Development of an Intuitive EMG Interface for Multi-dexterous Robotic Hand

Emanuele Lindo Secco(✉), Daniel McHugh, David Reid, and Atulya Kumar Nagar

Robotic Laboratory, Department of Mathematics and Computer Science, Liverpool Hope University, Liverpool L16 9JD, UK
{seccoe,13005235,reidd,nagara}@hope.ac.uk

Abstract. This paper presents an integrated EMG-based system for controlling a 5-fingers robotic hand: the system combines two commercial products, namely an Open-Bionics Hand and a wearable MYO controller in a new fashion where the end-user can control different postural grasping through an intuitive menu, cycling among different pre-set grasping configurations. According to preliminary tests, such a solution may represent an interesting novelty for a user-centered experience of patients with an upper limb prosthetic device.

Keywords: Smart prosthetics · Human-centered healthcare · Electromyography

1 Introduction

Nowadays there is a huge demand for dexterous and user-friendly devices for robotic and prosthetic manipulation. Human subjects, who has suffered from the loss of a limb, need to recover their daily life ability to manipulate and interact with objects and the other human beings. On the other side, current humanoid robots have increased the capability to interact with us and, therefore, to perform actions which belong to our nature [1–3, 14].

Unfortunately, even if many devices have been developed with high movement capability and dexterity, a lack of simplicity on the usage of the interface between such devices and the end-user is still occurring.

Here we are trying to show a simple and low-cost architecture where some of these drawbacks are overcome. This paper looks at the relationship between a prosthetic hand and an *ElectroMyoGraphy* (EMG) sensor to simulate different grasp types. These ones allow the user to interact with different objects in the real world through a robotic hand [4–6]. The approach is based on the integration of an Arduino Mega and an OpenBionics hand [7, 8]. The aim is to propose a novel and improved robotic and prosthetic system that can closely simulate a real limb via EMG which is used to control the robotic hand [9].

2 Materials and Methods

2.1 Hardware

In this section of the paper details of the hardware are reported. The main reasons of specific hardware selection are also mentioned.

The Open Bionics Robotic Hand
The robotic hand which has been used in this project is a 3D printed hand from Open Bionics (Fig. 1): each part of the hand is printed and assembled together to integrate the motion capability; the 4 fingers and the thumb in the hand can reach different positions allowing the manipulator to perform different types of grasping. This is performed through linear actuators, which are installed in the palm of the hand. Each finger and thumb are connected to a single actuator via a fishing wire.

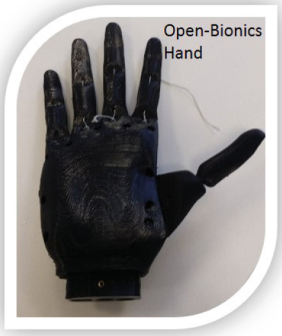


Fig. 1. The 5-finger Open-Bionics robotic hand.

The hand's task is to replicate different grasping configurations when a human input is given. The reason why such a device was chosen is because of how easy it is to interface and control the device itself: all you need to provide is the power supply and then to plug in the hand's controller into a computer through its USB cable: then the user may upload code to an Arduino-compatible controller which is embedded within the hand.

The MYO Band
MyoBand is a wearable device from Thalamic Labs which allows the end-user to collect electrical muscle activity or ElectroMyoGraphic signals (EMG) – Fig. 2. EMG are used in gesture control and many other applications, such as, for example, to develop a MyoWare muscle sensor. This device allows receiving 8 amplified and rectified EMG signals from the human forearm muscles: the larger is the amount of EMG signals, the higher is the information about the human gesture that may be collected by the sensor [10, 11].

When the device is worn on the arm, it will start to read the muscle electrical activity; data are then sent over a wireless Bluetooth protocol to a dongle that will be inserted

Fig. 2. The 5-finger Open-Bionics robotic hand.

into a PC. The PC reads the transmitted data and perform the action that relates to what the users are intended to do with their arm. The reason why this device was selected is because it is a wearable well integrated system, it is easy to be interfaced given the availability of software libraries and, moreover, it can get accurate readings.

In this project the Myoband will read the electrical activity of the user, transmit that data to a PC; this latter one will communicate to the robotic hand what to do, namely a proper type of grasping configuration, depending on which action and gesture the user has performed.

2.2 Software

In this section the software that has been used in the project is detailed.

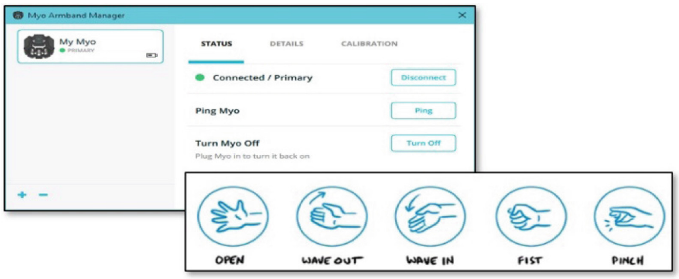
The MyoConnect

MyoConnect is the Thalmic Labs software interface which allow the user to perform gesture control vs. a PC operative system: it basically allows to use your PC without having to manipulate the mouse or keyboard. The interface also allows to perform a calibration procedure in order to use the MyoBand irrespective of the orientation of the device when you wear it. Among all the available features of the software, the typical MyoConnect mask is reported in Fig. 3.

MyoConnect can be conceived as the hub to the MyoBand, since all the preliminary information can be accessed from this software. Basic commands can be sent to the wearable device; a “ping” will vibrate the band if it is wirelessly connected to the software; this also allows the user to perform basic tasks and have a direct understanding of how the band and the software work together.

The Myo system inherently embeds a human gesture classifier. 6 gesture can be recognised such as (see also Fig. 3):

- *Fist*
- *Rest*



Plug Myo in to turn it back on

Fig. 3. The MYO Arm bracelet

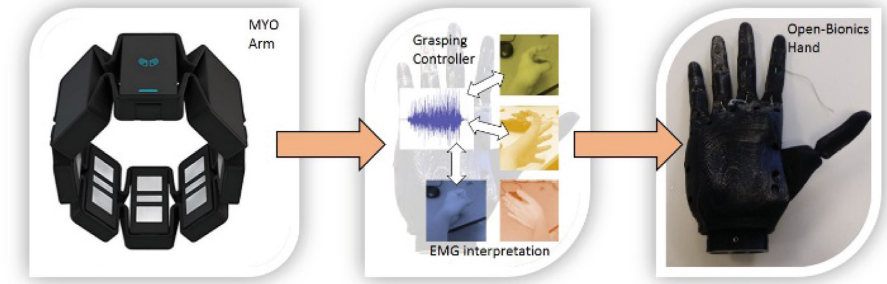


Fig. 4. The hardware architecture with the MyoBand and the open bionics robotic hand

- *Wave in*
- *Wave out*
- *Double tap*
- *Finger spread (i.e. Open Hand)*

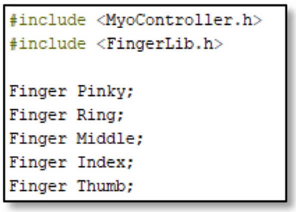
Clearly custom gestures may be designed by using the SDK of the device, however – at this stage – this project is going to make use of these 6 available classes.

MyoConnect is also used as a bridge to talk to another software, called MyoDuino, that is used to send data to the robotic hand. Meanwhile the MyoConnect is also used in order for the band to send data to the PC and then be able to understand what is happening on the human-user side. An overall overview of the system is also shown in Fig. 4.

The Arduino IDE

Arduino Software is an open-source Integrated Development Environment (IDE), which uses C/C++ programming language. It provides the user with the ability to write code known as a sketch and in turn the ability to upload the sketch within an Arduino board.

An Arduino sketch is split into four different sections, where Sect. 1 is where the necessary libraries are included: in this project a *FingerLib* library has been embedded within the software. This library is used by the robotic hand in order to move the actuators



```
#include <MyoController.h>
#include <FingerLib.h>

Finger Pinky;
Finger Ring;
Finger Middle;
Finger Index;
Finger Thumb;
```

Fig. 5. The mapping of the 5-fingers robotic hand

and then the fingers. A *MyoController* is also used to allow the MyoBand and the Open Bionics hand to work in synergy. Section 2 is the setup section where the initial values of the variables are set, such as the starting positions of the servos. Also, in the set up all the pin modes are declared. The 3rd section is the loop section, which houses the main body of the code. The 4th and final section are where the functions of the program are declared.

The MyoDuino Software

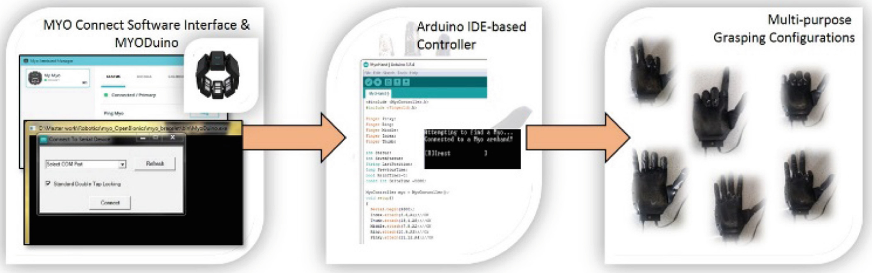
MyoDuino is a third-party software that can be downloaded on GitHub: this element interfaces the two devices, namely the Myoband - i.e. its Bluetooth I/O - with the Arduino board – i.e. its Serial I/O. MyoDuino allows the user to see what is happening on the Myoband side as a visual display can be seen once the wearable system has been detected.

3 Design of the Interface

An Arduino IDE is used to develop the interface, due to the fact that the robotic hand embeds an Arduino-compatible controller board. The main code also needs to incorporate the *FingerLib* library, which allows each finger and thumb to be assigned a label name and to be actuated. This element makes the design of the grasping configuration easier: all that is needed is (a) to call the finger that you want to move and (b) then to write the position the finger has to reach. The finger labelling as it was performed in the project is reported within Fig. 5.

After setting the labelling, the further step is to make the Myoband able to set some actions into the hand device: this task is performed by using another library in Arduino IDE called *Myocontroller* which allowed the Myoband and the Arduino to dialogue each other: this is done by assigning the library to a variable such as *myo* and then using the functions inside the library to extract the classification output of the human movement from the Myoband itself. In particular, one of the attribute or function that was used in the project is the *myo.getcurrentposistion*. The function returns the current output of the Myoband allowing the Arduino board (i) to recognise a certain configuration of the end-user hand, such as, for example, the *fist* or *rest* configuration and therefore (ii) to trigger the robotic hand. An overview of this process is shown in Fig. 6.

In order to connect the Myoband and the Arduino a third party software called *MyoDuino* is used. MyoDuino acts as the bridge that allows communication between the two devices and it runs alongside the Arduino IDE: it connects the Myoband to the

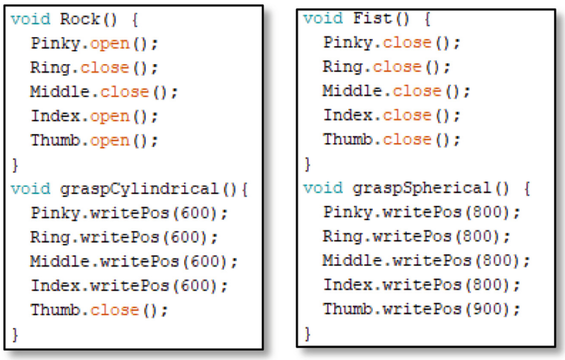


```
Attempting to find a Myo...
Connected to a Myo armband!

[R]rest        ]
```

Fig. 6. Software architecture and functional diagram of the MYO-OpenBionics interface

MyoDuino via the Bluetooth dongle that comes with the Myoband and the MyoDuino to the hand via a COM port. Given such an interaction between the parts, a code to map the signals vs the hand movements can be designed. Figure 7 displays some of these movements that were implemented. Such a code should allow the hand to follow the human movements such that the robotic device will mirror the movements of the user. This process inherently requires a continuous stream of data from the Myoband to the hand which collides with the limited amount of memory of the Arduino board, unless the data that was being sent would be delayed (which would cause the unacceptable behaviour of having the hand performing the movement with a delay).



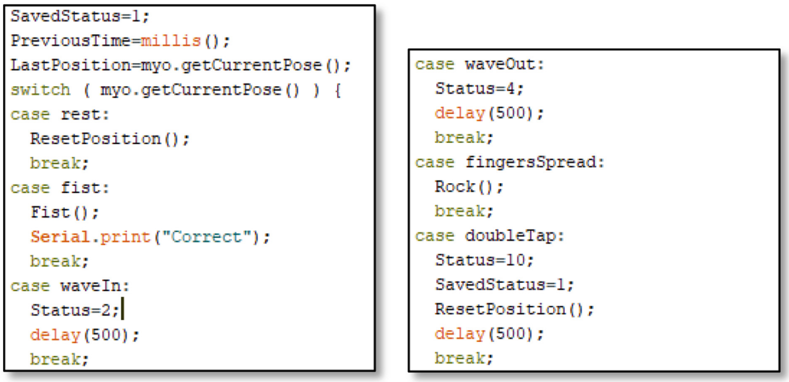
```
void Rock() {
  Pinky.open();
  Ring.close();
  Middle.close();
  Index.open();
  Thumb.open();
}
void graspCylindrical(){
  Pinky.writePos(600);
  Ring.writePos(600);
  Middle.writePos(600);
  Index.writePos(600);
  Thumb.close();
}
```

```
void Fist() {
  Pinky.close();
  Ring.close();
  Middle.close();
  Index.close();
  Thumb.close();
}
void graspSpherical() {
  Pinky.writePos(800);
  Ring.writePos(800);
  Middle.writePos(800);
  Index.writePos(800);
  Thumb.writePos(900);
}
```

Fig. 7. A set of available grasping configuration that were coded into the Arduino board

To bypass this issue a *SavedStatus* was designed within the Arduino IDE, which would allow the user to perform more movement using the same gestures (Fig. 8): namely, all the band gestures are recognised in every state, then each label will call its own function giving the robotic hand an unlimited set of movements.

The action will change based on what *savedstatus* the user is in: for example, a *fist* status will move the hand into the homologous fist configuration, whereas other savedstatus will perform different action allowing for more movements of the robotic device. Similarly, to switch to a next status, the user have to perform a *waveIn*; to go



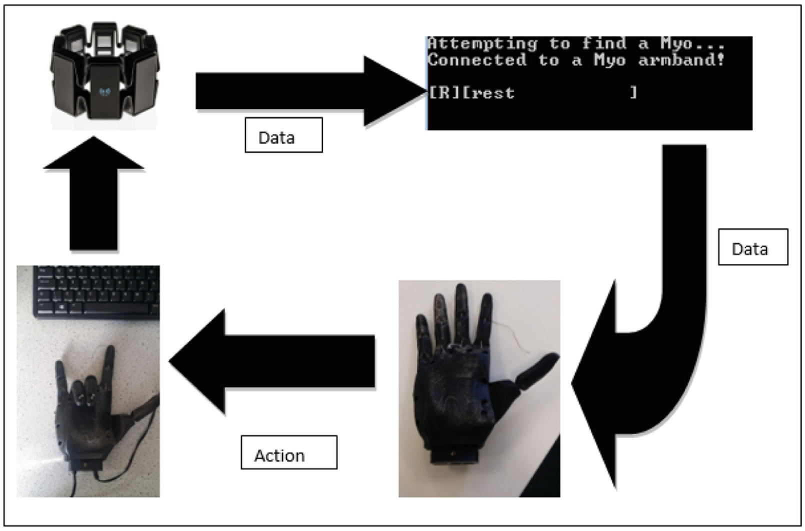
```
SavedStatus=1;
PreviousTime=millis();
LastPosition=myo.getCurrentPose();
switch ( myo.getCurrentPose() ) {
case rest:
  ResetPosition();
  break;
case fist:
  Fist();
  Serial.print("Correct");
  break;
case waveIn:
  Status=2;
  delay(500);
  break;
```

```
case waveOut:
  Status=4;
  delay(500);
  break;
case fingersSpread:
  Rock();
  break;
case doubleTap:
  Status=10;
  SavedStatus=1;
  ResetPosition();
  delay(500);
  break;
```

Fig. 8. An example of *SavedStatus* setting

back to the last savedstatus the user has to perform a *waveOut*; finally, to reset the device into the home position the user needs to perform a *double tap*.

An overall view of this architecture is reported within the Fig. 9. The complete system is here shown: the MyoBand sends the information to the MyoDuino which in turn sends a command to the hand. This latter one will perform the action and loop back to the Myoband to get the new end-user command.



```
Attempting to find a Myo...
Connected to a Myo armband!
[R][rest            ]
```

Fig. 9. Overview of the MyoBand and Open Bionics hand controller

4 Results

In this section of the paper, the results of some preliminary laboratory trials are reported. These results are shown in block diagrams to make clear which *savedstatus* were associated with which movements. Only few status are reported as representative of the overall set of the available status.

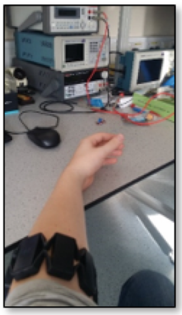
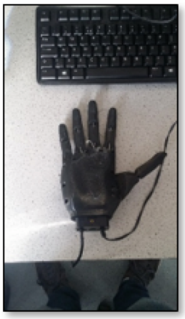
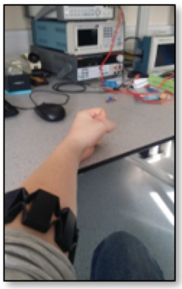
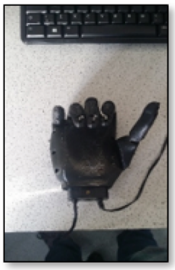
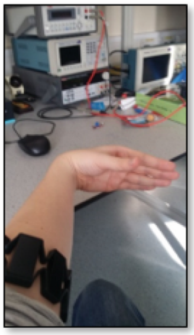
Hand Gesture Myo Band	**Robotic Hand** Open Bionics
 Rest	 Rest position
 Fist	 Fist position
 Wave in	➡ Next State

Fig. 10. Overview of the *SavedStatus 1*

Figure 10 shows the movement in *SavedStatus 1*: (i) waving in will change to the next savedstatus and (ii) waving out will go back to the last savedstatus, whereas (iii) double tap is used in case of any errors that happen while the hand is working.

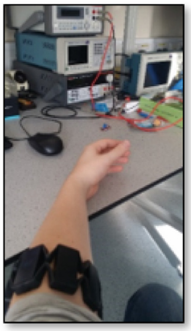
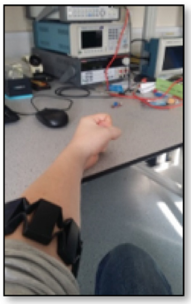
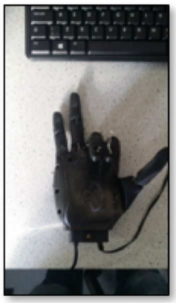
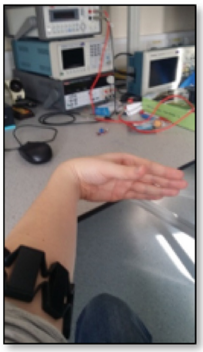
Hand Gesture Myo Band	**Robotic Hand** Open Bionics
 Rest	Rest position
 Fist	 Grasp Tripod position
 Wave in	Next State

Fig. 11. Overview of the *SavedStatus 2*

This latter command will just set the hand in its reset or home position but the user will still be in the same savedstatus. A similar approach is followed for the *SavedStatus 2*, which is illustrated within the Fig. 11.

A set of 4 different status has been designed: fopr each one of these four stata, different configurations are available, according to the chosen one.

Thanks to this set of status, a menu with different movements allows the hand to switch and move in relation to the gesture that the user has done.

5 Conclusions and Future Works

A low-cost system combining EMG signal detection and classification with a 5 fingers robotic hand has been presented. The proposed system allows controlling the robotic hand in a set of various grasping configuration. Thanks to the choice of the design and experimental set-up, a variety of configuration is also available and easily adaptable [12–14]. Even if the validation has been only performed with some preliminary experiments, results are quite encouraging in view of developing such a system on a real prosthetic application where the patient may benefit of a low-cost system for daily life activity.

The proposed system embeds open-source software, namely the Arduino IDE software interface, and therefore it represents the possibility of developing integrated and human-centered prosthetic system with small budgets.

This approach could go beyond the present scope as the used hardware could be modified to look at many different uses. For example, the robotic hand may be combined with a Brain Computer Interface such as - when the user wears the device interface – brain activity could condition the hand, which would finally result in some movement [15].

Acknowledgements. This work was presented in project & coursework form in fulfilment of the requirements for the MSc in Robotics Engineering for the student Daniel McHugh under the supervision of E.L. Secco from the Robotics Laboratory, Department of Mathematics & Computer Science, Liverpool Hope University.

References

1. Secco, E.L., Moutschen, C.: A soft anthropomorphic & tactile fingertip for low-cost prosthetic & robotic applications. Liverpool: EAI Endorsed Trans. Pervasive Health Technol. **1**, 10 (2018)
2. Secco, E.L., Magenes, G.: Bio-mimetic finger - human like morphology, control & motion planning for intelligent robot & prosthesis. In: Mobile Robotics, Moving Intelligence, pp. 325–348 (2006)
3. Matrone, C., Cipriani, C., Secco, E.L., et al.: A biomimetic approach based on principal components analysis for multi-D.O.F. prosthetic hand control. In: Corner Workshop (2009)
4. Hand Gesture Controlled Robot Using Arduino: Int. J. Recent Trends Eng. Res. **4**(4), 336–342 (2018)
5. Llop-Harillo, I., Pérez-González, A.: System for the experimental evaluation of anthropomorphic hands. Application to a new 3D-printed prosthetic hand prototype. Int. Biomech. **4**(2), 50–59 (2017)

6. Explore & Learn. MyoBand (2019). https://learn.adafruit.com/myo-armband-teardown/inside-myo. Accessed 1 Mar 2019
7. Mounika, M., Phanisankar, B., Manoj, M.: Design & analysis of prosthetic hand with EMG technology in 3-D printing machine. Int. J. Curr. Eng. Technol. **7**(1), 115, 119 (2019)
8. Pizzolato, S., Tagliapietra, L., Cognolato, M., Reggiani, M., Müller, H., Atzori, M.: Comparison of six electromyography acquisition setups on hand movement classification tasks. PLOS ONE **12**(10), e0186132 (2017). https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0186132
9. Shibata, T.: A study on our new prosthetic hand control method using a low-cost sEMG device. In: The Proceedings of JSME Annual Conference on Robotics and Mechatronics (Robomec), pp. 2A2–E01 (2018)
10. Welcome to Myo Support. Getting starting with Myo on Windows (2019). https://support.getmyo.com/hc/en-us/articles/202657596-Getting-starting-with-Myo-on-Windows. Accessed 4 Mar 2019
11. GitHub. CurrentlyAWey/Arnold-Palm-er (2019). https://github.com/CurrentlyAWey/Arnold-Palm-er. Accessed 4 Mar 2019
12. Secco, E.L., Magenes, G.A.: Feedforward neural network controlling a 3 D.O.F. finger. 5th International Conference on Cognitive & Neural Systems, 2001
13. Secco, E.L., Magenes, G.: Control of a 3 DOF artificial finger by a multi-layer perceptron. In: International Federation of Medical and Biological Engineering (IFMBE) Proceedings Medicon, pp. 927–930 (2001)
14. Magenes, G., Secco, E.: Teaching a robot with human natural movements. In: Pascolo, P.B. (ed.) Biomechanics and Sports. CCL, vol. 473, pp. 135–145. Springer, Vienna (2004). https://doi.org/10.1007/978-3-7091-2760-5_15
15. Elstob, D., Secco, E.L.: A low cost EEG based BCI prosthetic using motor imagery. Int. J. Inf. Technol. Converg. Serv. **6**(1), 23–36 (2016)