A low cost EMG Graphical User Interface controller for robotic hand

Manex Ormazabal Arregi[1] and Emanuele Lindo Secco[1]

[1] Robotics Laboratory, Liverpool Hope University, Liverpool, UK
20211323@hope.ac.uk, seccoe@hope.ac.uk

Abstract. We propose a low-cost EMG Graphical User Interface (GUI) controller for robotic hand, which is based on combining a wearable ElectroMyoGraphic (EMG) MyoWare sensor with a 5-fingers under-actuated Open Bionics Robotic Hand and a customized software. The implementation of the controlling GUI allows performing grasping configuration of the hand and real-time visualization of the EMG signal driving the hand activation. A set of preliminary test and validation shows the proper functioning of the system and suggest the possibility of developing very low-cost upper limb prosthetics for dexterous recovery of amputees.

Keywords: low cost EMG, robotic hand, prosthetic applications.

1 Introduction

Nowadays limb loss affects annually more than 1 million people worldwide, which is mainly caused by trauma events, disvascularity, foot ulcers, diabetes or neoplasia and it is expected that the numbers will grow by the near future [1, 2]. Such amputations are affecting the daily life of humans both in a psychological and physical way as they suffer the incapacity to perform some basic tasks that they were able to do before, which also leads to creating anxiety and depression problems in the person.

Therefore, the demand to develop prostheses to replace either an upper or lower lost limb is high, and recent scientific developments and research have demonstrated a huge improvement on the advancements of these devices, especially focused on the design of these elements [3]. Today´s one of the most used manufacturing techniques for the design of prosthetic limbs is three-dimensional (3D) printing.

This fabrication method, despite there are some concerns on the robustness and wide-user acceptance, offers the advantage of creating affordable and customizable solutions, and the possibility for easy maintenance and repair [4]. The main advantage of 3D printed devices is the low-cost manufacturing, as many other prostheses are much more expensive due to the complexity of the mechanism and the cost of the material quality. Nowadays, there are many companies dedicated to the manufacture of 3D printed prosthetics and working to make them accessible to everyone who needs them [5].

According to the type of prostheses, they are classified into passive or cosmetic types on the one hand, and active or functional types on the other. Cosmetic prostheses

can only restore the aesthetic aspect of the person, while the active ones, apart from the aesthetic aspect, can also replace the functionality of the lost limb [6]. The active prostheses are classified into three groups, and depending on the method of control they use, they can be body-powered through cable extension or contraction of the more proximal joints, button press, and by the use of myoelectric control [6,7].

ElectroMyoGraphy (EMG) is one of the most used control systems in nowadays bionic limbs [6,8]. This system interprets the electrical activity of a specific muscle of the human body by detecting the pulses and movements of muscles and sends a variable electrical signal depending on the contraction of the muscle, which can be processed and after used in different applications, such as assistive devices, teleoperation of robots, haptic devices, virtual reality, etc [9].

EMG signals have been used in prosthetic devices since 1948, and the most common technique used to control prostheses, is the on-off technique, which is normally used with one degree of freedom (DoF) devices and consists of the performance of two opposite movements, for example opening and closing a hand, that is changed when the EMG signal amplitude of two antagonist muscles exceed a preset threshold [6]. However, over the years researchers have developed new strategies in the use of myoelectric control-based systems and one of them, is the proportional control strategy, which in this case is considered the voltage applied to the motor or actuator to be proportional to the contraction of the muscle or the intensity of the EMG signal.

In this project, a 3D printed prosthetic hand developed by Open Bionics company, and a Myoware muscle sensor are used. The project aims to control the prosthetic hand using the Myoware sensor and replicating (the opening and closing movements of the hand) some gestures of the human hand with the robotic hand. For this, the on-off strategy is used, so that depending on the contraction of the muscle, the electric signal coming from the sensor will increase or decrease, allowing to change the gesture of the robotic hand whether the signal is below or above the threshold.

The motivation of this study comes from the need of developing low-cost system, given the very expensive cost of current multi-dexterous prosthetic hand. Moreover the adopted set-up provides a benchmark for the development of grasping algorithm in an open source and intuitive programming language.

The scope of this paper will look first at the description of the material and methods used in the project, which includes both hardware and software parts. After, the procedure followed to connect the hardware part and program the software part will be explained step by step. The results obtained with the project will be discussed in the Evaluation and Discussion section, and finally, the conclusion and future work that is required will be mentioned in the last two sections.

2 Materials and Methods

Here we detail the main components of the system and differentiate between the Hardware and Software.

2.1 Hardware

The Hardware architecture is substantially made of two main components: a robotic hand and an EMG low-cost interface.

The Open Bionics robotic hand

In this project, the ADA V1.1 robotic hand model from the Open Bionics company has been used (Figure 1). This company provides all the necessary design files to build the hand in 3D print, the instructions to easily assemble it, and also the internal elements, such as the electronic parts, which can be purchased on their website [9].

One of the main reasons to choose this device was that it is designed to be easy to build and repair, and because it is easy to manipulate it and to use the software, which can be programmed in Arduino. So, at the end, the hand is designed to be open source for any type of user or applications in which it is wanted to be used, being also ideal as a platform for robotics and prosthetic research.

Although the material of the hand is rigid, it is flexible enough so that the fingers can move and reach different positions. Thus, different gestures can be performed with the hand thanks to the linear actuators that are installed in the palm of the hand, which are attached to each finger with strands.

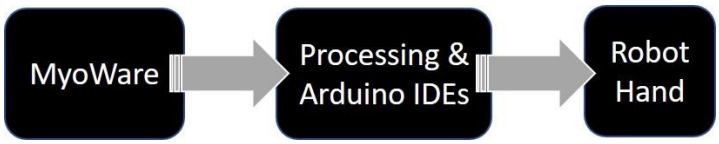

Fig. 1. Overview of the proposed system

One important element of the hand that needs to be considered, is the Almond V1.2 board (Figure 2). This microcontroller is in charge of controlling the linear actuators of the hand so that they perform the task to move the fingers according to the program that is uploaded to it. The *Printed Circuit Board* (PCB) has three connectors which are the power supply input, USB- type C connector to communicate to the computer, and the headphones jack cable input [9].

This last mini-jack input option is for any external device that is wanted to connect with the board as an input signal and thus, to be able to control the robotic hand with the connected element, which in this project the Myoware sensor was used as an input device. The type of cable to communicate the board of the hand with an external element must be a 4-pole headphone jack of 3.5mm [10]. This cable houses 4 internal wires which need to be connected with the Myoware sensor, as shown in Figure 2. The black wire is the ground and the red one is the positive which supplies 5 volts to the Myoware sensor. The green and blue wires are the signal wires, and depending on whether it is used one or two Myoware sensors it will be necessary to use one signal wire or both. If a single Myoware is used, which is the case of this project, the green wire will be used as the signal wire.

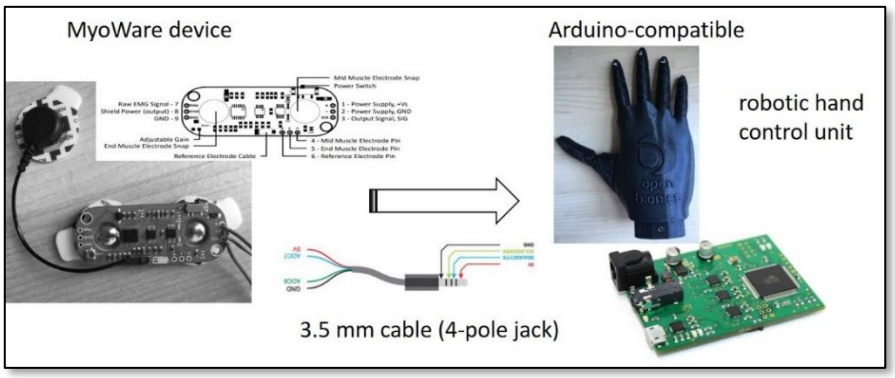

Fig. 2. The main Hardware components [9, 10, 14]

Myoware sensor

The Myoware muscle sensor, developed by Advancer Technologies company, is a non-invasive electromyographical (EMG) wearable sensor capable of detecting muscle activation of the human body and converting it into electrical signals, which can be processed in a controller (Figure 3) [13].

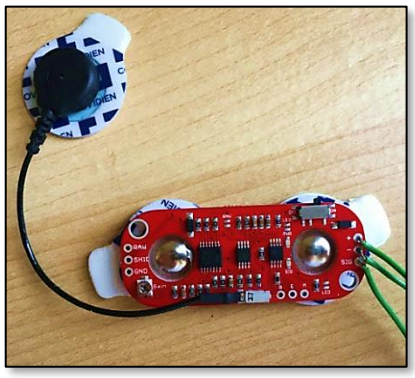

Fig. 3. The MyoWare sensor

Based on the Surface electromyography (sEMG), this sensor is attached to the human skin in different parts of the body, and thanks to its electrodes captures EMG signals, by measuring the electrical activity of the muscle, which varies depending on the contraction level of the muscles [11, 12]. The sensor provides either a rectified and integrated analog output signal, which is suitable to use directly as an input signal with a microcontroller's *Analog-to-Digital Converter* (ADC), or a RAW EMG signal which is not processed [13].

Focusing on the technical specifications of this device (Figure 4), the sensor produces an output voltage range between 0 and the supply voltage, which can be a

range of + 2.9 V to + 5.7 V. There are three basic electrodes included in the sensor, which two main ones are in the board to measure the muscle activation and an external electrode as a reference. However, the device provides the option to add three more electrodes if needed. In both ends of the board, there are the two types of output signals, which are the Envelope and RAW signal outputs, and a small potentiometer is also included to adjust the gain of the output signal. This gain needs to be calibrated so that when the muscle is relaxed it gives an output of 0 V [14].

There are different muscle parts of the human arm where the MyoWare sensor could be attached to, to interpret different gestures of the hand as it is shown in Figure 4 [10].

On the one hand, the two onboard electrodes are attached to the muscle that is wanted to measure, one in the middle of the muscle and the other one at the end of the muscle, and they can be positioned in two different parts of the arm, which can be on the inside forearm or outside forearm. Then the reference electrode, which should be away from the muscle, can be placed on the elbow, as shown in Figure 6, or on any part of the arm other than the muscle being measured.

Fig. 4. Positioning of the MyoWare electrodes

2.2 Software

The Software architecture is substantially made of two main components: an Arduino IDE code for the processing of the EMG sensor combined with an interface written in Processing for the data visualization and manipulation. The Arduino code is also comprehensive of the software which is controlling the robotic hand.

Arduino IDE

The Integrated Development Environment (IDE) of Arduino is an open-source software based on the C/C++ programming language, and it is mainly used for writing, compiling and uploading the code in an Arduino device [15].

An Arduino IDE program, also known as sketch, is divided into 4 sections. The first section is where all the variables are defined and the necessary libraries are imported, which in this project the only library needed was the *FingerLib* for the robotic hand. This library includes the code to move the actuators of the hand. The second section is the setup section where the initial values of the variables are set, the serial port is initialized and the output and input pins of the arduino board are defined. In the third

section, which is the main body of the code, all the commands to execute any type of function that is desired to perform with an Arduino board are included. Finally, the fourth section is used to declare the code functions that are needed in the program [16].

Processing IDE

Processing is an open-source rapid prototyping software sketchbook and an *Integrated Development Environment* (IDE) focused on learning computation design, which is very intuitive to learn and suitable for graphics applications [16, 17].

Although it is based on the Java language, Processing has several language modes, including Python. Besides, it allows the development of *Graphical User Interface* (GUI) and provides support for vector and raster drawing, image processing, mouse and keyboard events, network communication, object-oriented programming and lots more. Focusing on the code part of Processing, as in the Arduino, in Processing there is also a *void setup* section where all the declarations and initializations of the variables are made, and then there is the *void draw* section, where it runs repeatedly the code inside of it, which is similar to the *void loop* section of Arduino [18].

Many times, Processing is used to work together with Arduino, so that the data of the Arduino can be visualized in different ways in Processing, or also commands can be sent from Processing to Arduino in order to control an Arduino board from Processing. For this, both programs are connected through the serial bus, and to establish the communication between them, it is necessary to specify the same serial port in both the Arduino and the Processing side, as well as the same baud rate of the serial bus. In this project, Processing was used to plot the signal coming from the Myoware sensor.

3 System integration

This section will talk about the procedure carried out to connect the Myoware with the robotic hand, which will be discussed in the design part, and in the code part, the program that was created in order to move the robotic hand with the sensor will be explained.

3.1 Design and assembly

As the robotic hand that it was opted to use in this project already had an internal microcontroller, it was not needed to use any other board. Therefore, the only thing that had to be done to communicate the Myoware sensor with the robotic hand was through the use of a headphone jack cable.

First of all, each wire of the mini-jack cable was connected with the wires of the Myoware sensor, which is shown in Figure 2. As it was only used a single Myoware, only one signal wire was needed to use, which was connected to the signal pin of the sensor. The other two wires, which supply the power of 5 V, were connected to the positive (+) and ground (-) pins of the sensor. The other end of the jack cable, which is the head, was connected to the jack input of the hand’s board.

Once the connection between the Myoware sensor and the robotic hand was made, the robotic hand or rather the microcontroller was powered, simply connecting the power supply of it to the power input of the board. Finally, the hand was connected to the computer through the USB cable, so it could transfer the Arduino program to the board and also to read the sensor signal in real time from the computer (Figure 2).

3.2 Code

In this section, the code that was implemented for the project will be explained in two parts. In the first part, the main program created on the Arduino IDE for the control of the robotic hand with the Myoware sensor will be explained, and in the second part, the code of the Processing, which was created to plot the reading of the sensor signal.

Arduino code

In this program the code was written on the one hand to be able to read the signal sent by the sensor and on the other hand, to move the actuators of the robotic hand according to the input signal coming from the sensor. In the next sections, the procedure of the code created in the Arduino IDE will be explained step by step.

In order to control the robotic hand, as it was mentioned before, we adopted the *FingerLib* library (Figure 5), which allows to declare the variables of the actuators, assign a pin to each of them in the setup section and to execute them in the loop section.

```
#include <FingerLib.h> // Import the library of the robotic hand
```

Fig. 5. Import the library

Declarations

The declarations, shown in Figure 6, are global variables which can be used later in the next sections. Firstly, the actuators of the robotic hand were declared specifying the name of each finger. After, the serial bus name was defined as *MYSERIAL* and the pin of the Myoware sensor as *EMGpin*, which is declared as an integer variable. Next declarations are an integer variable named as *sensorValue*, which was used to store the input values coming from the sensor, and a float number defined as voltage, to store the voltage values, once the sensor input signal is converted to voltage. converted from those that are converted from the sensor input signal. was a float number, which was named as *thresValue*. Finally, a float variable defined as *thresValue* was created, to use as a threshold value to define when to open and close the robotic hand according to the sensor value, and a char variable was declared with an initial value of 0, which was used as a command to activate or deactivate the actuators of the hand in the void loop section.

```
// Declaration of the fingers (Actuators):
Finger Thumb;

Finger Index;

Finger Middle;

Finger Ring;

Finger Pinky;

#define MYSERIAL Serial    // Define the name for the serial bus

int EMGpin;               // Declare the variable for the pin of the sensor

int sensorValue;         // Declare the variable for storing the sensor values

float voltage;          // Define a float variable to store the voltage values of the sensor

float thresValue = 4.60; // Declare the variable for the threshold

char command = '0';     // Define the command to switch the gesture of the robotic hand
```

Fig. 6. Variable declarations

Setup

A *serial port* was then initialized with a baud rate of 38400 bits per second, and the Almond V1.2 pins for the Myoware input signal and the actuators of the robotic hand were set (Figure 7). As in this case only a single Myoware sensor was used, the pin attached was A6, and in case of using one more sensor, it will be attached to the pin A7.

```
void setup()
{

  MYSERIAL.begin(38400);  // Initialise the serial port with a baud rate of 38400 bps

  EMGpin = A6;          // Define the pin for the sensor:


 // Define the pins of the actuators (fingers):

  Thumb.attach(13, 4, A5);
  Index.attach(3, 6, A1);
  Middle.attach(7, 8, A2);
  Ring.attach(10, 9, A3);
  Pinky.attach(11, 12, A4);

}
```

Fig. 7. Void setup section

Loop

In the void *loop* section, the code to read the signal coming from the Myoware sensor, and to control the robotic hand was included.

In Figure 8, firstly the reading of the analog input signal of the Myoware sensor is shown, which was read using the *analogRead* function and stored in the *sensorValue* created initially. Then, the obtained values of the sensor, which are bits that are in a range of 0-1023 bits, were converted into voltage values using the calculation method that is shown in Figure 8, and stored in the variable voltage.

Once the reading of the Myoware sensor was properly set, the values were printed onto the serial monitor, in order to visualize the voltage value in real-time while testing the sensor in the muscle. For this, the function *MYSERIAL*. *Println* was used as well in order to print the message and the voltage value of the sensor.

```
void loop()
{

  // read the analog input value of the sensor and store in the variable:
  sensorValue = analogRead(EMGpin);

  //Convert the analog reading (Which goes from 0-1023) to a voltage (0V-5V)
  voltage = sensorValue * (5.0 / 1023.0);


  // print the results to the serial monitor
  MYSERIAL.println("Sensor value");
  MYSERIAL.println(": ");
  MYSERIAL.println(voltage); // Print the voltage values of the sensor onto the serial monitor
  MYSERIAL.print("\t");
```

Fig. 8. Void loop section – part 1

Next step was the selection of the command that would change the function of the robotic hand, which in this project was to open or close the hand (Figure 9). For this, the if and else functions were used to change the command.

- On the one hand, when the human hand is open the muscle is relaxed, so the signal amplitude would be below the threshold and hence, the command to send to the actuators of the robotic hand would be a character number 0, which would open all the fingers.
- On the other hand, when closing the human hand the muscles are contracted, which increase the voltage value of the sensor and hence, the amplitude of the signal passes the threshold. In this case, as the signal value is above the threshold the command value changes to 1, which changes the function of actuators and in this case it would close all the fingers.

```
 // Define the command depending on whether the sensor value is below or above the threshold:

if (voltage > thresValue){    // If the signal value is above the threshold, close the hand
   command = '1';            // Close the hand
 }
 else {                     // If the signal is below the threshold
   command = '0';          // Open the hand
 }
```

Fig. 9. Void loop section – part 2

As shown in Figure 10, to execute the different robotic hand configuration - depending on the type of the command that is selected – a *switch* function is used, which takes the value of the command and executes one function or other.

The first case is when the command value is 0, where all the fingers of the robotic hand will open. The second case is when the value of the command is 1, which will change the function of the actuators and in this case, all the fingers of the robotic hand will close. After each function a delay of 50 milliseconds was used, which was set by default, and also a delay of 500 milliseconds was added at the end of the loop, to give enough time to the analog-digital converter to settle after the last readings of the sensor value.

```
   switch (command) {

       case '0':    // Open the hand
       Thumb.open();
       Index.open();
       Middle.open();
       Ring.open();
       Pinky.open();

       delay(50);

       break;

       case '1':     // Close the hand
       Thumb.close();
       Index.close();
       Middle.close();
       Ring.close();
       Pinky.close();

       delay(50);

       break;

     default:

      break;
   }
  delay(500);    // wait 500 milliseconds before the next loop
}              // for the analog-to-digital converter to settle
              // after the last readings
```

Fig. 10. Void loop section – part 3

Processing code

In this section, a *Graphical User Interface* (GUI) was designed in the Processing Programming Language in order to plot the input signal of the MyoWare sensor is explained step by step.

Libraries and Declarations

In the first part of the program, the library *processing.serial* for the serial communication with the Arduino was imported (Figure 11).

After the library was imported, some variables were declared, such as the name of the *serial port myPort*, the *inByte* global variable to store the sensor values coming from the serial port, a boolean value *newData* set as false, and an integer variable xPosfor the horizontal position of the graph, with an initial value 1. Besides, in order to draw a

continuous line when plotting the signal, two more integer variables are declared which are *lastxPos*, to store the last value of the position *X*, and *lastheight*, to store the last value of the height position.

```
import processing.serial.*;

Serial myPort;        // The serial port
float inByte;         // Incoming serial data
boolean newData = false;
int xPos = 1;         // horizontal position of the graph

//Variables to draw a continuous line.
int lastxPos=1;
int lastheight=0;
```

Fig. 11. Importing the library and variables' declaration

Void Setup section

The second part is the setup, where the size of the screen to visualize the graph and the serial port were configured (Figure 12).

The values received from the serial port are stored in the previously created global variable *myPor*t. To automatically find the port to communicate with the Arduino, the function *Serial.list()[0]* was used and the baud rate used was 9600 bits per second, which is the same that specified in the Arduino part.

Finally, the *Serial.bufferUntil* function was used to print the values every time in a new line and the initial background was set with a 0 value.

```
void setup () {
  // set the window size:
  size(600, 400);

  myPort = new Serial(this,Serial.list()[0], 9600);

  // A serialEvent() is generated when a newline character is received :
  myPort.bufferUntil('\n');
  background(0);      // set inital background:
}
```

Fig. 12. The GUI setup section

Void draw section

In the void draw section, which is the main body of the program, the line of the graph was configured and it is divided into two parts (Figure 13).

In the first part, the color and the width of the line were specified, and then the setup to create the line according to the input byte that is received from the serial port. The height variable takes the input byte each time from the serial port and then writes in the graph as a value of the Y-axis.

The second part is about the setup of the X-axis, in order that the plotter moves forward while it draws the line and resets the screen when the line reaches the end of the window.

```
void draw () {
  if (newData) {
    //Drawing a line from Last inByte to the new one.
    stroke(127,34,255);     //stroke color
    strokeWeight(4);        //stroke wider
    line(lastxPos, lastheight, xPos, height - inByte);
    lastxPos= xPos;
    lastheight= int(height-inByte);

    // at the edge of the window, go back to the beginning:
    if (xPos >= width) {
      xPos = 0;
      lastxPos= 0;
      background(0);  //Clear the screen.
    }
    else {
      // increment the horizontal position:
      xPos++;
    }
   newData =false;
 }
}
```

Fig. 13. The GUI drawing section

SerialEvent function

In the final part of the program, the *serialEvent* function was included, in order to convert the sensor input values coming from the serial port into numerical values and plot them onto the screen. For this, firstly the input values coming from the serial port, which are in ASCII string format, need to be processed in order to get only the ASCII string values and remove everything else that is not needed, using the function trim. After, the ASCII string values are converted into numbers with the float function. Finally, the numerical values are scaled to a range of 0-1023 to fit on the screen height when plotting (Figure 14).

4 Results

After the assembly and the design and implementation of the aforementioned code, some validation tests were made by attaching the MyoWare sensor to our forearm in order to control the robotic hand and to perform the opening and closing gestures with it.

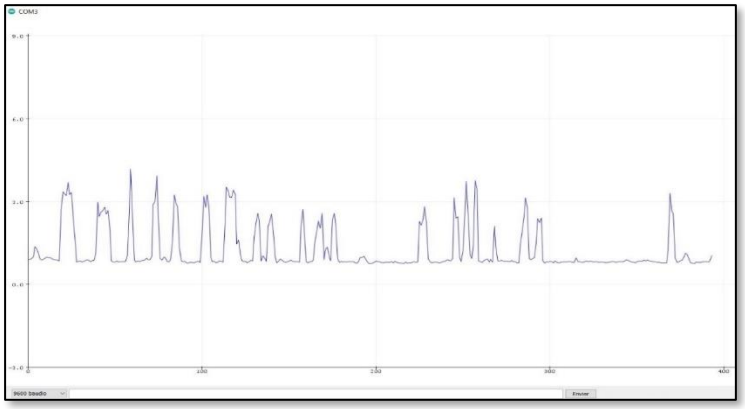

Fig. 14. The MyoWare signal plotting as reported within the Arduino IDE environment

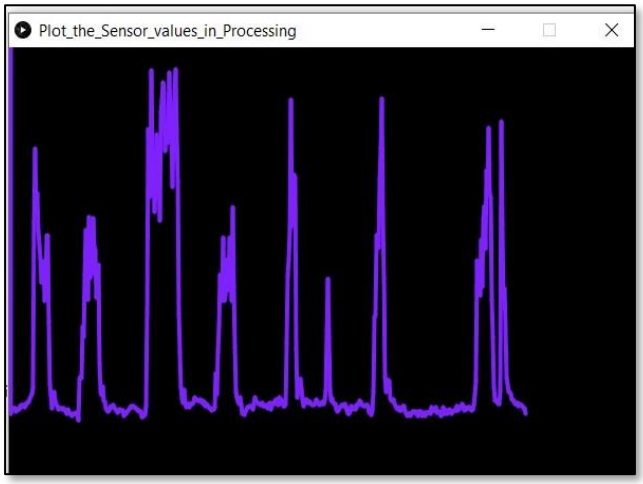

Fig. 15. The developed GUI as reporting the MyoWare signal plotting in the Processing IDE environment

When testing the sensor, at the beginning it was noticed that the signal value of the sensor was too high even without contracting the muscle or not closing the hand. Therefore, the output gain of the sensor was calibrated to 0, so that the signal reading on the Arduino program was clearer. In Figures 15, the reading of the sensor signal that was plotted in Processing and Arduino IDE software are shown, where the highest points are when the muscles are contacted (hand closed) and the lowest points when the muscles are relaxed (hand open).

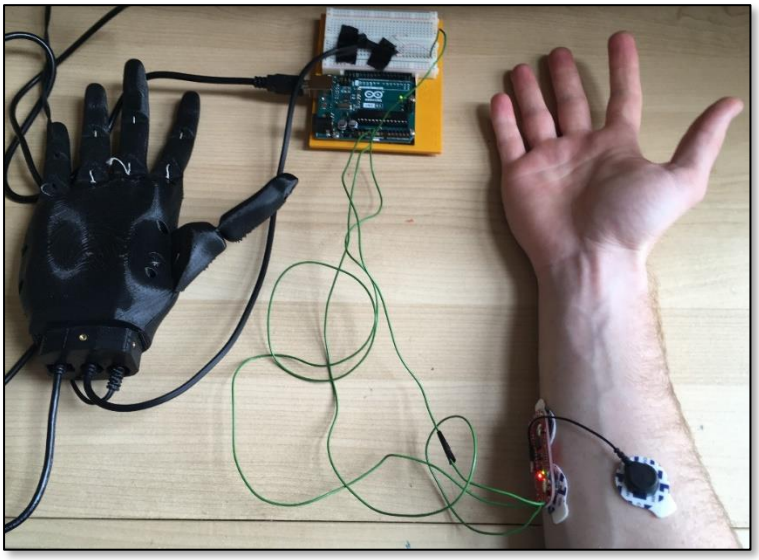

Fig. 16. The opening gesture of the robotic hand

Fig. 17. The closing gesture of the robotic hand

One of the drawbacks found during the testing with the sensor was the signal noise of the sensor. Before testing the sensor with the robotic hand, the sensor was also tested by itself, connecting it to an Arduino UNO board, just to check the reading of the sensor values, and it was seen that the sensor signal was stable and without any noise. However, when testing the MyoWare with the microcontroller of the robotic hand, and

connecting it through the headphones jack cable, it was discovered that the values of the sensor signal were a little bit unstable. This seemed to be because one of those elements, the microcontroller or the jack cable, was causing noise in the sensor signal.

Nevertheless, even if it had some noise, the signal values changed when moving the muscles and the variation of the signal was enough to change the gesture of the robotic hand. In order to start performing the different gestures with the robotic hand, first, a threshold value was set according to the variation of the sensor signal when contracting and relaxing the muscles of our hand. Thus, the opening and closing gesture performances of the robotic hand were successfully achieved using the MyoWare sensor, as shown in Figg. 16-17, which could be used to grasp soft materials, such as a ball.

5 Discussion & Conclusion

This study started out with the goal of developing a low-cost EMG interface to control a robotic hand. For this, the MyoWare EMG sensor and a 3D printed robotic hand from Open Bionics were used for the hardware part, and the Arduino IDE for the design of the code.

In the assembly of the system, the MyoWare sensor was connected to the internal board of the robotic hand through a 4-poles headphones jack cable, to supply the voltage for the sensor and to send the signal to the computer. Then, the robotic hand was connected to the computer with an USB cable, in order to read the sensor values and to control the actuators of the hand. Apart from the hardware assembly, a code was designed in the Arduino IDE to read the sensor signal and to control the hand, and the signal of the sensor was plotted both in Arduino and Processing programs.

After some testing with the sensor and the robotic hand, it was discovered that the sensor signal was not very unstable due to the noise caused probably for the headphone jack cable or the internal PCB of the hand, which would require future work on the study of this issue. However, the opening and closing gestures of the robotic hand were performed, controlling with the MyoWare sensor that was attached to one of the forearm muscles.

In summary, this study has achieved the main purpose of the project, demonstrating a development of a low-cost EMG controller for robotic hand which is capable of performing open and close gestures using an EMG based wearable sensor, and suggests the possibility of creating a low-cost upper limb prosthetic device for dexterous recovery of amputees [19-22].

Acknowledgements

This work was presented in coursework form in fulfilment of the requirements for the BEng in Robotics for the student Manex Ormazabal Arregi under the supervision of E.L. Secco from the Robotics Lab, School of Mathematics, Computer Science and Engineering, Liverpool Hope University.

References

1. Access Prosthetics. (2019). 15 Limb Loss Statistics that May Surprise You - Access Prosthetics. [online] Available at: https://accessprosthetics.com/15-limb-loss-statistics-may-surprise/ [Accessed 4 Jul. 2019].
2. Clement, R., Bugler, K. and Oliver, C., 2011. Bionic prosthetic hands: A review of present technology and future aspirations. The Surgeon, 9(6), pp.336-340.
3. Annemarie E. Orr., 2020. Orthotics and Prosthetics in Rehabilitation. 4th ed. pp.784-797.
4. Vujaklija, I. and Farina, D., 2018. 3D printed upper limb prosthetics. Expert Review of Medical Devices, [online] 15(7), pp.505-512. Available at: <https://doi.org/10.1080/17434440.2018.1494568>.
5. Fuentes, L., 2021. The Most Common 3D Printed Prosthetics in 2021. [online] All3DP. Available at: <https://all3dp.com/2/the-most-common-3d-printed-prosthetics/> [Accessed 3 April 2021].
6. Mereu, F., Leone, F., Gentile, C., Cordella, F., Gruppioni, E. and Zollo, L., 2021. Control Strategies and Performance Assessment of Upper-Limb TMR Prostheses: A Review. Sensors, [online] 21(6), p.1953. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8000641/> [Accessed 5 April 2021].
7. Manero, A., Smith, P., Sparkman, J., Dombrowski, M., Courbin, D., Kester, A., Womack, I. and Chi, A., 2019. Implementation of 3D Printing Technology in the Field of Prosthetics: Past, Present, and Future. International Journal of Environmental Research and Public Health, [online] 16(9), p.1641. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6540178/> [Accessed 5 April 2021].
8. Geethanjali, P., 2016. Myoelectric control of prosthetic hands: state-of-the-art review. Medical Devices: Evidence and Research, Volume 9, pp.247-255.
9. Open Bionics. 2016. Ada V1.1 Assembly Instructions — Open Bionics. [online] Available at: <https://openbionicslabs.com/obtutorials/ada-v1-assembly> [Accessed 11 April 2021].
10. Open Bionics. 2016. Muscle Control V1.2 — Open Bionics. [online] Available at: <https://openbionicslabs.com/obtutorials/muscle-control-v1-2> [Accessed 11 April 2021].
11. Toro, S., Santos-Cuadros, S., Olmeda, E., Álvarez-Caldas, C., Díaz, V. and San Román, J., 2019. Is the Use of a Low-Cost sEMG Sensor Valid to Measure Muscle Fatigue?.Sensors, 19(14), p.3204.
12. Prakash, A., Sharma, S. and Sharma, N., 2019. A compact-sized surface EMG sensor for myoelectric hand prosthesis. Biomedical Engineering Letters, 9(4), pp.467-479.
13. AdvancerTechnologies, 2015. MyoWare™ Muscle Sensor (AT-04-001) DATASHEET. [online] Available at: <https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MyowareUserManualAT-04-001.pdf> [Accessed 18 April 2021].
14. theoryCIRCUIT - Do It Yourself Electronics Projects. n.d. Myoware Muscle Sensor Interfacing with Arduino. [online] Available at: <https://theorycircuit.com/myoware-muscle-sensor-interfacing-arduino/> [Accessed 12 April 2021].

15. Aqeel, A., 2018. Introduction to Arduino IDE - The Engineering Projects. [online] The Engineering Projects. Available at: <https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html> [Accessed 10 April 2021].

16. E.L. Secco, D. McHugh, D. Reid, A.K. Nagar, Development of an Intuitive EMG Interface for Multi-Dexterous Robotic Hand, Wireless Mobile Communication and Healthcare, Chapter 16, Springer, 2019 – DOI: 10.1007/978-3-030-49289_16

17. Eagan, J., n.d. Introduction to Processing (Java). [online] Perso.telecom-paristech.fr. Available at: <https://perso.telecom-paristech.fr/eagan/class/ces-ds/labs/lab1-java> [Accessed 12 April 2021].

18. Processing.org. n.d. Processing.org. [online] Available at: <https://processing.org/> [Accessed 7 April 2021].

19. E.L. Secco, J. Scilio, Development of a symbiotic GUI for Robotic and Prosthetic Hand, Intelligent Systems Conference (IntelliSys) 2020, Amsterdam, The Netherlands

20. Howard AM, Secco EL, A low-cost Human-Robot Interface for the Motion Planning of Robotic Hands, Intelligent Systems Conference (IntelliSys) 2021, Advances in Intelligent Systems and Computing, Springer, *accepted*

21. A.T. Maereg, Y. Lou, E.L. Secco, R. King, Hand Gesture Recognition Based on Near-Infrared Sensing Wristband, Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2020), 110-117, 2020 - ISBN: 978-989-758-402-2 – DOI: 10.5220/0008909401100117

22. E.L. Secco, A.T Maereg, A Wearable Exoskeleton for Hand Kinesthetic Feedback in Virtual Reality, Wireless Mobile Communication and Healthcare, Chapter 15, Springer – DOI: 10.1007/978-3-030-49289_15