Individual Robotic Arms Manipulator Control Employing Electromyographic Signals Acquired by Myo Armbands

Paula Useche Murillo
Mechatronics Student, Department of Mechatronics Engineering,
Militar Nueva Granada University, Bogotá, Colombia.

Robinson Jimenez Moreno
Assistant Professor, Technology on Electronics and Communication,
Militar Nueva Granada University, Bogotá, Colombia.

Oscar F Avilés S
Titular Professor, Department of Mechatronics Engineering,
Militar Nueva Granada University, Bogotá, Colombia.

Abstract
This paper presents the control of two manipulating arms using EMG signals captured with a gesture control and motion control device, Myo Armband. It was used an electromyographic signal recognition obtained by Myo in order to control the degrees of freedom of two different manipulating arms to achieve a work altogether. First, the Myo armband was analyzed to acquire the control signals, after that robotic arms were characterized for associate its freedom degrees to the electromyographic signal moves. Finally, a test scenario was implemented to validate the robotic arms control through the moves of the user arms using the Myo’s.

Keywords: electromyographic signals, Myo, manipulator, control, control gestures, EMG, gesture, Arduino.

INTRODUCTION
During the last decade, great efforts has been taken to simplify the human-robot interfaces and make them more intuitive and natural, supporting on the signal processing directly from the body. In order to accomplish an interface with these features, an accurate body signal interpretation is required. Several researches have shown that electromyographic signals (EMG) are feasible for this application, insomuch as provides very accurate information regarding limb movements [1].

The use of electromyographic signals has been spread out by the medicine field, implementing it on prosthesis control for amputee like in [2] [3] [4], for patients who have suffered from limb paralysis like tetraplegia in [5], and for limb locomotion rehabilitation systems like shown in [6] [7] [8] [9], all in order the necessity of assist and restore the human locomotion, which is an esential factor for most daily activities [10].

Another researches have implemented EMG signals to manipulator control by tracking arm motions, using different analysis techniques to choose the best, from achieved results [11]. In another case, it was tried to reduce number of electrodes with the purpose of simplify the mobile robot navigation system like shown in [12].

As a consequence of the necessity of process and use the arm EMG signals to control different kind of robots, has begun to implement a wearable gesture control which ease the EMG signal detection and processing, the Myo Armband. In [13] can be seen an application focused on a three-wheel omnidirectional robot navigation control, using a muscle gesture-computer interface employing a wearable Myo. In [14] was developed a method to tele-control a robot arm in order to imitate human writing skills. It was used the haptic device Sensable Omni to control the robot motion, and the Myo to record the arm surface electromyographic (sEMG) signals and processing them to change the thickness of the draw lines, according to the user exerted force.

SIGNAL CAPTURE
Operation of Myo Armband
Is a wearable armband that uses biosensors to measure electromyographic signals produced on forearm muscles to recognize the hand motions and process the signals to send them by bluetooth, like control words, to another devices [15] [16], furthermore incorporate an accelerometer, gyroscope and magnetometer [17]. Requires direct skin contact, so cannot be use onto clothes [18] [19].

The device is compatible with Mac, iOS, Windows and Android, is powered by a rechargeable liithium battery and uses an ARM Cortex M4 processor [17]. Like in Figure 1, the device has eight segments of expandable casing, connected using stretchable material that allows adjust the Myo Armband to the user-arm dimensions, a USB charging port, the logo LED that shows the sync state of the Myo, which pulses when Myo is not synced and solid when is synced and located on the arm, and a status LED which shows the Myo state according to a color code shown on Figure 2 [20].
According to [20], Myo provides two kinds of data to an application: the spatial data and the gestural data. The spacial data informs about the orientation and movement of the user’s arm:

- The orientation defines the direction which Myo points. It is a data provided as a quaternion, and can be converted in a rotational matrix or Euler angles.
- The movement, or acceleration vector, represents the Myo acceleration at any given time. It is a data that is delivered as a three-dimensional vector.

On the other hand, the gestural data indicates the user-hand motion. It is a data delivered as one of several preset poses which represents a specific user hand configuration. Myo gives information about in which arm is being implemented, if it is placed on the arm, and which way is oriented, looking at the wrist or elbow. Also is possible to causes the Myo to vibrate, for a touch and sound feedback.

**Signal analysis**

According to [21], the armband uses commercial EMG sensor, WEMG-8 of Laxtha Co., equipped with 2.4-GHz wireless transceiver, a bandpass analog filter embedded in EMG electrode unit, with a frequency bandwidth of 13 and 430 Hz and a sampling rate of 800 Hz. The raw EMG signal is noisy like shown in Figure 3, so that requires a rectification, amplification and filtration stage. In order to obtain relevant information, the device has a bandpass filter with a frequency bandwidth of 0.01–2 Hz, which consists of second-order lowpass filter and highpass filter. The cutoff frequency of the lowpass filter was determined based on the maximum work frequency of the motion hand gesture, which is 2 Hz.

The device has, also, a notch filter to reduce 60 Hz noise generated by alternating current of wall electricity. The electrodes were located on a muscle specific group, like shown in Figure 4, so at the moment of process and evaluate EMG signals, the movement was determined by the sensor with the largest measurement value. For this reason, the detected movements by the device are those which depends of a muscle specific group, like the movements shown in Figure 5, so as to select the one of the eight sensors and then do the recognition.

However, the location of the muscles is different for each person, for this reason is necessary a calibration process in
order to improve gesture recognition. Using Fuzzy Logic and the gestures of Figure 5 like control rules, the gesture recognition was made by the relation between each sensor measure values like shown in Table 1, where the recorded magnitude for each sensor depends of the user calibration

Table 1: Rule Base of Fuzzy Logic [21]

<table>
<thead>
<tr>
<th>Gesture</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrist flexion</td>
<td>Mid</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Mid</td>
<td>Mid</td>
</tr>
<tr>
<td>Wrist extension</td>
<td>Mid</td>
<td>Low</td>
<td>Low</td>
<td>Mid</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Finger flexion</td>
<td>Mid</td>
<td>High</td>
<td>Mid</td>
<td>Low</td>
<td>N/A</td>
<td>Low</td>
</tr>
<tr>
<td>Finger extension</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Radial deviation</td>
<td>Mid</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>N/A</td>
<td>Mid</td>
</tr>
<tr>
<td>Ulnar deviation</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Mid</td>
<td>Low</td>
</tr>
</tbody>
</table>

N/A: Not applied

An example of two fuzzy rules can be seen on eq (1) and eq (2), where \(l, m, h\) corresponds to Low, Mid and High levels respectively, the subindices correspond to sensor number of the Table 1 and the subscript of the equation corresponds to the basic rule number of the Table 1, being 1 for “Wrist flexion”, and 5 for “Radial deviation”.

\[
\mu_1(k) = m_1(k)h_2(k)h_3(k)l_4(k)m_5(k)m_6(k) \quad (1)
\]

\[
\mu_2(k) = m_1(k)l_2(k)l_3(k)h_4(k)m_6(k) \quad (2)
\]

In [21], after experiment with multiple subjects, it was verified that this recognition method ensures a success rate higher than 90%, independent of sex and age

Signal recorded from Myo Armband to Arduino

The Myo Armband connects by a Bluetooth 4.0 Low energy USB adapter to the computer [20], and can be used to control different computer functions, from movement and mouse actions, to the seamlessly control presentation software, touch-free. Myo works reading electromyographic signals of the forearm with eight EMG sensors, and complements reading with a 3 axis accelerometer, gyroscope and magnetometer [22].

For the application developed was necessary to connect the Myo with an Arduino to obtain the electromyographic signals and turn them into manipulator control movements. The gestures captured by Myo are represented in the Figure 6.

**Figure 6: Control gestures detected by Myo Armband [22]**

The “MyoController” library was downloaded from the Myo Market in order to connect both devices. The library makes easier the reading, obtaining and classification of the gestures recognized by Myo, and creates a communication channel with the Arduino, employing the “myo.getCurrentPose()” function to gesture record, and a switch statement for the activation of certain Arduino outputs, like shown in Figure 7.

According to the gesture recognized by the Myo: Rest, Fist, Wave In, Wave Out, Fingir Spread y Double Tap; the program enters on one of the movement possibilities and executes an action, in the case of Figure 7, turn on and turn off LEDs.

The logic of the Figure 7 was used to develop the control program, modifying the execution actions in each movement.

Reading of two Myo Armband by Arduino

Two Myo controllers were used to make an independent control of two manipulator arms, one Myo for each arm, where each device gesture generates specific control signals for the corresponding manipulator. To achieve the connection, two Arduinos were used, so each Arduino connects with one Myo in an independent way, and exe-cutes the signal control depending of the acknowledge gesture.

The “MyoController” library allows the connection between Myo and Arduino through serial port, being that the library includes an executable responsible of the reading and recognition of Myo gestures, and send the results to the Arduino. As the library is designed for the connection with just one Myo, it was necessary to employ two Arduinos for the application, one per each device.

The control logic was the same for both Arduinos, just changing the servomotors displacement value from the previously characterization developed, and the number of degrees of freedom according to the manipulators design.

THE MANIPULATORS

In the Figure 8, can be seen the manipulators and Arduinos used in the application, also the location of each servomotor, the Arduinos, and the numbering of each motor. On the Table 2 was specified the type of Arduino employed and the numbering of each servomotor for the case of the large manipulator, and on the Table 3 were specified those of small manipulator.
To control the manipulators’ movement, it was necessary characterize each servomotor in order to know the zero position and the necessary control value to arrive at a specific position, for that, an Arduino program was carried out to control the position of a servomotor with a potentiometer, and send the control code to be able to view it and complete the characterization.

The angle and code values of each servomotor of the large manipulator were organized in the Table 4. On top is indicated the degree of inclination respect to floor, and in the left column is specified the number of servomotors, where one (1) is at the base and seven (7) which controls the gripper.

Referent to the degrees of freedom of the gripper, just the open and close, and the horizontal and vertical states were considered (respect to floor).

Because of the manipulator structure, the fourth servomotor do not arrive the zero position, but arrive a 225º inclination, like shown on the table, being the servomotor sending data equal to 180.

Table 4: Characterization of the large manipulator

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Large Manipulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>Arduino Mega</td>
</tr>
<tr>
<td>G1</td>
<td>Servomotor 1</td>
</tr>
<tr>
<td>G2</td>
<td>Servomotor 2</td>
</tr>
<tr>
<td>G3</td>
<td>Servomotor 3</td>
</tr>
<tr>
<td>G4</td>
<td>Servomotor 4</td>
</tr>
<tr>
<td>G5</td>
<td>Servomotor 5</td>
</tr>
<tr>
<td>G6</td>
<td>Servomotor 6</td>
</tr>
<tr>
<td>G7</td>
<td>Servomotor 7</td>
</tr>
</tbody>
</table>

The zero positions respect to floor of each degree of freedom of the manipulator can be seen in Figure 9, from which were obtained the angles of the Table 4.

In Figure 9 is presented the location of each of the manipulator links when the respective servomotor is on 0º. In BASE (1) is shown the base B, turned right, as indicated by the arrow that marks the route of the motor from 0º to 180º, and also is shown the gripper vertical position, respect to floor.

In LINK 1 (2) is shown the link one E1, located horizontally to the right for 0º. In LINK 2 (4) is shown the link two E2, tilted almost 45º to the right, because its structure does not let it arrive to 0º respect to floor. In LINK 3 (5) is shown the link three E3, located horizontally to the right for 0º.

In OPEN GRIPPER (3) and CLOSE GRIPPER (6), are shown both gripper positions employed in the application, the gripper opened (3) and the gripper closed (6).

Finally, in the left side of the Figure 9 is shown an image of both manipulators in almost completely stretched positions.
The zero positions respect to floor of each small-manipulator servomotor, were taken under the same reference of the large manipulator. Later, the small manipulator was characterized in the same way of the large manipulator, getting the results of the Table 5.

Table 5: Characterization of the small manipulator

<table>
<thead>
<tr>
<th>SERVOMOTORS</th>
<th>OPEN</th>
<th>CLOSE</th>
<th>90°</th>
<th>45°</th>
<th>0°</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Rotate the whole arm</td>
<td>0</td>
<td>71</td>
<td>180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Up-Down Link 1</td>
<td>0</td>
<td>100</td>
<td>140</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>3 Up-Down Clamp</td>
<td>188</td>
<td>81</td>
<td>39</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4 Open-Close Clamp</td>
<td>47</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The small manipulator consist of only four servomotors, so that has less degrees of freedom to control than the large manipulator. The control values of each servomotor rotation degrees were obtained by the characterization, so as to a posterior control.

CONTROL

Control gestures

The Myo recognizes six different gestures, which are presented in the Table 6 along with the gesture description, and the control type realized in the application.

Table 6: Control gestures

<table>
<thead>
<tr>
<th>Gesture Name</th>
<th>Gesture Type</th>
<th>Gesture</th>
<th>Control 1</th>
<th>Control 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST</td>
<td>Relax hand</td>
<td>Hold previous position</td>
<td>Hold previous position</td>
<td></td>
</tr>
<tr>
<td>FIST</td>
<td>Clenched fist</td>
<td>Close Clamp</td>
<td>Clamp Shrinks</td>
<td></td>
</tr>
<tr>
<td>WAVE IN</td>
<td>Hand tilted inwards</td>
<td>Manipulator Shrinks</td>
<td>Rotate to 0°</td>
<td></td>
</tr>
<tr>
<td>WAVE OUT</td>
<td>Hand tilted out</td>
<td>Manipulator Rises</td>
<td>Rotate to 180°</td>
<td></td>
</tr>
<tr>
<td>FINGER SPREAD</td>
<td>Extended hand</td>
<td>Open Clamp</td>
<td>Clamp Rises</td>
<td></td>
</tr>
<tr>
<td>DOUBLE TAP</td>
<td>Double tap between the thumb and the heart finger</td>
<td>Change of control type</td>
<td>Change of control type</td>
<td></td>
</tr>
</tbody>
</table>

A control command was assigned for each gesture, in such a way when the user wants to open or close the gripper, it has to be done the respective gesture for the desired movement, in order to the manipulator responds in the right way. For example, the application has to be on Control 1 and the user makes the Fist gesture to close the gripper, and the Finger Spread gesture to open it.

The application has two control types, in the first a specific gesture makes manipulator shrinks, and the same gesture, in the second control, makes manipulator rotates about the axis perpendicular to the ground, as in the case of Wave In gesture of the Table 5. The Double Tap gesture has to be done in order to change the control type, from Control 1 to Control 2, and vice versa.

Control logic

To control both manipulators with the recognized and sent gestures from Myo to Arduino, the initial position was appointed, and from it the servomotors angels were varied according to the user selected gesture. The initial position of each servomotor was 90° for both manipulators, leaving the robot arm fully extended up. The initial position is only achieved at the beginning of the program before any gesture is sent from Myo Controller, after the first control signal is sent by the user, the arm remains in the last position set, waiting the next order.

The “MoverServo” function was developed to control the motion speed of each servomotor by the manipulation of a delay time which retards the increase or decrease of the control code. When the manipulator is going to make a movement, the degrees of the servomotor are increased one by one unit, going from 0 to 1 in 10ms. The increase or decrease of the delay makes the manipulator moves faster or slower.

When a Finger Spread gesture in Control 2 is recognized, the manipulator raises the gripper, which means that just the gripper link moves in order to arrive at 0° respect to floor, like shown in Figure 9. The same gesture, in Control 1, sends the code corresponding to the opening of the gripper, and it opens.

When “Wave In” gesture is made in Control 1, the manipulator shrinks, it means that all servomotors in charge of moving links 1, 2 and 3 have to rotate in such a way that they approach to 180° position respect to floor, making the manipulator crouches down. In Control 2, the servomotor number 1 moves to rotate the manipulator respect to the axis perpendicular to the ground, in order to move it to the zero position of the base according to the Figure 9.

The another gestures work in the same way, but changing the movement direction.

In the code, the control gesture is responsible for indicating the rotation direction and for moving the respective motors, leaving those which do not belong to the movement in their last position. With two integer variables “pk” and “pk2”, the rotation direction is defined (towards 0° or towards 180° for each servomotor) and the actual code value, which defines the servomotor position, is increased or decreased depending on the “pk” value or “pk2” value, where the code is increased if the rotation variable is ’1’, and is decreased when is ’0’.

Because the codes that define the PWM for each servomotor are global variables, when any variable changes in any gesture, the value is saved for the entire program, allowing the storage of previous positions for each motor in order to continue with the movement since the last location, regardless of the gesture selected.

RESULTS

Then, a serie of images where can be seen the motions of the small manipulator according to the gesture control and the control type configured (Control 1 or Control 2 on Table 5), is presented.
In the Figure 10 is shown the Finger Spread gesture with Control 1, where the manipulator’s clamp opens and, subsequently, the Fist gesture with Control 1, where the manipulator’s clamp closes.

![Figure 10: Open clamp (left) and close clamp (right) of the small manipulator](image1)

In Figure 11 the Wave In gesture with Control 1 which makes the manipulator shrinks, and the Wave Out gesture with Control 1 which makes the manipulator rises, can be seen.

![Figure 11: Small manipulator shrinks (left) and rises (right)](image2)

In Figure 12 the Finger Spread gesture with Control 2 which makes the manipulator’s clamp rises, and the Fist gesture with Control 2 which makes the manipulator’s clamp shrinks, can be seen.

![Figure 12: Small manipulator’s clamp rises (left) and shrinks (right)](image3)

In Figure 13 the Wave In gesture with Control 2 which makes the manipulator rotates to 0º, and the Wave Out gesture with Control 2 which makes the manipulator rotates to 180º, can be seen.

![Figure 13: Small manipulator rotates to 0º (left) and to 180º (right)](image4)

In the following images, is shown the motion of both manipulators working simultaneously according to the control gesture made by each user’s hand, and the control type configured: Control 1 for large manipulator and Control 2 for small manipulator.

In Figure 14 the Wave Out gesture was made for the small manipulator, making the manipulator rotates to 180º, and the Wave In gesture was made for the large manipulator, making the manipulator shrinks. Then, the gestures were reversed, making the small manipulator rotates to 0º, and the large rises, like shown in Figure 14.

![Figure 14: Wave Out and Wave In (left), Wave In and Wave Out (right) for small and large manipulator, respectively](image5)

In Figure 15 the Finger Spread gesture was made for both manipulators, making the small manipulator clamp’s link rises, and the large manipulator opens the clamp. Later, the Fist gesture was made for both manipulators, making the small manipulator clamp’s link shrinks, and the large manipulator closes the clamp, like shown to the right of Figure 15.

![Figure 15: Rises and open the clamp (left), shrink and close the clamp (right) for small and large manipulator, respectively](image6)
In Figure 16 is shown the motion of both manipulators working simultaneously with the same control type, Control 1. The Finger Spread was made for both manipulator, making the clamp opens.

![Figure 16: Open the clamp for both manipulators](image)

In Figure 17 is shown the motion of both manipulators, with Control 1 for the small and Control 2 for the large. The Wave Out gesture was made to rises the small manipulator, and the Finger Spread gesture was made to rises the large manipulator clamp’s link.

![Figure 17: Rises the small manipulator and rises the large manipulator clamp’s link](image)

ANALYSIS

It was used a Myo Armband per each forearm, and they were connected independently with the respective Arduino, where Arduino One of the small manipulator was connected with the Myo of the left forearm, and the Arduino Mega of the large manipulator was connected with the Myo of the right forearm.

The first test was performed using the default gesture calibration of the Myo to the recognition of hand motions, nevertheless, this method only allowed the acceptable recognition of two gestures. The Wave Out, Double Tap and Wave In gestures were not detected easily, so the control of both manipulator was complicated.

In Figure 18 the error rate obtained during the default gesture calibration test, was summarized, where green section represents the right reading for each gesture, and the red section represents the wrong reading.

![Figure 18: Error rate obtained during the default gesture calibration test](image)

Like shown in Figure 18, the most difficult movements to recognize were Wave In and Wave Out, followed by Double Tap, which was not easily differentiated from relax position, or confused with another movement like Finger Spread. In case of Wave In and Wave Out, both movement were easily confused with Finger Spread or Fist, and was necessary repeat the gesture more pronounced in order to be able to identify it.

An average between the individual percents was also realized so as to calculate a wrong and right total percent of gesture recognition, obtaining results of almost 50% for the wrong recognition, so it was considered a non satisfactory recognition.

Because of the presented problems, it was necessary to create a personal calibration profile for each Myo, adjusting the gesture recognition to the user requirements. Using the calibration software of the Myo Connect, the program read the user gestures and saved them for recognition, generating a custom calibration profile, designed for a specific user.

After several tests, an improvement in gesture recognition was seen, achieving an optimum control with extremely low recognition errors, like shown in Figure 19.

![Figure 19: Error rate obtained during the custom gesture calibration test](image)

The identification errors arose mainly from incomplete realization of the gesture, like gently stretch the hand for Finger Spread gesture, or accidentally tilt in or tilt out the wrist during tests. Sometimes the Double Tap gesture, despite being adequately recognized, needed to be repeated, but it was not a significant problem because it was rarely presented.

The overall percentage improved considerably in comparison to default calibration, reaching a 98% of right recognition, versus 53% of the previous case.

Otherwise, a delay between one movement and the another was established of 10ms to regulate the serial data sending...
among Myo and Arduino, allowing a suitable communication midst both devices. However, because of the delay, the Arduino took a long time to recognize a gesture and starts the manipulator motion, in addition to affecting the gradual increase of servomotor angles, making their rotation slower than expected and slightly paused.

Then, while testing both manipulators working altogether, there were mechanic difficulties because of the large manipulator’s weight and its small support base. In spi-te of the soft motion of both manipulators, the large manipulator inclination made it fall. Additionally, the connection of servomotors with the power and the Arduino had several problems, insomuch as the wires were loosened in each movement, causing servo-motors ran out of power, or control signal loosened.

Besides, it was seen that the Myo simultaneous connection requires a greater effort from machine to support both serial communications and gesture recognitions, what increased the delays between each manipulator movement, so as to start and finish the rotation.

Finally, it was observed that the gradual increase of rotation angles of the manipulator allowed to obtain the enough speed to the robot control, so the delay times were omitted in both codes.

CONCLUSIONS

The control of two manipulator robots through EMG signals recorded and classified by Myo Armband, and the use of two Arduinos for the individual control of each manipulator, was achieved. The rotation speed of each servomotor was regulated using a one by one degree displacement, and delays between a movement and the next.

The need to create a costume calibration profile for an accurate gesture recognition for both arms, and the use of one Arduino for each manipulator because of the individual serial communication realized in each Myo, was evident.

An extensive control of the different degrees of freedom of the manipulators using just four gestures, was achieved, owing to the control type change produced with Double Tap gesture, which gives the possibility to augment the control types for future applications, according to the need of control a greater number of degrees of freedom.

It was observed that the slow and regulated rotation for each degree of freedom of the manipulators allows to make finer and more accurate movements, than direct and sizable rotations.

ACKNOWLEDGEMENTS

The authors thanks to the “Universidad Militar Nueva Granada” by the support in the process of develop of this work (ING1830).

REFERENCES


Robot-ics for Better and Safer Living (BRC), Salvador, 2014, pp. 1-6. doi: 10.1109/BRC.2014.6880975


