:</b>	12V at 400 mA
<b>Resistance per Phase:</b>	30 ohms
<b>Inductance per Phase:</b>	23 mH
<b>Holding Torque:</b>	2000 g-cm
<b>Detent Torque:</b>	220 g-cm max
<b>Weight:</b>	0.24 kg (0.5 lbs.)
<b>Max continuous power:</b>	5 W
<b>Rotor Inertia:</b>	22 g-cm <sup>2</sup>
<b>Bearings:</b>	Ball
<b>Leads:</b>	18 in. 26 AWG UL 1007
<b>Insulation resistance:</b>	>100 MΩ at 500VDC
<b>Dielectric strength:</b>	500V 50Hz/minute
<b>Mounting hole space diagonal:</b>	1.73 in.
<b>Mounting screws:</b>	3 mm dia. 0.5 mm pitch
<b>Shaft diameter:</b>	0.197 in. (5 mm)
<b>Motor footprint:</b>	1.7 in. × 1.7 in.
<b>Motor height:</b>	1.5 in.
<b>Ambient temperature:</b>	-10°C to +55°C

Figure 7 Specification of NEMA 17 stepper motor

The following is dimension of NEMA 17 stepper motor:

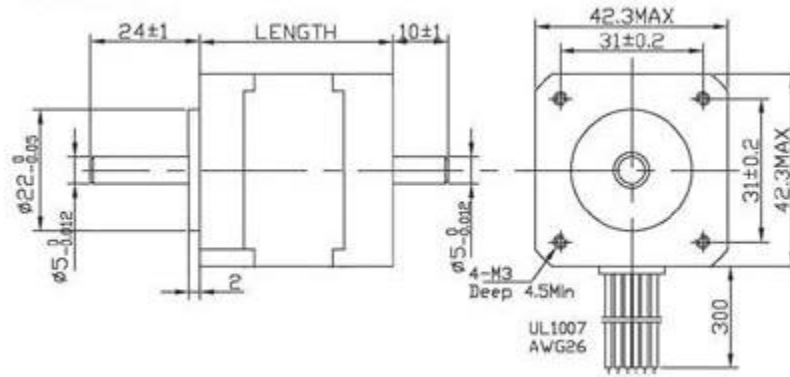


Figure 8 Dimension of NEMA 17 stepper motor[8]

### 3.3 28 BYJ-48 stepper motor

The 28BYJ-48 stepper motor connects with first link and second link for rotating the robot's end-effector. This stepper motor needs 5 volts and is nearly 15 RPM (rotation per minute) as the gear reduction ratio is approximately 64:1, This is why it is called higher-torque/lower speed stepper motor. The following is the specification of 28 BYJ-48 stepper motor.



Figure 9 A4988 Driving Board[7]

Table 1 The specification of 28 BYJ-48 stepper motor

Motor Type	Unipolar stepper motor
Voltage	5V-12V
Frequency	100Hz
Gear Ratio	64:1
Step Mode	Half-step mode: 5.625 degrees per step, 64 steps for one revolution Full-step mode: 11.25 degrees per step, 32 steps for one revolution
Stepper Driver	A4988 stepper driver

The following is dimension of 28 BYJ-48 stepper motor:

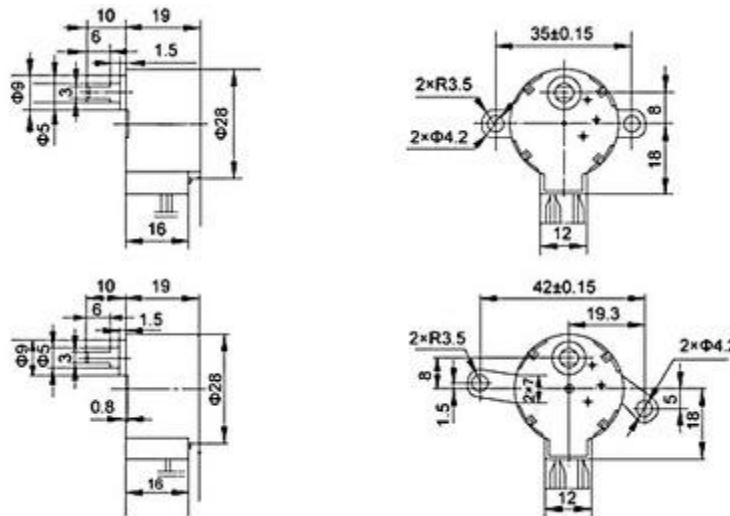


Figure 10 28 BYJ-48 stepper motor[9]

### 3.4 A4988 Stepper Motor Driver

A4988 stepper motor driver is easy to control bipolar stepper motors as a micro-stepping driver. This stepper driver can control five different kinds of step resolution: full-step, half-step, quarter-step, eighth-step and sixteenth-step. The following shows the feature of A4988 stepper motor driver:

Table 2 Feature of A4988 stepper motor driver[7]

Model	A4988
Dimension	5mmx5mmx0.9mm
Voltage	3~5.5v
Current	1A~2A
Operating Voltage	8~35v

In this project, the A4988 stepper motor driver is used for driving NEMA 17 stepper motor and 28 BYJ-48 stepper motor. The following shows the connection from two stepper motors to A4988:

Table 3 Connection between NEMA17 stepper motor and stepper motor driver

Grey	1B
Yellow	1A
Green	2A
Red	1B

Table 4 connection between 28 BYJ-48 stepper motor and stepper motor driver

Orange wire	1B
Pink	1A
Yellow	2A
Orange	1B

### 3.5 SG 90 servo motor

In order to accurately control the second link and the end-effector of the SCARA robot, servo motor would be a good choose to meet the requirement. Basically, the servo motor consists of DC motor and a gear box that provides low speed hand high torque. The servo motor receives a control signal. Then the position sensor, connected with shaft and gearbox, senses the position of shaft and sends signal of definite position to control circuit that decodes signal, which is used to compared with actual position of motor. Finally, the control circuit controls the direction of

shaft of the DC motor to get the required position. In this project, SG 90 serve motor can be a good choose. The following is dimension of the SG 90 servo motor:

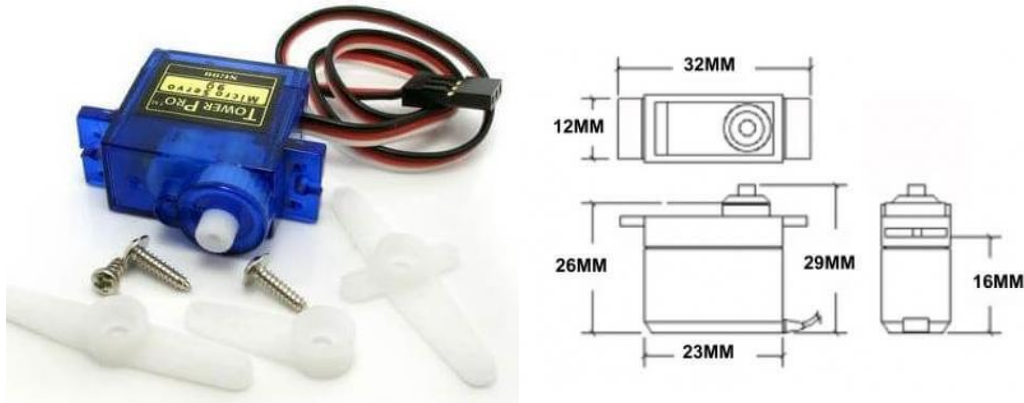


Figure 11 The dimension of SG 90 servo motor[6]

The feature of SG 90 servo motor

Table 5 The feature of SG90 servo motor[6]

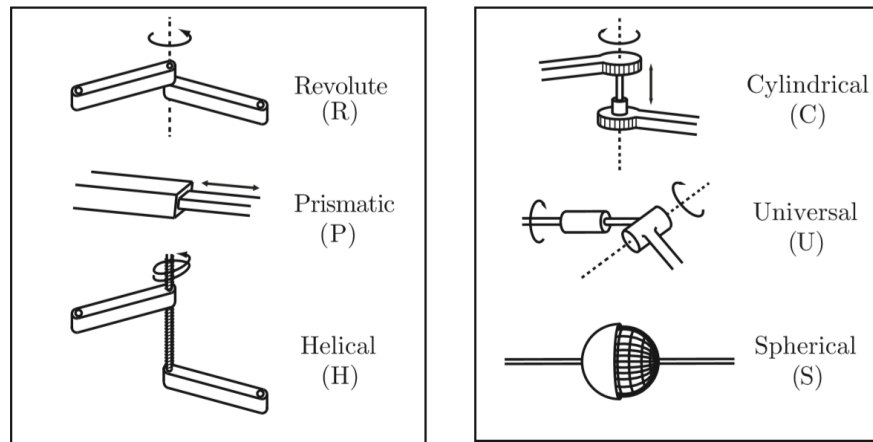
Model	SG90 Servo Motor
Working voltage	3.3v~5v
Operating Speed(6v)	0.12 Sec/60 Degree
Stall Torque(6v)	1.5kg/cm
Motor type	Brushed DC Motor
Gear Type	Plastic Gears
Dimensions	25mmx11.5mmx24mm

SG 90 servo motor can rotate from 0 to 180 degree. It already has connected with a gear box which has 2.5 kg·cm high output power. This servo motor can be easy to rotate the gripper and open or close the gripper to grasp the items I want to move. There are three wires, black wire, red wire and white wire, are connected with GND, power pin (5v or 3.3v) and signal pin on the Arduino.

## 4 The SCARA ROBOT

### 4.1 Type of joints

There are six basic types of joints: revolute joint, prismatic joint, helical joint, cylindrical joint, universal joint, spherical joint. Every joint can exactly connect with two links.



*Figure 12 illustrates the typical robot joints[1]*

Revolute joint(R): It allows rotational motion around joint axis.

Prismatic joint(P): It allows straight motion along the direction of the joint axis.

Helical joint(H): It allows rotational and translational motion about a screw axis.

Cylindrical joint(C): It can do the translation and rotation independently.

Universal joint(U): It consists of one pair of revolute joint.

Spherical joint(S): It consists of a ball and a socket.



Table 6 The number of degrees of freedom by basic joints[1]

Joint type	DOF
Revolute joint(R)	1
Prismatic joint(P)	1
Helical joint(H)	1
Cylindrical joint(C)	2
Universal joint(U)	2
Spherical joint(S)	3

## 4.2 Forward Kinematics

The Denavit-Hartenberg Algorithm is used to describe the kinematics of the SCARA Robot. [4]

The following is the SCARA Robot manipulator parameters:

Table 7 The D-H Algorithm of SCARA Robot

i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1$	0	120mm	0
2	$\theta_2$	0	45mm	0
3	$\theta_3$	20mm	0	180
4	$\theta_4$	0	0	0

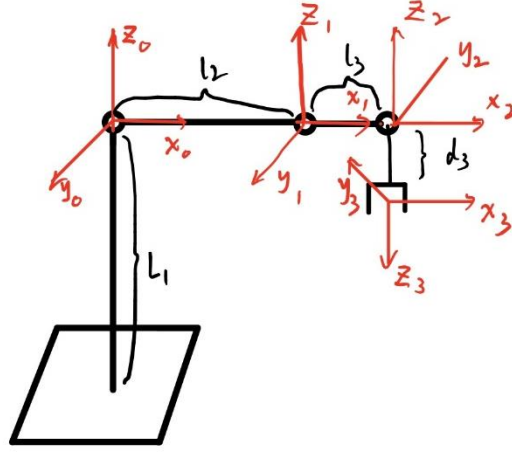


Figure 13 SCARA Robot frame placement

The transformation matrix from base to end-effector is  $T_{\text{base}}^{\text{end}}$ :

$$T_{\text{base}}^{\text{end}} = T_0^1 T_1^2 T_2^3$$

$$T_0^1 = \begin{bmatrix} C_1 & S_1 & 0 & l_2 C_1 \\ S_1 & -C_1 & 0 & l_2 S_1 \\ 0 & 0 & -1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^2 = \begin{bmatrix} C_2 & -S_2 & 0 & l_3 C_2 \\ S_2 & -C_2 & 0 & l_3 S_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^3 = \begin{bmatrix} C_3 & S_3 & 0 & 0 \\ S_3 & -C_3 & 0 & 0 \\ 0 & 0 & -1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{\text{base}}^{\text{end}} = \begin{bmatrix} (C_{12} + S_{12}) \times C_3 - (S_1 C_2 - C_1 S_2) \times S_3 & (C_1 S_2 + S_1 C_2) \times C_3 + (C_{12} - S_{12}) \times S_3 & 0 & l_3 \times (S_{12} + C_{12}) \\ (C_{12} - S_{12}) \times S_3 + (S_1 C_2 - C_1 S_2) \times C_3 & (-C_1 S_2 + S_1 C_2) \times S_3 + (C_{12} + S_{12}) \times C_3 & 0 & l_3 \times ((S_1 C_2 - C_1 S_2)) \\ 0 & 0 & 1 & l_1 \times d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1 - \theta_2 + \theta_3) & \sin(\theta_1 + \theta_2 + \theta_3) & 0 & l_3 \times \cos(\theta_1 - \theta_2) \\ \cos(\theta_1 + \theta_2 + \theta_3) & \cos(\theta_1 - \theta_2 - \theta_3) & 0 & l_3 \times \sin(\theta_1 + \theta_2) \\ 0 & 0 & 1 & l_1 \times d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where  $C_{12} = \cos \theta_1 \times \cos \theta_2$   $S_{12} = \sin \theta_1 \times \sin \theta_2$

### 4.3 Inverse Kinematics

The aim of the inverse kinematics is to get the value of the variable parameters:  $\theta_1$  and  $\theta_2$ . The parameters values are listed in table 7. The coordinate of end-effector is (x, y)

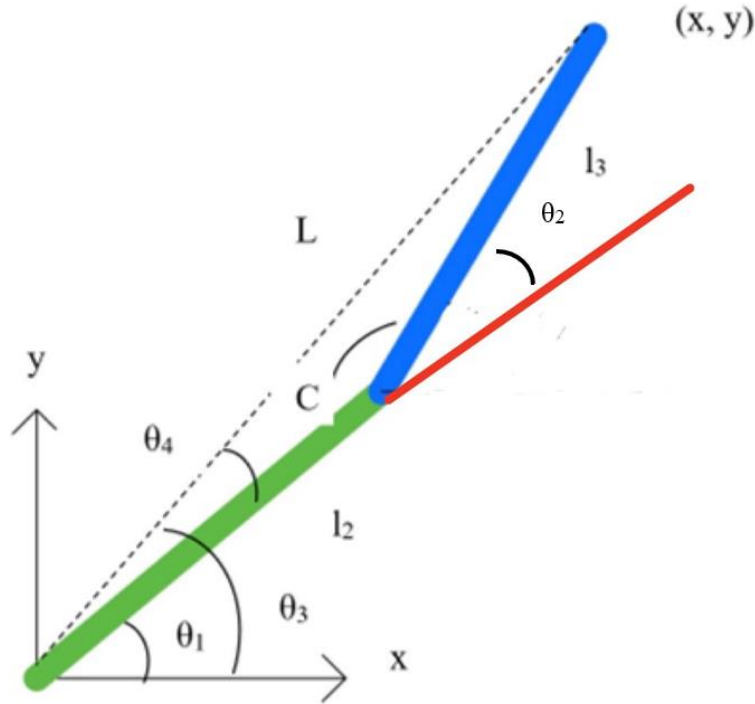


Figure 14 SCARA top view scheme

From ruler of Cosine:

$$\sqrt{x^2 + y^2} = l_2^2 + l_3^2 - 2 \times l_2 \times l_3 \times \cos C,$$

Where C is the angle between  $l_2$  and  $l_3$ .

$$\cos C = \frac{x^2 + y^2 - l_2^2 - l_3^2}{2 \times l_2 \times l_3}$$

$$C = \cos^{-1} \frac{x^2 + y^2 - l_2^2 - l_3^2}{2 \times l_2 \times l_3}$$

$$\theta_2 = \pm (\pi - C)$$

$\theta_3$  is the angle between L and x-axis.

$$\tan(\theta_3) = \frac{y}{x}$$

$$\theta_3 = \tan^{-1} \frac{y}{x}$$

$\theta_4$  is the angle between L and  $l_2$ .

From ruler of Cosine:

$$\cos \theta_4 = \frac{x^2 + y^2 + l_2^2 - l_3^2}{2 \times l_2 \times \sqrt{x^2 + y^2}}$$

$$\theta_4 = \pm \cos^{-1} \frac{x^2 + y^2 + l_2^2 - l_3^2}{2 \times l_2 \times \sqrt{x^2 + y^2}}$$

$$\theta_1 = \theta_3 - \theta_4$$

$$\theta_1 = \tan^{-1} \frac{y}{x} \pm \cos^{-1} \frac{x^2 + y^2 + l_2^2 - l_3^2}{2 \times l_2 \times \sqrt{x^2 + y^2}}$$

## 4.4 Path Planning

### 4.4.1 Workspace of SCARA Robot

The blue area is workspace of SCARA Robot shown in Figure 15, the area between two circles the circle1 with  $r_1 = 180\text{mm}$  and circle2 with  $r_2 = 52\text{mm}$ . The length of link1 is  $l_1 = 120\text{mm}$  and link2 is  $l_2 = 60\text{mm}$ . The end effector can reach any point between two circles.

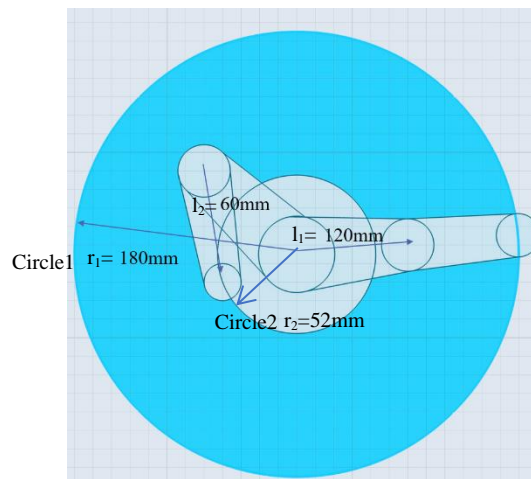


Figure 15 Workspace of SCARA Robot

#### 4.4.2 Path Planning of SCARA Robot

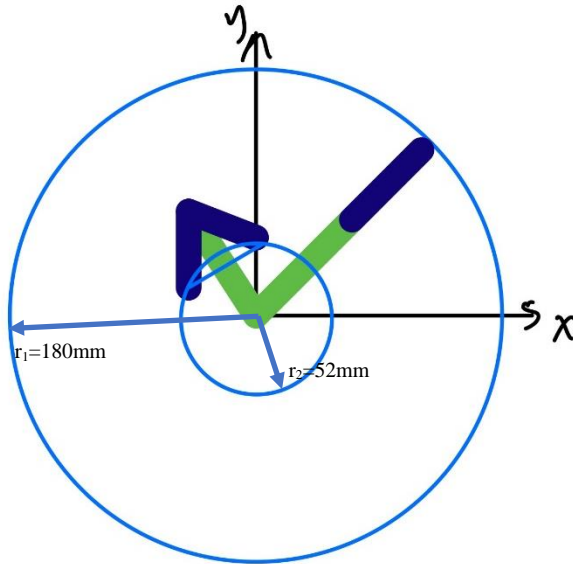


Figure 16 Top view of SCARA Robot

Figure 16 shows the top view of SCARA Robot in Cartesian coordinate system. The green line represents Link1 with  $l_1 = 120\text{mm}$  and purple line Link2 with  $l_2 = 60\text{mm}$  separately.

The method of path planning I used is to calculate the parameters  $\theta_1$  and  $\theta_2$  of starting point and end point by using the Inverse Kinematics Then get the difference of angles of these two points and transfer the angle to these steps from specification of stepper motors separately.

This example shows that SCARA Robot picks a wood cube from Point A (120,60) to place to Point B (60,120) in Cartesian Coordinate system. The Coordinate of Point A, Point B and Length of Link1 and Link2 are shown in Table 8. Two solutions shown in Figure 17 are shown in Table 9 for making robot move to Point A from Point Q (180, 0) as known A1 and A2. Two solutions shown in Figure 18 are shown in Table 9 for making robot move to Point B from Point Q (180, 0) as known B1 and B2.



Figure 17 Path planning of Way A1 and A2

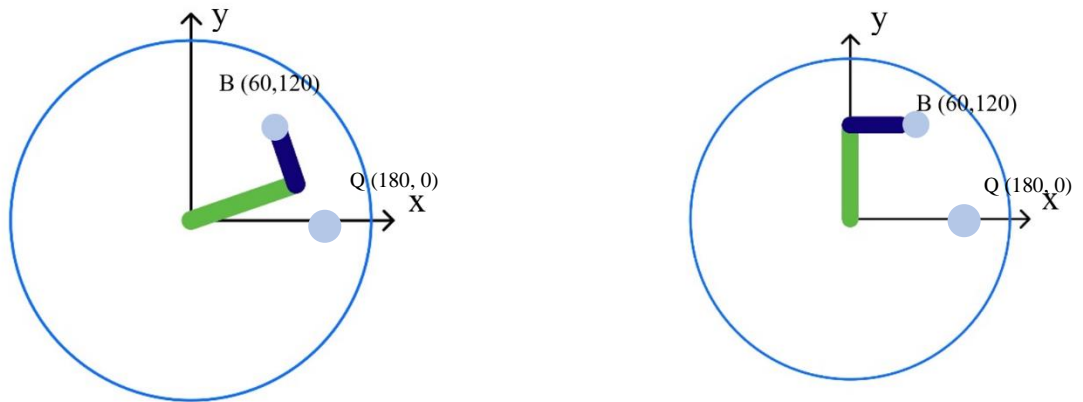


Figure 18 Path planning of Way B1 and B2

Table 8 Coordinate of Point A, Point B and Length of Link1 and Link2

Coordinate of start position		Coordinate of end position		Length of Link1 and Link2	
$x_1$	$y_1$	$x_2$	$y_2$	$l_1$	$l_2$
120	60	60	120	120	60

Table 9 Position of Point A and Point B

	Start position		End position	
	$\theta_1$	$\theta_2$	$\theta_1$	$\theta_2$
Solution1	0.0	1.6	1.6	-1.6
	0	90	90	-90
Solution2	0.9	-1.6	0.6	1.6
	53	-90	37	90

Table 10 Steps for stepper motors rotate

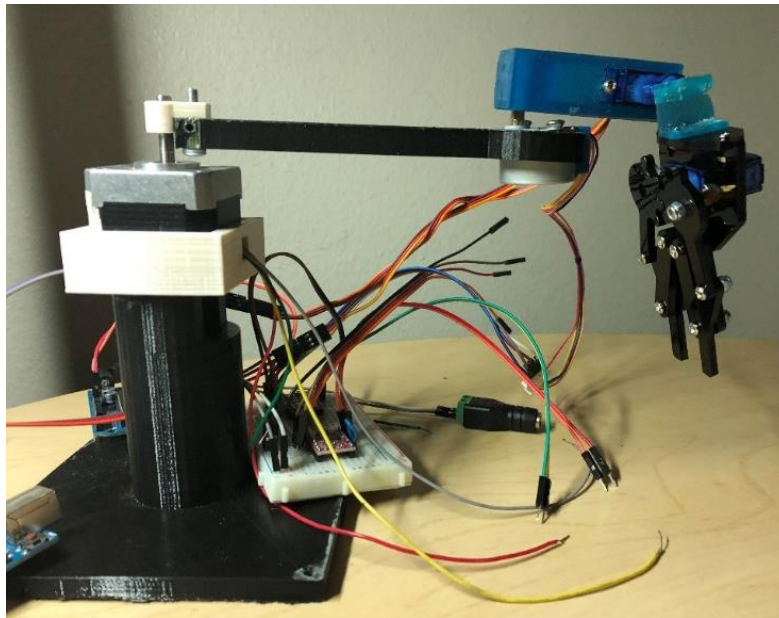
Solutions	A1 to B1		A1 to B2		A2 to B1		A2 to B2	
	Stepper Motor1	Stepper Motor2	Stepper Motor1	Stepper Motor2	Stepper Motor1	Stepper Motor2	Stepper Motor1	Stepper Motor2
Steps	-20	0	9	-32	-50	32	-20	0

These are four ways for robot rotates from Point A to Point B: A1 to B1, A1 to B2, A1 to B1 and A1 to B2. The Table 10 shows how much steps two stepper motors need to be rotated. (“-” means rotate in clockwise rotation) from one known Points A to Point B.

## 5 Mechanical Design

### 5.1 Dimension of the SCARA Robot

The dimension of this robot is 35cm x 12cm x 16cm. See the 3D view of SCARA Robot at Appendix D. All the components except the gripper are printed by 3D printer in an effective way. See the technical drawings of SCARA Robot in Appendix A. The following is overview of the SCARA Robot:



*Figure 19 Scheme of SCARA Robot*

### 5.2 Power for SCARA Robot

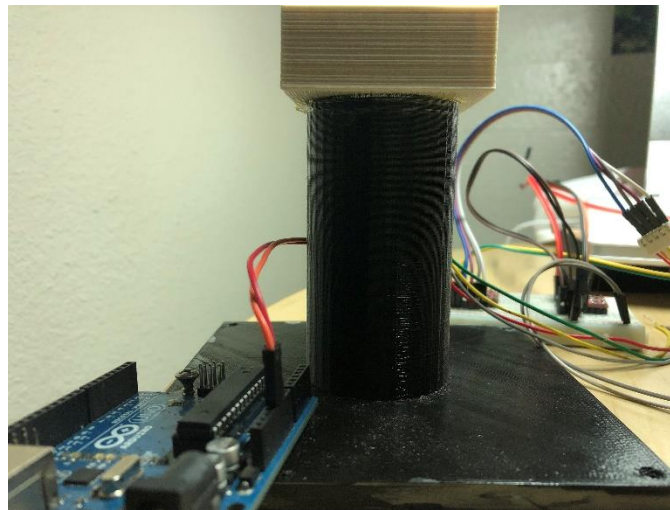
As shown in Figure 18, I need two power for two Arduino Uno. One of Arduino Uno boards is powered by AC Adapter that has adjustable output from 3 volts to 12 volts and 2.5A. It controls two stepper motors through A4988 stepper driver. Another Arduino Uno board is powered by computer through USB cable. This board controls two servo motors that are responsible for controlling end effector and gripper.



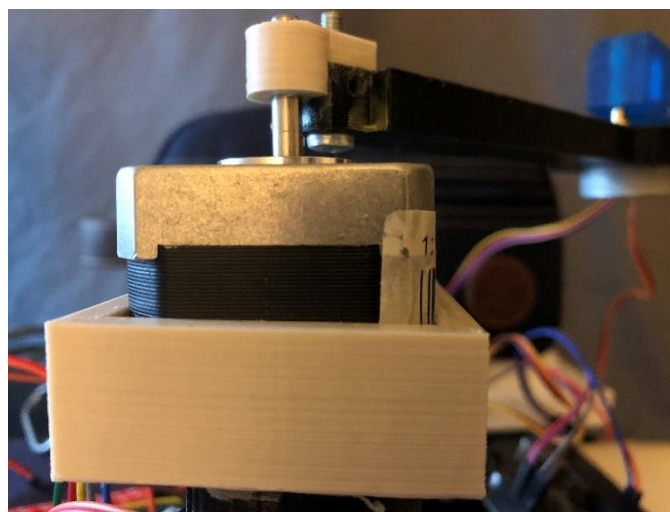
## 5.3 The Components Design

### 5.3.1 The Bases

There are two bases in this project. One of the bases is the black base which supports the second base and keep it in 80mm away from desk for tasks requirement. The dimension of the quadrangular and is 120mm x 120mm x 8mm and the dimension of the cylinder is 80mm high with base radius of 43mm. The NEMA 17 stepper motor is imbedded in the second base. See the technical drawing of base in Appendix A.



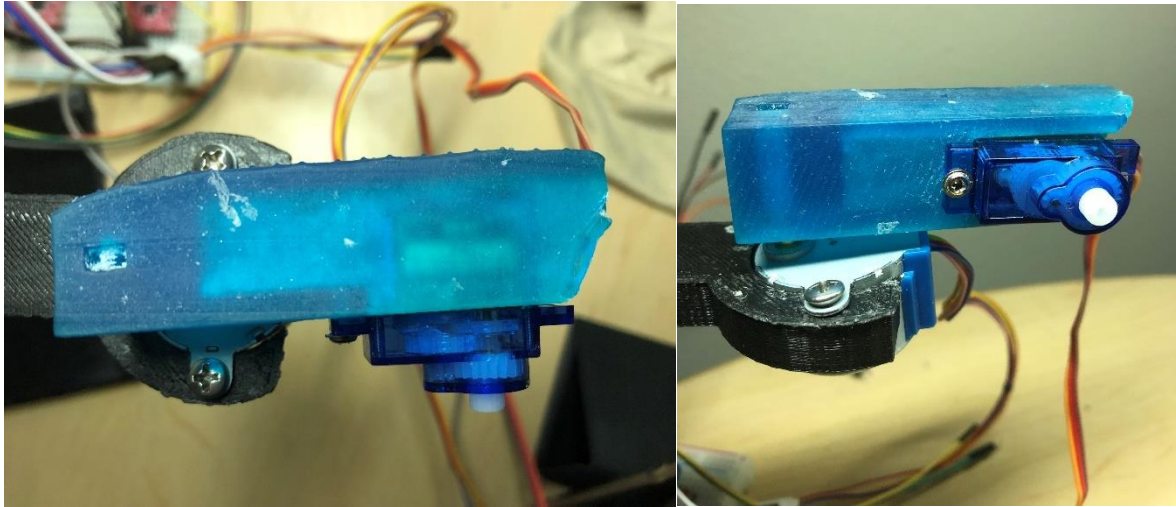
*Figure 20 The base of SCARA Robot*



*Figure 21 The connection between white base2 and NEMA stepper motor*

### 5.3.2 The links

There are two links in SCARA Robot. One of the links, the dimension is 140mm x 20mm x 10mm, connects the NEMA 17 stepper motor and 28 BYJ-48 stepper motor, another link (70mm x 20mm x 20mm) connects 28 BYJ-48 stepper motor and SG90 servo motor. See the technical drawing of base in Appendix A.



*Figure 22 The Second link*

### 5.3.3 The Gripper

The material of the gripper is plastic. In order to avoid slip between gripper and cube, rubber sticks are used in the gripper to increase the friction shown in Figure 27. The rubber stick is able to make the gripper easy to pick and tightly hold wood cube when robot moves. The gripper can open its fingers from 0 degree to 180 degrees in horizontal direction. When the gripper is closed, distance from tips of fingers (Point B) to its base (Point A) is 50mm shown in Figure 23 and distance from base (Point A) to one third of fingers (Point C) is 60mm shown in Figure 24. When the gripper is opened in maximum status the distance from base (Point A) to the Point D is 70mm shown in Figure 25. Gripper can pick the wood cube (20mm x 20mm x 20mm) from at least 60mm to but no more than 80mm from base shown in Figure 26. See the technical drawing of base in Appendix A. And see the shape of gripper controlled by servo motor in different angle input in Appendix C.

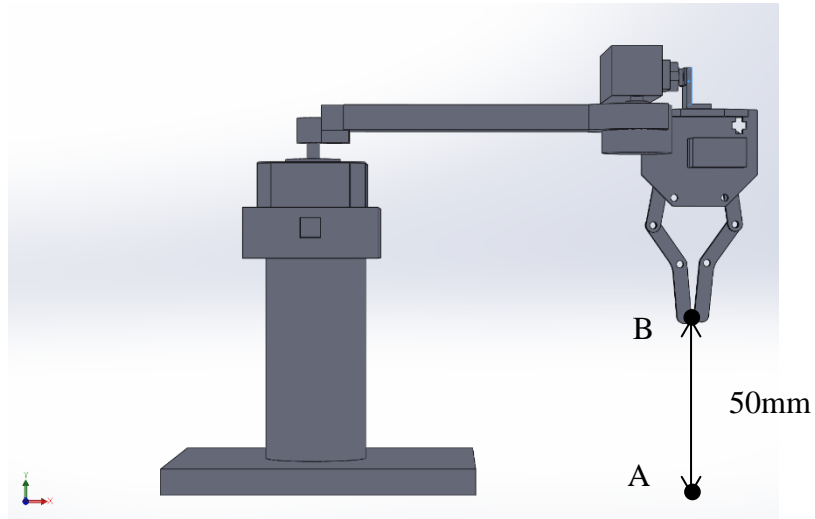


Figure 23 The front view of SCARA Robot

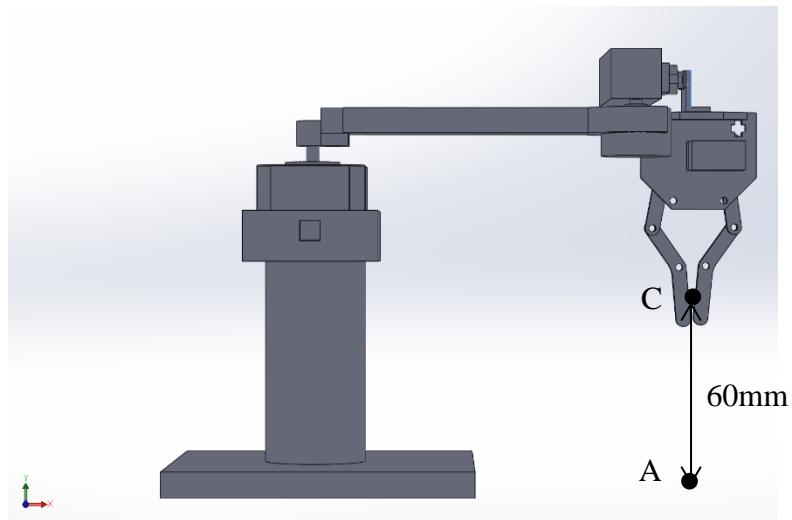


Figure 24 The front view of SCARA Robot

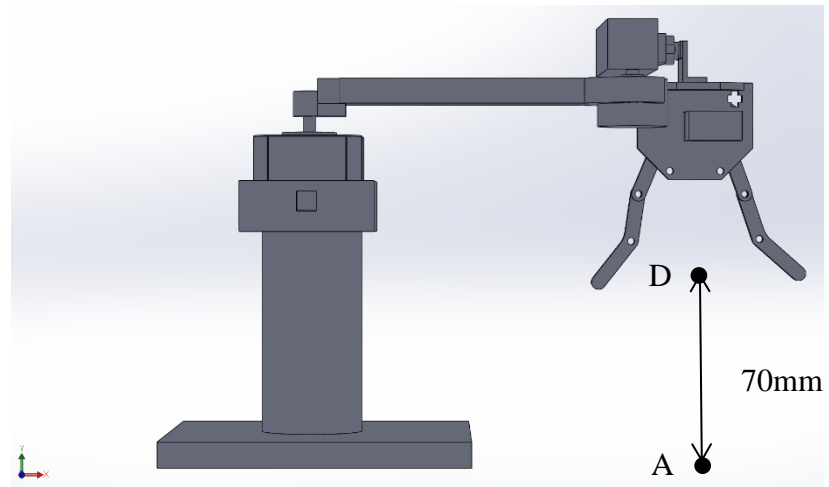


Figure 25 The front view of SCARA Robot

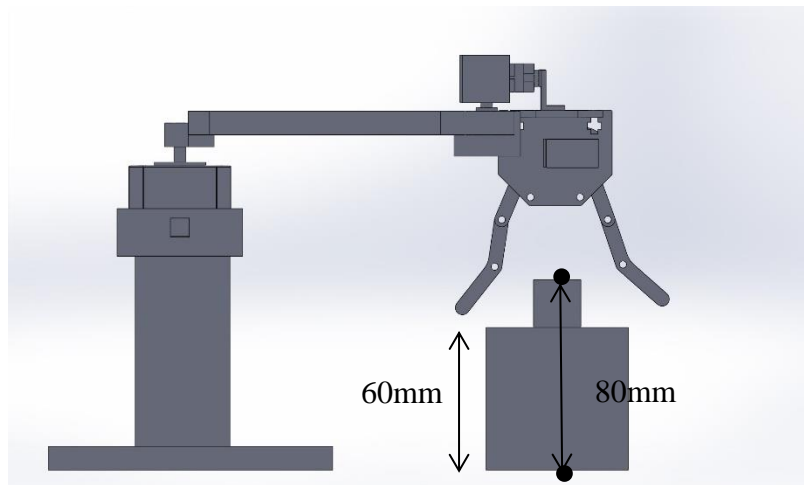
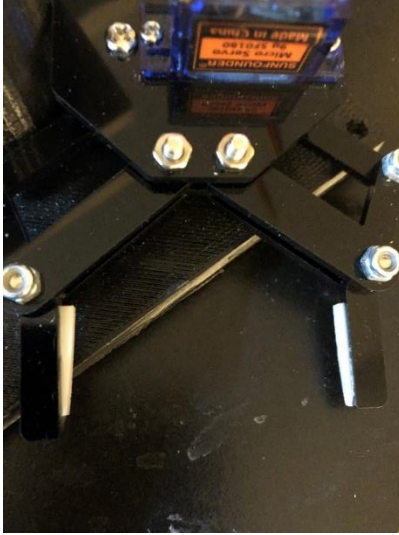


Figure 26 The front view of SCARA Robot



*Figure 27 The gripper with rubbers*

## 6 Coding

### 6.1 Pseudocode for stepper motor

Objective: Control two stepper motors with motor driver board A4988;

Include the header files <ros.h>, <Arduino.h>, <std\_msgs/Empty.h>, <std\_msgs/Int8.h>, "A4988.h";

Configure the pins connected with A4988 stepper motor driver board and Arduino Uno;

Start the node;

Define the A4988 with stepper motor1 and stepper motor2;

Control the stepper motors with digital. Write (0, HIGH) and digital. Write (0, LOW);

Build two subscribers, one for stepper motor1 and another for stepper motor2, communicate with other nodes through ROS Master;

Initialize ROS node in order to inform topics being published and listen topics you are interested in;

Subscribe the topics of motor1 and motor2 for get the information from these two topics.

### 6.2 Pseudocode for servo motor

Objective: Control two servo motors. Connect pin9 with servo1 and pin10 with servo2

Include the header files : <Arduino.h>,

<Wprogram.h>,<Servo.h>,<ros.h>,<std\_msgs/MultiArrayLayout.h>,

< std\_msgs/MultiArrayDimension.h>,< std\_msgs/UInt16MultiArray.h >;

Then starting the node;

Define servo motors: servo1 and servo2;

Set servo motors angle by `servo1.write()` and `servo2.write()`;

Build a subscriber with a topic named `servo`, which communicate with other node through ROS Master;

## 7 Conclusion

In this project, the 4 degree of freedom SCARA Robot was design and constructed. Two stepper motors are responsible for the links as actuators to make the end-effector to get the place I want. Two cheap servo motors are responsible for picking and holding wood cube. All components are printed by 3D printer and ROS is easy to study, which is comfortable for people who are interested in designing robots.

In the future, I will try to improve stability of the robot and build a close loop system for the SCARA robot to complete the repetitive task to replace workers who are working on assemble line.



## 8 Reference

- [1]. Jason M.O' Kane. A Gentle Introduction to ROS 2014. Columbia, SC.2014, P.4.
- [2]. Kevin M. Lynch and Frank C. Park. Modern Robotics mechanics, planning, and control. November 29, 2016
- [3]. Lentin Joseph. Mastering ROS for Robotics Programming. December 2015
- [4]. Claudio Urrea and John Kern. Design, simulation and comparison of controller for a redundant robot. December 2015
- [5]. Claudio Urrea and Juan Cortes and Jose Pascal. Design, construction and control of a SCARA manipulator with 6 degrees of freedom. December 2016
- [6]. <https://www.jsumo.com/sg90-micro-servo-motor>
- [7]. [https://www.pololu.com/file/0J450/a4988\\_DMOS\\_microstepping\\_driver\\_with\\_translator.pdf](https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf)
- [8]. <https://www.buildyourcnc.com/images/nema17-62oz-in-motor-datasheet.JPG>
- [9]. <https://solarbotics.com/product/22310/>
- [10]. <https://store.arduino.cc/usa/arduino-uno-rev3>
- [11]. [http://wiki.ros.org/rosterial\\_arduino/Tutorials/Arduino%20IDE%20Setup](http://wiki.ros.org/rosterial_arduino/Tutorials/Arduino%20IDE%20Setup)
- [12]. Tesi di Laurea. Matlab-based Control of a SCARA Robot. 2014/2015
- [13]. Claudio Urrea, John Kern. Case Studies in Mechanical System and Signal Processing. December 2015

[14]. [http://rraj.rsj-web.org/en\\_atcl/842](http://rraj.rsj-web.org/en_atcl/842)

## 9 Appendix A Technical Drawings of SCARA Robot

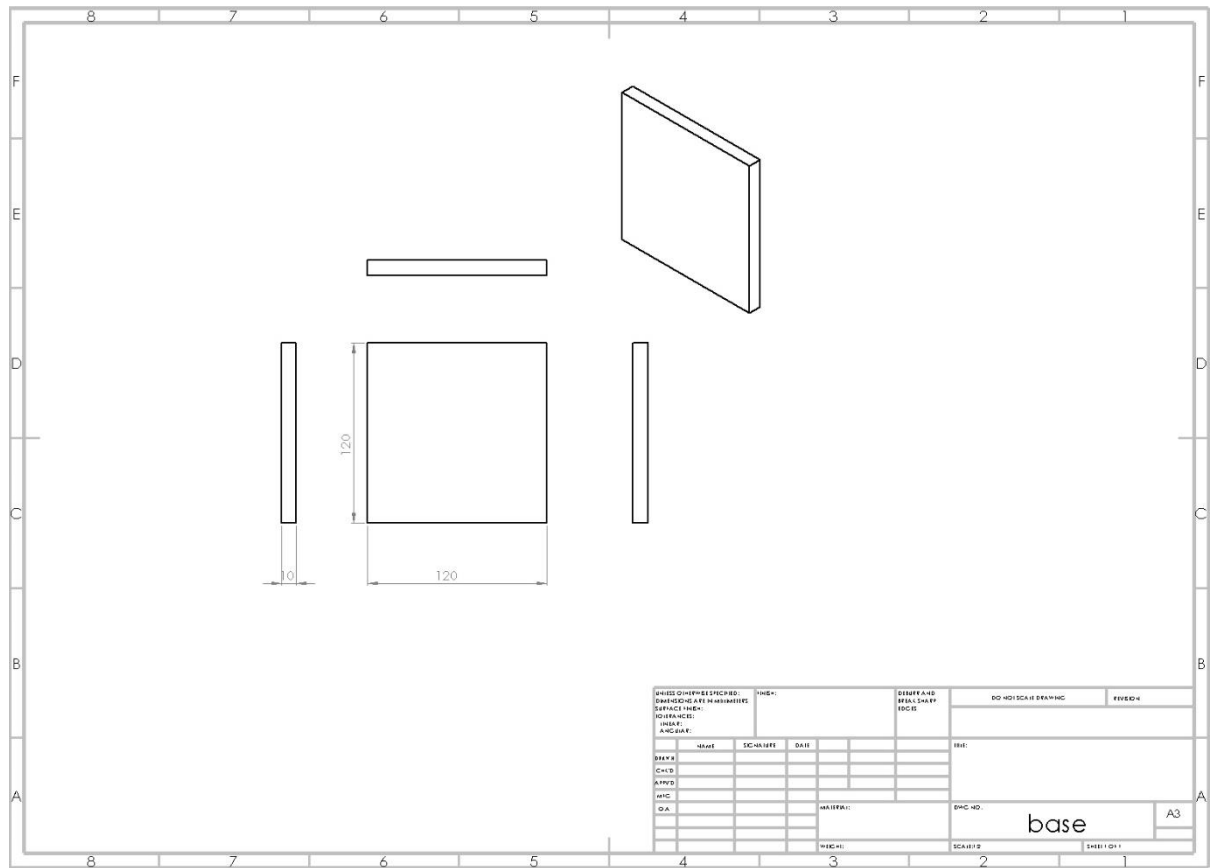
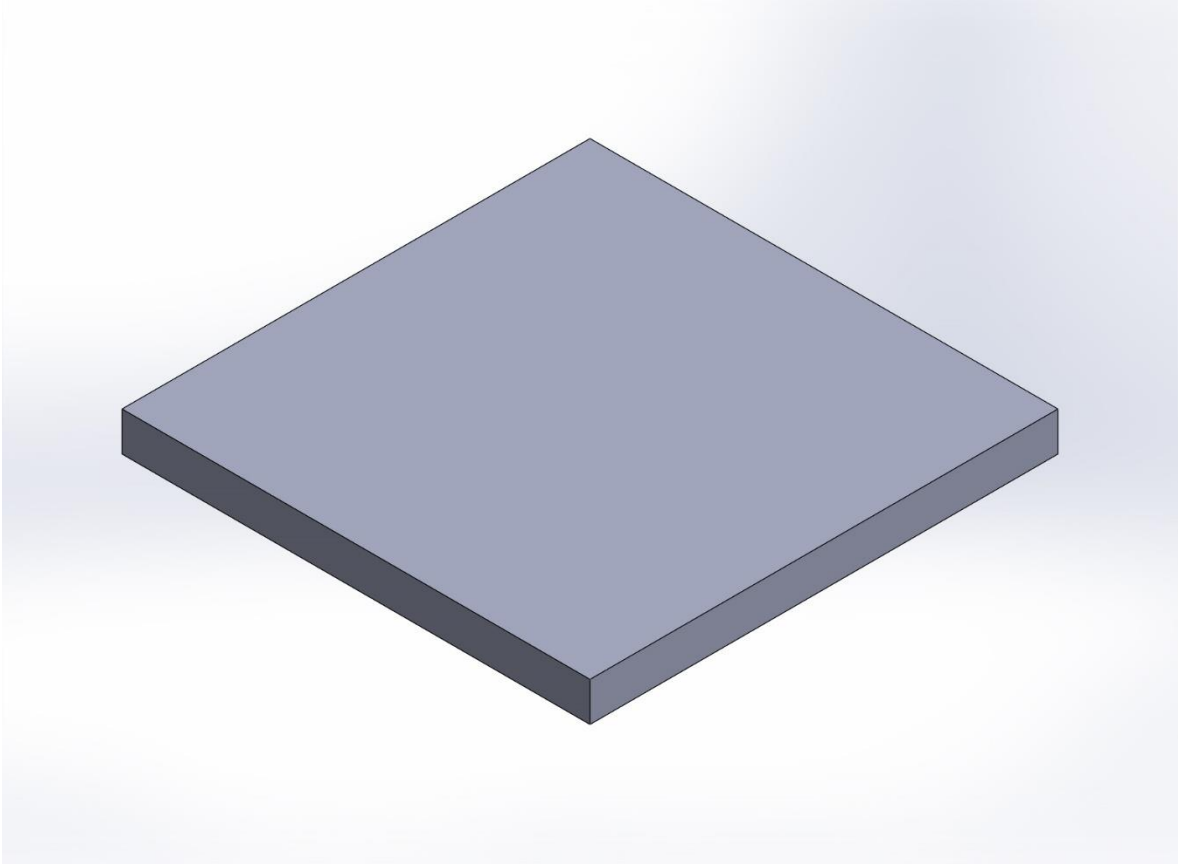
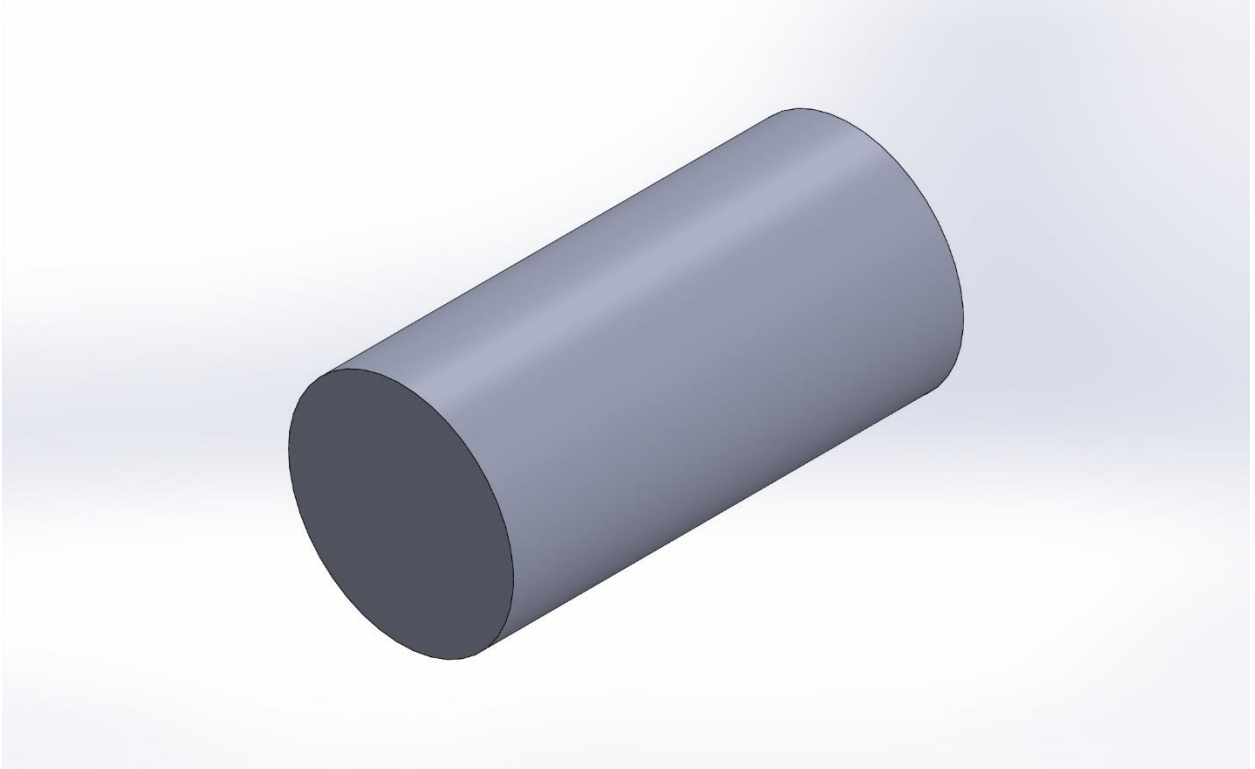


Figure 28 Technical drawing of Base1



*Figure 29 3D view of Base1*





*Figure 31 3D view of Base2*

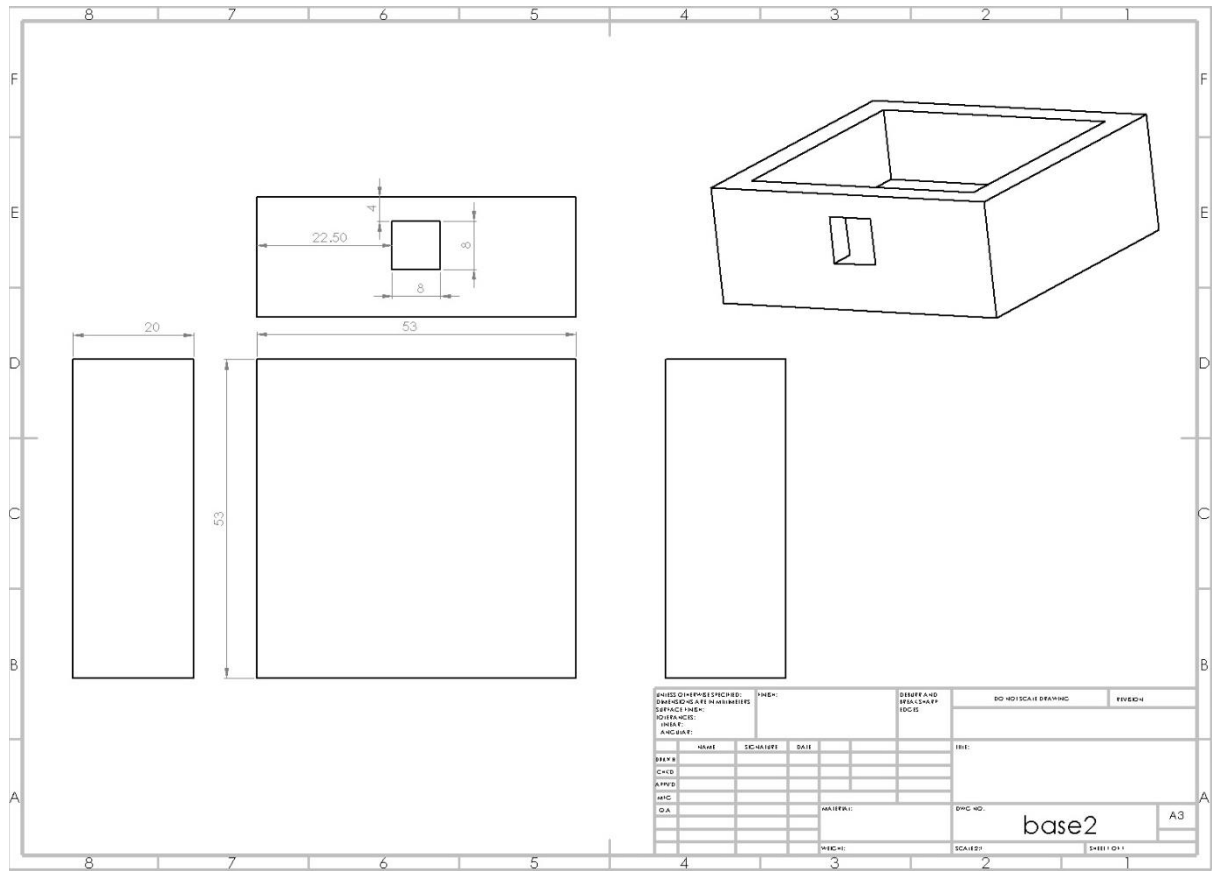
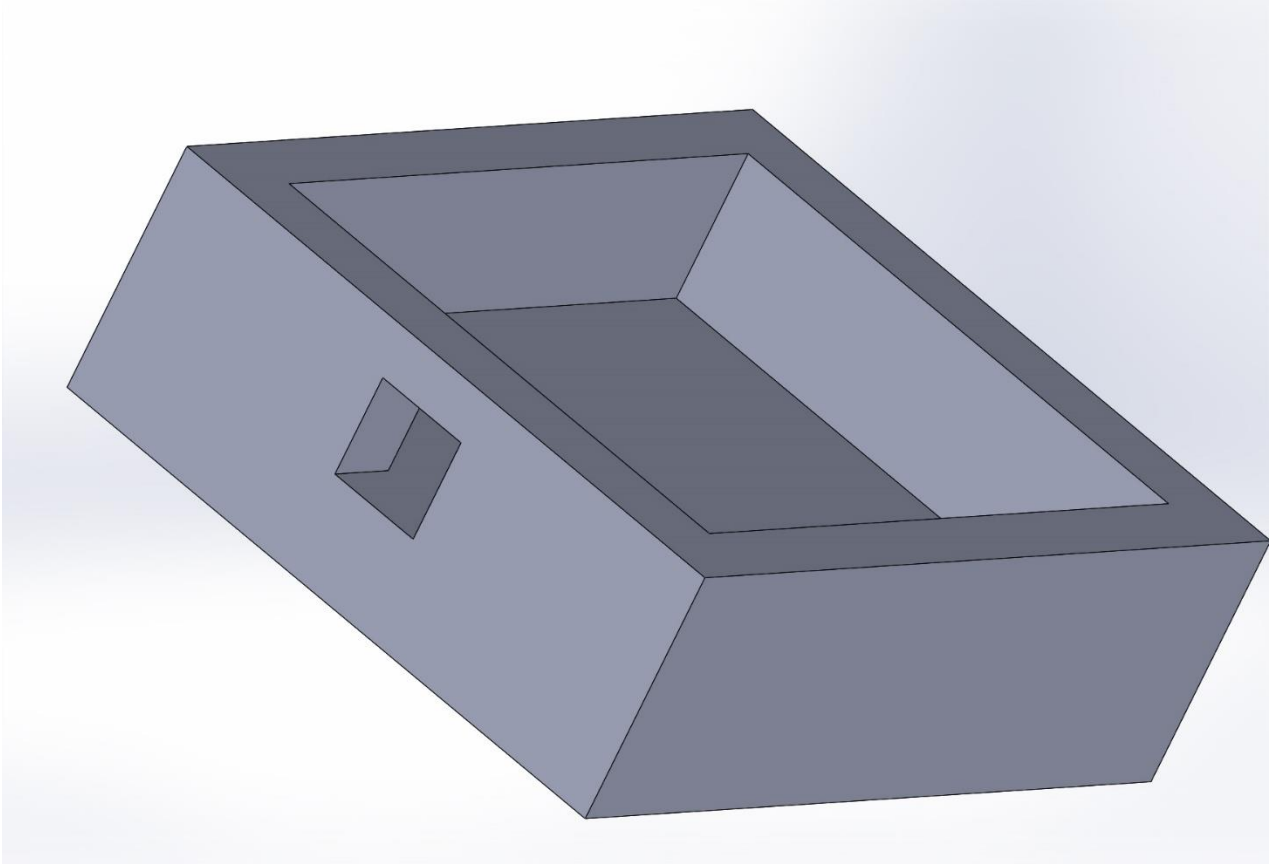


Figure 32 Technical drawing of Base3



*Figure 33 3D view of Base3*



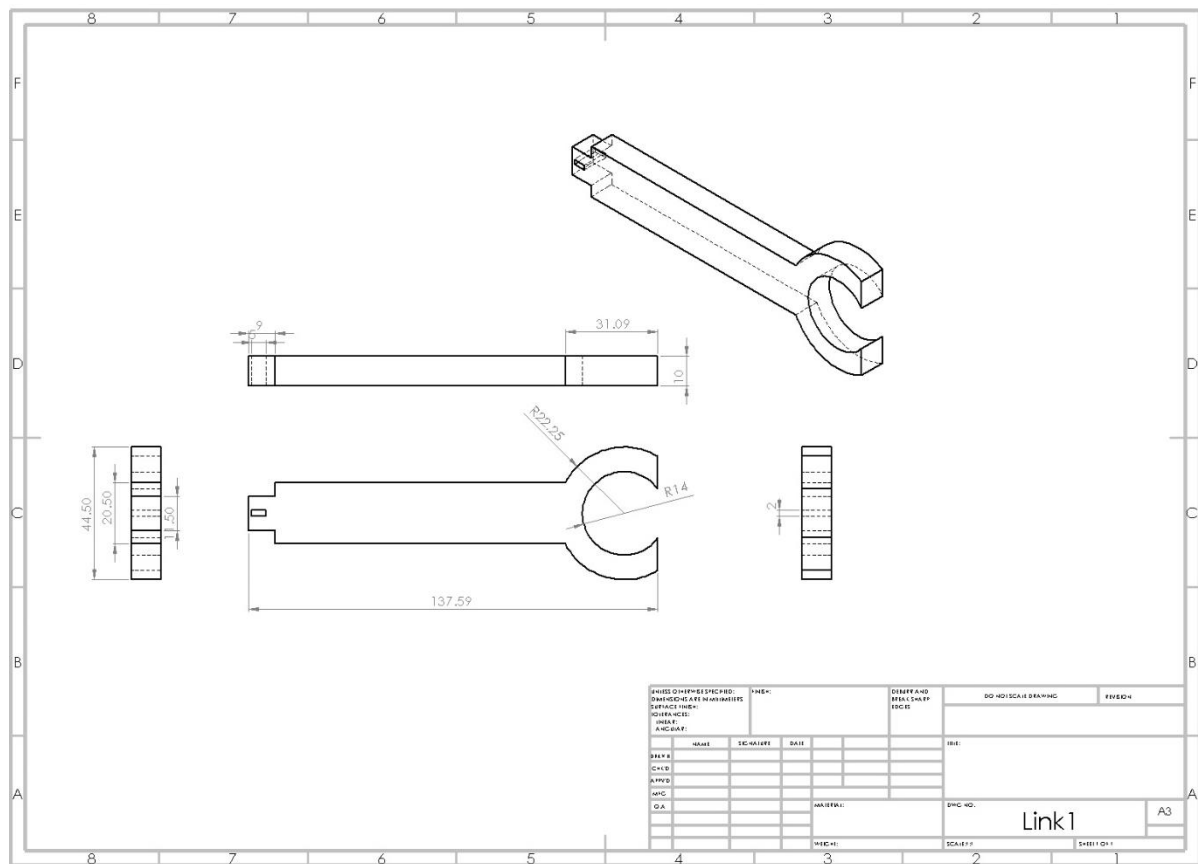
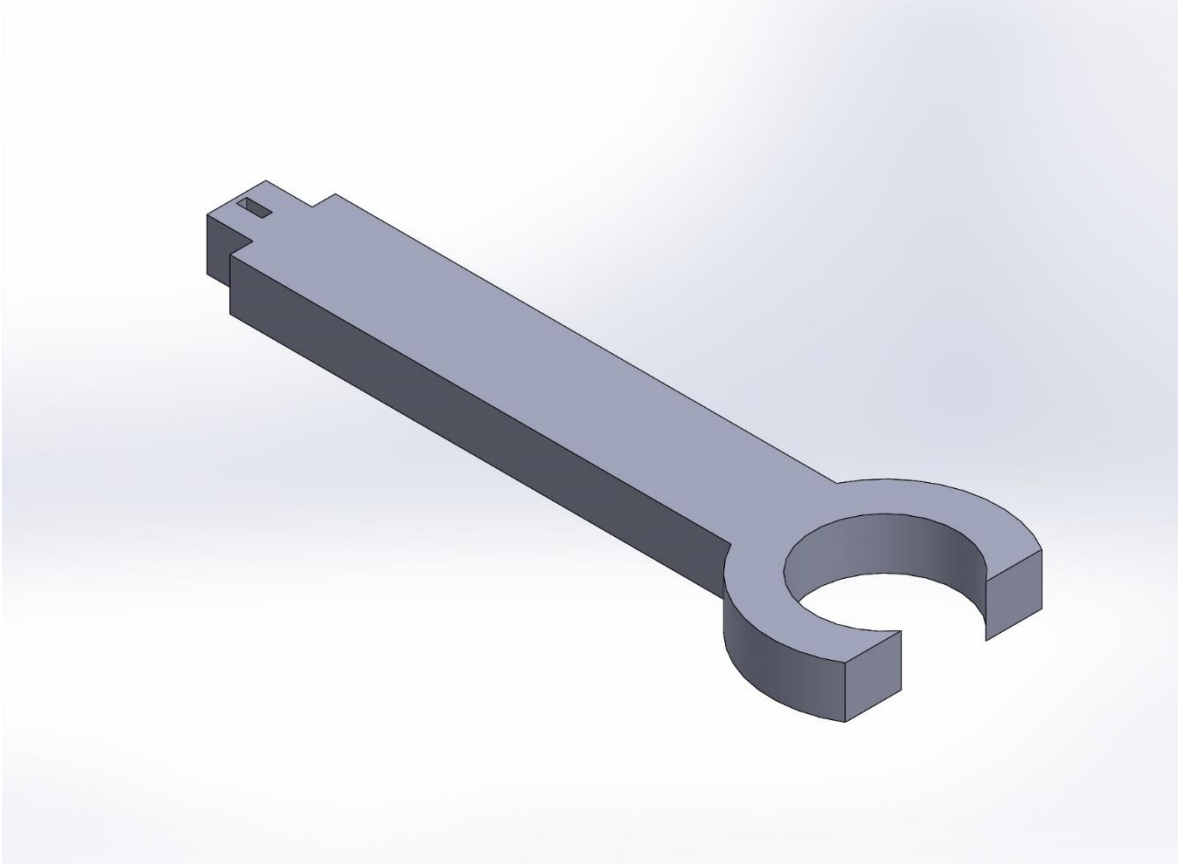
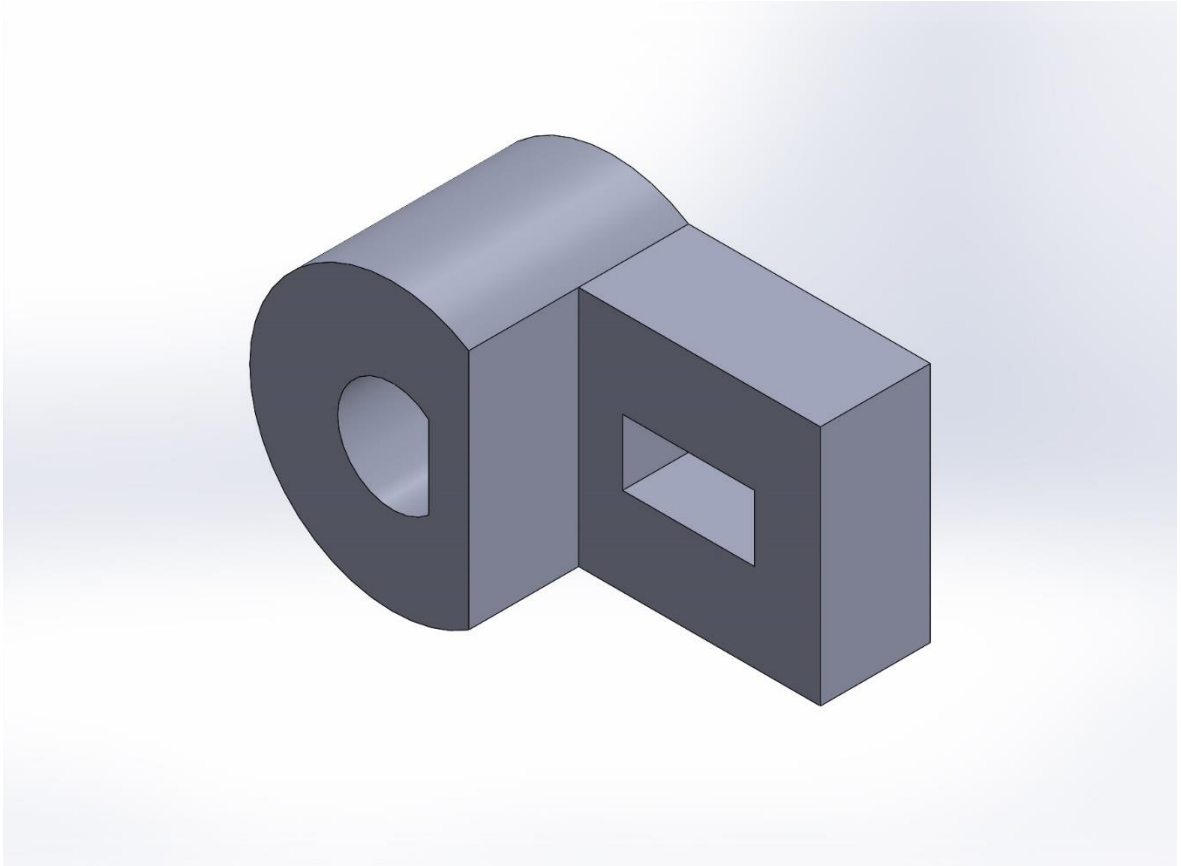


Figure 34 Technical drawing of Link1

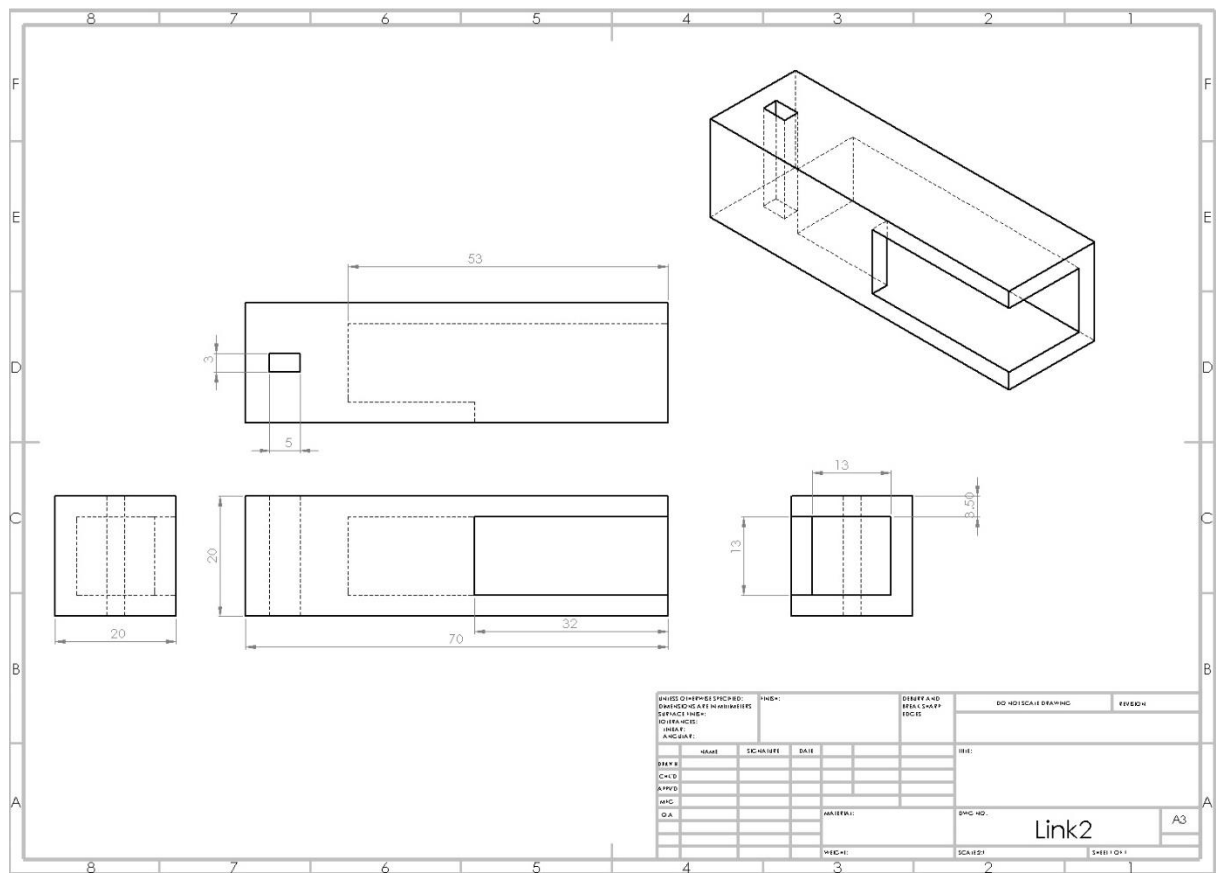


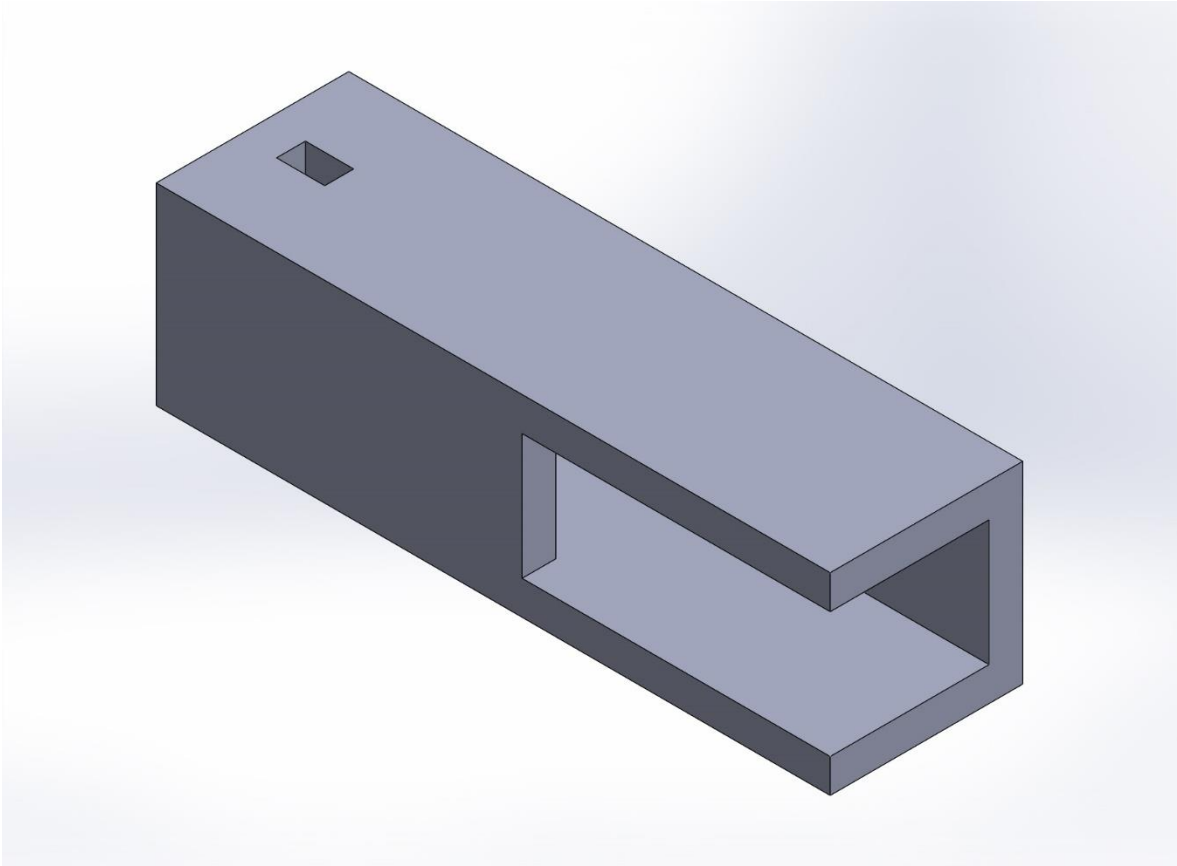
*Figure 35 3D view of Link2*





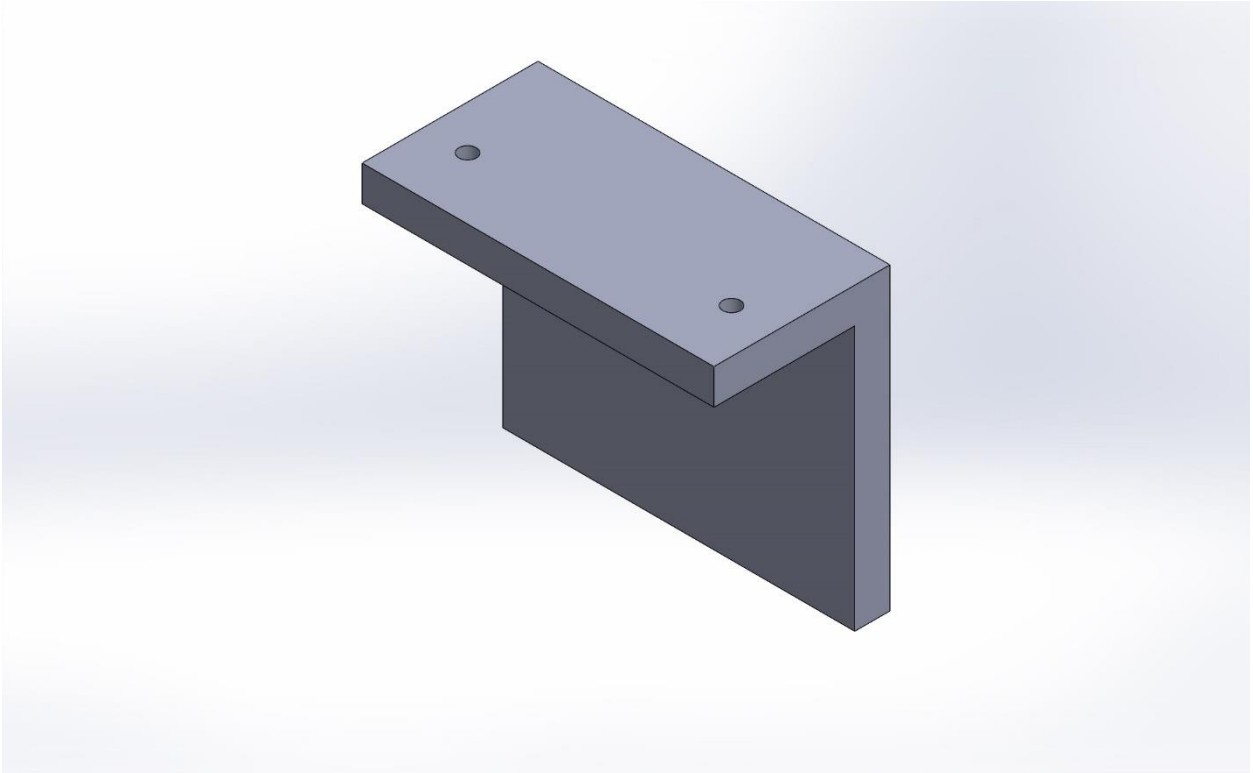
*Figure 37 3D view of connection component*





*Figure 39 3D view of Link2*

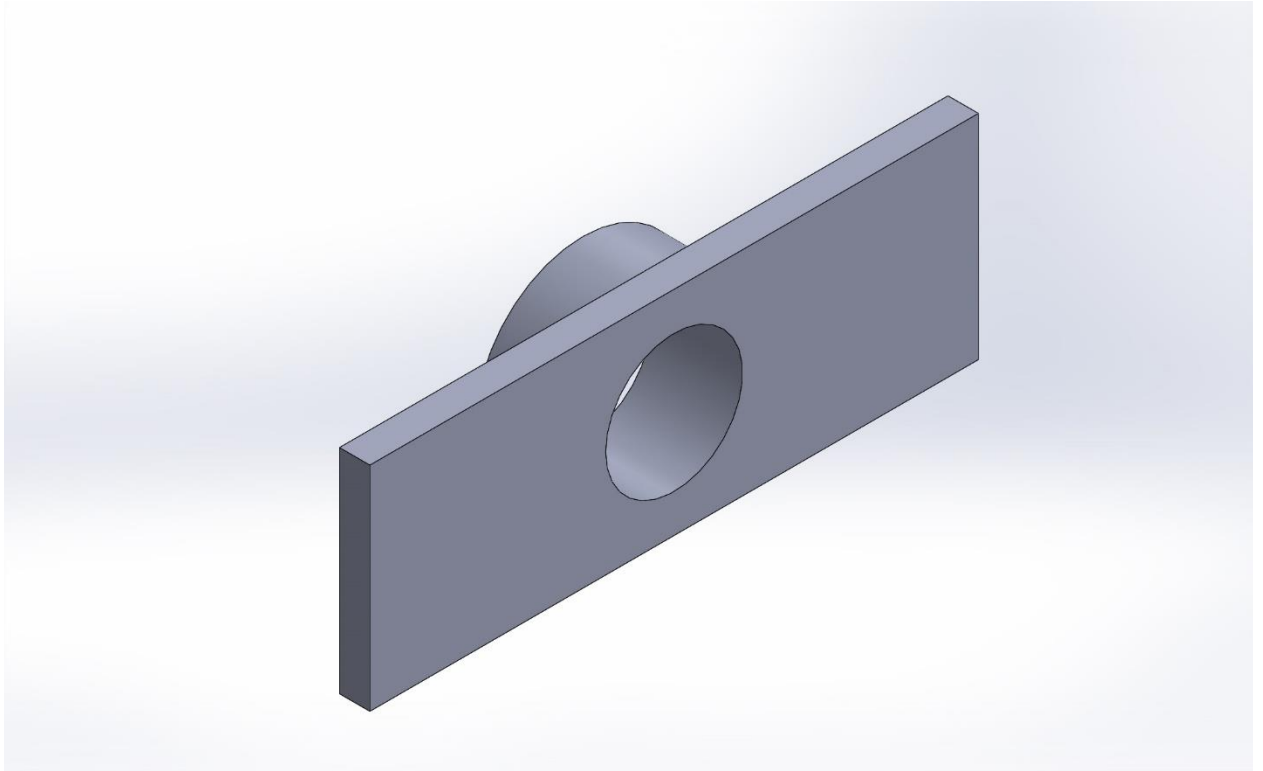




*Figure 41 3D view of Link3*







*Figure 43 3D view of Link4*

## 9.1 Code for stepper motor

```
#include <ros.h>

#include <Arduino.h>

#include <std_msgs/Empty.h>

#include <std_msgs/Int8.h>

#include "A4988.h"

#define MOTOR_STEPS 200

#define STEP1 3

#define DIR1 4

#define STEP2 5

#define DIR2 6


ros::NodeHandle nh;


A4988 stepper1(MOTOR_STEPS, DIR1, STEP1);

A4988 stepper2(MOTOR_STEPS, DIR2, STEP2);


void startMotor1(const std_msgs::Int8& rpm){

  digitalWrite(0, HIGH);

  stepper1.begin(rpm.data, 16);

  stepper1.rotate(360);

  digitalWrite(0, LOW);

}
```

```

void startMotor2(const std_msgs::Empty& toggle_msg){

    digitalWrite(1, HIGH);

    stepper2.begin(15, 16);

    stepper2.rotate(360);

    digitalWrite(1, LOW);

}

ros::Subscriber<std_msgs::Int8> motor1("motor1/start", &startMotor1);

ros::Subscriber<std_msgs::Empty> motor2("motor2/start", &startMotor2);


void setup() {

    pinMode(0, OUTPUT);

    pinMode(1, OUTPUT);

    nh.initNode();

    nh.subscribe(motor1);

    nh.subscribe(motor2);

}


void loop() {

    nh.spinOnce();

    delay(1);

}

```

## 9.2 Code for Servo Motor

```
#if (ARDUINO >= 100)
#include <Arduino.h>
#else
#include <WProgram.h>
#endif

#include <Servo.h>
#include <ros.h>
#include "std_msgs/MultiArrayLayout.h"
#include "std_msgs/MultiArrayDimension.h"

#include "std_msgs/UInt16MultiArray.h"

ros::NodeHandle nh;

Servo servo1;
Servo servo2;

void servo_cb( const std_msgs::UInt16MultiArray& cmd_msg){
    servo1.write(cmd_msg.data[0]); //set servo angle, should be from 0-180
    servo2.write(cmd_msg.data[1]);
    digitalWrite(13, HIGH-digitalRead(13)); //toggle led
}

ros::Subscriber<std_msgs::UInt16MultiArray> sub("servo", servo_cb);

void setup(){
    pinMode(13, OUTPUT);

    nh.initNode();
    nh.subscribe(sub);

    servo1.attach(9); //attach it to pin 9
    servo2.attach(11); //attach it to pin10
}

void loop(){
```

```
nh.spinOnce();  
delay(1);  
}
```

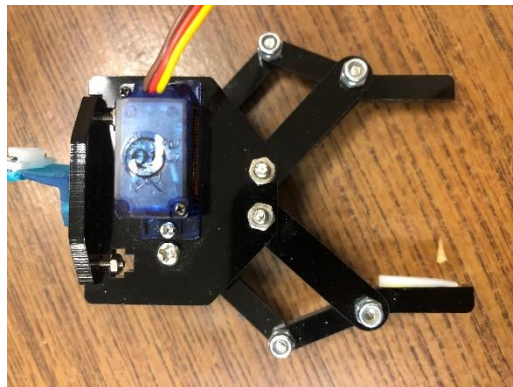
## 10 Appendix C Shape of gripper



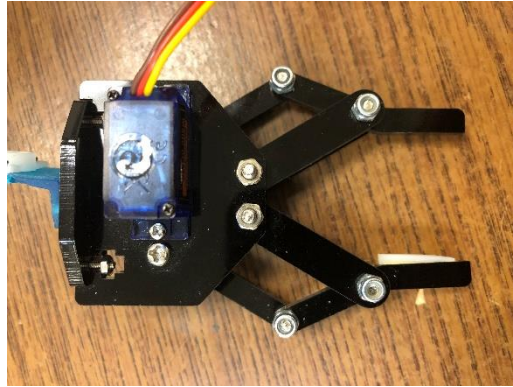
*Figure 44 Running motor with 30 degrees*



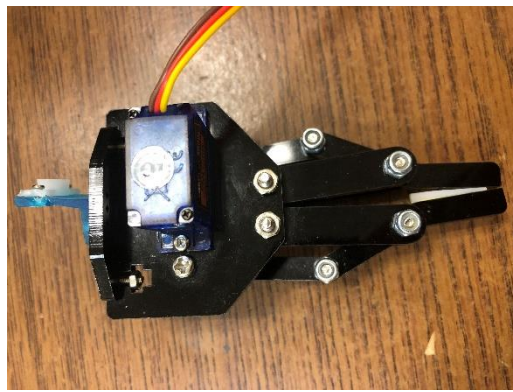
*Figure 45 Running motor with 60 degree*



*Figure 46 Running motor with 90 degree*



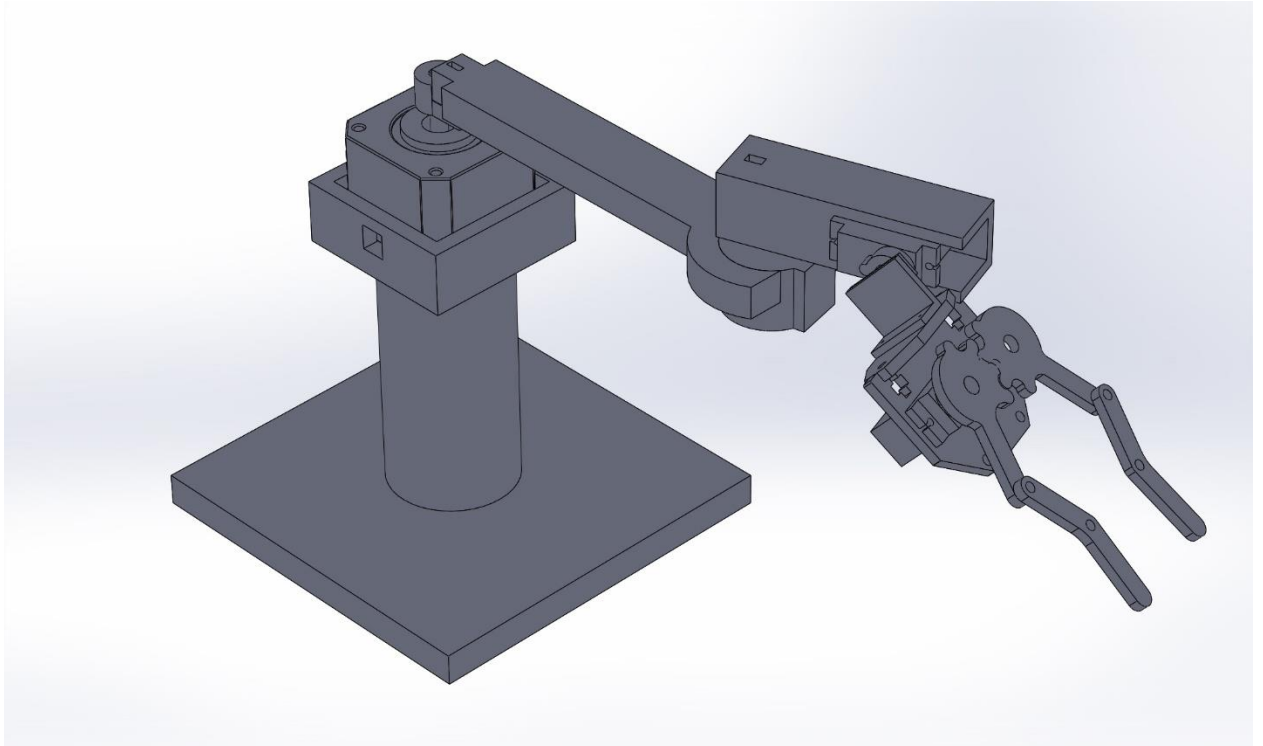
*Figure 47 Running motor with 120 degree*



*Figure 48 Running motor with 150 degree*



## 11 Appendix D Overview of SCARA Robot in 3D



*Figure 49 3D view of SCARA Robot*