ES2C6 Electromechanical System Design

2022 Group Project - Robotic Hand

Group F Design Portfolio

University Of Warwick

Team Members

Sam Packer 2103666

Ellen Nixon Anton 2162918

Amirah Olawale-Sanni 2115824

Isabel Okai 2148160

Jody Nwaelene 2100350

Osi Obomighie 2137813

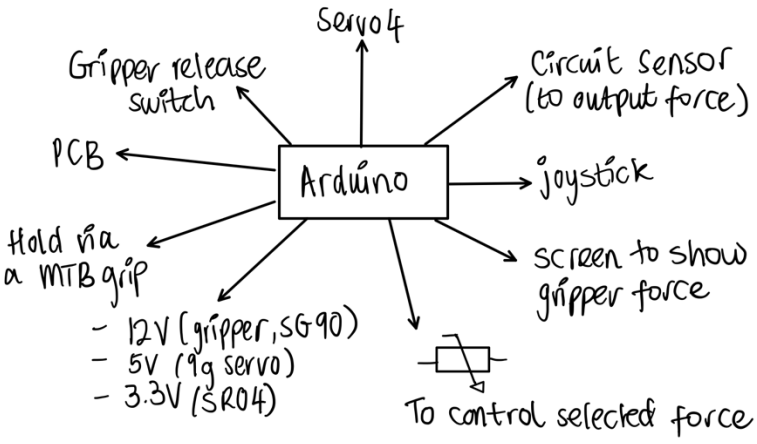
Contents

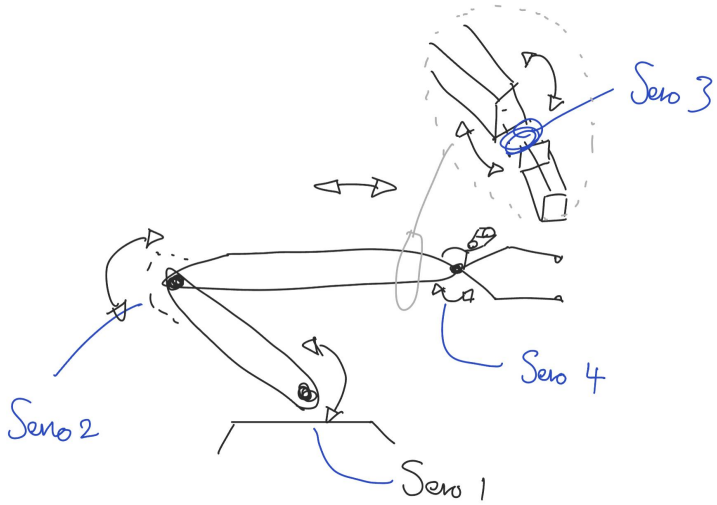
Initial Design Concept & Discussion

In the first session we were given instructions on wiring the robot hand and connecting it with the motor driver and Arduino Uno. This allowed us to gain an understanding on what would need to be done during our project and the particular areas we would need to focus more on such as taking inputs from sensors and using the Arduino to output a feedback response using a feedback loop and PID control.

We collated our thoughts on what we together thought would be the best direction to take the project in. We considered a simple gripper mounted to a handheld plate but thought it would lack complexity where we could develop a more advanced system. Hence, we agreed on instead creating an articulated robotic arm with a gripper mounted at the front and using servo motors to move sections of the arm in different directions and planes to achieve motion.

Pictured are some notes and initial thoughts of the motion of our arm and the components and processes needed to achieve the project.





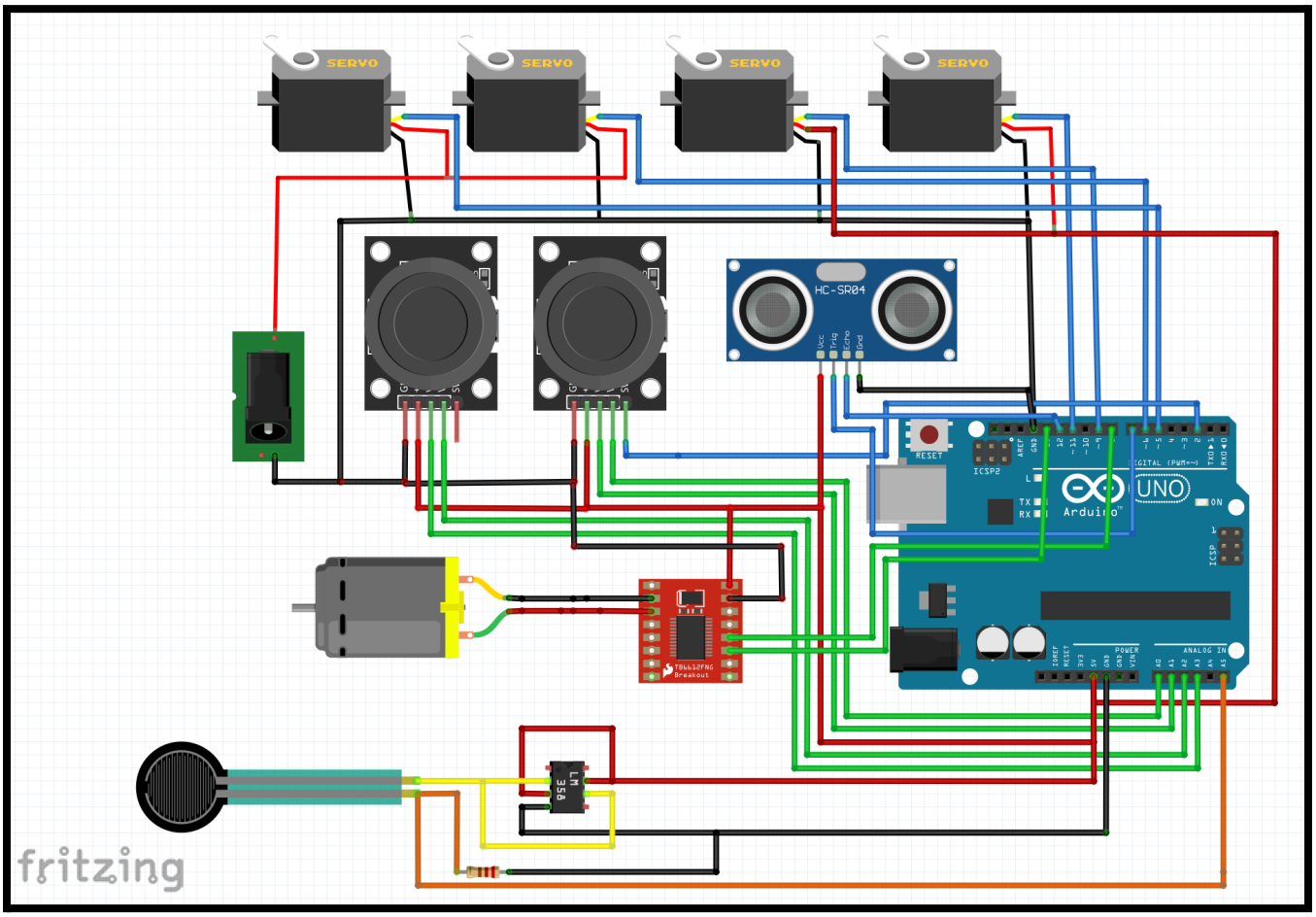
On considering how the robot arm sections would be built, we thought that having internal electrical cable routing would be essential as it keeps loose cables out of the way from the motion of the arm. This prevents the arm from failing through a faulty connection of a cable being pulled out of a socket etc. Given we were using 3D printing, Sam, who has a lot of 3D printing experience decided that the best settings would be 20% Infill with a wall thickness of 1.2mm which offers strong printed components whilst reducing cost and weight of a higher infill print.

We also considered how the arm when assembled would be placed relative to the Arduino Uno board. We decided to mount the Arduino, along with accompanying electrical equipment such as the Makeblock controller underneath the robotic arm. This allowed for a lower center of mass of the arm which is crucial to consider in this project given that if mass is placed further out on an arm section then the servo motors will have to work at a higher power to hold that given position. This will mean that a higher power source, or a LiPo with a high discharge rate is needed to power the system. In other words, power consumption of the system has to be minimized when using power hungry SG90 servos.

Design

Electrical Design

Below is the circuit schematic for our system

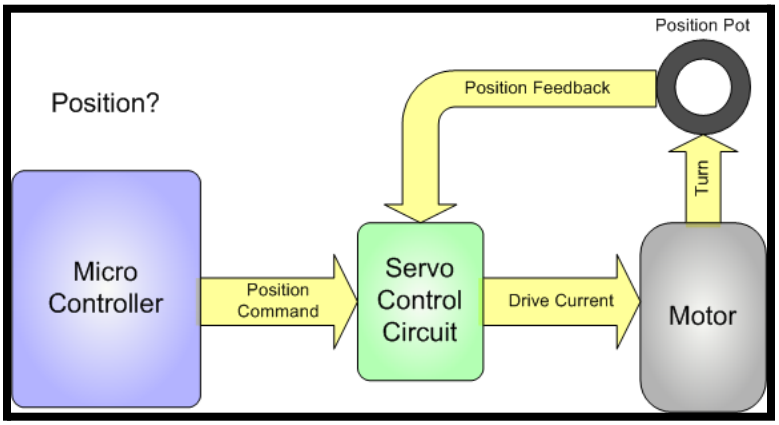


From the schematic, you can see we have two main power delivery parallel circuits, a 12V and a 5V. The circuit will be powered by a 12V LiPo. Two joysticks are used to control the arm via built in joysticks.

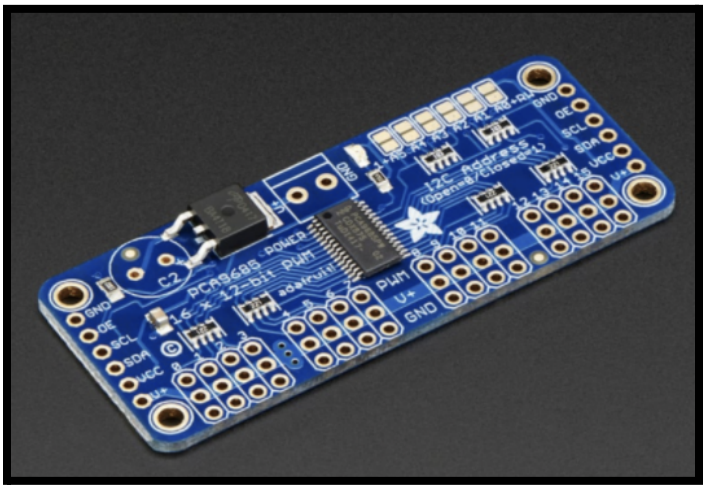
Most robots in the world are designed for heavy, repetitive manufacturing work. They handle tasks that are difficult, dangerous, or boring to human beings. For example, the robotic arm is frequently used in manufacturing roles. A typical robotic arm is made up of seven metal segments, joined by six joints. The computer controls the robot by rotating individual servo motors connected to each joint (some larger arms use hydraulics or pneumatics).

Unlike ordinary motors, servo motors consist of a DC motor and a potentiometer attached to the axle of the output of the DC motor. This potentiometer, often 10KOhm, gives a feedback signal to the onboard processor of the servo motor which computes this as a degree of motion from 0 to 270. The

servo motor requires a PWM input which can be delivered from the Arduino Uno using one of the 5 PWM pins. This diagram below explains through a diagram this concept.



Given the motor controller for the Makeblock gripper also requires one PWM input, that means we are limited to 4 servo motors using solely the Arduino Uno to control them. Knowing this influenced the design of our arm mechanism and we decided to not include rotational motion about the origin of the arm. We did look into the use of a servo control module from Adafruit such as the 16 servo board pictured below that can operate up to 16 servo motors using only Tx and Rx pins with I2C communication. However, we felt we did not need to pursue this.



One of the main concerns when working high power MG996 servos in such a project is if we have enough space to incorporate a suitable power supply which can supply the stall current (2.5A @12v per MG996 servo) plus power the 9g Micro Servo (0.75A @5v per servo SG90) and the gripper (0.110A @ 12v) and have additional current to power the arduino (minimal, expected to be sub 0.2A at 5v). This means our battery must be able to, at peak servo power, supply 69.82W. From a nominal voltage of 11.1 V (3S lip0) this means a current draw of 6.29A. To see if our battery can supply such power we

need to take in consideration the C rating for the battery. The C Rating is defined by the rate of time it takes to charge or discharge a battery. Batteries with a higher C rating can discharge at a faster rate and hence have a higher max operating current. Given the battery we have available is 2500mAh and has a C rating of 4, this means our max instantaneous discharge current of the battery is 10A since $2.5AH \times 4C = 10A$. Given our max current of the system is 6.29A, 10A is sufficient.

Code Development and PID Control System

When considering how to operate the servos with the joystick, a lot of the example code files from Arduino, Whadda(the manufacturer of the joystick) and the internet simply converted the input from the built in potentiometers and processed that to output a given degree for the servo to move to. This, in principle, sounds like it would work but in reality the joysticks have springs to return them to an idle 0 0 position. This means as soon as the operator were to let go of the given joystick position, the servos would return to their zero degree position which is not only an inconvenience but also potentially dangerous to the structure of the arm.

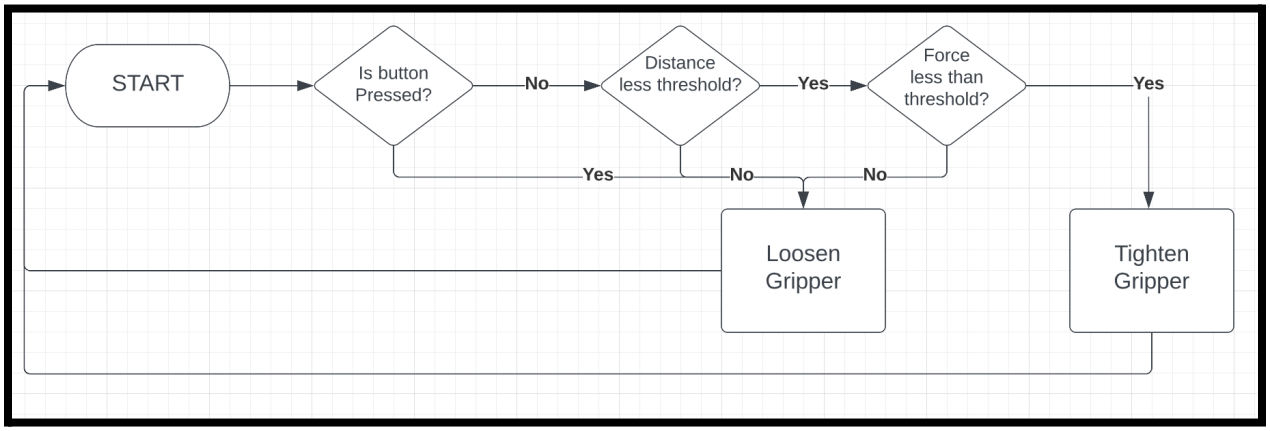
So, the fix to this is to follow the following pseudo code

```
READ Joystick Input Value
IF Joystick Input Value < 340
        THEN Servo motor move -4 degree
ELSE IF Joystick Input Value > 680
        THEN Servo motor move +4 degree
```

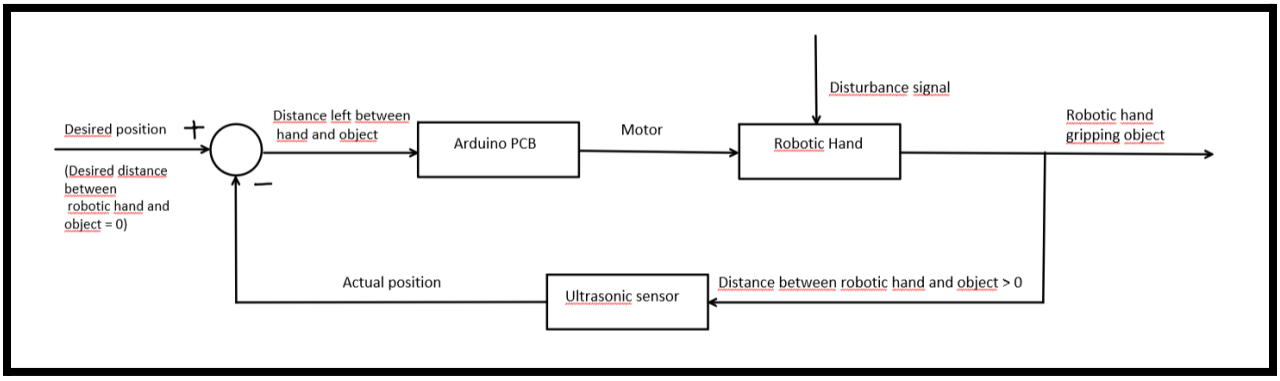
This section of code can be seen in our project code from line 33 onwards.

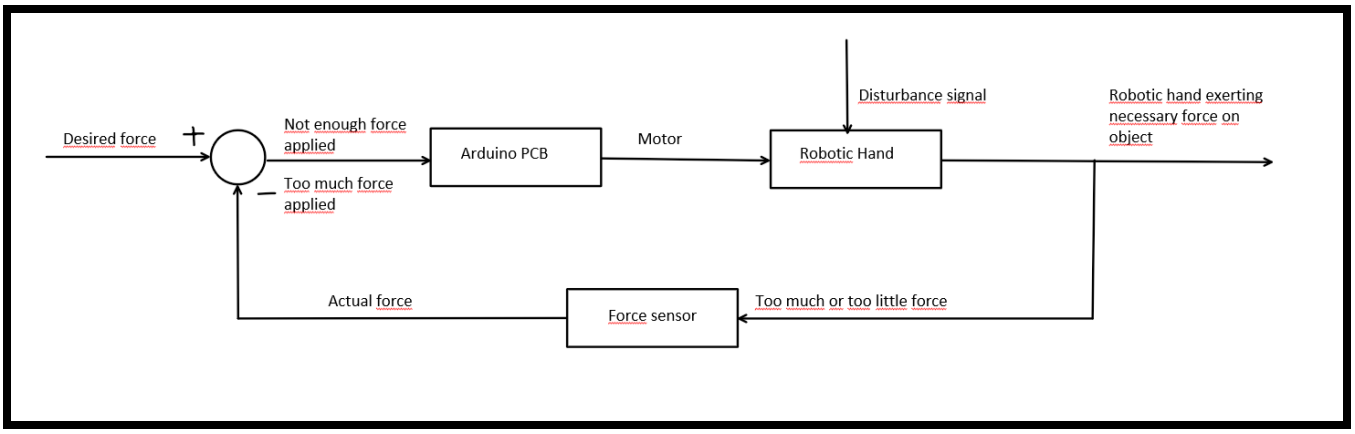
This means that the servo motor holds the position it's at even when the joystick returns to the origin which aids stability of operation (less noise input from human tremble) and makes the arm operate in a smoother fashion with fewer sudden ‘jerk’ movements which draw a lot of power.

The below flow chart outlines the main operation of our sensing system for the robotic hand

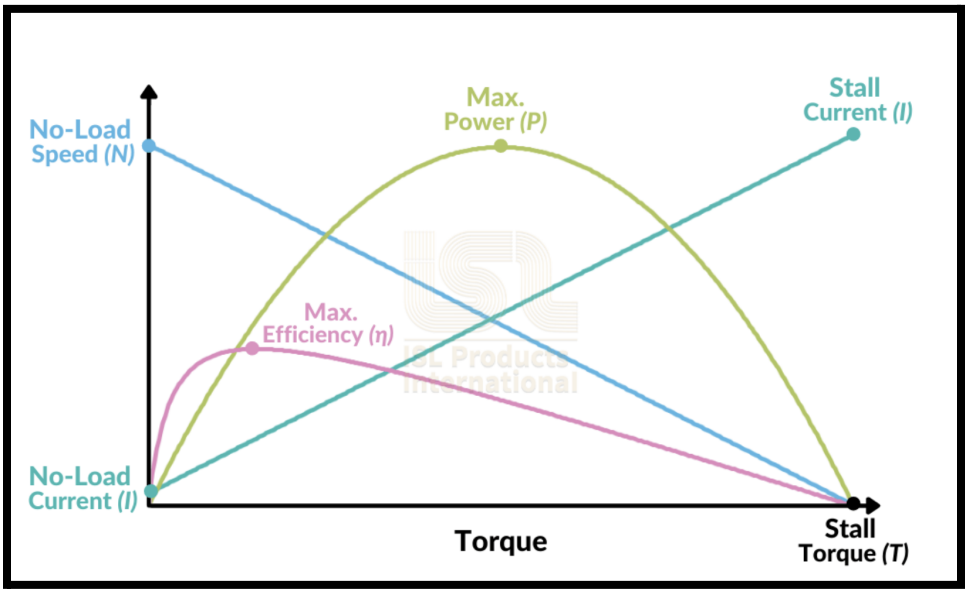


When deciding about the design of our robotic hand and its functioning, we had to take into account our feedback system and how it would work. First we thought about the possible feedback control systems the hand could have, and we came up with two: the distance and the force feedback shown below.





But, after considering the use of a current sensor instead of a force sensor, we had to adapt our approach here. It is a given that when the gripper exerts a higher grip force, the motor is operating at a higher power (so a higher current draw) to output a greater torque. The below graph shows that a



linear relationship in the form y =mx + c exists for this.

We ended up using the force sensor in place of the current sensor due to time constraints limiting us finding a suitable force sensor which had a high sensitivity to detect small current changes such as 0.0001mA; if time allowed, this was where we would have liked to extend the project.

Health & Safety

When powering an Arduino from a PC/laptop, it is recommended that one should remove external power supplies from the circuit. If we wired the circuit incorrectly, this could cause damage to the connected computer, so we ensured to double-check the wiring before connecting it to the computer. We took extra care when soldering so that the components on the Arduino were not damaged from heat conduction up the metal parts of the components, especially when soldering to the force sensor with its thin plastic membrane construction. By only powering our electronics, when necessary, we were able to minimize risks to everyone's safety and mitigate premature component failure.

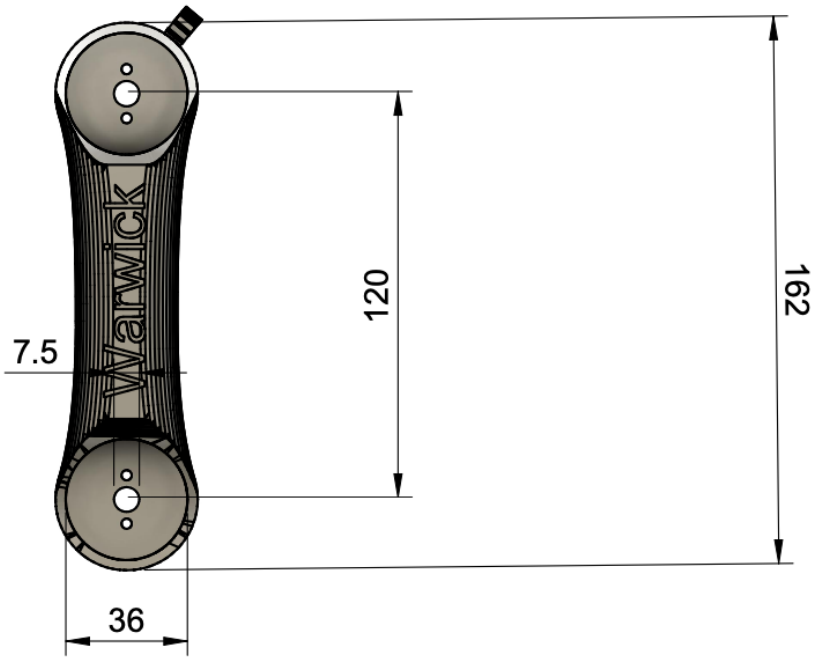
Component List & Budgeting

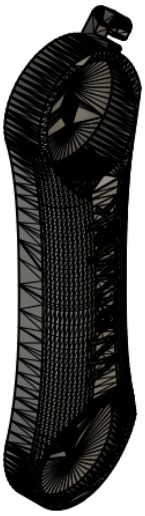
The project has a budget of £200 excluding the cost of the component starter set provided. This left us with £100 to purchase additional components which we used to buy the components as shown here.

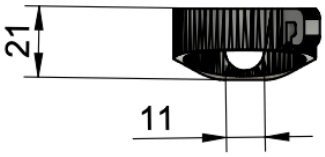
Component Description	Manufacturer	Manufacturer Part Number	Supplier	Supplier Part Number	Quantity	Unit Cost (£)	Total (£)
Servo SG90	Rapid	37-1330	Rapid	37-1330	1	5.03	5.03
Whadda Joystick x2	Whadda	73-4637	Rapid	73-4637	1	9.07	9.07
Hi Tec Servo	HiTec	64-1479	Rapid	64-1479	2	14.4	28.80
3D Print PLA	eBay	n/a	eBay	n/a	~200g	9.99/Kg	2.00
Wires	Rapid	JW-D1-MF	Rapid	JW-D1-MF	4	3.11	12.44
Total							

CAD Drawings

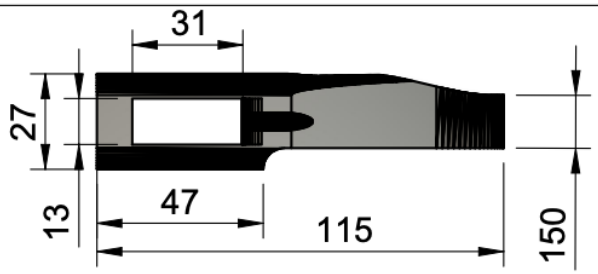
The following engineering drawings outline in detail the components used in the production of the robotic arm.

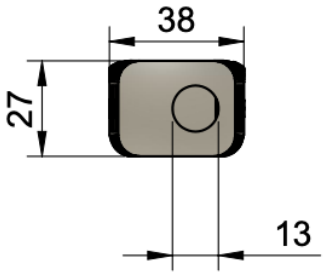


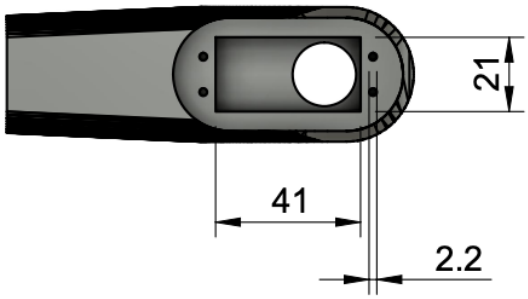


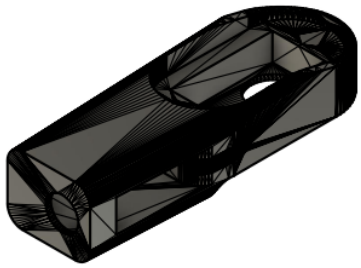


Dept.	Technical reference	Created by	Approved by
	Arm 01	Sam Packer 22/11/2022	
		Document type	Document status
		Title: Arm 01 v1	DWG No.: 01
			Rev. / Date of issue / Sheet: 1/1

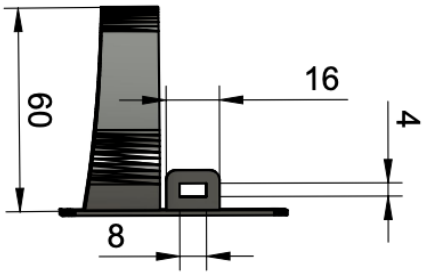


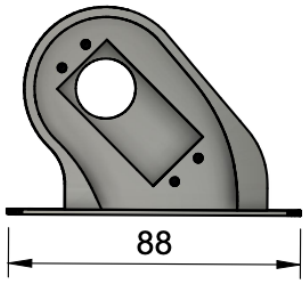


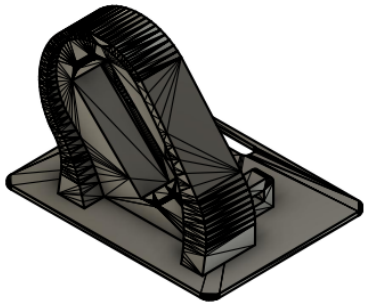




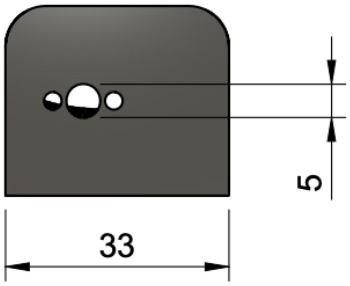
Dept.	Technical reference	Created by	Approved by
	Arm 02	Sam Packer 22/11/2022	
		Document type	Document status
		Title: Arm 02 v3	DWG No.: 02
			Rev. / Date of issue / Sheet: 1/1

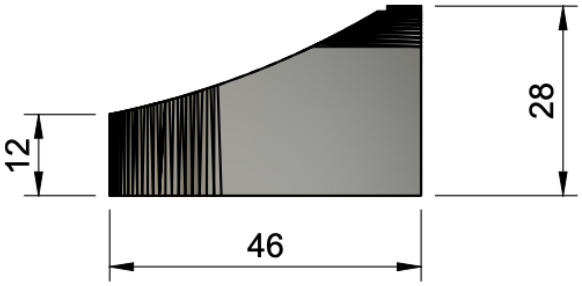


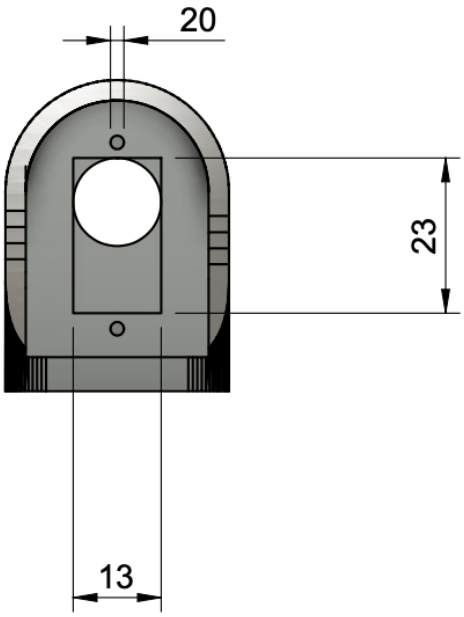


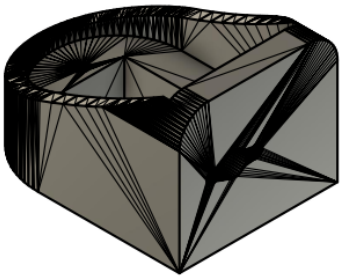


Dept.	Technical reference	Created by	Approved by
	Base	Sam Packer 22/11/2022	
		Document type	Document status
		Title: Base v2	DWG No.: 03
			Rev. / Date of issue / Sheet: 1/1

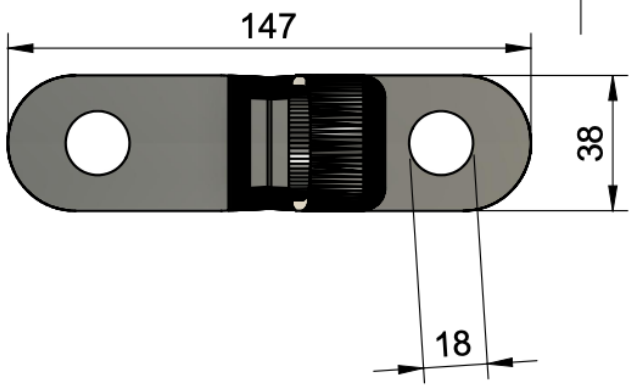


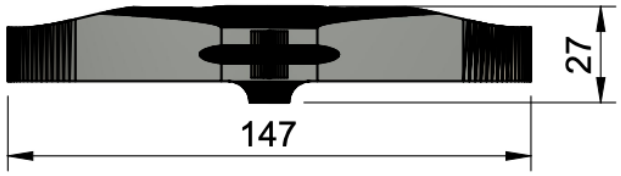


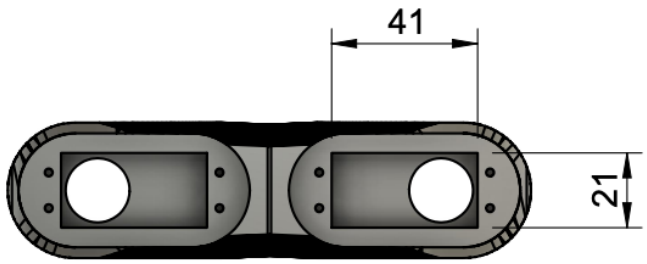


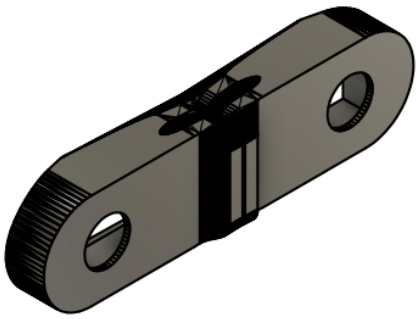


Dept.	Technical reference	Created by	Approved by
	Arm 3	Sam Packer 22/11/2022	
		Document type	Document status
		Title	DWG No.
		Arm 03	004
			Rev. / Date of issue / Sheet: 1/1

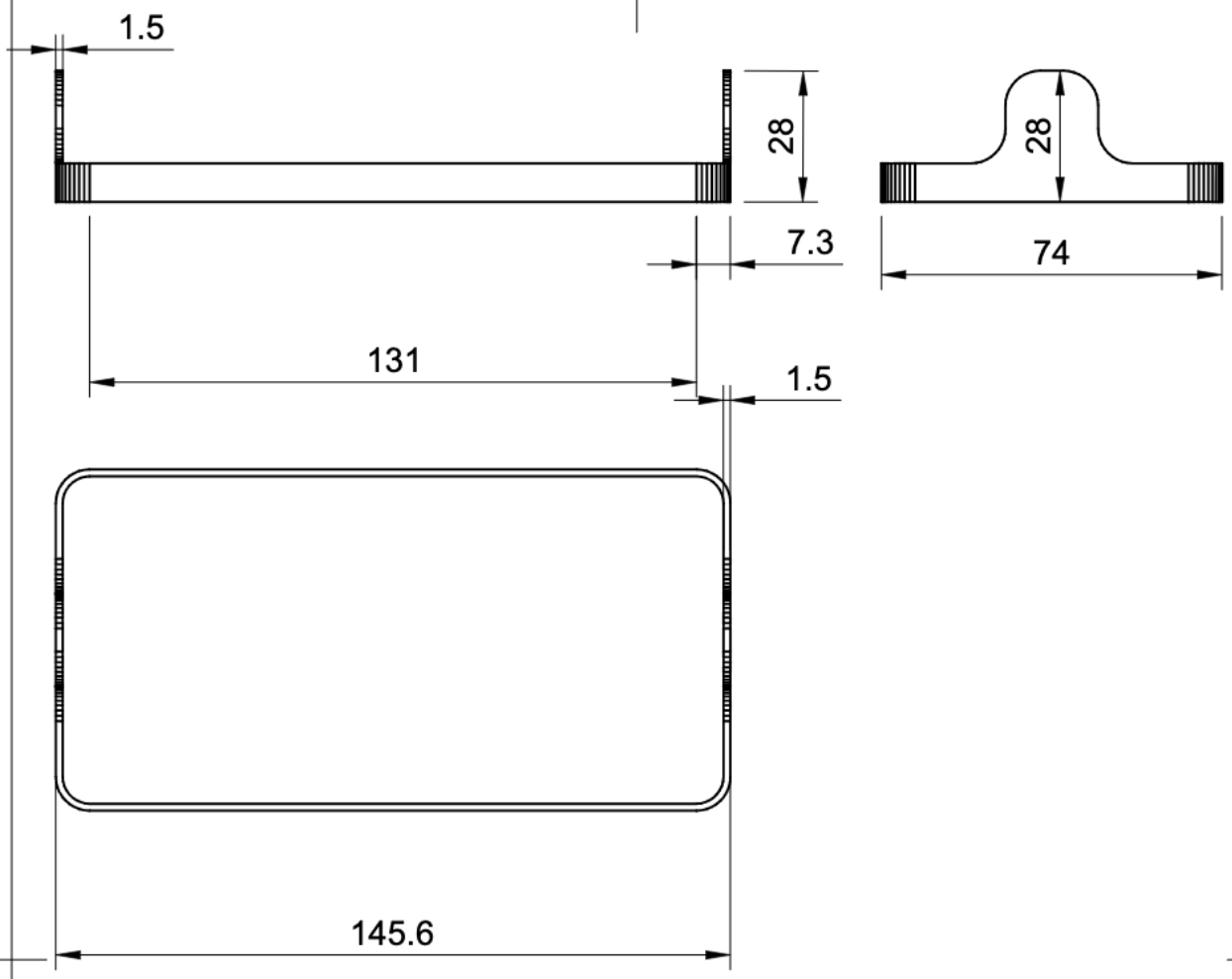


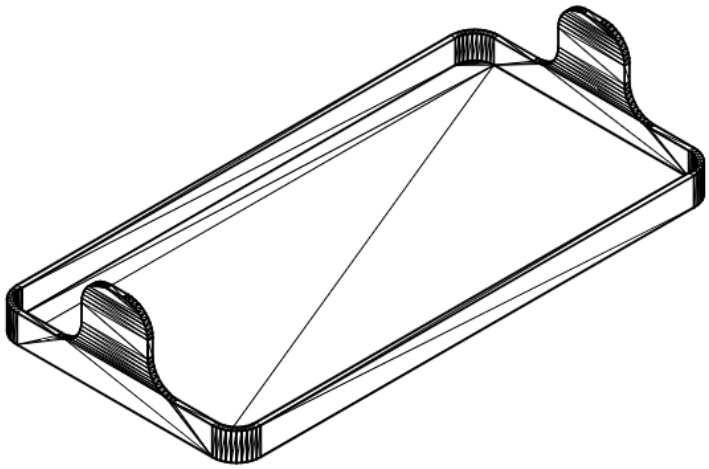




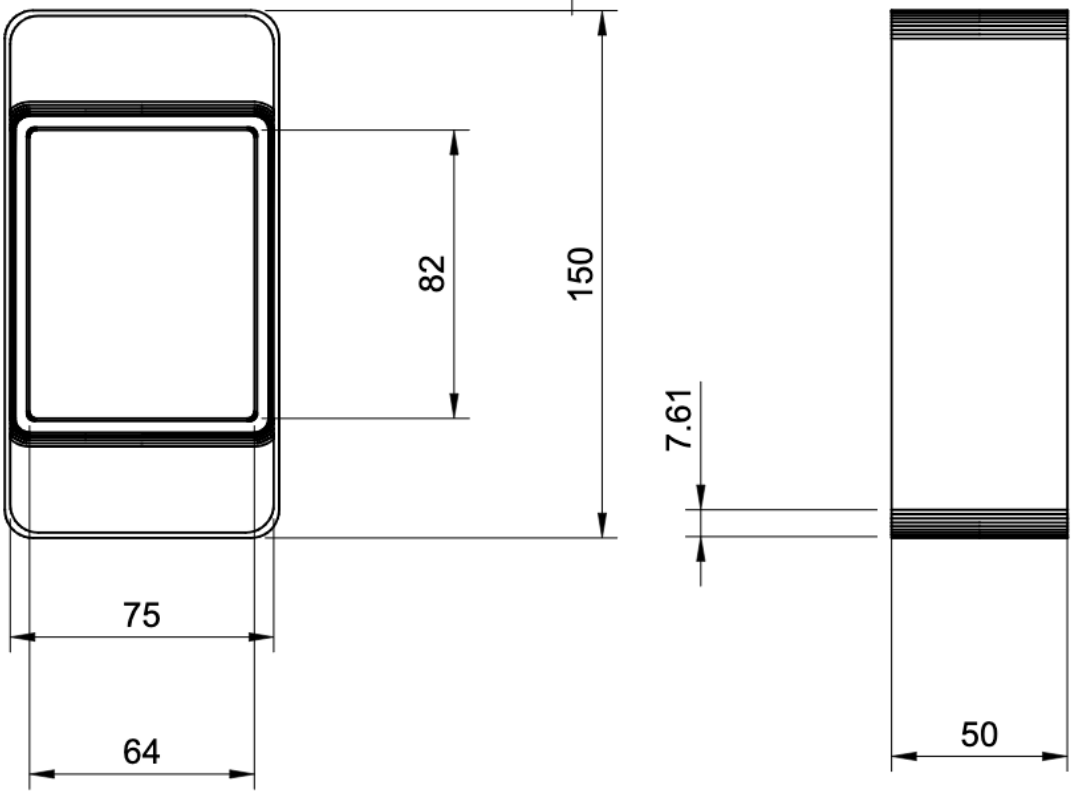


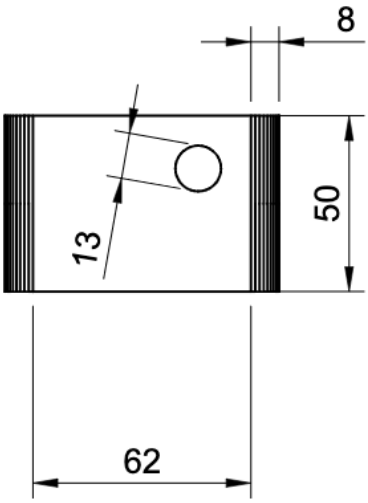
Dept.	Technical reference	Created by	Approved by
	Arm2 v2	sam packer 02/12/2022	
		Document type	Document status
		Title: Arm 02 v4	DWG No.: 005
			Rev. / Date of issue / Sheet: 1/1

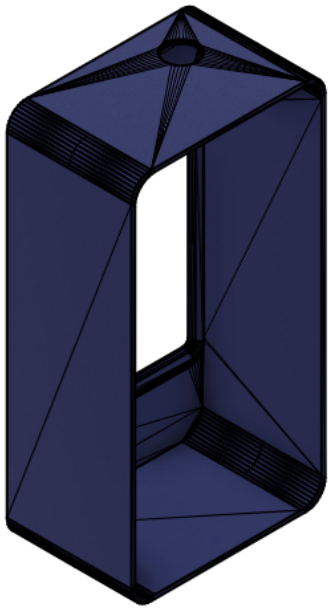




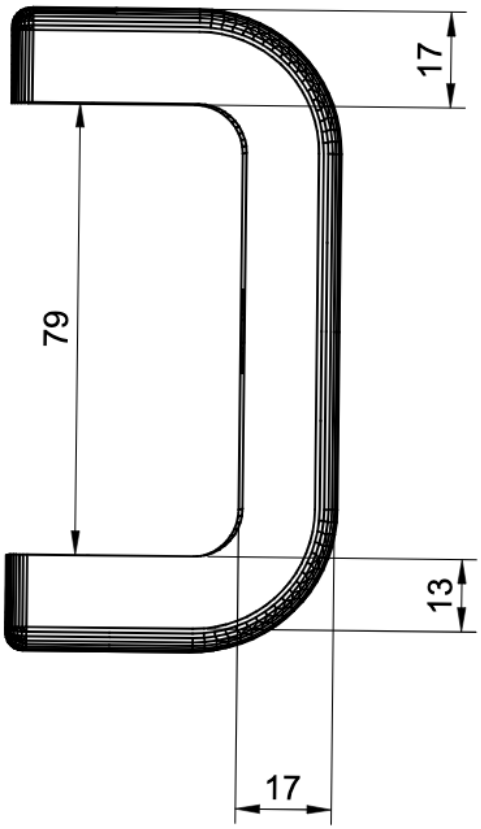
Dept.	Technical reference	Created by Jody Nwaelene	Approved by	
		Document type	Document status	
		Title robot arm casing drawing	DWG No. 006	
			Rev.	Date of issue / Sheet 1/1

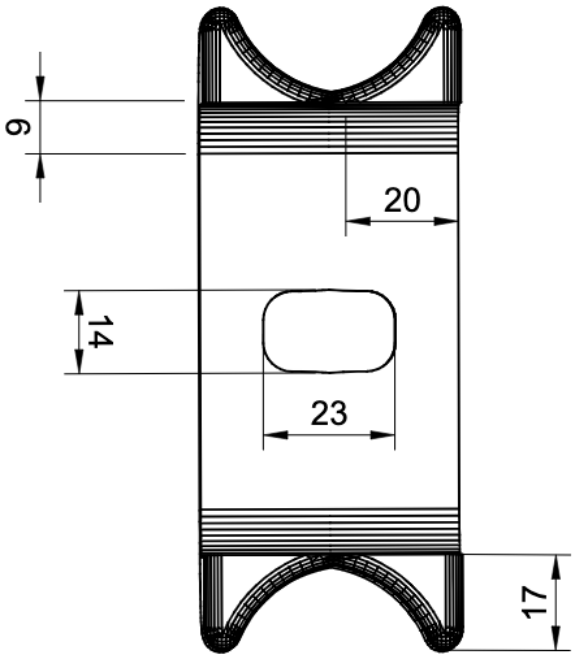


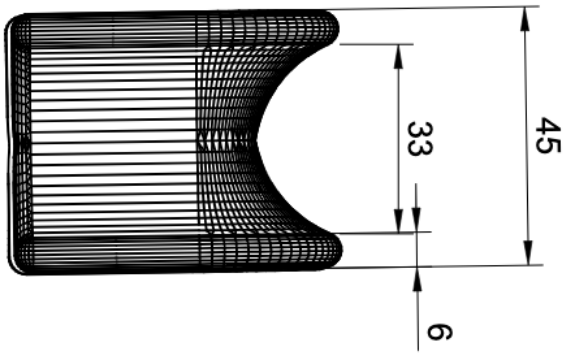


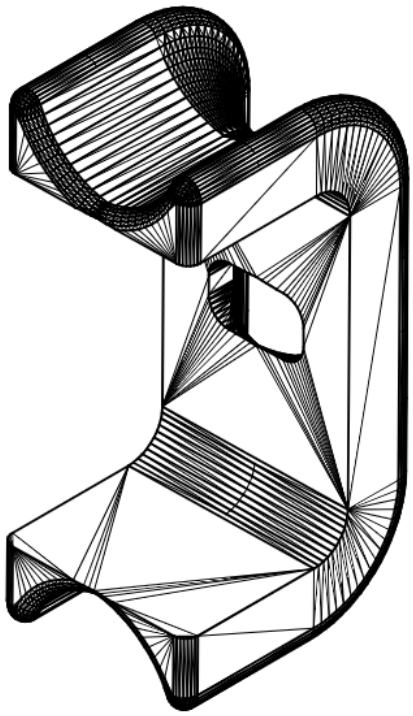


Dept.	Technical reference	Created by	Approved by
		Jody Nwaelene	
		Document type	Document status
		Title	DWG No.
		robot arm casing eng drawing	007
			Rev. / Date of issue / Sheet: 1/1



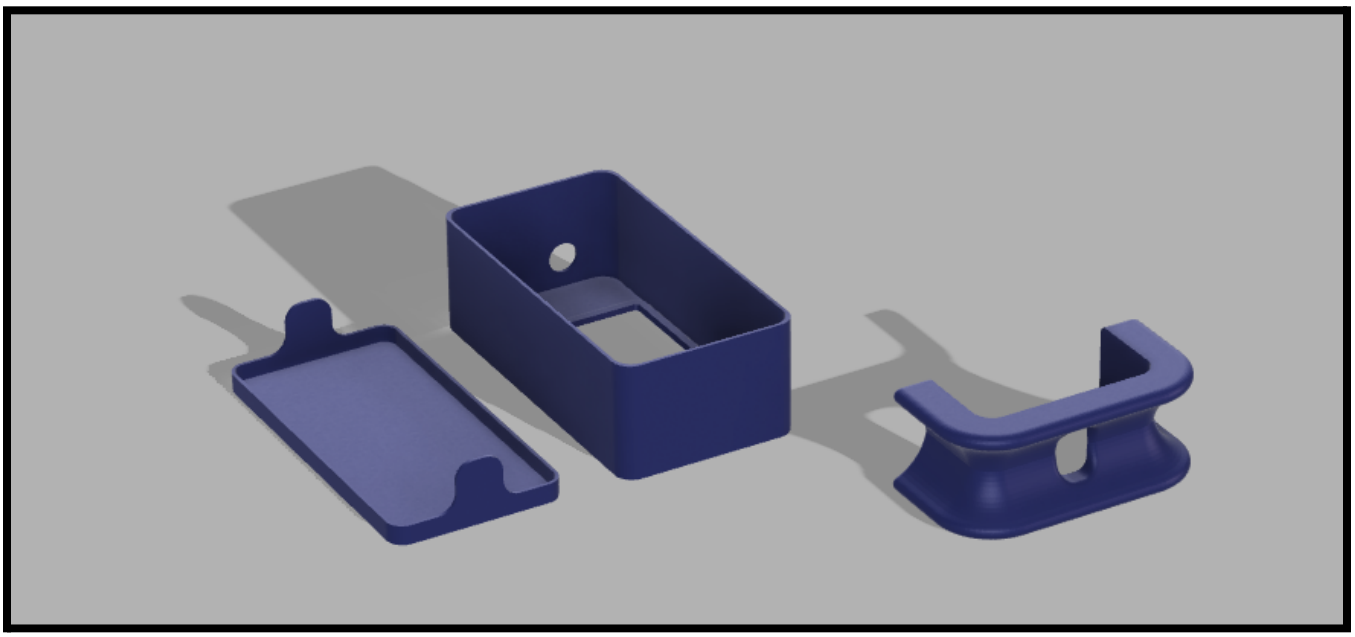


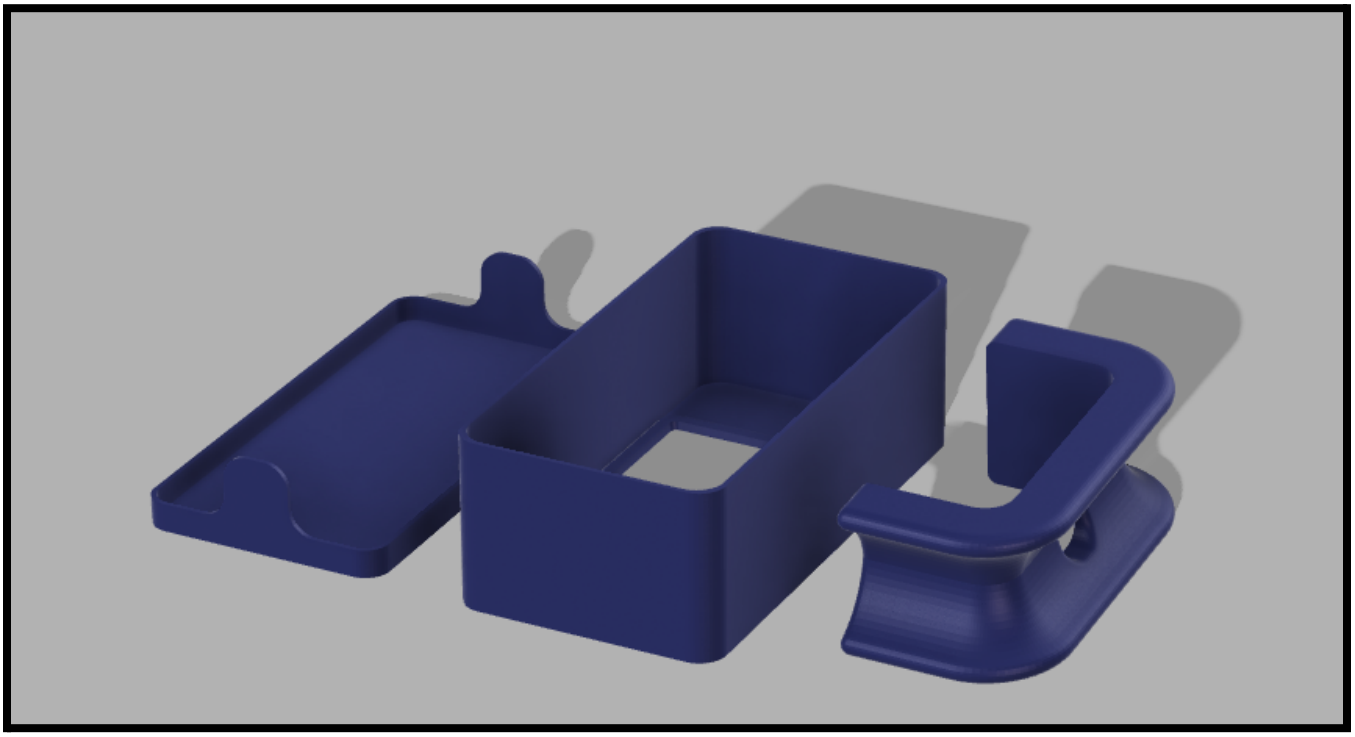


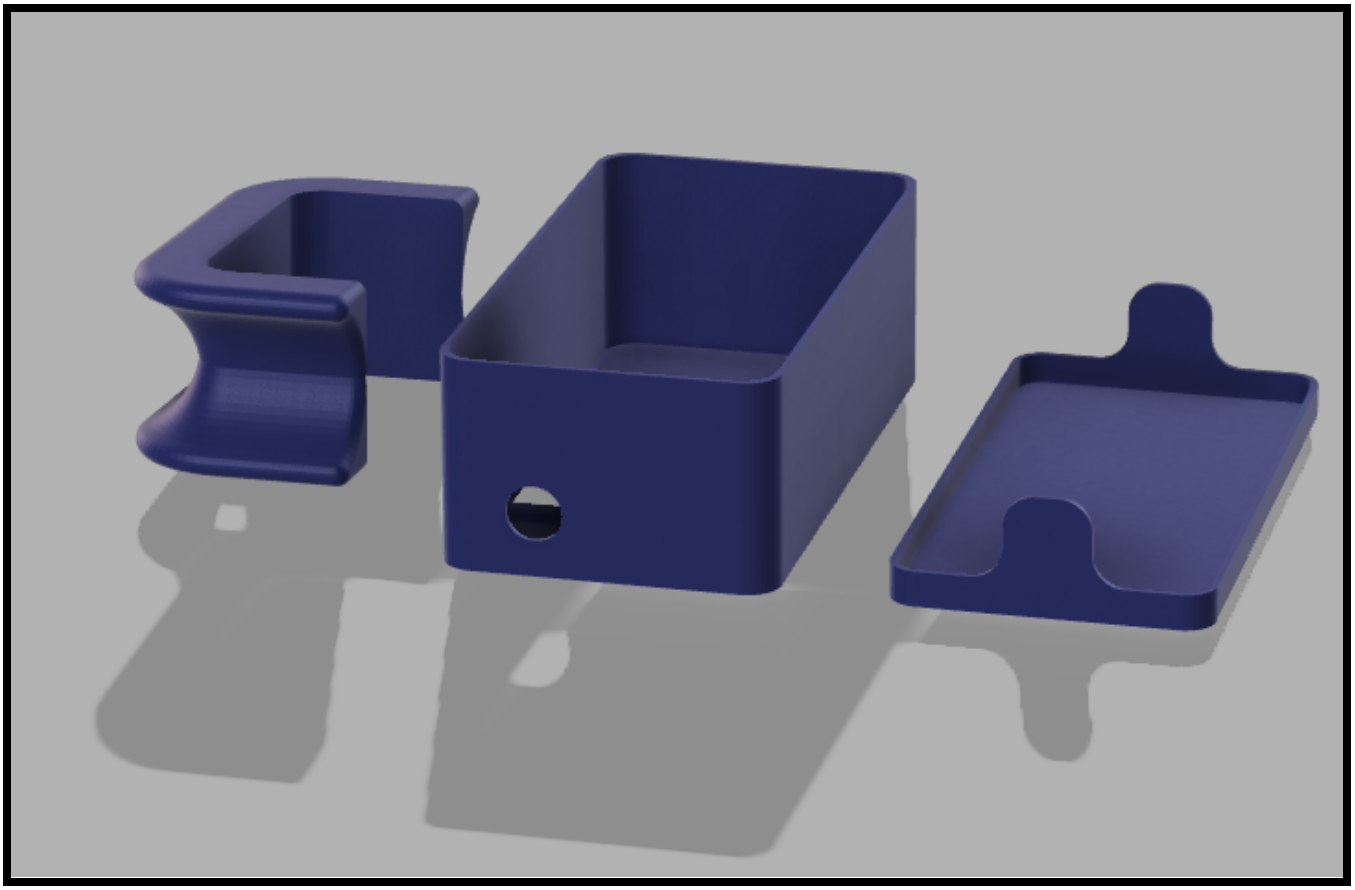


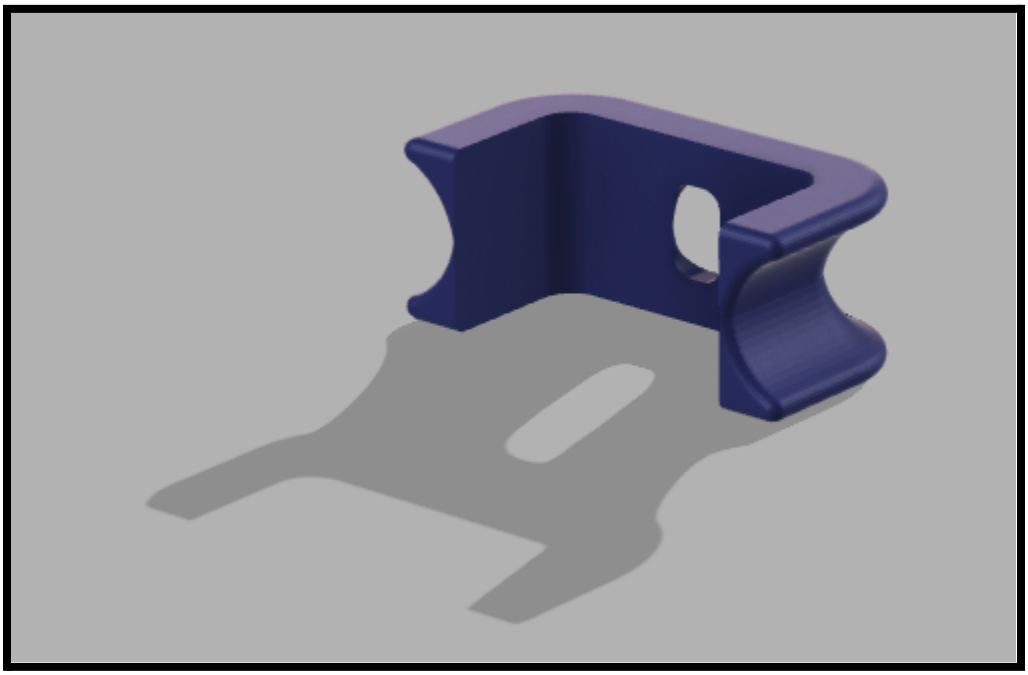
Dept.	Technical reference	Created by	Approved by
		Jody Nwaelene	
		Document type	Document status
		Title	DWG No.
		robot arm casing eng drawing 8	008
			Rev. / Date of issue / Sheet: 1/1

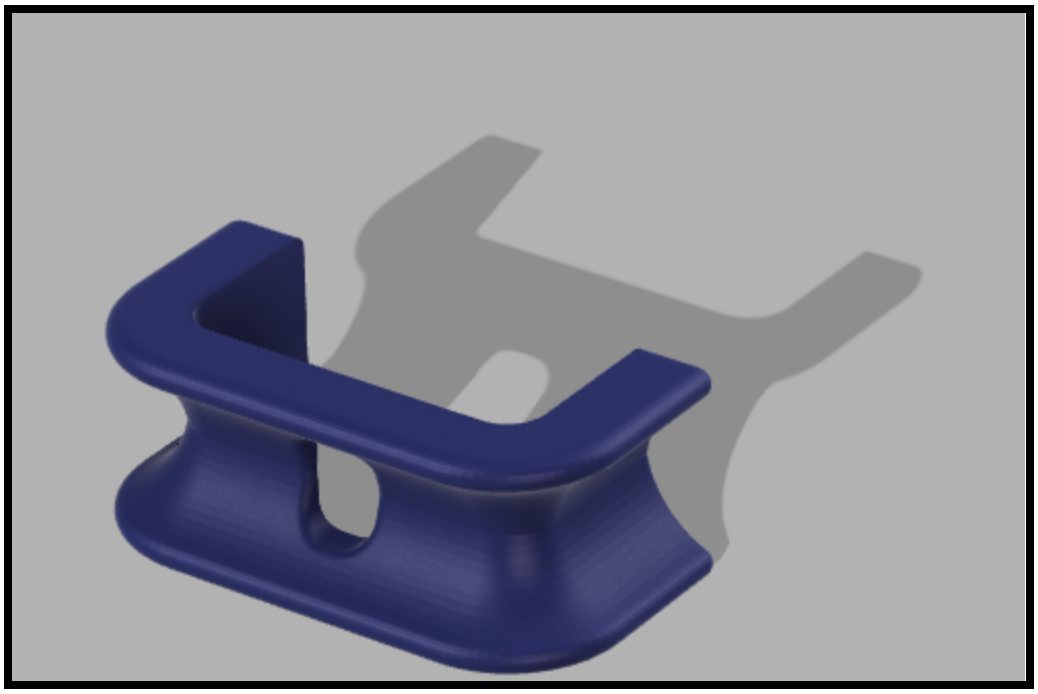
Casing Renders

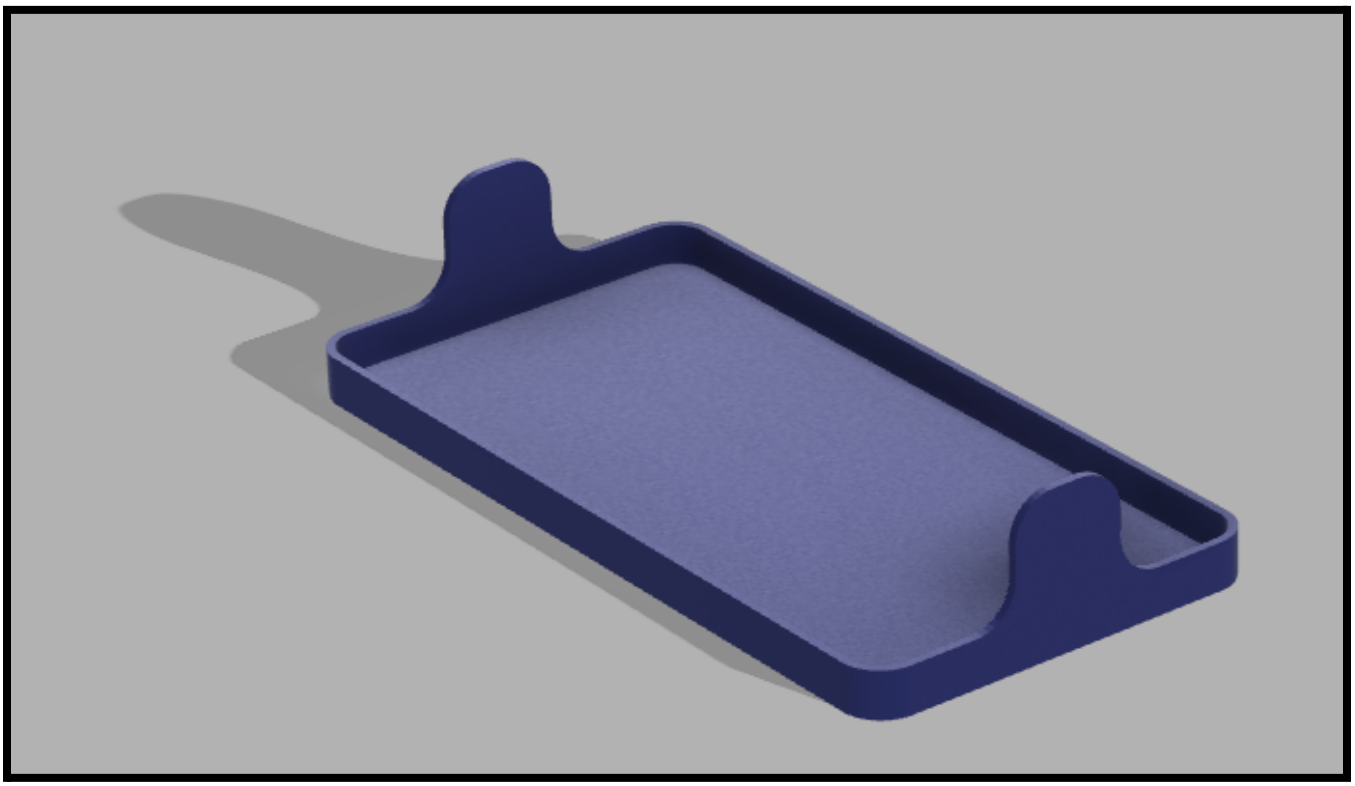


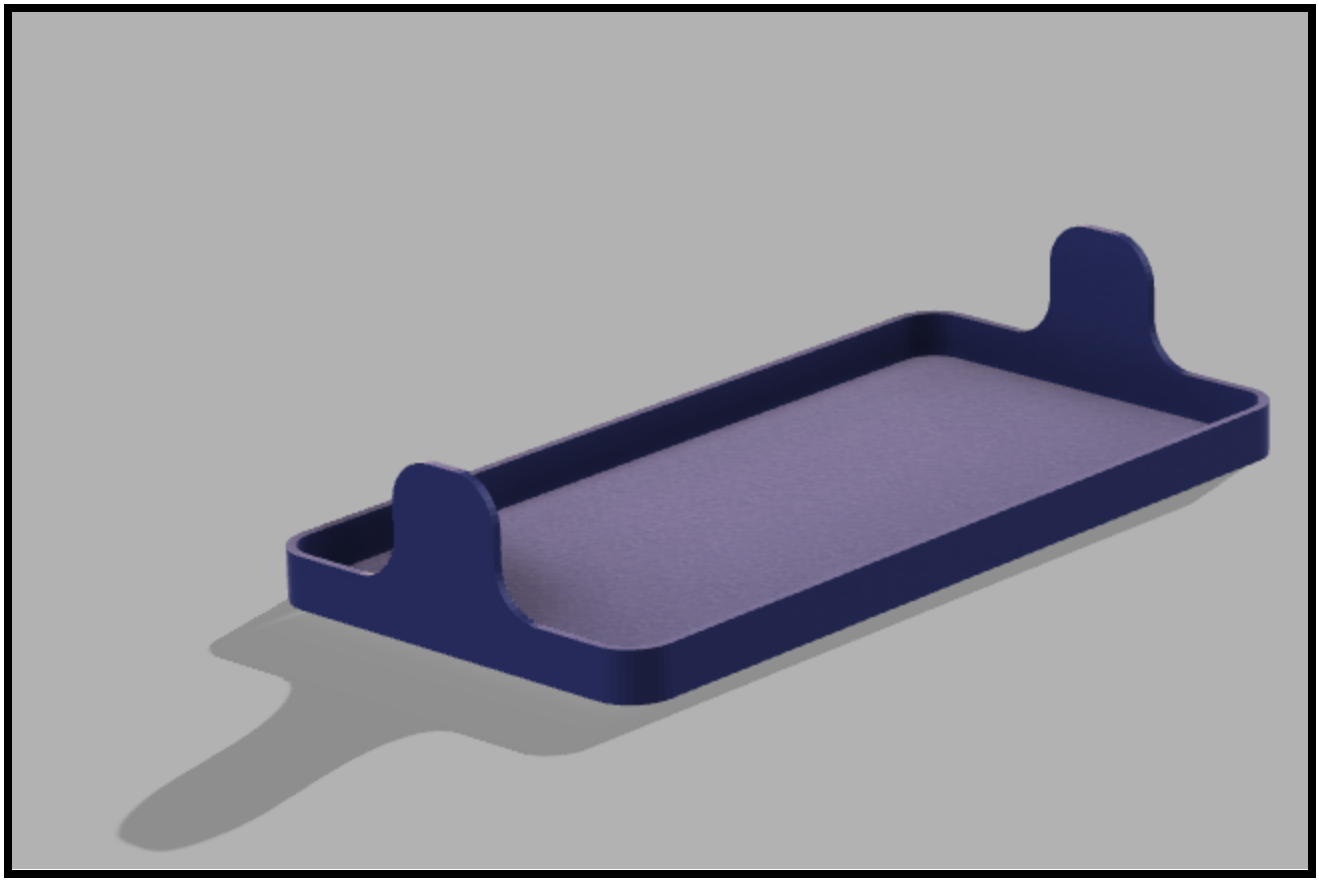


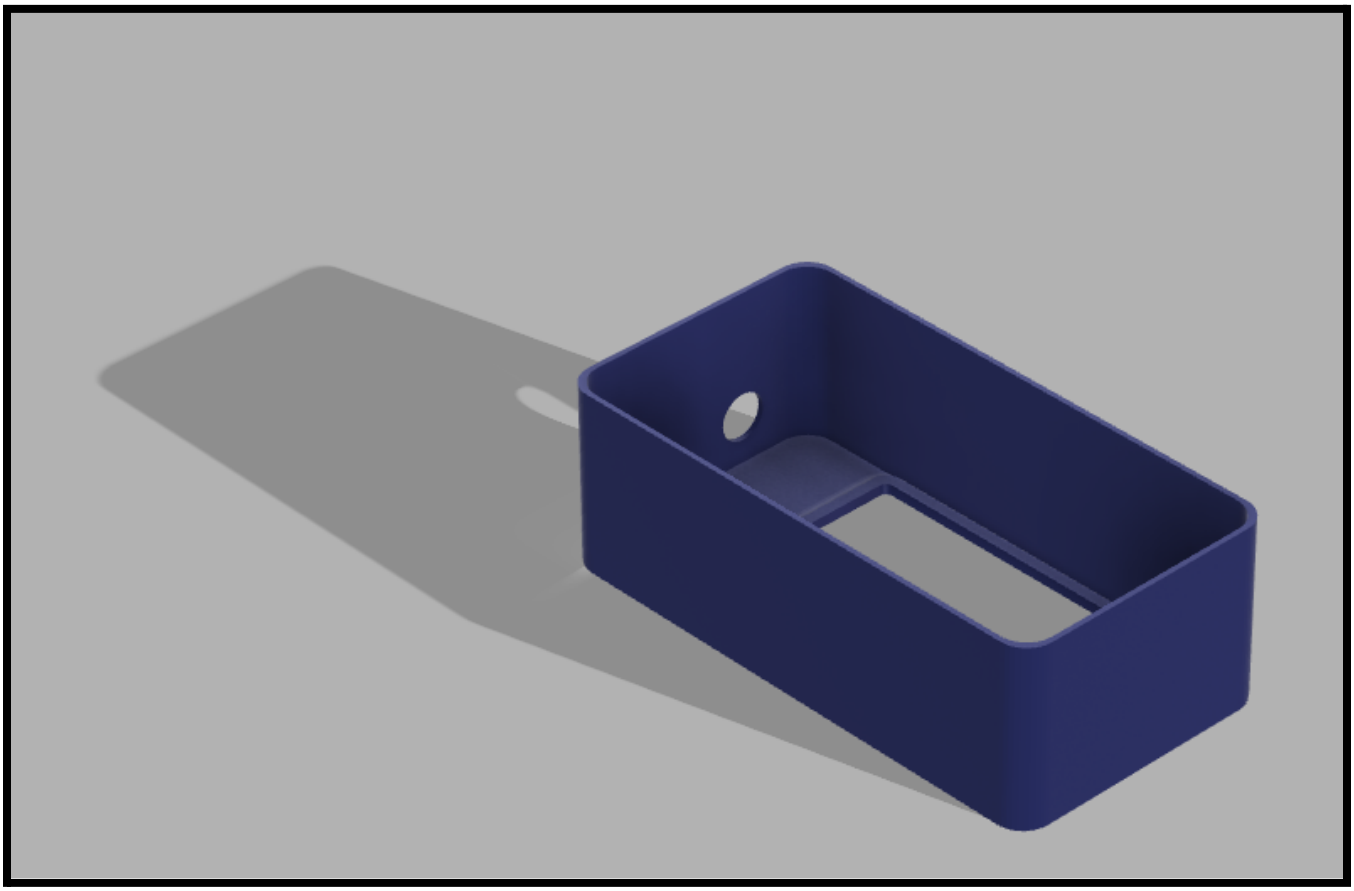












Testing + Troubleshooting

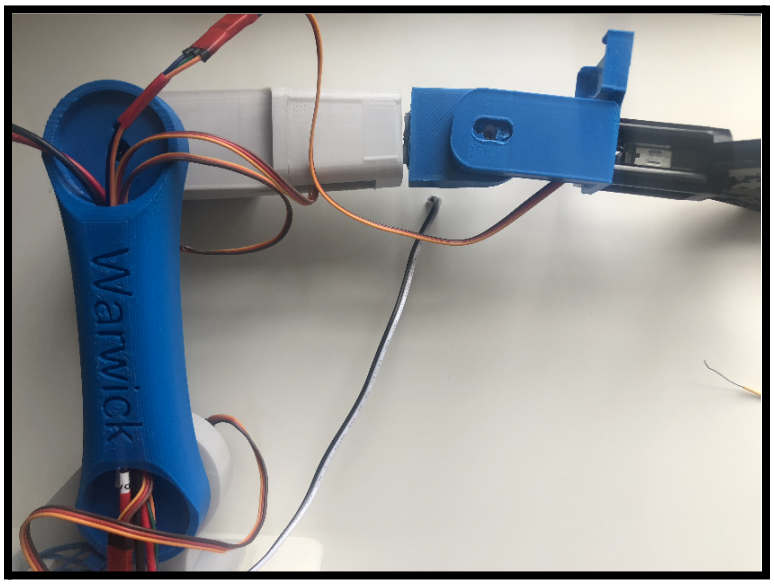
The following testing checklist has been developed to check our robot arm fulfills specific details given from the project brief PDF.

Test	***Outcome***	***Notes***
Do servo 1 and 2 move in accordance with the joystick input?	NO	High power is needed for servo 1 and 2 SG90 Shaking motion observed where the arm enters a state of oscillation. See below for fix
Do servo 3 and 4 move in accordance with the joystick input?	Yes	Smaller servo steps aids usability
Does the onboard button on the joystick read serial HIGH on the arduino?	Yes	Using INPUT_PULLUP
Does the ultrasonic provide accurate (within 10mm) distance measurements on serial?	Yes	First sensor was broken, second worked fine Also was given false positive values when in 3D Printed enclosure so we brought it closer to gripper which fixed situation
Does the gripper loosen if distance is greater than 10cm or button is pressed?	Yes	
Does the gripper tighten if distance is less than 10cm?	Yes	Ultrasonic sensor placement is crucial to avoid false +ve readings.

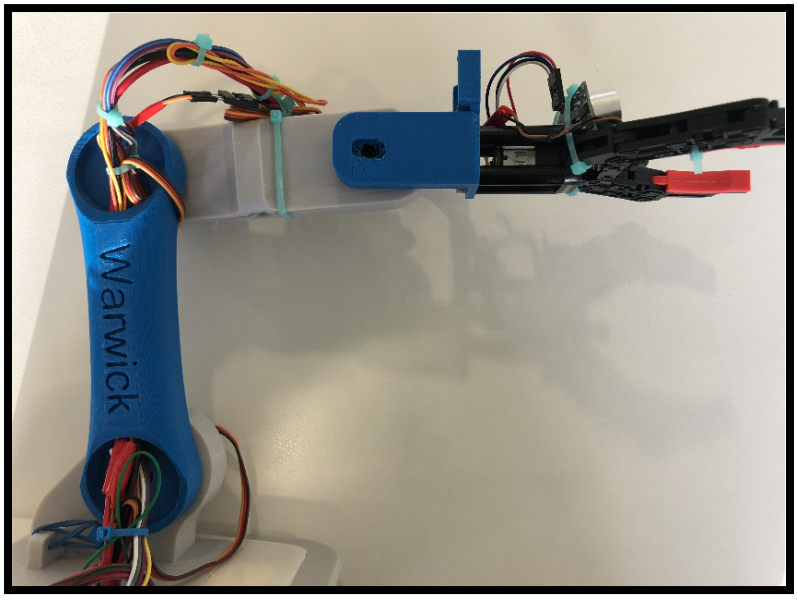
In testing on a half charged battery, our arm displayed a shaking and jittering movement on Servo1 where we believe there was insufficient power going to the servo leading to stalling. Our primary remedy was by recharging the battery fully to ensure we did not have a power shortage causing the servos to stall. We also changed the Arduino code to lower the speed of rotation of the arm and to set the initial position of the servo to the position it is already in. This would prevent a fast high-current servo actuation which could cause the servo to stall and move erratically again.

After further discussion with the rest of the group, we decided to swap the MG996 Servo for a HS-311 servo; essentially the same servo but built to a higher quality. This massively reduced the shake of the device.

After noticing substantial flex in the arm using the first version (see CAD 2 & 4), we redesigned the arm to use one less servo and switched from the SG90 to MG996 for more strength (see CAD 5). This did reduce the rotation of the gripper but was essential for functionality.



Version 1



Version 2

Code

```
const int TRIG_PIN = 7; const int ECHO_PIN = 12;
const int DISTANCE_THRESHOLD = 15;
float duration_us, distance_cm;

const int pwm = 3 ; const int dir = 8 ;
const int forcePin = A4;  const int ledPin = 13;
int forceValue = 0;  uint8_t motorSpeed = 60;
bool bHigh = false;  bool bChangeDir = false;
const int buttonPin = 2; int buttonState = 0;
// servo
#include <Servo.h>
Servo servo_x_axis; Servo servo_y_axis; Servo servo_z_axis; Servo servo_clamp;
int x_axis_degree = servo_x_axis.read();  int y_axis_degree = servo_y_axis.read();
int z_axis_degree = servo_z_axis.read(); int clamp_degree = servo_clamp.read();
#define left_joystick_x A0
#define left_joystick_y A1
#define right_joystick_x A2
#define right_joystick_y A3

void setup()
{
  Serial.begin (9600);
  pinMode(pwm,OUTPUT) ;  pinMode(dir,OUTPUT) ; pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(TRIG_PIN, OUTPUT); pinMode(ECHO_PIN, INPUT);
    servo_z_axis.attach(11); // servo 1
  servo_clamp.attach(5); // servo 2
  servo_x_axis.attach(6);// servo 3
  servo_y_axis.attach(9);// servo 4
}
```

```
void loop()
{
  forceValue = analogRead(forcePin);   buttonState = digitalRead(buttonPin);
  Serial.print(buttonState);
  digitalWrite(TRIG_PIN, HIGH); delayMicroseconds(5); digitalWrite(TRIG_PIN, LOW);
  duration_us = pulseIn(ECHO_PIN, HIGH);distance_cm = 0.017 * duration_us;

  int left_joystick_x_value = analogRead(left_joystick_x);
  int left_joystick_y_value = analogRead(left_joystick_y);
  int right_joystick_x_value = analogRead(right_joystick_x);
  int right_joystick_y_value = analogRead(right_joystick_y);

  if(left_joystick_x_value < 340) y_axis_degree -=2;
  else if(left_joystick_x_value > 680) y_axis_degree +=2;
  if(left_joystick_y_value < 340) clamp_degree -=2;
  else if(left_joystick_y_value > 680) clamp_degree +=2;
  if(right_joystick_x_value < 340) x_axis_degree -=2;
  else if(right_joystick_x_value > 680) x_axis_degree +=2;
  if(right_joystick_y_value < 340) z_axis_degree -=2;
  else if(right_joystick_y_value > 680) z_axis_degree +=2;

  z_axis_degree = min(145, max(15, z_axis_degree));
  x_axis_degree = min(175, max(40, x_axis_degree));
  y_axis_degree = min(180, max(5, y_axis_degree));
  clamp_degree = min(130, max(0, clamp_degree));

  Serial.print("x_axis_degree : ");
  Serial.print(x_axis_degree);
  Serial.print(", y_axis_degree : ");
  Serial.print(y_axis_degree);
  Serial.print(", z_axis_degree 4 : ");
  Serial.print(z_axis_degree);
  Serial.print(", clamp_degree : ");
  Serial.println(clamp_degree);
  Serial.print(", ButtonState : ");
  Serial.println(buttonState);
```

```
servo_clamp.write(clamp_degree);
servo_x_axis.write(x_axis_degree);
servo_y_axis.write(y_axis_degree);
servo_z_axis.write(z_axis_degree);

if ((forceValue < 100) && (buttonState == 1))
{ // There is a Low force applied to force sensor
  if((distance_cm < DISTANCE_THRESHOLD) && (buttonState == 1))
  { //object near
  // Tighten grip
  if(bHigh)
    // we were just Tightening
    bChangeDir = false;
  else
    // we were just looseing and need to change direction
    bChangeDir = true;
  bHigh = true;
}else
{ // There is a high force applied to force sensor
  // loosen grip
  if(bHigh)
    // we were just tightening and need to change direction
    bChangeDir = true;
  else
    // we were just loosening
    bChangeDir = false;
  bHigh = false;
}
```

```
  }else
  { // There is a high force applied to force sensor
    // loosen grip
    if(bHigh)
      // we were just tightening and need to change direction
      bChangeDir = true;
    else
      // we were just loosening
      bChangeDir = false;
    bHigh = false;
  }

  // Stop motor for a brief delay because we need to change directions
  if(bChangeDir)
  {
    analogWrite(pwm,0);
    delay(250); // delay is in milliseconds
  }

  // Set motor direction
  if(bHigh)
  {
    // Loosen the grip and turn on LED
    digitalWrite(ledPin, HIGH);
    digitalWrite(dir,LOW);
  } else
  {
    // Tighten the grip and turn off LED
    digitalWrite(ledPin, LOW);
    digitalWrite(dir,HIGH);
  }

  // Resume motor motion
  analogWrite(pwm,motorSpeed);
}
```