CHAPTER 1

INTRODUCTION

1.1 Concept of Project

Selective Compliance Assembly Robot Arm or SCARA robot was invented by Hiroshi Makino in 1978 is shown in Figure 1.1 Its arm is rigid in the Z-axis and pliable in the XY-axes, which allowed it to adapt to holes in the XY-axes . This was and still a revolutionary robot in many industries because of its speed and simple design.

In many industrial applications that require fast picking and placing operations, there is a need for such a robot because the human workers cannot handle these “boring” missions. SCARA robot is one of the options to do these missions accurately, fast and time efficient with less cost than other types of robots.

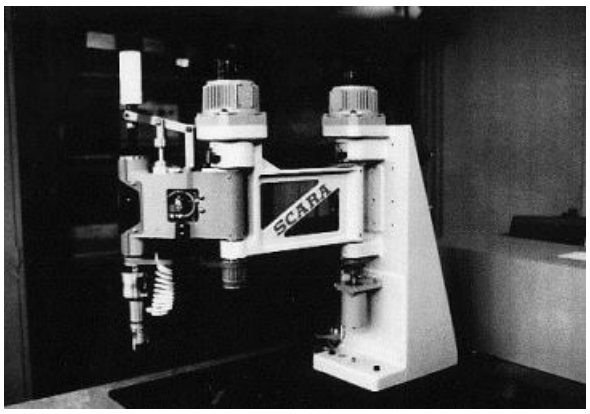


Figure 1 : SCARA robot prototype made by Hiroshi Makino

1.2 Recognition of The Need

Some operations need to sort objects by their color, shape or both faster than human labors, therefore there is a need to construct a high speed and accurate robotic arm to do such missions. Some of these sorting missions will be done on the horizontal plane, so that it is reasonable to construct a simple robotic structure to do these missions hence there is no need here to construct a complex robotic arm like German Aerospace Center (DLR) manipulator or anthropomorphic robotic arm, SCARA robotic structure is a good choice in this case.

On the other hand, some operations are strictly needed to be done by robotics systems like sensitive and small-scale assembly operations such as in PCB assembly lines. Also, some sorting missions may be needed to be done in dangerous environments.

1.3 Project Scope

This project aims to design, control and implement a 4DOF SCARA Robot to collect objects laied on horizontal plane and sort them into baskets by their colors with taking an advantage of image processing with computer vision within an $80 * 70 * 10\ cm$ working space.

The first stage of the project after determining the project limitations and assumptions is to design the manipulator. Then the mechanical structure of the manipulator will be modeled for the sake of controller design and simulation, more than one control algorithm will be used to control the manipulator to choose the most fitting controller for such a mission. The machine vision system will determine the pose of each object with respect to a reference frame, based on these poses the path planner will generate the trajectory that the manipulator has to track. The Robot is expected to sort objects based on their color with precise, accurate repeatable picking and placing missions.

1.4 Project Objectives

1. Optimize the mechanical structure of the robot to reduce the actuators energy consumption by setting the motors positions as close as possible to the base of the robot, in other words.
2. To simplify the poses sensing operation by taking advantage of computer vision system.
3. Build a robust and reliable SCARA robot.
4. To build an intelligent automated sorting system.

1.5 Methodology

The methodology to make this project contains several stages, theoretical calculations, which contains the dynamic model and design the mechanical structure for the robot.

After the theoretical calculations, several conceptual designs will be considered to show the connections between several components of the project (computer, interfacing circuits, controller, drivers and motors) and then choose the best design. After that, the mechanical structure will be modeled in SOLIDWORKS software to check the maximum deflection that occurs due to the maximum static and dynamic forces, after that, MATLAB and Simulink software will be used to simulate its motion and its behavior according to the dynamic model and the desired controller.

1.6 Literature Review

This section provides an overview of previous efforts of the industrial robot and construct selective compliance assembly robot arm " SCARA Robot".

1.6.1 The UNIMATE Model, 1979

In 1961 - The first industrial robot was online in a General Motors automobile factory in New Jersey it was called UNIMATE, it used to pick and place and in assembly lines Figure 2. However, as computer development began, the developers were starting using the computer to control the robots.

In 1979 the researchers developed the UNIMATE, it has used image processing with a special camera called TV camera, it takes the picture between 10 to 66 milliseconds where the required as the research is between 100 to 500 milliseconds, so the image processed slower. The TV camera moved with respect to reference frame to detect the pose of the object. Also, it used for visual feedback to control the motion of a robot, this method is called visual serving.

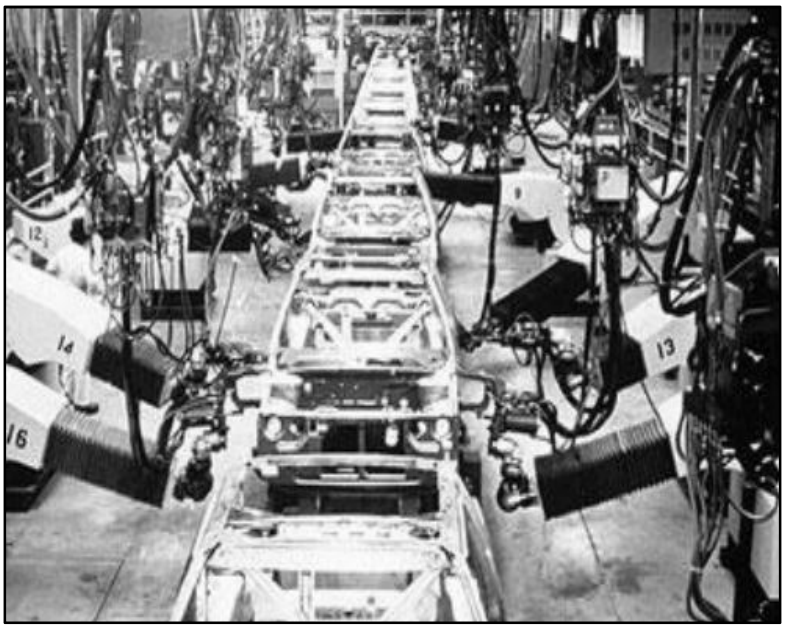


Figure 2: UNIMATE in assembly lines

The robot is actuated hydraulically until the actual reading of the joint encoders agree with the ordered set points, it was inserted by the program that runs in the LSI-11 microcomputer. UNIMATE is collect between two system power: electric and hydraulic power.

1.6.2 SCARA Robot First Model, 1978

The main idea of SCARA robots that is it has a simple structure with high speed, accurate and precise manipulation in the plane by taking advantage of the third prismatic joint for pick and place missions.

The first industrial application of the SCARA robot was in assembling printed circuit board (PCB) of audio amplifiers, where eight robots were used in this assembly line.

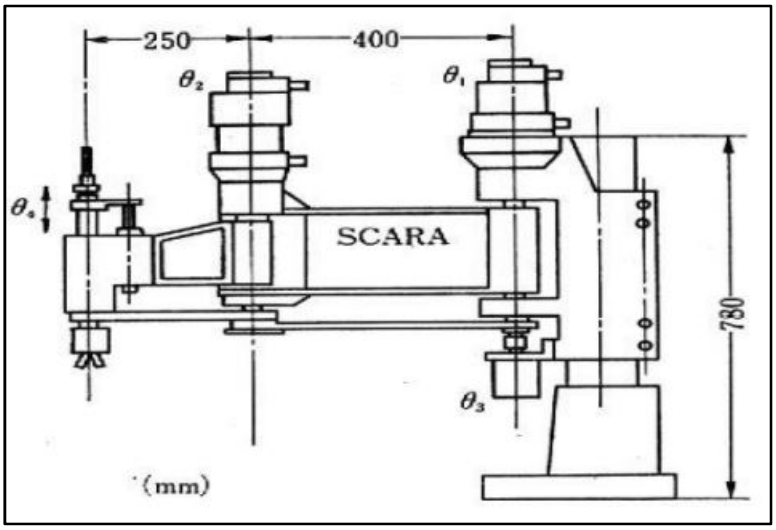


Figure 3: First SCARA detailed

1.6.3 The SCARA Robot Model, 2009

The researchers developed the SCARA robot to become more accurate and consumed low power by optimizing the mechanical structure for the manipulator. The 4-DOF SCARA shown in Figure 1.4, that is used for drilling task will be designed and developed using solid dynamic program and simulated using MATLAB and Simulink. Forward and inverse kinematics were derived using Denavit-Hartenberg notation .

The robot actuated using three electrical 24 V DC Servo motors and PD controller connect with SD program and MATLAB at real time.

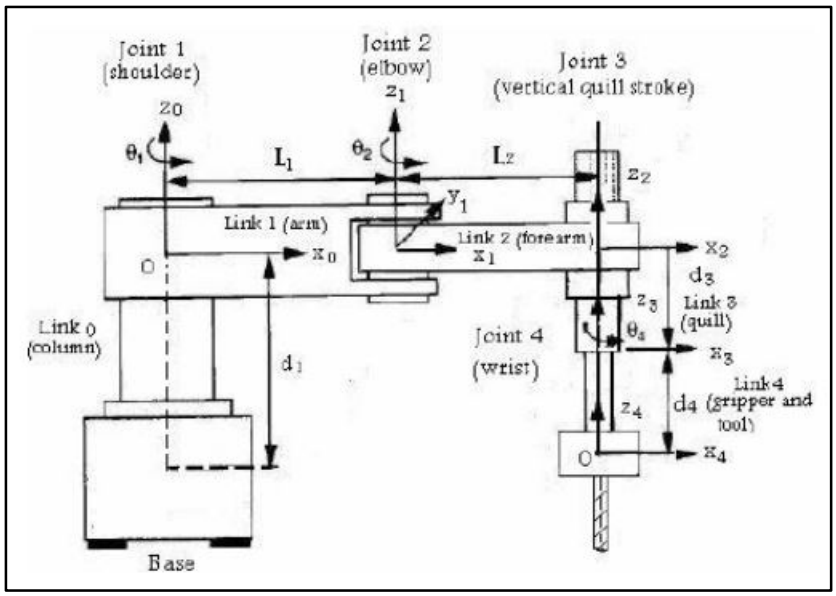


Figure 4: D-H Parameters for four- joint SCARA Robot.

1.6.4 The SCARA robot by resonance model, 2012

This model proposes an energy efficient method for pick and place tasks of SCARA robots. In the proposed method, an adaptive elastic device at each joint is effectively utilized to reduce the total energy of pick and place tasks. For practical pick and place tasks, start/end points must be changed depending on the required task ..

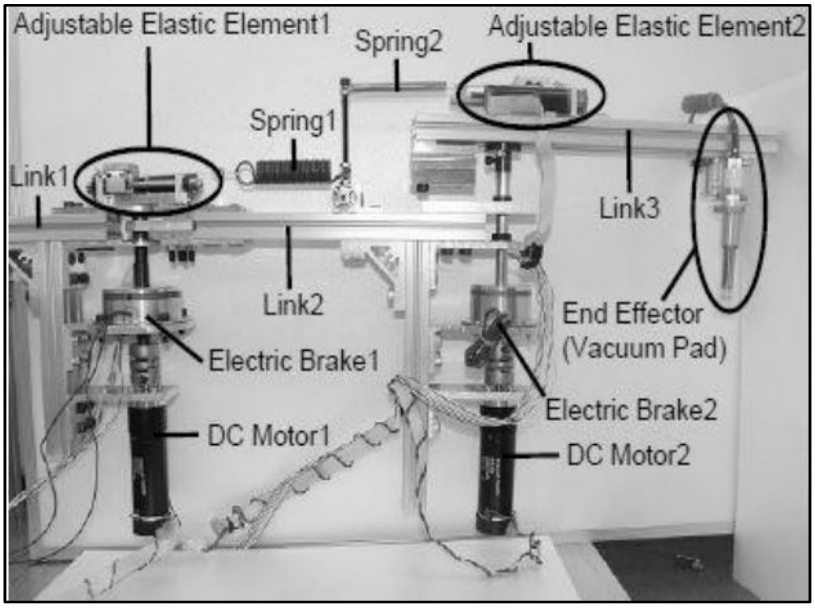


Figure 5: Front view of SCARA Robot

The robot with mechanical elastic elements can generate high energy efficient periodic motions by changing kinetic energy into potential energy stored in the elastic elements

This prototype shown in the Figure 5. The main component is electrical DC servo motor with gearbox with ratio (1:4.8), electric brake, and adjustable elastic element (spring) and end effectors (vacuum pad).

The prototype is used two power system: electrical system to actuate the Links (1, 2 and 3) and pneumatic system to actuate gripper (end-effectors).

Chapter 2

BILL OF MATERIALS

2.1 3d Printed Parts

- Waist.stl
- Shoulder.stl
- Elbow.stl
- WaistServoGear.stl
- WaistServoPrimeGear.stl
- WaistServoPrimeGearBase.stl
- ShoulderElowBase.stl
- EblowRod.stl
- ShouderRod.stl
- EblowUpper.stl
- ShouderMaintainer.stl
- ElbowUpperRod.stl
- Arm.stl
- Armo.stl
- Gripbase.stl
- GripClaw1.stl
- GripClaw2.stl
- GripClawGear.stl
- GripServoGear.stl

2.2 Non 3d Printed Parts

- 24 * 6mm Ball Bearings
- 5 * M3*25mm Bolts
- 5 * M3 Nuts
- 10 * M4 *25mm Bolts
- 10 * M4 Nuts

- 1 * M6*60 mm Threaded Rod
- 1 * M5*32mm Threaded Rod

2.3 Electronics Part

- 1 * Arduino Uno Board
- Male to Male Jumper Cable
- Male to Female Jumper Cable
- Female to Female Jumper Cable
- 3 * Joysticks modules
- 1 * ultrasonic module (HC-SR04)
- 3 * 180-Servo motor (SG 90)
- 3 * 360- Servo motor (MG-995)
- 1 * I2C-LCD Display
- 1 * Infrared Receiver
- 1 * Infrared Remote
- 1 * 5v volts dc power Adapter
- 1 * Incremental Rotary Encoder

Chapter 3

JOURNEY

3.1 Construction of SCARA Robotic Arm

1. Designing the components as per the requirements
2. Animating the Joints to check the motion og the robotic arm
3. 3d printing the mechanical Components.
4. Post processing the 3d Printed Parts.
5. Checking the electronics components with arduino with the test codes
6. Assembling the electronics components in the 3d printed parts.
7. Connecting the electronics components as per the cicuit.
8. Uploading Sketch file by using Arduino IDE.
9. Compiling the different Sketch files one by one to make the final Sketch

3.2 Design Modifications

- Reducing the Load on the servo motors with the concept mechanical Advantage.
- Giving the manual control with help of joystick
- Giving the wireless manual control with the help of IR Remote.
- In waist gear ratio is used to reduce the load.
- In shoulder & elbow the lever system is used to reduce the load.
- A operating system is developed to choose between IR remote, Pick & Place, Joystick Control, Description of the project
- Improvising the body structure so that it require less material to 3d print.

3.3 Final Look

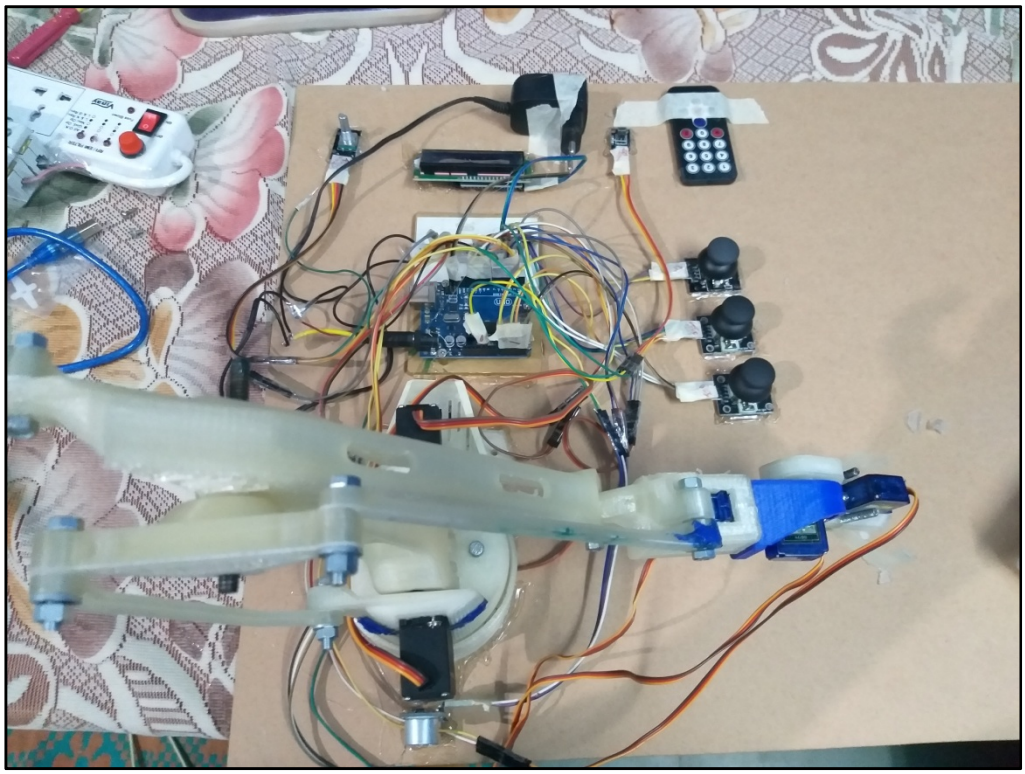


Figure 6: Photo taken after Completion of V2 Arm

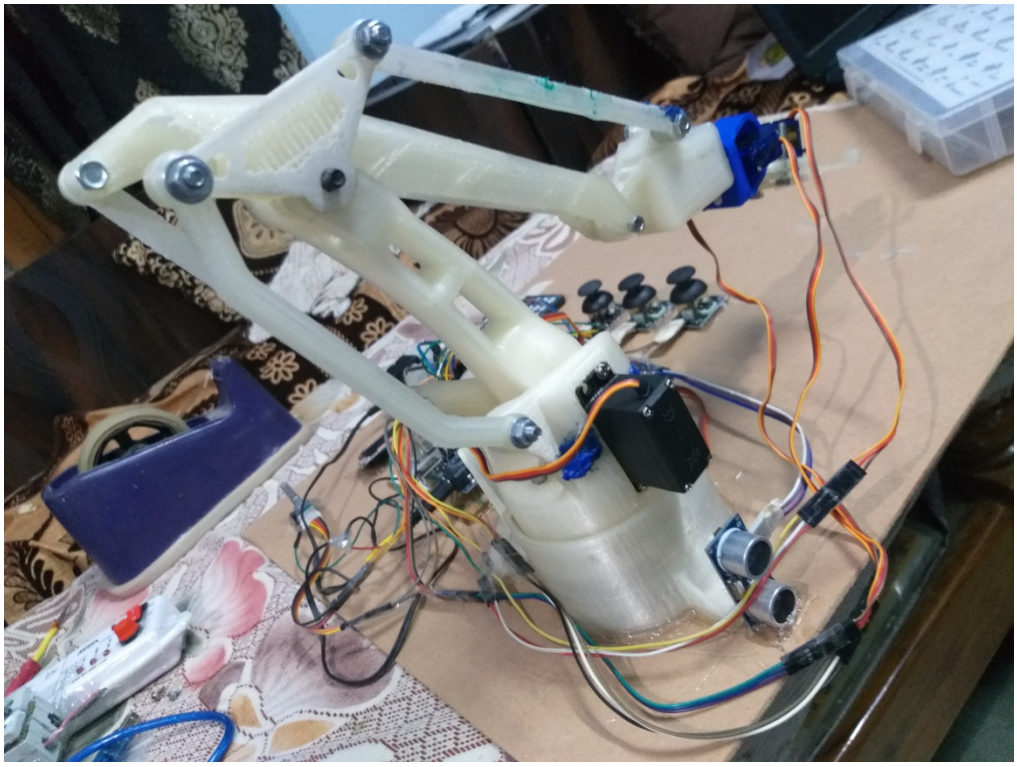


Figure 7: Photo taken after Completion of V2 Arm- 2

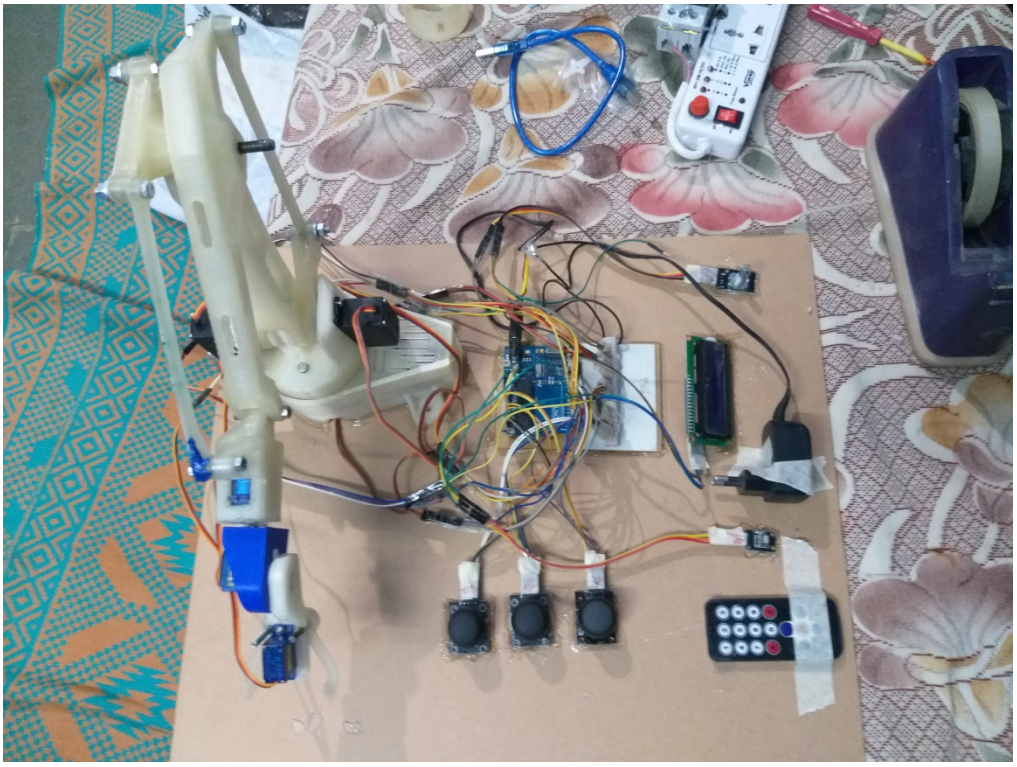


Figure 8: Photo taken after Completion of V2 Arm-3

Chapter 4

MECHANICAL DESIGN

4.1 Version -1 Design

4.1.1 3d Design

Design of the SCARA robot project is created on the Autodesk free and easy designing software Autodesk Fusion 360.

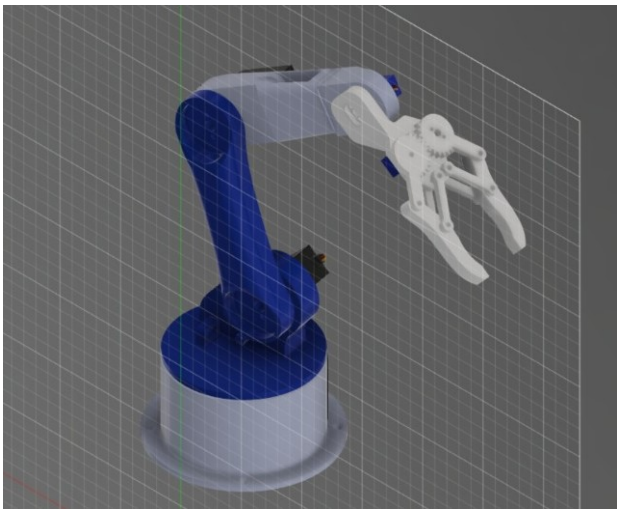


Figure 9: Isometric View

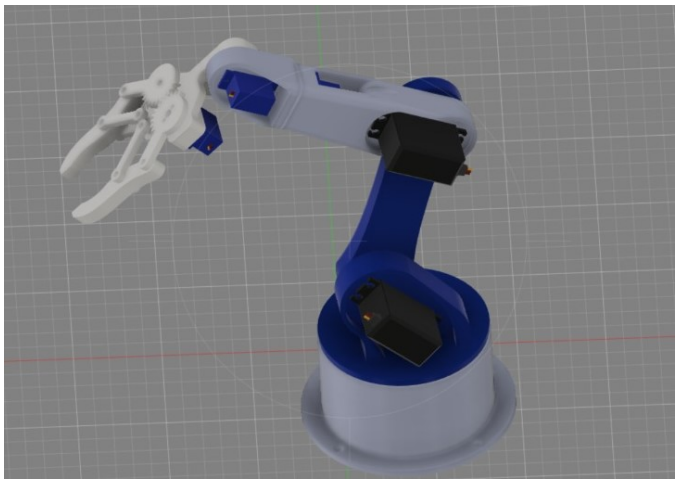


Figure 10: Side View

4.1.2 PARTS

All the parts of the robot that we created are manufactured on 3-D Printer.

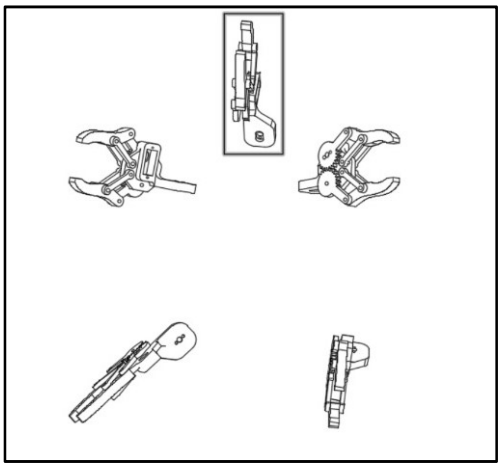


Figure 11: Claw Drawing

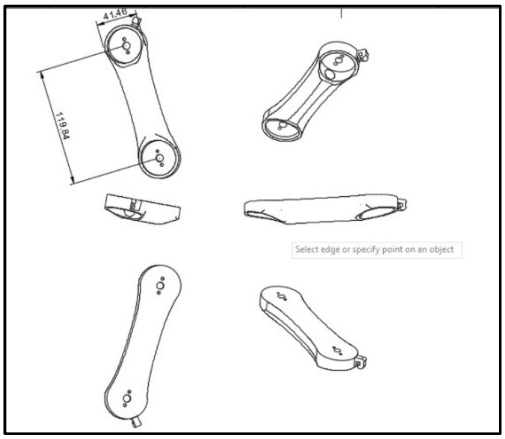


Figure 12: Main Arm Drawing

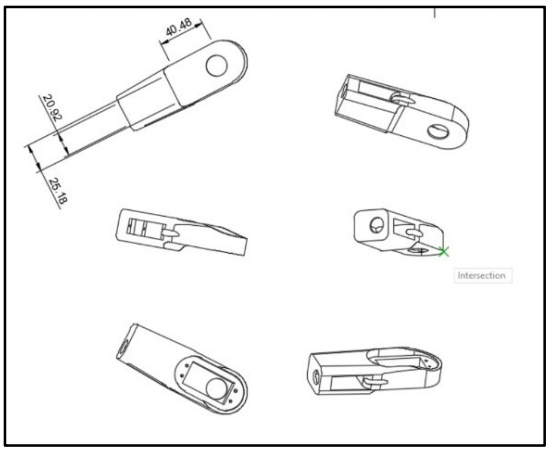


Figure 13: Secondary Arm Drawing

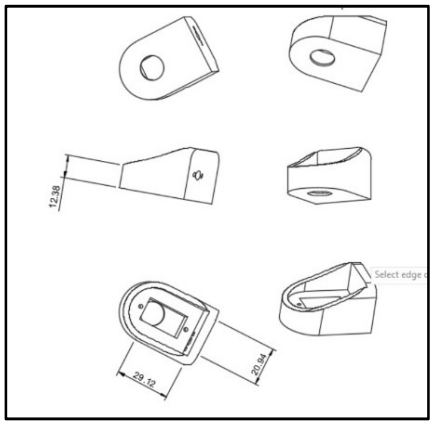


Figure 14: Rot Arm Drawing

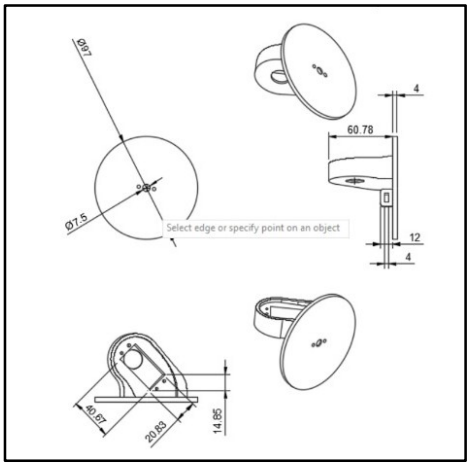


Figure 15: Waist Drawing

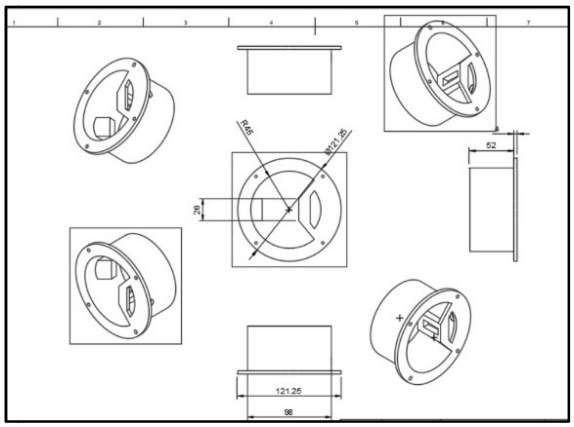


Figure 16: Base Cup Drawing

4.2 Version 2

4.2.1 3d Model

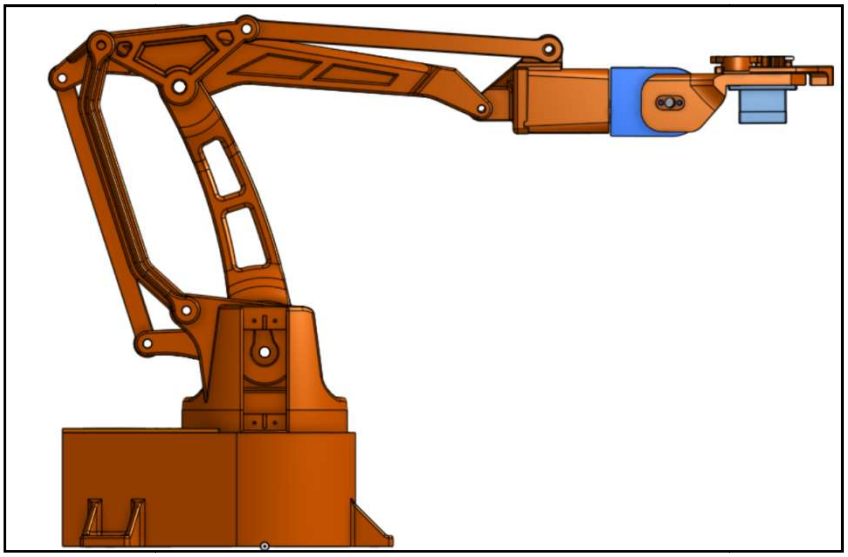


Figure 17 : V2 Side View

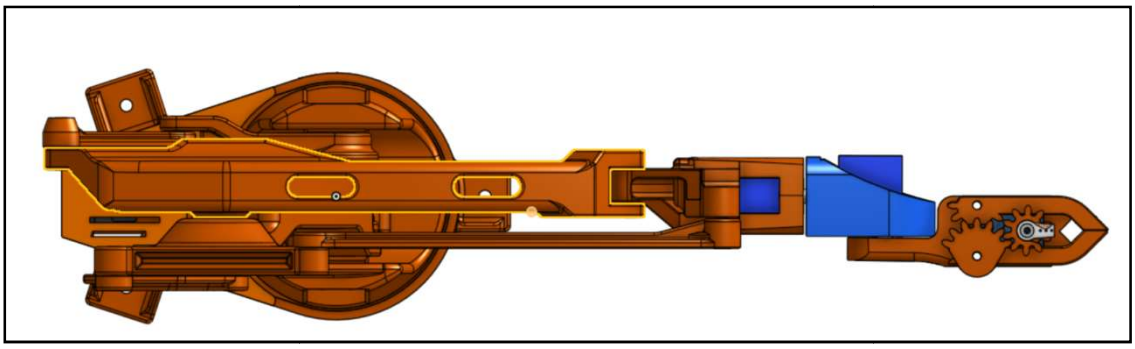


Figure 18 : V2 Top View

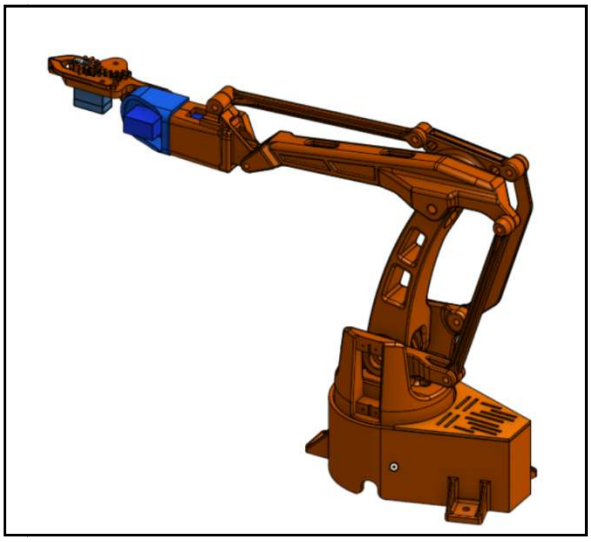


Figure 19 : V2 Isometric View

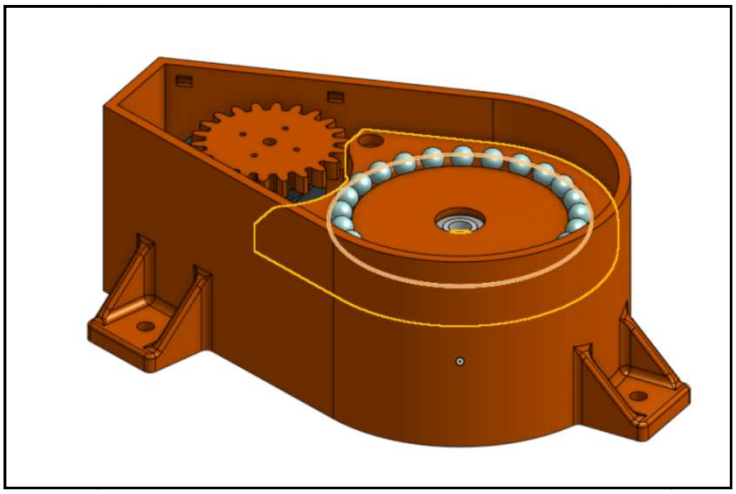


Figure 20 : V2 Base Isometric View

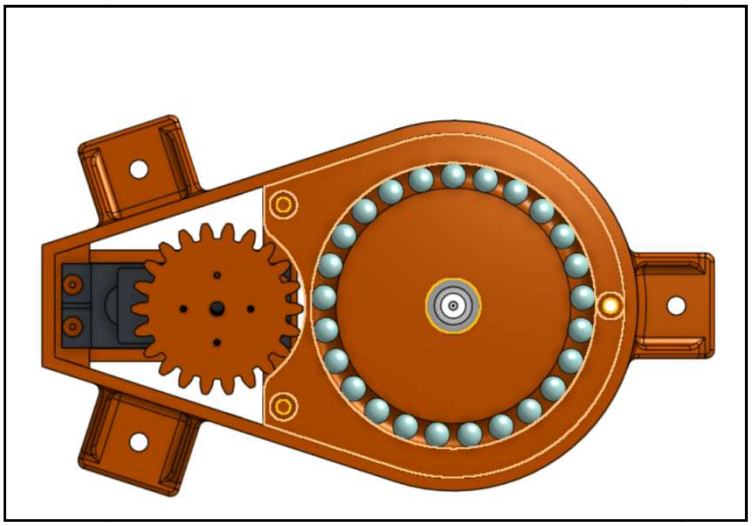


Figure 21 : V2 Base Top View

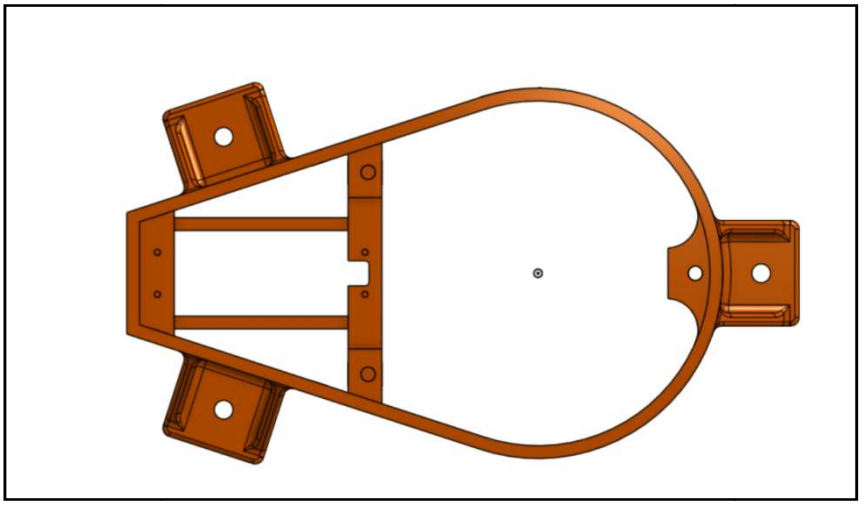


Figure 22 : V2 Base outer Body Top View

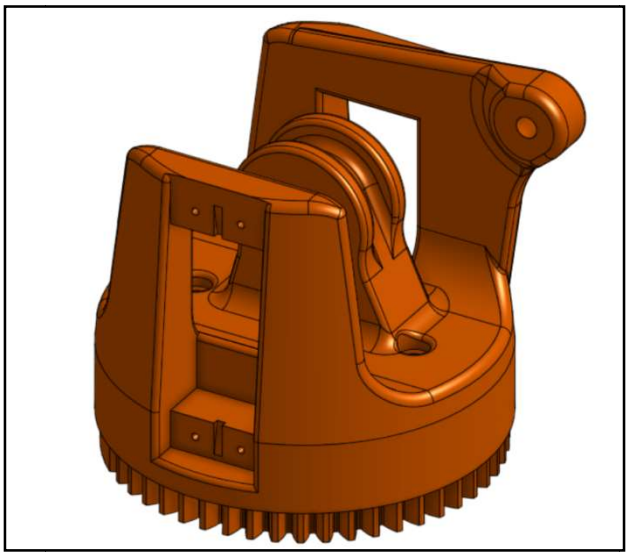


Figure 23 : V2 Waist Isometric View

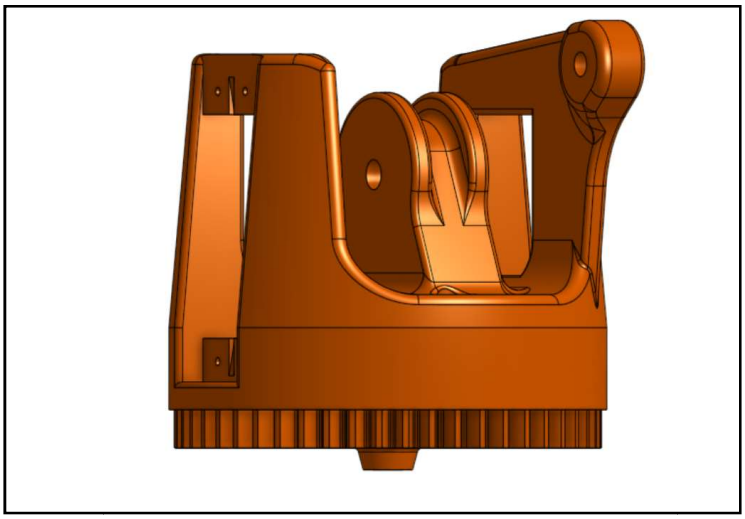


Figure 24 : V2 Waist Side-Isometric View

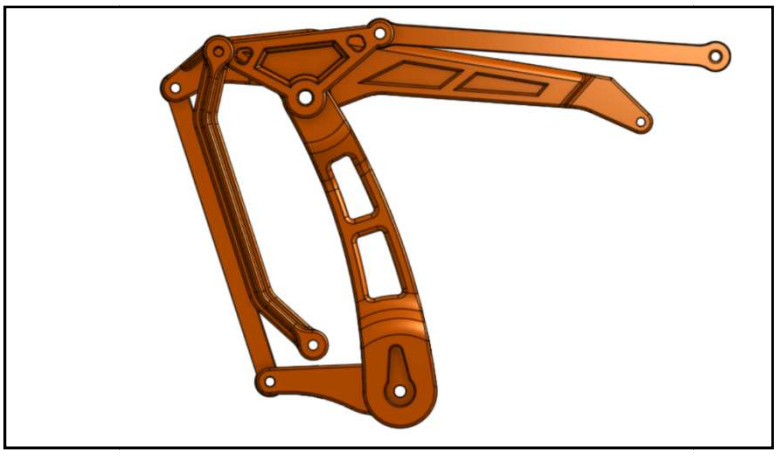


Figure 25 : V2 Arms Side View

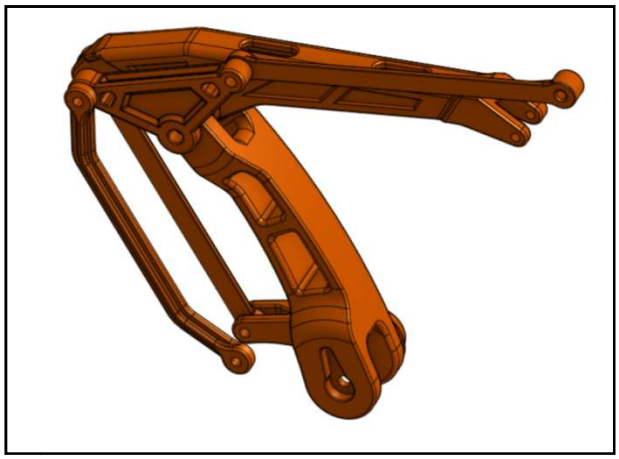


Figure 26 : V2 Arms Isometric View

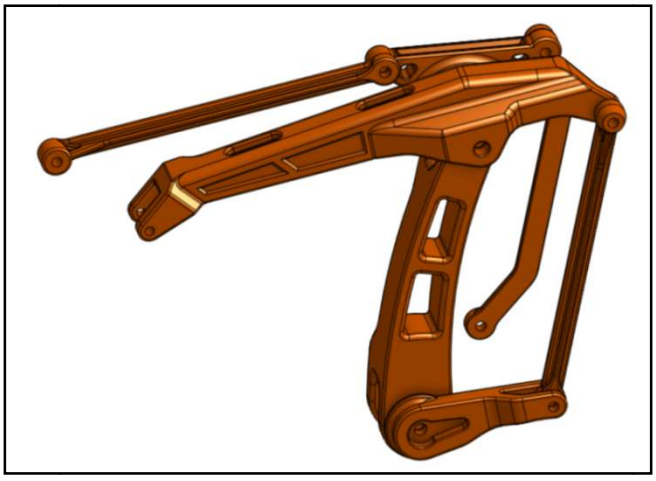


Figure 27 : V2 Arms Isometric View 2

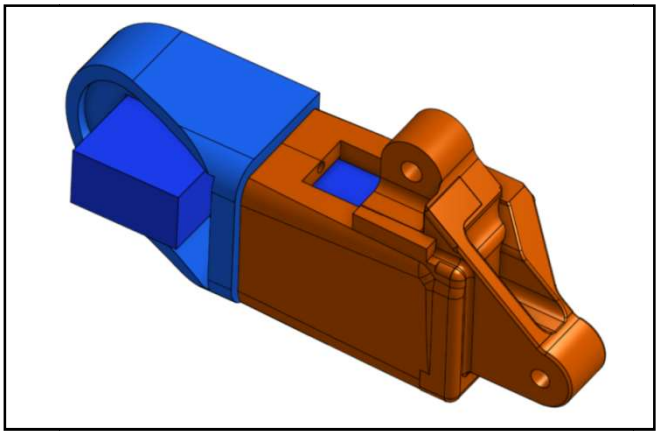


Figure 28 : V2 Front axis Rotation Component Isometric View

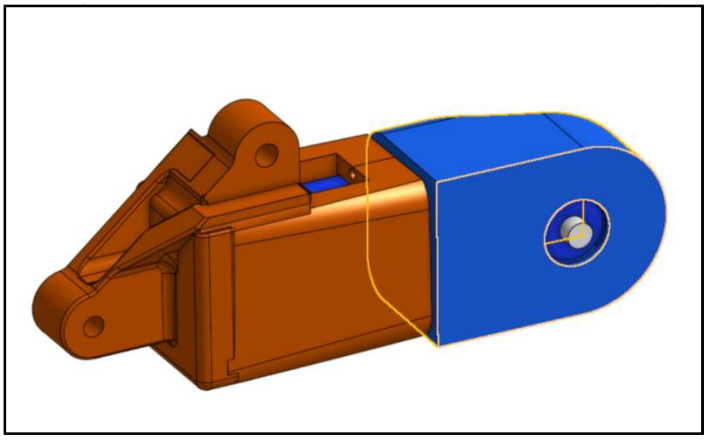


Figure 29 : V2 Front axis Rotation Component Isometric View 2

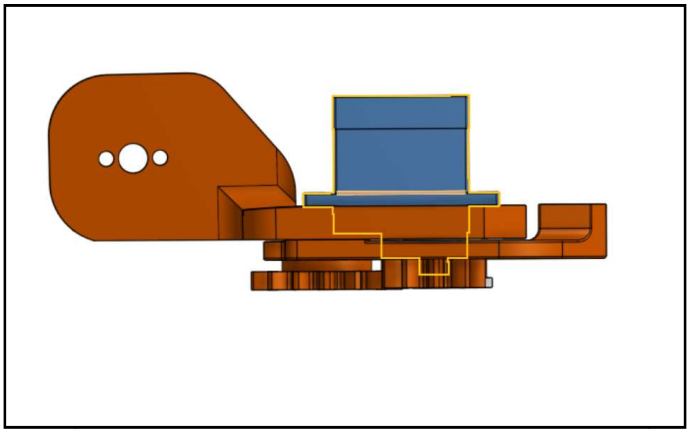


Figure 30 : V2 Claw Component Side View

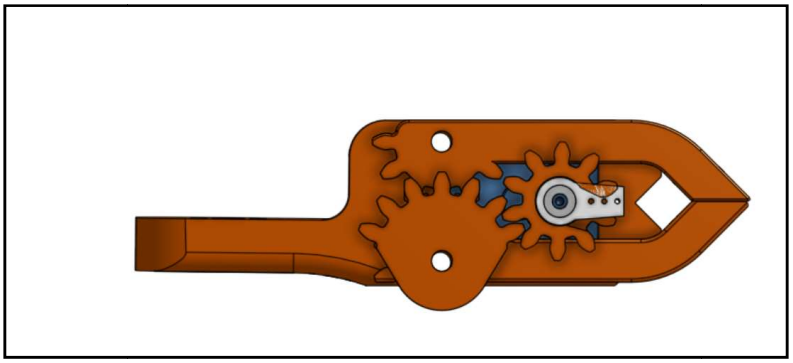


Figure 31 : V2 Claw Component Bottom View

4.2.2 V2 Drawings

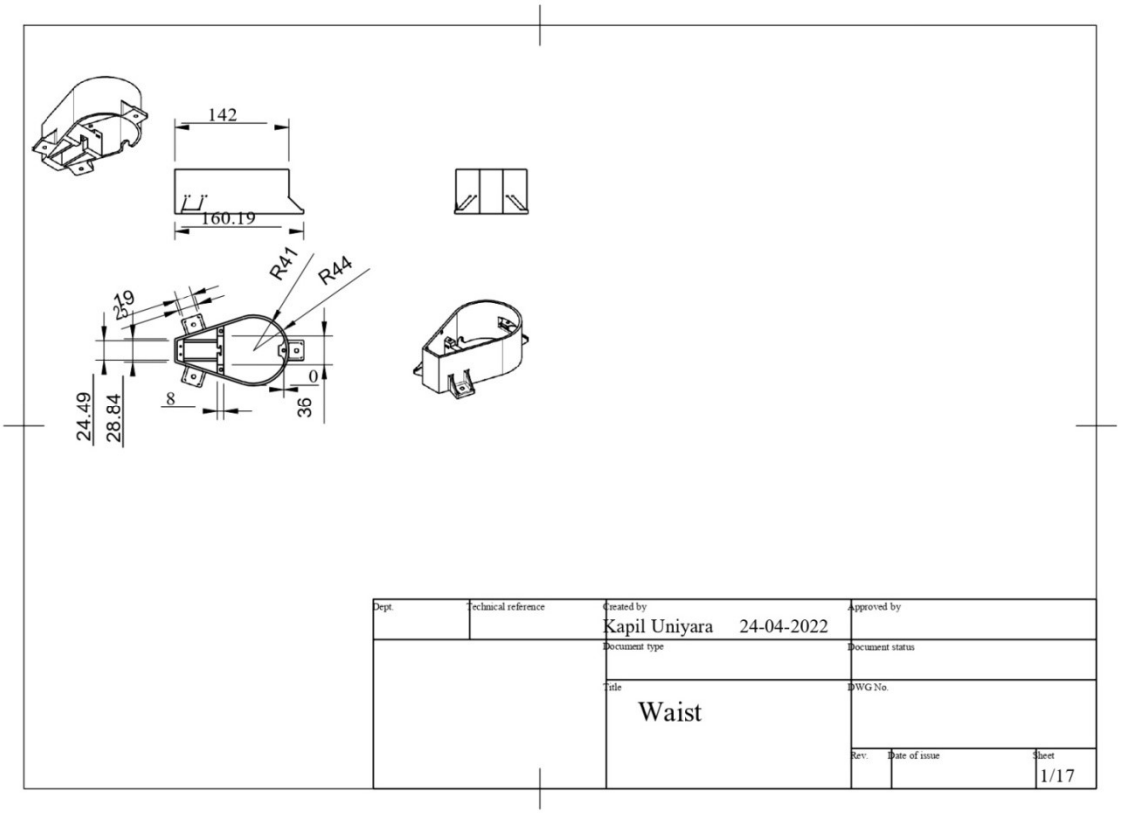


Figure 32 : Waist Drawing

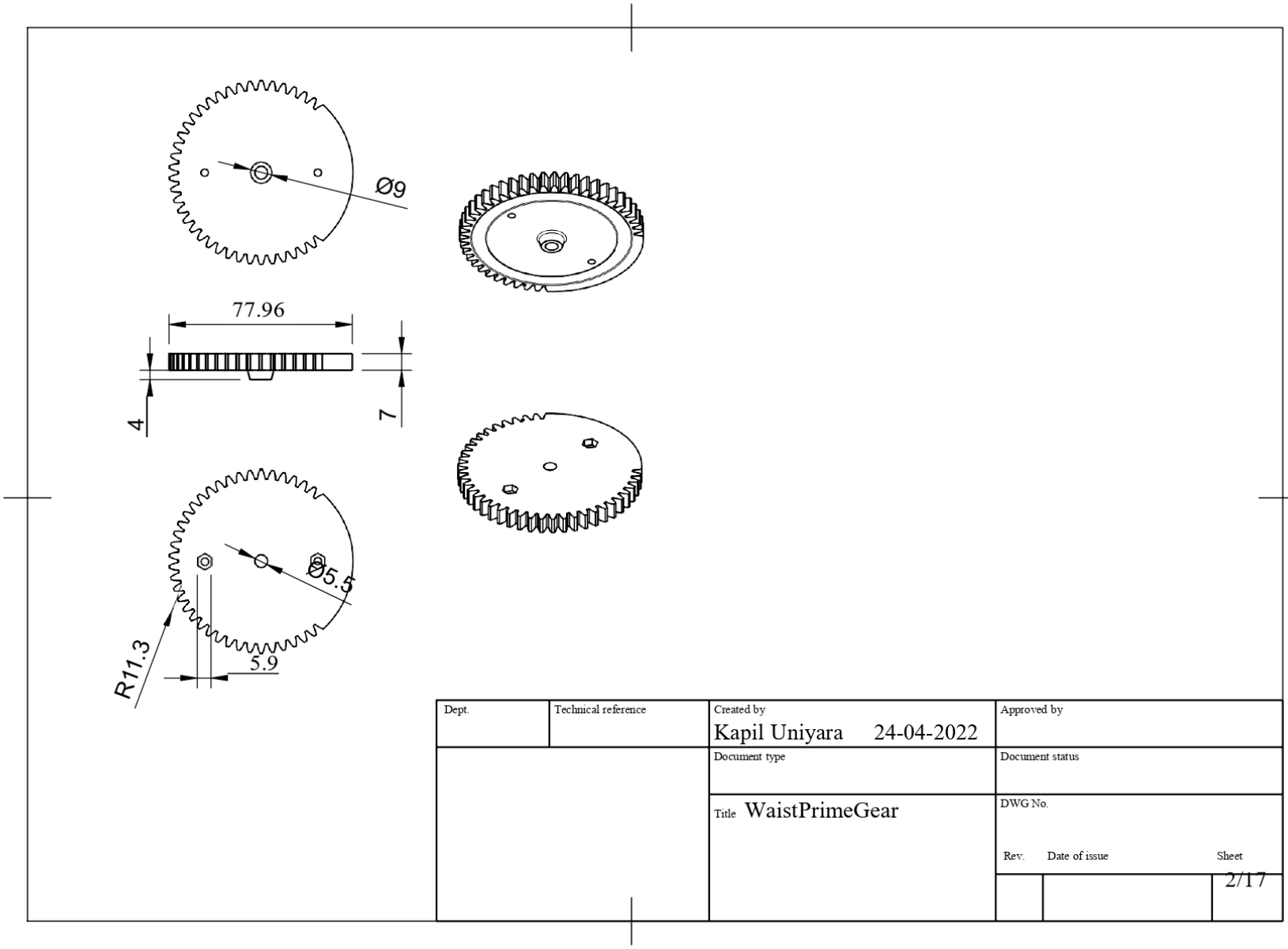


Figure 33 : Waist Prime Gear Drawing

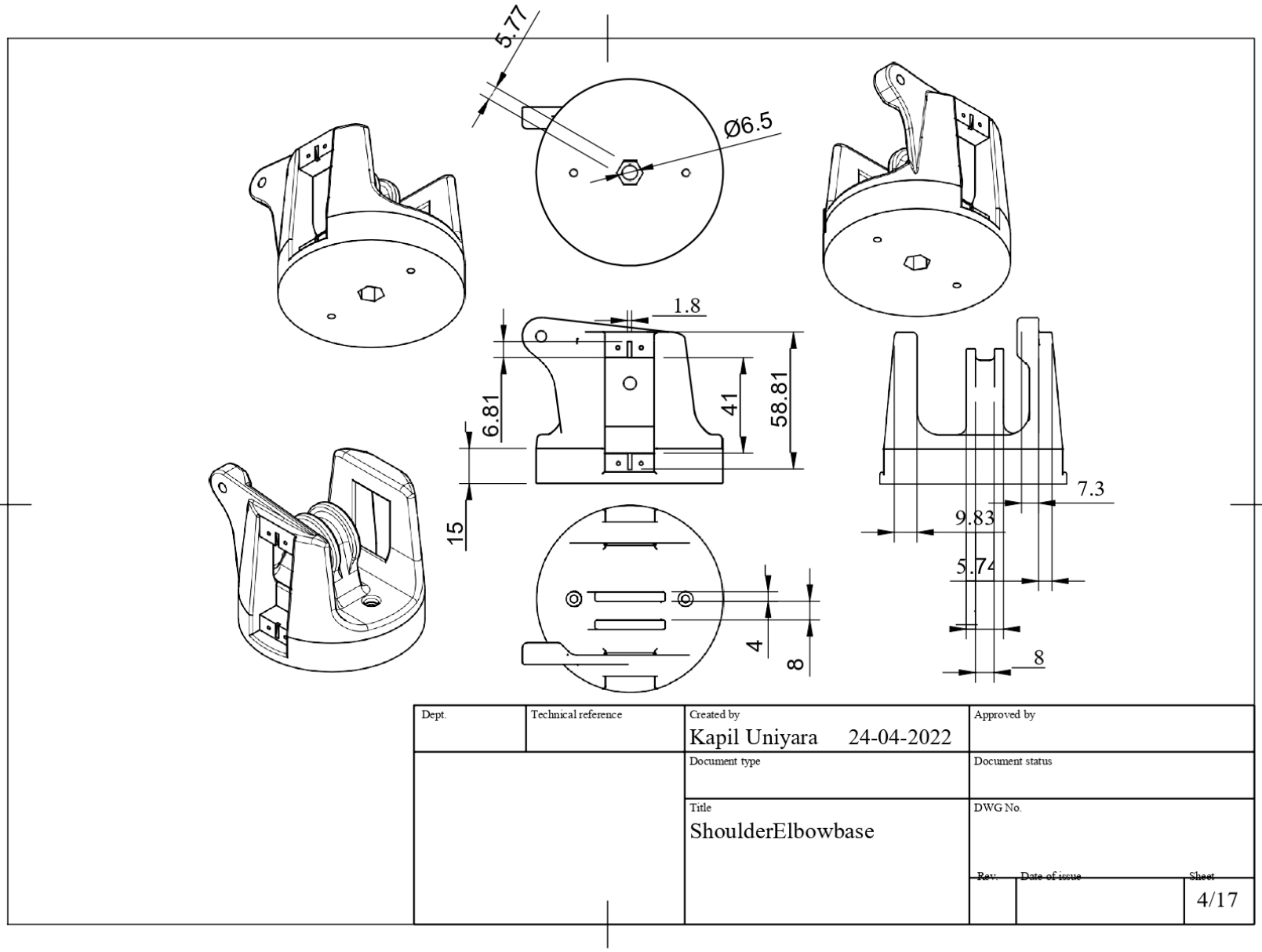
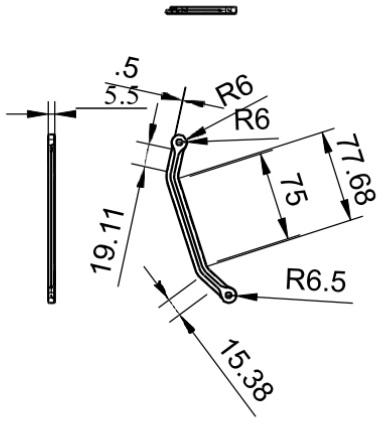


Figure 34 : Shoulder Elbow Base Drawing



Dept.	Technical reference	Created by Kapil Uniyara 24-04-2022	Approved by
		Document type	Document status
		Title ShoulderRod	DWG No.
			Rev. / Date of issue / Sheet 5/17

Figure 35 : Shoulder Rod Drawing

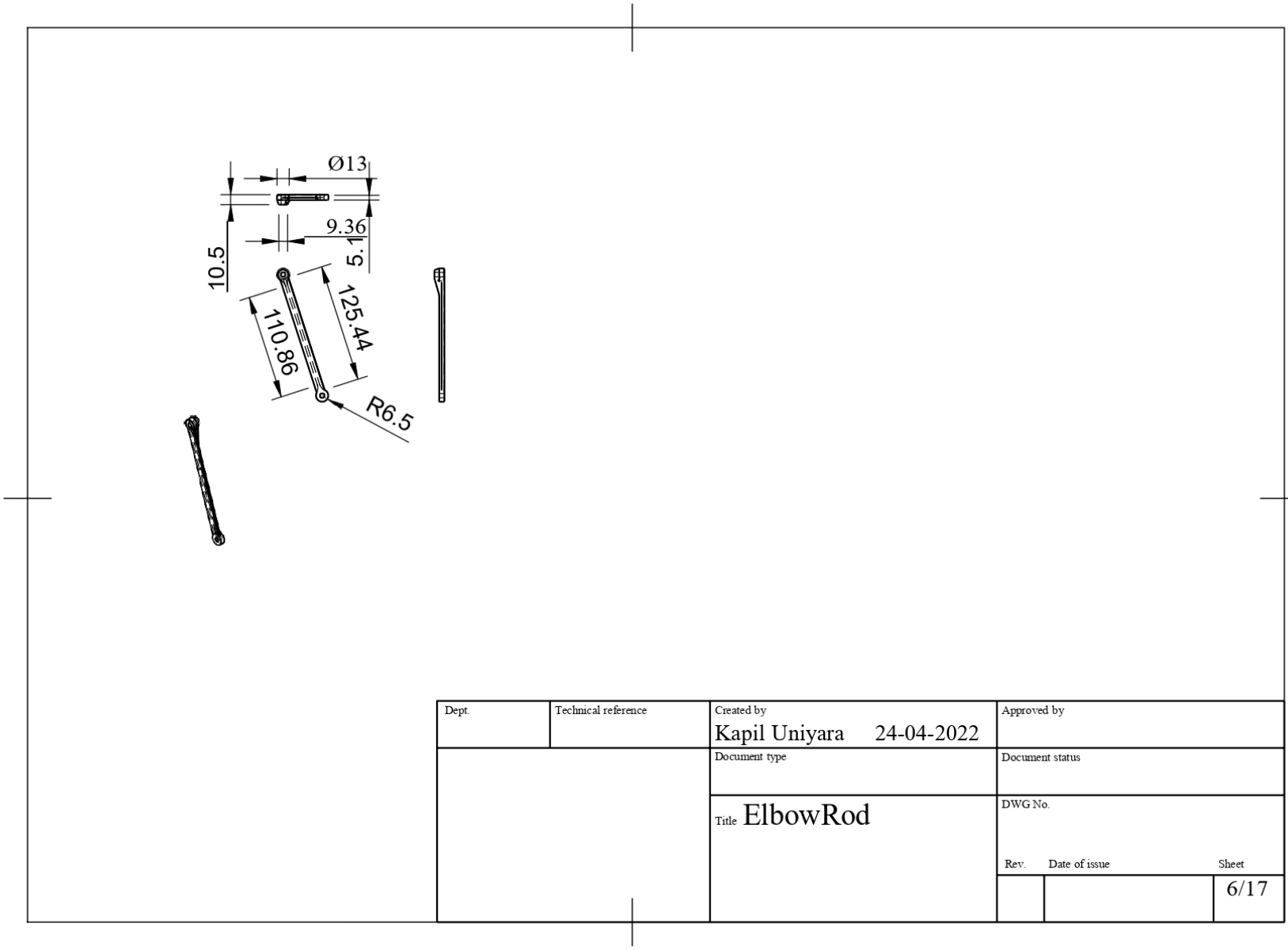


Figure 36 : Elbow Rod Drawing

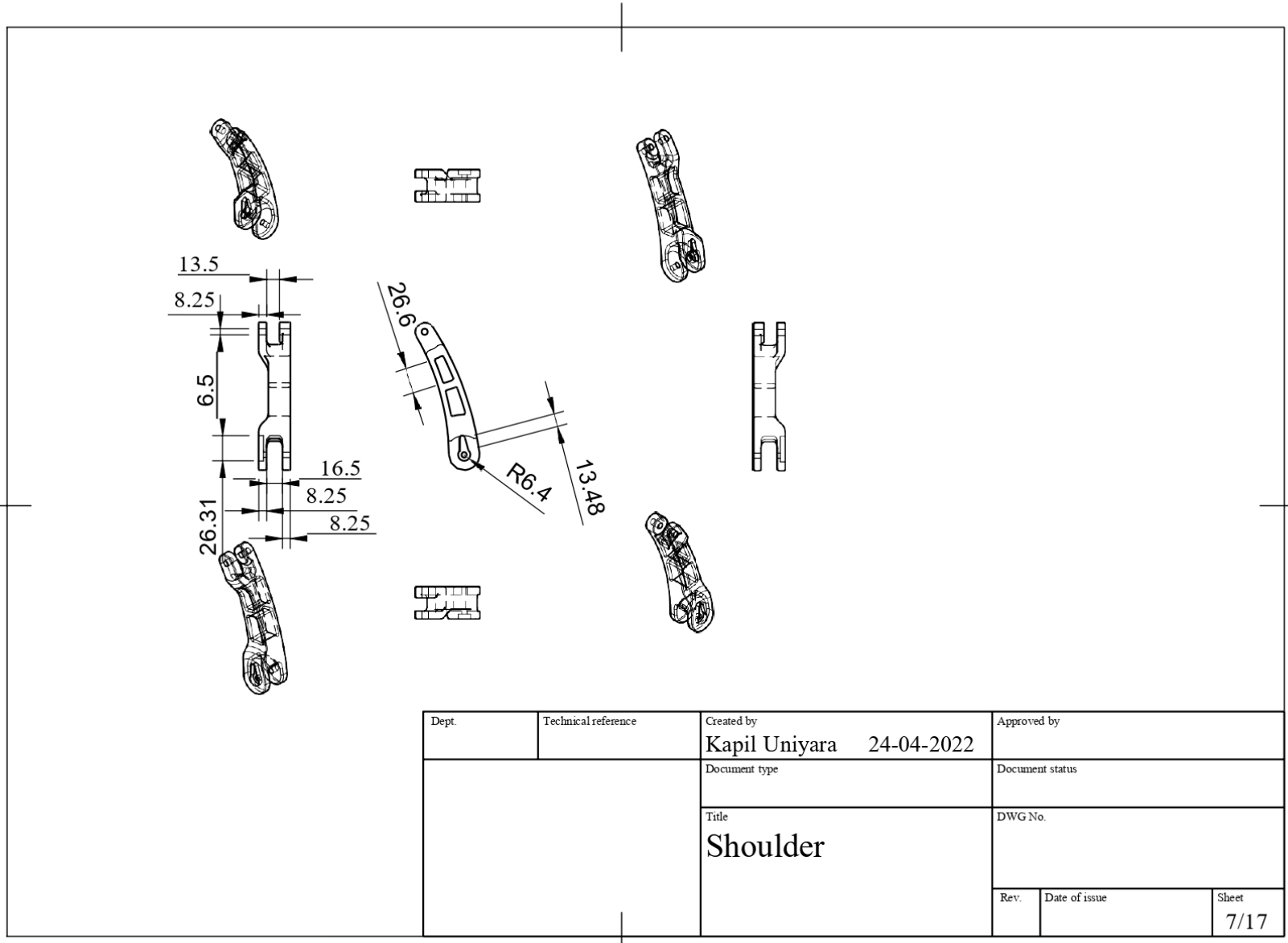
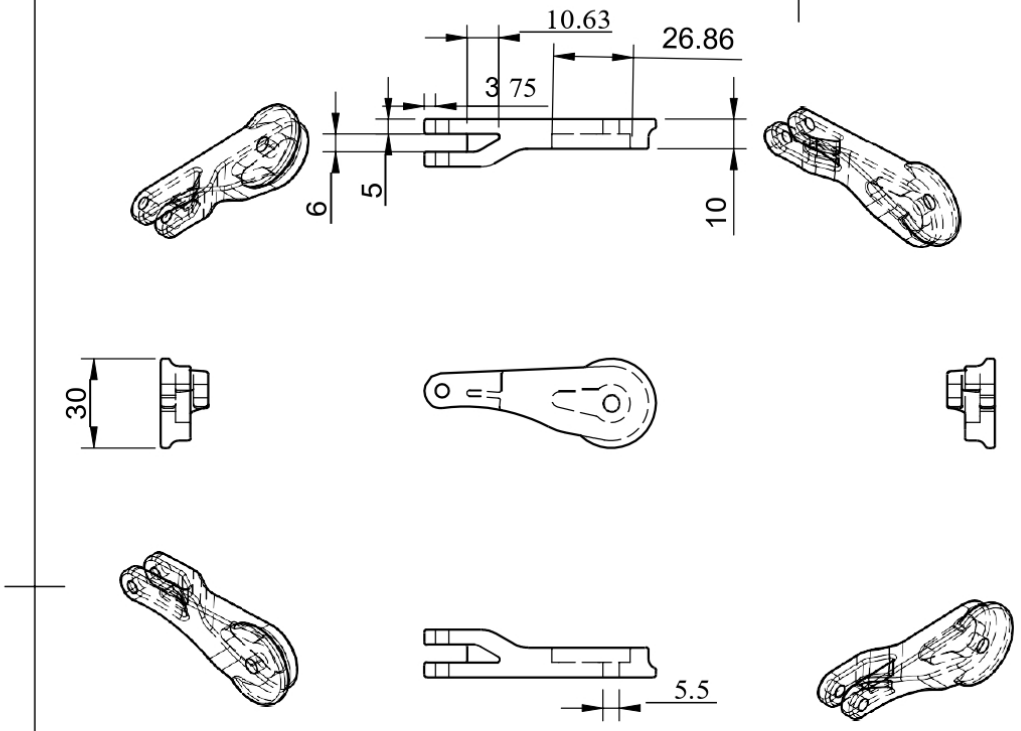
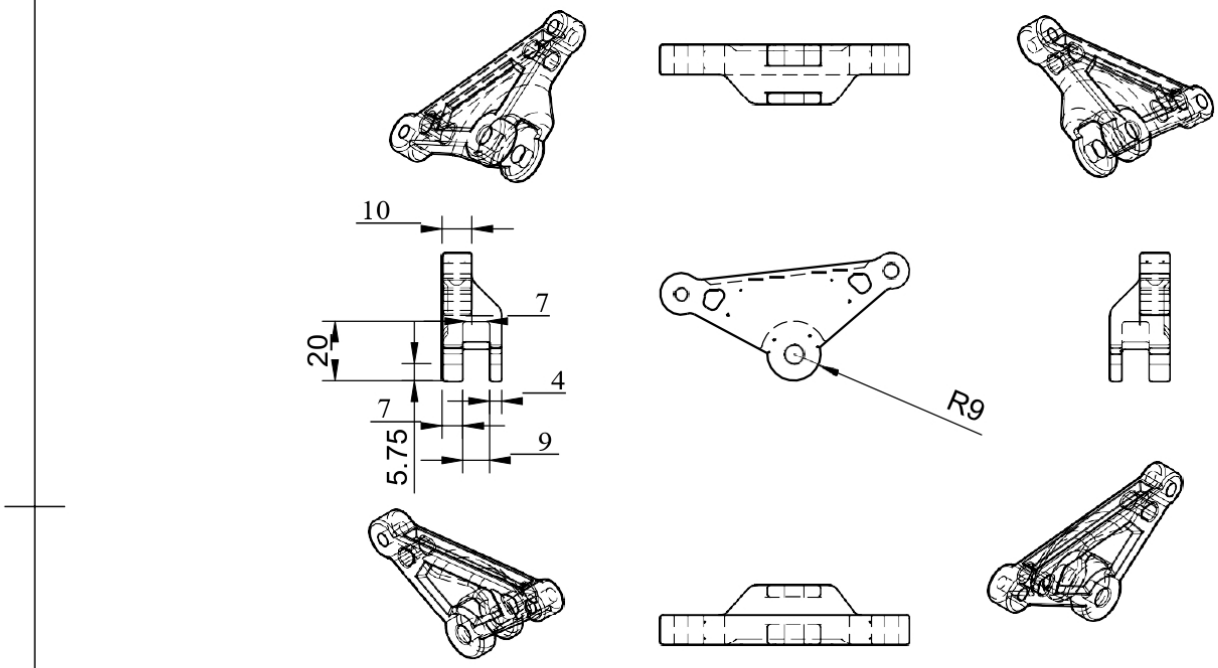


Figure 37 : Shoulder Drawing



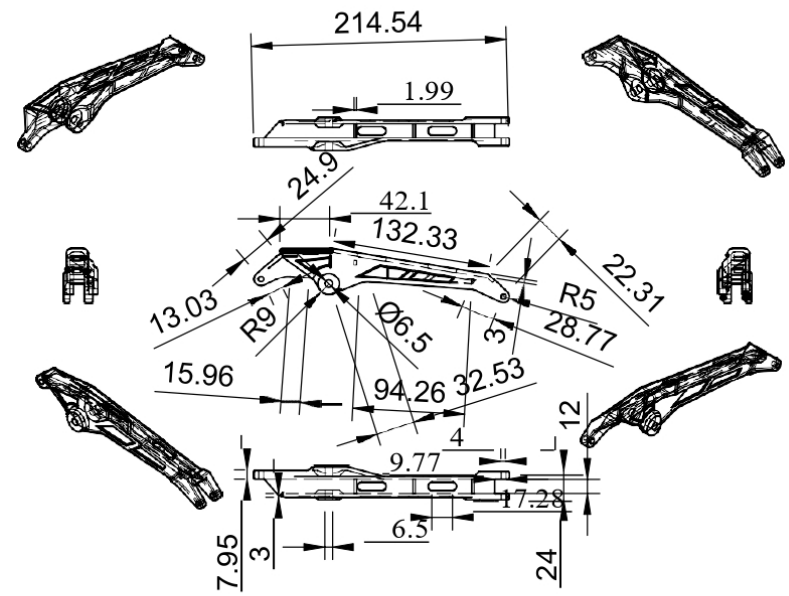
Dept.	Technical reference	Created by Kapil Uniyara 24-04-2022	Approved by
		Document type	Document status
		Title Elbow	DWG No.
			Rev. / Date of issue / Sheet 8/17

Figure 38 : Elbow Drawing



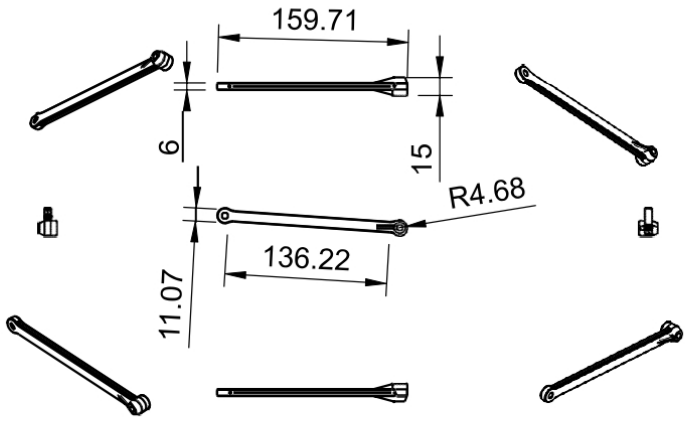
Dept.	Technical reference	Created by Kapil Uniyara 24-04-2022	Approved by
		Document type	Document status
		Title Shoulder Maintainer	DWG No.
			Rev. / Date of issue / Sheet 9/17

Figure 39: Shoulder Maintainer Drawing



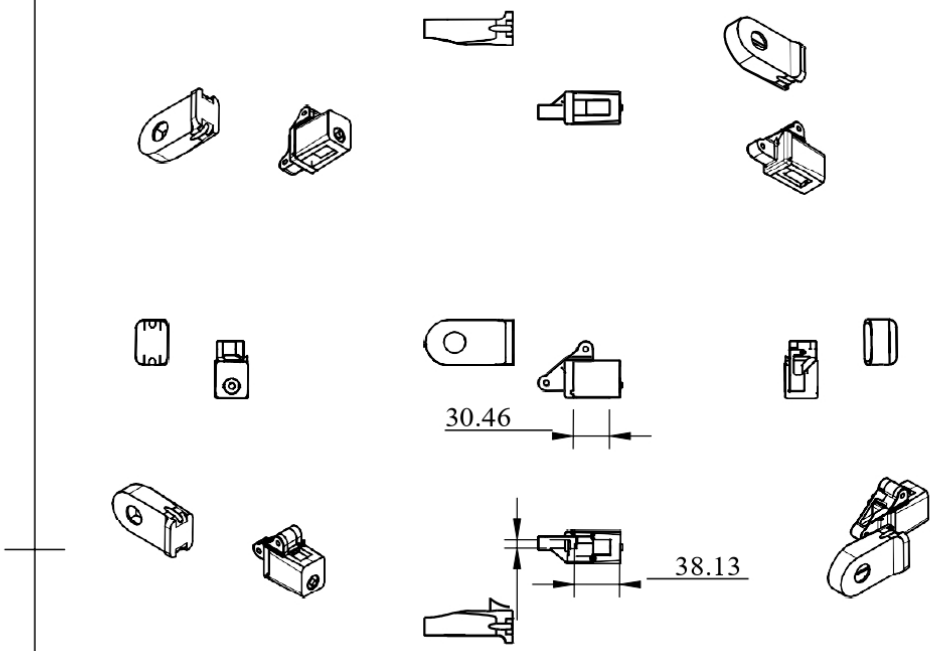
Dept.	Technical reference	Created by Kapil Uniyara 24-04-2022	Approved by
		Document type	Document status
		Title ElbowUpper	DWG No.
			Rev. / Date of issue / Sheet 10/17

Figure 40 : Elbow Upper Drawing



Dept.	Technical reference	Created by Kapil Uniyara 24-04-2022	Approved by	
		Document type	Document status	
		Title ElbowUpperRod	DWG No.	
			Rev. / Date of issue	Sheet 11/17

Figure 41 : Elbow Upper Rod Drawing



Dept.	Technical reference	Created by Kapil Uniyara 24-04-2022	Approved by	
		Document type	Document status	
		Title Arm	DWG No.	
			Rev. / Date of issue	Sheet 12/17

Figure 42 : Arm Drawing

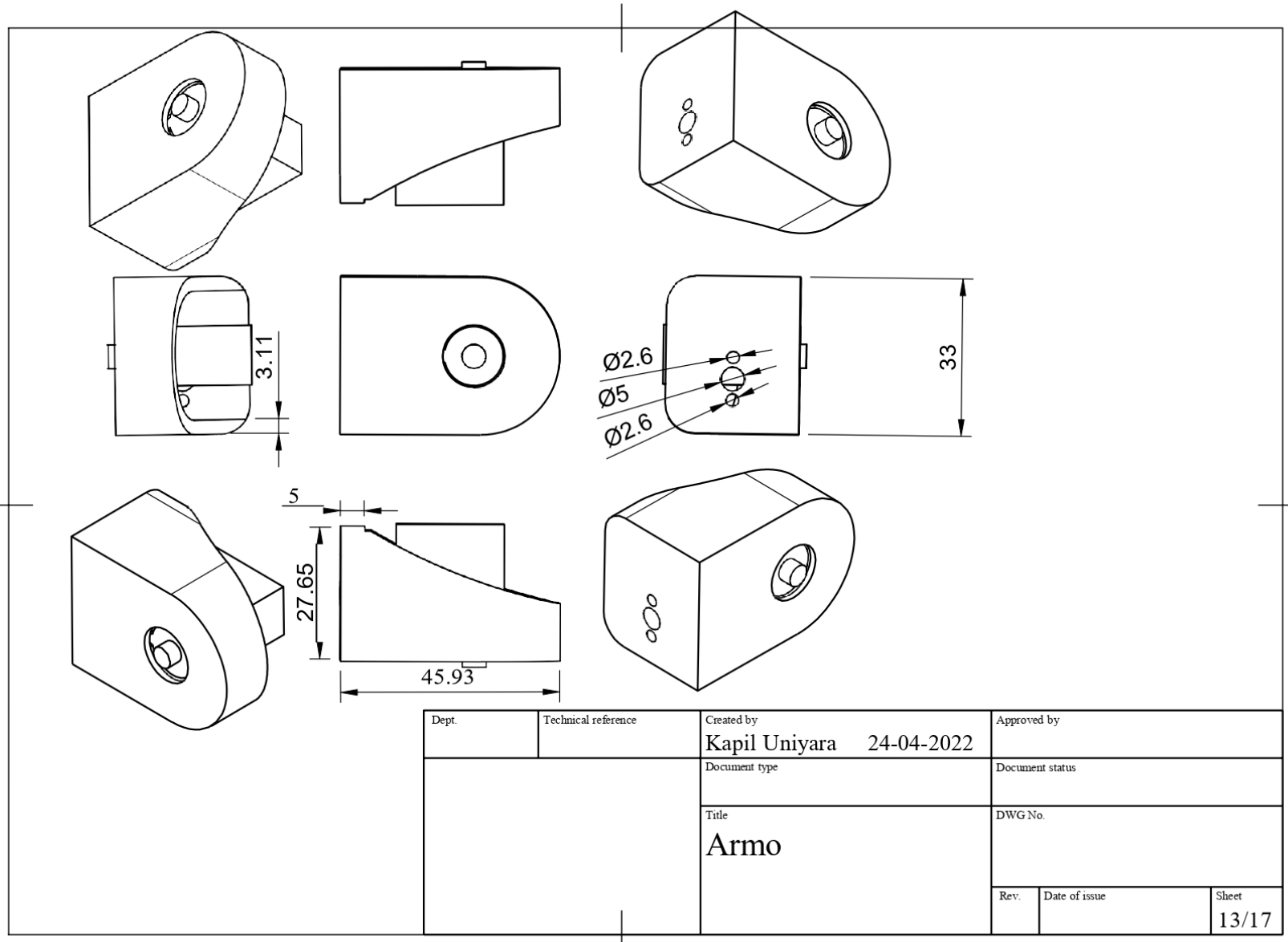


Figure 43 : Armo Drawing

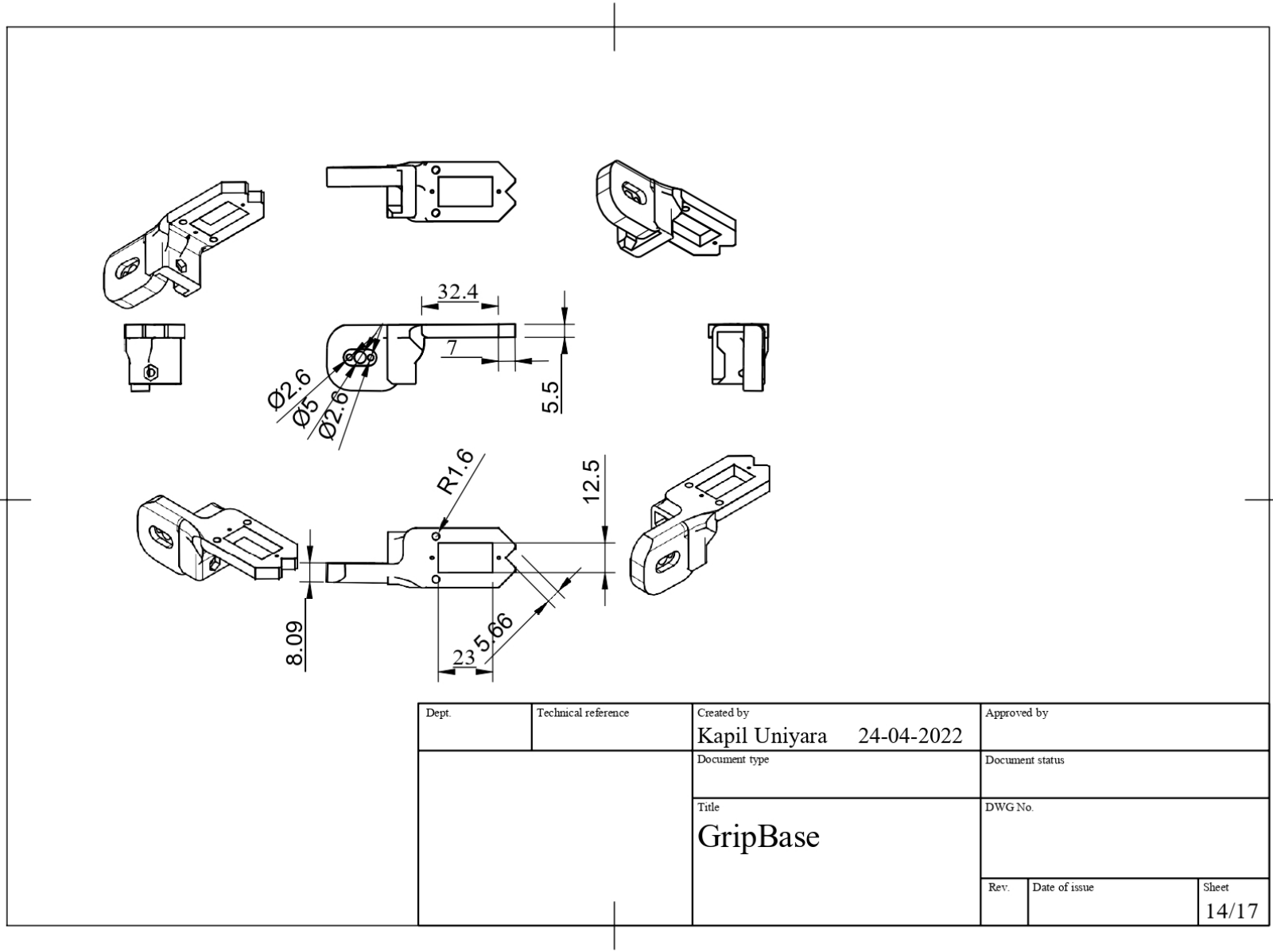


Figure 44 : Grip Base Drawing

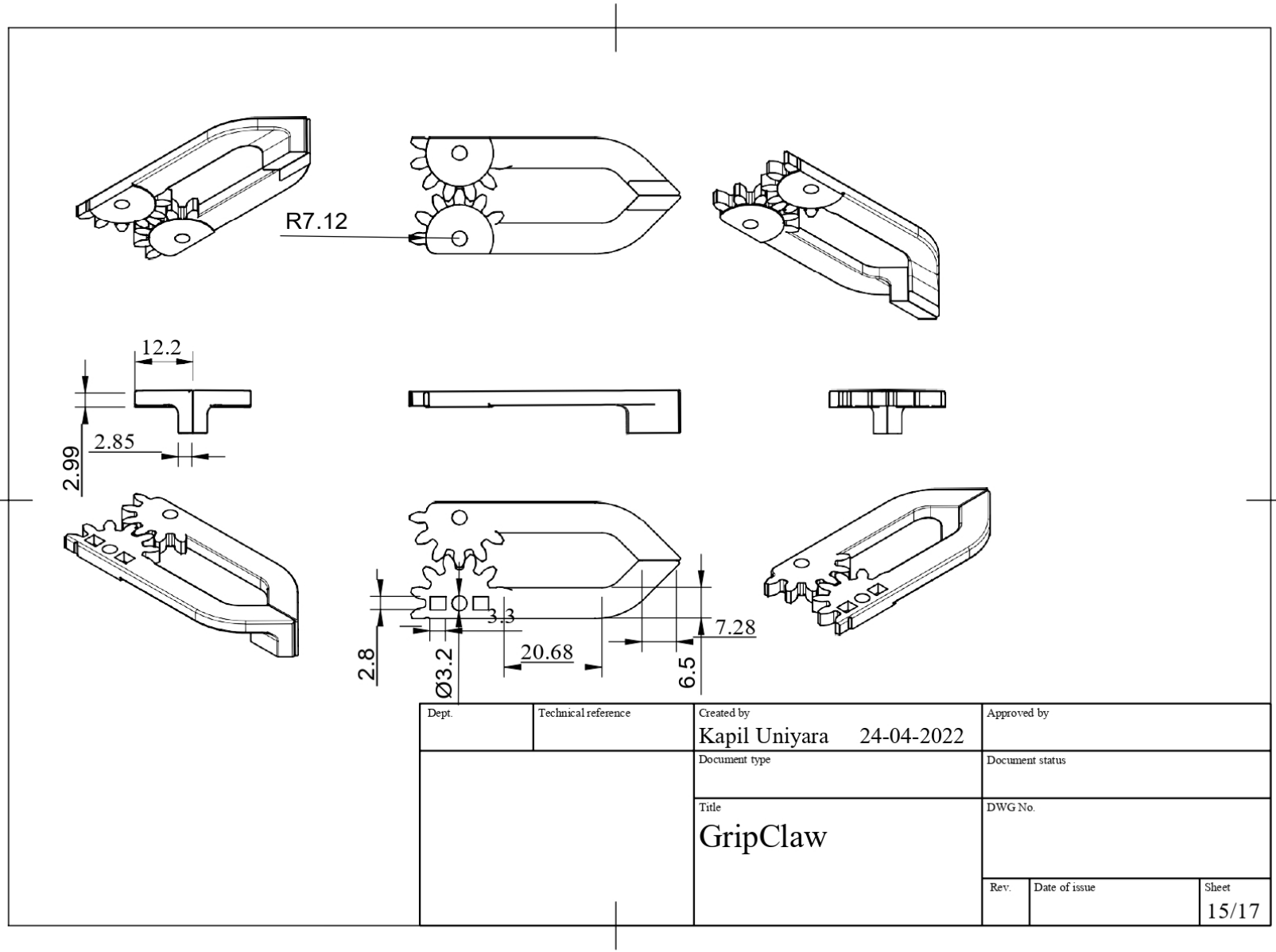


Figure 45 : Grip Claw Drawing

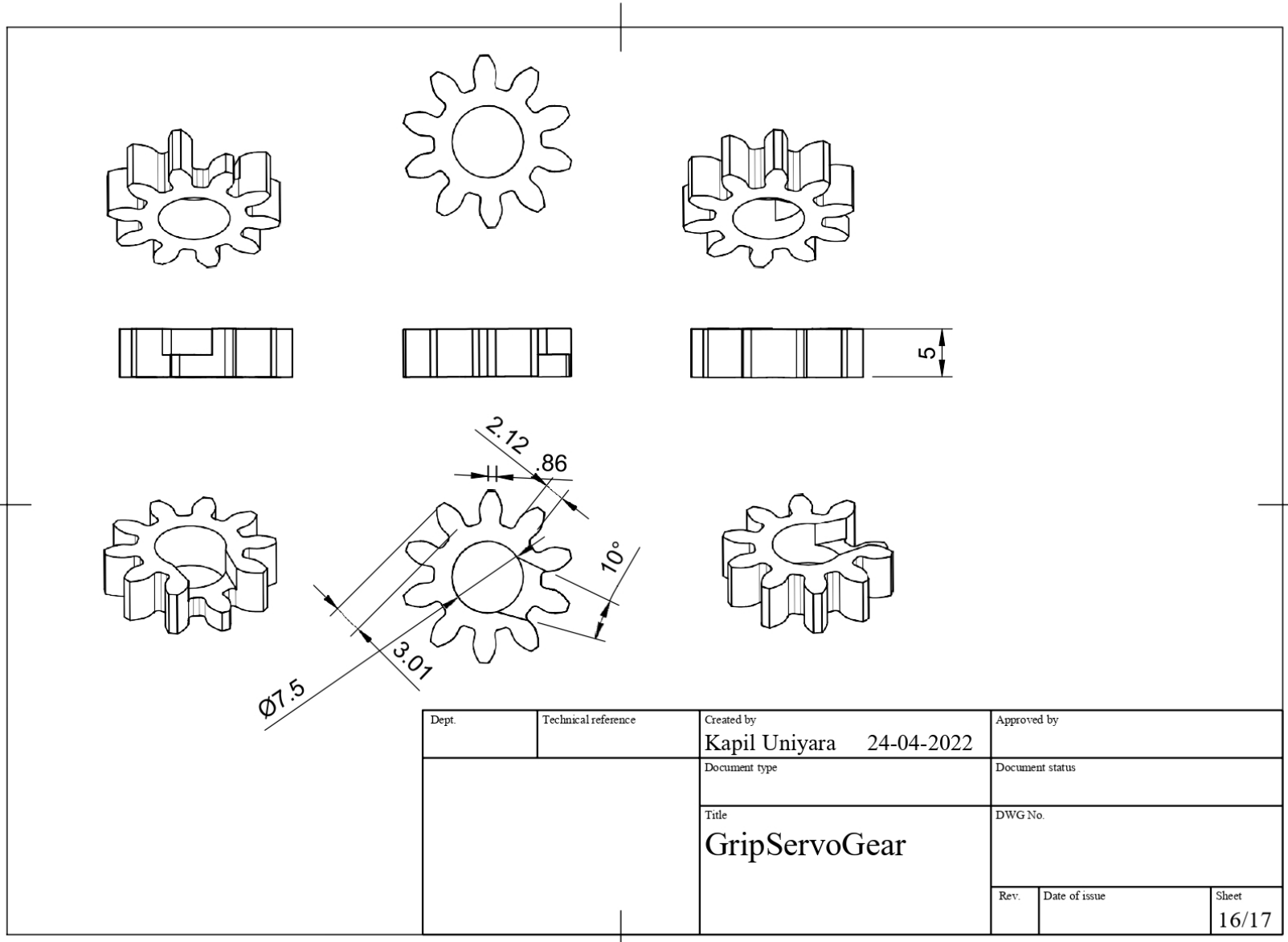


Figure 46 : Grip Servo Gear Drawing

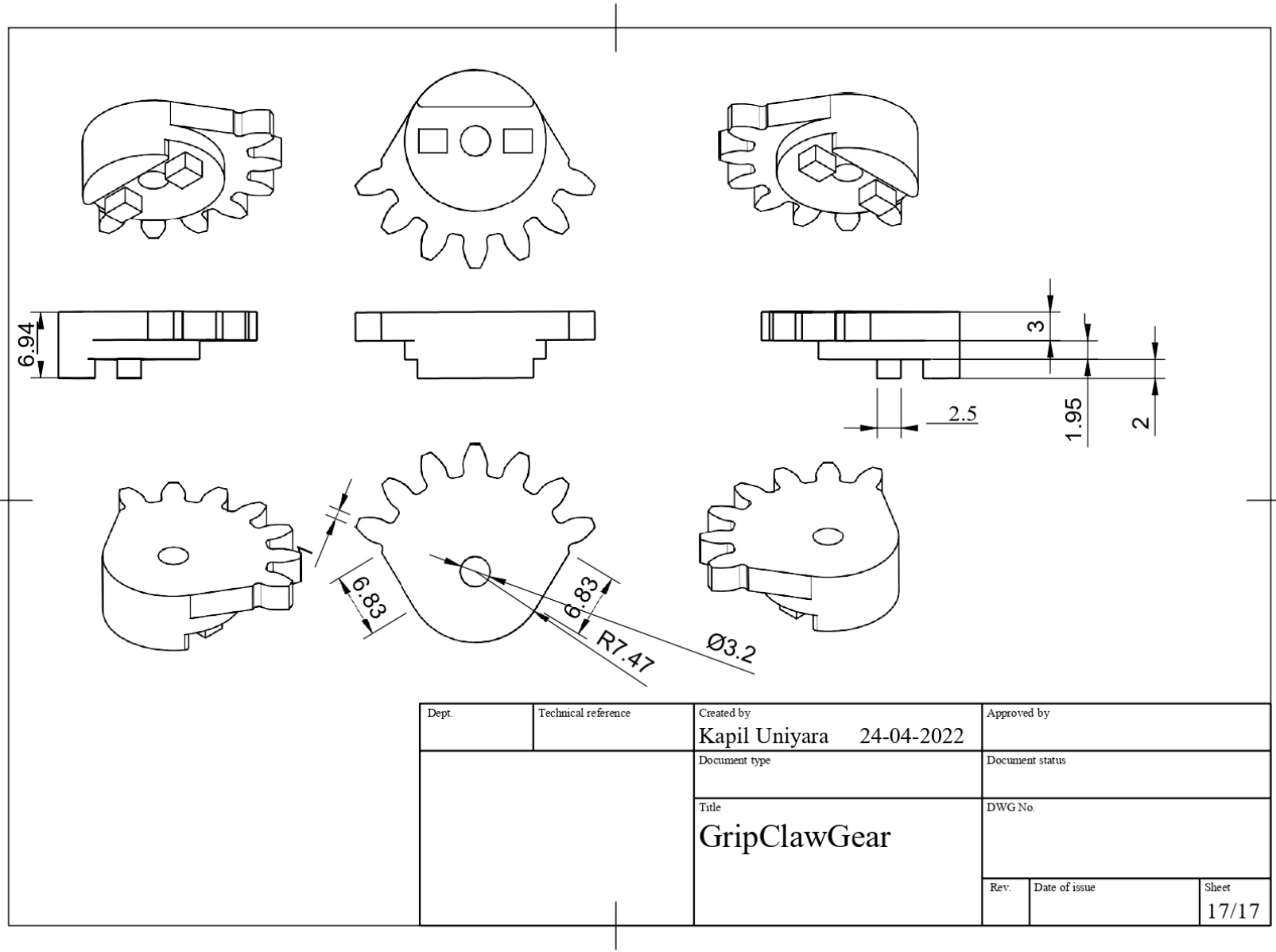


Figure 47 : Grip Claw Gear Drawing

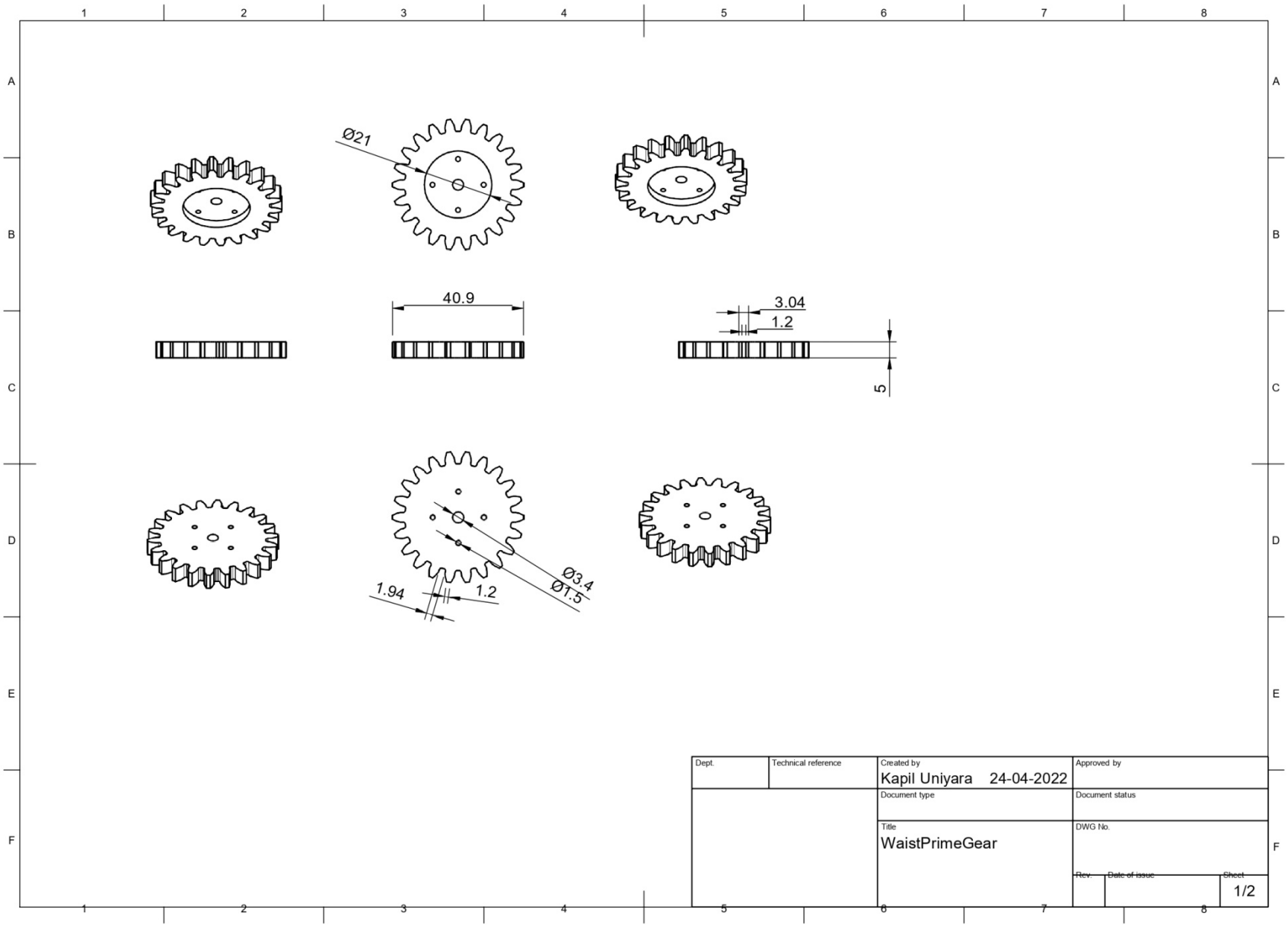


Figure 48 : Waist Prime Gear Drawing

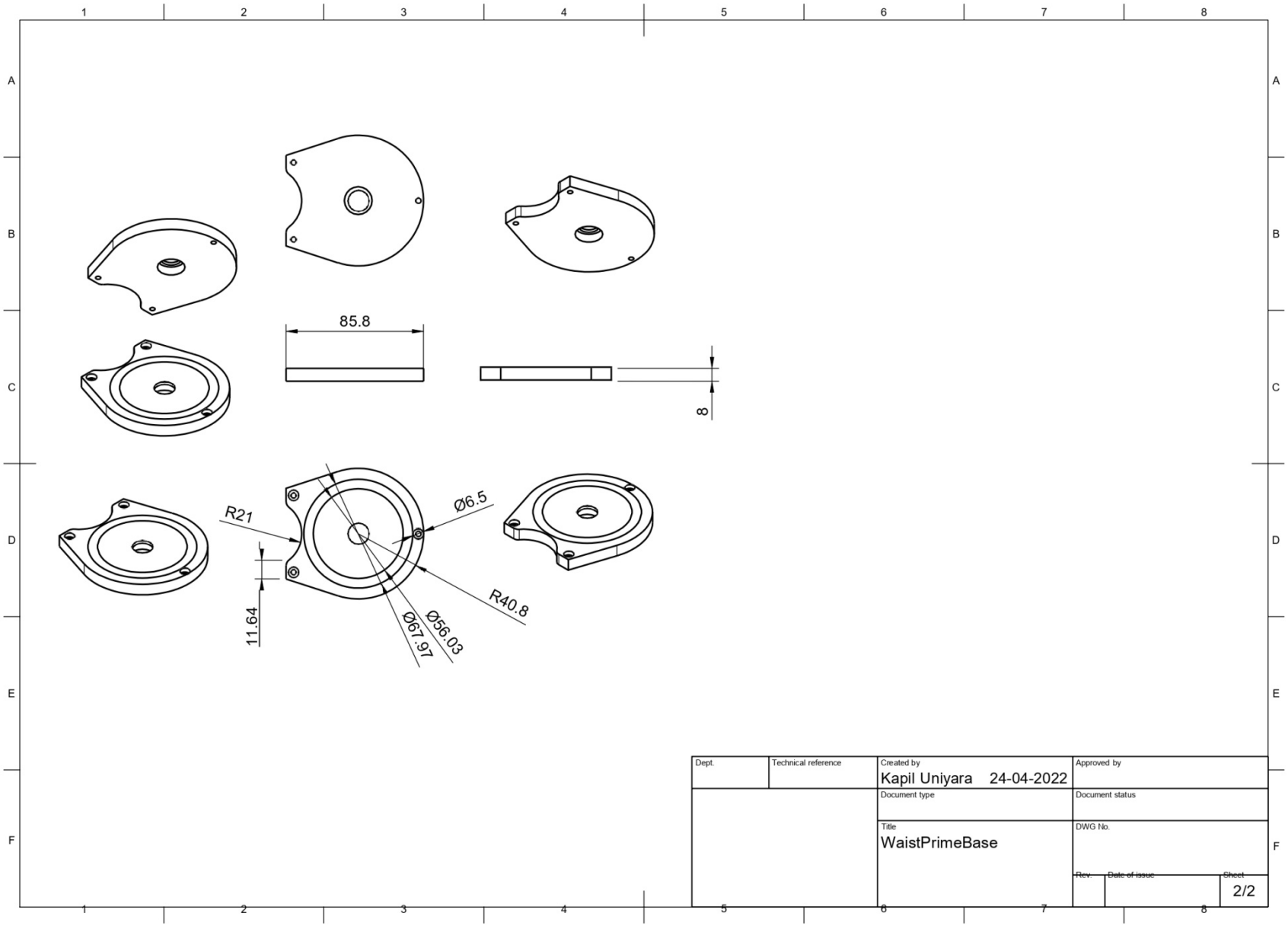


Figure 49 : Waist Prime Base Drawing

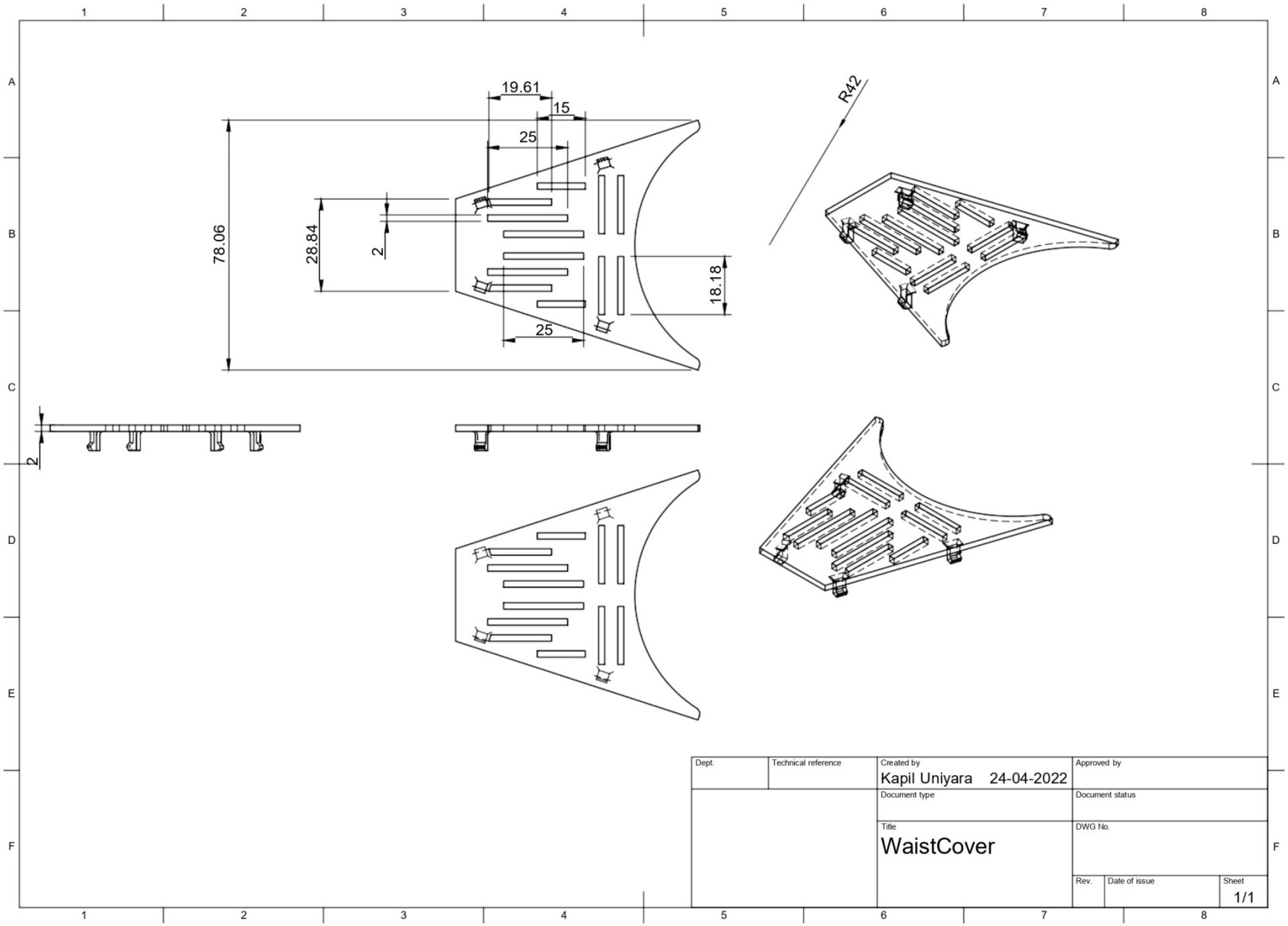


Figure 50 : Waist Cover Drawing

Chapter 5

ELECTRICAL DESIGN

5.1 ELECTRICAL CIRCUITS

A very important part of this project, is the electrical circuits, they are used to send and receive the signals from motors drivers, limit switches and encoder.

The interfacing circuit of the Arduino UNO controller and the servo motors, and the Ultrasonic Sensor is given in Figure 14

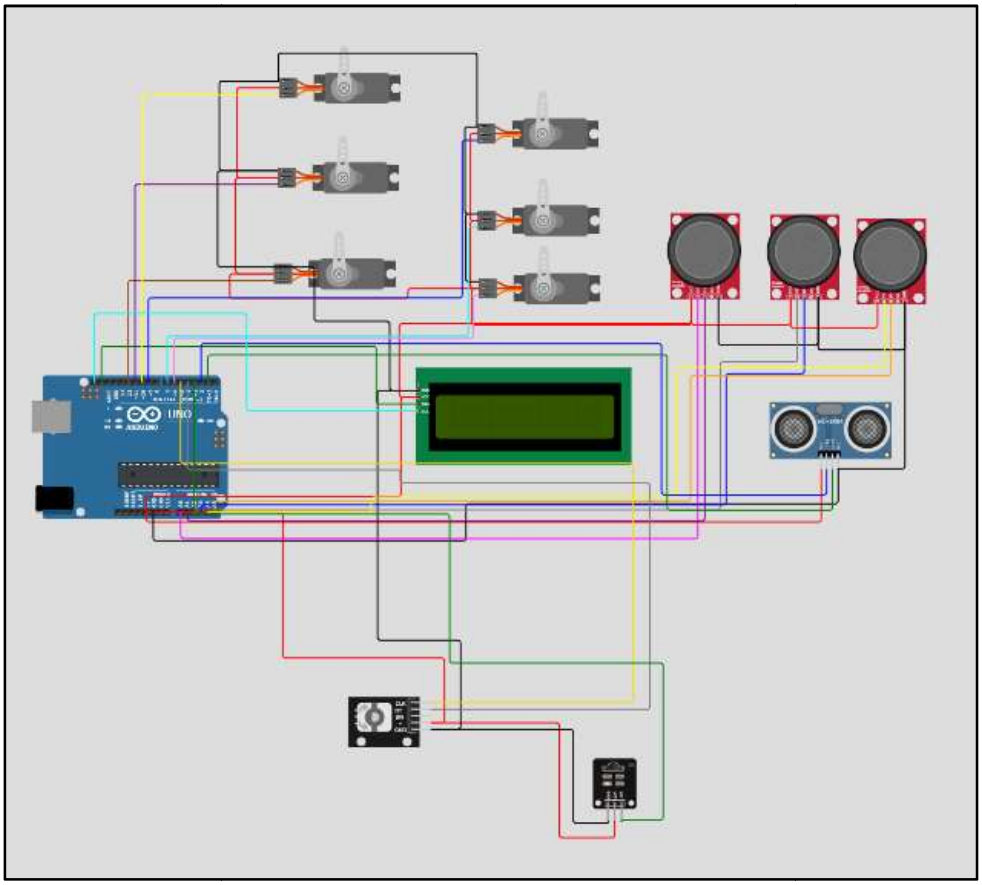


Figure 51 : Arduino circuit diagram

5.2 ELCTRICAL COMPONENTS

In this section a detailed description is presented for each component mentioned in the electrical design.

5.2.1 Motors

The main factors for choosing the robot actuators are the availability of them in the market and the motor rated torque and speed to ensure that they are able to actuate the robot mechanisms.

5.2.1.1 SG90 Servo Motor

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kg*cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 µs
- Temperature range: 0 ºC – 55 ºC

Micro Servo Motor SG90 is a tiny and lightweight server motor with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos.

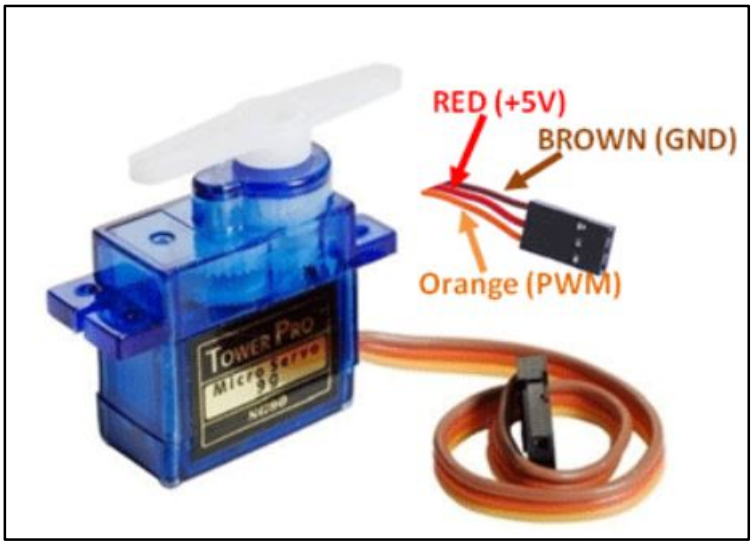


Figure 52: SG90 Servo Motor

5.2.1.2 MG996 Servo Motor

- Weight: 55 g
- Operating voltage: 4.8 V to 7.2 V
- Stall torque: 9.4 kg*cm (4.8 V), 11 kg*cm (6 V)
- Stall Current: 2.5 A
- Rotate angle: 180 degrees.

This motor used in End effectors mechanisms (for Rack and pinion, wrist joint, and gripper).

Figure 53 : MG996 Servo Motor

5.2.2 Arduino UNO Controller

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on

the Arduino website. Layout and production files for some versions of the hardware are also available.

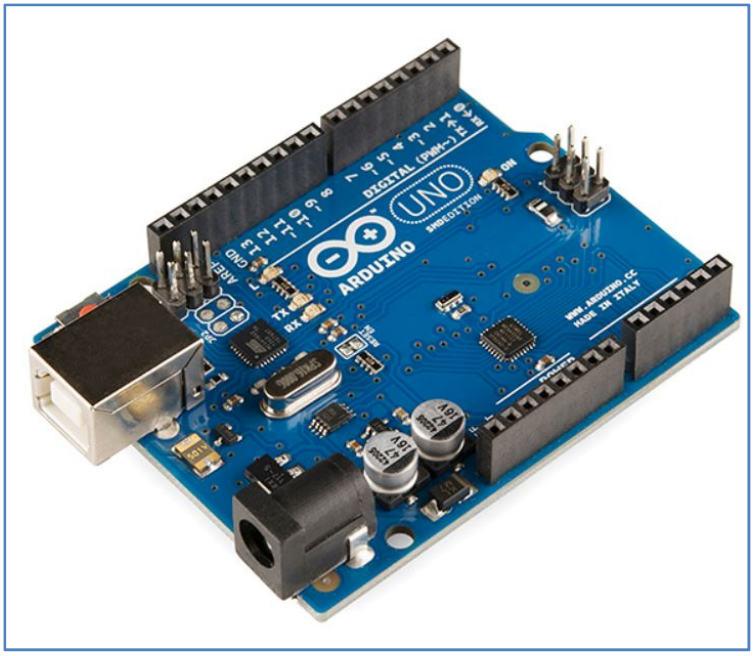


Figure 54 : Arduino UNO

The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes preprogrammed with a bootloader that allows uploading new code to it without the use of an external hardware programmer.

While the Uno communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

5.2.3 Bluetooth Transceiver Module

The Bluetooth Transceiver HC-05 TTL Module with enable/disable button Breakout is the latest Bluetooth wireless serial cable! This version of the popular Bluetooth uses the HC-05/HC-06 module. These modems work as a serial (RX/TX) pipe. Any serial stream from 9600 to 115200bps can be passed seamlessly from your computer to your target.

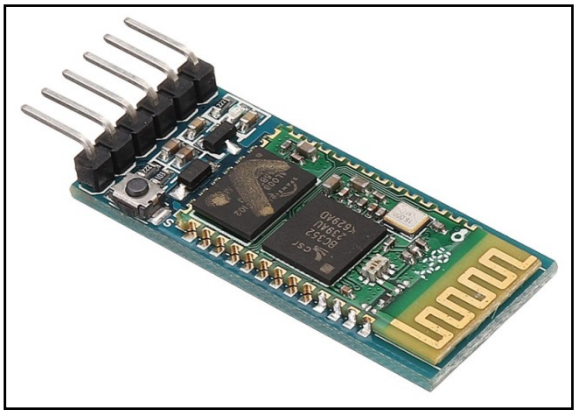


Figure 55 : HC-05 Bluetooth Transceiver Module

Input Supply voltage (V): 3.3 ~ 6

Frequency: 2.4GHz ISM band

Modulation: GFSK (Gaussian Frequency Shift Keying)

Emission power: 4dBm, Class 2

Sensitivity: 84dBm at 0.1% BERCSR

Range of operation (m): 8 ~ 10

Length (mm): 44

Width (mm): 16.5

Height (mm): 7

Weight (gm): 5

5.2.4 Joystick Module

The Joystick module is a very simple self-centered spring-loaded module similar to S2 (PlayStation 2). It consists of two variable resistors and a push button. Variable resistors provide output in the form of analog voltage and push-button provides output in digital form either low state or high state. One Potentiometer is for x-axis control and another Potentiometer is for Y-axis control.

An important thing to note here is that the Joystick features two 10k potentiometers one for each axis that provides variable voltage. The KY-023 joystick module aims to provide two-dimensional (x-axis and y-axis) motion to a microcontroller.

Figure 56 : Joystick Module

5.2.5 Rotary Encoder

A rotary encoder, also called a shaft encoder, is an electro-mechanical device that converts the angular position or motion of a shaft or axle to analog or digital output signals.

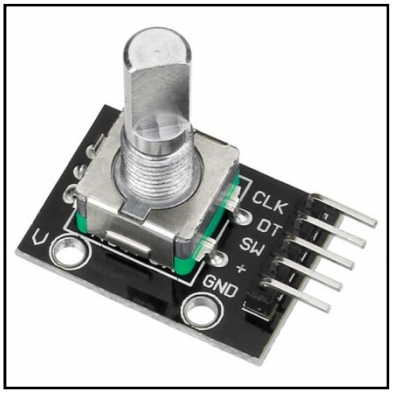


Figure 57 : Rotary Encoder

There are two main types of rotary encoder: absolute and incremental. The output of an absolute encoder indicates the current shaft position, making it an angle transducer. The output of an incremental encoder provides information about the *motion* of the shaft, which typically is processed elsewhere into information such as position, speed and distance.

5.2.6 I2C LCD Display

A typical I2C LCD display consists of a HD44780 based character LCD display and an I2C LCD adapter.

5.2.6.1 Character LCD Display

True to its name, these LCDs are ideal for displaying text/characters only. A 16×2 character LCD, for example, has an LED backlight and can display 32 ASCII characters in two rows with 16 characters on each row.

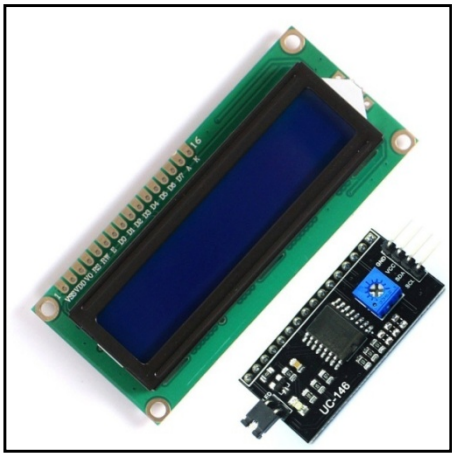


Figure 58 : LCD Display with I2C Module

5.2.6.2 I2C LCD Adapter

At the heart of the adapter is an 8-Bit I/O Expander chip – PCF8574. This chip converts the I2C data from an Arduino into the parallel data required by the LCD display. The board also comes with a small trimpot to make fine adjustments to the contrast of the display. In addition, there is a jumper on the board that supplies power to the backlight. To control the intensity of the backlight, you can remove the jumper and apply an external voltage to the header pin that is marked as ‘LED’.

Chapter 6

ARDUINO CODE

6.1 About C Language

C is a general-purpose, high-level language that was originally developed by Dennis M. Ritchie to develop the UNIX operating system at Bell Labs. C was originally first implemented on the DEC PDP-11 computer in 1972.

It can be defined by the following ways:

1. Mother language
2. System programming language
3. Procedure-oriented programming language
4. Structured programming language
5. Mid-level programming language

6.2 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

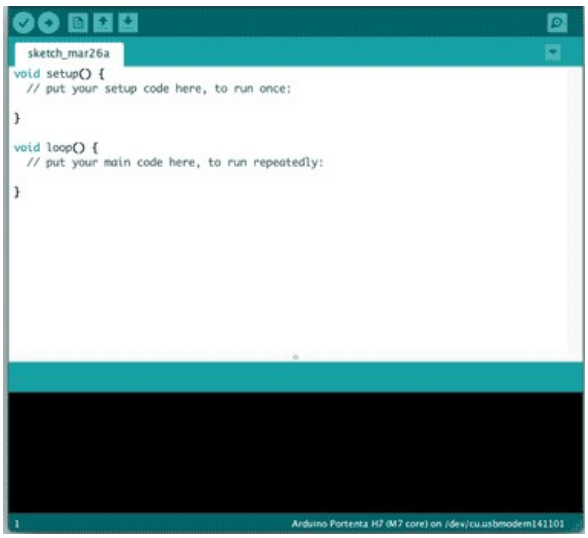


Figure 59 : Arduino IDE

6.2.1 Writing Sketches

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

6.3 Sketches

6.3.1 Version 1 Sketch

```
#include<Servo.h>

// Servo motor name
Servo shoulder;
Servo elbow;
Servo arm;
Servo amro;
Servo grip;
Servo waist;

const int trigpin=5;   // defining ultrasonic sensor position
const int echopin=4;
long duration;
int distance;

void setup() {
  // put your setup code here, to run once:
        pinMode(trigpin,OUTPUT);
        pinMode(echopin,INPUT);
        Serial.begin(9600);
//  defining Servo position on arduino board
        waist.attach(9);
        shoulder.attach(12);
        elbow.attach(11);
        arm.attach(10);
```

```
        amro.attach(8);
        grip.attach(7);
}
void loop() {
  // put your main code here, to run repeatedly:
        amro.write(0);
        grip.write(180);
        arm.write(90);
        digitalWrite(trigpin,LOW);
        delayMicroseconds(2);

        digitalWrite(trigpin,HIGH);
        delayMicroseconds(10);
        digitalWrite(trigpin,LOW);

        duration= pulseIn(echopin,HIGH);
        distance=duration*0.034/2;

        Serial.print("distance ->");
        Serial.println(distance);
        if(distance==5){  // activate scara robot when the object is placed at 5 cm from the
.                       sensor
                shoulder.write(180);
                delay(150);
                shoulder.write(90);
                delay(200);
                 elbow.write(0);
                delay(100);
                 elbow.write(90);
                delay(100);
                 amro.write(180);
                 delay(200);
                grip.write(0);
                delay(200);
                amro.write(90);
                 delay(200);
                arm.write(180);
                delay(200);
                elbow.write(180);
                 delay(170);
                 elbow.write(90);
                 delay(100);
```

```
                shoulder.write(0);
                 delay(170);
                shoulder.write(90);
                delay(100);
                 waist.write(0);
                 delay(1000);
                waist.write(90);
                delay(100);
                shoulder.write(180);
                delay(150);
                shoulder.write(90);
                delay(200);
                elbow.write(0);
                delay(100);
                elbow.write(90);
                delay(100);
                arm.write(90);
                delay(200);
                 amro.write(180);
                 delay(200);
                grip.write(180);
                 delay(200);
                amro.write(90);
                delay(200);
                elbow.write(180);
                elbow.write(90);
                delay(100);
                shoulder.write(0);
                delay(170);
                shoulder.write(90);
                delay(100);
                waist.write(180);
                delay(1000);
                waist.write(90);
                delay(100);

        }

}
```

6.3.2 Servo Motor Check Sketch

```
#include <Servo.h>

Servo servo3;

void setup() {
  // put your setup code here, to run once:
          servo3.attach(9);
}

void loop() {
  // put your main code here, to run repeatedly:
            servo3.write(180);
            delay(50);
            servo3.write(90);
            delay(1000);
            servo3.write(0);
            delay(50);
            servo3.write(90);
           delay(1000);


}
```

6.3.3 Joystick Servo Check Sketch

```
#include <Servo.h>

Servo waist;
Servo shoulder;
Servo elbow;
Servo  arm;
Servo amro;
Servo grip;

int servoVal;
int servoVal2;
int servoVal3;

void setup() {
```

```
          shoulder.attach(12);
          elbow.attach(11);
         waist.attach(9);
          arm.attach(10);
          amro.attach(8);
          grip.attach(7);
}
void loop(){

         servoVal = analogRead(A0);
          servoVal = map(servoVal, 0, 1023, 0, 180);
          shoulder.write(servoVal);
          servoVal = analogRead(A1);
          servoVal = map(servoVal, 0, 1023, 0, 180);
          elbow.write(servoVal);
         servoVal2 = analogRead(A2);
          servoVal2 = map(servoVal2, 0, 1023, 0, 180);
          waist.write(servoVal2);
          servoVal2 = analogRead(A3);
          servoVal2 = map(servoVal2, 0, 1023, 0, 180);
         arm.write(servoVal2);
         servoVal3 = analogRead(A4);
          servoVal3 = map(servoVal3, 0, 1023, 0, 180);
          amro.write(servoVal3);
          servoVal3 = analogRead(A5);
          servoVal3 = map(servoVal3, 0, 1023, 0, 180);
          grip.write(servoVal3);
}
```

6.3.4 I2C-LCD Screen Check Sketch

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27,16,2);


void setup() {
  // put your setup code here, to run once:
         lcd.begin();
         lcd.backlight();
           lcd.setCursor(3,0);
```

```
            lcd.print("Welcome");
           delay(1000);
           lcd.setCursor(1,1);
            lcd.print("Getting ready");
            delay(1000);
            lcd.clear();
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

6.3.5 Rotary encoder Check Sketch

```
int DT=2; //GPIO #4-DT on encoder (Output B)
int CLK=3; //GPIO #5-CLK on encoder (Output A)


int counter = 0;
int angle = 0;
int aState;
int aLastState;
int function=1;



void setup() {
  // put your setup code here, to run once:
            Serial.begin(9600);
           Serial.println(angle);
           pinMode(CLK,INPUT_PULLUP);
            pinMode(DT,INPUT_PULLUP);
           aLastState = digitalRead(CLK);

}

void loop() {
  // put your main code here, to run repeatedly:
```

```
        aState = digitalRead(CLK);

 //Encoder rotation tracking
        if (aState != aLastState){
                if (digitalRead(DT) != aState) {
                        counter ++;
                        angle ++;
                }
                else {
                        counter--;
                        angle --;
                }
                if (counter >=5 ) {
                        counter =5;
                }
                if (counter <=1 ) {
                        counter =1;
                   }
                Serial.println(counter);
                }
         aLastState = aState;

 }
```

6.3.6 Final Code Sketch

```
#include<Servo.h>
#include <IRremote.hpp>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27,16,2);

const int recpin=6;
int pin=13;
int resw=300;
int no=0;
int mov=300;
int rot=90;
int rot1=90;
int rot2=90;
int DT=2; //GPIO #4-DT on encoder (Output B)
```

```
int CLK=3; //GPIO #5-CLK on encoder (Output A)
int counter = 0;
int angle = 0;
int aState;
int aLastState;
int servoVal;
int servoVal2;
int servoVal3;
const int trigpin=5;
const int echopin=4;
long duration;
int distance;


IRrecv irrecv(recpin);
decode_results results;

Servo waist;
Servo shoulder;
Servo elbow;
Servo  arm;
Servo amro;
Servo grip;


//Button press hanlding function


void setup() {
  // put your setup code here, to run once:
          Serial.begin(9600);
          Serial.println(angle);

          pinMode(CLK,INPUT_PULLUP);
          pinMode(DT,INPUT_PULLUP);
          aLastState = digitalRead(CLK);

          waist.attach(9);
          shoulder.attach(12);
          elbow.attach(11);
          arm.attach(10);
         amro.attach(8);
```

```
        grip.attach(7);


        irrecv.enableIRIn();

         lcd.begin();
         lcd.backlight();
         lcd.setCursor(3,0);
         lcd.print("Hello");
        delay(2000);
        lcd.setCursor(1,1);
        lcd.print("Professor");
        delay(2000);

        lcd.clear();
        lcd.setCursor(1,0);
        lcd.print("initialising");
        delay(2000);

        lcd.clear();
        lcd.setCursor(1,0);
        lcd.print("Current");
        lcd.setCursor(1,1);
        lcd.print("Mode = CM");
        delay(2000);

        lcd.clear();

        pinMode(trigpin,OUTPUT);
        pinMode(echopin,INPUT);



}

void loop() {
  // put your main code here, to run repeatedly:

 // counter function for selection of menu

        aState = digitalRead(CLK);
```

```
//Encoder rotation tracking
        if (aState != aLastState){
                if (digitalRead(DT) != aState) {
                        counter ++;
                        angle ++;

                 }
                else {
                        counter--;
                        angle --;

                 }

                if (counter >=4 ) {
                        counter =4;

                }

                if (counter <=1 ) {
                        counter =1;

                }

                if(counter==1){
                        lcd.setCursor(1,0);
                        lcd.print("CM=>IR REMOTE");
                }
                else if(counter==2){
                         lcd.setCursor(1,0);
                        lcd.print("CM=>Pick & ");
                        lcd.setCursor(5,1);
                        lcd.print("Place");
                }
                else if(counter==3){
                        lcd.clear();
                        lcd.setCursor(1,0);
                        lcd.print("CM=>JOYSTICK-C");
                  }
                else if(counter==4){
                        lcd.clear();
                        lcd.setCursor(1,0);
                        lcd.print("DESCRIPTION");
```

```
        }

        Serial.println(counter);
}

aLastState = aState;

 if(counter==1){

        if(irrecv.decode(&results)){
                resw=results.value;
                Serial.println(resw);
                irrecv.resume();
                if(resw==-23971) {
                        no=1;
                        lcd.clear();
                        lcd.setCursor(1,0);
                         lcd.print("Waist Servo");
                        lcd.setCursor(1,1);
                        lcd.print("Selected");
                }

                 else if(resw==25245) {
                        no=2;
                        lcd.clear();
                         lcd.setCursor(1,0);
                        lcd.print("Shoulder Servo");
                         lcd.setCursor(1,1);
                         lcd.print("Selected");
                 }

                else if(resw==-7651) {
                        no=3;
                        lcd.clear();
                        lcd.setCursor(1,0);
                        lcd.print("Elbow Servo");
                        lcd.setCursor(1,1);
                        lcd.print("Selected");
                 }

                else if(resw==8925) {
                         no=4;
```

```
                lcd.clear();
                lcd.setCursor(1,0);
                 lcd.print("ArmServo");
                lcd.setCursor(1,1);
                lcd.print("Selected");
        }
        else if(resw==765) {
                no=5;
                lcd.clear();
                lcd.setCursor(1,0);
                lcd.print(" Amro Servo");
                lcd.setCursor(1,1);
                lcd.print("Selected");
        }
        else if(resw==-15811) {
                no=6;
                lcd.clear();
                lcd.setCursor(1,0);
                 lcd.print("Gripper Servo");
                lcd.setCursor(1,1);
                lcd.print("Selected");
          }

}

if(no==1){    //waist
         if(resw==23205){
                 lcd.clear();
                 lcd.setCursor(1,0);
                 lcd.print("waist CLK");
                 lcd.setCursor(1,1);
                 lcd.print(mov/10);
                waist.write(100);
                delay(mov);
                waist.write(90);
                 resw=0;
        }
        else if(resw==4335){
                 lcd.clear();
                lcd.setCursor(1,0);
                 lcd.print("waist ANCLK");
                 lcd.setCursor(1,1);
```

```
                    lcd.print(mov/10);
                   waist.write(80);
                   delay(mov);
                   waist.write(90);
                    resw=0;
          }
 }
 else if(no==2){   // shoulder
           if(resw==23205){
                   lcd.clear();
                   lcd.setCursor(1,0);
                   lcd.print("Shoulder CLK");
                   lcd.setCursor(1,1);
                   lcd.print(mov/10);
                   shoulder.write(100);
                   delay(mov);
                   shoulder.write(90);
                    resw=0;
           }
          else if(resw==4335){
                   lcd.clear();
                   lcd.setCursor(1,0);
                    lcd.print("Shoulder ANCLK");
                    lcd.setCursor(1,1);
                    lcd.print(mov/10);
                    shoulder.write(80);
                   delay(mov);
                   shoulder.write(90);
                   resw=0;
          }
  }
else if(no==3){   // elbow
          if(resw==23205){
                     lcd.clear();
                     lcd.setCursor(1,0);
                     lcd.print("Elbow CLK");
                     lcd.setCursor(1,1);
                     lcd.print(mov/10);
                     elbow.write(100);
                     delay(mov);
                     elbow.write(90);
                     resw=0;
```

```
        }
        else if(resw==4335){
                lcd.clear();
                 lcd.setCursor(1,0);
                lcd.print("Elbow ANCLK");
                lcd.setCursor(1,1);
                lcd.print(mov/10);
                elbow.write(80);
                delay(mov/4);
                elbow.write(90);
                resw=0;
        }
 }
else if(no==4){   // arm
        if(resw==23205){
                rot=rot+(mov/10);
                 lcd.clear();
                  lcd.setCursor(1,0);
                 lcd.print("ARM Position");
                 lcd.setCursor(1,1);
                 lcd.print(rot);
                 arm.write(rot);
                 resw=0;

        }
        else if(resw==4335){
                   rot=rot-(mov/10);
                   lcd.clear();
                    lcd.setCursor(1,0);
                   lcd.print("ARM Position");
                   lcd.setCursor(1,1);
                   lcd.print(rot);
                   arm.write(rot);
                   resw=0;
        }
 }
 else if(no==5){   // amro
        if(resw==23205){
                    rot1=rot1+(mov/10);
                    lcd.clear();
                   lcd.setCursor(1,0);
                  lcd.print("AMRO Position");
```

```
                    lcd.setCursor(1,1);
                    lcd.print(rot1);
                    amro.write(rot1);
                    resw=0;

          }
           else if(resw==4335){
                    rot1=rot1-(mov/10);
                    lcd.clear();
                     lcd.setCursor(1,0);
                    lcd.print("AMRO Position");
                    lcd.setCursor(1,1);
                    lcd.print(rot1);
                    amro.write(rot1);
                    resw=0;
           }
}
else if(no==6){   // grip
          if(resw==23205){
                     rot2=rot2+(mov/10);
                     lcd.clear();
                     lcd.setCursor(1,0);
                    lcd.print("GRIP Position");
                    lcd.setCursor(1,1);
                    lcd.print(rot2);
                    grip.write(rot2);
                    resw=0;

          }
          else if(resw==4335){
                    rot2=rot2-(mov/10);
                     lcd.clear();
                     lcd.setCursor(1,0);
                    lcd.print("GRIP Position");
                    lcd.setCursor(1,1);
                    lcd.print(rot2);
                    grip.write(rot2);
                    resw=0;
           }
 }
```

```
        if(resw==6375){
                mov=mov+50;
                Serial.println(mov);
                lcd.clear();
                lcd.setCursor(1,0);
                lcd.print("step size=");
                lcd.setCursor(1,1);
                lcd.print(mov);
                resw=0;
        }
        if(resw==19125){
                mov=mov-50;
                lcd.clear();
                Serial.println(mov);
                lcd.setCursor(1,0);
                lcd.print("step size=");
                lcd.setCursor(1,1);
                lcd.print(mov);
                resw=0;
        }

  }
else if(counter==2){
        amro.write(0);
        grip.write(180);
        arm.write(90);
        digitalWrite(trigpin,LOW);
        delayMicroseconds(2);

        digitalWrite(trigpin,HIGH);
        delayMicroseconds(10);
        digitalWrite(trigpin,LOW);

        duration= pulseIn(echopin,HIGH);
        distance=duration*0.034/2;

        Serial.print("distance ->");
        Serial.println(distance);

        lcd.clear();
        lcd.setCursor(1,0);
        lcd.print("distance");
```

```
lcd.setCursor(1,1);
lcd.print(distance);


if(distance==5){
            lcd.clear();
          lcd.setCursor(1,0);
          lcd.print("Object ");
          lcd.setCursor(1,1);
          lcd.print("detected");
          delay(1000);

            shoulder.write(180);
            delay(150);
            shoulder.write(90);
            delay(200);
            elbow.write(0);
            delay(100);
            elbow.write(90);
            delay(100);
            amro.write(180);
            delay(200);
            grip.write(0);
            delay(200);
            amro.write(90);
            delay(200);
            arm.write(180);
            delay(200);
            elbow.write(180);
            delay(170);
            elbow.write(90);
            delay(100);
            shoulder.write(0);
            delay(170);
            shoulder.write(90);
            delay(100);
            waist.write(0);
            delay(1000);
            waist.write(90);
            delay(100);
            shoulder.write(180);
            delay(150);
```

```
            shoulder.write(90);
            delay(200);
            elbow.write(0);
            delay(100);
            elbow.write(90);
            delay(100);
            arm.write(90);
            delay(200);
            amro.write(180);
            delay(200);
            grip.write(180);
            delay(200);
            amro.write(90);
            delay(200);
            elbow.write(180);
            delay(170);
            elbow.write(90);
            delay(100);
            shoulder.write(0);
            delay(170);
            shoulder.write(90);
            delay(100);
            waist.write(180);
            delay(1000);
            waist.write(90);
            delay(100);

        }

}

else if(counter ==3){

        servoVal = analogRead(A0);
        servoVal = map(servoVal, 0, 1023, 0, 180);
        shoulder.write(servoVal);
        servoVal = analogRead(A1);
        servoVal = map(servoVal, 0, 1023, 0, 180);
        elbow.write(servoVal);
        servoVal2 = analogRead(A2);
        servoVal2 = map(servoVal2, 0, 1023, 0, 180);
        waist.write(servoVal2);
```

```
        servoVal2 = analogRead(A3);
        servoVal2 = map(servoVal2, 0, 1023, 0, 180);
        arm.write(servoVal2);
        servoVal3 = analogRead(A4);
        servoVal3 = map(servoVal3, 0, 1023, 0, 180);
        amro.write(servoVal3);
        servoVal3 = analogRead(A5);
        servoVal3 = map(servoVal3, 0, 1023, 0, 180);
        grip.write(servoVal3);
 }
if(counter==4){

        lcd.clear();
        lcd.setCursor(1,0);
        lcd.print("Presenting ");
        lcd.setCursor(1,1);
        lcd.print("");
        delay(2000);

        lcd.clear();
        lcd.setCursor(1,0);
        lcd.print("SCARA ROBOTIC");
        lcd.setCursor(1,1);
        lcd.print("ARM V2");
        delay(2000);

        lcd.clear();
        lcd.setCursor(1,0);
        lcd.print("Graduation ");
        lcd.setCursor(1,1);
        lcd.print("Project");
        delay(2000);

        lcd.clear();
        lcd.setCursor(1,0);
        lcd.print("Created ");
        lcd.setCursor(5,1);
        lcd.print("by");
        delay(2000);

        lcd.clear();
        lcd.setCursor(1,0);
```

```
            lcd.print("Devansh");
            lcd.setCursor(1,1);
            lcd.print("Pahadiya");
            delay(2000);
            lcd.clear();
            lcd.setCursor(1,0);
            lcd.print("Kapil");
            lcd.setCursor(1,1);
            lcd.print("Uniyara");
            delay(2000);

            lcd.clear();
            lcd.setCursor(1,0);
            lcd.print("Pranjal");
            lcd.setCursor(1,1);
            lcd.print("Mittal");
            delay(2000);

            lcd.clear();
            lcd.setCursor(1,0);
            lcd.print("IR Remote");
            lcd.setCursor(1,1);
            lcd.print(" in 2000");
            counter=1;
            delay(2000);


        }

}
```

Chapter 7

WORKING OF SCARA ROBOT

7.1 ARTICULATION

The number of joints a robot has, combined with the number of axes in each joint, determines its degree of freedom. SCARAs are four-axis robots, with motion in the X-Y and Z planes, and 360-degree rotational movement about the Z-axis. Inverse kinematics and data interpolation allow the robot to move dynamically, quickly, and intelligently.

Figure 60 : Articulation IN SCARA Robot

7.2 Work Footprint and Work Envelope

The work envelope or the area of space that a robot can physically reach is a critical consideration. Whether SCARA, Delta or six-axis robots, the lengths of various links, and the limitations of the joint motion are important factors to review.

Often, when choosing between SCARA, Delta, or six-axis robots, the work envelope is the deciding factor. Due to their full rotation about the fixed Z-axis, SCARA robots have a cylindrical work envelope. In some applications, a SCARA's work envelope is limited to the front and side.

The rear may not be useable if cables and pneumatic hoses exit from the back. Still, having the ability to work in all, or mostly all, of their reach allows SCARA robots to maintain a minimal footprint while maximizing workspace.

Figure 61: Footprint of SCARA Robot

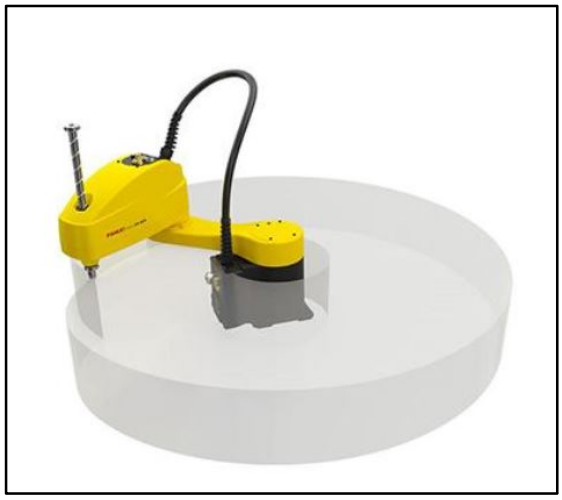


Figure 62: Work Envelope of SCARA Robot

7.3 SPEED

Figure 63: Speed of SCARA Robot

Speed is an important factor when choosing a robot, and SCARAs are typically one of the fastest on the market. With four axes, they have less moving joints and are configured so that J1 and J2 control the X-Y motion, and J3 and J4 control the Z and rotation motion. This

simplifies inverse kinematic calculations, requiring less computational time. When cycle time is critical, consider a SCARA solution.

7.4 Repeatability

SCARA robots typically have the best repeatability performance of all robot types. Errors that occur in the X-Y position are due to having two motors at J1 and J2. Other robot types have three or more motors contributing to the X-Y position. The more motors, the more likely errors could occur. Excellent repeatability is crucial in small assembly applications where tolerances are required to be within several microns.

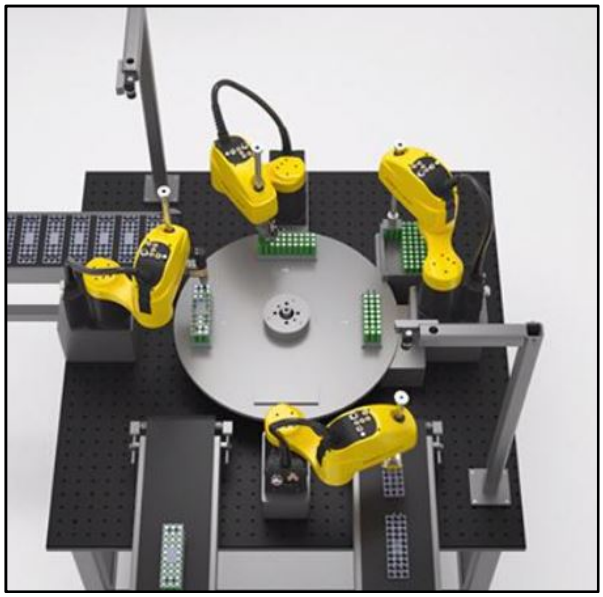


Figure 64: Repeatability of SCARA Robot

BIBLIOGRAPHY

1. https://www.fanuc.eu/de/en/robots/robot-filter-page/scara-series/selection-support#:~:text=SCARA%20Robots%20are%20a%20popular,variety%20of%20material%20handling%20operations.

2. https://diy-robotics.com/blog/scara-robots/

3. https://www.flexibowl.com/scara-robot.html

4. https://www.machinedesign.com/mechanical-motion-systems/article/21831692/the-difference-between-cartesian-sixaxis-and-scara-robots

5. https://www.autodesk.com/products/fusion-360/personal