Controlling a DARwIn-OP Robot by Myoelectric Recognition Device

Ricardo Morales[1], Adrián Castañeda[1], and David Elías[2]

[1] UPIITA - IPN, Electronics Department
D.F., MX, 5729-6000 ext. 56882
ricardo.m.bonilla@gmail.com(student), acastanedag@ipn.mx,
http://www.upiita.ipn.mx

[2] Centro de Investigación y de Estudios Avanzados del IPN
D.F., MX, 5747-3800
delias@cinvestav.mx, http://http://www.cinvestav.mx/

Abstract. The main objective of this work is the creation and implementation of an interface that connects a myoelectric recognition device with a robot, so that, through movements of the hand and the arm, the robot executes certain actions previously defined, and that such actions are consistent with the movements of the operator. To achieve this, a gadget called Myo was used, which provides a discreet signal of five hand gestures plus spatial information that comes from its IMU; as for the robot, a virtual and a real model of DARwIn-OP were used, as well as an already elaborated walking routine. The movements and gestures of the hand were chosen so they result natural. The communication between Myo and the model of DARwIn was done through an SDK offered by Thalmic the same company behind Myo, and its written in a programming language called LUA. The virtual model of the robot is taken from the robotics simulation software Webots, and thanks to the purchase of a license for the software on its website, it was possible to continue testing the movement and control of the robot. The result is both a virtual model of DARwIn walking with complete freedom in a 3D environment, and the control of the real robot in a wireless way according to the movements of the arm, as well as several more movements indicated by the hand.

Keywords: Contlo, Myoelectric, Robot, DARWIN-OP, Interactive, Interface, MYO

1 Introduction

From time to time there is a substantial technological advance right after a paradigmatic impasse about what can and cannot be achieved. And so, out of some examples of this evolution, the human-machine interface development is remarkable [1, 2].

The road for this technology began at a time when buttons and levers were the norm. Not much time passed before this fantastic idea was introduced by science fiction about one day being able to manipulate objects by waving and moving the hand without even touching them. Now, thanks to the advances made in myoelectrics[3–5], these fantasies are not far from reality.

Today, technology is trending towards a revolution were gadgets have become incredibly portable so they can integrate more with people: there are smartphones, smartwatches, tactile interfaces, activity monitors, etc. The basis of this article is then settled in the idea that the next step in control interfaces will be the human body itself, blurring even more the edge between man and machine.

Now that electronic components have been miniaturised to levels that were not thought before, implementation of myoelectrics in control applications is a viable option. Starting there, a new generation of myoelectric recognition devices that wasn't limited to interaction solely with prosthetics [6, 7], but includes a much wider range of applications was born[8, 9]. Controlling DARwIn via this method is intended as a proof of concept project, i.e. showing the capabilities and results of what the synergy between myoelectrics and other engineering areas could have. Through whats described here, it was possible to make DARwIn interpret the movements of the hand and the arm of an operator, so it could perform predefined actions the same way it would if it were connected to a remote.

Lastly, the structure of this paper follows the next order: section two describes the myoelectric acquisition device, as well as the design of the interface to connect it with the robot; section three is all about the implementation for both a virtual and a real model of DARwIn and what were the results obtained. Section five is reserved for the discussion about this work, its scope and its future.

2 Myoelectric Recognition Interface and Control

Despite the fact that today myoelectric applications are common[10, 11], the market seems not to care less about them. On the other hand, myoelectric enhanced control is seen as an excellent option for academic purposes because of its versatility and precision.

Myoelectric control is centred in the recognition of patterns that can be later interpreted as instructions by a controlled device. Of course, there are limitations as to what can be manipulated with such interfaces since every recognised gesture of the hand has to be very well defined to avoid readings in between poses. By doing so, the myoelectric recognition system requires less processing power and it results in a much more compact device. Furthermore, adding a group of spatial sensors, like an IMU, improves functionality while, at the same time, increases usability. The myoelectric sensor used for this project is called Myo, its made by a canadian company named Thalmic, and its focused on rather domestic applications, which makes it a lot easier to use.

2.1 Sub: Myo

Myo is as close as it gets to have a personal myoelectric interface. It consists of a band that has several electrical sensors all around it, and an integrated IMU.

It is placed over the thickest part of the forearm as a bracelet and, once worn and connected, it provides information about the position of the operators arm, as well as data related to one of the five possible gestures of the hand that it can detect: an open hand, a fist, a double tap on the middle finger and a wave in or out of the palm. These digital poses are the result of what the myoelectric sensors "see" in conjunction with a preprocessing stage on the proprietary ARM chip in Myo[12]. It uses the Bluetooth Low Energy protocol for communication, and can be programmed in two different ways: C++ and Lua (Thalmic provides the SDK for both languages and, in the special case of Lua, the compiler as well). The simplicity of Lua made it the choice for this work, and since it can run on different platforms, like Windows and MacOS, it adds up to the versatility of the project.

2.2 Sub: Webots

The first step towards controlling DARwIn was a simulation (which happened to help in the communication with the robot, as will later be seen too). Webots provided a virtual environment for a 3D model of DARwIn. It uses a physics engine to simulate the behaviour of the robot the same way it would do in real life. Besides, it includes a compiling tool allowing for coding in-program instead of using different software for the same purpose. Apart from that, Webots has a remote control module that can be used to send instructions directly into the robot once it is connected to the computer either via LAN or wireless connection. A licence had to be purchased as well, since the free trial to use the software only lasts 30 days.

2.3 Sub: DARwIn

An open source anthropomorphic robot acted as target for the controller. DARwIn was developed by the company ROBOTIS and it's intended as a means of education, development and research. It comes equipped with an Intel Atom Z530 processor and 4GB of SSD[13]. It has more than 20 points of movement, which made it a perfect candidate to combine with the myoelectric sensors since it can perform many different actions. Besides, it is a modular robot, so additions can be done in the future to expand the capabilities of the control interface.

3 Interface between Myo and Webots

The first approach to the solution of this problem is to choose which movements are going to be done by the robot, and which ones by the operator using Myo. Determining both sets of movements marks the direction that the work must follow, and since this project acts as a proof of concept, a good start would be making the robot walk[14]. Myo has the ability to detect five different hand gestures that can later be combined with the spatial information from the IMU, so there is basically an infinite amount of possibilities to control something. It is then proposed to make the arm act as a joystick, therefore moving the robot in the direction the hand is pointing, e.g. rising the forearm will make DARwIn walk forwards, while lowering it will make DARwIn move backwards. Additionally, several more hand gestures were used to provide a wider range of motion apart from walking.

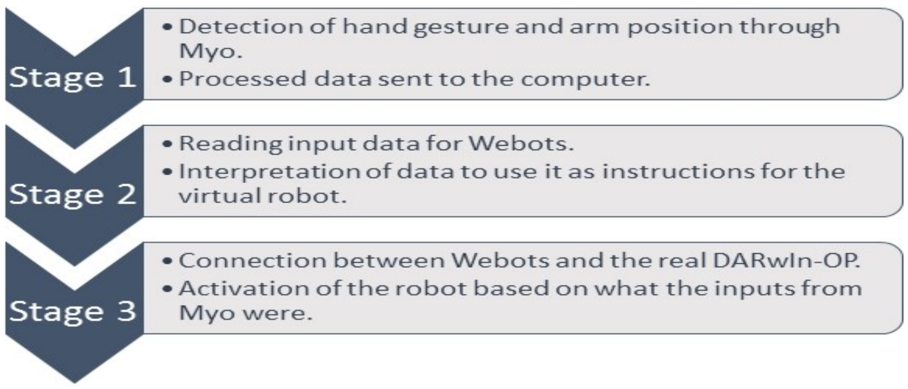
Stage 1
• Detection of hand gesture and arm position through Myo.
• Processed data sent to the computer.
Stage 2
• Reading input data for Webots.
• Interpretation of data to use it as instructions for the virtual robot.
Stage 3
• Connection between Webots and the real DARwIn-OP.
• Activation of the robot based on what the inputs from Myo were.

Fig. 1. Stages to control DARwIn with Myo

DARwIn already includes a library that lets the user play movement pages (animations) and control its walk (a gait manager). This reduced the amount of time spent modeling what the robot should do, since it only needed to be adapted to whatever input Myo was giving when it was tested; thus, practically have 3 stages to control the robot DARwIn, as can be seen in the figure 1.

3.1 Sub: Detection from Myo

Myo must be worn over the thickest part of the forearm, like a bracelet, allowing it to detect the electrical impulses coming directly from the muscles. Then, the processor can interpret those signals as a gesture, and the IMU provides additional information as to what is the relative position of the arm at a certain moment. In order to control DARwIn, the operator must move its hand as if it were a joystick, and joysticks have neutral points where no actions take place. Since guessing where the arm of the operator is is not an option, he must be able to set a neutral point whence he is able to move freely the way he intends to. So it was defined that whenever he closed his hand making a fist, Myo would set that point in space as a relative zero. Moving away from this zero will make Myo send a signal that DARwIn must interpret as an instruction to walk towards that direction. Of course, if no parameters are set, the slightest move of the arm will trigger an action in the robot, and therefore it was needed to establish not just a neutral point but a neutral zone surrounding the zero to keep the robot under control. This zone is 0.4 radians wide both vertical and horizontal, and was implemented as a safe lock via programming. The algorithm and how the neutral zone is implemented can be seen in figure 2.

Algorithm: Myo detection

```
if fist = detected then
    centre yaw  read yaw value
    centre pitch  read pitch value
if centre yaw =/= 0 then
    current yaw  read yaw value
    current pitch  read pitch value
    delta yaw = current yaw - centre yaw
    delta pitch = current pitch - centre pitch
    if delta yaw > 0.2 radians then
        subroutine move to left
        if delta pitch > 0.2 radians then
         subroutine move forwards
        else if delta pitch < -0.2 radians then
         subroutine move backwards
        else
         subroutine no movement
        end
    else if delta yaw < -0.2 radians then
        subroutine move to right
        if delta pitch > 0.2 radians then
           subroutine move forwards
        else if delta pitch < -0.2 radians then
         subroutine move backwards
        else
            subroutine no movement
        end
    else
        subroutine no movement
        if delta pitch > 0.2 radians then
         subroutine move forwards
        else if delta pitch < -0.2 radians then
           subroutine move backwards
        else
           subroutine no movement
        end
   end
end
```

Fig. 2. Algorithm to control Darwin with Myo device

Other movements of the robot are defined in a similar manner. For example, the robot can nod if the operator waves out with its palm, and it shakes its head when he waves in. Similarly, these gestures can be combined with spatial information from the IMU to create more commands according to what the operator and the user want to achieve.

3.2 Sub: Communicating with Webots

Once the set of movements for Myo and DARwIn was defined, the communication problem between Myo and Webots had to be addressed. That's where Lua comes in: any script written for Lua using the SDK provided by Thalmic can run over another layer of software, i.e. it can work the same way a touchpad or a keyboard work: as inputs. Regardless of what kind of program is open, Myo (using Lua) will keep sending input signals coming directly from the operator's arm and hand as long as it is programmed for that purpose. Considered this, the problem is reduced to how would Webots interpret the commands sent by Myo (since now Webots is able to "listen" to them), and in what way these would reach the physical robot.

A basic walking program for DARwIn, included in its simulation was made, so that the outputs from Myo and the inputs for Webots matched. In other words, the command necessary to make DARwIn walk forward is the same signal sent from Myo when the arm is risen. This allowed for a simple, yet useful, series of instructions compatible with each other that includes (but are not limited to) four commands to make the robot walk: forwards (arm up), backwards (arm down), left (arm to the left) and right (arm to the right). Since Webots accepts characters as inputs for the robot to move, different letters were assigned to each of the outputs sent from Myo, for example, whenever the arm was risen, a letter W was sent to Webots, and then the simulation, in return, would make DARwIn move forwards since the W was programmed to trigger this action.

Other moves performed by the robot were assigned with several different characters as well. For instance, number 1 and number 2 are, respectively, used to make the robot nod and shake its head; numbers 3, 4, 5, 6, 7, 8, and 9 have other reactions too. And for each instruction that is programmed into DARwIns code, another movement must also be adapted for Myo. DARwIn has many more preloaded pages of routines to perform, but if a new one were necessary, it could be done via RoboPlus, an application for modeling them.

4 Implementation in the Real Robot

With the simulation, commands and movements in place, it was time to change from a virtual to a real environment. This was accomplished once more with the help of Webots and one of its tools to program and control the virtual models of its robots.

It was originally intended to connect to DARwIn using a simple serial connection, with Myo dictating characters to a serial output terminal, and the other end receiving them through a cable or a wireless device. This implied that an additional stage for serial communication had to be built directly into DARwIn so it could read the output from Myo. Webots for DARwIn includes something called remote-control session which lets the user connect directly to the robot. Once the link has been established, three options become available: uploading the controller, beginning a remote-control session, and unistalling the controller.

Then, it was natural the first step was to connect DARwIn to the computer using an Ethernet cable, so the link between both could be tested. The minimum configuration parameters to comunicate with the robot are: a default IP (which is 192.168.123.1) a username and password to DARwIn's domain (included in the manual).

After DARwIn is placed in the correct position, Webots uploads the controller to the internal memory, then it requests for permission to act as a remote control for the robot. From that point whatever input the computer receives within Webots will act directly into the real robots, therefore, the characters needed to make the virtual DARwIn move will be equally valid to make the real DARwIn move too. This procedure was intended to work as a bridge between Myo and DARwIn. The results were successful: whenever Myo sent a signal to Webots, the real DARwIn reacted accordingly, manipulating the robot using nothing more than just the arm, as was planned from the beginning. However, this was still a tethered communication process since the robot had to be plugged all the time into the computer using the Ethernet cable. In order to provide DARwIn with complete freedom of movement, a wireless network connection had to be implemented. Two solutions were proposed: one, creating a wireless network using an additional router so both the computer and the robot were connected to it; or two, having the computer act as an ad hoc so DARwIn could detect it as a wireless network and connect to it the same way it would to a router. As far as simplicity goes, the first solution was optimal, but since creating an ad hoc is by no means a hard task, and it would avoid having to use an additional component, it was preferred over the router.

What had to be done was to enable Windows as an ad hoc via command prompt (since versions 8.x and above had this functionality removed from the UI), just then DARwIn could directly connect without cables. By taking advantage of the already established tethered connection, it was possible to access the graphic user interface of DARwIn (which is Ubuntu) to look for the recently created ad hoc and connect to it the same way a laptop would do to any wireless network. Two different methods can be used to get into DARwIns Ubuntu UI: one involves connecting the robot to a monitor using an HDMI cable, a keyboard and a mouse; the other (which was used for this work) requires the use of software called UltraVNC, which creates a virtual network connection that lets the user view, access and control another computer remotely. Once inside DARwIns computer, connecting to the ad hoc, it could obtain the IP address needed by Webots to create a remote-control session too. As for the final stage of the implementation, duplicating steps to connect to DARwIn via Ethernet was enough for a wireless connection too (the only difference being the IP address Webots used). Since the procedure had basically no changes, the results were pretty much the same, but now the robot was untethered from the computer and could move freely across the room. Figure 3.

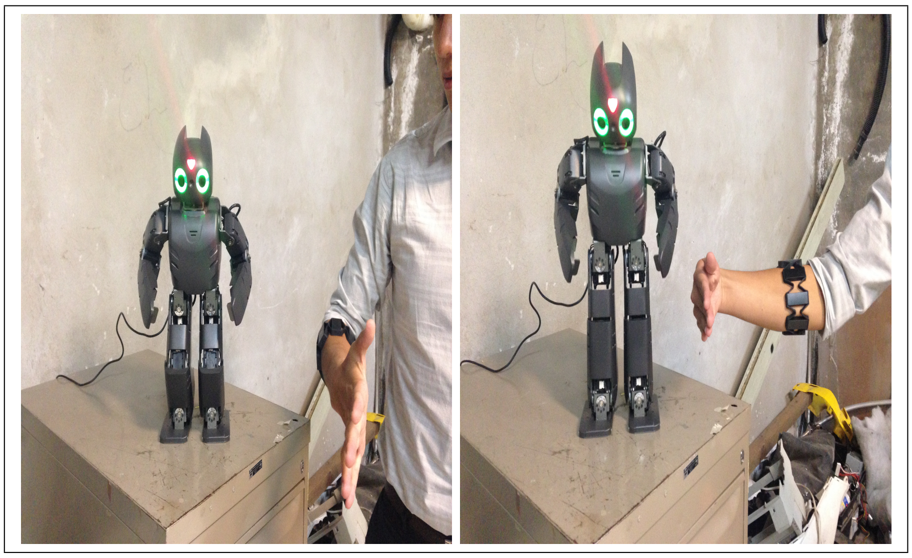

Fig. 3. DARwIn and Myo interacting

5 Results and Discussion

Right after DARwIn was successfully connected to Webots, and Webots was successfully connected to Myo through a script written in Lua, the bridge between these two was completed; the inputs coming from the movements of the arm and hand were interpreted as commands that were later send from Myo to Webots, Webots then read them and used them to control the virtual model of DARwIn, which was also linked to the real robot, making it move exactly as intended from the beginning. The result is full control over DARwIn, without even touching the robot, using an interface where actions are triggered by the natural movement of the arm and the hand rather than pressing buttons or tilting joysticks. Expectations about the reach of the project were fulfilled. As a proof of concept work, its even possible to say that merging myoelectrics with areas of engineering that involve control is viable.

There are limitations, though, as to what can be achieved with current myoelectric technology. Being able to reduce the size of myoelectric sensors comes with a price; processing power is ceded in order to get components to fit in the shape of a bracelet, and battery life is another concern too. This is the main reason why portable myoelectrics are not quite common yet, additional precision is needed for applications with finer control requirements, and even though the technology exists, it involves having bigger sensors and a larger processing units to obtain the desired results, which is expensive and impractical for the most part. The proposed set of movements for both the arm and hand, as well as the robot, obey this premise. They result simple (yet really useful) because the myoelectric capabilities of Myo are reduced to what the proprietary processor can manage. Engineering is an iterative process, and in some years the processing power will double, or triple, allowing for more significant applications for the technology available today.

Several improvements can be done to both the robot and what Myo sees as input and transforms into an output. The movements of the robot are limited to two of its library components: the gait manager (to walk), and the motion manager (to play motion pages), but since these can be modeled based on the necessities of the project, they could be increased to make motion more fluid, more natural, or both. Pattern recognition in Myo is limited too, but the information from the IMU is easily processed and it allows to increase the number of possibilities that a single hand gesture could have. Communication was done through Webots, but before that, it was said that a serial fully functional interface had been developed. This interface allows Myo to connect to virtually any serial compatible device, leaving an open door to further applications, not just with robots, but with drones, utility vehicles, robotic arms, even Arduino based platforms.

This project has an enormous potential, not just to create videogame-like interfaces as the one demonstrated here, but to built entire machinery applications around myoelectric technology. It is not hard to imagine a crane operator controlling it without being inside, having a wider view of the construction site through images coming from cameras, in a room where is safe to work far from dangers such as falling from a high place, heavy rain, electricity, etc. Opening the garage door, or turning down the temperature without even touching the thermostat are not less real now that what this project has achieved.

Acknowledgments

The authors thank the support from UPIITA-IPN and CINVESTAV for carrying out this work.

References

1. Schmalfu, L., Duttenhoefer, W., Meincke, J., Klinker, F., Hewitt, M., Tuga, M. R., Liebetanz, D.: Myoelectric control by auricular musclesan alternative human-machine interface. Clinical Neurophysiology, S116, 125 (2014)
2. Shin, S., Matson, E. T., Park, J., Yang, B., Lee, J., Jung, J. W.: Speech-to-speech translation humanoid robot in doctor's office. InAutomation, Robotics and Applications (ICARA), 6th International Conference on IEEE 484-489 (2015, February)

3. Young, A. J., Smith, L. H., Rouse, E. J., Hargrove, L. J.: A comparison of the real-time controllability of pattern recognition to conventional myoelectric control for discrete and simultaneous movements. J Neuroeng Rehabil, 11(5) (2014).
4. Scheme, E., Lock, B., Hargrove, L., Hill, W., Kuruganti, U., Englehart, K.: Motion normalized proportional control for improved pattern recognition-based myoelectric control. Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 22(1), 149-157 (2014).
5. Tkach, D. C., Young, A. J., Smith, L. H., Rouse, E. J., Hargrove, L. J.:Real-time and offline performance of pattern recognition myoelectric control using a generic electrode grid with targeted muscle reinnervation patients. Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 22(4), 727-734 (2014)
6. Hwang, H. J., Hahne, J. M., Mller, K. R.: Channel selection for simultaneous and proportional myoelectric prosthesis control of multiple degrees-of-freedom. Journal of neural engineering, 11(5), 056008 (2014)
7. Fougner, A. L., Stavdahl, ., Kyberd, P. J.: System training and assessment in simultaneous proportional myoelectric prosthesis control.Journal of neuroengineering and rehabilitation, 11(1), 75 (2014).
8. Ameri, A., Scheme, E. J., Kamavuako, E. N., Englehart, K. B., Parker, P.: Real-time, simultaneous myoelectric control using force and position-based training paradigms. Biomedical Engineering, IEEE Transactions on,61(2), 279-287 (2014)
9. Oskoei, M. A., Hu, H.: Myoelectric based virtual joystick applied to electric powered wheelchair. In Intelligent Robots and Systems, IROS 2008, IEEE/RSJ International Conference on IEEE 2374-2379 (2008, September)
10. Siomau, M., Jiang, N.: Myoelectric control of artificial limb inspired by quantum information processing. Physica Scripta, 90(3), 035001 (2015)
11. Ameri, A., Kamavuako, E. N., Scheme, E. J., Englehart, K. B., Parker, P. A.: Real-time, simultaneous myoelectric control using visual target-based training paradigm. Biomedical Signal Processing and Control, 13, 8-14 (2014). `http://www.xataka.com.mx/eventos-de-tecnologia/taekwondo-y-el-peto-electronico-en-los-panamericanos`, [Consulted 22/01/2014]
12. Thalmic Labs: Tech Specs de Thalmics Labs. `https://www.myo.com/techspecs` [Consulted 12/08/2015]
13. ROBOTIS: DARwIn-OP de ROBOTIS: `http://www.robotis.com/xe/darwin_en` [Consulted 12/08/2015]
14. Hong, Y. D., Lee, B.: Experimental Study on Modifiable Walking Pattern Generation for Handling Infeasible Navigational Commands. J. Elect. Eng. Technol (2015).
15. Roche, A. D., Rehbaum, H., Farina, D., Aszmann, O. C.: Prosthetic myoelectric control strategies: a clinical perspective. Current Surgery Reports,2(3), 1-11 (2014).
16. Oskoei, M. A., Hu, H.: Adaptive myoelectric control applied to video game. Biomedical Signal Processing and Control, 18, 153-160 (2015).
17. Oskoei, M. A., Hu, H.: Adaptive myoelectric human-machine interface for video games. In Mechatronics and Automation, International Conference on IEEE, ICMA 2009, 1015-1020 (2009, August)