Department of Engineering

Laboratory Assignment: Creating a system able to capture EMG signals and correlate them to gripping patterns

Forename: Ema

Surname: Shek

Year of submission: 2022

Module: 601066_A22_T22

Degree Programme: Biomedical Engineering

For the attention of: Dr Anthony Bateson

Abstract

In this report, the aim was to create a program able to capture EMG signals, classify them and pair them with 4 gripping patterns called pinch, point, spherical, and power. This was done by creating three separate Arduino programs first. The first one was a program to obtain the gripping patterns. This was done by using a prosthetic hand connected to an Arduino Uno and 5 servo motors, one for each finger. The captured EMG waveforms were obtained by using a myoelectric sensor and the Arduino Uno circuit, without the prosthetic hand. This code found both raw and processed EMG signals and is shown there in the serial plotter of the Arduino software. The EMG classify code was very similar to the EMG waveforms code, except this time the if statement with the condition was applied. Finally, by combining the EMG classify and the gripping patterns code, the final code was obtained and able to connect each gripping pattern to a specific range of signals shown in the time domain. The muscle chosen for this experiment was the flexor carpi radialis due to it being responsible for wrist bending and connection to the palm. A series of further improvements to the mechanism were listed at the end of the report, the major being using a higher amount of electrodes.

Contents

Table of Figures

1. Introduction

EMG signals can be used for a series of different reasons in both biomedical and clinical fields. Even though this technology is new and with its limitations, a lot of progress has been made through the years to obtain accurate and reliable signals describing neuromuscular activities. When acquiring these signals, which are usually a function of time, from muscles, a series of factors have to be considered, such as classification. In this report, how to obtain these EMG signals and use them is described.

2. Grip pattern program

This report has been divided into sections, the first one is the *grip pattern program*. To be able to understand this section, it is important to learn how to set up the lab work before starting to code it. The prosthetic hand, which is a 3D-printed hand with a circuit including servo motors controlled with an Arduino Uno board to form the gripping patterns, must be coded using the Arduino platform. The task requires that it is coded so that it makes the following gripping patterns: pinch, point, spherical, and power gripping patterns, as shown in figure 1.

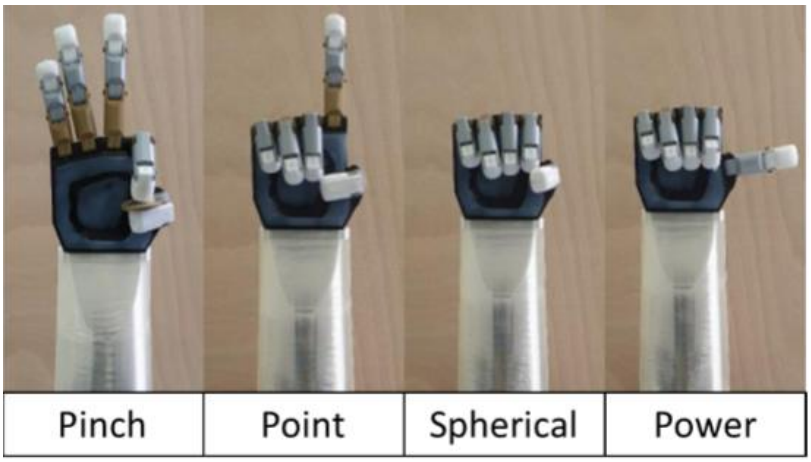


Figure 1 shows the grip pattern that the prosthetic hand has to follow

Each finger is connected to a specific servo motor, this is very important to understand which Arduino Pin moves a specific motor and therefore finger, as shown in figure 2.
The pinky is connected to servo 1, the ring finger to servo 2, the middle finger to servo 3, the index finger to servo 4, and the thumb to servo 5.

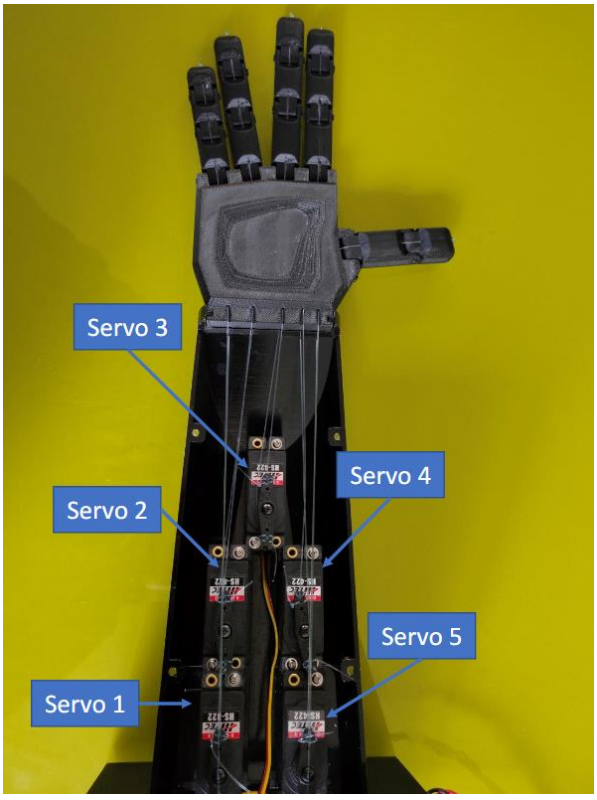
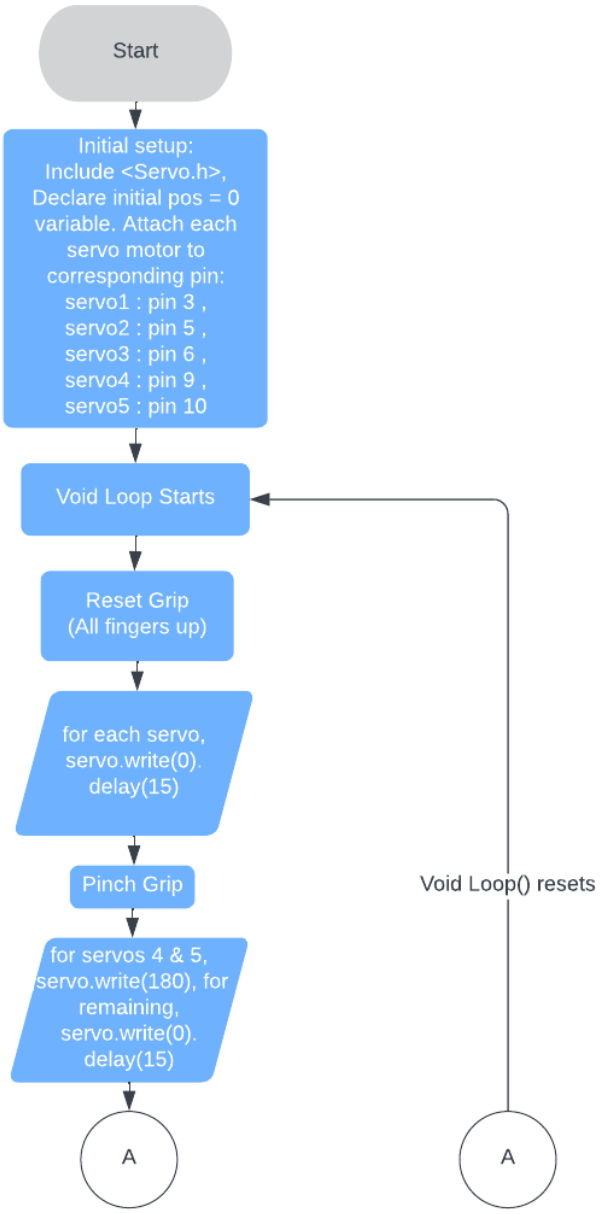


Figure 2 shows how each finger is connected to a specific servo motor

Before developing the code for the mechanism, a flowchart that describes in detail the function of the program and how data flows through it, to aid in development and to help understand the task at hand, as shown in figure 3. The flow chart shows each step of the setup portion of the program, such as each variable that must be declared, libraries that are needed for some of the program's functionality, and which pins correspond to which servo motors. It then shows the void loop() function that holds the main body of the code. The function starts by resetting the current hand position to the default (all fingers up, .write(0)), and one by one changes to different gripping positions by adjusting the .write() for each motor, the gripping positions are, default, pinch, point, spherical, power.

The code for each position is held within a for loop to adjust the .write() command by incrementing the value by one each time instead of instantly to 180 or 0, creating a smoother motion, there is also a short delay of 15 milliseconds before moving onto the next gripping pattern, giving enough time for each to be observed. Once this code has run, the loop will begin again, resetting the hand and looping through each of the gripping positions till the program is stopped.



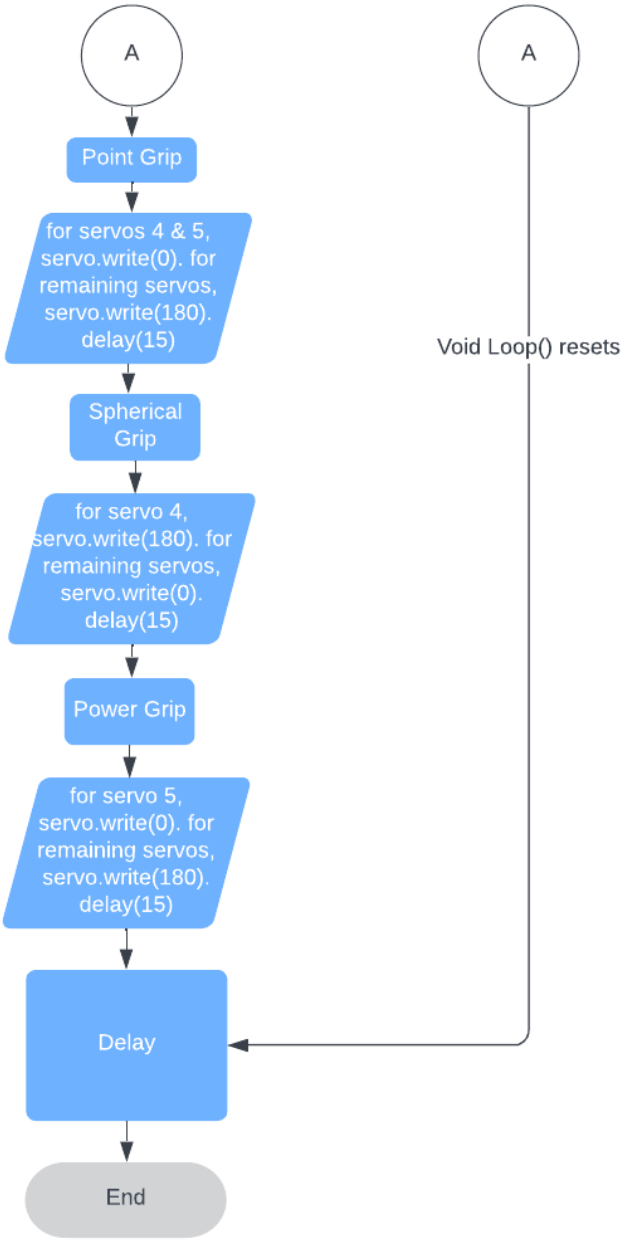


Figure 3 shows the flowchart showing the tasks to be completed to be able to code successfully the arm

This pattern shown in figure 3 was then followed in the code. The code has a few main functions. First, as explained in the flowchart in figure 3, all the variables have to be set before starting the loop. These variables are the library used, which in this case is <servo.h>, then all the servos used have to be specified, which are servo myservo 1 through 5 and the pins they are attached to, as shown in figure 4. It was also important to set up a variable of type int, a data type able to store integer numbers (Arduino, 2023), for pos=0, this just means the integers that determine the position of the fingers will start from 0.

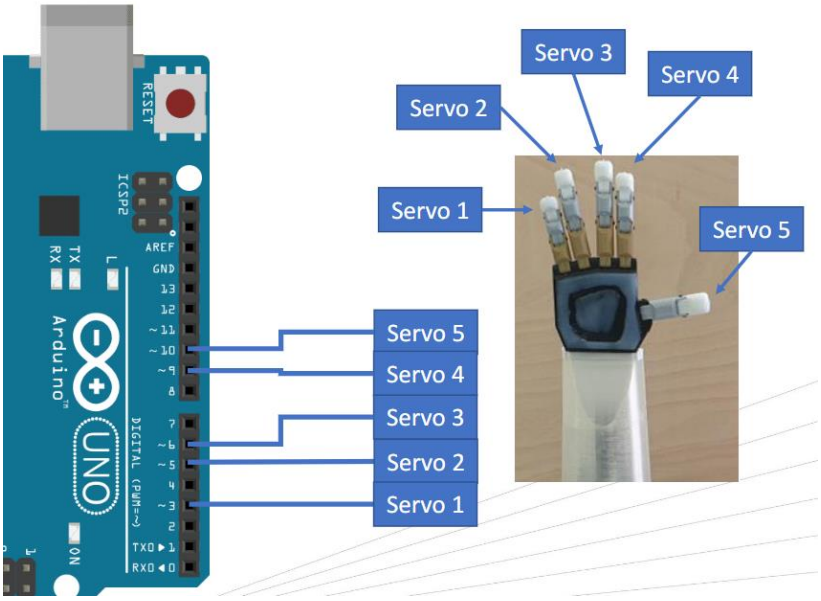


Figure 4 shows how the board was connected to the servos and which pins have been used.

Then, it is important to notice that there are two *for*-loop statements that need to be included in the code.

To code the fingers at their resting position (finger up), the *for* loop Is used: `for (pos = 0; pos <= 180; pos += 1)`, this means that the servo moves from zero to 180 degrees in steps of 1 degree. To move the fingers 180 degrees from their resting position, the same *for* loop is used, except it goes: `for (pos = 180; pos >= 0; pos -= 1)`, which means movements from 180 to zero at steps of –1 degree (-1 degrees shows that the finger is going down). The other function used was my '*myservo.write1.(pos)*', which defines which servo moves and what position it reaches between 0 and 180 degrees. By using these *for* statements and the *myservo.write* function, each movement needed for the hand was coded, starting from the reset pattern, followed by pinch, point, spherical, and power movement. Each movement was separated by a delay of 0.015 seconds, this was done to be able to see each pattern before the next started.

To improve code reusability and ease manipulation of each of the different gripping positions, each of the movements have been placed into separate functions as shown in the code below. This allows each movement to be called by just writing the function name, eliminating the need for the code to be rewritten each time, for example, to change the grip pattern to pinch, point, pinch, the code is just pinch(); point(); pinch();.

```
#include <Servo.h>
Servo myservo1;
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo5;

int pos = 0;
void setup(){
 myservo1.attach(3);
```

```
 myservo2.attach(5);
 myservo3.attach(6);
 myservo4.attach(9);
 myservo5.attach(10);
}

void loop(){
  //RESET VALUE
  for (pos = 0; pos <= 180; pos += 1) {
    myservo1.write(0);
    myservo2.write(0);
    myservo3.write(0);
    myservo4.write(0);
    myservo5.write(0);
    delay(15);
  }

  //PINCH MOVEMENT
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo4.write(180);
    myservo5.write(180);
  }
  for (pos = 0; pos <= 180; pos += 1) {
    myservo1.write(0);
    myservo2.write(0);
    myservo3.write(0);
    delay(15);
  }

  //POINT MOVEMENT
  for (pos = 0; pos <= 180; pos += 1) {
    myservo4.write(0);
    myservo5.write(0);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo1.write(180);
    myservo2.write(180);
    myservo3.write(180);
    delay(15);
  }

  //SPERICHAL MOVEMENT
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo4.write(180);
  }
  for (pos = 0; pos <= 180; pos += 1) {
    myservo1.write(0);
    myservo2.write(0);
```

```
59      myservo3.write(0);
60      myservo5.write(0);
61      delay(15);
62    }
63
64    //POWER MOVEMENT
65    for (pos = 0; pos <= 180; pos += 1) {
66      myservo5.write(0);
67    }
68    for (pos = 180; pos >= 0; pos -= 1) {
69      myservo1.write(180);
70      myservo2.write(180);
71      myservo3.write(180);
72      myservo4.write(180);
73      delay(15);
74    }
75  }
76
```

3. EMG plot Program

In this laboratory, a myoelectric sensor was attached through three sticky electrodes to the skin surface, as shown in figure 5. The only difference from the previous lab was that this time, instead of focusing on finding gripping patterns, the focus will be on reading the Raw and Processed EMG signals provided through the sensor. The Myoelectric sensor can output the EMG envelope used to control the servo and therefore processed EMG Signals are received from AnalogReadA0 after the raw signals are rectified through the sensor.

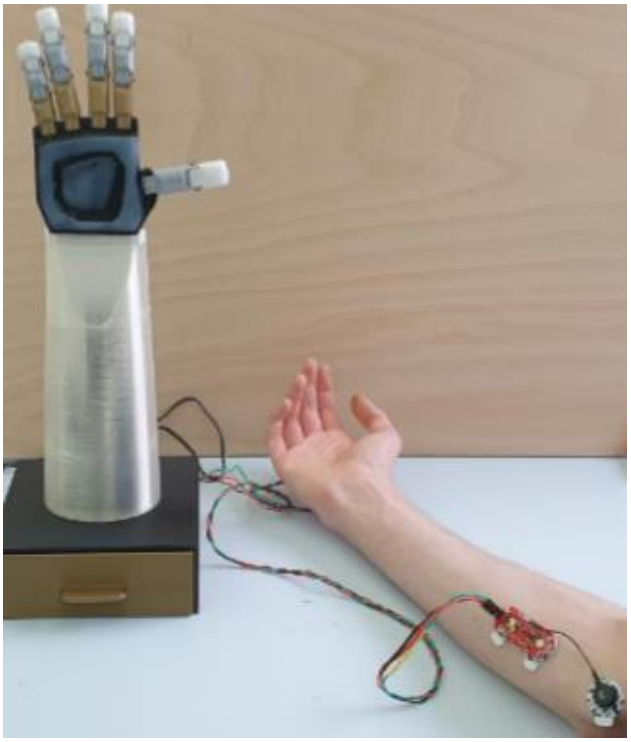


Figure 5 shows how the three electrodes are attached to the volunteer arm

For this lab, a flowchart was made as well to better understand the tasks to be completed before trying to develop the code for the mechanism. The first part is to define the variables. In this case, as shown in figure 6, the sensors were attached to pins A0 and A1. The variables 'PROCESSED' and 'RAW' where given the values of these pins using a built-in function called AnalogRead() later in the program after their initial declaration. This function reads the values received in the Analog pins and shows them in a numerical range between 0 and 1023. The processed EMG is obtained through AnalogReadA0 after the raw signal is rectified through the sensor used. Raw signals are obtained from AnalogReadA1.

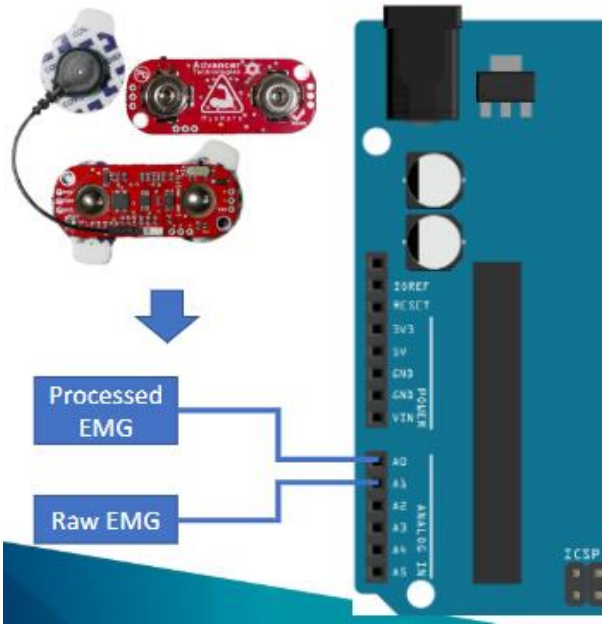


Figure 6 shows how the Myolectri sensor is attached to the Arduino board

Once variables are set up, the <servo.h> library is included, the serial monitor is started with a baud rate of 9600. The values of the variables 'PROCESSED' and 'RAW' will be printed to the serial monitor in the format of: processed_value "," raw_value. This will display each value in a CSV (Comma Separated Values) format, every time the loop is run, this is not only so the values can be plotted using the Arduino built-in serial plotter but also so an additional program can be used to read in a CSV file of the outputs and graph the results if necessary. The Arduino serial plotter will graph the values for Processed and Raw EMG signals and any differences can be seen, after each reading, a small delay has been added before ending the code. The main body of the code is run within the void loop() function and so will be repeated till the program is manually terminated, or a given condition tells it to exit the loop as with most Arduino programs.

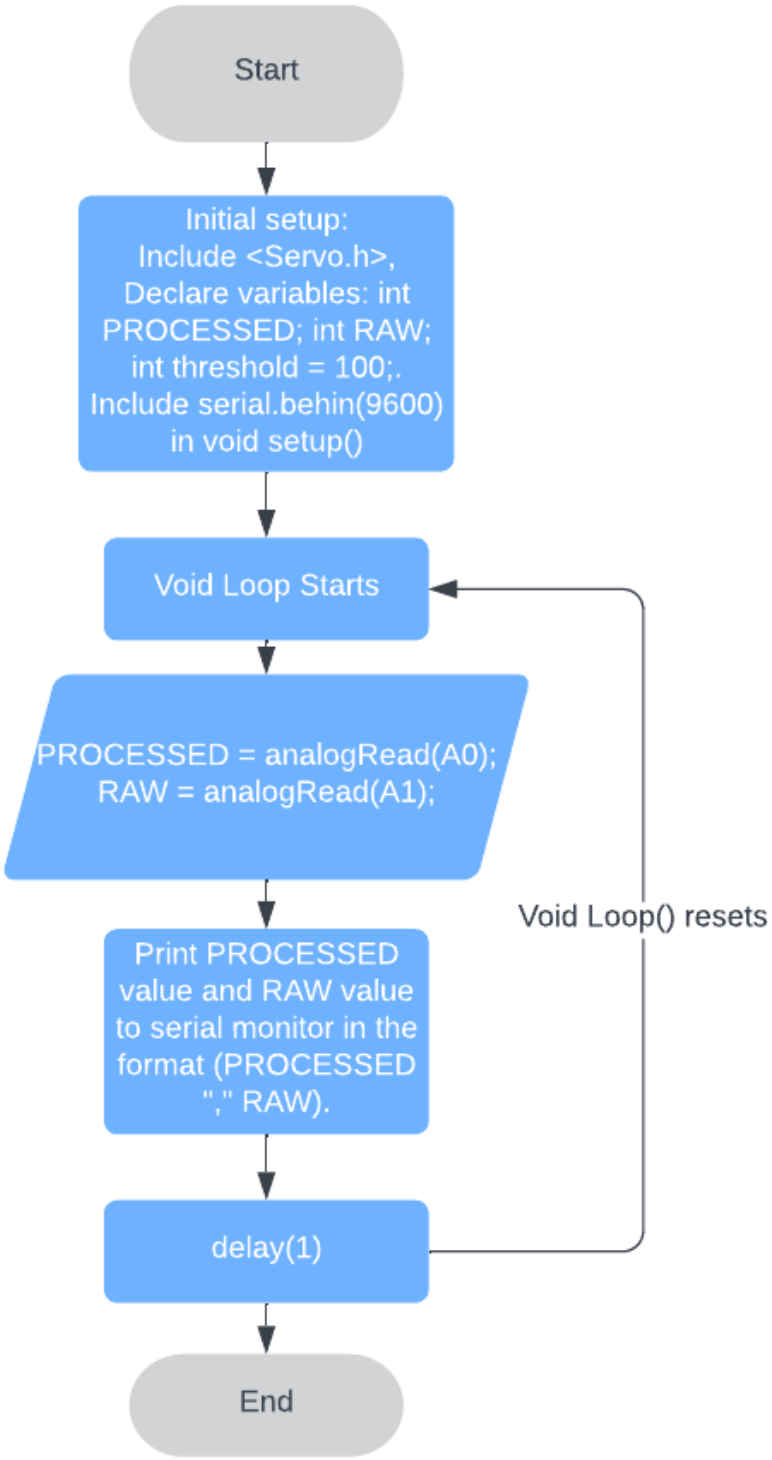


Figure 7 shows the flowchart used to plan the EMG plot program

The code for this laboratory has been shown below. First of all, the <servo.h>> library has been added, this library is helpful as it helps to avoid having to code the sensors too for the role they have been given. Values for Processed and Raw signals are both integers and as such have been defined, this helps to store the integers values obtained. *Int threshold* is an Arduino variable that tells the Arduino when to find an actual EMG signal, in fact, increasing the threshold decreases the sensitivity and vice versa. In the void setup, *Serial.begin(9600)* has been added, this value is the baud rate and represents

how fast the data has to be sent (Monk, 2021). In void loop, which is the part of the code that will run in a loop (the rest of the code will be read only once), AnalogReadA0 and AnalogReadA1 have been defined, they are telling the Arduino to collect the data provided by AnalogReadA1, rectify it through the sensor and output the results as Processed EMG signals from AnalogReadA0. Once that is done, each time after a delay of 0.001 seconds, the serial plotter will show values for both Processed and Raw EMG signals.

```
#include <Servo.h>
int PROCESSED;
int RAW; //RAW EMG
int threshold = 100;

void setup() {
 Serial.begin(9600);
}

void loop() {
 PROCESSED = analogRead(A0);
 RAW = analogRead(A1);
 Serial.println(PROCESSED);
 Serial.print(",");
 Serial.println(RAW);

 delay(1);
}
```

3.1 EMG plot program graphs

Screenshots of the serial plotter during the EMG plot program have been taken. As can be seen in figure 8, at rest where EMG signals are lower than the threshold, there are no activities and both Raw and Processed EMG are very linear as there are no events. The Raw EMG signal has been shown in blue and the red one is the processed one.

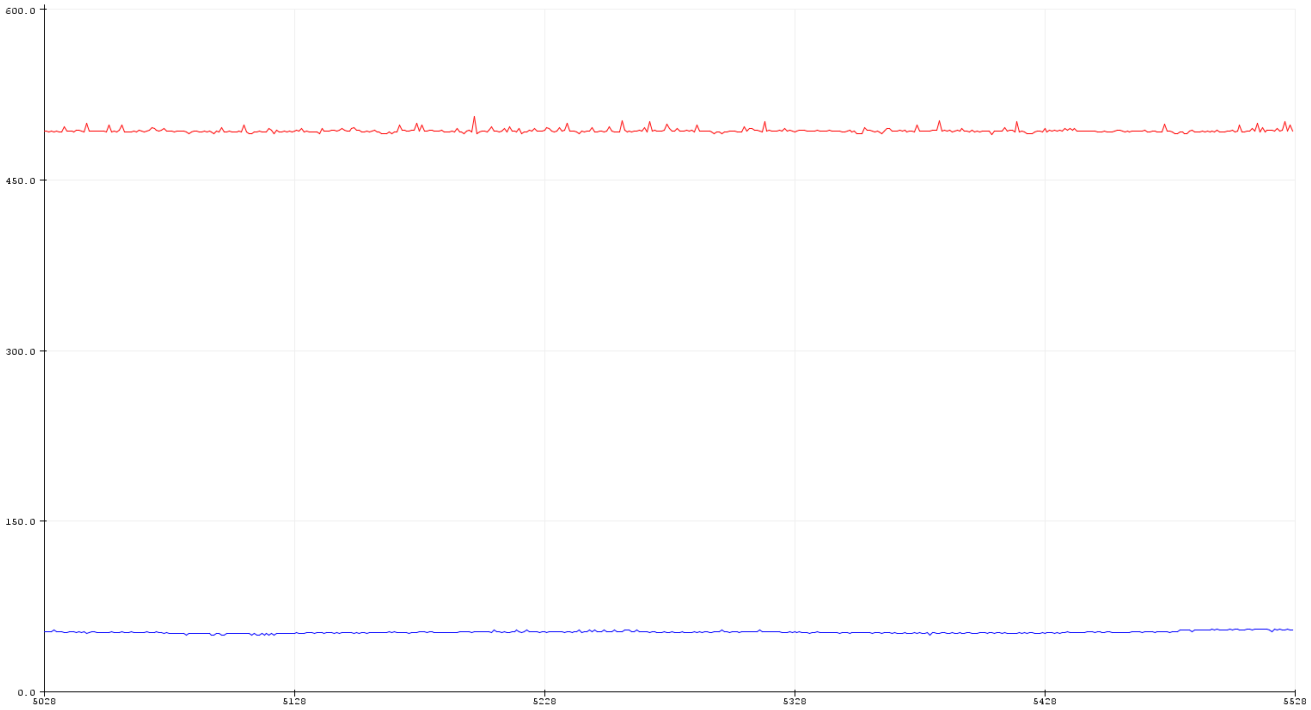


Figure 8 shows the EMG raw and processed signals when the person is at rest

Figure 9 has been shown to illustrate what happens when an event occurs. It is important to notice that all the graphs are showing analog values and not voltage readings. When obtaining the values for figure 9, instead of moving the hand with a specific movement, random patterns were followed to be able to see as much muscle activity and obtain a big range of values.

It can be noticed that whenever the event reaches a peak, highlighted by a black circle, the same peak is shown in the processed signals. These peaks are values above the baseline result of the EMG signal increasing due to movement. The threshold used to obtain these signals is 100.

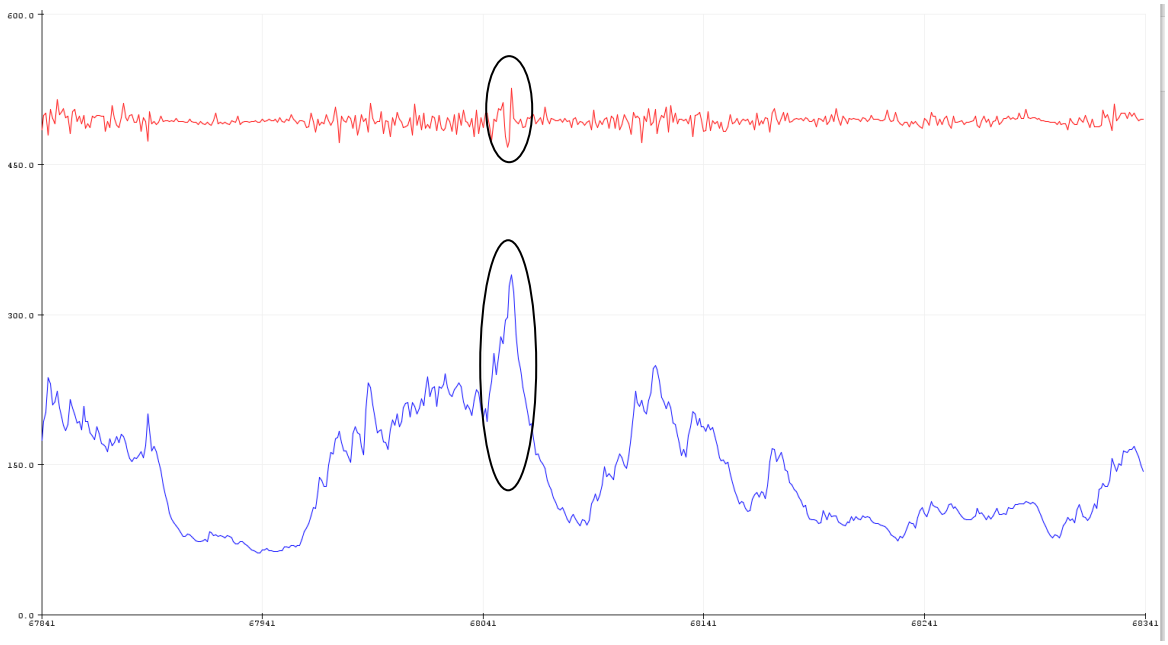


Figure 9 shows an EMG graph with both Raw and processed signals when the person is not a rest and moves the hand randomly.

Another graph of the results shown through the serial plotter has been shown in figure 10 to compare it with figure 9, this time however a sequence of movements has been repeated several times, these movements were opening and closing the fist. It can be noticed when comparing both graphs to each other that most EMG raw signals have been captured between analog 100 and 350, where 350 is the peak. This corresponds to 0.45V and 1.71V received when analysing the signals. These values were obtained by timing the Analog value by 5 and then dividing it by 1023, as 5V is the voltage supplied by the board and 1023 is the maximum numerical analog value. The peak is seen when the volunteer closes their fist as tight as they could while the lowest values are seen when the fist is open, and the hand is at rest.
When analysing the values obtained for processed EMG, it can be noticed that in both figure 9 and figure 10, these values have a range between 450 and 500, in voltage, this is a range between 1.95V and 2.44V.

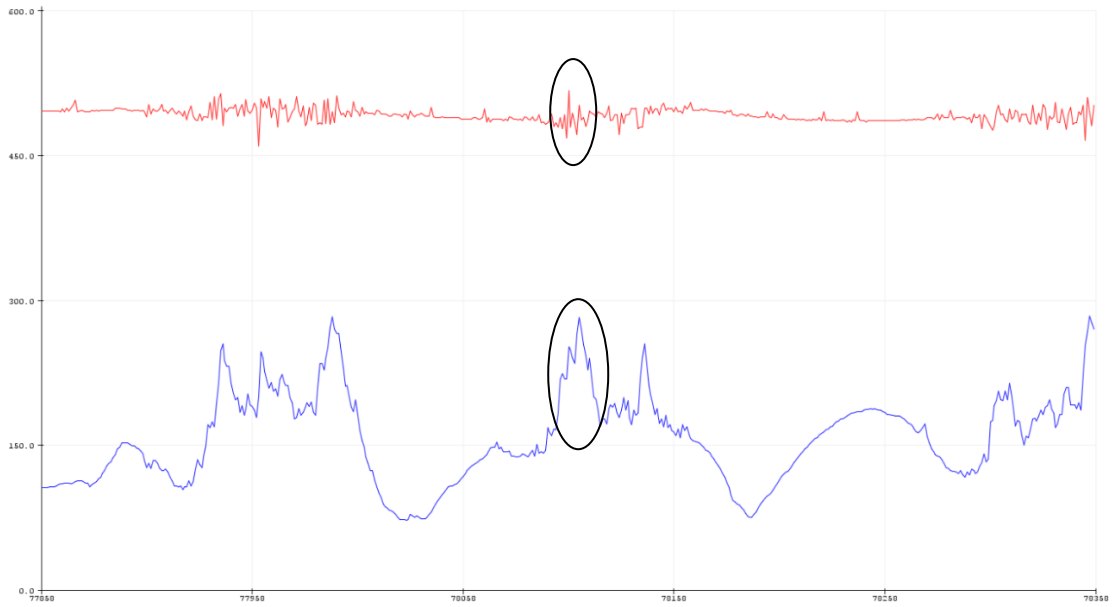


Figure 10 shows values for EMG Raw and processed when the hand is closed and opened

By comparing figures 8, 9, and 10 It can be said that the code has been successfully executed, as at rest, the serial plotter plot does not read any event, just the background noises, while when an event happens, both Raw and Processed EMG signals move from the baseline.

4. EMG classify program

This section shows the same setup as for section 3 except this time an *If* statement is introduced in the loop. Instead of reading any value from the baseline to any peak, this if statement defines that only values above the threshold are plotted so that only events occurring are shown in the graph. After the variables have been set, a threshold must be set as well. This threshold must be the smallest value for which an event can be in order to be considered and plotted. Once the threshold has been chosen, both AnalogRead() values have to be set. This is where the Arduino makes a decision using an if statement. If the processed EMG signals are greater than the threshold, then the code will display both values for Raw and Processed EMG and if the processed EMG value is less than the threshold, it will not display those values and the loop will be reset. When the code ends, the loop resets and will read new values from the sensors as the beginning of the loop will update the processed and raw variables with a new AnalogRead() result and run through the entire process again, plotting these new results, etc, as shown in figure 11.

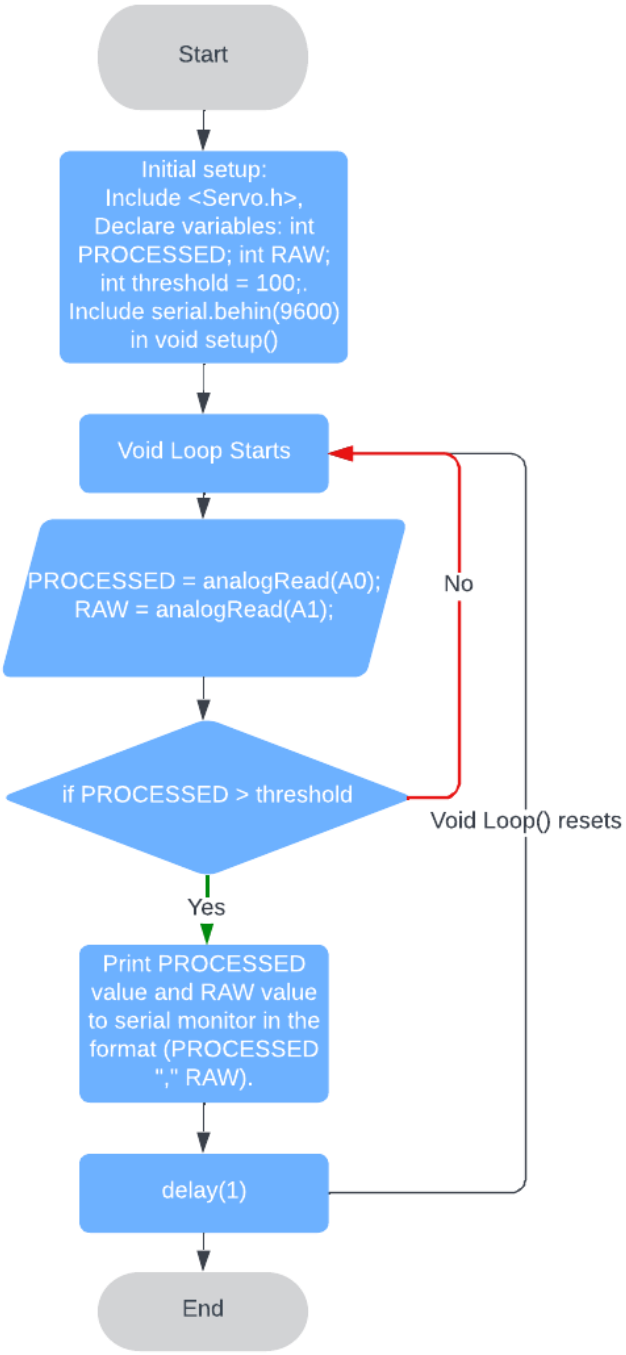


Figure 11 shows the flowchart illustrating the plan to code the EMG classify program

It has been noticed that an event occurs above 100, and therefore the threshold has been defined as 100 as it is an integer, it has been stored in the int variable, as well as processed EMG and Raw EMG signals. Serial.begin(9600) has been set for this code too so that the Arduino can send the commands through its USB connection at the baud rate. Then after AnalogRead() has been defined as per section 3, the *if* statement has been defined in the loop and it commands that if the processed EMG signals are higher than the threshold, then the serial plotter will create a graph for both raw and EMG signals,

if the values are below the threshold, they will not be shown. This threshold was chosen by looking at the graph and locating the minimum value at which an event occurred and therefore was created to get rid of motion artifacts and other types of noises. At the end, a delay of only 0.001 seconds has been added. The delay can be increased but was kept small to see the change in the serial plotter graph as the event happened.

```
#include <Servo.h>
int PROCESSED; //PROCESSED EMG
int RAW; //RAW EMG
int threshold = 100;
void setup() {
 Serial.begin(9600);
}

void loop() {
 PROCESSED = analogRead(A0);
 RAW = analogRead(A1);

 if(PROCESSED > threshold) {
 Serial.println(PROCESSED);
 Serial.print(",");
 Serial.println(RAW);
 }
 delay(1);
}
```

5. Combined code

The final code will be a combination of the EMG classification program and the gripping pattern program, the following flow chart will be used to aid in the development of that code and act as a guide on how it should be structured. The first step is to set up the variables, threshold, the AnalogRead() values, 'PROCESSED' and 'RAW', these parameters will tell the Arduino what to read and from where. The 'PROCESSED' and 'RAW' values are printed to the serial monitor for later plotting. Next is the if statement, this checks if the 'PROCESSED' value is greater than the threshold. If no, then the program starts the loop again and reads new values for 'PROCESSED' and 'RAW'. If yes, then multiple if/else if statements check to see if the variable posCount is equal to either 0, 1, 2, or 3 and runs the corresponding grip pattern based on this value,

If the EMG signal is lower than the threshold set, which is set to be corresponded to when an event occurs, then no gripping pattern is read and nothing is displayed, the hand will go back to its default reset position, as shown in figure 12, at the end of the code a small section of logic has been included that checks the value of posCount, if this value is greater than 3 which is the max value the program considers, then it resets the value to zero and restarts the loop to avoid any errors. An alternative to this would be a sort of try/catch error code to state that if the value is outside the acceptable range then display an error message to the console/serial monitor or reset the loop after updating the value.

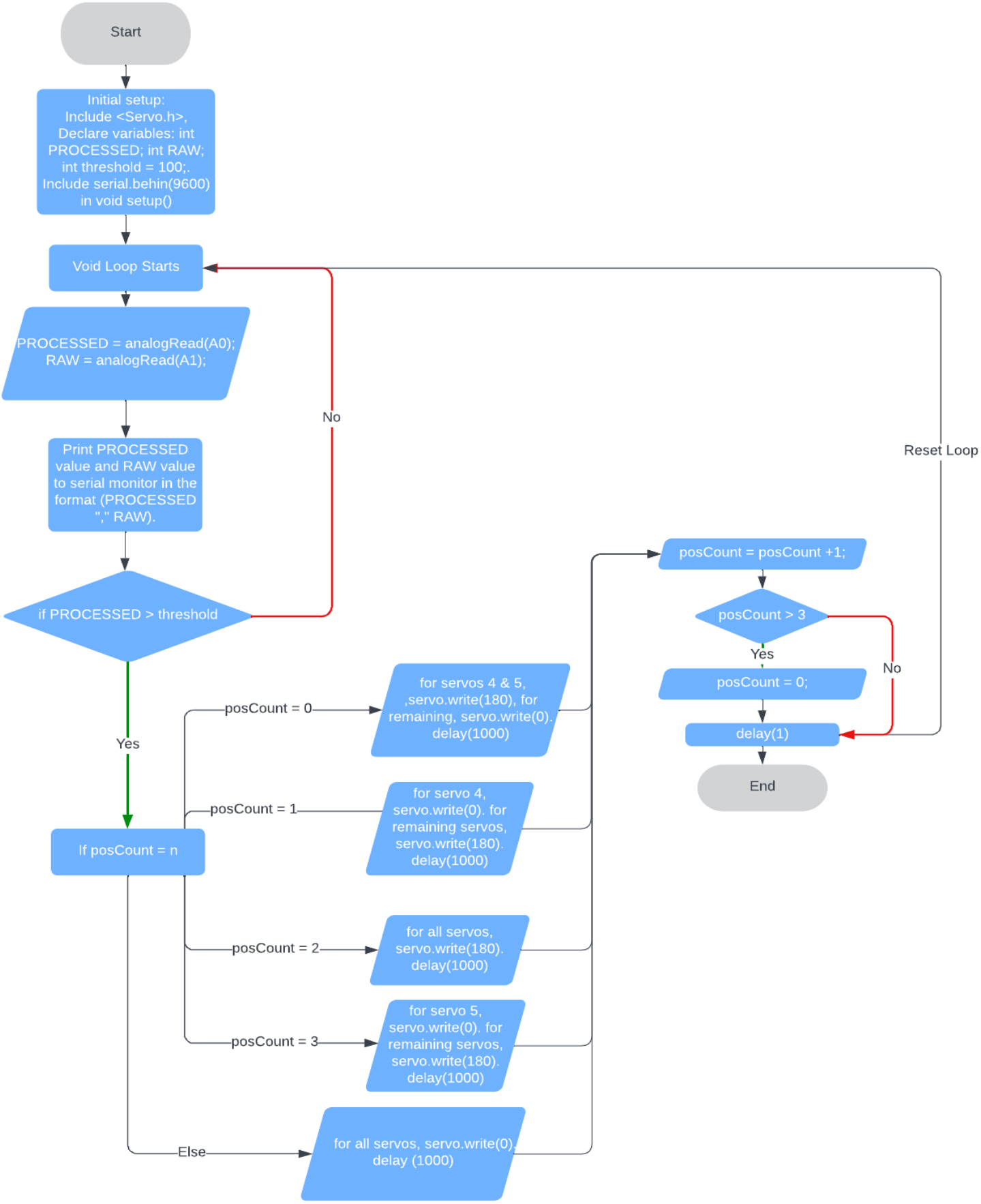


Figure 12 shows the flowchart with the plan to create the combined code

The code created following the flowchart in figure 12 shows at the beginning, before the void setup, all the relevant variables. First thing, servo.h library was added to the code and then the results for Processed EMG, Raw EMG, and threshold were saved in *int.* As mentioned previously, this is so the data obtained through these variables are saved into number storage. The *Int threshold*, which could be a value between 0 and 1024, determines which signal has to count as an EMG signal and which one should be ignored. All the servos are also defined, this gives the Arduino a chance to know where to read data from. Data in *Int* was stored for both PosCount and Pos. Pos, as defined previously, comes from the servo.h library and defines the position of the servo in degrees, this variable helps to define if the servo is moving 180 degrees back to position 1 or 0 where 0 is the point of origin (all fingers are up) and 1 is the position where all fingers are down in a fist. The pin attachments, which servo is attached to pins 3, 5, 6 9, and 10, are also defined. Then the void setup can be started by introducing *serial.begin(9600)* which is the speed the information is being sent. The *if* statement from EMG classify program is reintroduced, but the *else* statement is added. Instead of not displaying anything if the Processed EMG signal is not higher than the threshold, this time the hand will just go back to its resting position (all fingers are up). PosCount is a variable of Arduino used to count the number of positions increased once the loop is started. It is defined as PosCount = PosCount +1, also known as PosCount++, which means increments after the use of the operation. For posCount==0 the hand will do the pinch movement, for posCount==1, it will point, for PosCount == 2, it will do the spherical movement and for PosCount == 3, it will do the power sign. For anything above 3 or different from these operations, the hand will go back to its resting position, defined by the statement *else*. After each movement, a delay of 1 second has been added to be able to see the movement clearly before the next.

```
#include <Servo.h>
int PROCESSED;
int RAW;
int threshold = 50;
Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;
Servo servo5;
int posCount = 0;
int pos = 0;


void setup() {
 Serial.begin(9600);
 servo1.attach(3);
 servo2.attach(5);
 servo3.attach(6);
 servo4.attach(9);
 servo5.attach(10);
}

void loop() {
 PROCESSED = analogRead(A0);
 RAW = analogRead(A1);
 Serial.println(PROCESSED);
```

```
Serial.print(",");
Serial.println(RAW);

if(PROCESSED > threshold) {
   if (posCount == 0){
        servo1.write(0);
        servo2.write(0);
        servo3.write(0);
        servo4.write(180);
        servo5.write(180);
        delay(1000);
   }

   else if (posCount == 1){
        servo1.write(180);
        servo2.write(180);
        servo3.write(180);
        servo4.write(0);
        servo5.write(180);
        delay(1000);
   }

   else if (posCount == 2){
        servo1.write(180);
        servo2.write(180);
        servo3.write(180);
        servo4.write(180);
        servo5.write(180);
        delay(1000);
   }

   else if (posCount == 3){
        servo1.write(180);
        servo2.write(180);
        servo3.write(180);
        servo4.write(180);
        servo5.write(0);
        delay(1000);
   }

   else {
        servo1.write(0);
        servo2.write(0);
        servo3.write(0);
        servo4.write(0);
        servo5.write(0);
        delay(1000);
   }
```

```
 }

 posCount = posCount +1; //increments of 1
 if (posCount > 3){
    posCount = 0;
 }

 delay(1);
}
```

6. Electrodes placement, justification, and reasoning

In this section, an explanation of the electrode placement has been laid out.

6.1 Surface preparation

The skin surface was prepared by rubbing an alcohol wipe onto the skin to improve the electrical contact between the electrode and the skin, however, a better result would have been given by vigorously rubbing with a hypoallergic pad to remove dead skin and further improve the quality of the signal (Jonasson, 2007). Once the skin was prepared, the electrode was laid vertically parallel to the muscle fibres as the electrical charge runs vertically through them (Hernández-Ochoa and Schneider, 2018). The electrode was placed between the innervation zone and the tendon, at the muscle belly of the muscle to avoid cross-talk and obtain a clear signal (Roy, De Luca and Schneider, 1986), as explained in section 6.2.

6.2 Location

Surface EMG has been used for this experiment, this is because it is easier to use than other types of electrodes used to collect EMG signals due to mainly two reasons, first, it does not count discomfort when applied, and second other electrodes such as needles can affect the results due to the patient feeling discomfort because of it (Péter et al., 2019). To be able to understand the placement of the electrodes, it is important to understand what is happening to the muscles in the arm where it is placed.

As can be seen from the figure, the flexor carpi radialis was used for the target electrodes (the top electrode touches the middle of the flexor carpi radialis and the second one the bottom part of the same muscle) while the reference electrode was placed on the elbow portion.

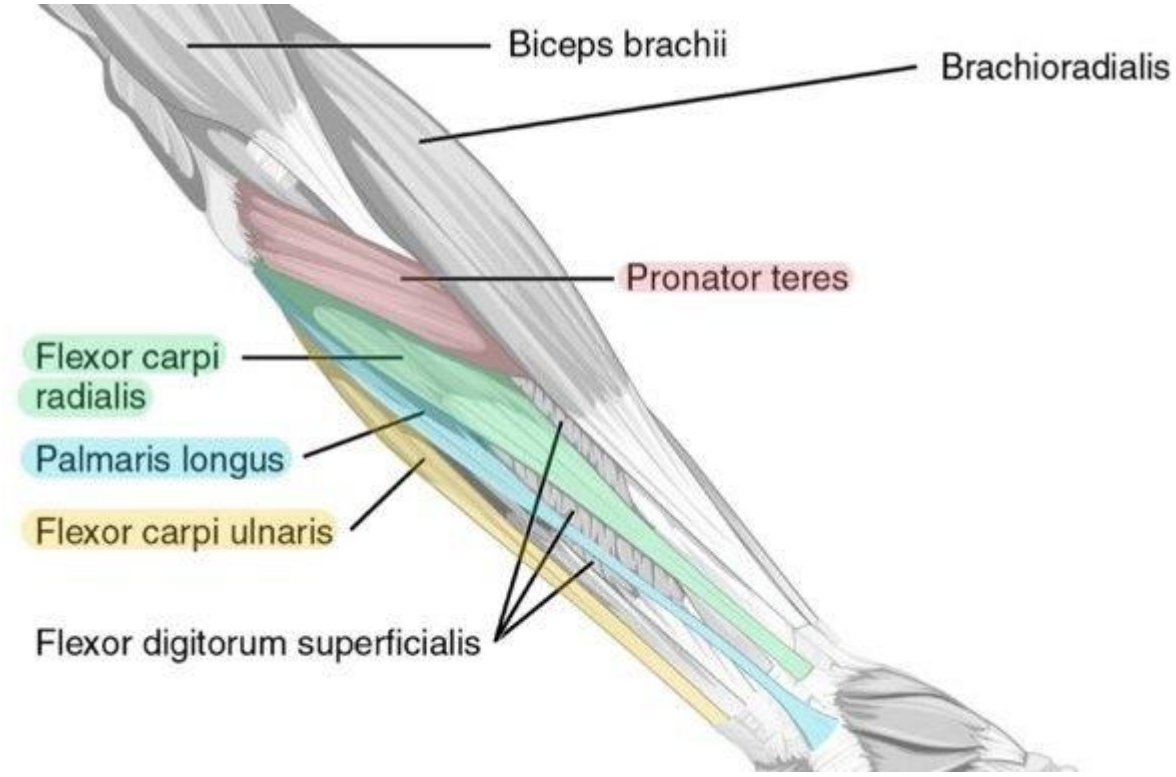


Figure 13 showing the location of the flexor carpi radialis in green (Jones, 2018)

Whenever a muscle contracts, it generates an accidental signal that spreads through the muscle, this is what we know as the EMG signal and is caused by random signals created by motor units present in

the muscle, these motor units are completely random and cannot be predicted. However, due to the large amounts of motor units, when the muscle is contracted, large amplitudes will be created and a signal generated. These signals, when measured, tend to be very small as they are not the whole force generated but just a fraction of it, in fact, they usually tend to be between 10mV, and the higher the signal wanted, the more effort the voluntary contraction will need to have (Jamal, 2012). These signals received are what is known as EMG signals and in this laboratory, these raw signals were processed by rectifying them and using low-pass filtering to remove noise.

6.3 reasoning

Now that location has been clarified, it is important to notice that these specific muscle groups are responsible for palm and wrist movements, and in our case (American Society for Surgery of the Hand, 2022), the robotic arm has to simulate palm movements through the gripping pattern.

In fact, if we analyse it more in-depth, we have to consider both flexor carpi ulnaris and radials and we will notice that:

The flexor carpi ulnaris is an elbow muscle and is divided into two heads, the biggest one coming from the ulna and covering 2/3 of the forearm in length, at the end, it becomes a cord made out of flexible tissue, known as tendon, that crosses the wrist and ends at the base of the palm, where a small bone found in the carpus, known as pisiform bone (American Society for Surgery of the Hand, 2022). The role of this muscle is to bend and move the wrist away from the thumb, also known as the second stage of throwing dart motion (Themes, 2019), as shown in the figure.

Figure 14 showing throwing dart motion (Themes, 2019)

The flexor carpi radial is another elbow muscle, near the pronator (American Society for Surgery of the Hand, 2022) as shown in figure 14. This muscle crosses the elbow and wrist and its role is to bend the wrist and move it towards the thumb, it helps to complete the first stage of the throwing dart motion (Themes, 2019). When recording the EMG signals, a reference point is needed. A reference point is the zero potential point between the volunteer getting their EMG signals taken and the system used to take them, it is also used to remove background noises picked up by the sensor. The signal detected by the active electrodes is compared to the signal received from the reference point, which is usually placed on connective tissue, this is why in this case is placed on the bone. This reference point has a neutral electrical charge, as its net electrical charge is zero (Araz Rawshani, 2017).

The active electrodes were placed between the innervation zone and the tendon (A section of figure 15) of the flexor carpi radialis as it's the part of the muscle with the highest frequency ranges compared to the other parts of the muscle, as shown in figure 15. In fact, near the myotendinous

junction (D of figure 15) the muscle fibres are less and thinner and therefore electrode placement is harder along with the risk of cross-talk. Risk of cross-talk is also found at the edge of the muscle as well (C of figure 15) (De Luca, 1994).

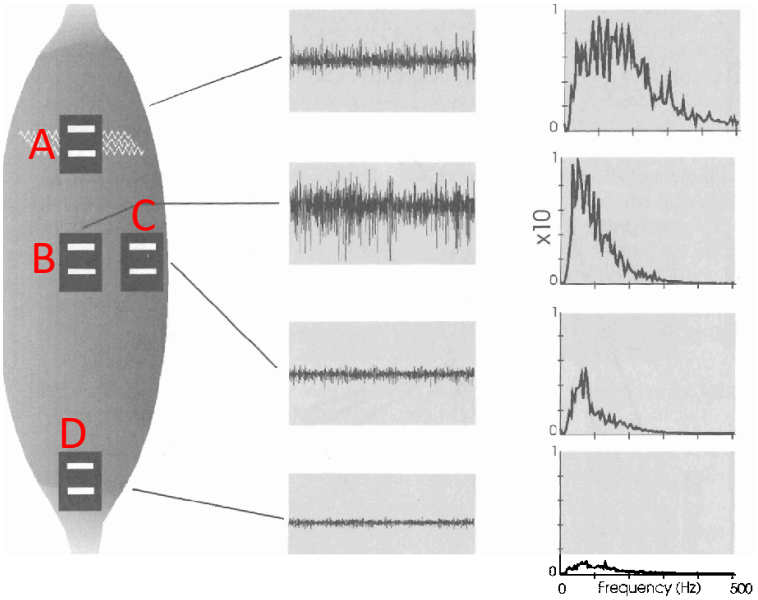


Figure 15 shows 4 electrode placements: in A - a point between the innervation zone and the myotendinous junction, B- the muscle belly, C-lateral edge of the muscle, and D on the myotendinous junction (De Luca, 1994).

7. Improvements

The EMG signals were obtained by using a single threshold method, this method works by comparing the rectified signals to the chosen threshold depending on the background noise observed. The main issue with choosing this method is that it is dependent on the choice of threshold and therefore has a very high risk of carrying the noise in the signal. When obtaining EMG signals, a series of noises are obtained due to physiological factors, inherent noise already present in the electrode, movement artifacts, crosstalk, electromagnetic noises, and internal noises which all affect the results obtained. To improve these results, a series of actions can be taken, which are listed below.

7.1 Inherent noises

Some of these factors, such as inherent noises from the electrode (between 0 and several thousand Hz) are quite hard to remove especially when the impedances are not below 10,000, causing the signal to pick up both motion artifact and radiated interference as high impedance reduces signal quality and gives a low signal to noise ratio. The best way to improve impedance is to use smaller electrodes as higher impedance is directly correlated to the size of the electrode. Alternatively, the noise can be removed by creating a more complex circuit or using higher-quality equipment (Chowdhury et al., 2013).

7.2 Motion artifact

It is known that muscle fibres generate an electrical activity when said muscles are used as their length will decrease and therefore the skin and electrode interface will cause motion artifacts due to them moving against each other. As usual, this type of noise ranges between 1-10Hz (Navarro et al., 2005). Even though recessed electrodes were used for the experiment reducing the motion artifact significantly due to the interface created by the electrolyte gel between the electrode and the skin, some will remain affecting the results (Beltran-Alacreu et al., 2022). The remaining motion artifact is usually mainly due to the potential difference between the skin layers and is not only improved by

reducing the skin impedance but also by preparing the skin better before applying the electrode (Chowdhury et al., 2013). It was found that by vigorously rubbing the skin with a hypoallergic tape and removing the dead skin after cleaning it with an alcohol wipe (Ad Instruments, 2022), the potential difference decreases along with motion artifact, this is due to the stratum corneum decreasing and therefore decreasing the resistance found in the epidermis and making it easier for electrical charge to pass through. Another way to improve this would have been using electrode needles, and percutaneous electrodes rather than transcutaneous but due to the nature of the experiment, it is easier to use surface electrodes even though percutaneous electrodes would have got rid of the skin/electrode interface and would have reached the target muscles more easily and accurately. Surface electrodes have been chosen to use percutaneous electrodes a better knowledge of anatomy would have been needed, there would have been some ethical issues linked with the test being done on humans and the contact with human blood, and in some cases, local anaesthesia needed (Beltran-Alacreu et al., 2022). The wavelet procedure is a procedure that uses time-limited wavelengths that help to examine noise in two domains, frequency and time, and exist by superimposing the signal for a specific short amount of time to reduce noise. It reduces it by decomposing the signal. The signal is divided into two sections, high and low frequencies, and the low frequency is further divided into high and low frequencies again where the high frequency is the wanted signal details while the low is considered a small approximation of said signal and is then removed. Compared to other methods, it has been proven to be the most adequate when trying to remove as many motion artifacts as possible (Frye, 2016).

7.3 Electromagnetic and Internal noise

Electromagnetic noises are continuously generated by the human body and the environmental source of these environmental noises affects the results as sometimes it cancels them out or it is greater than the signal recorded (Reaz, Hussain and Mohd-Yasin, 2006). It is impossible for the human body to not emit these electromagnetic radiations and interact with the environment that has noises around 60Hz, also known as Power-Line Interface (PLI). It is easy to get rid of PLI if it is not within the EMG signals, by using a high pass filter but if it is, then PLI references signal should be found and removed by using cancellation systems such as an adaptive notch or Laguerre filter (Clancy, Morin and Merletti, 2002). When obtaining EMG signals from muscle tissues, conductivity depends on the frequency and the amount of tissue between the specific muscle and the electrodes affects the results by reducing the EMG signal's magnitude. This is due to skin having very low conductivity and the signal having to travel through the muscle tissues. Therefore, it is advised to use subjects with less subcutaneous fat as it has been shown that as the subcutaneous tissue increases and therefore the distance between the electrode and active muscle increases, EMG activity drops (Hemingway, Biedermann and Inglis, 1995).

7.4 Cross talk and electrode positioning

Crosstalk from nearby muscles can be avoided easily by choosing the appropriate electrode position. During the course of this experiment, the electrode was placed by pinpointing where the chosen muscles were located. The wrong placement of said electrodes would have caused the wrong signals to be obtained and crosstalk from nearby muscles as the muscles chosen are relatively thin and it is very easy to end up taking the signal from a nearby muscle or the edge of the chosen muscles rather than the muscle belly. Therefore, it is advised to find the position of the chosen muscles by knowing that the reference electrode has to be placed in the elbow, which position is very easy to find, while the other two electrodes have to be placed on: the flexor carpi ulnaris that can be found by bending the right elbow with the palm towards the person’s face and kept at the level of the person’s head, the pinky side of the right forearm is the flexor carpi ulnaris while the radialis is found at the centre of the forearm when the palm is pointed upwards (Lung and Siwiec, 2019). Both muscles have been

mentioned as a more accurate result would have been obtained by using both radialis and ulnaris rather than only ulnaris as they are both fundamental for hand movements and therefore would have given a more accurate representation of the signals needed to use the prosthetic hand. It is also important to notice that a stronger signal is obtained from the biceps even though less noise is obtained in the forearm therefore including a higher range of electrodes and EMG signals would have helped to understand if the signals obtained from a specific muscle were not accurate and adjust its position.

7.5 The arm

When taking the EMG signals, it was noticed that the prosthetic arms phalanges were very loose and did not perfectly move from 0 to 180 degrees, this affected the performance of the arm, and the fingers should be screwed tighter. Using surgical tape would have helped to keep the electrodes in position as well as they kept moving and therefore affecting the accuracy of the results.

Along with the use of more electrodes, the data obtained and analysed on Arduino serial plotter could have been analysed in Excel to see exactly the peaks and baseline values and find a better threshold.

8. Conclusion

This laboratory activity went as planned. Code was developed to let a prosthetic hand follow 4 gripping patterns: pinch, point, spherical, and power. The code to capture EMG waveforms and one to classify them were created successfully as proven by the Emg Raw and processed signal graphs at rest and while the muscle is activated. Finally, a final program connecting the EMG signals received from the user's flexor carpi radialis to the gripping patterns was created and tested. It also has been explained that the location of the electrode on the flexor carpi radialis was due to the muscle being responsible for the throwing dart motion. Finally, the improvements suggested were mainly related to the pre-processing stage of signal acquisition and increment of electrodes.

9. References

AD Instruments (2022). *How does one prepare the skin prior to recording with Delsys EMG sensors?* [online] ADInstruments. Available at: https://www.adinstruments.com/support/knowledge-base/how-does-one-prepare-skin-prior-recording-delsys-emg-sensors.

American Society for Surgery of the Hand (2022). *Body Anatomy: Upper Extremity Muscles | The Hand Society*. [online] www.assh.org. Available at: https://www.assh.org/handcare/safety/muscles.

Araz Rawshani (2017). *The ECG leads: electrodes, limb leads, chest (precordial) leads, 12-Lead ECG (EKG)*. [online] ECG learning. Available at: https://ecgwaves.com/topic/ekg-ecg-leads-electrodes-systems-limb-chest-precordial/.

Arduino (2023). *Arduino Reference*. [online] cdn.arduino.cc. Available at: https://cdn.arduino.cc/reference/en/language/variables/data-types/int/ [Accessed 1 Jan. 2023].

Beltran-Alacreu, H., Serrano-Muñoz, D., Martín-Caro, D., Fernández-Pérez, J.J., Gómez-Soriano, J. and Avendaño-Coy, J. (2022). Percutaneous versus transcutaneous electrical nerve stimulation for the treatment of musculoskeletal pain. A systematic review and meta-analysis. *Pain Medicine*, 8. doi:10.1093/pm/pnac027.

Chowdhury, R., Reaz, M., Ali, M., Bakar, A., Chellappan, K. and Chang, T. (2013). Surface Electromyography Signal Processing and Classification Techniques. *Sensors*, [online] 13(9), pp.12431–12466. doi:10.3390/s130912431.

Clancy, E.A., Morin, E.L. and Merletti, R. (2002). Sampling, noise-reduction and amplitude estimation issues in surface electromyography. *Journal of Electromyography and Kinesiology*, 12(1), pp.1–16. doi:10.1016/s1050-6411(01)00033-5.

De Luca, C.J. (1994). The wartenweiler memorial lecture the use of surface electromyography in biomechanics. *Journal of Biomechanics*, [online] 27(6), p.724. doi:10.1016/0021-9290(94)91124-x.

Frye (2016). *Wavelet Analysis - an overview | ScienceDirect Topics*. [online] Sciencedirect.com. Available at: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/wavelet-analysis.

Hemingway, M.A., Biedermann, Heinz-J. and Inglis, J. (1995). Electromyographic recordings of paraspinal muscles: Variations related to subcutaneous tissue thickness. *Biofeedback and Self-Regulation*, 20(1), pp.39–49. doi:10.1007/bf01712765.

Hernández-Ochoa, E.O. and Schneider, M.F. (2018). Voltage sensing mechanism in skeletal muscle excitation-contraction coupling: coming of age or midlife crisis? *Skeletal Muscle*, 8(1). doi:10.1186/s13395-018-0167-9.

Jamal, M.Z. (2012). *Signal Acquisition Using Surface EMG and Circuit Design Considerations for Robotic Prosthesis*. [online] *www.intechopen.com*. IntechOpen. Available at: https://www.intechopen.com/chapters/40131.

Jonasson, L. (2007). *A prospective study on the relevance of skin preparation for noise, impedance and ECG intervals among healthy males*. [online] Available at: https://www.diva-portal.org/smash/get/diva2:238038/FULLTEXT01.pdf.

Jones, O. (2018). *Muscles in the Anterior Compartment of the Forearm*. [online] TeachMeAnatomy. Available at: https://teachmeanatomy.info/upper-limb/muscles/anterior-forearm/.

Lung, B.E. and Siwiec, R.M. (2019). *Anatomy, Shoulder and Upper Limb, Forearm Flexor Carpi Ulnaris Muscle*. [online] Nih.gov. Available at: https://www.ncbi.nlm.nih.gov/books/NBK526051/.

Monk (2021). *Arduino Lesson 5. The Serial Monitor*. [online] Adafruit Learning System. Available at: https://learn.adafruit.com/adafruit-arduino-lesson-5-the-serial-monitor/arduino-code.

Navarro, X., Krueger, T.B., Lago, N., Micera, S., Stieglitz, T. and Dario, P. (2005). A critical review of interfaces with the peripheral nervous system for the control of neuroprostheses and hybrid bionic systems. *Journal of the Peripheral Nervous System*, 10(3), pp.229–258. doi:10.1111/j.1085-9489.2005.10303.x.

Péter, A., Andersson, E., Hegyi, A., Finni, T., Tarassova, O., Cronin, N., Grundström, H. and Arndt, A. (2019). Comparing Surface and Fine-Wire Electromyography Activity of Lower Leg Muscles at Different Walking Speeds. *Frontiers in Physiology*, [online] 10(10), p.1283. doi:10.3389/fphys.2019.01283.

Reaz, M.B.I., Hussain, M.S. and Mohd-Yasin, F. (2006). Techniques of EMG signal analysis: detection, processing, classification and applications. *Biological Procedures Online*, [online] 8(1), pp.11–35. doi:10.1251/bpo115.

Roy, S.H., De Luca, C.J. and Schneider, J. (1986). Effects of electrode location on myoelectric conduction velocity and median frequency estimates. *Journal of Applied Physiology*, [online] 61(4), pp.1510–1517. doi:10.1152/jappl.1986.61.4.1510.

Themes, U.F.O. (2019). *The Kinematics and Clinical Implications of the Dart-Throwing Motion*. [online] Musculoskeletal Key. Available at: https://musculoskeletalkey.com/the-kinematics-and-clinical-implications-of-the-dart-throwing-motion/.