ES2C6 Electromechanical System Design
Group Project: ROBOTIC HAND
Group F

Amirah Olawale-Sanni 2115824
Ellen Nixon Anton 2162918
Isabel Okai 2148160
Jody Nwaelene 2100350
Osi Obomighie 2137813
Sam Packer 2103666

Table of Contents

Introduction

A robotic arm is made up of a range of components with the main components being the Servo's, the gripper, and the power source. As a team we had to combine our skills and research to plan, design and build the robotic hand.

Aim & Objectives

The aim of the project is to design an autonomous grip for a robotic hand using materials provided as well as ordering any extras required. To be able to complete this project, we had to build a circuit on a breadboard, do a substantial amount of programming, 3D printing as well as understanding the best solutions to resolve electronic issues we faced during the process.

Scope

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 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 together and decided 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 in Figure 1 and 2.

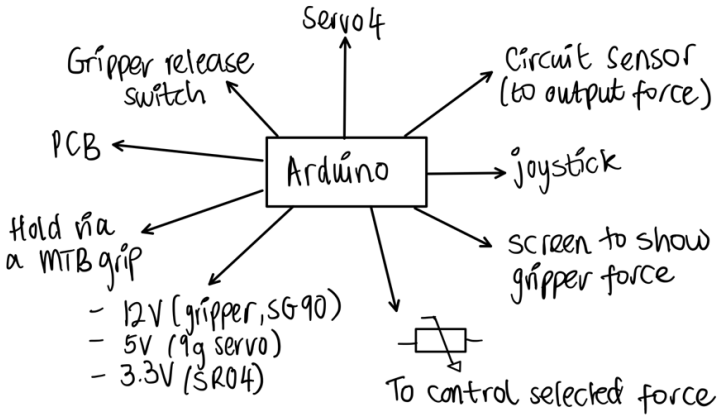


Figure 1

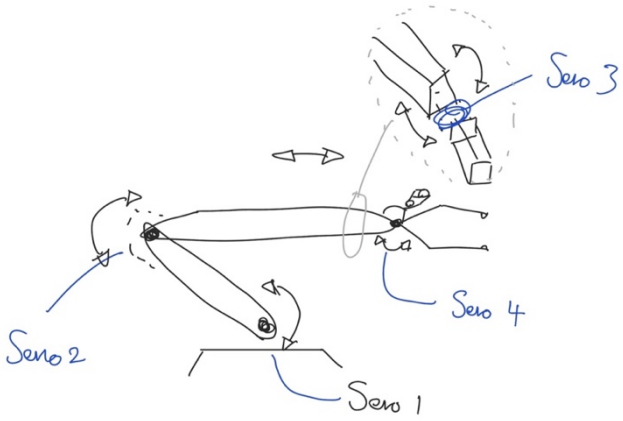


Figure 2

During the process of designing the initial robot arm, the team planned to fit the wiring into ridges (Figure 3) which would mean less printing required of additional components. The robot hand would have been mounted on the square based end allowing it to 'stand' without being held despite the shape also being perfect in terms of the ease of grip. After some consideration this design was not followed through and in fact modified because of the difficulty in permanently attaching the wiring onto ridges without them falling out. The team also decided to take aesthetics into large consideration, as it would be difficult to organise the wires in a way for the overall design to look exemplary as the exposed wires could lead to our design looking quite messy and not up to standard. The wires also have the potential of getting snagged in the motion of the sections, resulting in failure from a disconnected cable.

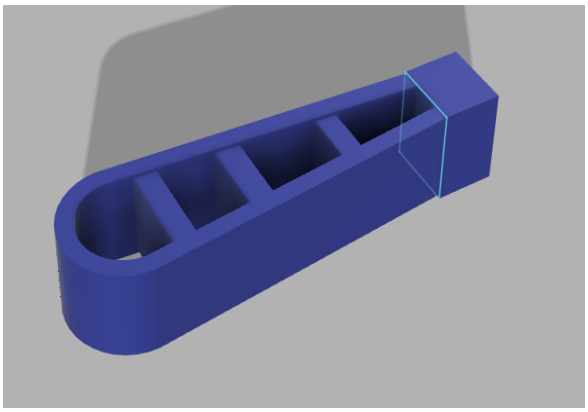


Figure 3

Literature Review

Most robots in the world are designed for heavy, repetitive manufacturing work. They handle difficult, dangerous, or 'boring' tasks. 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 and a computer controls the robot by rotating individual servo motors connected to each joint (Harris and Pollette, 2002).

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. Given that 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 not to include rotational motion about the origin of the arm. We did investigate using 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.

Project Management – 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. By only powering our electronics, when necessary, we were able to minimise risks to everyone's safety.

Materials Received

- Makeblock Robot Gripper with 12V DC Motor x1
- Makeblock Me Dual DC Motor Driver (possibly with soldered line socket) x1
- Arduino Uno board R3 (or comparable) x1
- USB2 3m A male to B male cable x1
- Interlink Electronics 0.2" diameter force sensing resistors x2
- 360 tie point prototyping breadboard x1
- 10K resistors x3
- OP177GPZ Ultra-Precision OP Amps x2
- HC-SR04 Ultrasonic Distance sensor x1
- Breadboard jumper wires (various colours, sizes)
- 12V, 1A, 12W, Plug In Power Supply
- Adafruit Adapter, 2.1 mm DC Jack - 2 Position Terminal Block
- ELEGOO UNO R3 Project Super Starter Kit

We also received a £200 discretionary budget, excluding the costs of the components. This left us with £100 to purchase additional components which we used to buy the components as shown in Table 1.

Component Description	Manufacturer	Manufacturer Part Number	Supplier	Supplier Part Number	Quantity	Unit Cost (£)	Total (£)
Servo 9g	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							57.34

Table 1

Design development & CAD

Initially, we planned to insert wiring into ridges with the robot hand mounted onto a square based end (as seen in Figure 5). This shape would allow for an easy grip. However, we decided to modify this design due to it being difficult to permanently attach the wiring onto the ridges. Furthermore, aesthetically, exposed loose wires would look messy. 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 investigate the use of a servo control module from Adafruit such as the 16-servo board (Figure 4) 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.

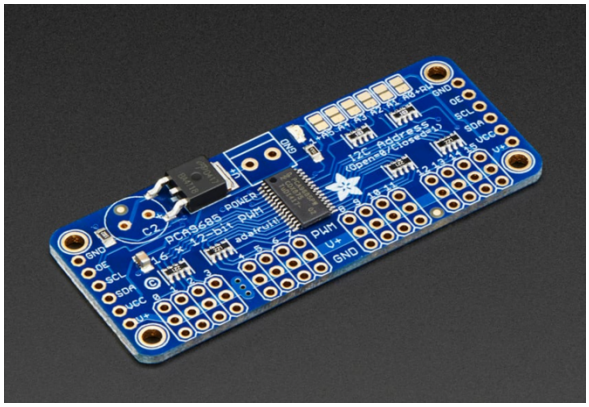


Figure 4

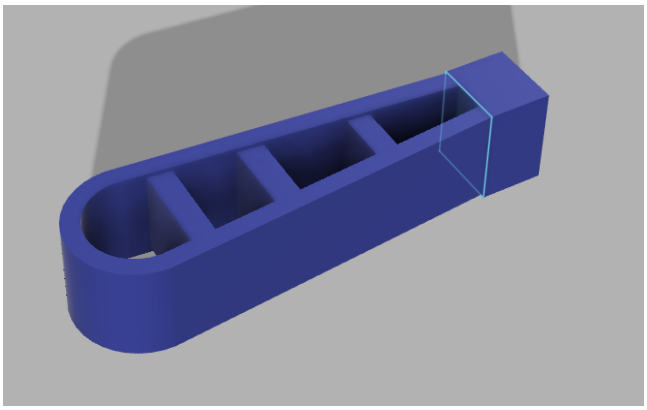


Figure 5

The components in Figure 6 are ultrasonic sensors which emit ultrasonic sound waves and converts it into an electrical signal. The component in white was the initial 3D print but then the team realised it was too small to fit the robotic hand, so the dimensions were adjusted, and we reprinted it in blue, to give an ultrasonic sensor that fits the design. This was an early issue we faced which meant that we had to re-evaluate the design but were not pressed for time as it occurred at an early stage. Figure 7 shows the CAD version of this component.

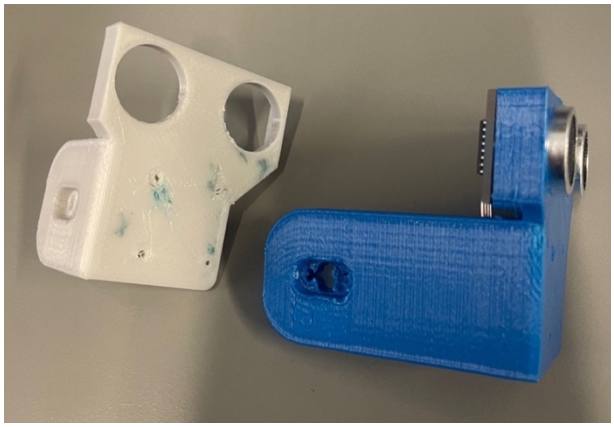


Figure 6

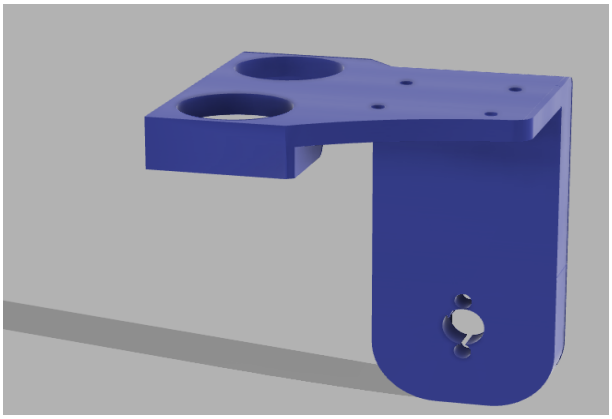


Figure 7

We decided to increase the complexity of the design by splitting the robot arm into parts. This decision also made 3D printing more straightforward. The server was attached on a rectangular gap, which is shown below in Figure 8, then the robotic arm (Figure 9) was attached onto the other side, where the circular hole is, in Figure 8. Figure 10 shows the engineering drawings for Figure 8.

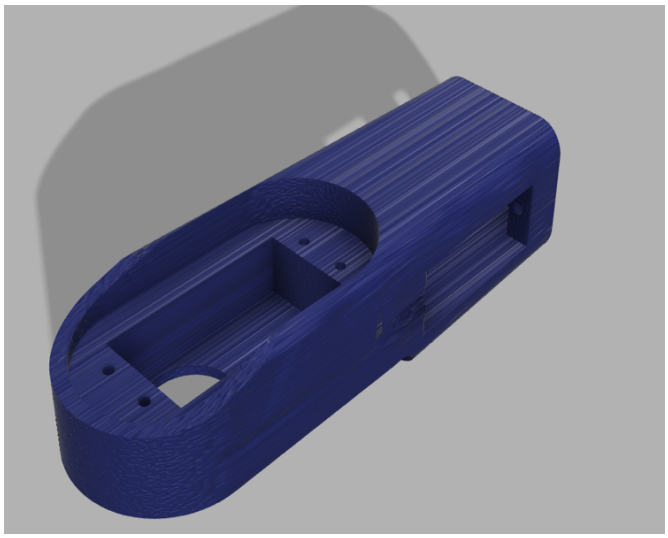


Figure 8

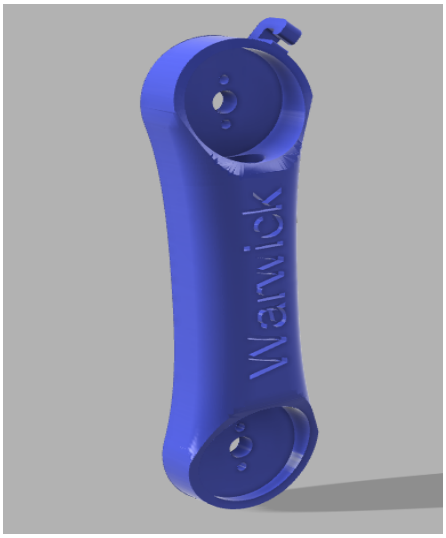


Figure 9

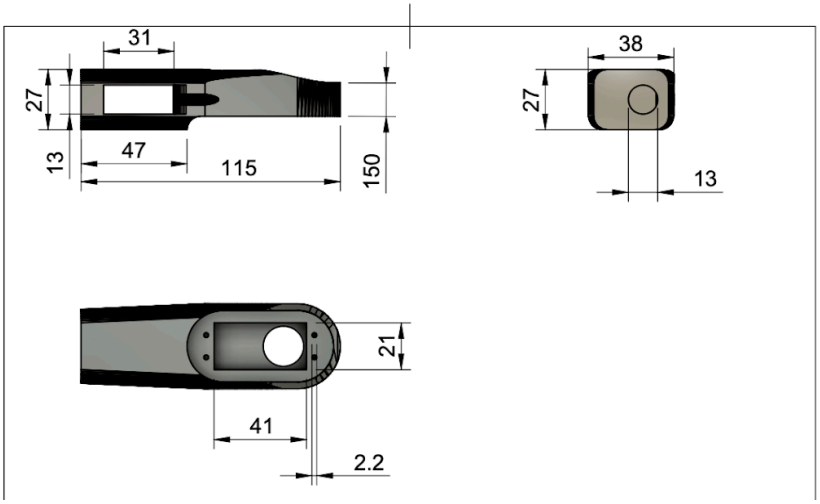


Figure 10

The gripper base is shown in Figure 11. Figures 12 and 13 showcase the drawings for the section of the robotic arm seen in Figure 9.

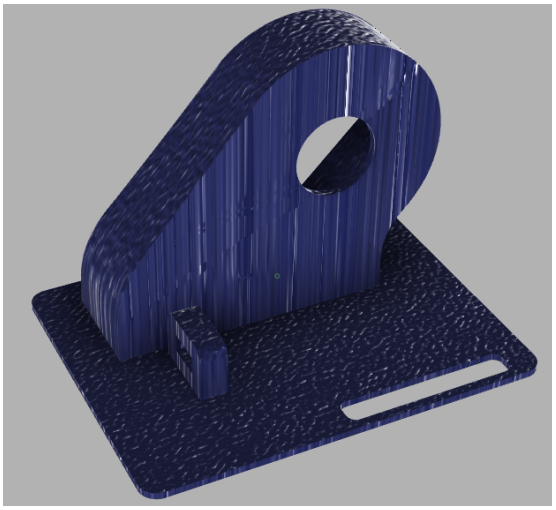


Figure 11

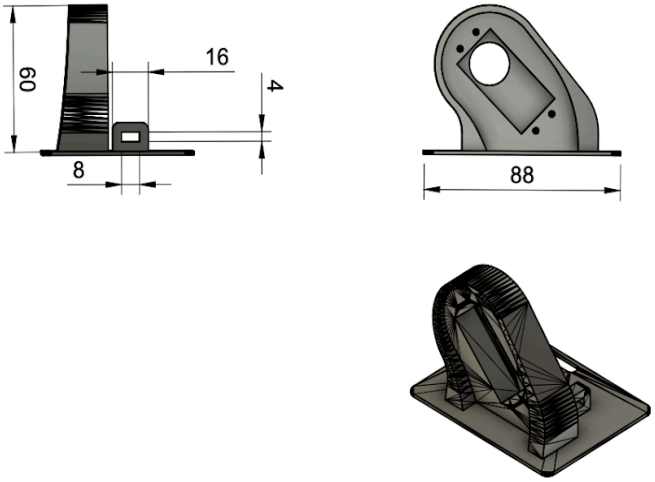


Figure 12

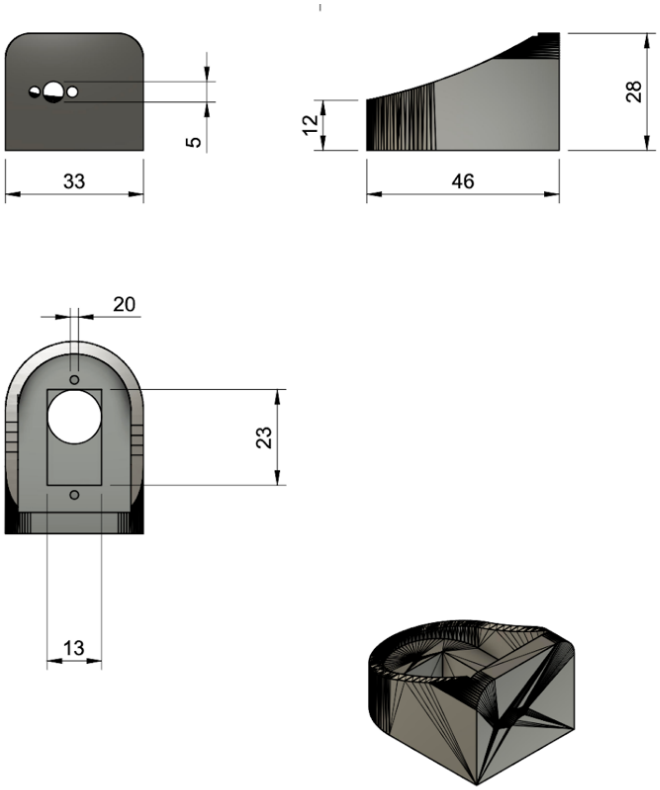


Figure 13

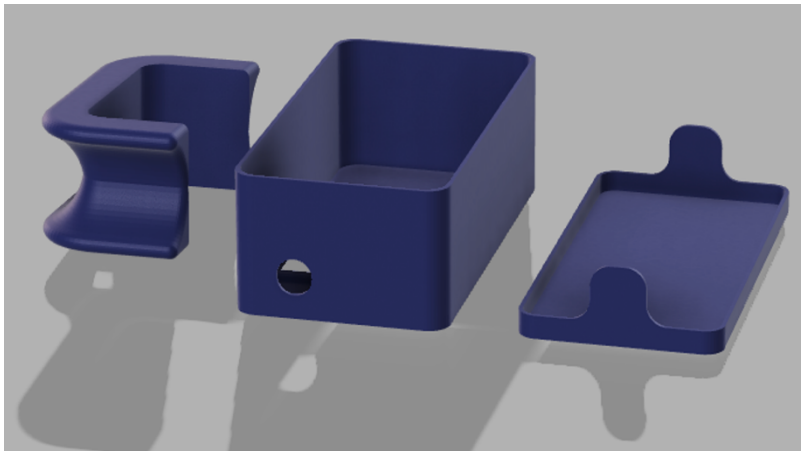
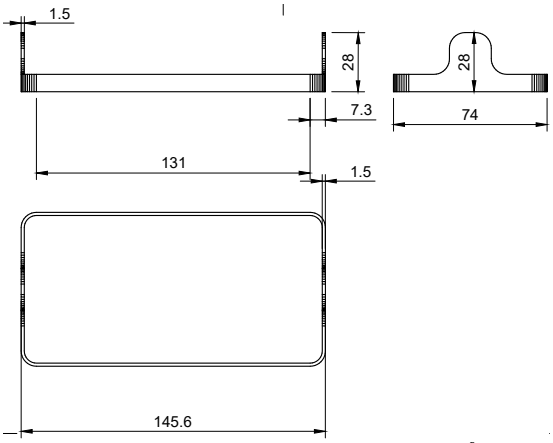


Figure 14



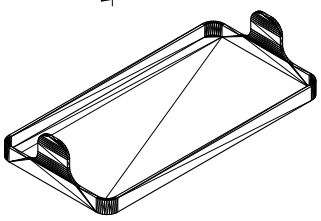
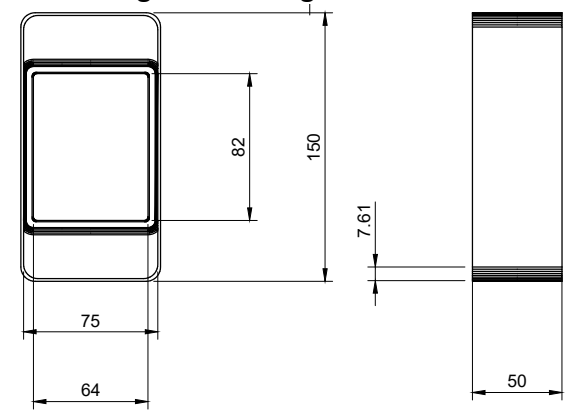
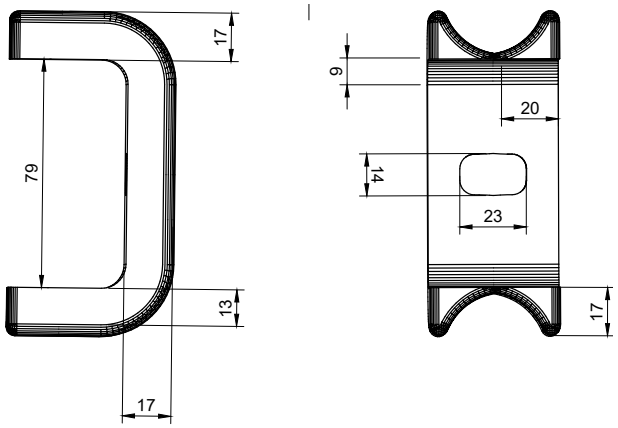


Figure 15

Figure 14 shows the casing made for the base of the Robotic Arm with Figure 15, 16 and 17 showcasing the drawings made.





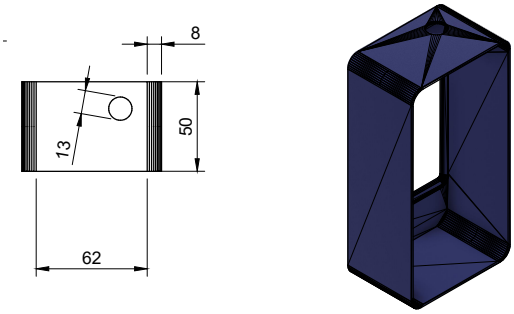


Figure 16

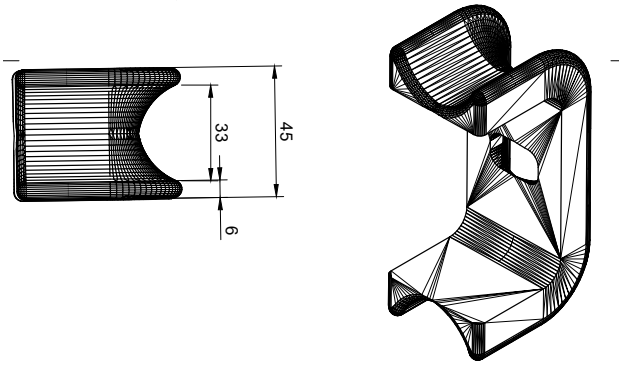


Figure 17

Final Design

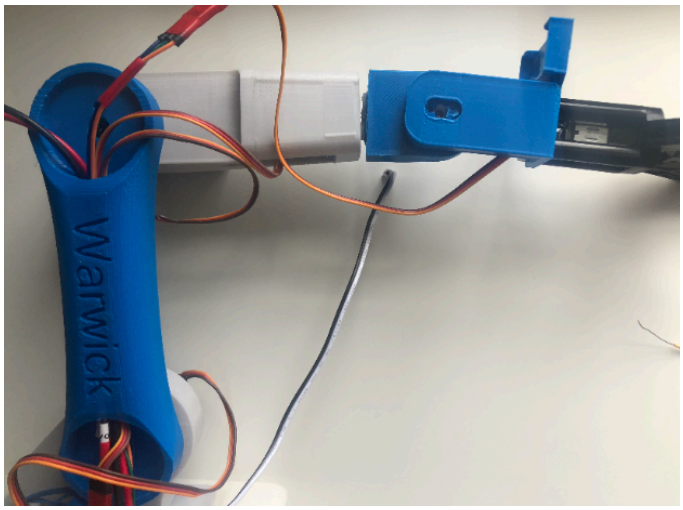


Figure 18

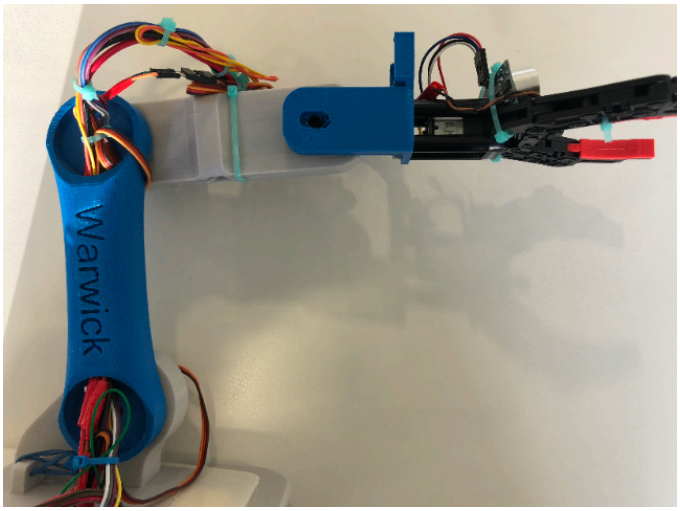


Figure 19

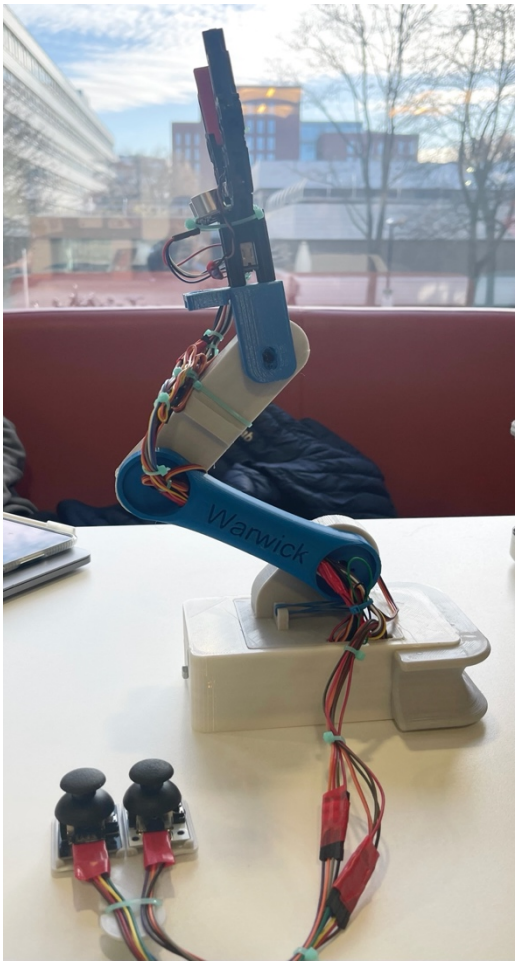


Figure 20

Figure 18 shows our initial final design which had to be changed after testing as it was not strong enough to hold on its own. Adjustments were then made resulting in a design that is sturdier, shown in Figure 19 and Figure 20.

Circuit Diagram

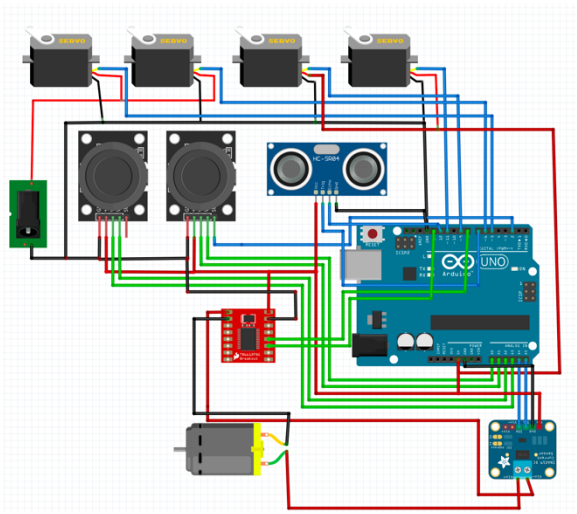


Figure 21

There are two main power delivery parallel circuits, a 12V and 5V in the schematic shown in Figure 21. The circuit will be powered by a 12V LiPo. Two joysticks are used to control the arm via built in joysticks.

Arduino Coding

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, 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
```

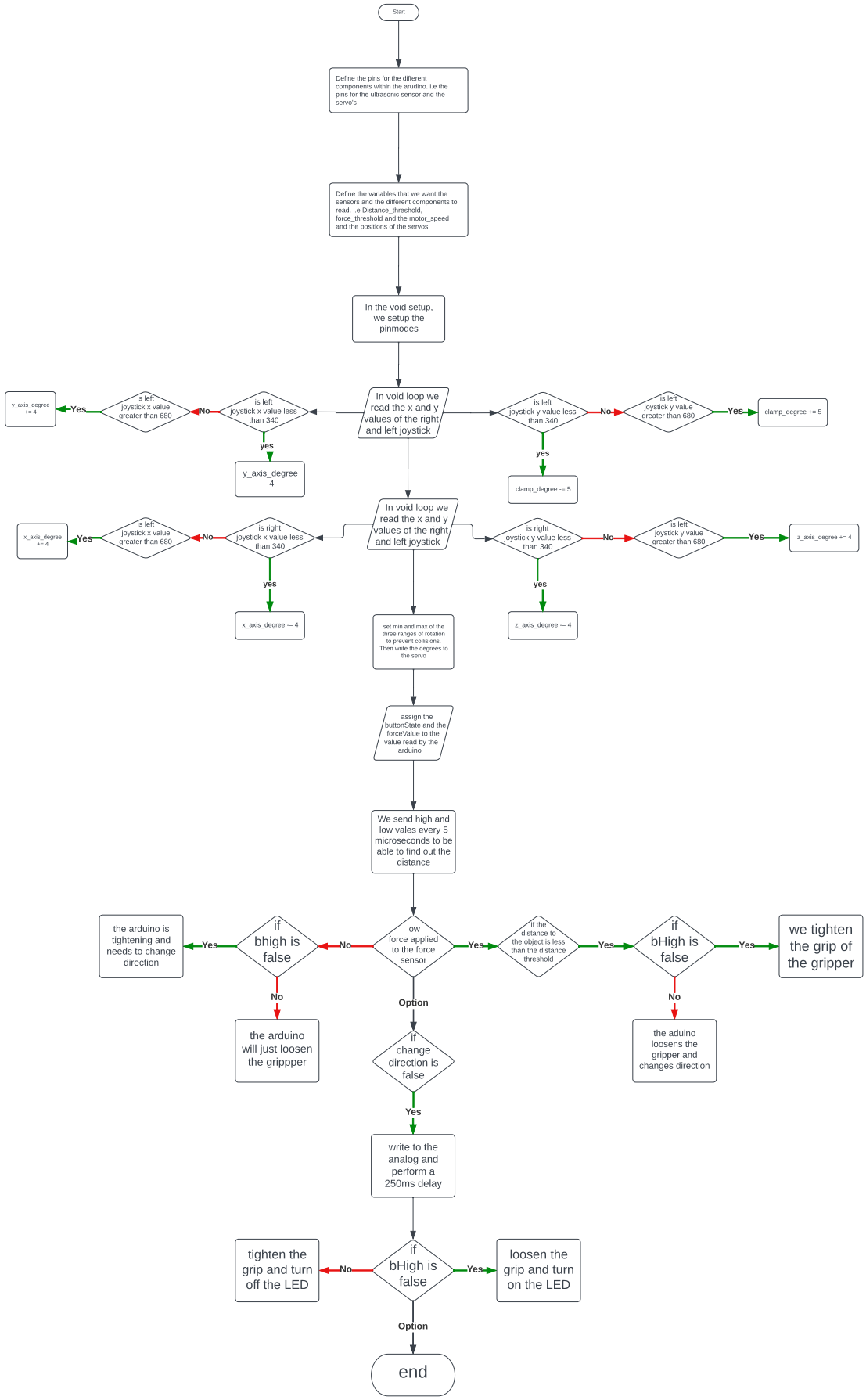
This section of code can be seen in our project code from line 33 onwards.
This means that the servo motor holds the position it's at even when the joystick returns to the origin which aids stability of operation (less noise input from human tremble) and makes the arm operate in a smoother fashion with fewer sudden ‘jerk’ movements which draw a lot of power.

Below is the Arduino code used to run the robotic arm:

```
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);
}
```

The flowchart below explains how the code works for the Robotic Hand.



Control System

When deciding about the design of our robotic hand and its functioning, we had to consider 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 in Figure 22 and Figure 23.

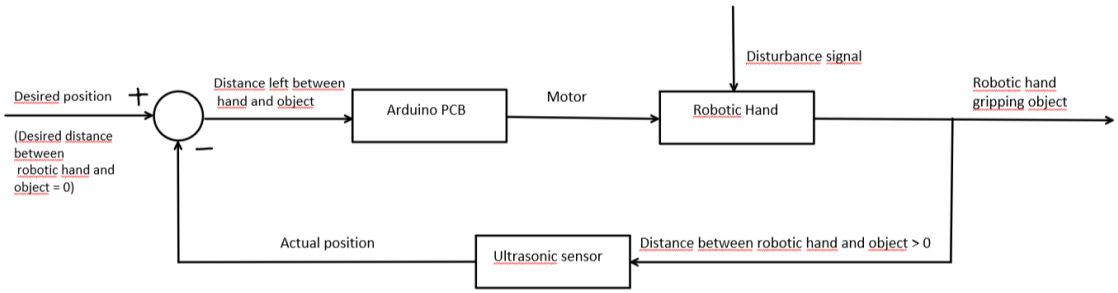


Figure 22

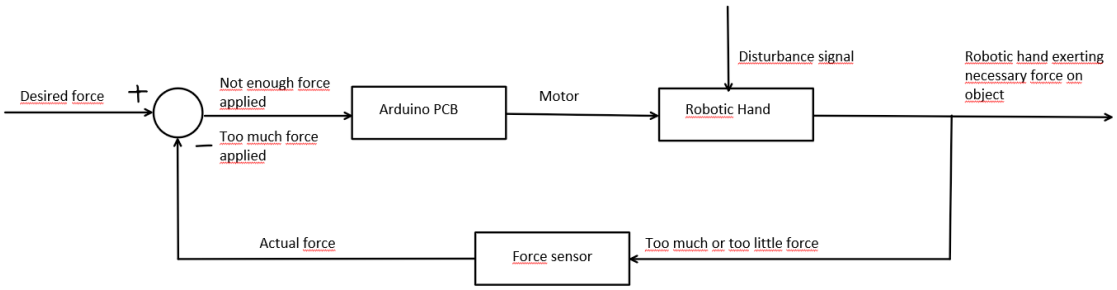


Figure 23

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 (Figure 24) shows that a linear relationship in the form y = mx + c exists for this.

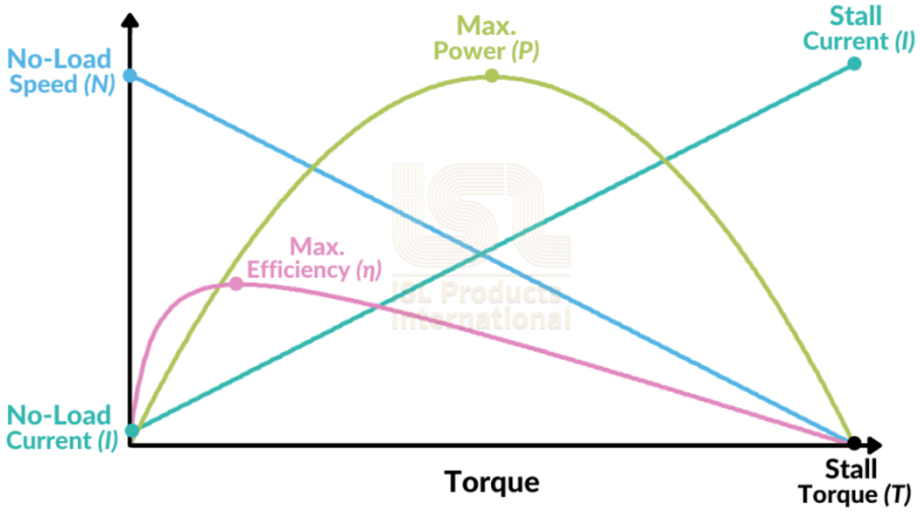
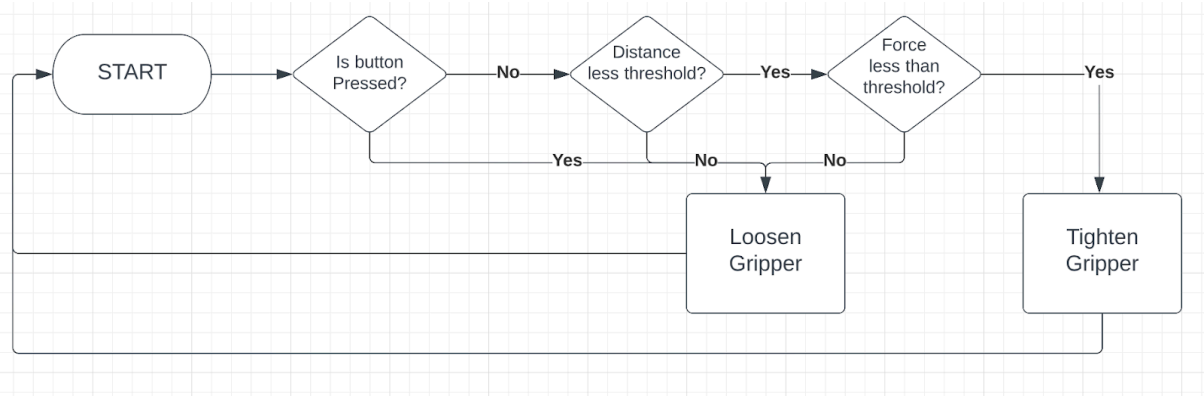


Figure 24

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



Results

Test	***Outcome***	***Notes***
Do servo 1 and 2 move in accordance with the joystick input?	Yes	High power is needed for servo 1 and 2 SG90
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
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.

Conclusion

The project went well overall, and we were able to reach our main goal of creating a robotic hand that works well. However, we did incur a few issues along the way with the code aspect as well as the deign itself which led to alterations. Improvements for next time could include further research to widen our knowledge on both the technical and non-technical aspects and to see if there are other components that would have improved the quality of the design.

References

Harris, T. and Pollette, C. (2002). *How Robots Work*. [online] HowStuffWorks. Available at: https://science.howstuffworks.com/robot2.htm.