Biomimetics & the 3D Printed Arm

De Haan Bosch, Lucas Enric

Curs 2015-2016

Director: MARIO CERESA

GRAU EN ENGINYERIA INFORMATICA

upf. Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

BIOMIMETICS AND THE 3D PRINTED ARM

LUCAS ENRIC DE HAAN BOSCH

END OF DEGREE PROJECT

BACHELOR’S DEGREE IN COMPUTER SCIENCE

UPF POLYTECHNIC SCHOOL

YEAR 2015-2016

PROJECT DIRECTOR

MARIO CERESA

Dedication: To my family.

Acknowledgements

I want to thank my family for keeping me centred on my work, and my friends for distracting me when it was needed.

Summary

This project aims to produce a Biomimetic software program capable of controlling the InMoov Robot's Right Arm, using inputs from EMG sensors placed around the upper forearm of its controller.

The InMoov robot, designed by Gael Langevin, is an Open Source robot available to all. It is built with an Arduino board, 3D printed pieces, and various other commercially available materials from any hardware store. The 3D Printed pieces' schematics can be downloaded from the website.

Along with the actual construction there are the sensors that are used to read and transmit the body's signals. The sensors used will be the Myo Sensor from Thalmic Labs, a Canadian firm who developed this bracelet of eight EMG sensors that is placed around the forearm. To combine both sides together this project will be exploring various robot interface environments to govern the interaction between the sensor, the computer and the robot (via Arduino).

Sumario

Este proyecto tiene como objetivo producir un programa de software biomimético capaz de controlar el brazo derecho del robot InMoov, utilizando entradas de sensores EMG localizados alrededor del antebrazo superior del usuario.

El robot InMoov, diseñado por Gael Langevin, es un robot 'Open Source' al alcance de todos. Está construido a base de un Arduino, piezas de impresión 3D y varios otros materiales que están disponibles en cualquier ferretería. Los planes de las piezas 3D se pueden descargar desde el sitio web.

Además de la propia construcción también hay sensores que son utilizados para leer y transmitir las señales del cuerpo. Estos sensores forman parte del brazalete sensor Myo, de Laboratorios Thalmic, una compañía canadiense que ha desarrollado este brazalete con ocho sensores EMG que se ponen alrededor del antebrazo. Para combinar ambos aspectos del proyecto explorare las diferentes interfaces robóticas para manejar la interacción entre el sensor, el ordenador y el robot (vía Arduino).

Prologue

The aim of this project is to create a robotic arm capable of imitating the movements of a hand, in particular the movements of the right hand's fingers. With this in mind the project tutor promoted the use of Gael Langevin's InMoov Open Source Robot as the hardware base.

The InMoov Robot is an ongoing project to create the world's first fully available open source robot, mostly as an educational tool for aspiring roboticists. The pieces are freely available for download on the InMoov website, only requiring access to a 3D Printer and easy to find hardware materials. In fact, the only piece that may pose some difficulty to find is the Arduino board used to control the robot's movement, which can be ordered via the InMoov website itself or via other means. In this case the univsersity provided an Arduino board.

The InMoov Robotics Arm project had been previously explored as an end of degree study in the 2014-2015 term by another group of students. They centred on the construction and correct function of the robot. In that particular instance, there were two main focuses: the viability of the InMoov as a robotics learning tool, and the use of the 3D Printer beyond the creation of spirometers and the occasional test statue.

This 2015-2016 term, the aim was to go beyond what the previous year's group had achieved and create actual software projects involving the robot. Concentrating, particularly, on constructing a new right arm using the experience from the previous iteration to avoid some of the shortcomings, such as the quality of the printed pieces and the material used as the internal ligaments to control the movements of the hand.

This writing's software project was aimed towards achieving a Biomimetic program capable of imitating the movement of a person's hand via EMG sensors strapped to the forearm. This would be significant as it would allow control of the robotic hand based on the movement of the muscles in the forearm useful for prosthetic substitution. Another way to centre the study would be in the remote manipulation of objects via said robotic hand. This in particular would be achieved through the use of the Myo Sensor, a relatively new tool developed by the Canadian company Thalmic Labs. The Myo Sensor is a wireless set of EMG sensors that transmits data between itself and the computer and comes with an on board set of ready-made gesture commands.

Finally, the most important piece of this project will be uniting both the Robot Arm and the Sensor Data into a congruent project, for this the project has explored the two distinct environments to act as the interface between the various pieces. These environments are My Robot Lab and the Robot Operating System, both systems function in a similar programming paradigm with a distributed program architecture where a particular project or robot will have more nodes added to it that allow it to do various different things, but the differences will be explored further on.

Combining all these different components into a single functioning entity will be the final objective of this project, and the process which had been undertaken to do this will be explained in the following text.

Index

1. INTRODUCTION

1.1 Motivation

The reasons for engaging in this project are varied, but from a personal point of view an interest in robotics, and in more recent years the area of person-machine interaction has helped in deciding on the focus of the project. This interest stems from a young age and has grown with the years, with the creation of more varied robots in the news and the advent of interactive robotic toys, such as the old Lego Mindstorms or the original electronic Lego sets, which allowed for a surprising amount of inventiveness.

From this basis it was natural to begin to look into various topics relating to the area. In particular, looking into the various optional subjects available in the UPF Computer Engineering degree, with a special interest in the subjects related to the interaction between person and machine or world and machine as well as the offered introductory course in Robotics.[1] With this background it was natural while looking through the list of available end of degree projects to gravitate towards this particular project offered by Mario Ceresa.

Apart from these personal reasons there are other justifications for realizing this project; the endeavour of creating a successful person-machine interface is interesting and worthwhile investment. In learning more about this area of study and thanks to the nature of the resources used, which are easy to replicate in an educational capacity, as well as to develop more advanced projects. In the experience I've had, the use of the InMoov robot as a learning aid is particularly notable, especially as an introductory large project to building, programming and wiring up a successful robot. [2]

As a final reason, there are a wide range of applications for a biomimetic software designed to function with a 3D-printed robotic arm within the realm of science and medicine. In the scientific community, a more evolved version of program could be used for remote manipulation of materials for a relatively low price. A similar argument can be made for its possibilities within the world of medicine with cheaper prosthetics and new ways for substituting for loss of motor flexibility.

1.2 Objectives

This section will describe how far exactly this projects hopes to take itself using the tools that are available.

The overall objective of this project is the following:

- To create a robotic arm and associated software project capable of reading a sensor and copy the movements of a hand in an acceptable manner.

To begin with the project must be defined more clearly in the sense of what exactly will be built physically, since this project partially consists of building a robot and thus the limitations must be correctly expressed. The robot itself will consist of the right forearm and right hand of the InMoov Robot as designed by Gael Langevin which will have the capacity to extend and contract each finger independently of each other. As well as allowing the wrist to have a movement range of 180º degrees, where 90º degrees is having the hand perpendicular to the ground. All of these should be controllable from a microcontroller capable of connecting to the computer.

With this part clear the more detailed objectives can define the exact capacities of the program. The program must be capable of:

- Translating the data from a sensor, in particular, the data from the Myo Armband Sensor.
- Transmit said data into the arm
- Have the arm move accordingly, with relatively little delay between the hand movement and the robot movement.

When these objectives are completed, the evaluation and analysis of the results can begin, discussing the limitations and possible improvements to the project in future.

1.3 Structure of the Document

This project has a practical basis, therefore there will be a thorough description of how it was carried out. However, firstly this writing will explain the context in which the project is being realized, such as the various works and previous projects related to the use of prosthetics and EMG as a basis to control them, as well as a brief overview of the tools used.

Once this is done the actual progress of the Project will be described as well as the various tools starting from the arm, passing through the Myo, with a brief dip into the anatomical structures involved in the detection of EMG signals. With both pieces of hardware covered the actual program and programming environment will be described. In each of these sections, a brief final overview of difficulties and errors during the experience will conclude the section. The final two chapters will be on the testing and evaluation of the final product and the entire project, respectively.

2. STATE OF THE ART

2.1 Introduction

This section will be going over the previous works and various other contextual information not directly related to the involved pieces of hardware, which are explained in their respective sections. In particular, it shall endeavour to explain how the idea of EMG controlled prosthetics has evolved over time, starting with the two ideas separately and then as a unit.

2.2 Robotic Prosthetics & Control [3] [4]

Robotic prosthetics have existed within the human mind for a long time, although for many the idea of a robotic prosthetic fills them with ideas of sleek, silver futuristic arms from various sci-fi shows or in some cases, of brass coloured contraptions of clockwork and steam.

In the beginning, a prosthetic was for the most part a fairly basic affair, with the traditional hook. As time passed, more advanced forms of prosthetics began to appear over time. But one of the principal problems with prosthetics is the social stigma associated with them, along with the added long period of adjustment. The stigma has been stymied in recent years with more aesthetic designs beginning to arise, although the long period of therapy and the price of the actual prosthetic have made them prohibitive or impossible for various users. As such, amongst the first things that people have focused on when trying to improve bionic arm prosthetics is their availability to their target users.

Hands and generalised upper body prosthetics are a much harder problem for the scientific community. The use of hands in a person’s day to day life is critical for many and is a unique and useful tool for humanity. As such, many of the issues can be summarised in trying to recreate the hand in an effective way. The first problem that arises in this is the capacity for the hand to achieve a certain amount of fine motor control. The most basic systems consist in a series of cables attached to harness which take advantage of the residual arms movements to achieve a facsimile of action. Achieving more complex gestures without turning the prosthetic into a hindrance is difficult but various ideas have appeared over the years especially with the discovery and development of new lighter materials to work with. Some propose using hydraulics as it would offer more precision but for the most part the main component to achieve fine motor control is finding an effective way for the person to interact with the prosthetic.

Nowadays, this is reflected by a rising interest in organic to digital signal transformation. Apart from EMG other ideas have occurred, such as the innervation of the muscles. The remaining nerve endings of the muscles are redistributed within the arm’s stump creating multiple pairs of muscle-nerve endings, each of these, with extensive therapy, can be trained to move independently from one another to a certain degree, increasing the number of possible control points for a new arm.

Another option for control is the direct implantation of intra-muscle EMG probes. This is supported by the idea of “Biomechatronic design” in which the design of the prosthetic is heavily inspired by the actual hand. This emphasis on a more natural design would support a more complete integration between machine and human. It might even allow for the passing of information in both directions, machine to human and vice-versa. This is particularly important for the second most important factor in prosthetic limb design, the capacity of sensation. For the most part in the main idea for allowing a limited capacity for sensation is substituting the sense of touch with something else, for example a sound, although with the emphasis on biomechatronic design the possibility of creating artificial touch sensation has begun to take some important strides.

Finally, there has been some exploration into non-standard limb substitution to allow for a greater amount of flexibility or to simplify costs in certain aspects. For example, the use of a robotic tentacle instead of a hand, inspired by how manoeuvrable octopi are, allows for a fairly wide variety of movements.

2.3 About InMoov

The InMoov Robot, designed by Gael Langevin a French sculptor, it is the world's first 3D Printed Open Source Robot. Begun in January of 2012 the InMoov robot began in it's infancy as part of a job commissioned to build a futuristic prosthetic, the job itself was not successful, but Langevin decided to release the design to the Internet, at that time consisting of just an arm. The prosthetic became a hit within the Internet and grew in popularity with positive responses and ideas flowing in. [5]

Thanks to the community's response Langevin decided to continue his work. With the input of the community, the InMoov Project moved on forwards with the intervention of new people and the added feedback coming from various individuals and groups building the robot in their own workshops.

The first iteration of the robotic hand was originally created for a photo commercial and commissioned to be able to make most of the human hand movements. From this empty but mobile model Langevin began to experiment using cables or fishing line to move the fingers. This first version of the hand was also the sculptor's first time working with Arduino and trying to control the servos from it.

Working from the hand and the forearm, the first designs for a shoulder were crude but effective, capable of sustaining the weight of the entire forearm and also positioning them in various angles. Notable about this iteration was the use of pistons in the movement of the arm, which later on would be substituted by high torque servos and printed gears.

Once a stable base to work with had been achieved, Langevin began to work on making the hand's movements more realistic by adding wrist movement. This was achieved via a servo motor added to hands wrist thanks to a pair of gears that would turn the hand one way or another. Of particular importance for this part was allowing the finger controlling cables to pass through the wrist without it impeding the hand's movements. This was achieved via a set of cable guiding pieces placed at various intervals along the new wrist piece.

From this point forward, there was a major change in the design by having the entire Arduino board removed from the forearm to make space for five individual servomotors to control each finger individually. The Arduino at this point would be moved up into the Arm's biceps.

Langevin began to experiment with both voice recognition and object recognition to move and command the InMoov robot, at this moment consisting of two arms, the shoulders and a base to hold both. All of these improvements were achieved during a period of six months during which the robot's most iconic and most built section was first designed, it's arms.

By October of the same year, the InMoov robot began to have more advanced control characteristics with the collaboration of GroG (pseudonym) creator of the MyRobotLab, a multi-faceted interface based in java and python, who helped Langevin create a voice recognition software. Along with this the first iteration of the robot's head was created consisting of the robot's eyes (with incorporated cameras), ears (with microphones) and a basic faceplate. This new head would allow for limited neck and eye movement. Later on a jaw and the rest of the head would be added to the existing structure.

During the following year, the InMoov robot would gain an upsurge of popularity appearing at various robotics expositions and fairs across Europe along with those versions produced by other people following Langevin's designs. During this timem progressive minor improvements were made along with the expansion of the torso and added aesthetic plastic plates. By the fall of 2014, the complete InMoov robot would consist of two arms, a head including its neck, the upper torso with shoulders and the lower torso with waist movement. By Spring of 2015, the beginning of two legs would be added.

2.4 Hand Anatomy[6]

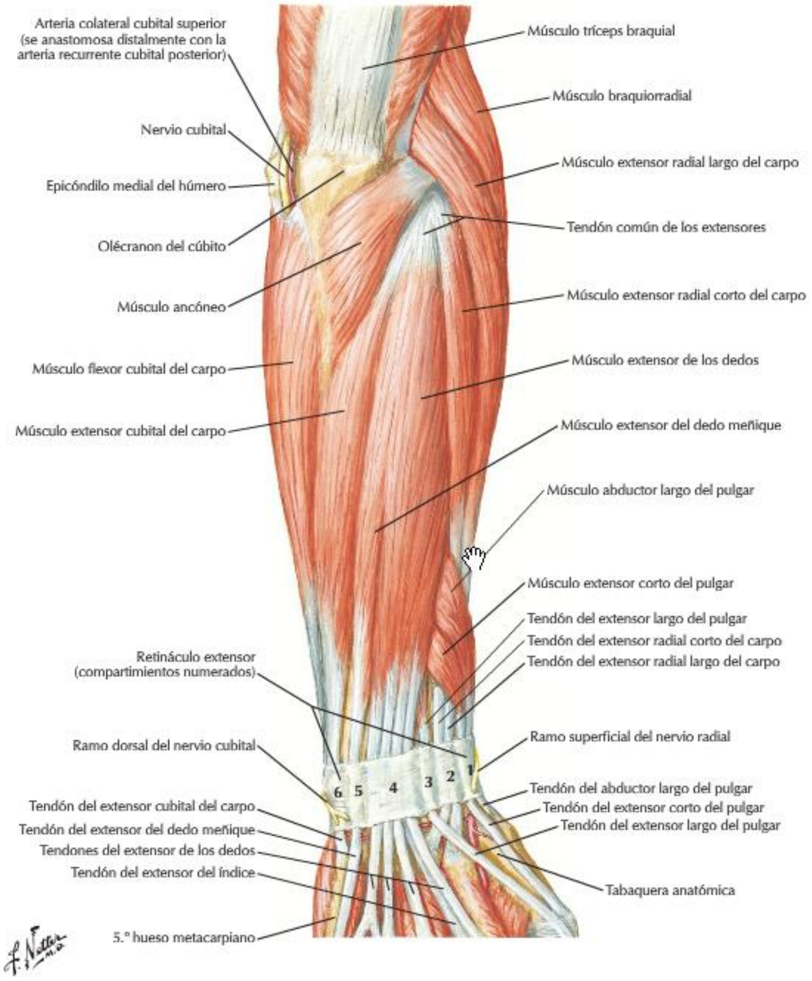
The hands are a very important and versatile portion of the human body, allowing humans to manipulate the world around them in ways that no other species found outside the order of the primates can. A single human hand consists of 34 muscles to move the fingers and the thumb, 29 major and minor bones, 48 named nerves and 30 named arteries. That is only a small overview of what the hand and forearm contain, that allow them to move the hand with incredible precision.

The objective in this is to imitate a very limited range of these movements. For this it will use, as mentioned previously, the Myo Armband, which uses 8 EMG sensors and an accelerometer, but before that there must first be an understanding of what is happening beneath the skin.

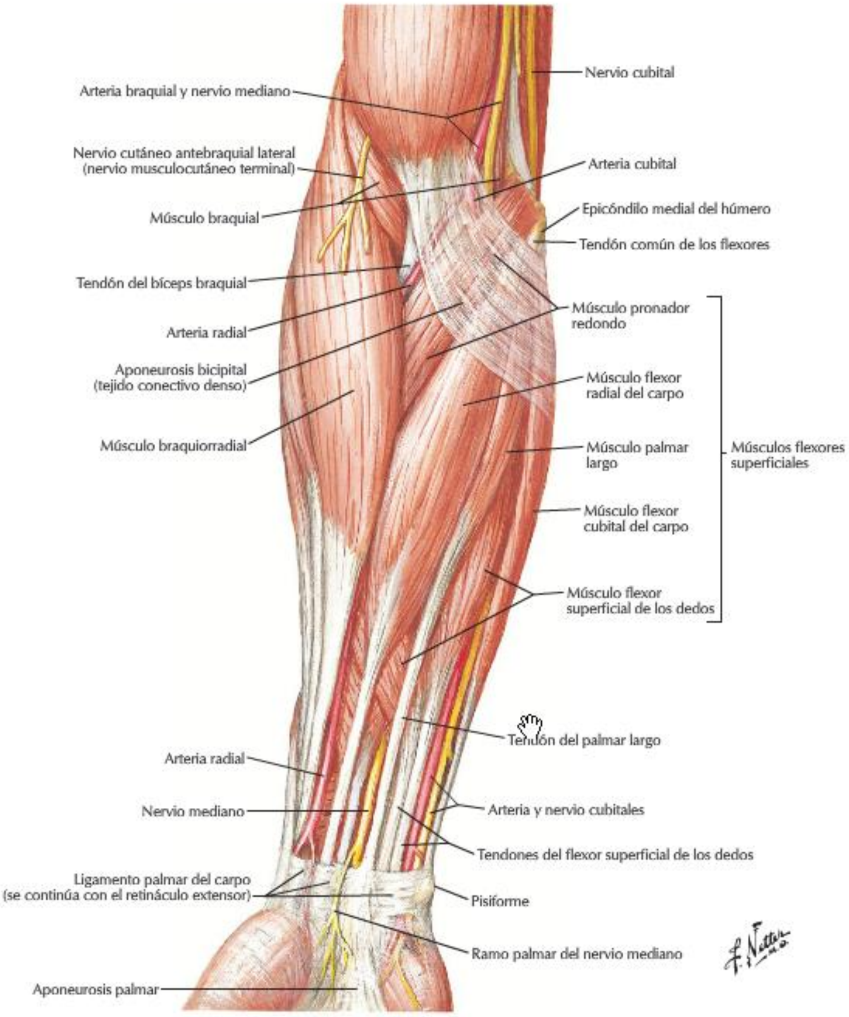
The hand by itself consists of 27 bones: the carpal bones (which articulate it with the wrist), the metacarpals and the phalanges (proximal, closest to the body, intermediate and distal, farthest from the body). The muscles of the hand can be divided into extrinsic, located on the forearm, and intrinsic muscles, located within the hand.

- The extrinsic muscles are divided into flexors, on the underside of the forearm, and the extensors, on the upper side of the forearm.
    - The fingers have two flexors: the deep flexor and the superficial flexor. There are three extensor muscles: for the index finger (*extensor indicis*), for the little finger (*extensor digiti minimi*), and the *extensor digitorum communis*, which handles the proximal phalanges.
    - The thumb has a long and a short extensor (*extensor pollicis longus*, *extensor pollicis brevis*), a long flexor (*flexor pollicis longus*) and a long abductor (*abductor pollicis longus*). This last one allows the thumb to move away from the centre of the palm.

- The intrinsic muscles are divided into different groups: the thenar muscles (thumb), hypothenar (little finger) and the interossei muscles and lumbrical muscles.


Arteria colateral cubital superior (se anastomosa distalmente con la arteria recurrente cubital posterior)
Músculo tríceps braquial
Músculo braquiorradial
Nervio cubital
Músculo extensor radial largo del carpo
Epicóndilo medial del húmero
Tendón común de los extensores
Olécranon del cúbito
Músculo extensor radial corto del carpo
Músculo ancóneo
Músculo extensor de los dedos
Músculo flexor cubital del carpo
Músculo extensor del dedo meñique
Músculo extensor cubital del carpo
Músculo abductor largo del pulgar
Músculo extensor corto del pulgar
Tendón del extensor largo del pulgar
Tendón del extensor radial corto del carpo
Retináculo extensor (compartimientos numerados)
Tendón del extensor radial largo del carpo
Ramo superficial del nervio radial
Ramo dorsal del nervio cubital
6
5
4
3
2
1
Tendón del abductor largo del pulgar
Tendón del extensor cubital del carpo
Tendón del extensor corto del pulgar
Tendón del extensor del dedo meñique
Tendón del extensor largo del pulgar
Tendones del extensor de los dedos
Tendón del extensor del índice
Tabaquera anatómica
5.º hueso metacarpiano
F. Netter M.D.

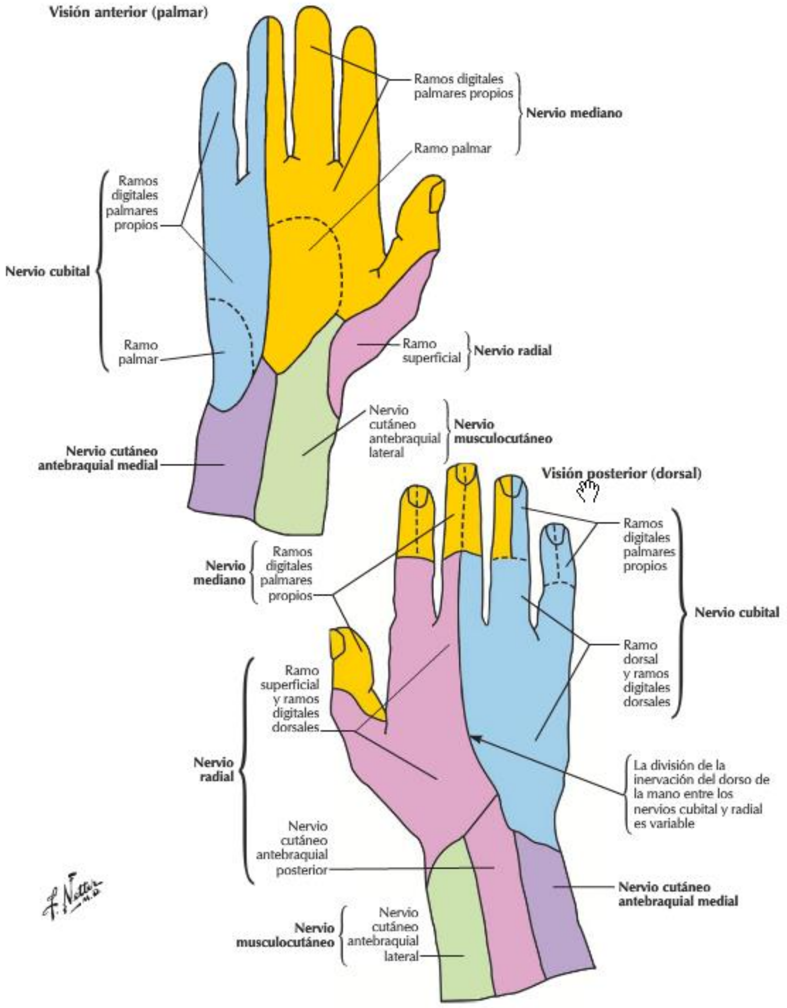
(Figure 1: Forearm Muscles, Surface plane, anterior side.)[7]


Arteria braquial y nervio mediano
Nervio cubital
Nervio cutáneo antebraquial lateral (nervio musculocutáneo terminal)
Arteria cubital
Músculo braquial
Epicóndilo medial del húmero
Tendón común de los flexores
Tendón del bíceps braquial
Músculo pronador redondo
Arteria radial
Aponeurosis bicipital (tejido conectivo denso)
Músculo flexor radial del carpo
Músculo braquiorradial
Músculo palmar largo
Músculos flexores superficiales
Músculo flexor cubital del carpo
Músculo flexor superficial de los dedos
Tendón del palmar largo
Arteria radial
Nervio mediano
Arteria y nervio cubitales
Tendones del flexor superficial de los dedos
Ligamento palmar del carpo (se continúa con el retináculo extensor)
Pisiforme
Ramo palmar del nervio mediano
Aponeurosis palmar

(Figure 2: Forearm Muscles, Surface plane, anterior side.)[8]

The movement of the hand’s fingers are linked to major nerves: the median, the ulnar and the radial nerves. The median nerve runs from the shoulder to the hand, in the forearm, the median nerve innervates the muscles on the underside of the forearm (ventral, anterior) which are the flexors, mentioned above. The median continues innervates the thenar muscles (mentioned above) and the first two lumbrical muscles (index and middle fingers). The ulnar nerve innervates the hypothenar region (side of the hand opposite the thenar/thumb side), and the last two lumbrical muscles (the ring finger and pinkie finger) as well as the palm. The radial nerve innervates the dorsal or posterior muscles, which are extensors.

These explanations are heavily simplified but they give a relatively clear picture of the main nerves being dealt with in this project, although note that each of these nerves also serves to transmit sensory data.


Visión anterior (palmar)
Ramos digitales palmares propios
Nervio mediano
Ramo palmar
Ramos digitales palmares propios
Nervio cubital
Ramo palmar
Ramo superficial
Nervio radial
Nervio cutáneo antebraquial lateral
Nervio musculocutáneo
Nervio cutáneo antebraquial medial
Visión posterior (dorsal)
Ramos digitales palmares propios
Nervio cubital
Nervio mediano
Ramos digitales palmares propios
Ramo dorsal y ramos digitales dorsales
Ramo superficial y ramos digitales dorsales
Nervio radial
La división de la inervación del dorso de la mano entre los nervios cubital y radial es variable
Nervio cutáneo antebraquial posterior
Nervio cutáneo antebraquial medial
Nervio musculocutáneo
Nervio cutáneo antebraquial lateral

(Figure 3: View of the areas each nerve handles, pink is radial, blue is ulnar and yellow is median) [9]

All these nerves are built atop the muscles of the hand, most of the ones that are dealt with are called skeletal muscles as they are in charge of moving the hand.

Finally, the wrist will be explained, which in this case is not really worked on very extensively due to the limitations of the Myo Armband but it is still important to know about.

The wrist is a complex joint that bridges the hand and the forearm. The joint is a collection of multiple bones: the distal, farthest from the body, ends of the radius and the ulna, the 8 carpal bones (scaphpoid, lunate, triquetral, pisiform, trapezium, trapezoid, capitate and hamate) and the proximal, closest to the body, portions of the 5 metacarpal bones.

There are three main movements permitted by the wrist. Palmar and dorsal flexion (tilting towards the palm and towards the back of the hand, respectively). This movement takes place through an oblique axis, which results in an ulnar deviation with palmar flexion and a radial deviation with the dorsal flexion. Abduction (radial deviation, moving towards the thumb) and aduction (ulnar deviation, moving towards the little finger), these movements take place along the dorsopalmar axis. The rotation of the whole wrist, which includes pronation and supination, occur with the help of the elbow joint (radioulnar joint).

In order to understand the movement of the wrist and the function of the muscles, one has to consider the two main axis of the joint: the transversal axis (palmar and dorsal flexion) which will divide the muscles into flexor muscles (anterior, palmar side) and extensors (posterior). The sagittal axis will divide the muscles into lateral and medial muscles, which allow for radial and ulnar deviation, respectively.

The muscles of the forearm allow for these complex movements of the wrist. There are different muscles of the forearm will allow for the movement of the different parts of the hand.

2.5 Electromyography[10]

Electromyography (EMG) is a diagnostics technique used to evaluate the electrical activity of the muscles responsible for movement, also known as skeletal muscles. This is done by detecting the electrical potential generated when the cells are activated, either by electricity or neurological activity. As such, this tool is capable of detecting and diagnosing various forms of neural or muscular problems, from involuntary muscular spasms that might be invisible to the naked eye, to various forms of nerve damage. One of the most common problem forms of nerve damage comes in the form of Carpal Tunnel Syndrome, where the median nerve is compressed by the wrist, either by the bones or muscles, a fairly common occurrence for those that work with computers.

For this form of diagnostic there exists two type of EMG sensors: intramuscular and superficial. In the case of the Myo Armband, the sensors are superficial, for ease of use and installation. Before delving into the Myo armbands functions, a brief word on how intramuscular EMG works.

In the case of intramuscular EMG, an electrode probe must be inserted into the muscle itself, this probe can be a fine wire inserted via needle into the muscle or pairs of electrodes systematically inserted into the muscle. In medicine, the most common form of electrode probe is normally tough enough to penetrate skin and is insulated from outside interference, apart from the tip. In a typical EMG medical test, the electrode probe is inserted into the relaxed muscle at various points to evaluate the electrical activity during insertion, called the insertional activity, abnormalities in this case are reflected by unusual potential fluctuations lasting more than 100ms, indicative of neural damage. Afterwards the potential is registered for the relaxed muscle, followed by a repeat of the process on various points. Once these baseline records are registered the test follows up by having the patient slowly contract the muscle to see the electrical fluctuations are recorded and judged, then the needle is moved out a bit and test is repeated, sometimes as many as 10 to 20 times per location. For an exact study of the muscle in question this must be repeated at various points along the muscle due to how the internal structure differs. Afterwards, the data is collected, filtered for noise and evaluated by a physician for possible abnormalities.

In the case of the superficial EMG, the method is widely different and much easier to do. The superficial EMG measures the muscle activity from atop the skin, generally by using pairs of electrodes to measure the potential difference between two points, this potential is generated by neural activity involved in the activation and movement of the muscles.

For more complete EMG recordings, it is typical to use complex arrays of multiple electrodes working in tandem. In the case of the Myo Armband it has 8 EMG sensor with incorporated electrodes, measuring the muscle activity of the forearm. The results are much more limited in their capacity to be used as a diagnostic tool, but have the advantage of being non-invasive and before the invention of the armband, involved shaving the arm and covering it in conductive gel.

2.6 EMG as a HCI Tool[11] [12] [13]

The use of Electromyography (EMG) within the realm of medicine is well documented, with the first record of the phenomenon being from 1666 by Italian physician Francesco Redi, who noted how the contraction of the muscles of an electric eel generated a spark of electricity. Later on, the same was observed for the contraction of muscles, and throughout most of the 20th century the use of electricity to treat various disorders and begin to make some leeway into its use as a diagnostic tool. It was not until the 1980s when true progress was made thanks to the advances in electrode sensitivity, amplification of signals and easier level of manufacture. From then on, the understanding of EMG signals has been growing over time and some of that understanding came in the fact that EMG could theoretically translate the analogue nature of hand movement into digital, a very important step in Human to Machine interaction. From these first ideas various others began to form on how these signals should be read and interpreted.

In the interpretation of EMG signals into a readable there are two principal schools of thought: on the one hand there is the idea of directly interpreting the data into movement. This system, while useful in the fact that it can be used for just about every conceivable combination of muscle movements in the hand, has the added cost that training such a system for each specific user would be copious and largely impractical due to the fact that EMG signals can vary wildly from one user to the next. The other more popular school of thought is that of teaching a system to recognize a limited set of gestures, this allows for a much wider range of potential users as these types of models generally expound on the probabilistic nature of the muscle signals being related to one movement or another.

This last school of thought, more commonly known as gesture recognition, has various ideas on how the translation of EMG into a detectable gesture could be used for Human-Computer interaction. For this various different types of algorithms were designed trying to take advantage of different probabilistic models ranging from Gaussian bell curve models to various forms of neural networks. All in all, each of these models have one thing in common which is their interest in allowing mobility disabled people to be more capable of interacting with the world around them.

2.7 Robot Operating System (ROS)

Robot Operating System or ROS is described as a collection of software frameworks used in robotic software development. This is a fairly accurate summary of its main functions. The ROS environment allows one to develop software for a multitude of various robots and other devices, providing a standardised base from which various services are provided. Amongst them are Hardware Abstraction, allowing the program to simulate or emulate other system and their limitations, device control, such as driver and interfaces allowing for some level of globalised common control of a myriad of devices, message-passing between process, a very important aspect in the project since various programs working in tandem are needed.

The ROS environment was first created as a response to the ongoing phenomenon where various forms of software projects based on previous work or thesis had to remake the entire program from scratch as the documentation and code were not published, was left uncommented or was impossible to work with. In response to the ROS project was begun as a standardised development environment by Stanford Artificial Intelligence Laboratory. Its objective was to unify the, until then, fragmented approach towards the development of AI development by incorporating the concepts of machine learning, vision, navigation, planning, logic, reasoning and natural language processing. All these areas had been previously explored within AI but often as separate projects within this larger area of study, a tendency that was noted by the creators of ROS to have been the norm for the previous 30 years of study.

A ROS project functions in a distributed node design, with a central Core node handling the communicating environment. A ROS project begins within a workspace, this workspace contains the code and protocols to create and manage each individual node that is activated.

A node in ROS is the working unit for the environment, it is a self-contained program or executable which functions within the ROS-core central node. Said nodes can be implemented to do a great variety of actions, allowing them to interface with hardware, make complicated operation, subscribe to data-streams, publish data into the communication environment so that other nodes may subscribe to them, imitate network and various other options. Generally speaking, a node is implemented by calling a ROS package.

A ROS package is the container for the code and configuration for the building and execution of nodes. These packages are installed within the ROS project workspace either by creating a new package from scratch or downloading said package from somewhere else. Distinguishing a package from other workspace folder is the fact that within there is a file called package.xml which defined the configuration of the package, including the available node programs, their dependencies, the build order and optionally the package author information. Once a package is included within a workspace the package itself must be built into the workspace so that ROS recognizes the nodes that is adds. The method for this varies depending on which ROS-distribution is used, but in the case of ROS JADE the command is ‘catkin_make’ followed by ‘catkin_make install’.

Once the package is installed the nodes within can be called with a simple command, which varies depending on which version of the ROS environment is used, but these will only function if the ROS-core Node is activated first.

ROS is designed to work with three different programming languages, C++, Python and LISP, as such most packages are written with said programs sometimes even both C++ and Python. This is possible thanks to the fact that a ROS environment workspace during creation includes the installation of both the ‘ros.h’ and the ‘rospy’ dependencies. As such the entire environment functions with these languages, although other languages can be implemented into the ROS system.

The most important factor in the use of ROS within C++ and Python is its capacity to publish and subscribe topics from various nodes. For a node to be able to do this it must first contain an instantiated node handler object. The node handler is what turns a normal executable into a ROS node. Once a node handler is instantiated it can be used to do various actions. In the case of this project, the node handler is used to handle the subscription and publication of topics, as explained previously.

In the case of publishing data towards a topic, a Publisher object must be defined and filled in using the advertise function. The advertise function takes a string, as the name of the topic, and an integer which defines the queue, how much data it saves up to publish afterwards. With the publisher defined it can also optionally be defined as a Rate object which manages how often a new piece of data is published on the topic. Once this is done, a loop must be created which will run for as long as the ROS node is functioning (it can be system interrupted). Within the loop, the entered data that will be published, transform it into a suitable datatype and pass it into publisher via the publish method. Then the previous Rate object will be called to manage the loop rate.

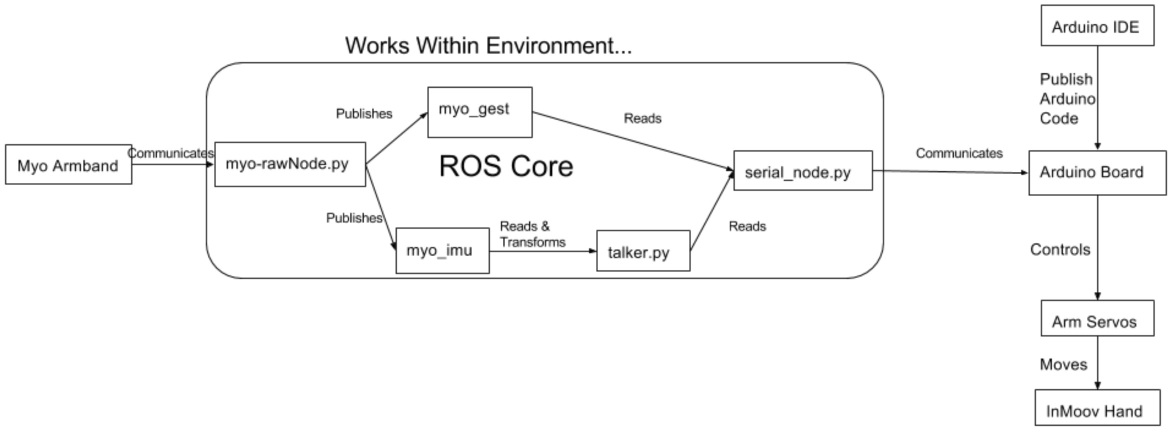
For subscriptions the basic setup is similar, defining a node handler, having a loop, instantiating a ROS object. In this case the subscription object is instantiated with three parameters, the name of the subscribed topic, the amount of data that will queue up to be processed and most importantly the call back function. The call back function will execute every time a new piece of data from the topic is received by the node, the function is defined like a regular function but must take as a parameter the datatype that the topic is publishing. The function then executes. To make sure that the subscription renews its data and executes the call back function it can either call the spin() function, which will imitate the loop from the publisher in one function, or include the function spinOnce() within an existing loop to make the subscriber renew its data and execute the call back function once.

Both of these types of nodes can be implemented in both Python and C++, and can also be combined into a single node, for example the ‘talker.py’ within the ‘imu_ros’ package.

3. MATERIALS & METHODS

The following figure describes the basic functioning of the actual system. Starting from the armband, it communicates with the computer. Within the computer the ROS Environment is running. Within this environment the myo-RawNode.py program communicates with the Myo Armband's systems and translates them into a series of published topics, 'myo_gest' and 'myo_imu'. The latter is read by the talker.py program and is transformed into a number between 0 to 180º corresponding to the armbands relative roll position and is published as 'roll'. The former, contains what gesture the Myo Armband has recognized. The last node within the environment is serial_node.py which works as an interface between the ROS environment and the Arduino.

The Arduino IDE is loaded with the corresponding code, seen later on in this chapter, and then uploaded into the Arduino board. From here the Arduino board communicates with the serial_node.py to read the afore mentioned topics. Using these topics, it controls the arm servos which in turn control the movement in the arm. Unfortunately, due to issues with the data format of the topic transmission the control of the wrist with the roll topic is flawed and causes stability issues, thus the wrist servo is disconnected.



(Figure 4: System Control Diagram)

3.1 InMoov Robot Arm

In this chapterthe nature of the first piece of hardware that was used and worked on will be explored; the InMoov Hand. Throughout this chapter the writing will endeavour to give a contextual description of the InMoov Robot Project as well as how it's construction went for the participants.

To note this section of the project was done with fellow student Laia Mas de Xaxars, both participants would be using the hand for their own individual software projects.

3.1.1 Building the Arm[14]

During this part of the project,t the work was done with the help of Laia Mas de Xaxars. This was decided due to the fact that both participants expressed interest on the project proposed by the project tutor, Mario Ceresa, thus each of them would work on the same piece of hardware while realizing two distinct software projects later on, thus cutting down the building time.

The InMoov is built almost exclusively from 3D printed pieces downloaded from the InMoov website, as such most of the time was dedicated to printing these pieces. To achieve this, the UPF gave us access to a WitBox 3D Printer on campus. Said printer was used for several weeks to print out the pieces of the hand and forearm, with the occasional failure due to errors during the printing process.

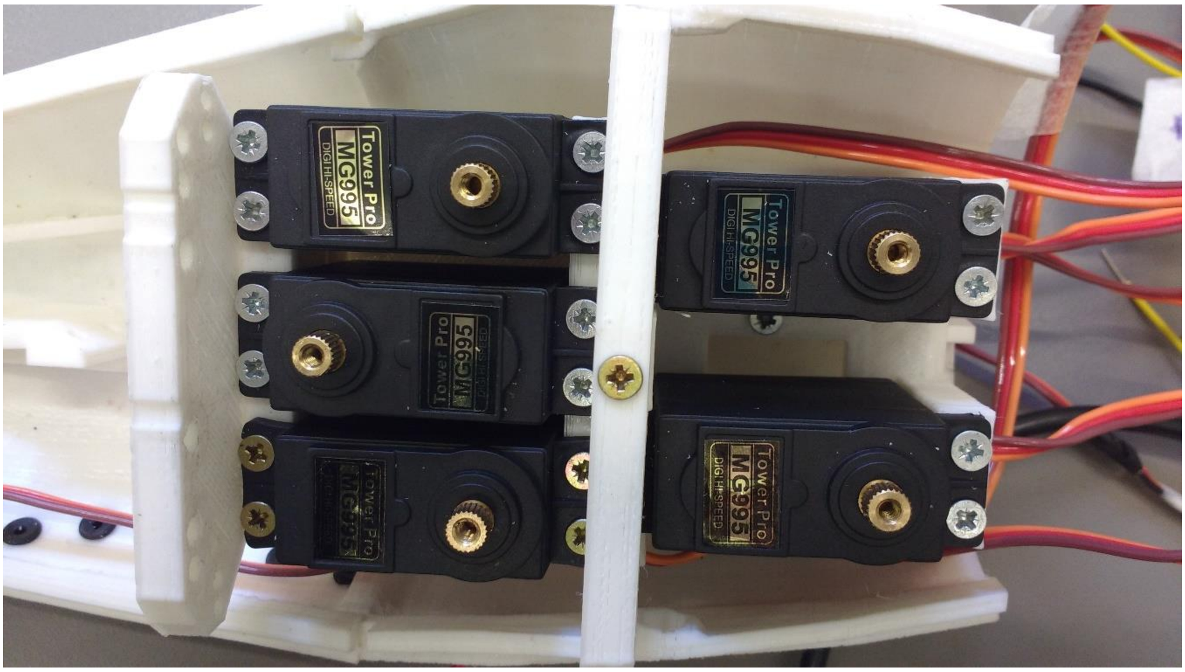
During this period, some experimentation was done with 3D printing, with the help of staff member, Genis Caminal Villodres. He explained and recommended how to use the models downloaded from the website and edit the printer settings to try and get a relatively smooth but solidly printed piece.

The Printing process is as follows:

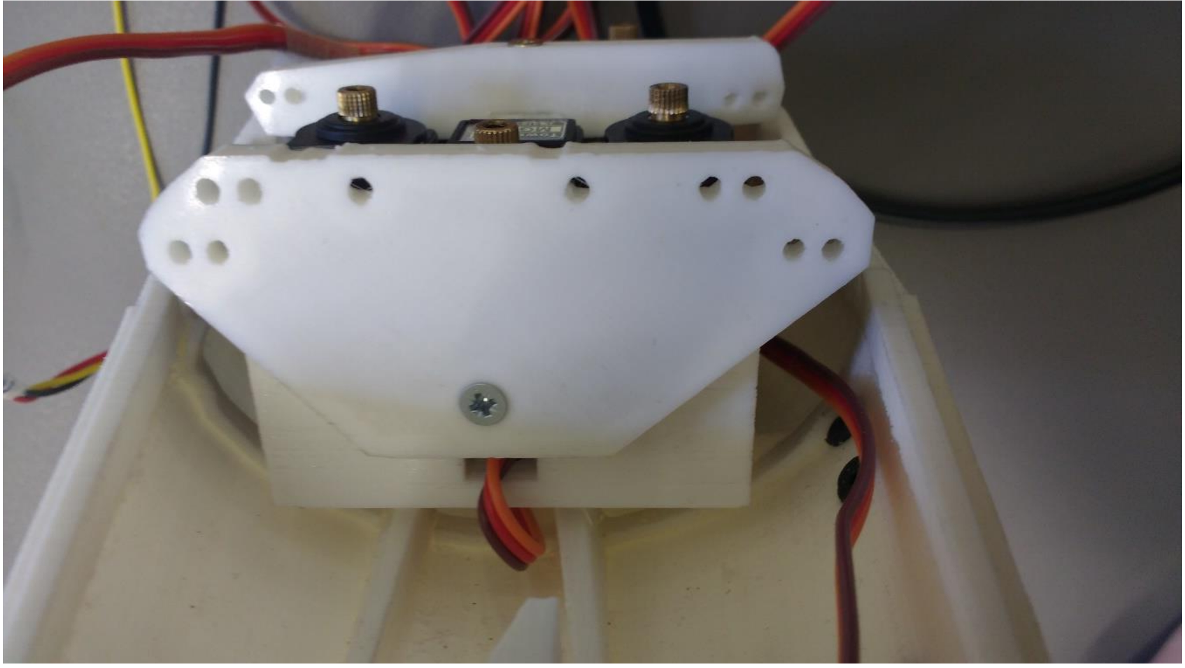
1. Download the desired piece from the InMoov website, do this for as many pieces as desired. It was found that depending on the size of the pieces it could fit up to four schematics but the ideal is two to three.
2. Once downloaded open the Cura 3D Printing Slicing Software by Ultimaker, formatted for your chosen 3D printer, in this case a WitBox.
3. The program will show a virtualization of the printer’s tray.
4. Go to File and Load Model, a file browser will open, add your chosen schematics.
5. These schematics should automatically load evenly distributed on the tray virtualization. Now edit the printing characteristics according to the InMoov instructions by changing the shell thickness and the interior density, check any warnings given by the program. This will affect the printing time accordingly.
6. Change the default temperature setting, ideally it was found that 210ºC was the ideal temperature to avoid malformations and excess of filaments during the printing process. This may vary with other printers.
7. Go to File and Save G-code, a file browser will open, save the G-code with a recognizable name to the media that will connect to the printer.
8. Prepare the printer for printing by having the tray sprayed with commercial lacquer or some other protective non-flammable spray.
9. Connect the chosen media to the printer and execute the G-code. Wait for the printer to finish, this could take anywhere from half an hour to the majority of the day.
10. Using a spatula or similar, separate the newly printed piece from the tray.
11. Repeat for as many pieces that are needed.

Once a significant portion of the necessary pieces were printed, the process of sanding and polishing off any surface defects from the pieces must be done to assure a correct fit. This was a critical step as some of the pieces have interior supports that must be removed to allow the piece to properly form without collapsing on itself. Also due to errors in the temperature control various pieces had to be sanded off excessively to remove a lot of plastic filaments that form during the printing.

Once the pieces were corrected, the process of putting them together could begin. Starting with the main structural pieces, the forearm casing. The forearm casing contains the servomotors and the cables that would lead to each individual finger. It consists of two pairs of pieces which are glued together into the back and front of the forearm, within one of these the servomotor bed is attached via glue and screws, while the other functions as both an aesthetic and structural cover. Also important is guiding the cables so that they all exit via the non-hand side of the forearm.



(Figure 5: Servo bed from above, note the cable positioning)



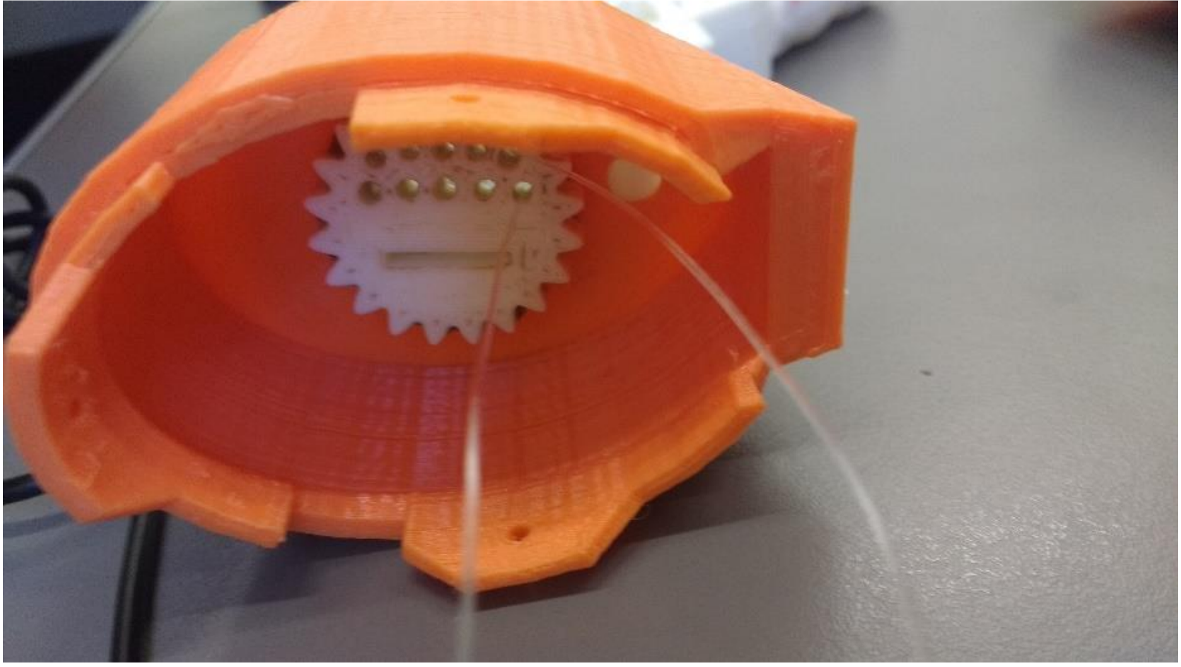
(Figure 6: Servo-bed from the hand side of the forearm. Note the cable guide holes)

From this base the construction progressed towards the hand, leaving the wrist for later. The hand consists of the palm, the back-hand plates, five sets of six fingers pieces, the palm base of the pinkie and ring fingers and a series of plastic bolts. All of these pieces, besides the plastic bolts, have a series of holes running through to allow the control cables to pass through.

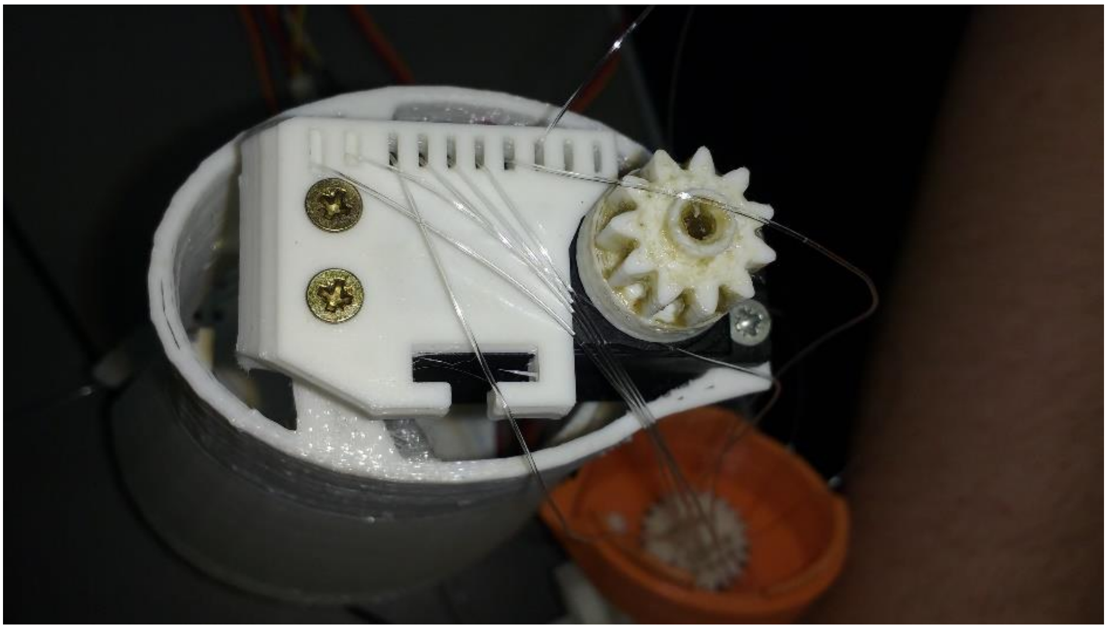
The first thing that had to be done to build the hand was assembling the finger piece sets, the finger sets could be glued together to form sets of three pieces; the finger-tip, the mid-finger and the finger base. Each of these has the space for two cables to pass through and be capped off at the finger-tip.

Once the fingers were done assemblage could be moved to the palm, where the first difficulties were found. The palm itself was a very solidly printed piece but it had the problem of having three sets of loops through which bolts had to be driven through. As such finding a way to make the loops larger without breaking them or the bolts proved difficult. This fact brought up the construction time significantly and broke a lot of printed bolts until the solution was found. Finally used a hand drill and some lubricant to ease the bolts in the palm was finished. Once the bolts were attached the preliminary build could begin, starting by attaching the fingers and wrist joint to the hand.

After checking that the hand itself could be assembled without trouble, the structure was disassembled and prepared to install the cables. Before this could be done, the wrist had to be finished. The wrist consists of three large pieces, two small gears and a servo-motor. The wrist must be constructed before installing the cables as they must pass through it.



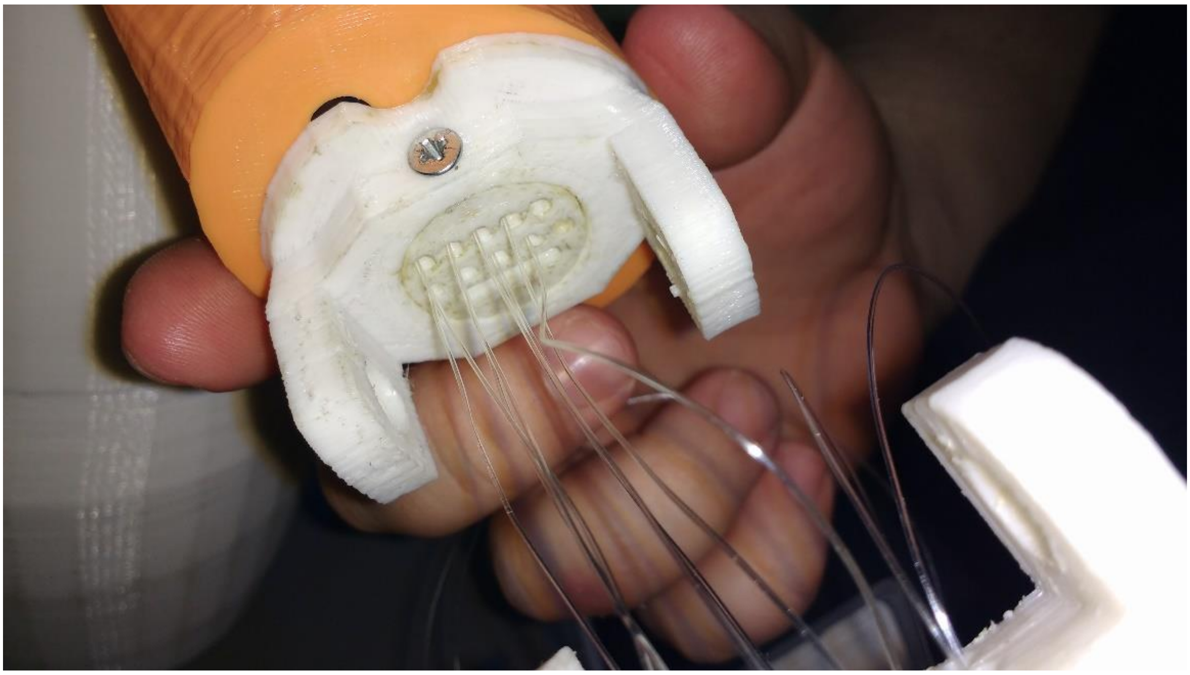
(Figure7: The top of the wrist with a gear)



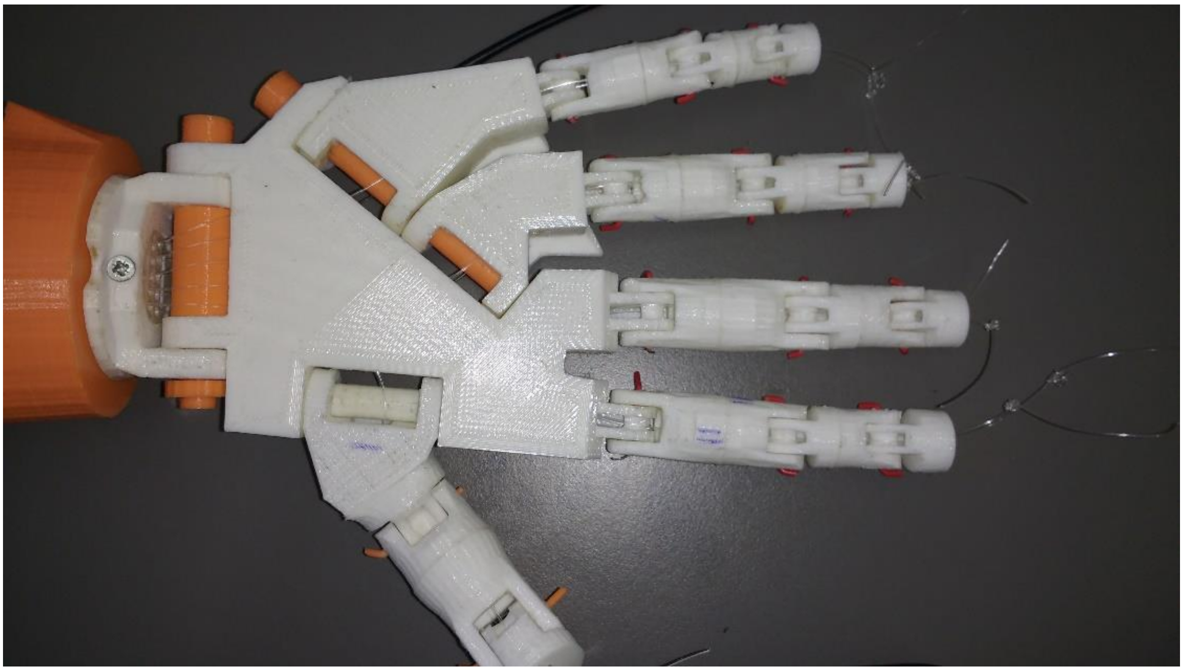
(Figure 8: The bottom of the wrist, the servo-motor, the cable guide piece and a gear covered in lubricant)

Constructing both the bottom and top wrist pieces with their respective gear pieces the construction can proceed on to the more complicated piece of the construction. The Installation of the control cables, there are a total of five pairs of cables, each of them associated to one finger and one servo motor. Each pair must pass through the two wrist pieces (the guide and the larger of the gears), through the wrist joint, go through their respective holes in the palm, go through their respective fingers and reach the finger tips without getting twisted. Each of these begins from the servo on the servo bed and is put through each of the aforementioned pieces.

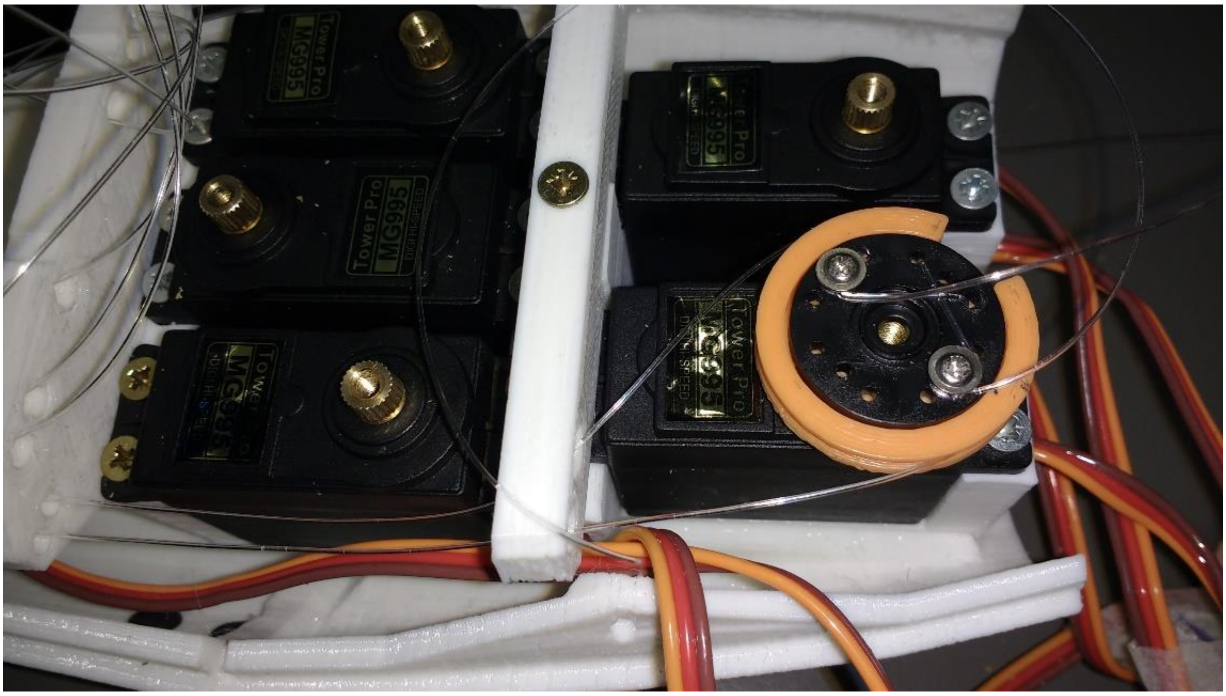
Starting from the servo the cable must be first attached to the specially printed servo-rotors to the end of each cable. These circular rotors hold one end of the cable and will later be used to tighten the cable to allow for fluid and exact movement. The cable then is passed along the guide holes on the servo bed and into the wrist, where it passes first through the guide pieces and through one of the holes found on the larger of the two gears, pictured previously. Having passed through the forearm it must go through one of the three holes next to the wrist joint and into the corresponding finger. This must be done twice per finger making sure that they do not get twisted. Once a finger has both cables added the servo-rotor is attached to a servo-motor put at the 90º position, this will be described as the neutral position, via the program provided by InMoov. Once the cables have been put through all five fingers, the hand was reassembled with the bolts and the finger joints.



(Figure 9: Cables put through the wrist joint)

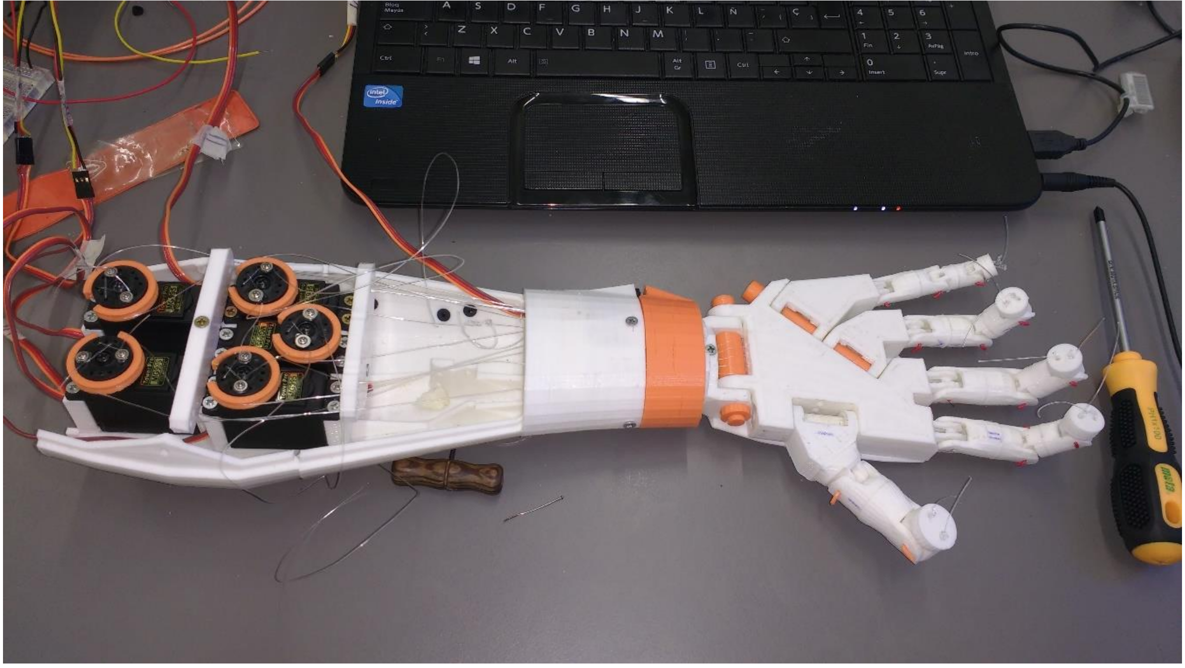


(Figure 10: The cables installed and the hand reconstructed)



(Figure 11: A Servo-rotor wheel fully tightened and attached to the servo-motor, all attached to the servo-bed.)

Once the servo-rotor is attached to the motor, the rotors must be turned to 0 degrees and the cable tightened. This has to be done until the finger is at its maximum position for that side, then the same must be done 180 degrees. Afterwards the same is done for the other four fingers. Afterward just test that the servos all work individually and then find a way to power both the Arduino and the servos at the same time, this is done via a AA-size DC battery power pack from an old battery powered toy.



(Figure 12: Finished Hand with tightened cables, note neutral position)

3.1.2 Mistakes and Errors

During the process of printing and building the hand there were several notable setbacks to the project, most of them related to the process of installing the cable control system and with the printed bolts.

The cable control system is a tricky installation involving most of the pieces found in the arm, as such that complications should arise were inevitable, luckily by consulting with the previous year's project group the most obvious error of using an unsuitable cable material was avoided. The previous group recommended the use fishing line instead of any other threaded or easily snapping material. Unfortunately, even with the other groups warnings there were still problems with the hand.

One particularly noteworthy problem was in capping off the cables once they passed through the hand. They had to be tied up at the end of the fingertip without slipping or letting it unravel. This proved complicated due to the size of the knots involved and the somewhat brittle nature of the fingertips material.

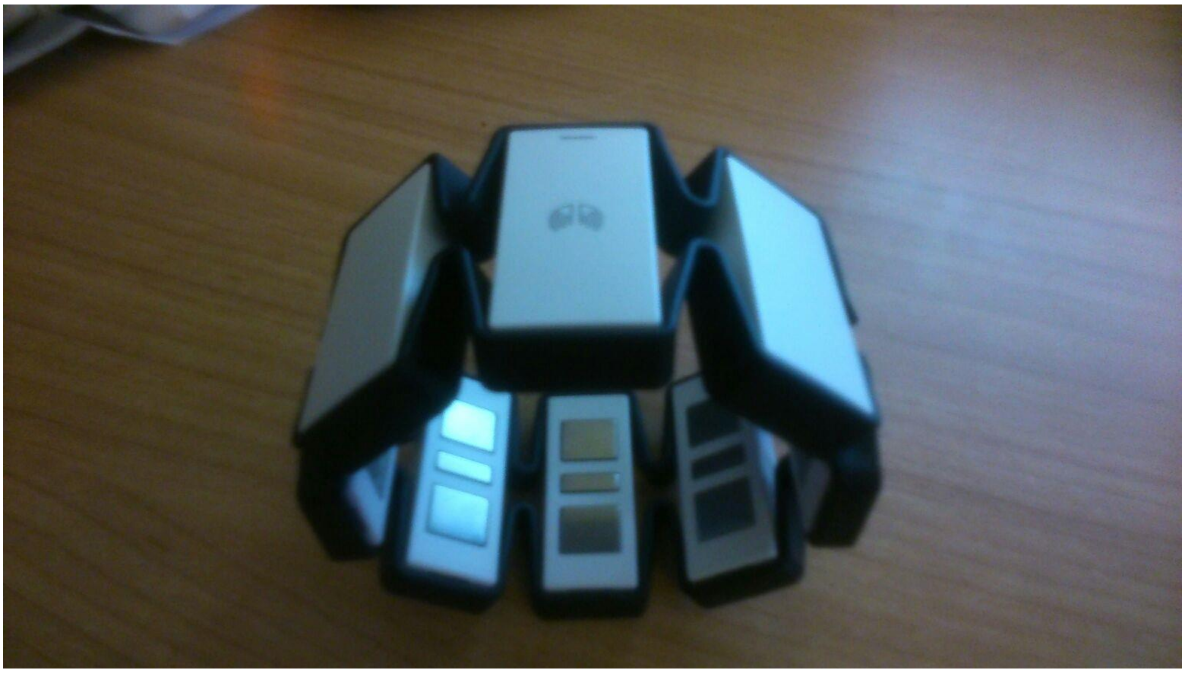
Other issues found, was the fact that the instructions were rather unclear on how the servomotors had to be positioned when the cable had to be tensed to their maximum and minimum positions. This was eventually figured out through trial and error. A final difficulty was the final tightening as the servo-rotors required the use of two screws to tighten the cables and could if not done carefully cut the cable with their edge.

3.2 Myo Sensor & EMG[15] [16]

In this chapter the writing will be explaining the physiology of the hand and the use of EMG sensors in detecting its movements, along with the explanation for the tool being used, the Myo Sensor by Thalmic Labs. Unlike the InMoov section from here onwards the work is individual.

3.2.1 Processing the EMG signals with the Myo Armband

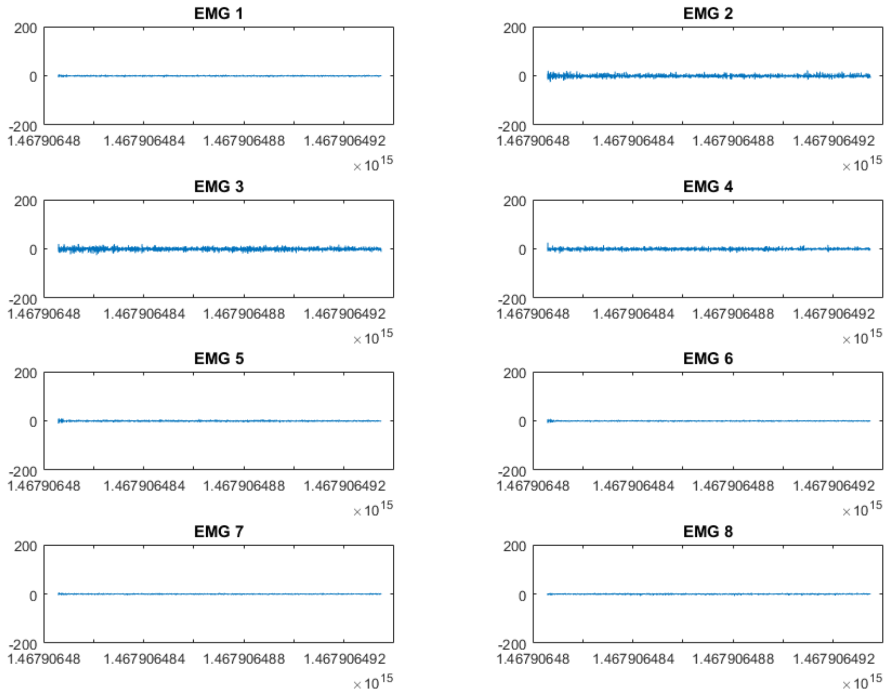
It is important to note that this project does not include the processing of the signal itself within the code, beyond that of the accelerometer's roll, instead the system is using the Myo's own incorporated system to identify which gestures are being detected. The Myo armband's firmware by itself, without computer support, can identify a series of pre-set gestures via the 8 EMG sensors it carries, this data is compacted into a readable Bluetooth signal for the computer. To get the raw EMG data, the option must be activated from the computer itself manually as it does consume much more battery life. The EMG signal is processed within the armband itself.



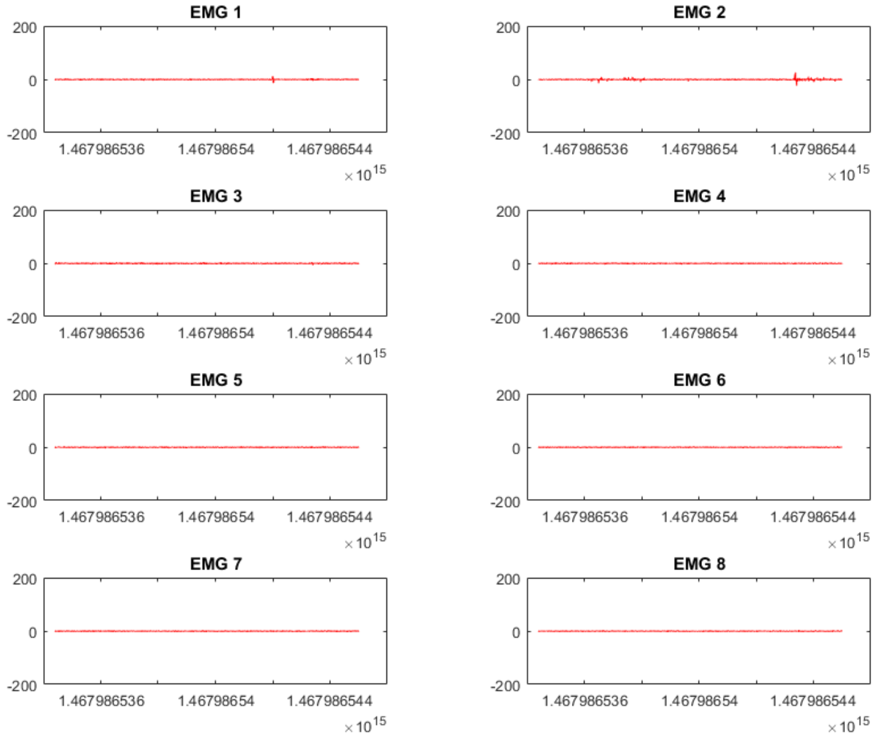
(Figure 13: Myo Armband System, not pictured Bluetooth USB Dongle)

From the activity detected by the EMG the extrapolation of what parts of the forearm's muscles are being detected to make the gesture recognitions can eb done. The Myo armband's instructions of use recommend putting the armband around the widest section of the forearm and that it has a tight or snug fit around it.

Below are the raw EMG signals detected by the Myo Armband for the gestures, as well as the default rest position, the only changes done to the data is the fact that it's been amplified for better visualization. Using the following graphs it is possible to see how the EMG process the signal to some degree, mainly by comparing the comparative signals between each of the sensors and from there extrapolate which gesture is the most probable to create said signals.

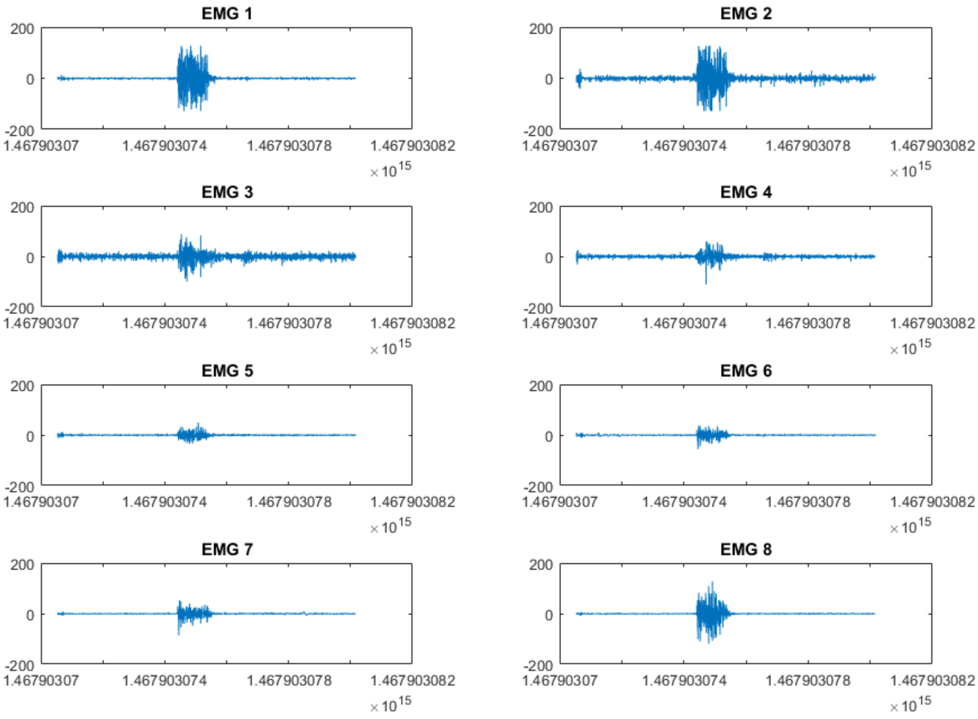


(Figure 14: EMG signals during rest, first user)

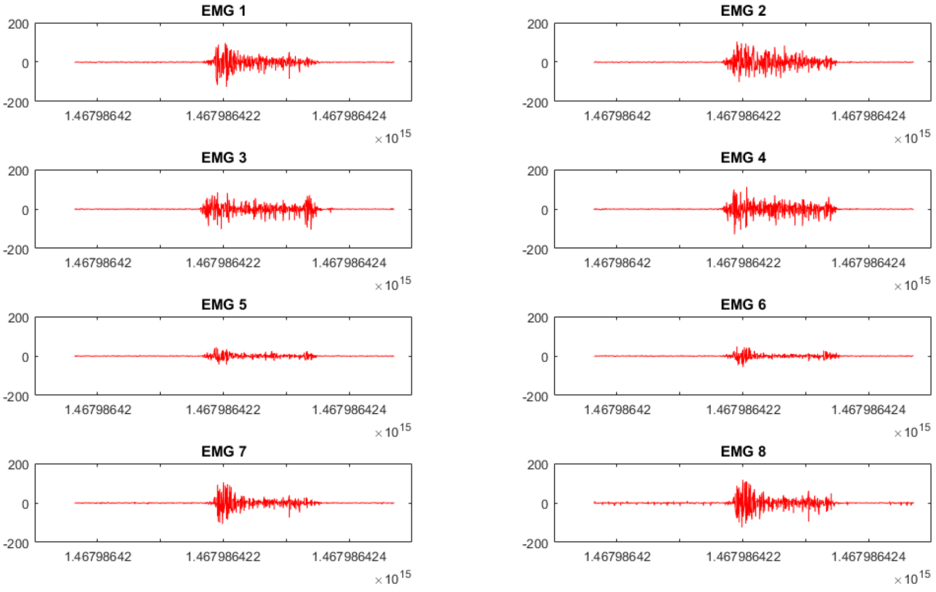


(Figure 15: EMG signals during rest, second user)

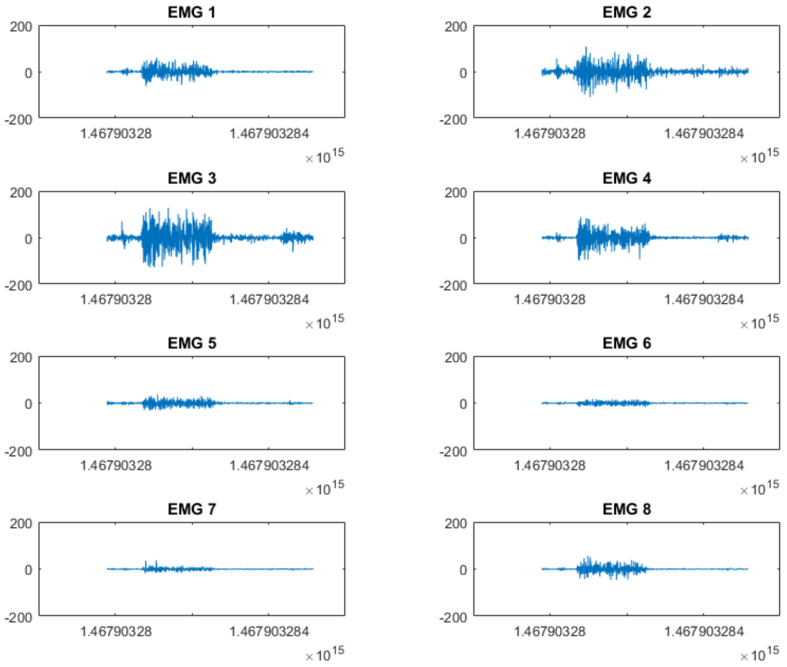
The above graphs represent the Myo sensors output while it is at rest, as can be seen the fluctuations are minimal and hardly noticeable. In the next one on the other hand the signals are very distinct and clearly has an effect on all the sensors.



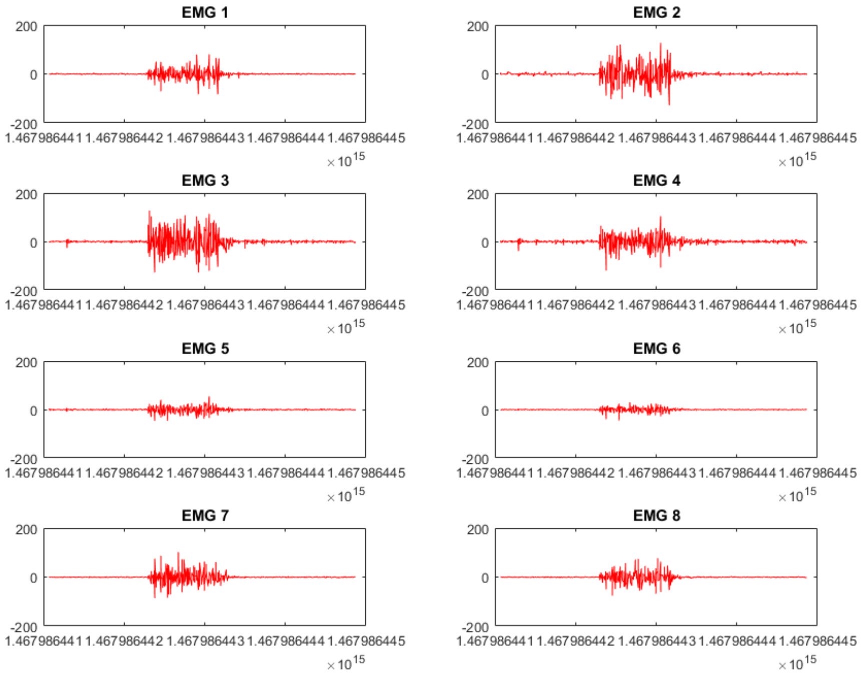
(Figure 16: EMG signals detected during fist gesture, first user)



(Figure 17: EMG signals detected during fist gesture, second user)

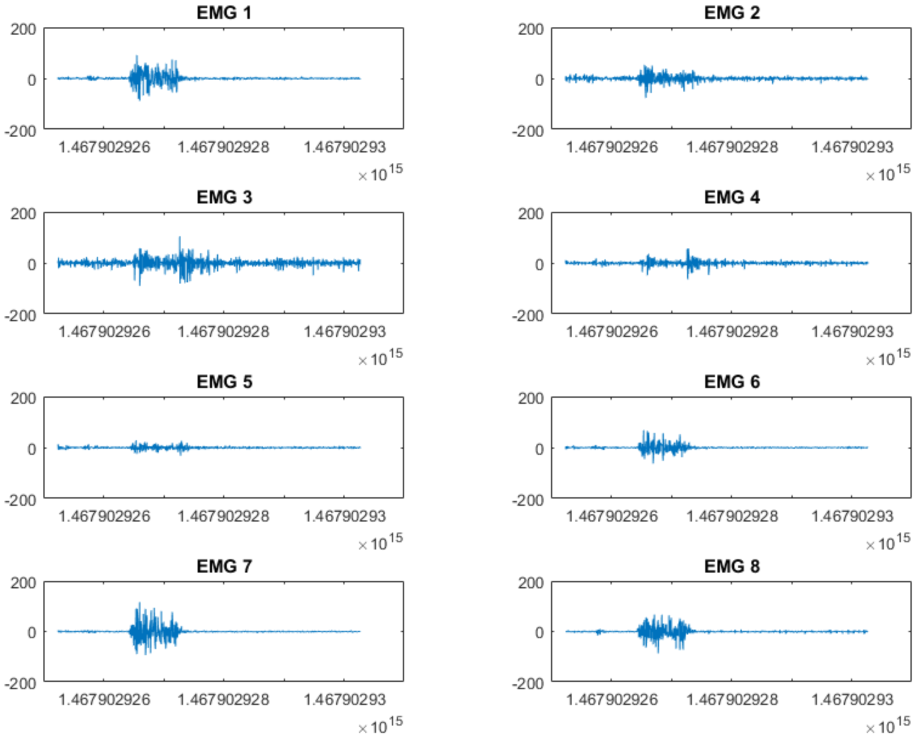


(Figure 18: EMG signals detected during extended fingers gesture, first user)

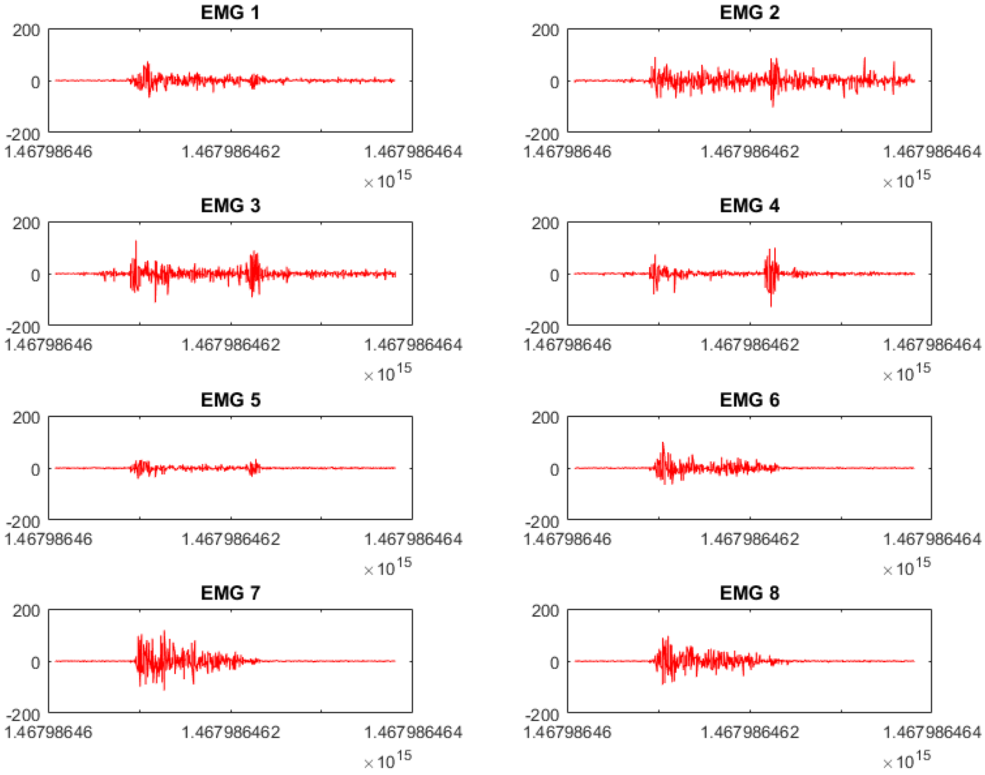


(Figure 19: EMG signals detected during extended fingers gesture, second user)

Unlike the previous fist diagram these sets of graphs have a more prolonged period of action before detection, although half the sensors note a markedly reduced amount of activity detected.

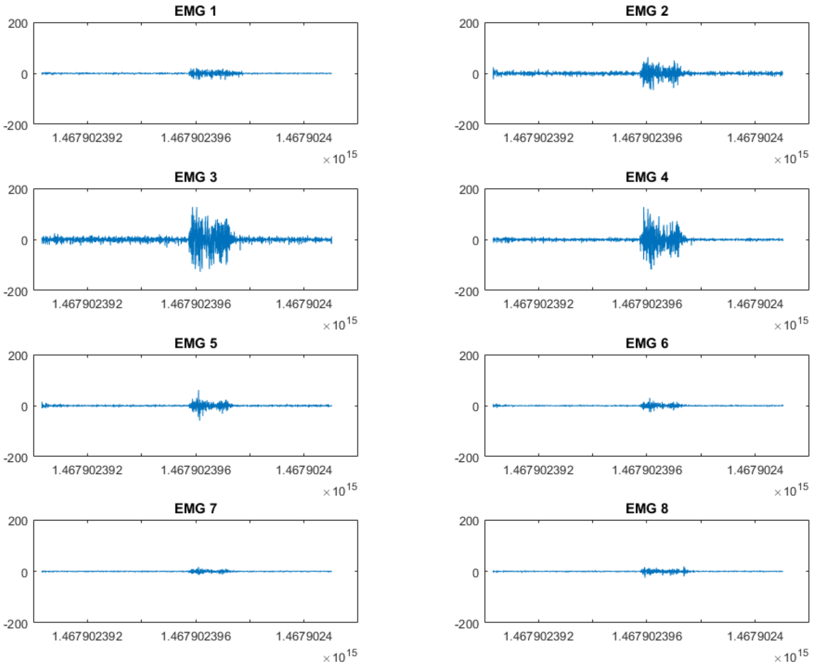


(Figure 20: EMG signals detected during wave in gesture, first user)

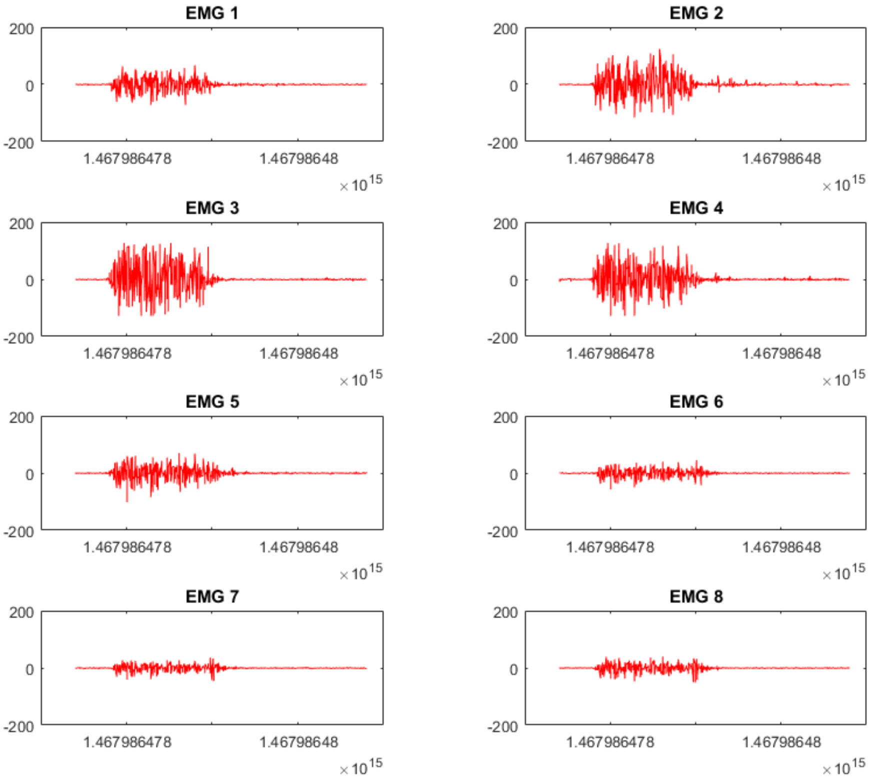


(Figure 21: EMG signals detected during wave in gesture, second user)

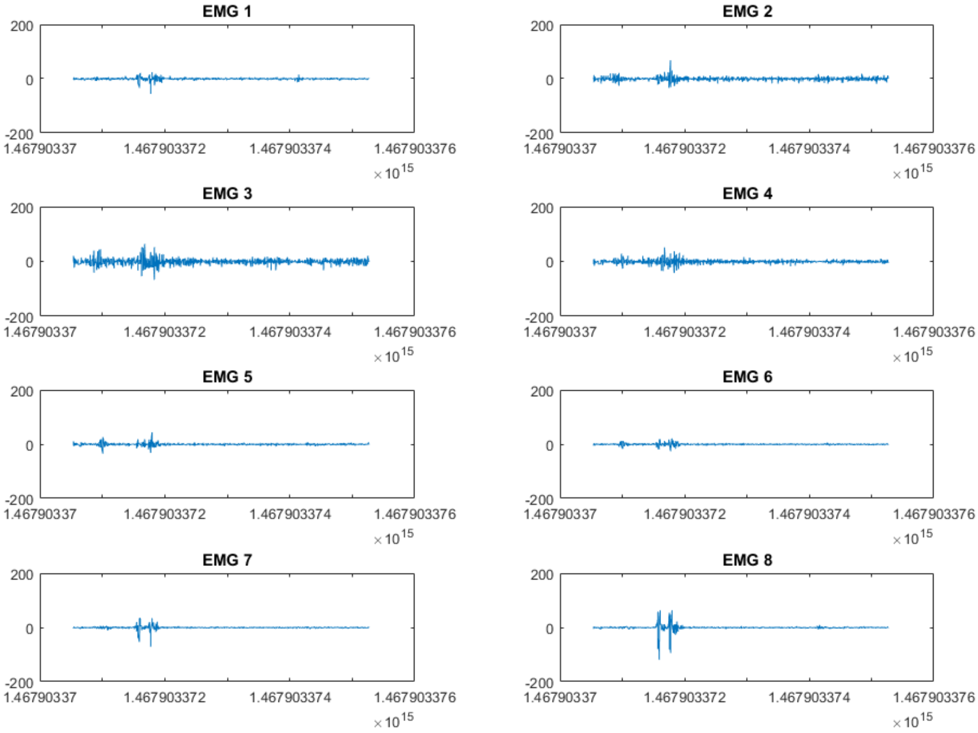
Similar in scope to the extended fingers gesture it can be remarked that the signals intensity are the reverses of each other, approximately in the cases of the wave in and wave out gestures.



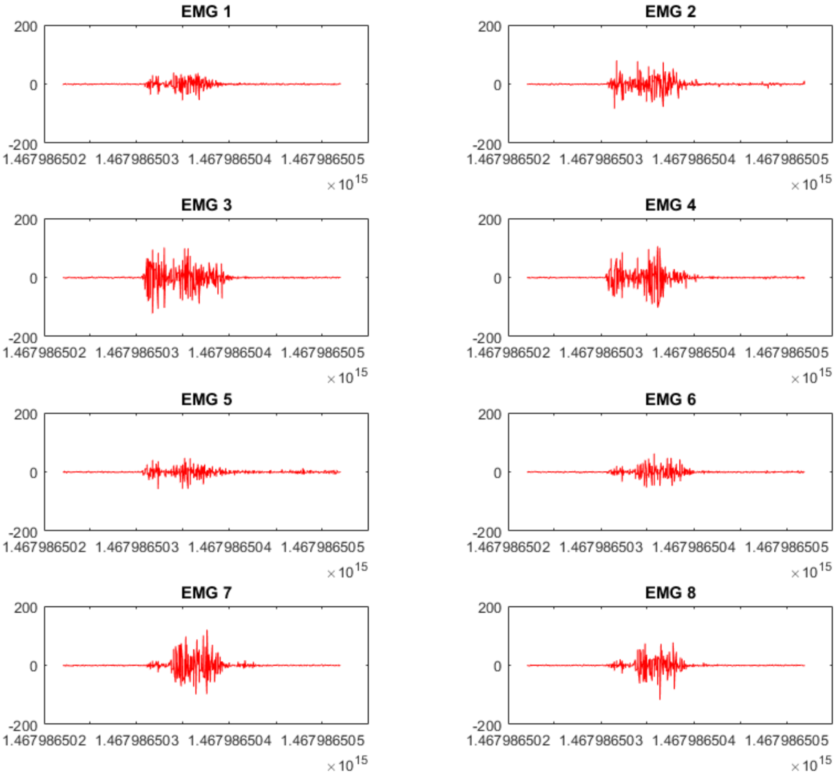
(Figure 22: EMG signals detected during wave out gesture, first user)



(Figure 23: EMG signals detected during wave out gesture, first user)



(Figure 24: EMG signals detected during double tap gesture, first user)



(Figure 25: EMG signals detected during double tap gesture, second user)

Finally, the double tap gesture is notable due to the difficulty in its detection, due to its similarity to the rest position and only having two peaks more or less uniformly distributed throughout all the sensors.

3.3 MyRobotLab & ROS

In this section of the project the writing will be talking about the robotic interface that was used to transmit and control the InMoov arm. In particular, the paper will be talking about the interface offered by the InMoov creator, the MyRobotLab and the one recommended by the project tutor, the Robot OS (ROS). Along with this it will also explain about the control mechanisms put into action within the Arduino board used to control the InMoov servo movements.

For the project itself it was decided that the use of the ROS was preferable due to the more comprehensive program and interface distribution as well as the clarity of their tutorials and wikis.

3.3.1 MyRobotLab

The MyRobotLab interface is the environment proposed by the creator of the InMoov to control it and develop programs on. The MyRobotLab itself was initially created by user ‘GroG’ (Pseudonym, unknown actual name), it is a Java service based framework specialised for Robotics and various forms of machine control. This includes various other robots along with the InMoov, along with various tools to interface with various plugins and programs. These tools range speech recognition, machine vision/image processing, servo control, motor control, graphic interfaces and various form of microcontroller communication services (such as Arduino, Raspberry Pi and others).

The MyRobotLab controls robots and other systems via either being installed on the robot itself. In the case that it carries an on-board computer the MRL (MyRobotLab) can be installed straight on. Otherwise it can be controlled via Microcontrollers allowing communication by cable, radio as found in most RC toys, Infrared, Bluetooth or wirelessly via TCP/IP.

In the case of Arduino, instead of just loading an action code to the Arduino it turns it into a pure I/O machine between the computer and the robot. Said code to implement this can be converted to other microcontrollers fairly easily, and is already done for several others.

Other capacities are the capacity for two computers with MRL installed with different services implemented can connect to each other, showing what services are being used on both sides. This lets various computers be they on-board or just controllers to work in tandem with one another.

Finally, there is an ongoing project from the creator (GroG) to implement a future purely graphic interface allowing for simple plug and control use of the interface, although more complex projects would require programming said behaviour. It would also allow web page applets to be created.

Turning back to the topic of the InMoov robot, within the list of various tools there already existed a control packet for the InMoov. In fact, there are various different service packets for the distinct sections of the robot, from the arms to the head, torso, and arms. These also came with the software example which allows for voice command control of the InMoov, allowing for the robot to be commanded to make various pre-programmed gestures. From the simple start of this service the system can control the various servos to set them at certain angles in near real-time. This first requires loading the I/O Arduino code into the InMoov board but it is very useful in making the first adjustments in controlling the robot.

Along with the InMoov services, their also came an in-program Myo device listener, but unfortunately this system had issues interfacing with the current Myo firmware and thus was never successfully used or tested within the project.

Other tools that could be used by the interface was a graphic interface that allows one to see the various connections between the different services and see what exactly is happening between the various pieces, customising it to some degree.

3.3.2 ROS within the Project

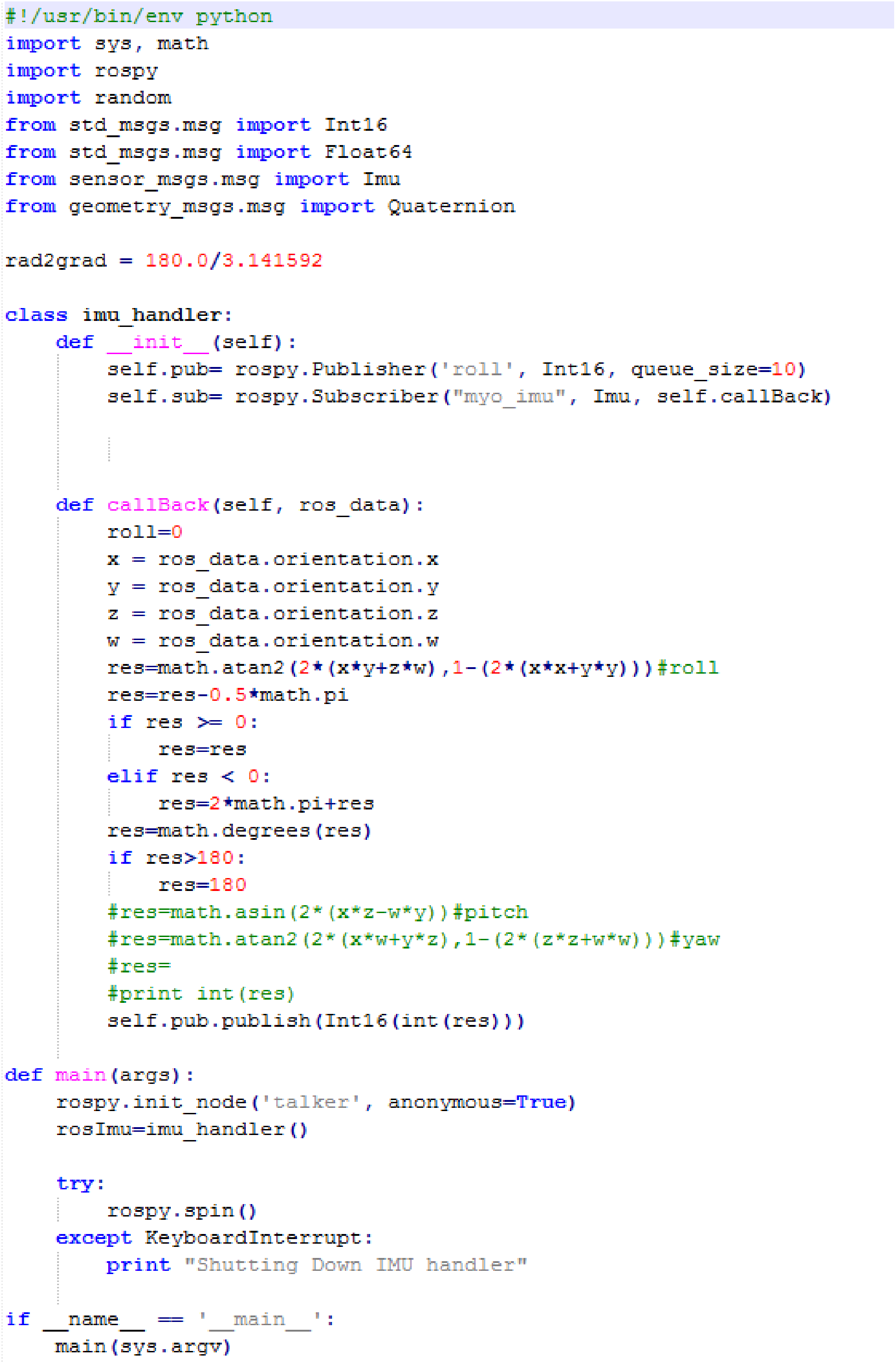
To achieve the project's objectives, the first task is to research what tools were available within the ROS system. In particular, the main issues that were had with ROS were three-fold, how to access the information from the Myo sensor from the ROS environment, how to interpret the information of the Myo sensor and how to transmit the data so that the arm may move.

The first of these problems was examined starting from the Myo website which offered a Myo app market, which had various examples of Myo used to interface with various devices. In particular, exploring the use of the Myo Arduino app and several others that involved the communication with various robots. Unfortunately, most of these applets functioned separate from ROS or were impossible to execute outside of a Visual Studio environment making them untenable for the project. In the end, an exploration was taken on the ROS community to see if there had been any previous work with the Myo and ROS. In this case there was more successful finding the 'ros_myo' package, which implemented a ROS node which takes the raw EMG data from the Myo and publishes four ros-topics:

- 'myo_arm', which publishes two unsigned 8-Byte integers, the first of which says on which arm the armband is on and the second says if the direction of the x-axis (used in positioning) is facing the elbow or wrist.
- 'myo_emg', the eight raw EMG data values stored within an array of 16-Byte integers. This data can be used to directly interpret the data of Myo from ROS. (In this project this was not done due to time constraints, but the process was studied previously in Chapter 2.)
- 'myo_gest', an unsigned 8-byte integer which identifies a series of predetermined gestures from the armband, in particular: the rest position, making a fist, waving inwards, waving outwards and tapping the thumb with another finger (the middle finger or the pinkie work best).
- 'myo_imu', this topic publishes an Inertial Measurement Unit as defined by the ROS sensor_msgs dependency. This contains the orientation, angular velocity and linear acceleration that the armband is undergoing.

Using the 'ros_myo' package the system could find the gestures that a hand was doing, as well as approximate the current roll position and transform it into an angle for the wrist movement. Thus solving both the first and second problem.

Although in the case of the wrist movement a new had to be made own package called 'imu_ros'. This package implemented a node (in python) which would subscribe to the 'myo_imu' topic, read the IMU object, extract the quaternion data. Using Euler transformation to get an angle between 0 and 180, it then publishes a new topic called 'roll' which contains said number.



```
#!/usr/bin/env python
import sys, math
import rospy
import random
from std_msgs.msg import Int16
from std_msgs.msg import Float64
from sensor_msgs.msg import Imu
from geometry_msgs.msg import Quaternion

rad2grad = 180.0/3.141592

class imu_handler:
    def __init__(self):
        self.pub= rospy.Publisher('roll', Int16, queue_size=10)
        self.sub= rospy.Subscriber("myo_imu", Imu, self.callBack)



    def callBack(self, ros_data):
        roll=0
        x = ros_data.orientation.x
        y = ros_data.orientation.y
        z = ros_data.orientation.z
        w = ros_data.orientation.w
        res=math.atan2(2*(x*y+z*w),1-(2*(x*x+y*y)))#roll
        res=res-0.5*math.pi
        if res >= 0:
            res=res
        elif res < 0:
            res=2*math.pi+res
        res=math.degrees(res)
        if res>180:
            res=180
        #res=math.asin(2*(x*z-w*y))#pitch
        #res=math.atan2(2*(x*w+y*z),1-(2*(z*z+w*w)))#yaw
        #res=
        #print int(res)
        self.pub.publish(Int16(int(res)))

def main(args):
    rospy.init_node('talker', anonymous=True)
    rosImu=imu_handler()

    try:
        rospy.spin()
    except KeyboardInterrupt:
        print "Shutting Down IMU handler"

if __name__ == '__main__':
    main(sys.argv)
```

(Figure 26: talker.py code from ‘imu_ros’ package)

Now it comes to the third and most important problem which is communication with the arm. As mentioned previously the InMoov arm is controlled via Arduino, which in turn controls the servo motors, which govern the movement of the fingers and wrist. To do this the various options must be explored, again. Principally the serial Arduino communication that exists within the normal Arduino code, but this was insufficient for what the system required since it lacked a good way to interface with the ROS node system, but it served well to know what was needed. Once again, looking through the ROS community for previous work with Arduino Serial communication and the ‘rosserial’ package was found.

The package consists of two parts; the ROS package itself to be installed within the workspace and the ‘rosserial’ extension to be installed within the Arduino environment. Once the new package was installed and functioning the control code could be uploaded nto the arm’s Arduino and begin to test out the movement of the servos and the general sturdiness of the arm. The Arduino’s particularities will be mentioned further on.

Once all three pieces of the packages were installed and found to be functioning the system could begin to make the arm move according to what the Myo armband said. To do this the following had to be done.

1. Open a new terminal
2. Use the sourcing code to allow ROS commands to run from terminal.
   ```
   source ~/biomime/devel/setup.bash
   ```
3. Use the following command to start the ros core node, note: this is an obligatory step as it sets up the ROS environment on which the rest of the nodes work.
   ```
   roscore
   ```
4. Open a new terminal, use the previous source command from step 2.
5. Activate the Myo armband sensor reading node by typing the following command:
   ```
   rosrun ros_myo myo-rawNode.py
   ```
6. Open a new terminal, use the source command again.
7. Activate the IMU reader and translator to govern the wrist movement with the following command.
   ```
   rosrun imu_ros talker.py
   ```
8. Open the Arduino IDE and connect the arm’s Arduino to the computer, compile and upload the file: ‘arduinoInMoovControl.ino’. Note the port where the arm is connected.
9. Open a new terminal, use the source command again.
10. Activate the ROS to Arduino communication serial by using the following command, where [SerialPORT] should be the port that has been noted previously.
    ```
    rosrun rosserial_python serial_node.py [SerialPORT]
    ```
11. Everything should be active and the arm should move, check the position of the Myo armband and do a wave out motion until it vibrates, (this indicates that it has synchronized.)

3.3.3 Arduino IDE & Code

As mentioned in the previous section the Arduino is a necessary to control the hand as it handles the movement of the servo motors which cause the hand to move. To be able to interact with the Board it first has to be connected to the computer and interface with it. This is done with the Arduino IDE which is used to configure the connection between board and computer; define the communication port, load example code, update firmware, etc.

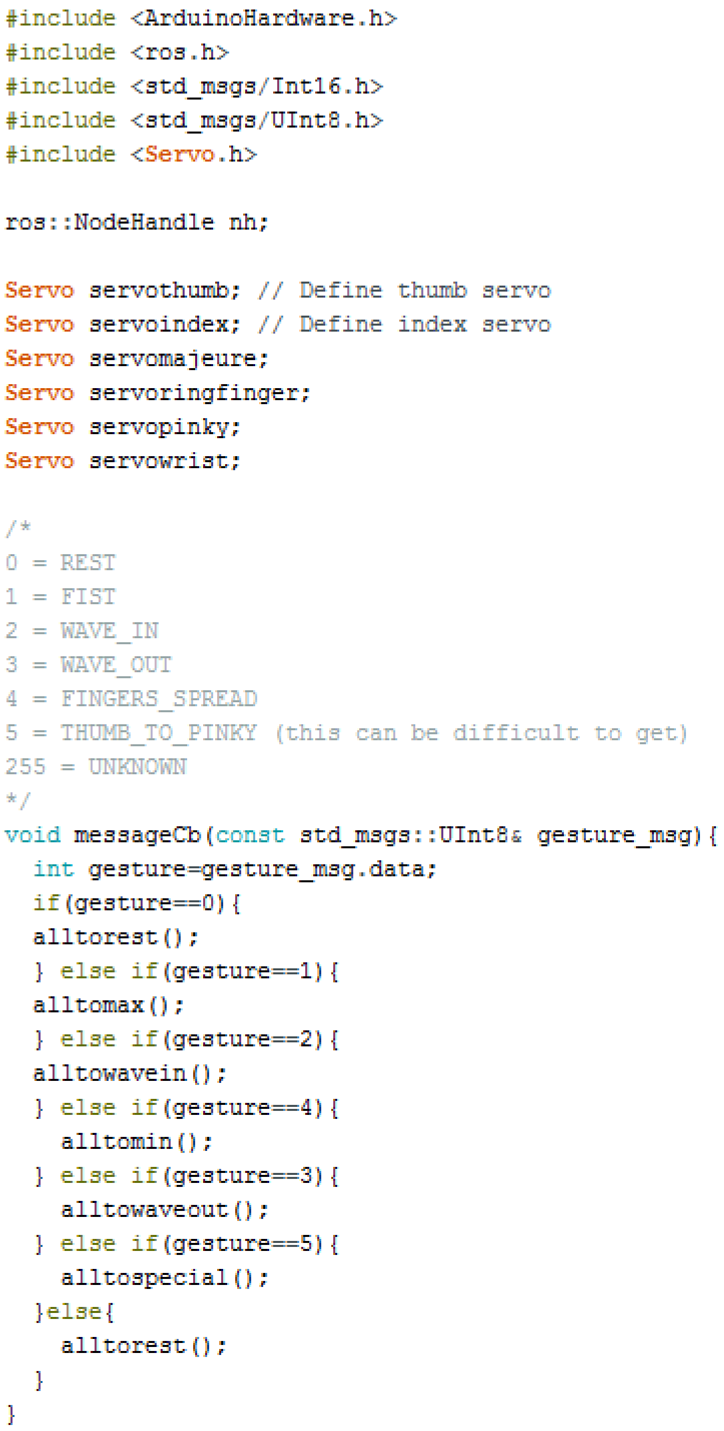
Once the Arduino IDE is installed on the computer (by downloading it from the Arduino website) the system can begin to work on the Arduino and the physical movement of the arm. At first, to check the correct functioning of the arm (and at the time of construction) the code from the InMoov website is used to make the servo-motors move, as well as test out the viability of the different gestures that the arm will be doing.[17]

With the movement and initial setup fully completed the actual Myo to InMoov biomimetic interaction can be looked into. The code uploaded in this case includes the previous one's gesture functions along with the code to read two topics from the ROS environment 'myo_gest' and 'roll'. Respectively, it will extract the gesture recognized by the 'myo-rawNode.py' node from the 'ros_myo' package for the former and in the latter case it will get the roll translated into an angle between 0 and 180 from the talker.py from the 'imu_ros' package. This is possible thanks to the afore mentioned rosserial package installed on the Arduino IDE, as well as the different 'msgs' or message types that are associated with said package, they define the different datatype formats that a topic can be published in. In the case of the Arduino, for every type of message that will be received by the Arduino via the 'rosserial', the corresponding header for that datatype must be included to allow it to know what it's working with.

Once the message headers are added to the existing code along with the ROS header, the node handler must be added along with its initialization code. The node handler object is defined in the same way as the servo objects from before only in this case the object comes from the ROS header and must be defined before all other objects after the headers. Afterwards within the setup function the node handler must be initialized.

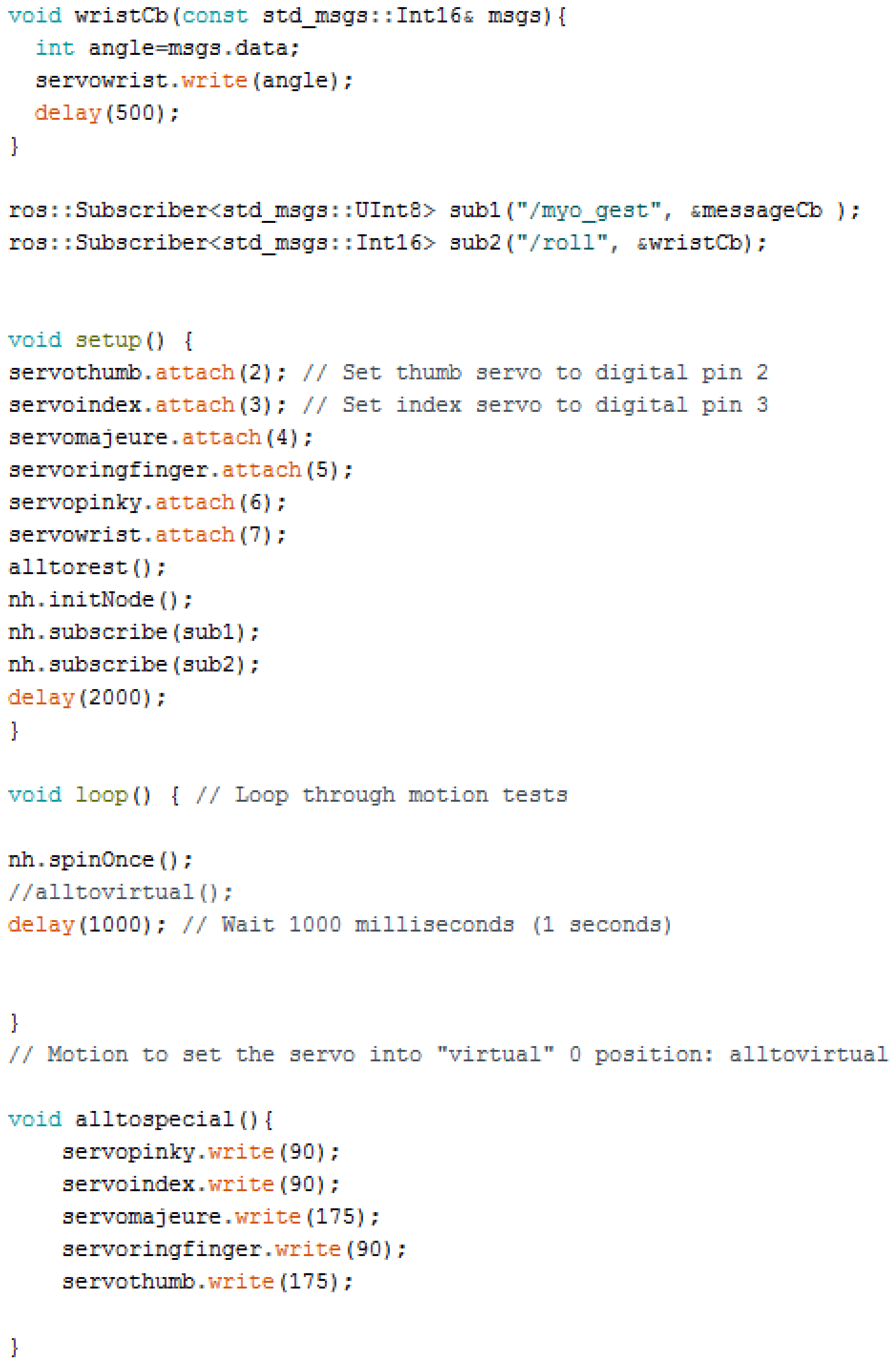
With the preliminary setup done the subscriptions can be defined. To define the subscriptions, the Subscription object must be called from the ROS header, along with the datatype that they will be operating within the definition. The object constructor function must contain thus, the topic to which the Arduino will listen to and the function it must call when it receives new data from said topic. In this case the topics will call 'messageCb' for the 'myo_gest' topic and 'wristCb' for the 'roll' topic. Each of these will operate with the data it has received.

In the case of 'messageCb' it will take the gesture identified and execute the associated gesture function. In the case of the 'wristCb' it will receive the angle and put the wrist servo at that angle. With both of these functions defined the subscriptions only need to be defined within the node handler within the setup function and then afterwards said node handler will spin once to read for said topics using the 'spinOnce()' function in the loop function.



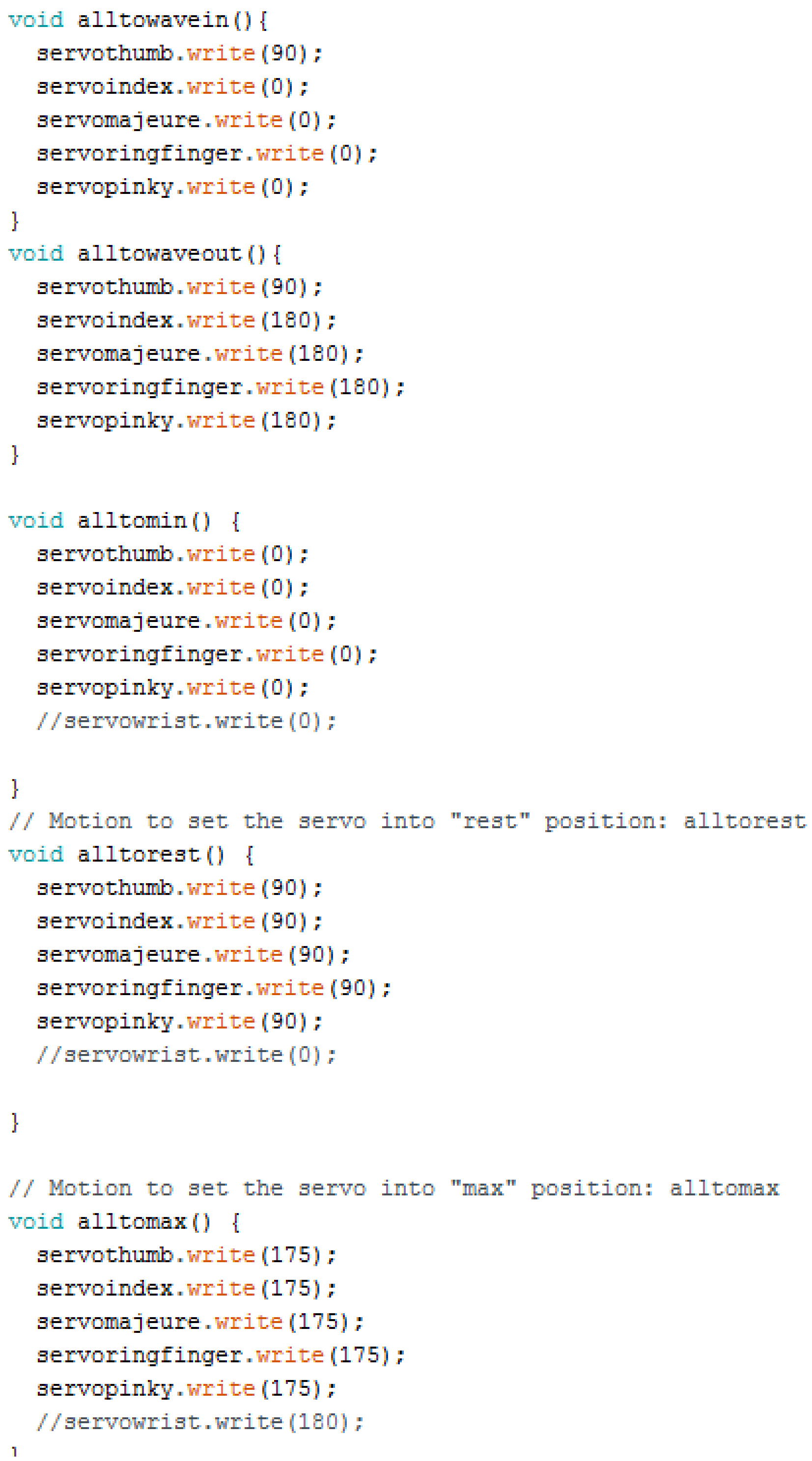
```
#include <ArduinoHardware.h>
#include <ros.h>
#include <std_msgs/Int16.h>
#include <std_msgs/UInt8.h>
#include <Servo.h>

ros::NodeHandle nh;

Servo servothumb; // Define thumb servo
Servo servoindex; // Define index servo
Servo servomajeure;
Servo servoringfinger;
Servo servopinky;
Servo servowrist;

/*
0 = REST
1 = FIST
2 = WAVE_IN
3 = WAVE_OUT
4 = FINGERS_SPREAD
5 = THUMB_TO_PINKY (this can be difficult to get)
255 = UNKNOWN
*/
void messageCb(const std_msgs::UInt8& gesture_msg){
  int gesture=gesture_msg.data;
  if(gesture==0){
  alltorest();
  } else if(gesture==1){
  alltomax();
  } else if(gesture==2){
  alltowavein();
  } else if(gesture==4){
    alltomin();
  } else if(gesture==3){
    alltowaveout();
  } else if(gesture==5){
    alltospecial();
  }else{
    alltorest();
  }
}
```

(Figure 27: myoInMoovControl.ino, part 1)



```
void wristCb(const std_msgs::Int16& msgs){
  int angle=msgs.data;
  servowrist.write(angle);
  delay(500);
}

ros::Subscriber<std_msgs::UInt8> sub1("/myo_gest", &messageCb );
ros::Subscriber<std_msgs::Int16> sub2("/roll", &wristCb);


void setup() {
servothumb.attach(2); // Set thumb servo to digital pin 2
servoindex.attach(3); // Set index servo to digital pin 3
servomajeure.attach(4);
servoringfinger.attach(5);
servopinky.attach(6);
servowrist.attach(7);
alltorest();
nh.initNode();
nh.subscribe(sub1);
nh.subscribe(sub2);
delay(2000);
}

void loop() { // Loop through motion tests

nh.spinOnce();
//alltovirtual();
delay(1000); // Wait 1000 milliseconds (1 seconds)


}
// Motion to set the servo into "virtual" 0 position: alltovirtual

void alltospecial(){
    servopinky.write(90);
    servoindex.write(90);
    servomajeure.write(175);
    servoringfinger.write(90);
    servothumb.write(175);

}
```

(Figure 28: myoInMoovControl.ino, part 2)

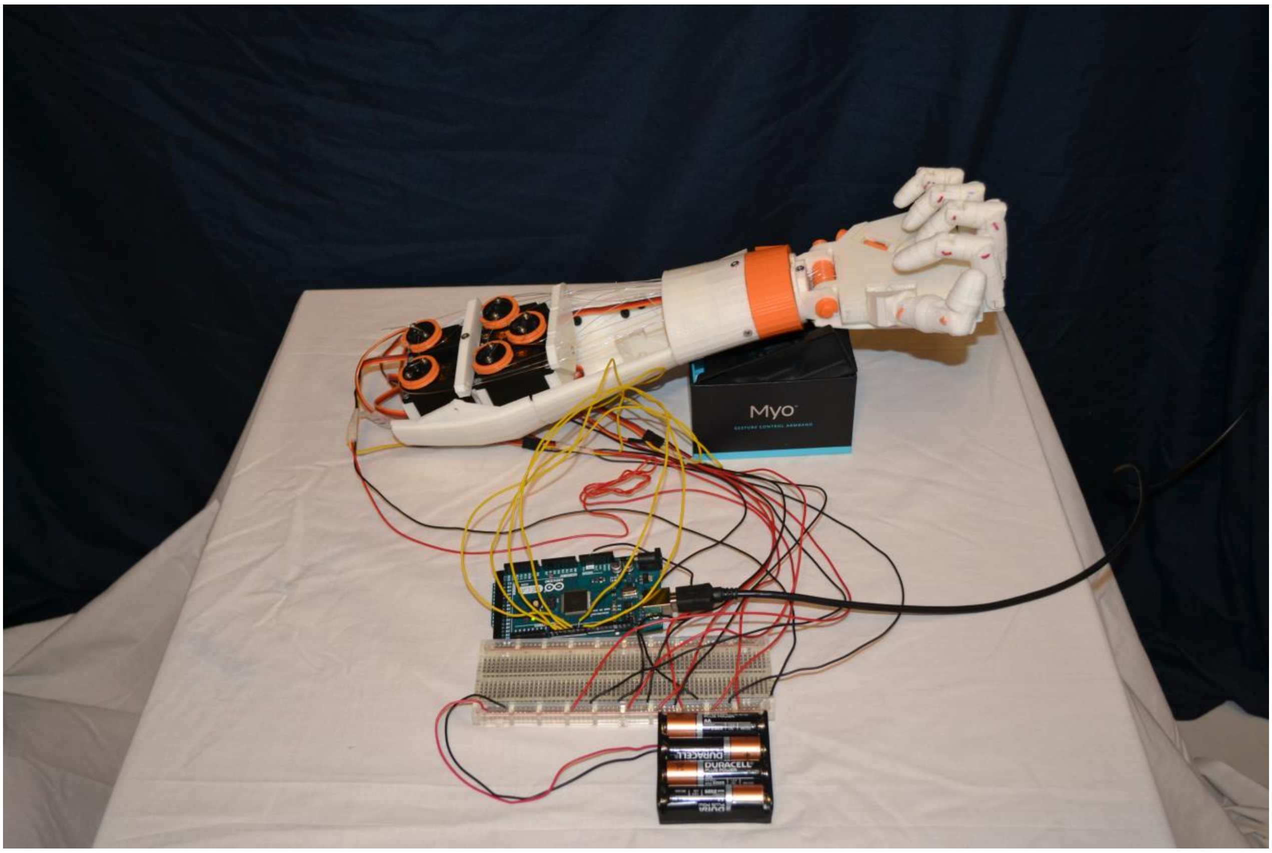


```
void alltowavein(){
  servothumb.write(90);
  servoindex.write(0);
  servomajeure.write(0);
  servoringfinger.write(0);
  servopinky.write(0);
}
void alltowaveout(){
  servothumb.write(90);
  servoindex.write(180);
  servomajeure.write(180);
  servoringfinger.write(180);
  servopinky.write(180);
}

void alltomin() {
  servothumb.write(0);
  servoindex.write(0);
  servomajeure.write(0);
  servoringfinger.write(0);
  servopinky.write(0);
  //servowrist.write(0);

}
// Motion to set the servo into "rest" position: alltorest
void alltorest() {
  servothumb.write(90);
  servoindex.write(90);
  servomajeure.write(90);
  servoringfinger.write(90);
  servopinky.write(90);
  //servowrist.write(0);

}

// Motion to set the servo into "max" position: alltomax
void alltomax() {
  servothumb.write(175);
  servoindex.write(175);
  servomajeure.write(175);
  servoringfinger.write(175);
  servopinky.write(175);
  //servowrist.write(180);
}
```

(Figure 29: myoInMoovControl.ino, part 3)

With the entire section of software related to the Arduino explained, the physical aspects related to the Arduino board itself can be explained. In this project the Arduino board used was the Arduino Mega 2560. The connection between the Arduino board and the computer is done via a USB port, which provides both the upload code and the initial electricity to power the Arduino. This is the basic part that would allow us to power one servo without burning out the board, but in this project six servos must be powered and processed by the board as such an external power source was needed. In this case an old DC battery rack from an old toy was used to power the Arduino and the servos, through a breadboard.

The breadboard is a long board of input connectors which in this case, using it to power the servos, bypassing the Arduino circuit to avoid burning it out. As such the connection is structured in the following way. Each servo has three cables, a control cable, a positive cable and a negative cable. The control cable (yellow) is attached to a control pin on the Arduino board. The positive and negative cables (red and black respectively) are attached to the breadboard along the sides, where there's two columns of connectors, one with a + and the other with a -. This is done for each servo motor and in the end the distribution looks like this.



(Figure 30: Photo of InMoov arm connected to Arduino Board, with all cable connections and power source.)

As pictured the power source is attached to the breadboard and thus the entire setup is powered up. An important note is that the Arduino board must also be connected to the breadboard via a grounding cable so as to also avoid burning out either board.

3.3.4 Mistakes & Errors

This particular area is slightly more difficult to evaluate the errors or mistakes due to the more subjective nature of programming, but there were some areas that were clearly more difficult to get right.

Beginning with the ROS environment there were several issues when it came to using the environment and setting up the executable files for each package, in particular there were several pathing issues. When calling a node function the command ‘rosrun’ must be used with it. Sometimes the command would search for the executables within the wrong section of the environment and changing the path proved complicated or prone to self-editing (changes were overwritten almost immediately). To solve this, the system’s environment had to be remade from the beginning and the packages had to be copied over from the old environment.

Within the Arduino IDE, the problems arose due to problems with the data format of the topics subscribed as well as the fact that the Arduino was subscribing to two topics, ‘roll’ and ‘myo_gest’. There were problems with the transmission of data through the rosserial node.

On the case of the hardware itself, the main issues arose with the amount of cables involved and how easy it was to confuse which cable went into the correct PIN. As well as the usual issues of strange behaviour when there was low battery.

4. RESULTS

In this section the writing shall be testing the final resulting product by testing the program and arm on various users and trying to see how well the system performs in completing the objectives. This will allow the quantification of how well the actual system performs as well as make observations on how well various users take to using it.

4.1 Testing

In this section the testing of the system will describe with the aim of checking the correct functioning of the entire product as such. This will do double duty as a stress test for the robot and a functionality test allowing for any defect were to be. The system was tested on 10 users. Each user would have to realize the gesture put forward and then the resulting gesture would be noted down, thus checking how well the system translated the movement to the arm.

The following instructions will describe the testing methodology for the project.

Aim:

- To find the Systems effectiveness in recognizing the gestures given.

Material required:

- The InMoov Arm with functioning computer connection
- The Biomimetic software project
- A set of 25 flashcards or similar containing 5 sets of 5 cards each with one of the recognized gestures.

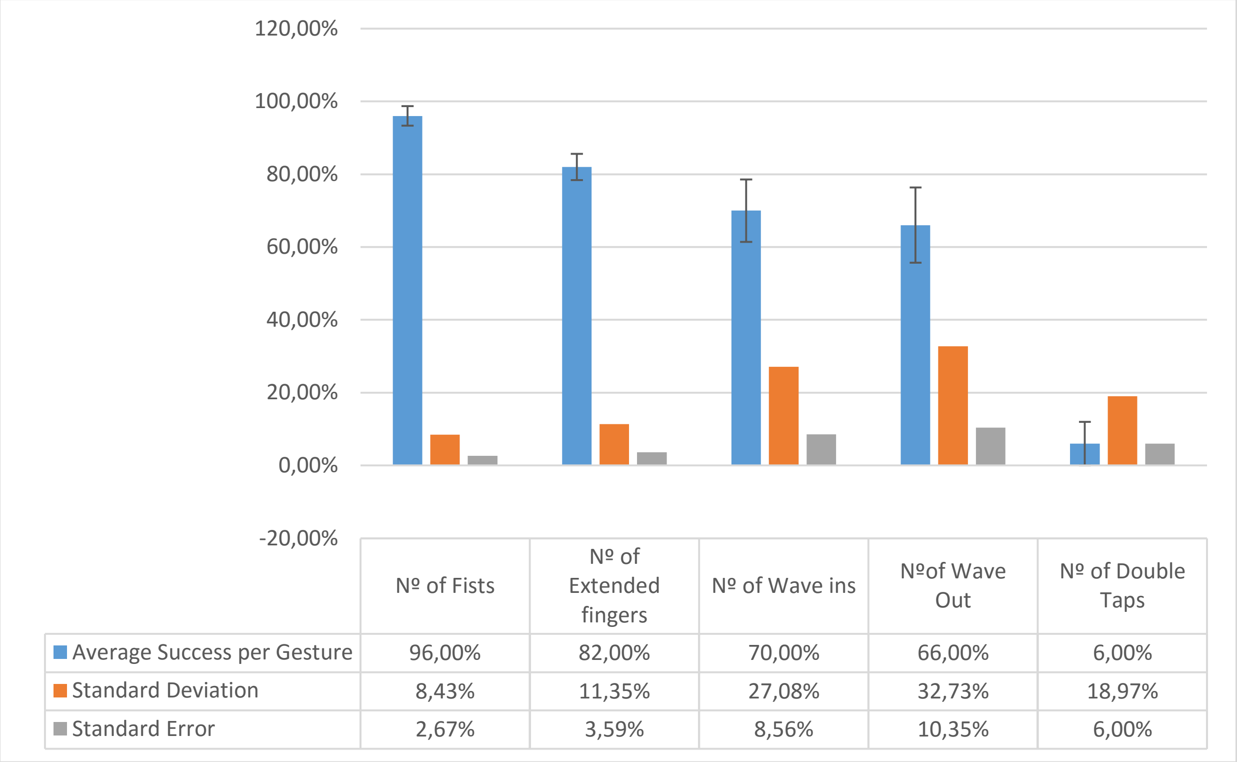
The test will function as follows:

1. The user must put on the Myo Armband and allow the program to synchronize.
2. The tester must shuffle the set of cards randomly.
3. The tester extract one card from the top of the deck of cards and show it to the user, the tester must take note of what gesture it is.
4. The user must make the gesture on the card once from a relaxed state and then return to a relaxed state.
5. Repeat until there are no more cards left in the deck.

Another possible test was the evaluation of the EMG signals themselves but after carefully looking through the Thalmic labs raw EMG display system it can be concluded that this test would not serve any productive function at this point in time, due to the fact that the EMG signals vary wildly depending on the user.

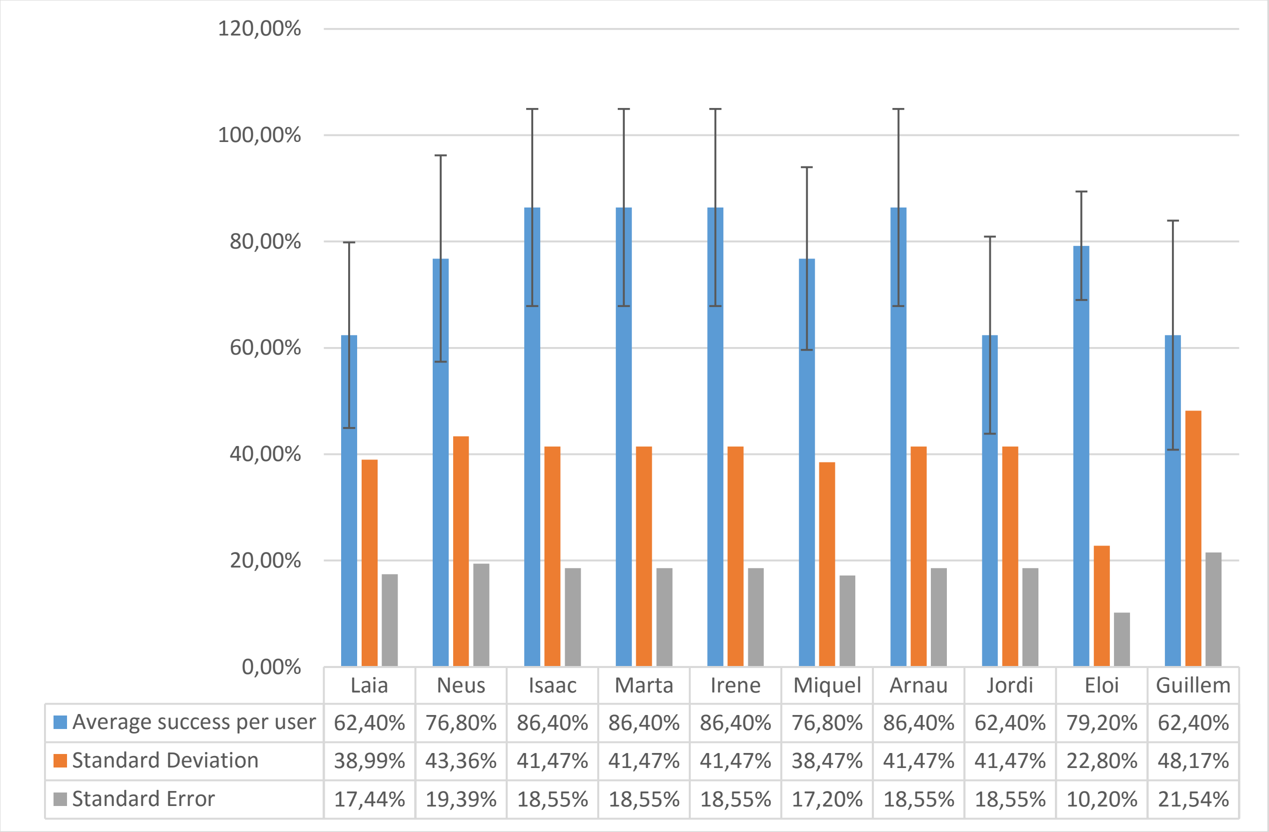
4.2 Observations on the tests

The test was run ten times on ten different subjects. Several users were noted to have very thin forearms which was in the past noted to affect some of the possible results of the Myo Armband. Using this data, it can be seen how well each type of gesture has been recognized as well as the associated standard deviation and error, visualized in the following bar graphs.



	Nº of Fists	Nº of Extended fingers	Nº of Wave ins	Nºof Wave Out	Nº of Double Taps
🟦 Average Success per Gesture	96,00%	82,00%	70,00%	66,00%	6,00%
🟧 Standard Deviation	8,43%	11,35%	27,08%	32,73%	18,97%
⬜ Standard Error	2,67%	3,59%	8,56%	10,35%	6,00%

(Graph 1: Average Success rate for gesture recognition, per gesture)



	Laia	Neus	Isaac	Marta	Irene	Miquel	Arnau	Jordi	Eloi	Guillem
■ Average success per user	62,40%	76,80%	86,40%	86,40%	86,40%	76,80%	86,40%	62,40%	79,20%	62,40%
■ Standard Deviation	38,99%	43,36%	41,47%	41,47%	41,47%	38,47%	41,47%	41,47%	22,80%	48,17%
■ Standard Error	17,44%	19,39%	18,55%	18,55%	18,55%	17,20%	18,55%	18,55%	10,20%	21,54%

(Graph 2: Average Success rate for Gesture Recognition, per user)

From these results it can be said that the gesture recognition software works on average, at a rate of 64% chance of correctly making the correct gesture on the arm, with a standard deviation of 34,47% and an error of 15,41%, this results are not ideal and are heavily biased due to the difficulty in performing the double tap gesture in a detectable manner. This is further corroborated by the fact that the double tap gesture was only recognized for one user, Eloi, who had previous experience with the Myo Armband.

These results could be further improved on by a more in-depth study on how the actual EMG is processed by the Myo Armband, but sadly the system functions as a black box on the actual recognition software it uses.

4.3 Comparison to Other Systems

This section will give a brief overview of how the system performs to other similar projects and biomimetic systems. In particular, centred on those that use the same tools, the Myo Armband, ROS, Arduino and InMoov robot. In the last case it is possible to observe the various experiments that Gael Langevin, the creator of the InMoov, has realized to further tests the robot’s capabilities, as well as his endeavours towards a new version of the robot.

Gael Langevin’s new experimental robot consists currently of a completetly redesigned hand with a self-contained set of controllers (they are contained within the hand itself instead of the forearm like in the version used in this project), as well as a more lifelike and ergonomic design. Unlike the design in the project this system uses the MyRobotLab environment and runs the InMoov packets found within. From there it interfaces similarly with an Arduino Board (an Arduino Nano), which is in fact incorporated within the structure. Like the current system it is also controlled by the Myo Armband from Thalmic Labs. Although the current amount of gestures possible with the MyRobotLab Myo interface is slightly more limited and reliant on more overt gesture to generate a response, the author notes that it has shown some promise and has proposed an idea to incorporate a Bluetooth component to the new InMoov hand design to allow for more direct communication between both systems. To date there hasn’t been more reports in this new system’s development, with the last news coming from February 2015.[18] Reportedly there had been some issues with the latency between the Myo and the Hand’s movement.[19] Notably the new InMoov Hand 2 is a purely prosthetic type project instead of the more educational design of the original, hence the more compact control system.

One of the earliest projects realized with the InMoov robot, and largely used as an example in using the MyRobotLab based control of the robot is a program that allows the InMoov to move according to voice commands. The MyRobotLab system contains a microphone system and also loads a controller code into the Arduino and depending on the voice commands it can recreate a number of different custom gestures or movements. The limitations to this system exist mainly in the form of the systems latency and the fact that it can only do pre-programmed actions, similar to this projects’ system, as well as the natural problems in audio recognition programs (noise, distortion etc.).

5. CONCLUSIONS

In this section will evaluate the final resulting product of the project as a whole, to see if the objectives that was set was accomplished. As such it will be evaluating each separate piece of the project independently and then as a whole. Also the difficulties and errors that occurred during will be explored.

5.1 Discussion

In this section the benefits, impacts and future perspectives that are possible with the current project as well as the current limitations (beyond those already explained) that may be improved on later on. This section exists as an expose on different possible ideas on future expansions of the current project.

5.1.1 Benefits & Impacts

This project has provided a fairly positive experience in the use of a large variety of tools and design paradigms when it came to creating project. The mixture of such different tools like the ROS environment or in the case of hardware, the use of 3D printed pieces and the printer was a very notable novelty. Other possible benefits that this project con give, not related to the project experience, is the following.

The exploration of the Myo Armband as a possible project piece has been explored and its viability of it interfacing with other robots has been explored, at the very least in an introductory capacity. This means that more projects could be introduced into the end of degree ideas relating to the world of Machine-Person Interaction.

The use of the InMoov as an end of degree project has been proved before, but it’s function within the ROS environment was not documented or was not available to work with. Using the InMoov within the ROS environment is a great step forward towards its use as an education tool and a great testbed for different projects relating to it.

Finally, the benefits it offers for the medical community pertaining to the use as cheap, easy to repair prosthesis for those in need of them. Pairing said prosthesis with a Myo armband would also allow for a wide variety of possible solutions to mobility problems. For example, in the case of the loss of the forearm downwards the program could be adapted to mirror the movements of the opposite remaining arm.

5.1.2 Limitations

This section will describe in what ways the scope of the project has limited the project’s capacities and how they could be remedied. Most of these limitations relate to the actual hardware that the system was working on. The InMoov hand and forearm offers a great base to begin to build a complex robot, but it has some very notable limits when it comes to functionality. The first problem is that it’s just the hand and forearm without including the elbow, limiting the possible movements to just this area.

Without a way to move the entire forearm its capacity to interact with the surrounding world is limited to where it can move it physically and even then its grip is fairly loose due to the lack of any form of grip pads. To both of these issues the InMoov website offers specific solutions. To help with its interaction problem it can either attach the hand to a stand of some kind or build the rest of the arm, at least up to the shoulder. Having studied the instructions of these section its complexity is not that much greater than the hand and forearm's. Albeit the pieces are notably larger and their printing must be made carefully so as to avoid imperfections that may cause structural deficiencies, since most of the weight will be held by this structure. To the problem of the grip the option is to gain access to a high friction grip pads, cut them into appropriately sized pieces and attach them to the palm side of the hand and fingers.

Other limitations come from the actual Myo Armband. The Myo Armband was designed by Thalmic Labs to be capable of processing the EMG signals from the arm and translate them into gestures directly from within the armband. Unfortunately, the methods by which this is done has not been released to the project, but it does allow the option for accessing the raw EMG data of the eight sensor with an additional energy cost. With these limitations I've had to cut the project back to only use the existing gestures that are recognized by the armband, but given more time and a more profound understanding of EMG patterns it could be possible to create a more comprehensive database of recognizable gestures or directly turning the EMG into finger movements, but in the latter case its viability for use with more users might be more complex or impossible. This is due to the fact that the Myo Armband learns to recognize gestures by using a wide variety of different user data-sources, thus allowing it to process data for a large variety of arm types, mainly by having a user-data report opt-in policy for the armband. If a new gesture was wanted the same would have to be done for said gesture, with a large amount of tests and permutations of the same movement.

A final limitation the hand had is that the test while useful for a preliminary report on the effectiveness of the system, does have some problems in some areas. For one the bias on the recognition of the double tap gesture: Consistently the gesture failed to be registered correctly on the system, except for one case. This user reportedly had previous experience with the Myo Armband and found it much simpler to make said gesture. As such future tests could be implemented with a preliminary training of the user for the Myo, a basic rundown of the gestures and how to successfully make them. Also a larger pool of users could remedy some of the deviation and error found on the testing.

5.1.3 Possible Future Projects

Working from the above limitations and benefits, possible future projects stemming from this can be devised. The most obvious one is expanding on the robot's build, by adding shoulders and maybe the other arm. With these new structures more developed projects could be arranged for example using two Myo Armbands simultaneously along with the included accelerometers could allow for a limited movement of the arms.

Other possible projects that could be explored with a basis on this one could be ones involving the use of the Myo Armband. They are a rather interesting and easy to use medical tool with a wide range of possibilities, also they are relatively cheap to purchase compared to other similar sensors.

5.2 InMoov Construction

The construction of the robotic arm went rather well but there were some rather notable problems in time management. The time it took for the pieces to be printed did slow down the progress done in the building, as well as the subsequent sanding and preparation of them before the build, but for the most part there was little trouble during the process.

The main issues that were found were due to the nature of several joints and tubes that formed important moving parts. These multiple problems due to the fact that the plastic used expands afterwards and thus fitting both of these types of pieces together was difficult to do without breaking them. In the end a solution was found by using a handheld drill to make the holes larger. Other issues arose but they had more to do with the quality of the screws used in fixating the pieces together and were promptly substituted.

The final problem arose in the cables used to move the fingers. It was a very finicky process of carefully threading the fishing line through multiple holes and affixing it to the ends of the fingers via knots. After several tries the best method to do this was found to be threading the fishing line from the fingers towards the servos, instead of the other way around.

5.3 Myo Armband & Software

Experimentation with the Myo armband was rather simple due to the ease of installation of the base software, making it interface with the computer was decidedly a more complex endeavour. ROS does not have an official release of a EMG sensing package, only a user created interface specific for the armband, which is what was used in this project. Before finding it, however, various experiments were realized with various ideas using visual studio or other tools. In the end the package proved to be most useful. Other issues that arose are the transformation of the roll via the new package, where consulting some basic trigonometry formulas was needed to achieve the transformation of the roll into degrees.

5.4 Final Product & Testing

The final complete product resulted in a person to machine interface capable of recognizing several types of hand movements and imitate them to a certain degree with the connected robot. This was thoroughly tested with the five recognizable gestures of the Myo Armband, problems did arise when it tried to move its wrist at the same time, the problem is believed to be caused by low battery and difference in latency between processing the Gestures and the Wrist angle movements. As a whole it was found that the final product reaches the objectives it set out to do and that there is still room for improvement, but it’s capacity to recognize gestures and reproduce said gestures on the robot arm has been proved.

To conclude it is good to revisit the original objective of the project as relayed at the start:

- To create a robotic arm and associated software project capable of reading a sensor and copy the movements of a hand in an acceptable manner.

This is the generalised objective to encompass the entire project and it has been completed to the fullest extent of the project. The robot arm has been created, the software project has been proven to sense and translate the gestures of the hand into the robotic arm. By testing the robot arm with various types of users the results prove that it has been able to see that it’s capable of sensing gestures fairly reliably, except for one gesture, the double tap gesture, which is difficult to do even for the base Myo program.

Bibliography

[1] Mataríc, Maja J. (2007). *The Robotics Primer*. Cambridge, Massachusetts; The MIT Press

[2] Gael Langevin. (2016). InMoov | open-source 3D printed life-size robot. Accessed on 12/June/2016, http://inmoov.fr/

[3] R.G.E. Clement, K.E. Bugler, C.W. Oliver. (2011). Bionic prosthetic hands: A review of present technology and future aspirations. The Surgeon, 9, 336-340.

[4]Todd A. Kuiken, Guanglin Li, Blair A. Lock, Robert D. Lipschutz, Laura A. Miller, Kathy A. Stubblefield, Kevin B. Englehart. (2009). Targeted Muscle Reinnervation for Real-time Myoelectric Control of Multifunction Artificial Arms. JAMA, February 11, 2009—Vol 301, No. 6.

[5] Gael Langevin. (2012-2016). InMoov. Accessed on 12/June/2016, http://inmoov.blogspot.com

[6] Shünke, Michael. Schulte, F.J. (2011). *PROMETHEUS. TEXTO Y ATLAS DE ANATOMIA. TOMO 1: ANATOMIA GENERAL Y APARATO LOCOMOTOR (2ª ED.)*. (pg. 332-347) España, Madrid: Editorial Medica panamerican.

[7] Netter, Frank H. (2015). *Atlas de anatomía humana [Recurs electrònic] / Frank H. Netter.* (pg. 430) Barcelona: Elsevier

[8] Netter, Frank H. (2015). *Atlas de anatomía humana [Recurs electrònic] / Frank H. Netter.* (pg. 432) Barcelona: Elsevier

[9] Netter, Frank H. (2015). *Atlas de anatomía humana [Recurs electrònic] / Frank H. Netter.* (pg. 459) Barcelona: Elsevier

[10] Todd A. Kuiken, Guanglin Li, Blair A. Lock, Robert D. Lipschutz, Laura A. Miller, Kathy A. Stubblefield, Kevin B. Englehart. (2009). Targeted Muscle Reinnervation for Real-time Myoelectric Control of Multifunction Artificial Arms. JAMA, February 11, 2009—Vol 301, No. 6.

[11] Md. Rezwanul Ahsan, Muhammad Ibrahimy, Othman O. Khalifa. (2009). EMG Signal Classification for Human Computer-Interaction A Review. European Journal of Scientific Research, January, 2009.

[12] Sebastian Bitzer. Patrick van der Smagt (2006). Learning EMG control of a robotic hand: Towards Active Prostheses. Proceedings of the 2006 IEEE International Conference on Robotics and Automation, May, 2006. Orlando, Florida.

[13] George N. Saridis, Thomas P. Gootee (1982). EMG Pattern Analysis and classification for a Prosthetic Arm. IEEE Transactions on Biomedical Engineering, Vol, BME-29, no. 6, June 1982

[14] Gael Langevin. (2013). Hand and Forearm. Accessed on 12/June/2016, http://inmoov.fr/hand-and-forarm/

[15] Thalmic Labs Inc. (2013-2016). Myo™. Accessed on 12/June/2016, https://www.myo.com/

[16] Thalmic Labs Inc. (2014-2016). Myo™Developer. Accessed on 12/June/2016, https://developer.thalmic.com/

[17] Gael Langevin. (2013). Sketch servo zero, rest, max, 90. Accessed on 12/June/2016, http://inmoov.fr/sketch-servo-zero-rest-max/

[18] Gael Langevin. (2012-2016). InMoov. Accessed on 05/July/2016, http://inmoov.blogspot.com.es/2015/02/january-with-lots-of-progress.html

[19] Gael Langevin. (2015). InMoov 2 Hand + Myo Thalmic first control 3D printed prosthesis. Accessed on 05/July/2016, Published 10/February/2015 https://www.youtube.com/watch?v=WNhZKFth9EM